WORKSHOP
ON
STATE−OF−THE ART COMPUTATIONAL METHODS
AND
SOFTWARE
FOR CONTROL SYSTEMS

Presenter: Biswa Nath Datta

Department of Mathematical Sciences
Northern Illinois University
DeKalb, IL. 60115 USA

E−mail: dattab@math.niu.edu

URL: www.math.niu.edu/~dattab

A Reference Book

My Lectures are based on the book

"Numerical Methods for Linear Control Systems, Design and Analysis"

by

B.N. Datta
Elsevier Academic Press, 2003.

Information on the book is available at the website:

`http://www.math.niu.edu/~dattab`

Softwares Associated with the Book

- MATCONTROL (A MATLAB-based software implementing most algorithms in the book).

  URL: `www.math.niu.edu/ ∼ dattab`

- **Control Systems Professional - Advanced Numerical Methods** (MATHEMATICA - based Numerical Control Software).

  URL: `www.math.niu.edu/ ∼ dattab`

Computational Strategy for Control Problems

A Strategy for Solving Computational Control Problems

Step 1. Reduce the problem to a more manageable one by transforming the system matrices (A, B, C) to some "**Condensed forms**".

Step 2. Solve the reduced problem.

Step 3. Recover the solution of the original problem from the solution of the reduced problem.

Condensed Forms

Widely used condensed forms in control theory text books are:

- Companion Forms
- Jordon Canonical Forms

Unfortunately, none of these forms can be achieved in a numerically effective way.

- The transforming matrices can be highly unstable.

Transformation of a Matrix A to a Companion Form

Stage I. Transform A to an upper Hessenberg matrix H.

$$A \xrightarrow{P} P^T AP = H \ (P - \ \textbf{Orthogonal})$$

- **Numerically Stable.**

Householder's or Givens Method can be used.

Stage II. Reduce H further to a companion matrix C.

$$H \xrightarrow{X} X^{-1}HX = C \ (X - \ \textbf{Nonorthogonal})$$

- **Unstable.**

The transforming matrix is **highly ill-conditioned**, if H has small subdiagonal entries.

- **Ill-Conditioning**: X is **ill-conditioned** if

$$\text{Cond } (X) = ||X^{-1}||\ ||X||$$

  is too **large**.

- *Orthogonal matrices are well-conditioned*

$$(\text{Condition Number} = 1)$$

Suggestion: *Avoid these forms in numerical computations*

The condensed forms of choice are

- Hessenberg Forms
- Real Schur Forms
- Hessenberg-triangular Forms
- Controller and Observer Hessenberg Form.

These forms can be achieved using orthogonal transformations which are very well-conditioned.

- Hessenberg Forms

$$\begin{pmatrix} * & * & & 0 \\ \vdots & & \ddots & \\ * & \vdots & & * \\ * & * & \cdots & * \end{pmatrix} \qquad \begin{pmatrix} * & \cdots & * & * \\ * & \cdots & * & * \\ & \ddots & \vdots & \vdots \\ 0 & & * & * \end{pmatrix}$$

Lower Hessenberg **Upper Hessenberg**

- *Companion Matrices* - special Hessenberg matrices

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \times & \times & \times & \times \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 0 & \times \\ 1 & 0 & 0 & \times \\ 0 & 1 & 0 & \times \\ 0 & 0 & 1 & \times \end{pmatrix}$$

Lower Companion **Upper Companion**

• **Real Schur Form**: A Quasi-triangular matrix with either 1×1 or 2×2 block matrices on the diagonal.

Example

$$H = \begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \boxed{\times} & \times & \times & \times \\ 0 & \boxed{\times} & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{pmatrix}.$$

- Real Schur Form by QR Iteration

  Two-stage procedure:

Stage I: $A \xrightarrow[\text{Direct}]{\text{Householder}} H$

Stage II: $H \xrightarrow[\text{Iterative}]{QR \text{ Iteration}} \text{Real Schur Form}$

$$H = \begin{bmatrix} 0.2190 & -0.0756 & 0.6787 & -0.6391 \\ -0.9615 & 0.9032 & -0.4571 & 0.8804 \\ 0 & -0.3822 & 0.4526 & -0.0641 \\ 0 & 0 & -0.1069 & -0.0252 \end{bmatrix}.$$

Iteration	h_{21}	h_{32}	h_{43}
1	0.3860	-0.5084	-0.0084
2	-0.0672	-0.3773	0.0001
3	0.0089	-0.3673	0
4	-0.0011	-0.3590	0
5	0.0001	-0.3905	0
...			

The computed RSF is

$$H = \begin{bmatrix} 1.4095 & 0.7632 & \text{-}0.1996 & 0.8394 \\ 0.0001 & \boxed{\begin{matrix} 0.1922 & 0.5792 \\ \text{-}0.3905 & 0.0243 \end{matrix}} & & 0.0494 \\ 0 & & & \text{-}0.4089 \\ 0 & 0 & 0 & \text{-}0.0763 \end{bmatrix}.$$

The eigenvalues of $\begin{bmatrix} 0.1922 & 0.5792 \\ -0.3905 & 0.0243 \end{bmatrix}$ are $0.1082 \pm 0.4681j$.

MATLAB Commands

- Hessenberg: $[P, H] = \mathbf{hess}(A)$
  $$P^T A P = H$$
- **Real Schur Form**: $[U, T] = \mathbf{Schur}\ (A)$
  $$U^T A U = T.$$
- **Efficiency and Numerical Stability**

  Transformations to Hessenberg

Two Important Properties of Matrix Algorithms

- **Efficiency** – Measured by flop-count
  flop – floating point operation $(+, -, *, \div)$.

  Computations involving $n \times n$ matrices are efficient if it does not require more than $O(n^3)$ flops.

- **Numerical Stability**– *If the computed solution is the exact solution of a nearby problem.*

  Example The QR iteration algorithm for finding the RSF is numerically stable:

  $$Q^T(A + E)Q = \hat{T} \text{ (Computed RSF)}$$

  where

  $$||E||_F \leq c\mu||A||_F \text{ (**small**)}$$

MODELLING
AND
SYSTEM RESPONSES
(Chapter 5)

State-Space Representations of Control Systems

Linear time-invariant continuous-time

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t),\ x(t_0) = x_0, \\ y(t) &= Cx(t) + Du(t). \end{aligned}$$

- $x(t)$ is the n-dimensional **state vector**

- $u(t)$ is the m-dimensional **input** (control) **vector** ($m \leq n$).

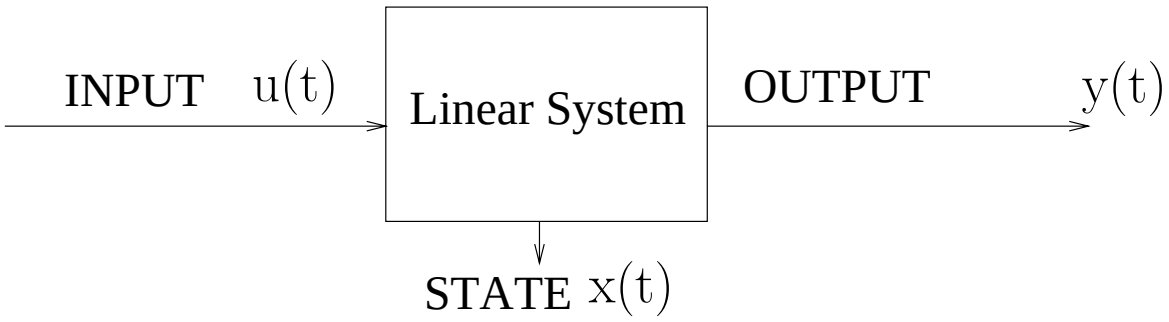
- $y(t)$ is the r-dimensional **output vector**.

The matrices A,B, C, and D are **time-invariant matrices**, respectively, of dimensions $n \times n, n \times m, r \times n$, and $r \times m$.

Discrete-time System

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k + Du_k$$

- These lectures will be confined mostly to continuous-time systems only.
- The discrete-time systems will be discussed only occassionally.



Representation of a Continuous-time State-Space Model.

Solutions of a Continuous-Time System: System Responses

Theorem. (Continuous-Time State-space Solution)

- $x(t) = e^{A(t-t_0)}x_0 + \int_{t_0}^{t} e^{A(t-s)}Bu(s)ds$

- $y(t) = Ce^{A(t-t_0)}x_0 + \int_{t_0}^{t} Ce^{A(t-s)}Bu(s)ds + Du(t).$

Remarks: (i) If $u(t) = 0$, then

$$x(t) = e^{A(t-t_1)}x(t_1)$$

for every $t \geq t_0$ and any $t_1 \geq t_0$.

Computational Methods for Computing the Exponential Matrix

- **The Eigenvalue-Eigenvector Method**
- **Padé Methods**
- **ODE Methods**
- **Matrix Decomposition Methods**

The Eigenvalue-Eigenvector Method

A difficulty with this approach arises when A has some nearly equal eigenvalues. This can be seen from the following theorem (Moler and Van Loan (1978)).

Theorem. Let $X^{-1}AX = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$, where $\lambda_1, \lambda_2, \cdots, \lambda_n$ are the eigenvalues of A. Then

$$\|fl(e^{At}) - e^{At}\|_2 \leq n\mu e^{\rho(A)t}\text{Cond}_2(X),$$

where $\rho(A) = \max|\lambda_i|$ is the spectral radius of A.

- If eigenvectors of A are almost lineary independent, then e^{At} can not be computed accurately.

The Padé Approximation Method

The (p, q) **Padé approximation** to e^A :

$$R_{pq}(A) = [D_{pq}(A)]^{-1} N_{pq}(A),$$

where

$$D_{pq}(A) = \sum_{j=0}^{q} \frac{(p+q-j)!q!}{(p+q)!j!(q-j)!}(-A)^j$$

and

$$N_{pq}(A) = \sum_{j=0}^{p} \frac{(p+q-j)!p!}{(p+q)!j!(p-j)!}A^j.$$

- **Round-off errors due to catastrophic cancellation is a major concern for this method.**
- It is less when $||A||$ is not too large and the diagonal approximants $(p = q)$ are used.

Padé Approximation to e^A using Scaling and Squaring (Algorithm 5.3.1)

Input: $A \in \mathbb{R}^{n\times n}$, $\delta > 0$, an error-tolerance.

Output: $F = e^{A+E}$ with $\|E\|_\infty \leq \delta \|A\|_\infty$.

Step 1. Choose j such that $\|A\|_\infty \leq 2^{j-1}$. Set $A \equiv A/2^j$.

Step 2. Find p such that p is the smallest non-negative integer satisfying

$$\left(\frac{8}{2^{2p}}\right)\frac{(p!)^2}{(2p)!(2p+1)!} \leq \delta.$$

Step 3. Set $D \equiv I, N \equiv I, Y \equiv I, c = 1$.

Step 4. For $k = 1, 2, \cdots, p$ do

$$c \equiv c(p-k+1)/[(2p-k+1)k]$$

$$Y \equiv AY, N \equiv N + cY, D = D + (-1)^k cY.$$

End

Step 5. Solve for $F : DF = N$.

Step 6. For $k = 1, 2, \cdots j$ do

$F \equiv F^2$.

End

Flop-count · The algorithm requires about $2(p+j+\frac{1}{3})n^3$ flops.

Numerical Stability Property.

- $\boldsymbol{e^A}$ **may grow before it decays during the squaring process** known as **"hump"** phenomenon.
- **MATLAB Note:** The MATLAB function **expm** computes the exponential of a matrix A.
- **MATCONTROL Note:** Algorithm **5.3.1** has been implemented in MATCONTROL function: **expmpade**.

Computing e^A via the Real Schur Form.

- $P^T A P = R,$ a real Schur form
- $e^A = P e^R P^T.$

Real Schur Form

$$R = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1k} \\ 0 & R_{22} & \cdots & R_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{kk} \end{pmatrix}$$

- Each R_{ii} is either a scalar or a 2×2 matrix
- The QR iteration algorithm is used to compute P and R.
- **MATLAB Command**: $[U, T] =$ **schur (A)**

The Schur Algorithm for e^A (Algorithm 5.3.2).

Input: $A \in \mathbb{R}^{n\times n}$

Output: e^A.

Step 1. Transform A to R an **upper triangular** matrix using the QR iteration algorithm:

$$P^T AP = R.$$

(Note that when the eigenvalues of A are all real, the real Schur form is upper triangular).

Step 2. Compute $e^R = G = (g_{ij})$:

For $i = 1, \cdots, n$ do

$g_{ii} = e^{r_{ii}}$

End

For $k = 1, 2 \cdots, n-1$ do

For $i = 1, 2, \cdots, n-k$ do

Set $j = i + k$

$$g_{ij} = \frac{1}{(r_{ii} - r_{jj})} \left[r_{ij}(g_{ii} - g_{jj}) + \sum_{p=i+1}^{j-1} (g_{ip} r_{pj} - r_{ip} g_{pj}) \right].$$

End

End

Step 3. Compute $e^A = Pe^R P^T$

Flop-count. Computation of e^R in Step 2 requires about $\frac{2n^3}{3}$ flops.

MATCONTROL Note: The Algorithm has been implemented in MATCONTROL function **expmschr**.

Comparison of Different Methods for Computing the Exponential Matrix

- The **Padé approximation method** (with scaling and squaring) and the **Schur method** should, in general, be attractive from computational view points.
- Avoid Taylor Series methods and companion or Jordan Canonical methods.
- Use ODE Method when A is large and sparse

Steady-State Response in the Frequency Domain

$$y(t) = C(j\omega I - A)^{-1}Bve^{j\omega t} + Dve^{j\omega t}.$$

Definition. Frequency Response Matrix:

$$G(j\omega) = C(j\omega I - A)^{-1}B + D$$

Computing the Frequency Response Matrix

Assume $D = 0$.

The computation of $(j\omega I - A)^{-1}B$ is equivalent to solving m systems:

$$(j\omega I - A)X = B.$$

A usual scheme for computing the frequency response matrix is:

Step 1. Solve the m systems for m columns $x_1, x_2, \ldots, x_m$ of X:

$$(j\omega I - A)x_i = b_i, \qquad i = 1, 2, \ldots, m$$

where b_i is the i-th column of B.

Step 2. Compute CX.

Remark: **Too Expensive** - for each ω, Approximately $2n^3 + 2mn^2 + 2mnr$ flops.

A Hessenberg Method

- $PAP^T = H$, Upper Hessenberg
- $G(j\omega) = C(j\omega I - A)^{-1}B$
$= C(j\omega I - PHP^T)^{-1}B$
$= C(P(j\omega I - H)P^T)^{-1}B$
$= CP(j\omega I - H)^{-1}P^T B.$

$$\begin{pmatrix} * & \cdots & * & * \\ * & \cdots & * & * \\ & \ddots & \vdots & \vdots \\ 0 & & * & * \end{pmatrix}$$

- Householder or Givens Methos via orthogonal similarity.

$$MATLABCommand: \ [P, H] = \ \mathbf{hess}\ (A).$$

A Hessenberg Algorithm for the Frequency Response Matrix (Algorithm 5.5.1)

Input. A–**The** $n \times n$ **state matrix**
ω–Frequency, a **real number**
B–The $n \times m$ **input matrix**
C–**The** $r \times n$ **output matrix.**

Output. The Frequency Response Matrix

$$G(j\omega) = C(j\omega I - A)^{-1}B.$$

Step 1. Transform A to an upper Hessenberg matrix H:
$P^T AP = H$.

Step 2. Compute $B' = P^T B$
and $C' = CP$, using the factored form of P.

Step 3. Solve the m **Hessenberg** systems:

$$(j\omega I - H)x_i = b'_i, \qquad i = 1, \ldots, m,$$

where b'_i is the i-th column of B'.

Step 4. Compute $C'X$. ■

• Hessenberg systems require $O(n^2)$ flops to solve using Gaussian elimation with partial pivoting.

Comparison of the Efficciency

For N values of ω

- Hessenberg Method:

  $\frac{10}{3}n^3+4(n-2)(m+r)n$ real $+[2mn^2+2rnm]N$ complex flops

- Non-Hessenberg Method:

  $[2n^3+2mn^2++2mnr]N$ complex flops

Numerical Stability: If the data is well-conditioned, then the frequency response of the computed Hessenberg form is $(C + \triangle C)(j\omega I - A - \triangle A)^{-1}(B + \triangle B)$, where $\triangle A, \triangle B$, and $\triangle C$ are small. Thus, the **Hessenberg-method is stable.**

MATCONTROL Note: Algorithm 5.5.1 has been implemented in **freqresch**.

CONTROLLABILITY, OBSERVABILITY, and DISTANCE TO CONTROLLABILITY (Chapter 6)

Theoretical Criteria of Controllability

Let $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}(m \leq n)$.

- The $n \times nm$ matrix
$$C_M = (B, AB, A^2B, \cdots, A^{n-1}B)$$
has full rank n
- The matrix **(Controllability Grammian)**
$$W_C = \int_0^{t_1} e^{At} B B^T e^{A^T t} dt$$
is nonsingular for any $t_1 > 0$.
- If (λ, x) is an eigenpair of A^T, i.e., $x^T A = \lambda x^T$, then $x^T B \neq 0$. **(Eigenvector Criterion)**
- Rank $(A - \lambda I, B) = n$ for every eigenvalue λ of A. **(Eigenvalue Criterion)**
- The eigenvalues of $A - BK$ can be arbitrarily assigned (assuming that the complex eigenvalues occur in conjugate pairs) by a suitable choice of K. **(Pole-placement Criterion)**

Numerical Stability

- Computational algorithms based on most theoretical criteria are numerically unstable.

- **SVD**: $A_{m\times n} = U\Sigma V^T (m \geq n)$

$$U_{m\times m} - - - \text{ orthogonal}$$
$$V_{n\times n} - - - \text{ orthogonal}$$

$$\Sigma_{m\times n} = \text{ diag } (\sigma_1, \ldots, \sigma_n)$$

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0 \textbf{ (singular values).}$$

- $\sigma_1 = \sigma_{\max}$ = Largest singular value
- $\sigma_n = \sigma_{\min}$ = Smallest singular value
- **MATLAB Program:** $[U, S, V] =$ **svd** (A)
- Rank = Number of nonzero singular values
- Numerical Rank = Number of nonzero sigular values above a threshold.

Example.

$$A = \begin{pmatrix} 1 & & & \\ & 2^{-1} & & \\ & & \ddots & \\ & & & 2^{-9} \end{pmatrix}_{10\times 10}, B = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \dots \\ 1 \end{pmatrix}$$

- The pair (A, B) is controllable.
- The controllability matrix

  $$C_{AB} = (B, AB, \dots, A^9 B)$$

  has **three small singular values**

  $$0.613 \times 10^{-12}, 0.364 \times 10^{-9}, 0.712 \times 10^{-7}$$

- Numerical rank is less than 10.
- **Conclusion:** In floating point arithmetic the pair (A, B) is **not** controllable.

A Numerically Effective Test

- $PAP^T = H$, A Block Upper Hessenberg Matrix
- $\bar{B} \equiv PB = \begin{pmatrix} B_1 \\ \\ 0 \end{pmatrix}$.

$$H \equiv \begin{pmatrix} H_{11} & H_{12} & H_{13} & \cdots & H_{1k} \\ H_{21} & H_{22} & H_{23} & \cdots & H_{2k} \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & H_{k,k-1} & H_{kk} \end{pmatrix}, \quad \bar{B} \equiv \begin{pmatrix} B_1 \\ 0 \end{pmatrix},$$

The pair $(H, \overline{B})$ is called *Controller-Hessenberg* pair.

Test of Controllability

The pair (A, B) is controllable if $H_{k,k-1}$ has full rank.
It is uncontrollable if $H_{k,k-1} = 0$.
(Staircase Algorithm: Algorithm 6.2.1)

MATCONTROL Function: cntrlhs

Example. (An uncontrollable pair).

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \; B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

- The **Controller-Hessenberg Pair**:

$$H = \begin{pmatrix} 2.3333 & -0.4714 & 0 \\ 0.9428 & 0.6667 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\bar{B} = \begin{pmatrix} -1.7321 & -1.7321 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Clearly the pair (A, B) is not controllable.

Controllability Test in the Single-Input Case

$$PAP^T = H = \begin{pmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & \cdots & h_{2n} \\ 0 & h_{32} & \ddots & \cdots & h_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{n,n-1} & h_{nn} \end{pmatrix},$$

$$Pb = \bar{b} = \begin{pmatrix} b_1 \\ 0 \\ \\ \vdots \\ 0 \end{pmatrix}.$$

- (A, B) Controllable $\Longleftrightarrow$ H is **unreduced** and $b_1 \neq 0$.
- Unreduced $\equiv$ Subdiagonal entries are different from zero.

A Numerically Effective Test for Observability (Section 6.8)

- **Reduction to Observer-Hessenberg Pair:**

$$H = PAP^T = \begin{pmatrix} H_{11} & H_{12} & \cdots & H_{1k} \\ H_{21} & \ddots & & \vdots \\ & \ddots & \ddots & \vdots \\ 0 & & H_{k,k-1} & H_{kk} \end{pmatrix},$$

$$\bar{C} = CP^T = (0, C_1).$$

- **The pair (A, C) is observable if H is block unreduced (that is, all the subdiagonal blocks have full rank) and the matrix C_1 has full rank.**

Flop-Count. The observer-Hessenberg form requires roughly $6n^3 + 2n^2r$ flops.

MATCONTROL Note: MATCONTROL function **obserhs** can be used to obtain the reduction to observer-Hessenberg form.

Distance to Uncontrollability

- **An Obviously Controllable Pair**

$$A = \begin{pmatrix} -1 & -1 & \cdot & \cdot & \cdot & \cdot & -1 & -1 \\ 1 & \cdot & & & & & & -1 \\ & 1 & \cdot & & & & & -1 \\ & & \cdot & \cdot & & & & \cdot \\ & & & & & & & \\ & & & \cdot & \cdot & & & \cdot \\ & & & & \cdot & \cdot & & \cdot \\ & 0 & & & & \cdot & \cdot & -1 \\ & & & & & & 1 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{pmatrix}.$$

- Add $(-2^{1-n}, -2^{1-n}, \ldots, -2^{1-n})$ to the last row of (B, A).

The result is an uncontrollable pair.

- **Conclusion:** The controllable pair (A, B) is close to an uncontrollable pair.

A Measure of the Distance to Uncontrollability

Definiton of the distance to uncontrollability, $\mu(A, B)$:

- $\mu(A, B) \equiv$ min $\{\|\Delta A, \Delta B\|_2$ such that the system defined by $(A + \Delta A, B + \Delta B)$ is uncontrollable$\}$.

- Smallness of $\mu(A, B) \rightarrow$ closeness to uncontrollabilty

Perturbations are assumed to be over the field of complex numbers.

Distance to Uncontrollability in terms of Singular Values.

σ_n = the smallest singular value

$\mu_(A, B) = \min_{s \in \mathbb{C}} \sigma_n(sI - A, B)$

- If $\mu(A, B)$ is small, then the original pair (A, B) is close to an uncontrollable pair.
- Two Algorithms: **Newton's Method** and the **Wicks De Carlo Method**

The Wicks-DeCarlo Method for Distance to Uncontrollability

Minimizing the above function is equivalent to minimizing

$$\mu(A,B) = \min_{u \in \mathbb{C}^n} ||(u^* A(I - uu^*)u^* B||,$$

subject to $u^* u = 1$.

Definition. Distance measure $d_1(A,B)$

$$[d_1(A,B)]^2 = |||[e_n^*(A(I - e_n e_n^*)\ B)]||_2^2$$

$$= \sum_{j=1}^{n-1} |a_{nj}|^2 + \sum_{j=1}^{m} |b_{nj}|^2.$$

Then

$$\boxed{\mu(A,B) = \min_{\substack{U \in \mathcal{C}^{n \times n} \\ U^* U = I}} d_1(U^* AU, U^* B)}$$

Idea of an Algorithm

Construct a set of matrices $\{A_k, B_k\}$ from (A, B) such that

- $A_{k+1} = U_k^* A_k U_k$
- $B_{k+1} = U_k^* B$

$$d_1(A_{k+1}, B_{k+1}) < d_1(A_k, B_k)$$

Then, $\lim_{k\to\infty} d_1(A_k, B_k)$ is a real minimum of $\mu(A, B)$.

Algorithm. An Algorithm for Computing $\mu_C(A, B)$

Inputs: The matrices $A \in R^{n\times n},\ B \in R^{n\times m}$

Output: $\mu(A, B)$.

Step 0. Ste $A_1 \equiv A,\ B_1 \equiv B$.

Step 1. For $k = 1, 2, ...$ until convergence.

Step 1.1. Form $M_k = (A_k - (a_{nn})_k I \ \ B_k)$.

Step 1.2. Factor $M_k = L_k V_k$,

Step 1.3. Find the QR factorization of

$$L_k = U_k^* R_k.$$

Step 1.4. Set $A_{k+1} = U_k^* A_k U_k,\ \ B_{k+1} = U_k^* B_k$.

Step 1.5. If $d_1(A_{k+1}, B_{k+1}) = d_1(A_k, B_k)$, stop.

End.

Example.

$$A = \begin{bmatrix} 0.950 & 0.891 & 0.821 & 0.922 \\ 0.231 & 0.762 & 0.445 & 0.738 \\ 0.607 & 0.456 & 0.615 & 0.176 \\ 0.486 & 0.019 & 0.792 & 0.406 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.9350 & 0.0580 & 0.1390 \\ 0.9170 & 0.3530 & 0.2030 \\ 0.4100 & 0.8130 & 0.1990 \\ 0.8940 & 0.0100 & 0.6040 \end{bmatrix}$$

- **Tol** $= 0.00001$.

Define $\mu_k = d_1(A_k, B_k)$.

The algorithm produces the following converging sequence of μ_k:

k	μ_k	k	μ_k
0	1.42406916966838	10	0.41450782001833
1	0.80536738314449	11	0.41450781529413
2	0.74734006994998	12	0.41450781480559
3	0.52693889988172	13	0.41450781475487
4	0.42241562062172	14	0.41450781474959
5	0.41511102322896	15	0.41450781474904
6	0.41456112538077	16	0.41450781474899
7	0.41451290008455	17	0.41450781474898
8	0.41450831981698	18	0.41450781474898
9	0.41450786602577	19	0.41450781474898

and after 19 iteration the algorithm returns

$\boxed{\mu = 0.41450781474898.}$

MATCONTROL Implemention: Function **discntrl**.

Stability, Robust Stability and Distance to Instability (Chapter 7)

Stability and Inertia

- The continuous-time linear system:

$$\dot{x}(t) = Ax(t)$$

  is **asymptotically stable** if and only if all the eigenvalues of A have negative real parts.

- Lyapunov approach

$$XA + A^T X = -I$$

  is **unpractical**.

(The widely-used Schur Method for Lyapunov equations is based on finding the real-Schur form of A. The real schur form displays the eigenvalues anyway).

- Routh-Hurwitz criterian requires computing the characteristic polynomial: **Not numerically stable.**

- From numerical view point, the best approach is to **Compute all the eigenvalues explicitly** using the standard QR iteration algorithm.

  MATLAB Function: **eig** (A) computes all the eigenvalues.

An Indirect Matrix-Equation Approach

(Carlson and Datta (1979)).

- Does not solve any Lyapunov matrix equation or compute the eigenvalues **explicitly**.
- Finds a nonsingular symmetric matrix X such that

$$XA + A^*X = C \geq 0.$$

  In $(A) =$ In (X).

  X is negative definite if and only if A is **Stable.**

- Three times faster than explicitly computing eigenvalues.
- Numerical stability of the method not established yet.

Distance to an Unstable System

Let A be an $n \times n$ **complex stable matrix.**

Question: How "nearly unstable" is the stable matrix A?

Definition. *Let $A \in \mathbb{C}^{n\times n}$ have no eigenvalue on the imaginary axis.*

Distance to Instability:

$$\beta(A) = \min\{\| E \| \, | A + E \in U\}.$$

• $U =$ Set of matrices with atleast one imaginary eigenvalue.

An Example of a Nearly Unstable Matrix

$$A = \begin{pmatrix} -0.5 & 1 & 1 & 1 & 1 & 1 \\ 0 & -0.5 & 1 & 1 & 1 & 1 \\ 0 & 0 & -0.5 & 1 & 1 & 1 \\ 0 & 0 & 0 & -0.5 & 1 & 1 \\ 0 & 0 & 0 & 0 & -0.5 & 1 \\ 0 & 0 & 0 & 0 & 0 & -0.5 \end{pmatrix}$$

- A is perfectly stable.
- Change (6, 1)th entry from zero to $\frac{1}{324}$ and compute the eigenvalues agian of the perturbed matrix.
- The eigenvalues of the perturbed matrix are: $-0.8006, -0.7222 \pm 0.2485j, -0.3775 \pm 0.41201, 0.$
- **Conclusion: A is very close to an unstable matrix.**

A Characterization of the Distance to Instability

Let $\sigma_{\min}(A - j\omega I)$ be the smallest singular value of $A - j\omega I$. Then

- $\beta(A) = \min_{\omega \in \mathcal{R}} \sigma_{\min}(A - j\omega I)$.

A Bisection Algorithm to Measure the Distance to Instability

Define $2n \times 2n$ **Hamiltonian matrix** $H(\sigma)$, given $\sigma \geq 0$, by

$$H(\sigma) = \begin{pmatrix} A & -\sigma I \\ \sigma I & -A^* \end{pmatrix}.$$

Theorem: *$\sigma \geq \beta(A)$ if and only if $H(\sigma)$ has a purely imaginary eigenvalue.*

Algorithm The Bisection Algorithm for the Distance to an Unstable System. (Algorithm 7.6.1)

Inputs: *A - An $n \times n$ stable complex matrix*

τ - Tolerance (> 0).

Outputs: *Real numbers α and ν such that* either $\nu/10 \leq \alpha \leq \beta(A) \leq \nu$ or $0 = \alpha \leq \beta(A) \leq \nu \leq 10\tau$.

Step 1. *Set* $\alpha \equiv 0$, $\nu = \frac{1}{2}\|(A + A^*)\|_2$

Step 2. *Do while* $\nu > 10 \max(\tau, \alpha)$

$\sigma \equiv \sqrt{\nu \max(\tau, \alpha)}$

If $H(\sigma)$ has a purely imaginary

eigenvalue,

then set $\nu \equiv \sigma$; else $\alpha \equiv \sigma$

Example *Find $\beta(A)$ for the matrix*

$$A = \begin{pmatrix} -1 & 1 \\ 0 & -0.0001 \end{pmatrix}.$$

$$\tau = 0.0100$$

Iteration 1.
Step 1. Initialization: $\alpha = 0, \nu = 1.2071.$

Step 2. $10 \times max(\tau, \alpha) = 0.0100.$

$$\sigma = 0.0059$$

$$H(\sigma) = \begin{pmatrix} -1 & 1 & -0.0059 & 0 \\ 0 & -0.0001 & 0 & -0.0059 \\ 0.0059 & 0 & 1 & 0 \\ 0 & 0.0059 & -1 & 0.0001 \end{pmatrix}$$

The eigenvalues of $H(\alpha)$: $-1, 1, \pm 0.0083j$.

Set $\nu = \sigma = 0.0059$.
$\nu = 0.0059$ less than $10 \times max(\tau, \alpha) = 0.0100$, **stop**.

Conclusion: $\beta(A) \leq 0.0059 < 10\tau$.

Remark: *Signficant Computational Cost* for finding if $H(\alpha)$ has imaginary eigenvalue. **Not practical for large problems.**

Convergence. If $\tau = \frac{1}{2}10^{-p}\|A + A^*\|$, then at most $\log_2 p$ bisection steps are required; for example, if $\tau = \frac{1}{2} \times 10^{-8}\|A + A^*\|$, then at most three bisection steps are required.

MATCONTROL NOTE. The Bisection algorithm has been implemented in MATCONTROL function **disstabc**.

Distance to an Unstable System and Lyapunov Equation

- Let $\boldsymbol{A}$ **be complex stable**
- Let X satisfy the Lyapunov equation:

$$XA + A^*X = -M,\ M > 0.$$

- Then

$$\beta(A) \geq \frac{\lambda_{min}(M)}{2 \parallel X \parallel_2}$$

- $\lambda_{min}(M)$ denotes the smallest eigenvalue of M.

Example: Consider the same A as in the previous example.

- Take $M = I_2$.
- $\beta(A) \geq 5.002 \times 10^5$

Stability Radius
(Chapter 10)

- Measures the distance of a stable matrix from the set of unstable matrices, where the distance is measured by the size of additive perturbations.
- $r_F(A, B, C) = \inf \{\sigma_1(\Delta) : A + B\Delta C \text{ is unstable}\}$.

$F = \mathbb{C}$ *or* $\mathbb{R}$ (Complex or Real)

Δ – – Variable

$B, C,$ – – Fixed

σ_1 – – Largest singular value.

Theorem: (Stability Radius and Algebraic Riccati Equation)

Let A be a complex stable matrix and let $r \equiv r_{\mathbb{C}}(A, B, C) < \infty$. Let $\rho \in (-\infty, r^2)$. Then there exists a unique Hermitian stabilizing solution X of the Riccati equation:

$$XA + A^*X - \rho C^*C - XBB^*X = 0.$$

Moreover, when $\rho = r^2$, there exists a unique solution X having the property that the matrix $A - BB^*X$ is unstable.

Conversely, if A is stable and if there exists a Hermitian solution X of the above algebraic Riccati equation, then necessarily $\rho \leq r^2$.

A Characterization of Complex Stability Radius

- Define $H_\rho = \begin{pmatrix} A & -BB^* \\ \rho_{CC^*} & -A^* \end{pmatrix}$

- Then

$\rho < r_F(A, B, C)$ if and only H_{ρ^2} does not have on eigenvalue on the imaginary axis

Algorithm: A Bisection Method for the Complex Stability Radius. (Algorithm 10.7.1)

Inputs:

1. The system matrices A, B, and C.
2. Some upper and lower estimates ρ_0^+ and ρ_0^- of the complex stability radius ρ.

Output:
An approximate value of the complex stability radius ρ.
For $k = 0, 1, 2, \cdots$, do until convergence.

Step 1. Take $\rho_k = \dfrac{\rho_k^- + \rho_k^+}{2}$ and compute $H_{\rho_k^2}$.

Step 2. If $H_{\rho_k^2}$ has eigenvalues on the imaginary axis, set $\rho_{k+1}^- \equiv \rho_k^-$ and $\rho_{k+1}^+ \equiv \rho_k$. Otherwise set $\rho_{k+1}^- \equiv \rho_k$ and $\rho_{k+1}^+ \equiv \rho_k^+$.
End

Example. $A = \begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix}, B = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, C = (1, 0)$.

Take $\rho_0^- = 0,\ \rho_0^+ = 1$.

$\underline{k = 0.}$ **Step 1.** $\rho_0 = \dfrac{1}{2}$. $H_{\rho_0^2}$ does not have an imaginary eigenvalue.

Step 2. $\rho_1^- = \dfrac{1}{2},\ \rho_1^+ = 1$.

$\underline{k = 1.}$ **Step 1.** $\rho_1 = \dfrac{3}{4}$. $H_{\rho_1^2}$ does not have an imaginary eigenvalue.

Step 2. $\rho_2^- = \dfrac{3}{4},\ \rho_2^+ = 1$

$\underline{k=2.}$ **Step 1.** $\rho_2 = \dfrac{7}{8}$. $H_{\rho_2^2}$ has an imaginary eigenvalue.

Step 2. $\rho_3^- = \dfrac{3}{4},\ \rho_3^+ = \dfrac{7}{8}$

$\underline{k=3.}$ **Step 1.** $\rho_3 = \dfrac{13}{16}$. $H_{\rho_3^2}$ does not have an imaginary eigenvalue.

Step 2. $\rho_4^- = \dfrac{13}{16},\ \rho_4^+ = \dfrac{7}{8}$

$\underline{k=4.}$ $\rho_4 = \dfrac{27}{32}$.

The iteration is converging towards $r = 0.8660$. The readers are asked to verify this by carrying out some more iterations.

MATCONTROL function: **stabradc**.

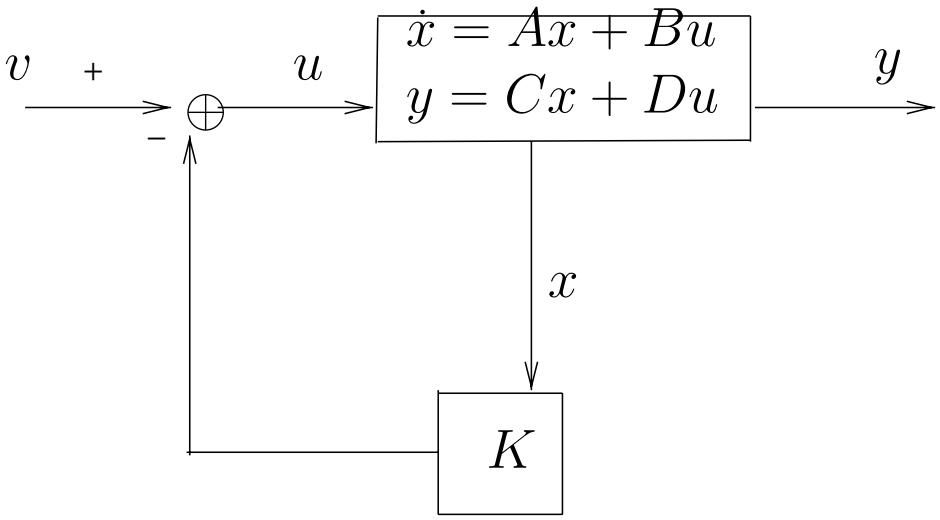
Feedback Stabilization and LQR Design (Chapter 10)

Feedback Stabilization Problem

Find a Stabilizing matrix K such that $(A - BK)$ is stable

Two Approaches for State Feedback Stabilization

- Lyapunov Equation Approach
- LQR Approach.



title

Figure 10.1: State Feedback Configuration

A Lyapunov-Equation Method For Stabilization

- (A, B) Controllable
- $|\lambda_{max}(A)|$ - Eigenvalue of A with the largest real past

Step 1. Choose $\beta > |\lambda_{max}(A)|$
Step 2. Solve the Lyapunov equation for Z:

$$-(A + \beta I)Z + Z[-(A + \beta I)]^T = -2BB^T.$$

Step 3. Obtain the stabilizing feedback matrix K

$$K = B^T Z^{-1}.$$

MATCONTROL Note: The above method has been implemented in MATCONTROL Function: **stablyap**.

• Similar Method for Discrete-time stabilization (**Theorem 10.2.4**) MATCONTROL Function: **stablyapd**.

Example (***Stabilizing the motion of the Inverted Pendulum***) (The **problem of a cart with inverted pendulum**) with the following data:

$$m = 1kg$$

$$M = 2kg$$

$$l = 0.5 \text{ meters}$$

$$\text{and } g = 9.18 \text{ meters per sec}^2.$$

Then

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 00 & -3.6720 & 0 & \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 22.0320 & 0 \end{pmatrix}.$$

The eigenvalues of A are $0, 0, \pm 4.6938$. Thus, *with no control input, there is an instability in the motion and the pendulum will fall.*

Stabilization using Lyapunov Equation.

$$B = \begin{pmatrix} 0 \\ 0.4 \\ 0 \\ -0.4 \end{pmatrix}.$$

Step 1. $\beta = 5$. $-(A + \beta I)$ stable.

Step 2. $Z = \begin{pmatrix} 0.0009 & -0.0044 & -0.0018 & 0.0098 \\ -0.0044 & 0.0378 & 0.0079 & -0.0593 \\ -0.0018 & 0.0079 & 0.0054 & -0.0270 \\ 0.0098 & -0.0593 & -0.0270 & 0.1508 \end{pmatrix}$

(The computed Z is symmetric positive definite but highly ill-conditioned).

Step 3. $K = B^T Z^{-1} = 10^3(-0.5308, -0.2423, -1.2808, -0.2923)$

Verify: The eigenvalues of $A-BK$ are $\{-5\pm 11.2865j, -5\pm 0.7632j\}$.

- $A - BK$ Stable

Continous-time LQR Program

- Given

$$Q = Q^T \geq 0 \text{ (Weight for the State)}$$

$$R = R^T > 0 \text{ (Weight for the Control)}$$

- Find the optimal control vector $u(t)$ such that

$$J_c(x) = \int_0^\infty [x^T Q x + u^T R u] dt$$

is minimized
subject to

$$\dot{x} = Ax + Bu,\ x(0) = x_0$$

$$y = Cx$$

Solution of the LQR Problem

Suppose

- (A, B) Stabilizable
- (A, Q) Detectable
- X unique symmetric positive definite solution of the CARE:

$$XA + A^T X + Q - XBR^{-1}B^T X = 0.$$

Then

- Optimal control vector $u^0(t) = -Kx(t)$

  where

$$K = R^{-1}B^T X$$

- $A - BK$ is **Stable**
- Minimum value of $J_c(x)$ is $x_o^T X x_o$.

Definition: The algebraic Riccati equation

$$XA + A^T X + Q - XSX = 0,$$

where $S = BR^{-1}B^T$ is called the **Continuous-Time Algebraic Riccati Equation** or in short **CARE**.

Definition: The matrix H defined by

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}$$

is the Hamiltonian matrix associated with the CARE.

Definition: A symmetric solution X of the CARE such that $A - SX$ is stable, is called a **stabilizing solution.**

Algorithm: The Continuous-time LQR Design Algorithm

Inputs: The matrices A, B, Q, R, and $x(0) = x_0$.

Outputs: X–The solution of the CARE
K–The LQR feedback gain matrix
$J_{c\min}$–The minimum value of the cost function $J_C(x)$.

Assumptions:

1. (A, B) is stabilizable and (A, Q) is detectable.

2. Q is symmetric positive semidefinite and R is symmetric positive definite.

• **Detectbility**: (A, C) is detectable if there exists a matrix L such that $A - LC$ is stable.

Step 1. Compute the stabilizing solution X of the CARE:

$$XA + A^T X - XSX + Q = 0, \quad S = BR^{-1}B^T.$$

Step 2. Compute the LQR feedback gain matrix:

$$K = R^{-1}B^T X$$

Step 3. Compute the minimum value of

$$J_C(x) : J_{c\min} = x_0^T X x_0.$$

Example: (LQR Design for the Inverted Pendulum).

and $Q = I_4, R = 1$, and $x_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$.

Step *1.* The unique positive definite solution X of the CARE (obtained by using MATLAB function **care**) is

$$X = 10^3 \begin{pmatrix} 0.0031 & 0.0042 & 0.0288 & 0.0067 \\ 0.0042 & 0.0115 & 0.0818 & 0.0191 \\ 0.0288 & 0.0818 & 1.8856 & 0.4138 \\ 0.0067 & 0.0191 & 0.4138 & 0.0911 \end{pmatrix}$$

Step *2.* The feedback gain matrix K is

$$K = (-1, -3.0766, -132.7953, -28.7861).$$

Step *3.* The minimum value of $J_C(x)$ is 3100.3.

• The eigenvalues of $A-BK$ are: $-4.8994, -4.5020, -0.4412 \pm 0.3718j$.

• *Entries of K smaller than those obtained by Lyapunor Method.*

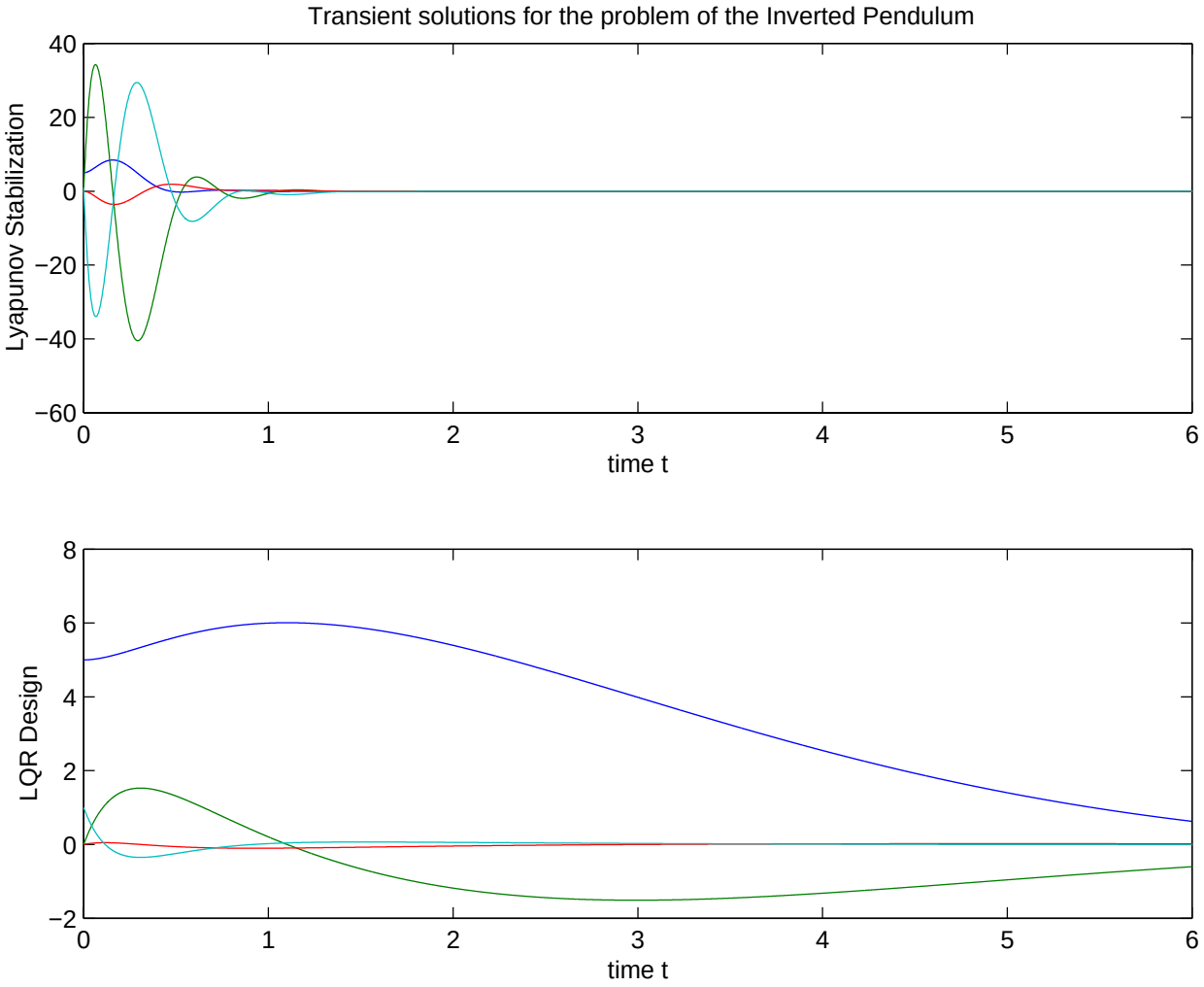


Figure Comparison of Transient Responses

- The largest magnitude in transient solution with Lyapunov approach is **SIX** times larger then the one with LQR design.

Numerical Solutions and Conditioning of the Lyapunov and Sylvester Equations (Chapter 8)

Lyapunov and Sylvester Equations (Chapter 8)

- $XA + A^T X = C$ **(Continuous-time Lyapunov Equation)**
- $A^T XA - X = C$ **(Discrete-time Lyapunov Equation)**
- $XA + BX = C$ **(Sylvester Equation)**

Applications

- Stability and Robust Stability Analysis
- Balancing and Model Reduction
- Solution of Riccati Equation via Newton's Method

In the above equations, the dimensions of A, B, and C are:

- $A \in \mathbb{R}^{n \times n}$
- $B \in \mathbb{R}^{m \times m}$
- $C \in \mathbb{R}^{m \times n}$

A Template for Numerical Solution of the Sylvester Equation

Step 1. **Transform A and B to *condensed forms* using similarity:**

$$U^{-1}AU = \tilde{A}, \;\; V^{-1}BV = B \text{ and } V^{-1}CU = \tilde{C}.$$

Step 2. **Solve the reduced problem:**

$$Y\tilde{A} + \tilde{B}Y = \tilde{C}.$$

where

$$Y = V^{-1}XU.$$

Step 3. **Recover the solution X:**

$$X = VYU^{-1}.$$

Some widely used condensed forms:

- Diagonal forms
- Companion forms
- Jordan Canonical forms

These forms have to be avoided.

Condensed forms of the choice should be:

- Hessenberg Form
- Real Schur Form

Numerical Methods for Lyapunov and Sylvester Equations

- The Schur Method for the Lyapunov Equation
- The Hessenberg-Schur Method for the Sylvester Equation
- The Modified Schur Methods for the Cholesky Factors of the Lyapunov Equations.
- **Solutions via diagonalization or companion form should be avoided** (see **Example 8.8.1** of the book).

A Remark on using Companion Form

Extensive numerical experiments show that the solution of the Lyapunov or Sylvester equation using companion form of A of sizes larger than 15 have errors almost as large the solutions themselves.

The Schur Method

Step 1. Reduction of the Problem

$$XA + A^T X = C \longrightarrow YR^T + RY = \hat{C}$$

- $R = U^T A^T U$ (Real-Schur form of A^T)
- $\hat{C} = U^T C U$
- $Y = U^T X U$

Step 2. Solve the reduced equation

$$YR^T + RY = \hat{C}$$

by solving algebraic linear systems.

Step 3. Recover the solution X:

$$X = UYU^T.$$

- **Flop-Count:** Approximately $32n^3$

  ($26n^2$ for Real Schur Form).

- **MATCONTROL Function**: **lyaprsc** (Real Schur)
- **MATLAB Function**: **lyap** (Complex Schur)

Notations: $Y = (y_1, \ldots, y_n)$

$\hat{C} = (c_1, \ldots, c_n)$.

An Illustration to compute Y: $YR^T + RY = \hat{C}$

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{pmatrix}, \quad \hat{C} = (c_1, c_2, c_3)$$

- Compute y_3 by solving a quasi-triangular system:

$$(R + r_{33}I)y_3 = c_3$$

- Compute y_1 and y_2 simultaneously by solving:

$$R(y_1, y_2) + (y_1 y_2) \begin{pmatrix} r_{11} & r_{21} \\ r_{12} & r_{23} \end{pmatrix} = (c_1 - r_{13}y_3, c_2 - r_{23}y_3).$$

The Hessenberg-Schur Method for Sylvester Equation (Algorithm 8.5.1).

Step 1. Transform the problem to a Hessenberg-Schur Problem

- $U^T A^T U = R$ (Real Schur Form of A^T). (**QR Iteration**)
- $V^T B V = H$ (Upper Hessenberg matrix). (**Householder's Method**)

Step 2. Solve the Reduced Equation: $YR^T + HY = \hat{C}$

- $Y = V^T X U$
- $\hat{C} = V^T C U$.

Step 3. Recover the solution: $X = VYU^T$.

An Example: Solving $YR^T + HY = \hat{C}$

$$R = \begin{pmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & r_{33} \end{pmatrix}, \quad \hat{C} = (c_1, c_2).$$

Then $YR^T + HY$ is equivalent to:

$$(H + r_{33}I)y_3 = c_3$$

and

$$\begin{pmatrix} H + r_{11} & r_{12}I \\ r_{21}I & H + r_{22}I \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} c_1 - r_{13}y_3 \\ c_2 - r_{23}y_3 \end{pmatrix}$$

- **Flop-Count:** Approximately

$$\left(10\frac{m^3}{3} + 26n^3 + 10m^2n + 5mn^2\right).$$

- **MATCONTROl Function**: **sylvhrsc**
- **MATLAB Function:** $X = \mathbf{lyap}\,(A, B, C)$ solves

$$AX + XB = -C$$

using **complex schur decomposition**.

The Cholesky Factors of the Lynpunov equation.

- Algorithms (**Algorithms 8.6.1 and 8.6.2**) exist to compute the Cholesky factors without expliciting computing the symmetric positive definite solutions X.
- **MATCONTROL Functions: LYAPCHLD** and **LYAPCSD**.
- No equivalent MATLAB Functions.

Comparisons of Different Methods and Recommendation

The numerical methods of choice are:

- The **Schur method** for the Lyapunov equations.
- The **Hessenberg-Schur** method for the Sylvester equation. **(Algorithm 8.5.1)**.
- **The modified Schur methods (Algorithms 8.6.1 and 8.6.2)** for the Cholesky factors.

Perturbation Analysis and Conditioning of Lyapunov and Sylvester Equation

Perturbation Analysis and Conditioning of the Sylvester Equation

- The Sylvester equation is equivalent to

$$Px = c$$

  where $P = (I_n \otimes B) + (A^T \otimes I_m)$.

- Define $\delta = ||P^{-1}||_2 \dfrac{(\alpha + \beta)||X||_F + \gamma}{||X||_F}$,

  where $||\Delta A|| \leq \epsilon\alpha$, $||\Delta B||_F \leq \epsilon\beta$ and $||\Delta C||_F \leq \epsilon\gamma$.

$$\epsilon = \max\left\{ \frac{||\Delta A||_F}{\alpha}, \frac{||\Delta B||_F}{\beta}, \frac{||\Delta C||_F}{\gamma} \right\}.$$

Then

$$\frac{||\Delta x||_F}{||x||_F} = \frac{||\hat{X} - X||_F}{||X||_F} \leq \sqrt{3}\epsilon\delta.$$

- Define the *sep function*

$$\frac{1}{sep(B,-A)} = \frac{1}{\sigma_{\min}(P)}$$

Then

$$\boxed{\frac{||\hat{X}-X||_F}{||X||_F} < \sqrt{3}\epsilon \frac{1}{sep(B,-A)} \frac{(\alpha+\beta)||X||_F+\gamma}{||X||_F}}$$

- $sep(B,-A)$ plays the dominat role in determining the conditioning of the Lyapunov, and Sylvester equations.

$$small\ sep \Longrightarrow \text{ Ill- Conditioning}$$
$$large\ sep \Longrightarrow \text{ Well-Conditioning}$$

Example:

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \; B, \begin{pmatrix} -0.9888 & 0 & 0 \\ 0 & -0.9777 & 0 \\ 0 & 0 & -0.9666 \end{pmatrix}$$

$$X = \textbf{ Exact Solution } = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

$$C = XA + BX + C.$$

- $sep(B, -A) = 1.4207 \times 10^{-6}$ (**small**)

The Sylvester equation is expected to be ill-condition

Verify: Change the (1, 1) entry of A to 0.99999

- Relative Error in $A = 0(10^{-7})$ (**quite small**)
- Relative Error in $X = 0.2366$ (**very large**)

  $\hat{X}$ = **Computed Solution**

$$= \begin{pmatrix} 1.0001 & 0.9920 & 1.7039 \\ 1.0000 & 0.9980 & 1.0882 \\ 1.0000 & 0.9991 & 1.0259 \end{pmatrix}$$

Conclusion

- If $\text{sep}(B, -A)$ is small, then the problem is likely to be ill-conditioned.

The Condition Number

- The condition number of the Sylvester equation has been implemented in MATCONTROL
  function CONDSYLVC.
  (NO MATLAB Function).

Computing the sep function

- A Bisection algorithm exists (**Algorithm 8.3.1**).
- **MATCONTROL Functions: sepkr** and **sepest**

Numerical Stability of the Schur and the Hessenberg-Schur Methods.

- **The relative residual:**

$$\frac{||C - (\hat{X}A + B\hat{X})||_F}{||\hat{X}||_F} \leq (\mu(||A||_F + ||B||_F)).$$

  is **guaranteed to be small.**

- The smallness of the residual does not guarantee numerical stability.
- The algorithms are **conditionally stable,** that is, they are numerically stable only for well-conditioned problems.

NUMERICAL SOLUTIONS and CONDITIONING of ALGEBRAIC RICCATI EQUATIONS

(Chapter 13)

Continuous- Time Algebraic Riccati Equations (CARE).

- $XA + A^TX - XBR^{-1}B^TX + Q = 0.$
- **Discrete-time Algebraic Riccati Equation (DARE)**

  $A^TXA - X + Q - A^TXB(R + B^TXB)^{-1}B^TXA = 0.$
- of interest is the **unique stabilizing solution**.
- **Assumptions for unique stabilizing solutions**

  (i) (A, B) is stabilizable (Discrete-stabilizable)

  (ii) The Hamiltonian (Symplectic matrix) does not have an imaginary eigenvalue (eigenvalue on the unit circle.)
- **Symplectic Matrix** (for the **Discrete-System)**

$$M = \begin{pmatrix} A + S(A^{-1})^TQ & -S(A^{-1})^T \\ -(A^{-1})^TQ & (A^{-1})^T \end{pmatrix}$$

$$S = BR^{-1}B^T.$$

Computational Methods for the CARE and DARE

- The Invariant Subspace Methods
- The Deflating Subspace Methods
- Newton's Methods
- Matrix Sign-Function Methods

Invariant Subspace Methods (e.g. Eigenvector Methods, Schur Methods)

- For the CARE

Compute the stable invariant subspace of the **Hamiltonian matrix**

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}$$

$S = BR^{-1}B^T$.

If this subspace is spanned by the columns of $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ and X_2 is invertible. Then the **stabilizing solution** X:

$$X = X_2 X_1^{-1}.$$

- For the DARE

Compute the stable invariant subspace of the **symplectic matrix**

$$M = \begin{pmatrix} A + S(A^{-1})^T Q & -S(A^{-1})^T \\ -(A^{-1})^T Q & (A^{-1})^T \end{pmatrix}$$

$$S = BR^{-1}B^T$$

The Eigenvector Method

- **For the CARE**

Step 1. Diagonalize $V^{-1}HV = \begin{pmatrix} -\overline{\Lambda} & 0 \\ 0 & \Lambda \end{pmatrix}$, $\Lambda = \text{diag}$ $(\lambda_1, \ldots, \lambda_n)$.

Step 2. Partition $V = \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}$

Step 3. Compute $X = V_{21}V_{11}^{-1}$.

- Highly unstable if the matrix H is defective or nearly defective.

  Cannot be Recommended for Practical use, in general.

- **MATCONTROL Function: riceig.**

- **For the DARE**

Analogous method. Based on diagonalization of the symplectic matrix.

- Not recommended for practical use.

The Schur Method (Based on Ordered Real Schur Form) For the CARE (Algorithm 13.5.1)

Step 1. Compute the ordered Real Schur Form of H:

$$U^T H U = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$$

• Spec (T_{11}) = {Eigenvalues of H with negative real parts}.

Step 2. Portion conformably $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$.

Step 3. Compute $X = U_{21} U_{11}^{-1}$.

MATCONTROL Functions: ricsch.

Numerical Stability:

- Numerical difficulties arise when H is nearly defective.
- Widely used in practice.

The Schur Method for the DARE

- Not used in practice, because of the requirement of computing $A-1$ to form the symplectic matrix.

Deflation Subspace Methods (Generalized and Inverse-Free Generalized Eigenvectors and Schur Methods)

For the CARE - Based on findings basis for stable deflating subspace of the pencil: $P_{CARE} - \lambda N_{CARE}$

- $P_{CARE} = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}, S = BR^{-1}B^T.$
- $N_{CARE} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}.$

For the DARE: The Pencil is $P_{DARE} - \lambda N_{DARE}$

- $P_{DARE} = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}$
- $N_{DARE} = \begin{pmatrix} I & S \\ O & A^T \end{pmatrix}$.

(Requires no inversion of A This approach is specially significant for the DARE)

Inverse-Free Option

For the CARE: *From the Extended Pencil*: $P^E_{CARE} - \lambda N^E_{CARE}$ of the order $(2n+m)$ **(If R^{-1} to be avoided)**

- $P^E_{CARE} = \begin{pmatrix} A & 0 & B \\ -Q & -A^T & 0 \\ 0 & B^T & R \end{pmatrix}$

- $N^E_{CARE} = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix}$

For the DARE: The extended pencil

- $P^E_{DARE} = \begin{pmatrix} A & 0 & -B \\ -Q & -I & 0 \\ 0 & 0 & R \end{pmatrix}$
- $N^E_{DARE} = \begin{pmatrix} I & 0 & 0 \\ 0 & A^T & 0 \\ 0 & B^T & 0 \end{pmatrix}$.

No Inversion of R Necessary

The Compressed Pencil Option of Order $2n$

For the Care: $P^{EC}_{CARE} - \lambda N^{EC}_{CARE}$

- QR Factorization: $\begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} \quad \begin{pmatrix} R \\ B \end{pmatrix} = \begin{pmatrix} \tilde{R} \\ U \end{pmatrix}$

- $P^{EC}_{CARE} = \begin{pmatrix} W_{22}A & W_{21}B^T \\ -Q & -A^T \end{pmatrix}$

- $N^{EC}_{CARE} = \begin{pmatrix} W_{22} & 0 \\ 0 & I \end{pmatrix}.$

For the DARE: $P^{EC}_{DARE} - \lambda N^{EC}_{DARE}$

- QR Factorization: $\begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} \begin{pmatrix} R \\ -B \end{pmatrix} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}$.
- $P^{EC}_{DARE} = \begin{pmatrix} W_{22}A & 0 \\ -Q & -I \end{pmatrix}$.
- $N^{EC}_{DARE} = \begin{pmatrix} W_{22} & W_{21}B^T \\ 0 & A^T \end{pmatrix}$.

- **The Inverse-Free Generalized Schur Algorithm (Algorithm 13.5.3) for the CARE**

Step 1. Transform the compressed pencil to the ordered RSF:

$$Q_1(P_{CARE}^{EC} - \lambda N E_{CARE}^{EC})Z = \tilde{M} - \lambda\tilde{N}$$

- $\tilde{M}$ = Quasi-upper triangular
- $\tilde{N}$ = Upper triangular
- The n stable eigenvalues appear first.

Step 2. Partition $Z = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$

Step 3. Compute X: $X = Z_{21}Z_{11}^{-1}$

MATLAB functions: **care** and **dare** are based on inverse-free generalized schur algorithms.

- Analogous Method for the DARE **(Algorithm 13.5.4)**
- **Implementation of step 1**
  Step 1 is implemented by using QZ algorithm for the matrix pencil $A - \lambda B$ (see **Chapter 4**).

Newton's Method for the CARE (Algorithm 13.5.8).

Step 1. Start with an Initial Approx. X_0.

Step 2. Compute the Succesive Approximations $\{X_i\}$:

$$(A - SX_k)^T X_{k+1} + X_{k+1}(A - SX_k) = -X_k S X_k - Q.$$

(Lyapunov Equation)

$$S = BR^{-1}B^T.$$

Step 3. Continue until Convergence.

Newton's Method for the DARE (Algorithm 13.5.10)

- Analogous
- Based on the successive solution of the discrete-time Lyapunov equations.

Stopping Criterion: Stop if for certian value of k and prescribed tolerenace ϵ:

$$\frac{||X_{k+1} - X_k||_F}{||X_k||_F} \leq \epsilon$$

or

k **exceeds certain porescribed number** N

Convergence Analysis of Newton's Method for the CARE

- **Assumptions**
  (i) (A, B) is stabilizable
  (ii) $R > 0$
  (iii) The CARE has a unique stabilizing solution X.

Convergence Results

- All X_i are stabilizing.
- $X \leq \ldots \leq X_{i+1} \leq X_i \ldots \leq X_1$. (Monotocally decreasing)
- $\lim_{i \to \infty} X_i = X$
- **Quadratic Convergence:** $||X_{i+1} - X| \leq c||X_i - X||^2$ for $i \geq 1$.

Convergence Analysis of Newtons's Method for the DARE

- Analogous results exist (**Theorem 13.5.11**).

Example Solve the CARE with

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & 3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad Q = I_3, R = 1.$$

$$X_0 = \begin{pmatrix} 0.4 & 0.1 & 0.1 \\ 0.1 & 0.3 & 0 \\ 0.1 & 0 & 0.2 \end{pmatrix}.$$

Table of Relative Changes

i	Relative Change $\|\|X_{i+1} - X_i\|\|/\|\|X_i\|\|$
0	0.1507
1	0.0038587
2	2.4025×10^{-6}
3	5.5392×10^{-13}

Newton's Method with Line Search (Algorithm 13.5.9)

Newton's iterates can be written in the form:

$$X_{i+1} = X_i + \Delta_i$$

where Δ_i is the solution of the Lyapunov equation

$$(A - SX_i)^T\Delta_i + \Delta_i(A - SX_i) + A^TX_i + X_iA + Q - X_iSX_i = 0.$$

Idea: Replace the iteration $X_{i+1} = X_i + \Delta_i$

by

$$X_{i+1} = X_i + t_i\Delta_i,$$

where t_i is a scalar to be chosen such that $||R_c(X_i + t_i\Delta_i)||_F$ is minimized.

- $R_c(X) = XA + A^TX - XSX + Q$

Recommendations

A. **For the CARE:** $XA + A^TX - XBR^{-1}B^TX + Q = 0$

- The Schur Method (**Algorithm 13.5.1**)
  or
  Inverse-Free Generalized Schur Method (**In case R is ill-conditioned**) **(Algorithm 13.5.3)**
  Followed by
- Newton's Method **(Algorithm 13.5.8)** (As Iterative Refinement)
  Line Search Algorithm Preffered (Algorithm in 13.5.9)

B. **For the DARE**

$$A^T X A - X + Q - A^T X B(R + B^T X B)^{-1} B^T X = 0.$$

- Inverse-Free Generalized Schur Method (**Algorithm 13.5.4**)

Followed by

- Newton's Method **(Algorithm 13.5.10)**. Line Search Algorithm Preferred **(Algorithm 13.5.11)**

Software for Riccati Equations

- **MATCONTROL**

RICSCHC — **The Schur Method**
RICNWTNC — **The Newton's Method**
RICNWLSC — **Newton's Method with Line Search**
RICSGNC — **The Matrix Sign-Function Method**

(Discrete versions are availabel)

- **MATLAB**
  CARE- **The Inverse-Free Generalized Schur Method.**
  DARE- **The Inverse-Free Generalized Schur Method for the DARE**
- **CSP-ANM**
  Riccati Solve $[a, b, q, r,$ *Solve Method* $\longrightarrow$ *Schur Decomposition]* $\longrightarrow$ *The Schur Method*

  Riccati Solve $[a, b, q, r,$ *Solve Method* $\longrightarrow$ *Generalized Schur Decomposition]* $\longrightarrow$ *The Generalized Schur Method*
  Riccati Solve $[a, b, q, r,$ *Solve Method* $\longrightarrow$ *Newton, Initial Guess* $\rightarrow w_0]$ $\longrightarrow$ *Newton's Method*
- **SLICOT**
  SBOZOD - The **Generalized Schur Method**

Conditioning of the Riccati Equations

Define three operators

- $\Omega(Z) = (A - SX)^T Z + Z(A - SX)$
- $\Theta(Z) = \Omega^{-1}(Z^T X + XZ)$
- $\Pi(Z) = \Omega^{-1}(XZX)$.

Define for any unitarily invariant norm.

- $l = ||\Omega^{-1}||^{-1}$
- $p = ||\Theta||$
- $q = ||\pi||$

Theorem on Perturbation Bound for the CARE

CARE: $XA + A^T X + Q - SXS = 0$

Perturbed CARE: $(X + \Delta X)(A + \Delta A) + (A + \Delta A)^T (X + \Delta X) + (Q + \Delta Q) - (X + \Delta X)(S + \Delta S)(X + \Delta X) = 0.$

$$\frac{||\Delta X||}{||X||} \leq \frac{||Q||}{l||X||} \cdot \frac{||\Delta Q||}{||Q||} + \phi \frac{||A||}{||X||} \cdot \frac{||\Delta A||}{||A||} + q \frac{||S||}{||X||} \cdot \frac{||\Delta S||}{||S||}$$

The absolute condition numbers with respect to Q, A, and S:

- $\kappa^{AB}_{CARE}(Q) = \dfrac{1}{l}$
- $\kappa^{AB}_{CARE}(A) = p$
- $\kappa^{AB}_{CARE}(S) = q$
- The relative condition numbers are similary defined.

Estimating the Condition Numbers

- **Assume that $A - SX$ is stable.**

Define $H_k, k = 0, 1, 2$ by

- $(A - SX)^T H_k + H_k(A - SX) = -X^k,\ k = 0, 1, 2.$ **(Lyapunov Equations)**.

The measures of sensitivity with respect to Q, A, and S:

- $r_1 = \dfrac{||H_0||\ ||Q||}{||X||}$ (**Sensitivity of X w.r.t. Perturbation in Q**)
- $r_2 = \dfrac{||H_1^{(1)}||||A||}{||X||}$ **(Sensitivity of X w.r.t. Perturbation in A)**
- $r_3 = \dfrac{||H_2||||S||}{||X||}$ **(Sensitivity of X w.r.t. Perturbation in S)**

$H^{(1)}$ is defined out of H_0 and H_2.

An Example

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0.0010 & 4 & 5 \\ 0 & 7 & 8 \end{pmatrix}, \; B = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, R = 1.$$

$$Q = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 5 & 3 \\ 1 & 3 & 5 \end{pmatrix}$$

- $H_2 = 4.8581 \times 10^{18}$.

The CARE is expected to be ill-conditioned w.r.t. the perturbation in B.

Verify: Change B to $B + \Delta B$

$$\Delta B = 10^{-8} \begin{pmatrix} -4.939 \\ 0.7715 \\ -0.9411 \end{pmatrix}.$$

- Relative Error in $B = \dfrac{||B_{new} - B||}{||B||} = 0(10^{-8})$.
- Relative Error in $X = \dfrac{||X_{new} - X||}{||X||} = 0(10^{-5})$.

Numerical Methods and Conditioning of Eigenstructure Assignment (Pole-Placement) (Chapter 11)

Motivation for the Eigenvalue and Eigenstructure Assignment

- Stability is not enough
- A designer should be able to choose feedback such that the closed-loop system has certain transient responses determined by the eigenvalues and eigenvectors of the system.

The Second-order System

$$\ddot{x}(t) + 2\zeta\omega_n\dot{x}(t) + \omega_n^2 x(t) = 0$$

The eigenvalues are:

$$\lambda_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$$

ζ - **damping ratio**

ω_n - **undamped natural frequency.**

The response, of the dynamical system depends upon ζ and ω_n.

- In general, for a fixed value of ω_n, *the larger the value of* $\zeta(\zeta \geq 1)$, *is, more smoother but slower the responses become.*
- *The smaller the value of* $\zeta(0 \leq \zeta < 1)$ *is, the responses become faster but more oscillatory.*

Eigenvalue Assignment by State Feedback

Given $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}$ $(m \leq n)$ and $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$, where Λ is closed under complex conjugation, find $K \in \mathbb{R}^{m\times n}$ such that

$$\Omega(A - BK) = \Lambda.$$

Here $\Omega(R)$ stands for the spectrum of R.

Theorem (The State Feedback Eigenvalue Assignment Theorem) *The EVA problem is solvable for all Λ if and only if (A, B) is controllable. The solution is unique if and only if the system is a single-input system (that is, if B is a vector). In the multi-input case, if the problem is solvable, there are infinitely many solutions.*

Ackermann's Formula (1972).

The well-known Ackermann formula (**Single-input Case**):

$$f = e_n^T C_M^{-1} d(A)$$

- C_M is the controllability matrix
- $d(A)$ is the characteristic polynomial of the desired closed-loop matrix.

- **Ackermann's formula is not numerically viable.**
- The MATLAB function **acker** has implemented Ackermann's formula and *comments have been made about the numerical difficulty with this formula in the MATLAB user's manual.*

Example: $A - 9 \times 9$ **Random Matrix**

$$B = \begin{pmatrix} -0.1510 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \Lambda = \begin{pmatrix} -1.0000 \\ -1.5000 \\ -2.0000 \\ -2.5000 \\ -3.0000 \\ -3.5000 \\ -4.0000 \\ -4.5000 \\ -5.0000 \end{pmatrix}$$

The eigenvalues assigned by the **Ackermann formula** are:

$$\begin{pmatrix} -0.8824 - 0.4891j \\ -0.8824 + 0.4891j \\ -2.2850 - 1.0806j \\ -2.2850 + 1.0806j \\ -3.0575 \\ -3.8532 \\ -4.2637 - 0.7289j \\ -4.2637 + 0.7289j \end{pmatrix}$$

- **The desired eigenvalues in Λ are completely different from those assigned by the Ackermann formula.**

The eigenvalues assigned by MATLAB function **place** are:

$$\begin{pmatrix} -4.9999 \\ -4.5001 \\ -4.0006 \\ -3.4999 \\ -3.0003 \\ -2.4984 \\ -2.0007 \\ -1.5004 \\ -0.9998 \end{pmatrix}$$

Similar results were obtained by the simple recursive Algorithm
(Algorithm 11.2.1)

Some Numerically Viable Algorithms for Eigenvalue Assignment

- The Single-Input Recursive Algorithm (Datta, (1987)).
- An RQ Implementation of the Recursive Algorithm (Arnold and Datta (1998)).
- The Multi-Input Recursive Algorithm (Arnold and Datta (1990)).
- The Explicit QR Algorithms (Miminis and Paige (1982), (1988)).
- The Implicit QR Algorithm (Patel and Misra (1984)).
- The Schur Method (Varga (1991)).
- Algorithm for Partial Assignment (Datta and Sarkissan (2002)).

A Template of Numerical Algorithm for EVA Problem

Step 1. Transform the pair (A, B) to **controller Hessenberg pair** $(H, \tilde{B})$:

$PAP^T = H$, an Unreduced Block Upper Hessenberg matrix,

$$PB = \tilde{B} = \begin{pmatrix} B_1 \\ 0 \end{pmatrix}, \ B_1 \text{ is upper triangular.}$$

Step 2. Solve the **Hessenberg Problem**; that is find F such that

$$\Omega(H - \tilde{B}F) = \{\lambda_1, \ldots, \lambda_n\}.$$

Note: In the single-input case, this step amounts to finding a row vector f^T such that $\Omega(H - \beta e_1 f^T) = \{\lambda_1, ..., \lambda_n\}$.

Step 3. Find K

$$K = FP.$$

Remark: Step 1 and Step 3 are standard. **The different algorithms differ in a way Step 2 is implemented.**

The Single-input EVA Problem

Given

- H – An unreduced upper Hessenberg Matrix
- $\{\lambda_1, \ldots, \lambda_n\}$, closed under complex conjugation

Find f – The feedback vector such that

$$\Omega(H - \beta e_1 f^T) = \{\lambda_1, ..., \lambda_n\}.$$

We will assume temporarily, without any loss of generality, that $\beta = 1$ (Recall that $Pb = \bar{b} = \beta e_1$.)

Algorithm: The Recursive Algorithm for the Single-Input Hessenberg EVA Problem (Algorithm 11.2.1)

Idea: Find a simultaneously a nonsingular matrix L and the feedback vector f such that

$$H^T L - L\Lambda^T = f e_1^T L$$

- $\Lambda^t = \begin{pmatrix} * & \lambda_1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & \ddots \\ & 0 & & \lambda_n \\ & & & * \end{pmatrix}$

Inputs:

- H — An Unreduced Upper Hessenberg
- $\{\lambda_1, \ldots, \lambda_n\}$ — The n Eigenvalues to be Assigned.

Output:

- The feedback vector f:

$$\Omega(A - bf^T) = \{\lambda_1, \ldots, \lambda_n\}.$$

Step 1. *Set* $l_1 = e_n$.

Step 2. *Construct a set of normalized vectors* $\{\ell_k\}$ *as follows:*

For $i = 1, 2, \ldots, n-1$ do

Compute $\hat{\ell}_{i+1} = (H^T - \lambda_i I)\ell_i$

$$\ell_{i+1} = \frac{\hat{\ell}_{i+1}}{\|\hat{\ell}_{i+1}\|_2}$$

End

Step 3. *Compute* $\ell_{n+1} = (H^T - \lambda_n I)\ell_n$.

Step 4. *Compute* $f = \dfrac{\ell_{n+1}}{\alpha}$, α *is the first entry of* ℓ_n.

Efficiency:

1. Steps 2 through 4: $\frac{n^3}{3}$ flops.
2. Reduction to Hessenberg form: $\frac{10}{3}n^3$.

Total: $\frac{11}{3}n^3$ flops.

- **Most efficient algorithm for single-input eigenvalue assignment proposed so far.**
- Extremely easy to implement.
- Extreme ill-conditioning might cause instability.

MATCONTROL: polercs.

Numerical Stability

- Reliable for all practical purposes
- Many numerical experiments suggest the algorithm works well even for some ill-conditioned problem.
- **The QR and RQ versions of the recursive algorithm are stable** (**Algorithms 11.2.2 and 11.2.3**)

  (Implemented in MATCONTROL Functions: **poleqrs** and **polerqs**, respictively)

Numerical Methods for Multi-input Eigenvalue Assignment

- The Multi-input Version (**Algorithm 11.3.1**) of the Recursive Algorithm (**Most Efficient**).
- The Explicit QR Algorithm (**Section 11.3.1**) (**Stable but can give Complex Feedback Matrix**)
- The Schur Method (Algorithm 11.3.3) (**Based on Real-Schur Decomposition**). *Very Expensive*

Algorithm 11.3.1. *The Recursive Algorithm for the Multi-input EVA Problem*

Idea: **Given**

- The controller Hessenberg pair $\left(H, \tilde{B} = \begin{pmatrix} R \\ 0 \end{pmatrix}\right)$

Find

- A nonsingular matrix L
  and
- A feedback matrix F
  such that

$$LH - \Gamma H = \begin{pmatrix} R \\ 0 \end{pmatrix} F.$$

Inputs:
A - The $n \times n$ state matrix
B - The $n \times m$ input matrix ($m \leq n$).
S - The set of numbers $\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$, closed under complex conjugation.

Assumption : (A, B) is controllable.

Output: A feedback matrix K such that $\Omega(A-BK) = \{\lambda_1, \lambda_2, \cdots, \lambda_n\}$.

Step 1. Transform
• $PAP^T = H$, an unreduced block upper Hessenberg matrix

• $PB = \tilde{B} = \begin{pmatrix} R \\ 0 \end{pmatrix}$, R is upper triangular and has full rank.

Step 2. Partition S in such a way that $S = \cup\Omega(\Lambda_{ii})$, where each Λ_{ii} is an $n_i \times n_i$ diagonal matrix (n_i's are defined by the dimensions of the blocks in $H = (H_{ij})$; $H_{ij} \in \mathbb{R}^{n_i \times n_j}$).

Step 3. Set $L_k = (0, \cdots, 0, I_{n_k})$.

Step 4. For $i = k-1, \cdots, 1$ do

4.1 Compute $\tilde{L}_i \equiv L_{i+1}H - \Lambda_{i+1,i+1}L_{i+1}$

4.2 Compute the QR decomposition of

$\tilde{L}_i^T : \tilde{L}_i^T = QR$

4.3 Define $L_i = Q_1^T$, where Q_1 are the first n_i columns of the matrix $Q = (Q_1, Q_2)$

End

Step 5. Solve the linear system $(L_{11}R)F = L_1H - \Lambda_{11}L_1$ for F, where L_{11} is the matrix of the first n_1 columns of L_1.

Step 6. Compute the feedback matrix K of the original problem: $K \equiv FP$

Theorem 11.3.1 *The feedback matrix K constructed by the above algorithm is such that*

$$\Omega(A - BK) = \{\lambda_1, \lambda_2, \cdots, \lambda_n\}.$$

proof. Proof follows from the discussion preceding the algorithm.

■

Flop-count: Approximately $\frac{19}{3}n^3 + \frac{15}{2}n^2m$ flops are required to implement the algorithm.

Numerical Stability. Stable in practice. Reliable in the sense that the stability can be monitored by the conditioning of the structure matrix L.

Conditioning of the Closed-loop Eigenvalues

Question: How far are the eigenvalues of the computed closed loop matrix $\hat{M}_c = A - B\hat{K}$ from the desired eigenvalues $\{\lambda_1, \cdots, \lambda_n\}$?

Answer: *Even though a feedback matrix has been computed using a* ***numerically stable*** *algorithm, there is no guarantee that the eigenvalues of the closed-loop matrix will be near those which are to be assigned.*

Contribution Factors for Conditioning

- The conditioning of the problem of determining the feedback matrix K from the given data.
- The condition number (with respect to a p-norm) of the eigenvector matrix of the closed-loop system.
- The distance to uncontrollability, and the distance between the open-loop and closed-loop eigenvalues.
- The norm of the feedback matrix.

Example *Consider EVA with the following data:*

$$A = \begin{pmatrix} -4 & 0 & 0 & 0 & 0 \\ 0.0001 & -3 & 0 & 0 & 0 \\ 0 & 0.0001 & -2 & 0 & 0 \\ 0 & 0 & 0.0001 & -1 & 0 \\ 0 & 0 & 0 & 0.0001 & 0 \end{pmatrix}, \; B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

$$S = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{10, 12, 24, 29, 30\}.$$

Then $K = (-115, 4.887 \cdot 10^7, -9.4578 \cdot 10^{12}, 8.1915 \cdot 10^{17}, -2.5056 \cdot 10^{22})$

- **The eigenvalue assignment problem with the above data is very ill-conditioned as the following computation shows.**
- Change $a_{51} \longrightarrow 10^{-6}$.
- The closed-loop eigenvalues: $\pm 1.5829 \times 10^8, -3, -2, -1$. **Completely Different.**

Explanation

- **Ill-conditioning of the feedback vector:** Let $\hat{K}$ be obtained by changing the first entry of $\hat{K}$ -114.999 and leaving the remaining entries unchanged.

  The eigenvalues of $(A - B\hat{K})$ are:
  $\{29.5386 \pm 0.4856j,\ 23.9189, 12.0045, 9.9984\}$.

  So, **the problem of computing the feedback vector K is ill-conditioned.**
- **Nearness to Uncontrollability** (Indicated by the smallness of the subdiagonal entries of A).
- **The open-loop eigenvalues are well-separated from those of the closed-loop eigenvalues.**
- **Ill-conditioning of the eigenvector matrix:** $Cond_2(X) = 1.3511 \times 10^{24}$. (**Large.**)

Note: The feedback vector K was computed using the Recursive Algorithm. *The MATLAB function* ***place*** *cannot place the eigenvalues.*

Robust Eigenvalue Assignment

- Find K such that the Closed-loop eigenvalues are **as insensitive as possible** due to small perturbations in the data.

  Solution Idea: Choose the eigenvector matrix X such that is **as well-conditioned as possible.**

- MATLAB Function: **place**. (Kautsky-Nichols-Van Dooren Algorithm)

- **Idea behind "place".**
- Factorize $B = [U_0, U_1] \begin{bmatrix} Z \\ 0 \end{bmatrix}$.
- Choose the vectors of the matrix X from the orthonormal bases of the subspaces:

$$s_j = N\{U_1^T(A - \lambda_j I)\}$$

and $\hat{s}_j =$ Complement of s_j.

- $K = Z^{-1}U_0^T(A - X\Lambda X^{-1})$
  $\Lambda =$ diag $(\lambda_1, ..., \lambda_n)$.
- For details, see **Algorithm 11.6.1**

Recommendations

A. **For single-input problem**

• Try *Recursive Algorithm* (**Algorithm 11.2.1**) first. In case of possible ill-conditioning, use its RQ **version (Algorithm 11.2.3)**

B. **For Multi-input problem**

• Try the multi-input version of the recursive Algorithm (**Algorithm 11.3.1**) first. If the accuracy is not good, use the **Explicit** QR **Algorithm (Section 11.3.1)**.

POLERCS - Single-input pole placement using the recursive algorithm

POLEQRS - Single-input pole placement using the QR version of the recursive algorithm

POLERQS - Single-input pole placement using RQ version of the recursive algorithm

POLERCM - Multi-input pole placement using the recursive algorithm

POLERCX - Multi-input placement using the modified recursive algorithm that avoids complex arithmetic and complex feedback.

POLEQRM - Multi-input pole placement using the explicit QR algorithm

POLESCH - Multi-input pole placement using the Schur decomposition

POLEROB - Robust pole placement

- The recursive algorithm is implemented as `StateFeedbackGains` [*system, poles,* `Method` → `Recursive`].
- The explicit QR algorithm is implemented as `StateFeedbackGains` [*system, poles,* `Method` → `QR Decomposition`].
- The Schur method is implemented as `StateFeedbackGains` [*system, poles,* `Method` → `Schur Decomposition`].
- The RQ implementation of the recursive single-input algorithm is implemented as `StateFeedbackGains` [*system, poles,* `Method` → `RecursiveRQDecomposition`].
- The implicit single-input RQ algorithm is implemented as `StateFeedbackGains` [*system, poles,* `Method` → `ImplicitRQDecomposition`].

SLICOT

Eigenvalue/Eigenvector Assignment

SB01BD	Pole assignment for a given matrix pair (A, B)
SB01DD	Eigenstructure assignment for a controllable matrix pair (A, B) in orthogonal canonical form
SB01MD	State feedback matrix of a time-invariant single-input system

POLEPACK

A collection of MATLAB programs for eigenvalue assignment, developed by G.S. Miminis **(1991)**. Available on **NETLIB**.

Partial Eigenvalue Assignment Problem (PEVAP).

Given

- A part of the spectrum $\{\lambda_1, \ldots, \lambda_p\}$, $p << n$.
- A self-conjugate set $\{\mu_1, \ldots, \mu_p\}$.
- A control matrix B
  Find F such that

$$\Omega(A - BF) = \{\mu_1, \ldots, \mu_p; \underbrace{\lambda_{p+1}, \ldots, \lambda_n}_{\text{No Change}}\}$$

- More practical, especially for large and sparse systems.

Challenges

- Solve $PEVAP$ by knowing only the first p eigenvalues and the corresponding eigenvectors.
- Prove the invariance of the large numbers $(n - p)$ eigenvalues by mathematical results.

  (*Not possible to compute all eigenvalues in practice, if A is very large and sparse*).

A Parametric Solution of PEVAP

- $\Lambda_1 =$ diag $(\lambda_1, ..., \lambda_p)$
- $Y_1 = (y_1, ..., y_p)$ **(The left eigenvector matrix)**
- $\Lambda_{cl} =$ diag $(\mu_1, ..., \mu_p)$.
- $\Gamma =$ An arbitrary parametric matrix
- **Assumptions** (i) (A, B) is partially controllable (ii) The set $\{\lambda_1, \ldots, \lambda_p\}, \{\lambda_{p+1}, \ldots, \lambda_{2n}\}$ and $\{\mu_1, \ldots, \mu_p\}$. are mutually disjoint.

- $F = \Phi Y_1^H$ solves PEVAP.
- ϕ is the solution of the $p \times p$ linear system: $\phi Z_1 = \Gamma$.
- Z_1 satisfies the $p \times p$ Sylvester equation $\Lambda_1 Z_1 - Z_1 \Lambda_{cl} = Y_1^H B \Gamma$.

Algorithm: A Parametric Algorithm for Partial Eigenvalue Assignment Problem (Algorithm 11.3.4)

Inputs:

- The matrices A and B.
- The self-conjugate subset $\{\lambda_1, \ldots, \lambda_p\}$ of the spectrum $\{\lambda_1, \ldots, \lambda_n\}$ of the matrix A.
- The left eigenvectors $\{y_1, ..., y_p\}$.
- A self-conjugate set of numbers $\{\mu_1, ..., \mu_p\}$.

Output:

- The real feedback matrix F such that the spectrum of the closed-loop matrix $A - BF$ is $\{\mu_1, \ldots, \mu_p; \lambda_{p+1}, \ldots, \lambda_n\}$.

Assumptions:

- The matrix pair (A, B) is *partially controllable* with respect to the eigenvalues $\lambda_1, \ldots, \lambda_p$.
- The sets $\{\lambda_1, ..., \lambda_p\}$, $\{\mu_1, ..., \mu_p\}$ and $\{\lambda_{p+1}, ..., \lambda_{2n}\}$ are disjoint.

Step 1. Form

$$\Lambda_1 = \text{diag}(\lambda_1, \ldots, \lambda_p),\ Y_1 = (y_1, \ldots, y_p),$$
$$\text{and } \Lambda_{c1} = \text{diag}(\mu_1, \ldots, \mu_p).$$

Step 2. Choose arbitrary $m \times 1$ vectors $\gamma_1, \ldots, \gamma_p$ in such a way that $\overline{\mu_j} = \mu_k$ implies $\overline{\gamma_j} = \gamma_k$ and form $\Gamma = (\gamma_1, \ldots, \gamma_p)$.

Step 3.
Find the unique solution Z_1 of the Sylvester equation

$$\Lambda_1 Z_1 - Z_1 \Lambda_{c1} = Y_1^H B \Gamma\,.$$

If Z_1 is ill-conditioned, then return to Step 2 and select different $\gamma_1, \ldots, \gamma_p$.

Step 4. Solve $\Phi Z_1 = \Gamma$ for Φ.

Step 5. Form $F = \Phi Y_1^H$.

Computational Requirements and Features

- Knowledge of only partial spectrum and eigenvectors.
- Solution of a $p \times p$ small Sylvester equation.
- Solution of a small $p \times p$ linear algebraic system.
- Mathematical results guarantee that the $(n-p)$ eigenvalues remain unchanged.
- Parametric nature is useful for robust partial eigenvalue assignment.

State Estimation, Kalman Filter and LQG Design

(Chapter 12)

State Estimation Problem

Estimate the state vector $x(t)$, knowing the input $u(t)$, the output vector $y(t)$, and the matrices A, B, and C.

- The estimate is denoted by $\hat{x}(t)$.
- The error vector $e(t) = x(t) - \hat{x}(t)$.

Two Common Approaches for State-Estimation

A. Eigenvalue Assignment Approach

- Find a feedback matrix K such that $(A - KC)$ is stable.
- Compute $\hat{x}$ by solving the system of differential equations:

$$(\hat{x})^T = (A - KC)\hat{x}(t) + Ky(t) + Bu(t).$$

(**Choosing the initial condition arbitrarily**)

- **Error**: $e(t) = (A - KC)e(t)$

$e(t) \to 0$ as $t \to \infty$.

B. **Sylvester-Equation Approach**

- Solve the **Sylvester-observer Equation** for a non-singular solution X:

$$XA - FX = GC.$$

- Choose F stable (The eigenvalues having negative real parts)
- Construct the observer: $\dot{z}(t) = Fz(t) + Gy(t) + XBu(t)$.
- The estimate $\hat{x}(t) = X^{-1}z(t)$
- **Error** $e(t) = z(t) - Xx(t) \longrightarrow 0$ for any $x(0), z(0)$, and $u(t)$.

The Sylvester-Observer Equation

$$XA - FX = GC$$

- A, C - Given
- X, F, G - To be chosen.

The Classical Sylvester Equation

$$XA - FX = C$$

- A, C, and F Given
- X needs to be found.

Algorithm: Reduced-order Observer Design via Sylvester-Observer Equation

Inputs: *The matrices $A, B,$ and C of order $n \times n, n \times m,$ and $r \times n$, respectively.*

Output: *An estimate $\hat{x}$ of the state vector x.*

Assumptions: *(i) (A, C) is observable. (ii) C is of full rank.*

Step 1. *Choose an $(n-r) \times (n-r)$* ***stable*** *matrix F.*

Step 2. *Solve the reduced-order* **Sylvester-observer equation**

$$XA - FX = GC,$$

choosing G such that (F, G) is controllable.

Step 3. *Compute $P = XB$*

- X is of order $(n-r) \times n$
- F is of order $(n-r) \times n$
- G is of order $(n-r) \times r$

Step 4. Construct the **reduced-order observer**:

$$\dot{z} = Fz + Gy + Pu$$

Step 5. *Compute the*

$$\hat{x} = \begin{pmatrix} C \\ X \end{pmatrix}^{-1} \begin{pmatrix} y \\ z \end{pmatrix}.$$

Example (Helicopter Problem)

$$A = \begin{pmatrix} -0.02 & 0.005 & 2.4 & -32 \\ -0.14 & 0.44 & -1.3 & -30 \\ 0 & 0.018 & -1.6 & 1.2 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \; B = \begin{pmatrix} 0.14 & -0.12 \\ 0.36 & -8.6 \\ 0.35 & 0.009 \\ 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 57.3 \end{pmatrix}.$$

- $Rank\,(C) = 2, \; r = 2.$

Step 1. *Choose* $F = \begin{pmatrix} -1 & 0 \\ 0 & -2 \end{pmatrix}$.

Step 2. *Choose* $G = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$.

$$X = \begin{pmatrix} -0.117 & -0.0822 & 62.1322 & 37.2007 \\ -0.1364 & -1.9296 & 428.2711 & -173.4895 \end{pmatrix}.$$

Step 3. $P = XB = \begin{pmatrix} 21.7151 & 1.2672 \\ 149.1811 & 20.4653 \end{pmatrix}$.

Step 4.

$$\hat{x} = \begin{pmatrix} C \\ X \end{pmatrix}^{-1} \begin{pmatrix} y \\ z \end{pmatrix}$$

$$= \begin{pmatrix} -24.5513 & -135.1240 & 124.1400 & -18.0098 \\ 1 & 0 & 0 & 0 \\ -0.0033 & -0.0360 & 0.0395 & -0.0034 \\ 0 & 0.0175 & 0 & 0 \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix}$$

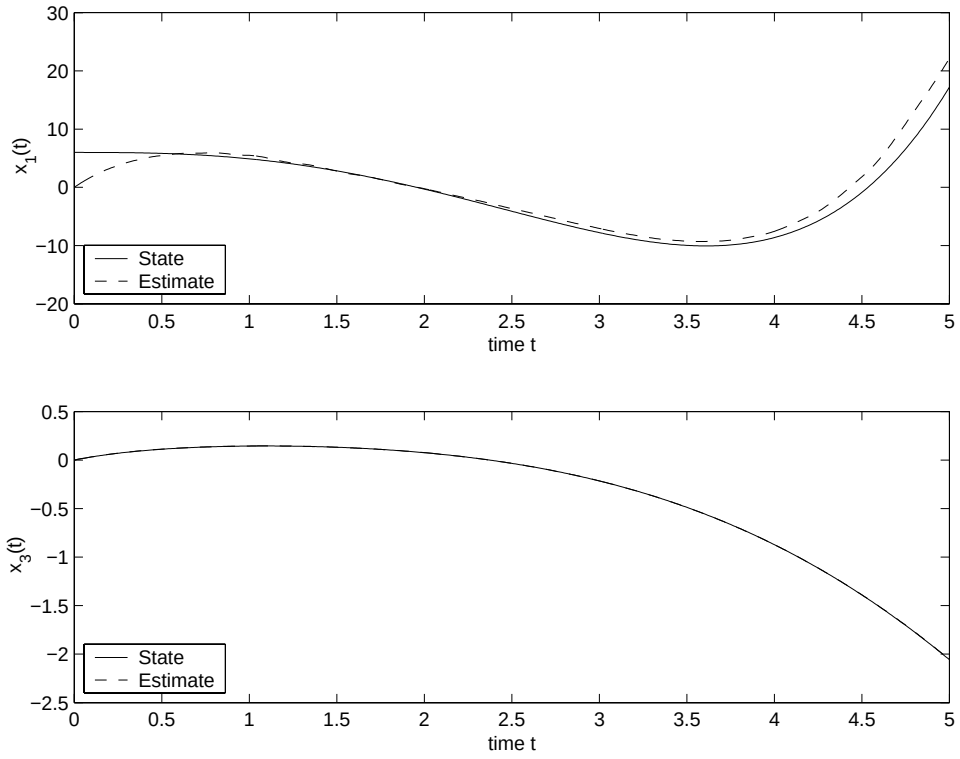
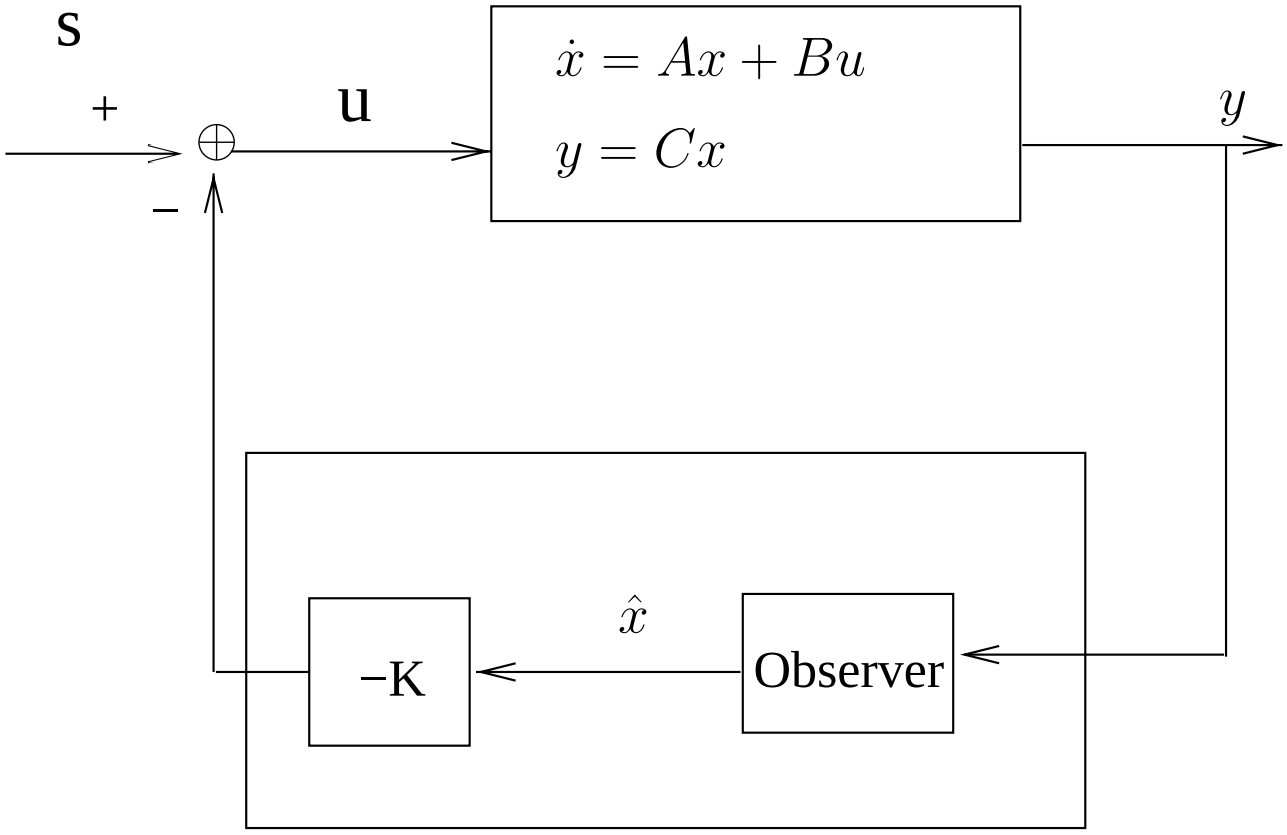


Figure 12.3. The First and Third Variables of the State $x(t)$ and the Estimate $\hat{x}(t)$, for the Helicopter Example.

- Pretty good estimates
- The third variables are **indistinguishable**.

State Feedback Control Vector

$$u(t) = K\hat{x}(t)$$

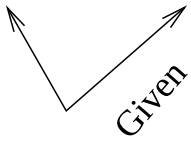


Separation Property

- The Observer design and feedback design can be carried out independently, and the *calculation of the feedback gain is not affected whether the true state* x *or the estimated state* $\hat{x}$ *is used.*

Numerical Solution of the Sylvester-Observer Equation:

$$X A - F X = G C$$



A Template

Step 1. **Reduce the pair (A, C) to Observer-Hessenberg pair $(H, \overline{C})$:**

$$\begin{aligned} OAO^T &= H, \text{ an unreduced block upper-Hessenberg matrix} \\ CO^T &= \bar{C} = (0, C_1) \end{aligned}$$

Step 2. **Solve the reduced Hessenberg Sylvester-observer equation**:

$$YH - FY = G\bar{C},$$

Step 3. **Recover the solution X of the original problem from the solution of the reduced problem:**

$$X = YO$$

An Observer-Hessenberg Method for $YH - FY = G\overline{C}$ (Van Dooren (1984))

- Set $q = n - r$. Set

$$Y = \begin{pmatrix} 1 & y_{12} & \cdots & & \cdots & y_{1,n} \\ & \ddots & \ddots & & & \vdots \\ & & & & & \\ 0 & & 1 & y_{q,q+1} & \cdots & y_{q,n} \end{pmatrix}$$

$$F = \begin{pmatrix} f_{11} & 0 & \cdots & \cdots & 0 \\ f_{21} & f_{22} & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ f_{q1} & \cdots & \cdots & \cdots & f_{qq} \end{pmatrix}, \; G = \begin{pmatrix} g_1^T \\ g_2^T \\ \cdots \\ g_9^T \end{pmatrix}.$$

where the diagonal entries f_{ii}, $i = 1, \ldots, q$ are preassigned.

- Thus, the off diagonal entries of F, Y, and all the entries of G are to be computed.

Computation of $Y, F,$ and G $n = 3, r = 1$

$$\begin{pmatrix} 1 & y_{12} & y_{13} \\ 0 & 1 & y_{23} \end{pmatrix} \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & h_{32} & h_{33} \end{pmatrix} - \begin{pmatrix} f_{11} & 0 \\ f_{21} & f_{22} \end{pmatrix} \begin{pmatrix} 1 & y_{12} & y_{13} \\ 0 & 1 & y_{23} \end{pmatrix} = \underbrace{\begin{pmatrix} g_{11} \\ g_{21} \end{pmatrix}}_{G} \underbrace{\begin{pmatrix} 0 & 0 & c_1 \end{pmatrix}}_{\bar{C}}.$$

- First row of Y and first row of G:

$$(y_{12}, y_{13}, g_{11}) \begin{pmatrix} h_{21} & h_{22} - f_{11} & h_{23} \\ 0 & h_{32} & h_{33} - f_{11} \\ 0 & 0 & -c_1 \end{pmatrix} = \begin{pmatrix} f_{11} - h_{11} \\ -h_{12} \\ -h_{13} \end{pmatrix}^T$$

- Second row of F, second row of Y and second row of G:

$$(f_{21}, y_{23}, g_{21}) \begin{pmatrix} -1 & -y_{12} & -y_{13} \\ 0 & h_{32} & h_{33} - f_{22} \\ 0 & 0 & -c_1 \end{pmatrix} = \begin{pmatrix} -h_{21} \\ f_{22} - h_{22} \\ -h_{23} \end{pmatrix}^T.$$

Algorithm: An Algorithm for the Multi-Output Sylvester-Observer Equation (Algorithm 12.7.1)

Step 0. Set $n - r = q$.

Step 1. Transform the pair (A, C) to the observer–Hessenberg pair $(H, \bar{C})$:

$$\begin{aligned} OAO^T &= H, \text{ an unreduced block upper Hessenberg matrix} \\ CO^T &= \overline{C} \end{aligned}$$

Step 2. Set $F_{q\times q}$ lower triangular: only off- **diagonal entries to be found**.

Step 3. Solve for Y:

$$YH - FY = G\bar{C},$$

where Y has the form as above, exploiting the structure of Y and F.

Step 4. Recover X from Y:

$$X = YO.$$

Example *Consider the Helicopter Example again.*

Here $n = 4, r = 2$.

Step 1. The observer-Hessenberg pair of (A, C):

$$H = \begin{pmatrix} -0.0200 & 2.4000 & 0.0050 & -32.0000 \\ 0 & -1.6 & 0.0180 & 1.200 \\ -0.1400 & -1.3000 & 0.4400 & -30.0000 \\ 0 & 1 & 0 & 0 \end{pmatrix},$$

$$\bar{C} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 57.3 \end{pmatrix}.$$

Step 2. Set $f_{11} = -1,\ f_{22} = -2$

Step 3. $y_1 = (0, 7, 6.7),\ g_1 = (10.085, -4.1065)$.
1st row of $y = (1,\ 0,\ 7,\ 6.7)$.

$f_{21} = 0.0007$
$y_2 = (-0.0053, -0.4068),$

$g_2 = (0, 0.0094).$

$$F = \begin{pmatrix} -1 & 0 \\ 0.0007 & -2 \end{pmatrix},\ G = \begin{pmatrix} 10.085 & -4.1065 \\ 0 & 0.0094 \end{pmatrix}.$$

The second row of $Y = (0, 1, -0.0053, -0.4068)$.
Therefore,

$$Y = \begin{pmatrix} 1 & 0 & 7 & 6.7 \\ 0 & 1 & -0.0053 & -0.4068 \end{pmatrix}.$$

Step 4. Recover X from

$$Y : X = YO = \begin{pmatrix} 1 & 7 & 0 & 6.7 \\ 0 & -0.0053 & 1 & -0.4068 \end{pmatrix}.$$

Verify: $\| XA - FX - GC \|_2 = O(10^{-14})$

Flop-count:
Solving for F, G, and Y (exploiting the special structures of these matrices): $\boxed{2(n-r)rn^2 \text{ flops.}}$

- **A block generalization of the Algorithm exists**.

Ref: **J. Carvalho and B.N. Datta**
A block algorithm for the sylvester-observation equation arising in state estimation, Proc. IEEE Conf. Dec. Control, Orlando, Florida, 2001.

MATCONTROL Function: SYLVOBSM.

This block algorithm is suitable for high performance computing and is also more efficient in sequential computing.

Comparison of the Actual and Estimation States

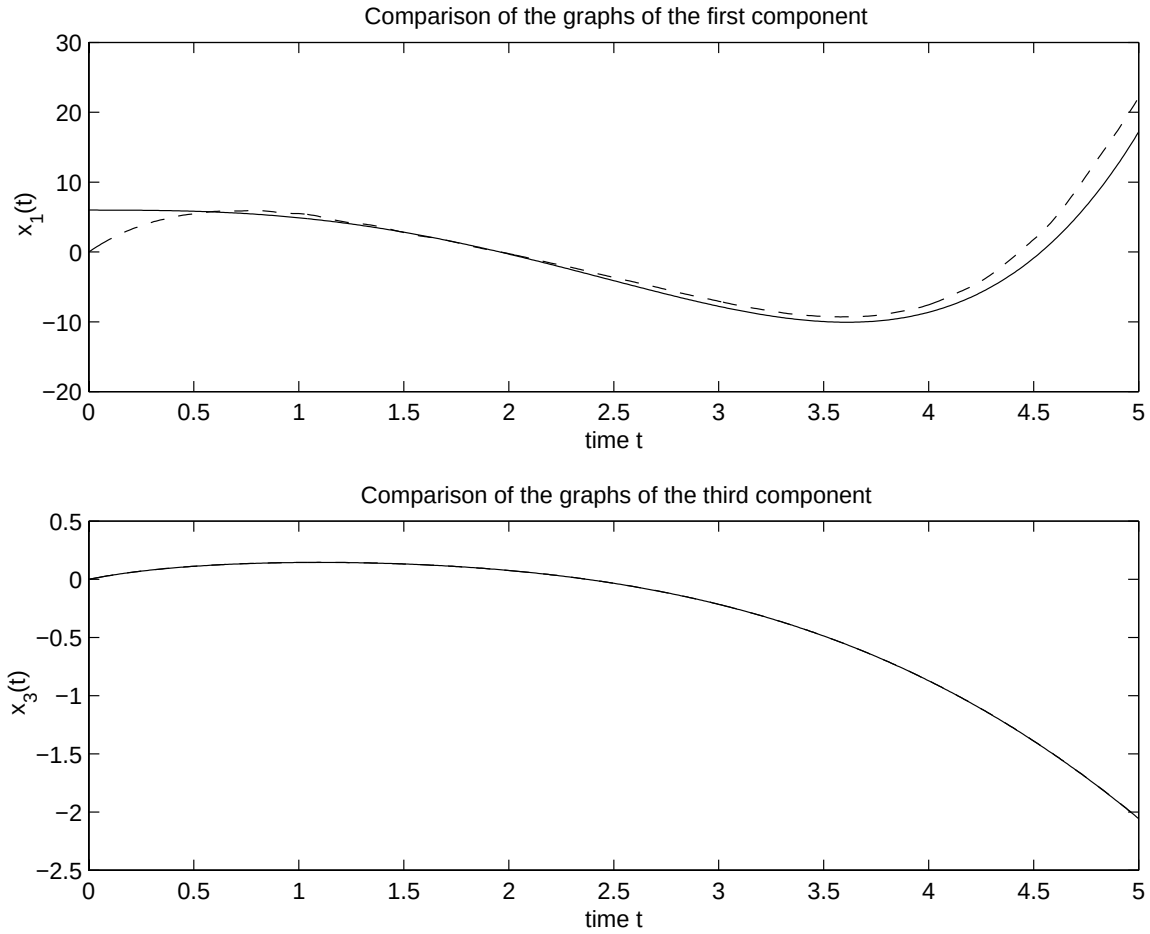


Figure: Comparisons of the First and Third Components of the Exact State $x(t)$ and the Estimated State $\hat{x}(t)$, obtained by Algorithm 12.7.1.

MATCONTROL Functions

SYLVOBSM - Block triangular algorithm for Sylvester-observer equation.

SYLVOBSM - Solving the multi-output Sylvester-observer equation.

SYLVOBSC - Solving the constrained Sylvester-observer equation.

ADVANCED NUMERICAL METHODS Functions

Design of reduced-order state estimator (observer)

- The reduced-order state estimator using pole assignment approach is computed by `ReducedOrderEstimator` [*system, poles*].
- The reduced-order state estimator via solution of the Sylvester-observer equation using recursive bidiagonal scheme is computed by `ReducedOrderEstimator` [*system, poles*, `Method` → `RecursiveBidiagonal`] and `ReducedOrderEstimator` [*system, poles*, `Method` → `RecursiveBlockBidiagonal`] (block version of the recursive bidiagonal scheme).
- The reduced-order state estimator via solution of the Sylvester-observer equation using recursive triangular scheme is computed by `ReducedOrderEstimator` [*system, poles*, `Method` → `RecursiveTriangular`] and `ReducedOrderEstimator` [*system, poles*, `Method` → `RecursiveBlockTriangular`] (block version of the recursive triangular scheme).

Optimal State Estimation: The Kalman Filter

Consider the stochastic system

$$\dot{x}(t) = Ax(t) + Bu(t) + Fw(t)$$
$$y(t) = Cx(t) + v(t),$$

- $w(t)$ - Noise in the Input.
- $v(t)$ - Noise in the Output.

Find $\hat{x}(t)$ Such that

$$E[\| x(t) - \hat{x}(t) \|^2], \text{ is } \textbf{minimized}$$

as $t \to \infty$.

Assumptions

1. The system is **controllable** and **observable**.

2. Both w and v are white noise, **zero-mean** stochastic processes.

3. The noise processes w and v are **uncorrelated** with one another; that is,

$$Ew(t)v^T(s) = 0.$$

4. The initial state x_0 is a **Gaussian zero-mean** random variable with known covariance matrix; and uncorrelated with w and v. That is,

$$E[x_0] = 0$$

$$E[x_0 x_0^T] = S,\ E[x_0 w^T(t)] = 0,\ E[x_0 v^T(t)] = 0,$$

where S is the **positive semidefinite covariance matrix**.

Theorem (Kalman Filter). *Under the above assumptions, the best estimate $\hat{x}(t)$ (in the linear least-mean-square sense) can be generated by the* **Kalman filter:**

$$\dot{\hat{x}}(t) = (A - K_f C)\hat{x}(t) + Bu(t) + K_f y(t)$$

- $K_f = X_f C^T V^{-1}$ **(Filter-Gain)**
- X_f is the symmetric positive definite solution of the **Continuous-time Filter Algebraic Riccati Equation** (CFARE):

$$AX + XA^T - XC^T V^{-1} CX + FWF^T = 0.$$

■

Note: CFARE is dual to CARE.

Algorithm: The State Estimation of the Stochastic System using Kalman Filter (Algorithm 12.9.1)

Step 1. Obtain the **unique symmetric positive definite solution X_f of the CFARE:**

$$AX_f + X_f A^T - X_f C^T V^{-1} C X_f + FWF^T = 0.$$

Step 2. Form the **filter gain** matrix $K_f = X_f C^T V^{-1}$.

Step 3. Obtain the **estimate** $\hat{x}(t)$ by solving the Kalman filter.

Duality Between Kalman Filter and the LQR Problems

1. **Guaranteed Stability**

   The filter matrix $A - K_f C$ is stable; that is, $Re\lambda_i(A - K_f C) < 0;\ i = 1, 2, \cdots, n$, where $\lambda_i, i = 1, \cdots, n$, are the eigenvalues of $A - K_f C$.

2. **Guaranteed Robustness**

   Define

   Kalman Filter Transfer: $G_{KF}(s) \equiv C(sI - A)^{-1}K_f$,

   Open-Loop Transfer: $G_{FOL}(s) \equiv C(sI - A)^{-1}F$.

$$\sigma_{\max}(I + G_{KF}(s))^{-1} \leq 1$$

and

$$\sigma_{min}(I + G_{KF}^{-1}(s)) \geq \frac{1}{2}.$$

MATLAB Note. The MATLAB function **kalman** designs a Kalman state estimator given the state-space model and the process and noise covariance data. **kalman** is available in MATLAB **control system toolbox**.

Example *Consider the stochastic system:*

$$\dot{x}(t) = Ax(t) + Bu(t) + w(t)$$

$$y(t) = Cx(t) + v(t)$$

with $A, B,$ *and* C *as in Helicopter Example.*

Take $W = BB^T$, $V = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $F = I_{4\times 4}$

Step 1. *The symmetric positive definite solution* X_f *of the CFARE*

$$AX + XA^T - XC^TV^{-1}CX + FWF^T = 0$$

is

$$X_f = \begin{pmatrix} 8.3615 & 0.0158 & 0.0187 & -0.0042 \\ 0.0158 & 9.0660 & 0.0091 & -0.0031 \\ 0.0187 & 0.0091 & 0.0250 & 0.0040 \\ -0.0042 & -0.0031 & 0.0040 & 0.0016 \end{pmatrix}$$

Step 2. *The filter gain matrix* $K_f = X_f C^T V^{-1}$ *is*

$$K_f = \begin{pmatrix} 0.0158 & -0.2405 \\ 9.0660 & -0.1761 \\ 0.0091 & 0.2289 \\ -0.0031 & 0.0893 \end{pmatrix}$$

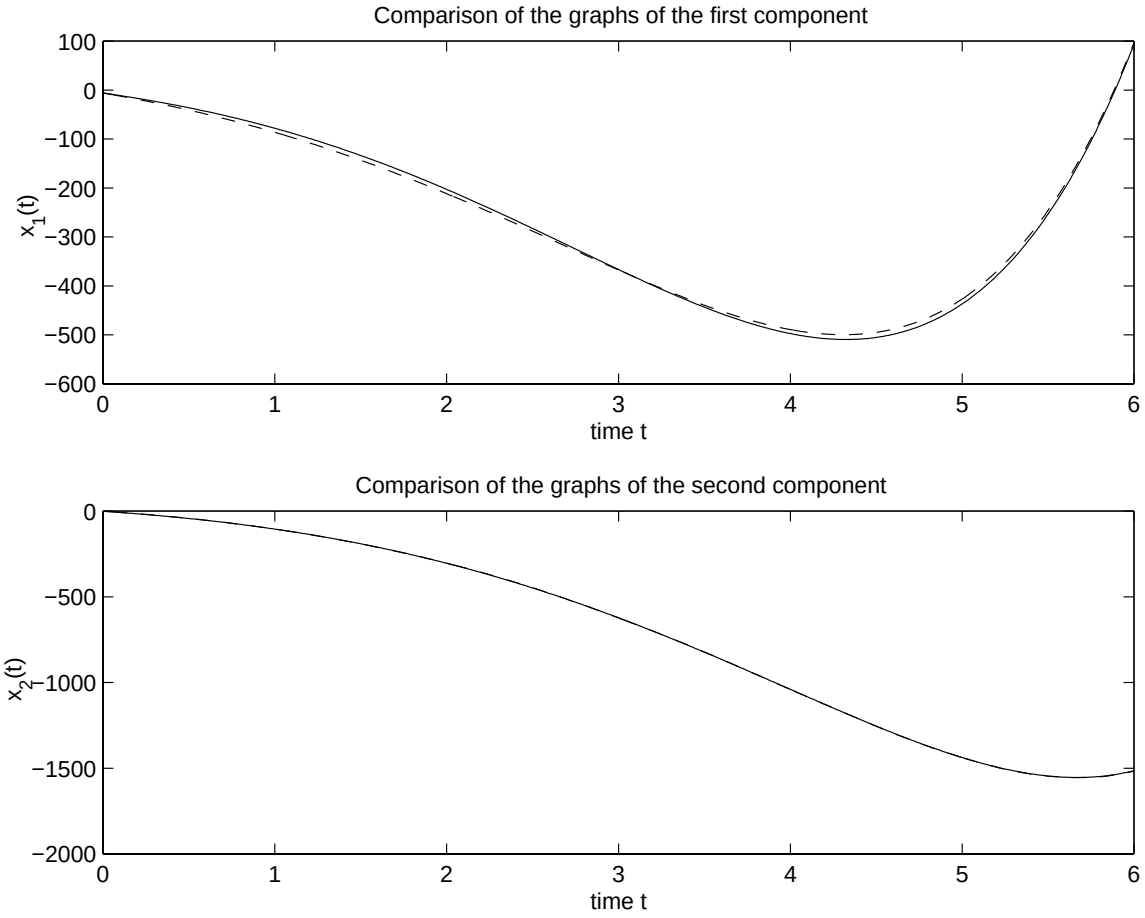
- *The optimal state estimator of* $\hat{x}(t)$:

$$\dot{\hat{x}}(t) = (A - K_f C)\hat{x}(t) + Bu(t) + K_f y(t).$$

- *The filter eigenvalues (The eigenvalues of* $A - K_f C$*)*:

$$\{-0.0196, -8.6168, -3.3643 \pm j2.9742\}.$$

Comparison of the Exact State and the Estimated State by Kalman Filters



Comparisons of the First and Second Components of the Exact State $x(t)$ and the Estimated State $\hat{x}(t)$ obtained by Kalman Filter.

- Pretty good estimates
- The second components are indistinguishable.

Linear Quadratic Gaussian Problem (LQG: Continuous-time)

Given

- The controllable and observable stochastic system:

$$\dot{x}(t) = Ax(t) + Bu(t) + Fw(t)$$

$$y(t) = Cx(t) + v(t)$$

- The weighting matrices $Q \geq 0, \quad R > 0$.
- The noises $w(t)$ and $v(t)$ **Gaussian, white, zero-mean, and stationary process**
- Covariant matrices $W = W^T \geq 0, \quad V = V^T > 0$.

Find the **optimal control** $u(t)$ minimizing

$$J_{QG} = \lim_{T\to\infty} \frac{1}{2T} E\left[\int_{-T}^{T} (x^T Q x + u^T R u) dt\right].$$

Solution of the LQG Problem

- LQG solution $\equiv$ Deterministic LQR + Kalman Filter
- **The optimal control vector** $u(t)$ for the LQG problem:

$$u(t) = -K_c \hat{x}(t)$$

(i) K_c is obtained by LQR

$$\boxed{K_c = R^{-1} B^T X_c,}$$

X_c stabilizing solutions of the CARE

$$X_c A + A^T X_c + Q - X_c B R^{-1} B^T X_c = 0$$

(ii) The vector $\hat{x}(t)$ is generated by the **Kalman filter**:

$$\boxed{\dot{\hat{x}}(t) = (A - K_f C)\hat{x}(t) + Bu(t) + K_f y(t).}$$

$K_f =$ **Filter Gain Matrix** $X_f C^T V^{-1}$

$X_f =$ **Solution of the continuous-time Filter Algebraic Riccati Equation**

$$AX_f + X_f A^T - X_f C^T V^{-1} C X_f + FWF^T = 0.$$

- The minimum value of the performance measure J_{QG}:

$$J^*_{QG} = trace(X_c K_f V K_f^T) + trace(X_f Q),$$

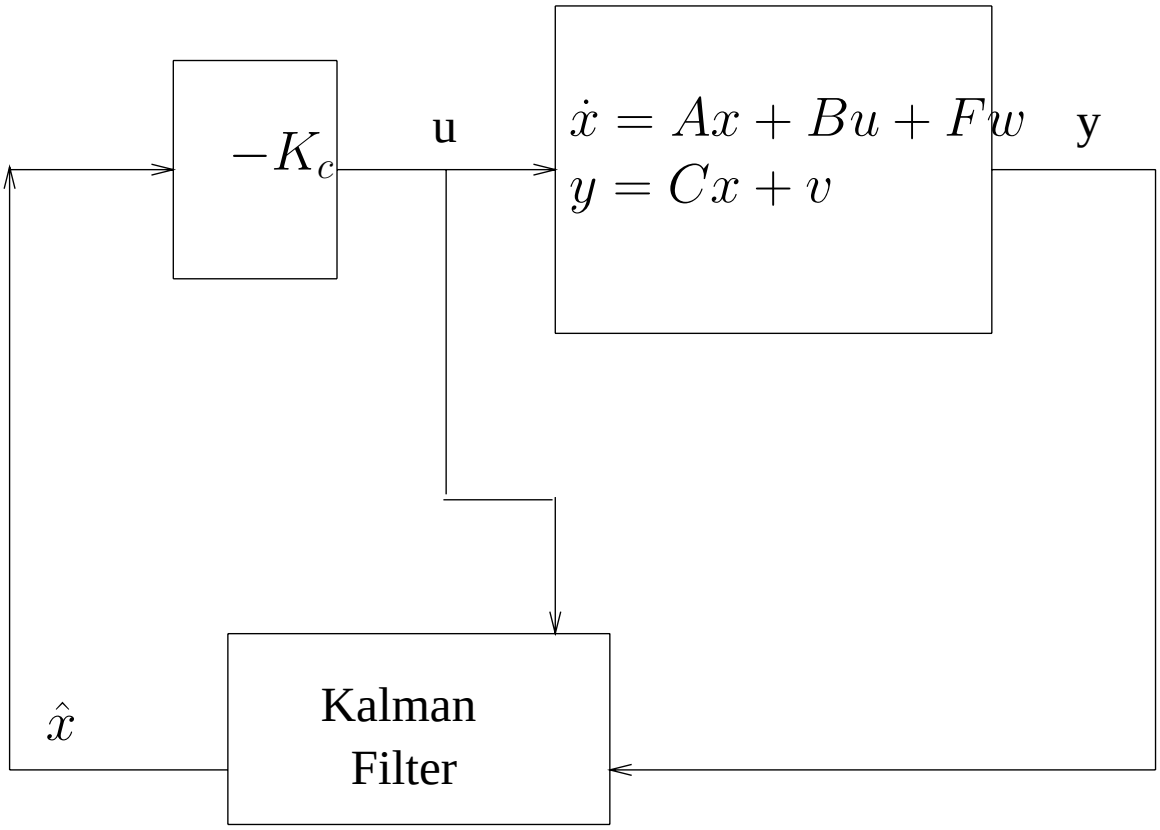


Figure 12.2: The LQG Design via Kalman Filter

Algorithm: The Continuous-time LQG Design Method (Algorithm 12.10.1)

Step 1. (LQR) Obtain the symmetric positive definite stabilizing solution X_c of the CARE:

$$XA + A^T X - XBR^{-1}B^T X + Q = 0.$$

Step 2. (LQR) Compute $K_c = R^{-1}B^T X_c$

Step 3. (Kalman Filter)

3.1. Solve the CFARE:

$$AX + XA^T - XC^T V^{-1} CX + FWF^T$$

to obtain the symmetric positive definite stabilizing solution X_f.

3.2. Compute filter gain matrix

$$K_f = X_f C^T V^{-1}$$

Step 4. (Kalman Filter)

$$\dot{\hat{x}}(t) = (A - BK_c - K_f C)\hat{x}(t) + K_f y(t)$$

Step 5. Determine the LQG control law

$$u(t) = K_c \hat{x}(t)$$

Example: *Consider the LQG design for the helicopter problem with*

$$Q = C^T C, \text{ and } R = I_{2\times 2},$$

and the same W *and* V.

Step 1. *The stabilizing solution* X_c *of the CARE*

$$X_c = \begin{pmatrix} 0.0071 & -0.0021 & -0.0102 & -0.0788 \\ -0.0021 & 0.1223 & 0.0099 & -0.1941 \\ -0.0102 & 0.0099 & 41.8284 & 174.2 \\ -0.0788 & -0.1941 & 174.2 & 1120.9 \end{pmatrix}$$

Step 2. *The control gain matrix* K_c

$$K_c = R^{-1}B^T X_c = \begin{pmatrix} -0.0033 & 0.0472 & 14.6421 & 60.8894 \\ 0.0171 & -1.0515 & 0.2927 & 3.2469 \end{pmatrix}$$

Step 3. *The filter gain matrix* K_f

$$K_f = \begin{pmatrix} 0.0158 & -0.2405 \\ 9.0660 & -0.1761 \\ 0.0091 & 0.2289 \\ -0.0031 & 0.0893 \end{pmatrix}.$$

- The close-loop eigenvalues:

$$\underbrace{\{-3.3643 \pm 2.9742j, -0.0196, -8.6168\}}_{\text{Controller Eigenvalues}} \cup \underbrace{\{-0.0196, -8.6168, -3.3643 \pm 2.9742j\}}_{\text{Filter Eigenvalues}}$$

- The minimum Value of J_{QG}: $J^*_{QG} = 42.5327$.

Selected Software

MATLAB CONTROL SYSTEM TOOLBOX

LQG design tools

- **kalman** - Kalman estimator

dkalman - Discrete Kalman estimator for continuous plant

lqgreg - Form LQG regulator given LQ gain and Kalman estimator

SLICOT

FB01RD	Time-invariant square root covariance filter (Hessenberg form)
FB01TD	Time-invariant square root information filter (Hessenberg form)
FB01VD	One recursion of the conventional Kalman filter
FD01AD	Fast recursive least-squares filter.

Internal Balancing and Model Reduction (CHAPTER 14)

Problem: Construct a **Reduced-order Model** (ROM) such that ROM is close, in some sense, to the original full-order model.

Measure of Closeness

- *Minimize* $||G(s) - G_R(s)||$.

Two Norms: (i) H_∞**– Norm**
(ii) **Hankel Norm**

Model Reduction Problem

Given the transfer function

$$G(S) = C(SI - A)^{-1}B + D,$$

Find a reduced-order transfer function

$$G_R(S) = C_R(SI - A_R)^{-1}B_R + D_R$$

Such that

$$||G(S) - G_R(S)||_\infty$$

is minimized.

Balanced Truncation Method

- *Internal Balancing + Truncation*

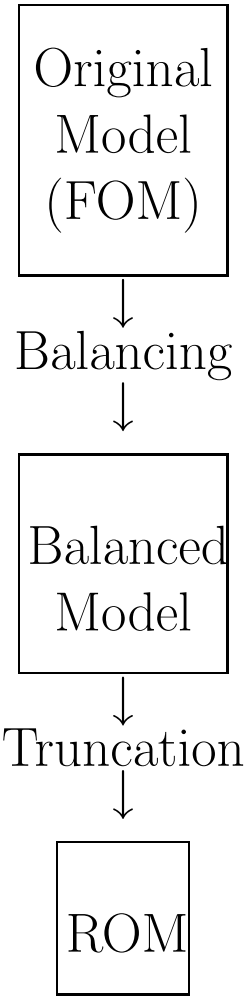


Figure: *Balanced Truncation*

Internal Balancing

Idea: Given a **Stable System** construct a transforming matrix T such that *the controllability and observability Gramminas are the same and is equal to a diagonal matrix:*

$$T^{-1}C_G T^T = T^T O_G T = \Sigma \text{ (\textbf{diagonal})}.$$

$\Sigma =$ diag $(\sigma_1, \ldots, \sigma_n)$

- **Hankel Singular Values:** $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$
- Balanced system $(\tilde{A}, \tilde{B}, \tilde{C})$:

$$\tilde{A} = T^{-1}AT$$

$$\tilde{B} = T^{-1}B$$

$$\tilde{C} = CT$$

An Algorithm for Internal Balancing (Algorithm 14.2.1)

Step 1. Compute the Controllability and Observability Grammians:

$$AC_G + C_G A^T + BB^T = 0$$

$$A^T O_G + O_G A + C^T C = 0$$

Step 2. Compute the Cholesky factors: (**Assumptions**: (A, C) is **observable** and (A, B) is **controllable**).

$$C_G = L_c L_c^T$$

$$O_G = L_o L_o^T$$

Step 3. Find SVD of $L_o^T L_c = U\Sigma V^T$.

Step 4. Compute the transforming matrix:

$$T = L_c V \Sigma^{-\frac{1}{2}}$$

Step 5. Compute the balanced realization $(\tilde{A}, \tilde{B}, \tilde{C})$:

$$\begin{aligned} \tilde{A} &= T^{-1} A T \\ \tilde{B} &= T^{-1} B \\ \tilde{C} &= CT. \end{aligned}$$

Example 14.2.1 Consider finding the balanced realization using Algorithm **14.2.1** of the system (A, B, C) given by:

$$A = \begin{pmatrix} -1 & 2 & 3 \\ 0 & -2 & 1 \\ 0 & 0 & -3 \end{pmatrix}, \; B = (1,1,1)^T, \; C = (1,1,1).$$

Step 1. By solving the Lyapunov equation for C_{G2}, we obtain

$$C_G = \begin{pmatrix} 3.9250 & 0.9750 & 0.4917 \\ 0.9750 & 0.3667 & 0.2333 \\ 0.4917 & 0.2333 & 0.1667 \end{pmatrix}.$$

Similarly, by solving the Lyapunov equation for O_G, we obtain

$$O_G = \begin{pmatrix} 0.5000 & 0.6667 & 0.7917 \\ 0.6667 & 0.9167 & 1.1000 \\ 0.7917 & 1.1000 & 1.3250 \end{pmatrix}.$$

Step 2. The Cholesky factors of C_G and O_G are:

$$L_c = \begin{pmatrix} 1.9812 & 0 & 0 \\ 0.4912 & 0.3528 & 0 \\ 0.2482 & 0.3152 & 0.0757 \end{pmatrix},$$

$$L_o = \begin{pmatrix} 0.7071 & 0 & 0 \\ 0.9428 & 0.1667 & 0 \\ 1.1196 & 0.2667 & 0.0204 \end{pmatrix}.$$

Step 3. From the singular value decomposition of $L_o^T L_c$ (using MATLAB function **svd**):

$$[U, \Sigma, V] = \mathbf{svd}(L_o^T L_c)$$

we have

$$\Sigma = diag\left(2.2589,\ 0.0917,\ 0.0006 \right)$$

$$V = \begin{pmatrix} 0.9507 & -0.3099 & 0.0085 \\ 0.3076 & 0.9398 & -0.1488 \\ 0.0381 & 0.1441 & 0.9888 \end{pmatrix}.$$

Step 4.

$$\Sigma^{\frac{1}{2}} = diag(1.5030, 0.3028, 0.0248).$$

Step 5. The transforming matrix T is:

$$T = L_c V \Sigma^{-\frac{1}{2}} = \begin{pmatrix} -1.2532 & 2.0277 & 0.6775 \\ -0.3835 & -0.5914 & -1.9487 \\ -0.2234 & -0.7604 & 1.2131 \end{pmatrix}.$$

Step 6. The balanced matrices are:

$$\tilde{A} = T^{-1} A T = \begin{pmatrix} -0.7659 & 0.5801 & -0.0478 \\ -0.5801 & -2.4919 & 0.4253 \\ 0.0478 & 0.4253 & -2.7422 \end{pmatrix}.$$

Verify:

$T^{-1} C_G T^{-T} = T^T O_G T = \Sigma = \text{diag}(2.2589, 0.0917, 0.0006).$

Computational Remarks:

- **The explicit computation of the product $L_o^T L_c$ can be a source of round-off errors.** The small singular values might be almost wiped out by the rounding errors in forming the explicit product $L_o^T L_c$. It is suggested that the algorithm of Heath et al. (1986), that *computes the singular values of a product of two matrices without explicitly forming the product, be used in practical implementation of this algorithm.*

MATLAB NOTES: The MATLAB function in the form:

$$\text{SYSB} = \textbf{balreal}\ (\text{sys})$$

returns a balanced realization of the system (A, B, C). The use of the function **balreal** in the following format:

$$[\text{SYSB, G, T, TI}] = \textbf{balreal}\ (\text{sys})$$

returns, in addition to $\tilde{A}, \tilde{B}, \tilde{C}$, of the balanced system, a vector G containing the diagonal of the Grammian of the balanced realization. The matrix T, the matrix of the similarity transformation that transforms (A, B, C) to $(\tilde{A}, \tilde{B}, \tilde{C})$ and TI is its inverse.

MATCONTROL NOTES: Algorithm 14.2.1 has been implemented in MATCONTROL function **balsvd**.

- **The Square-Root Algorithm (Algorithm 14.2.2)**

(Balanced Realization of a continous-time nommianial realization).

MATCONTROL Function: balsqt

Software for Balanced Realization

- MATLAB Function - **balreal** (works with **SYS**)
- MATCONTROl Function - **balsvd** (works with **matrices**) and **balsqt** (works with **matrices**).

Model Reduction via Balanced Truncation

Idea: Eliminate the less controllable and less observable states.

- Discard the small Hankel singular values

Algorithm for Model Reduction via Balanced Truncation (Chapter 14.4.1)

Step 1. Choose q, the order of ROM

- $q = \Sigma_{i=1}^{d} s_i$
- $s_i =$ the multiplicity of σ_i

Assume $\sigma_d >> \sigma_{d+1}$.

Step 2. Partition the balanced model $(\tilde{A}, \tilde{B}, \tilde{C})$ conformably:

$$\tilde{A} = \begin{pmatrix} A_R & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

$$\tilde{B} = \begin{pmatrix} B_R \\ B_2 \end{pmatrix}$$

$$\tilde{C} = (C_R, C_2).$$

$A_R \in \mathbb{R}^{q \times q}$, B_R and C_R are similar.

Step 3. ROM $= (A_R, B_R, C_R)$.

Properties of the Reduced Order Model

- ROM (A_R, B_R, C_R) is Stable.
- ROM by the Balanced Truncation Method **does not minimize** $||G(s) - G(R)(s)||_\infty$. *Only gives an upper bound.*

$$\bullet\ ||G(s) - G_R(s)||_\infty \leq 2(\sigma_{d+1} + \ldots + \sigma_N).$$

N - The numbers of distinct singular values

- If $d = N - 1,\ ||G(s) - G_{N-1}(s)||_\infty = 2\sigma_N$.

Example 14.4.1 Consider Example **14.2.1** once more. Choose $q = 2$. Then $A_R =$ The 2×2 leading principal submatrix of $\tilde{A}$ is :

$$A_R = \begin{pmatrix} -0.7659 & 0.5801 \\ -0.5801 & -2.4919 \end{pmatrix}.$$

The eigenvalues of A_R are: $-0.9900,$ and $-2.2678.$
Therefore A_R is stable.
The matrices B_R and C_R are:

$$B_R = \begin{pmatrix} -1.8602 \\ -0.6759 \end{pmatrix}, \; C_R = (-1.8602, 0.6759).$$

Let $G_R(s) = C_R(sI - A_R)^{-1}B_R$.

Verification of the Error Bound: $\|G(s) - G_R(s)\|_\infty = 0.0012$. Since $2\sigma_3 = 0.0012$, the error bound given by (14.4.4) is satisfied.

Numerical Stability of the ROM via BT

- Transforming Matrix T can be highly **ill-conditioned**.
  Example

$$\left(\begin{array}{c|c} A & B \\ \hline C & O \end{array}\right) = \left(\begin{array}{cc|c} -\frac{1}{2} & -\epsilon & \epsilon \\ 0 & -\frac{1}{2} & 1 \\ \hline 1 & \epsilon & 0 \end{array}\right)$$

$$T = \begin{pmatrix} \sqrt{\frac{1}{\epsilon}} & 0 \\ 0 & \sqrt{\epsilon} \end{pmatrix}.$$

- As $\epsilon \to 0$, Cond $(T) \to \infty$.

The Schur Method for Model Reduction

- No Balancing
- Only Orthogonal matrices used in transformations.
- Based on transformation of FOM using the **RSF** of $C_G O_G$:

$$Y = X^T C_G O_G X = \text{ Real Schur Form } (\textbf{ordered})$$

Idea: Compute the orthonormal bases of the right and left invariant subspaces corresponding to the **large** eigenvalues of the matrix $C_G O_G$ by finding the ordered real schur form.

The Schur Method for Model Reduction (Algorithm 14.4.2)

Step 1. Find the **RSF** of the product $C_G O_G$: $X^T C_G O_G X = Y$

Step 2. Reorder:

Ascending Order:

$$U^T Y U = \begin{pmatrix} \lambda_1 & & * \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}$$

Descending Order:

$$V^T Y V = \begin{pmatrix} \lambda_n & & * \\ & \ddots & \\ 0 & & \lambda_1 \end{pmatrix}$$

$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

Step 3. Partition

$$\begin{aligned} U &= (U_S, U_T) \\ V &= (V_S, V_T) \end{aligned}$$

- U_S - contains first $n - q$ columns of U
- V_S - contains q columns of V.

Step 4. Find SVD of $U_T^T V_S$

$$U_T^T V_S = Q \Sigma R^T$$

Step 5. Compute the transforming Matrices

$$S_1 = U_T Q \Sigma^{-\frac{1}{2}}$$

$$S_2 = V_T R \Sigma^{-\frac{1}{2}}$$

Step 6. Compute **ROM** $= (A_R, B_R, C_R)$:

$$\begin{aligned} A_R &= S_1^T A S_2 \\ B_R &= S_1^T B \\ C_R &= C S_2. \end{aligned}$$

(No matrix inversion for the transforming matrix)

- **MATCONTROL Function: modreds**

Comparison and Recommendation

- Model Reduction via Balanced Truncation is a standard procedure.
- Works well for well-equilibrated systems.
- Use the Schur method only in case of severe ill-conditioning.

Software for Balancing and Model Reduction

- **MATLAB Control System Toolbox**
  - balreal - Grammian based balancing
  - modred - model reduction

MATCONTROL

- BALSVD - Internal Balancing using the SVD.
- BALSQT - Square Root Algorithm for Balancing
- MODREDS - Model Reduction using the Schur Method
- HNAPX - Hankel-Norm Approximation.

CSP - ANM

- **The Schur Method**
  Dominant Subsystem [*System, Method* $\rightarrow$ *Schur Decomposition*].
- **The Square-Rood Method**
  Dominant Subsystem [System, Method $\rightarrow$ Square Root].

System Identification

(Chapter 9)

State-Space Realization

Given the transfer function $G(s)$ of order $r \times m$, find the matrix A, B, C, and D such that

$$G(s) = C(sI - A)^{-1}B + D.$$

- **Minimal Realization** (MR) : (A, B) is controllable and (A, C) is observable.

- **McMillan Degree**: The dimension of an MR is called the McMillan Degree.

Given a large number of Markov Parameters

$$H_k = CA^{k-1}B,\ k = 1, 2, \ldots$$

Find the minimal realization (A, B, C, D) of $G(s)$.

- Markov Parameters are easier to compute for a discrete-time system.

Hankel Matrix of Markov Parameters

$$M_k = \begin{pmatrix} H_1 & H_2 & \cdots & H_k \\ H_2 & H_3 & \cdots & H_{k+1} \\ \vdots & & & \\ H_k & H_{k+1} & \cdots & H_{2k-1} \end{pmatrix}$$

An SVD Algorithm for Minimal Realization

(Algorithm 9.3.1)

Inputs: *The set of Markov parameters:* $\{H_1, H_2, \cdots, H_{2N+1}\}$ *(N should be at least equal to the McMillan degree).*

Outputs: *The matrices* $A, B,$ *and* C *of a minimal realization.*

Step 1. *Find the SVD of the Hankel matrix*

$$M_{N+1} = \begin{pmatrix} H_1 & H_2 & \cdots & H_{N+1} \\ H_2 & H_3 & \cdots & H_{N+2} \\ \vdots & & & \\ H_{N+1} & H_{N+2} & \cdots & H_{2N+1} \end{pmatrix} = USV^T,$$

where $S = \ diag\ (s_1, s_2, \cdots, s_p, 0, \cdots 0)$,
and $s_1 \geq s_2 \geq \cdots \geq s_p > 0$

Step 2. *Form* $U' = US^{\frac{1}{2}}$ *and* $V' = S^{\frac{1}{2}}V^T$, *where* $S^{\frac{1}{2}} = diag(s_1^{\frac{1}{2}}, s_2^{\frac{1}{2}}, \ldots, s_p^{\frac{1}{2}}, 0, \cdots, 0)$.

Step 3. *Define*

U_1 = The first N block rows and the first p columns of U'

U_2 = The last N block rows and the first p columns of U'

$U^{(1)}$ = The first block row and the first p columns of U'

$V^{(1)}$ = The first p rows and the first block column of V'.

Step 4. *Compute* $A = U_1^{\dagger} U_2$, *Set* $B = V^{(1)}, C = U^{(1)}$.

■

Theorem 9.3.2 (Kung) Let E_i denote the error matrix; that is,

$$E_i = CA^{i-1}B - H_i, i \geq 1.$$

Assume that the given impulse response sequence $\{H_k\}$ *is*
convergent. That is, $H_k \to 0$, *when* $k \to \infty$.
Then

- $\sum_{i=1}^{2N+1} \parallel E_i \parallel_F^2 \leq \epsilon\sqrt{n+m+r}$, where ϵ is a small positive number, and n, m and r are, respectively, the number of states, inputs and outputs.

Properties of MR by Algorithm 9.3.1

- The minimal realization obtained by Algorithm 9.3.1 is (a) **discrete-stable** and (b) **internally balanced**; that is, the controllability and observability Grammians for this realization are the same and are equal to a diagonal matrix.

MATCONTROL Function: minresvd

Example 9.3.1 Let $N = 2$ and the given set of Markov parameters be:

$$\{H_1, H_2, H_3, H_4, H_5\} = \{3, 5, 9, 17, 33\}.$$

Step 1. $M_3 = \begin{pmatrix} 3 & 5 & 9 \\ 5 & 9 & 17 \\ 9 & 17 & 33 \end{pmatrix}$. Then

$$U = \begin{pmatrix} 0.2414 & -0.8099 & 0.5345 \\ 0.4479 & -0.3956 & -0.8018 \\ 0.8609 & 0.4330 & 0.2673 \end{pmatrix},$$

$$S = \text{diag}\,(44.3689 \;\; 0.6311 \;\; 0), \text{ and}$$

$$V^T = \begin{pmatrix} 0.2414 & 0.4479 & 0.8609 \\ -0.8099 & -0.3956 & 0.4330 \\ 0.5345 & -0.8018 & 0.2673 \end{pmatrix}.$$

Step 2. $U' = \begin{pmatrix} 1.6081 & -0.64340 & 0 \\ 2.9835 & -0.31430 & 0 \\ 5.7343 & 0.34400 & 0 \end{pmatrix}$,

$$V' = \begin{pmatrix} 1.6081 & 2.9835 & 5.7343 \\ -0.6434 & -0.3143 & 0.3440 \\ 0 & 0 & 0 \end{pmatrix}.$$

Step 3.

$$U_1 = \begin{pmatrix} 1.6081 & -0.6434 \\ 2.9835 & -0.3143 \end{pmatrix}$$

$$U_2 = \begin{pmatrix} 2.9835 & -0.3143 \\ 5.7343 & 0.3440 \end{pmatrix}.$$

$$U^{(1)} = (1.6081 \quad -0.6434)$$

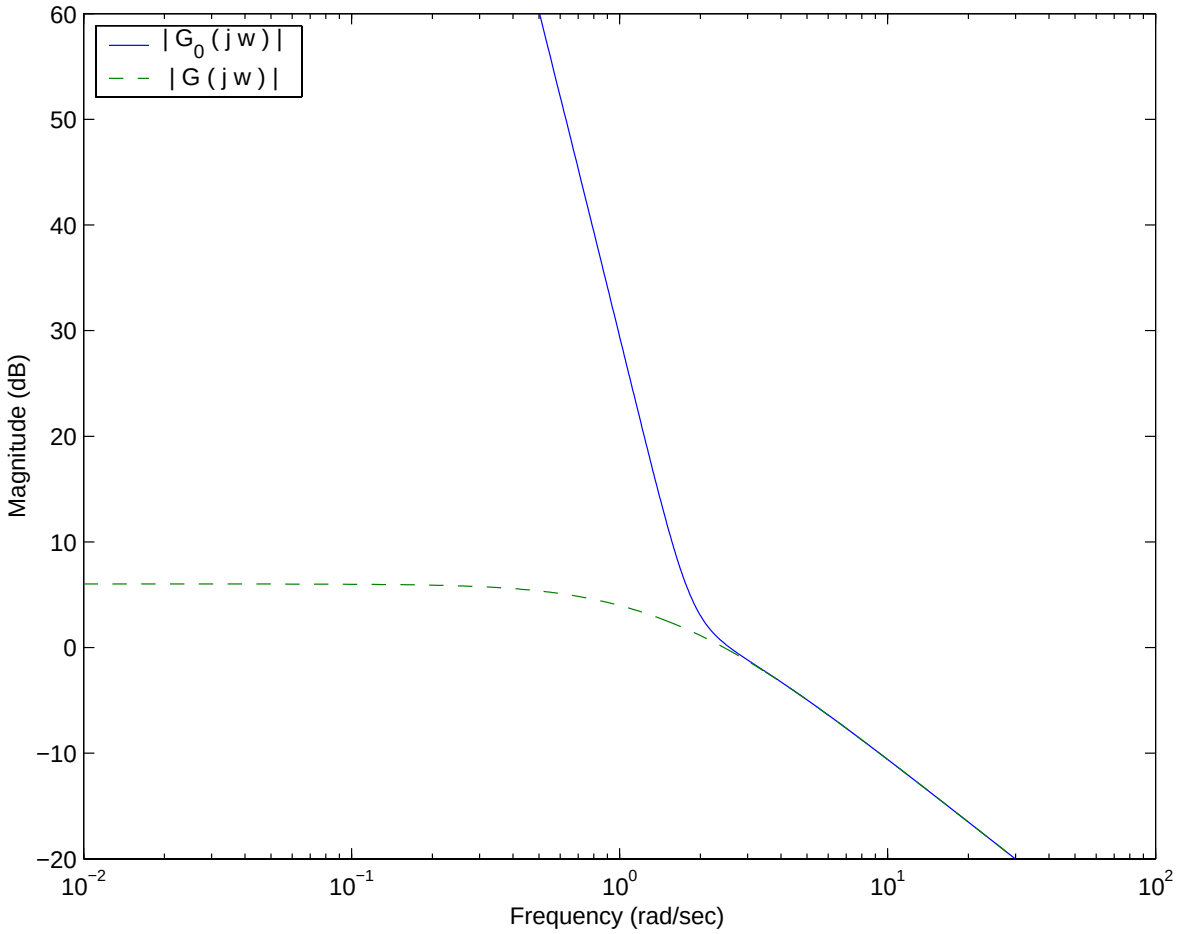
$$V^{(1)} = \begin{pmatrix} 1.6081 \\ -0.6434 \end{pmatrix}.$$

Step 4.

$$A = U_1^{\dagger} U_2 = \begin{pmatrix} 1.9458 & 0.2263 \\ 0.2263 & 1.0542 \end{pmatrix}$$

$$B = V^{(1)} = \begin{pmatrix} 1.6081 \\ -0.6434 \end{pmatrix}$$

$$C = U^{(1)} = (1.6081 \quad -0.6434).$$



Comparison of Transfer Functions

A Modified SVD Algorithm for Minimal Realization

- There exists a modified SVD algorithm for minimal realization **(Algorithm 9.3.2)**
- The modified algorithm requires **lower order block Hankel Matrices** in computing the matrices A, B, and C
- MATCONTROL function **minremsvd** implements this algorithm

Subspace Identification

Given a large number of input and output measurements, u_k and y_k, determine the order n of the unknown system and the system matrices (A, B, C, D) up to within a similarity transformation.

An SVD-Based Subspace Identification Algorithm

$$H_{k|k+i} = \begin{pmatrix} u_k & u_{k+1} & \cdots & u_{k+j-1} \\ y_k & y_{k+1} & \cdots & y_{k+j-1} \\ u_{k+1} & u_{k+2} & \cdots & u_{k+j} \\ y_{k+1} & y_{k+2} & \cdots & y_{k+j} \\ \vdots & \vdots & & \vdots \\ u_{k+i-1} & u_{k+i} & \cdots & u_{k+i+j-2} \\ y_{k+i-1} & y_{k+i} & \cdots & y_{k+i+j-2} \end{pmatrix}$$

$$H_{k+i|k+2i} = \begin{pmatrix} u_{k+i} & u_{k+i+1} & \cdots & u_{k+i+j-1} \\ y_{k+i} & y_{k+i+1} & \cdots & y_{k+i+j-1} \\ u_{k+i+1} & u_{k+i+2} & \cdots & u_{k+i+j} \\ y_{k+i+1} & y_{k+i+2} & \cdots & y_{k+i+j} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ u_{k+2i-1} & u_{k+2i} & \cdots & u_{k+2i+j-2} \\ y_{k+2i-1} & y_{k+2i} & \cdots & y_{k+2i+j-2} \end{pmatrix}$$

Algorithm 9.4.1 *(A Deterministic Subspace Identification Algorithm).*

Inputs: *The input and output sequence* $\{u_k\}$ *and* $\{y_k\}$*, respectively. The integers* $i \geq n$*, where* n *is the order of the sys tem to be identified and* j*.*

Outputs: *The identified system matrices* A, B, C *and* D*.*

Assumptions:

1. *The system is observable.*
2. *The integers* i *and* j *are sufficiently large, and in particular* $j >> \max(mi, ri)$*, where* m *and* r *are the number of inputs and outputs.*

Step 1. *Calculate U and S from the SVD of H, where*

$$H = \begin{pmatrix} H_{k|k+i} \\ H_{k+1|k+2i} \end{pmatrix}:$$

$$H = USV^T = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \begin{pmatrix} S_{11} & 0 \\ 0 & 0 \end{pmatrix} V^T$$

(Note that the dimensions of U_{11}, U_{12} and S_{11} are, respectively, $(mi + ri) \times (2mi + n)$; $(mi + ri) \times (2ri - n)$; and $(2mi + n) \times (2mi + n)$).

Step 2. *Calculate the SVD of $U_{12}^T U_{11} S_{11}$:*

$$U_{12}^T U_{11} S_{11} = (U_q, U_q^\perp) \begin{pmatrix} S_q & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_q^T \\ V_q^{\perp T} \end{pmatrix}$$

Step 3. *Solve the following set of linear equations for* A, B, C *and* D *(in the least-squares sense):*

$$\begin{pmatrix} U_q^T U_{12}^T U(mi+ri+1:(m+r)(i+1),:)S \\ U(mi+ri+m+1:(m+r)(i+1),:)S \end{pmatrix}$$

$$= \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} U_q^T U_{12}^T U(1:mi+ri:)S \\ U(mi+ri+1:mi+ri+m,:)S \end{pmatrix}$$

9.5.1 MATLAB CONTROL SYSTEM TOOLBOX

State-space models

minreal - Minimal realization and pole/zero cancellation.

augstate - Augment output by appending states.

9.5.2 MATCONTROL

MINRESVD - Finding minimal realization using singular value decomposition of Hankel matrix of Markov parameters
(**Algorithm 9.3.1**)

MINREMSVD - Finding minimal realization using singular value decomposition of Hankel matrix of lower order (**Algorithm 9.3.2**)

9.5.3 CSP-ANM

Model identification

- The system identification from its impulse responses is performed by `ImpulseResponseIdentify` [*response*].
- The system identification from its frequency responses is performed by `FrequencyResponseIdentify` [*response*].
- The system identification directly from input-output data is performed by `OutputResponseIdentify` $[u, y]$.

9.5.4 SLICOT

Identification

IB - Subspace Identification

Time Invariant State-space Systems

IB01AD	Input-output data preprocessing and finding the system order
IB01BD	Estimating the system matrices, covariances, and Kalman gain
IB01CD	Estimating the initial state and the system matrices B and D

TF - Time Response

TF01QD	Markov parameters of a system from transfer function matrix
TF01RD	Markov parameters of a system from state-space representation

In addition to the above mentioned software, the following toolboxes, especially designed for system identification are available.

- **MATLAB System Identification Toolbox**, developed by Prof. Lennart Ljung.
  (Website: http://www.mathworks.com)
- **ADAPTX**, developed by W. E. Larimore.
  (Website: http://adaptics.com)
- **Xmath Interactive System Identification Module**, described in the manual X-Math Interactive System Identification Module, Part 2, by P. VanOverschee, B. DeMoor, H. Aling, R. Kosut, and S. Boyd, Integrated Systems Inc., Santa Clara, California, USA, 1994
  (website: http://www.isi.com/products/MATRIX$_X$/Techspec/MATRIX$_X$-Xmath/xm36.html, -/MATRIX$_X$ XMATH/inline images/pg. 37 img.html and -/MATRIX$_X$-XMath/inlineimages/pg. 38img.html).

  For more details on these software packages, see the paper by DeMoor, Van Overschee and Favoreel (1999).

Control Software (Appendix A)

Information on

Existing Control Software

- **MATLAB Control Systems Toolbox.**
  Built on MATLAB. Implements some of the best known numerical algorithms for control problems.

only one algorithm for each problem

Information: *http://www.mathworks.com*

- **Control Systems Professional-Advanced Numerical Methods**

Based on *Mathematica*

A collection of Mathematica programs to solve control problems. Extends the slope of the existing CSP by *adding the state-of-the-art numerical methods* form the book:

Numerical Methods for Linear Control Systems Design and Analysis by B. N. Datta

Information:
http://www.wolfram.com/products/applications/anm

- **MATCONTROL**

  A collection of M-files implementing major algorithms of the book "*Numerical Methods for Linear Control Systems Design and Analysis*" by B.N. Datta

  A useful educational toolbox. The students, the instructors and the researchers will be able to compare different algorithms for the same problem with respect to *efficiency, stability, accuracy,* and *easiness-to-use* and specific design and analysis requirements.

Listing of MATCONTROL Files (Appendix B)

B.2 CHAPTER-WISE LISTING OF MATCONTROL FILES

Here is the Chapter-wise listing of MATCONTROL files.

Reference: Numerical Algorithms for Linear Control Systems

Design and Analysis, by B.N. Datta.

Chapter 5: Linear State Space Models and Solutions of the State Equations

* EXPMPADE - The Padé approximation to the exponential of a matrix **(Algorithm 5.3.1)**

* EXPMSCHR - Computing the exponential of a matrix using Schur decomposition **(Algorithm 5.3.2)**

EXMPHESS - Computing the exponential of a matrix using Hessenberg decomposition

* FREQRESH - Computing the frequency response matrix using Hessenberg decomposition **(Algorithm 5.5.1)**

INTMEXP - Computing an integral involving a matrix exponentials

∗ **Most important ones discussed in this workshop.**

Chapter 6: Controllability, Observability and Distance to Uncontrollability

* CNTRLHS	- Finding the controller-Hessenberg form **(Algorithm 6.7.1)**
* OBSERHS	- Finding the observer-Hessenberg form **(Section 6.8)**
CNTRLC	- Finding the controller canonical form (Lower Companion)
DISCNTRL	- Distance to controllability using the Wicks-DeCarlo algorithm

Chapter 7: Stability, Inertia and Robust Stability

* INERTIA	- Determining the inertia and stability of a matrix without solving a matrix equation or computing eigenvalues **(Algorithm 7.5.1)**
H2NRMCG	- Finding H_2-norm using the controllability Grammians
H2NRMOG	- Finding H_2-norm using the observability Grammian
* DISSTABC	- Determining the distance to the continuous-time stability **(Algorithm 7.6.1)**
* DISSTABD	- Determining the distance to the discrete-time stability **(Algorithm 7.6.2)**
* ROBSTAB	- Robust stability analysis using Lyapunov equations

Chapter 8: Numerical Solutions and Conditioning of Lyapunov and Sylvester Equations

* CONDSYLVC	-	Finding the condition number of the Sylvester equation problem **(Section 8.3)**
* LYAPCHLC	-	Finding the Cholesky factor of the positive definite solution of the continuous-time Lyapunov equation **(Algorithm 8.6.1)**
* LYAPCHLD	-	Finding the Cholesky factor of the positive definite solution of the discrete-time Lyapunov equation **(Algorithm 8.6.2)**
LYAPCSD	-	Solving the discrete-time Lyapunov equation using complex-Schur decomposition of A
LYAPFNS	-	Solving the continuous-time Lyapunov equation via finite series method
LYAPHESS	-	Solving the continuous-time Lyapunov equation via Hessenberg decomposition
* LYAPRSC	-	Solving the continuous-time Lyapunov equation via real-Schur decomposition **(Section 8.5.2)**
* LYAPRSD	-	Solving the discrete-time Lyapunov equation via real-Schur decomposition

* SEPEST - Estimating the *sep* function with triangular matrices

* SEPKR - Computing the *sep* function using Kronecker product

* SYLVHCSC - Solving the Sylvester equation using Hessenberg and complex Schur decompositions **(Algorithm 8.5.2)**

SYLVHCSD - Solving the discrete-time Sylvester equation using Hessenberg and complex-Schur decompositions

SYLVHESS - Solving the Sylvester equation via Hessenberg decomposition

* SYLVHRSC - Solving the Sylvester equation using Hessenberg and real Schur decompositions

SYLVHUTC - Solving an upper triangular Sylvester equation

Chapter 9: Realization and Subspace Identification

* MINRESVD - Finding minimal realization using singular value decomposition of the Hankel matrix of Markov parameters (**Algorithm 9.3.1**)

* MINREMSVD - Finding minimal realization using singular value decomposition of a Hankel matrix of lower order (**Algorithm 9.3.2**)

Chapter 10: Feedback Stabilization, Eigenvalue Assignment, and Optimal Control

* STABLYAPC - Feedback stabilization of continuous-time system using Lyapunov equation **(Section 10.2.2)**

* STABLYAPD - Feedback stabilization of discrete-time system using Lyupunov equation **(Section 10.2.2)**

* STABRADC - Finding the complex stability radius using the bisection method **(Algorithm 10.7.1)**

* HINFNRM - Computing H_∞-norm using the bisection method **(Algorithm 10.6.1)**

Chapter 11: Numerical Methods and Conditioning of the EVA Problems

* POLERCS - Single-input pole placement using the recursive algorithm

POLEQRS - Single-input pole placement using the QR version of the recursive algorithm

* POLERQS -Single-input pole placement using RQ version of the recursive algorithm

* POLERCM - Multi-input pole placement using the recursive algorithm **(Algorithm 11.3.1)**

* POLERCX - Multi-input pole placement using the modified recursive algorithm that avoids complex arithmetic and complex feedback. (**Algorithm 11.3.1**)

* POLEQRM - Multi-input pole placement using the explicit QR algorithm **(Section 11.3.2)**

POLESCH - Multi-input pole placement using the Schur decomposition **(Algorithm 11.3.3)**

∗ POLEROB - Robust pole placement **(Algorithm 11.6.1)**

Chapter 12: State Estimation: Observer and Kalman Filter

* SYLVOBSC	- Solving the constrained multi-output Sylvester-observer equation
* SYLVOBSM	- Solving the multi-output Sylvester-observer equation
* SYLVOBSMB	- Block triangular algorithm for the multi-output Sylvester-observer equation

Chapter 13: Numerical Solutions and Conditioning of the Algebraic Riccati Equations

RICEIGC - The eigenvector method for the continuous-time Riccati equation

* RICSCHC - The Schur method for the continuous-time Riccati equation **(Algorithm 13.5.1)**

RICSCHD - The Schur method for the discrete-time Riccati equation

RICGEIGD - The generalized eigenvector method for the discrete-time Riccati equation

* RICNWTNC - Newton's method for the continuous-time Riccati equation **(Algorithm 13.5.8)**

* RICNWTND - Newton's method for the discrete-time Riccati equation **(Algorithm 13.5.8)**

RICSGNC - The matrix sign-function method for the continuous- time Riccati equation **(Algorithm 13.5.6)**

RICSGND - The matrix sign-function method for the discrete-time Riccati equation **(Algorithm 13.5.7)**

* RICNWLSC - Newton's method with line search for the continuous-time Riccati equation **(Algorithm 13.5.9)**

* RICNWLSD - Newton's method with line search for the discrete-time Riccati equation **(Algorithm 13.5.1)**

Chapter 14: Internal Balancing and Model Reduction

* BALSVD - Internal balancing using the singular value decomposition **(Algorithm 14.2.1)**

BALSQT - Internal balancing using the square-root algorithm **(Algorithm 14.2.2)**

∗ MODREDS - Model reduction using the Schur method **(14.4.2)**

∗ HNAPRX - Hankel norm approximation **(Algorithm 14.5.1)**

A Case Study: Control of a 9-State Ammonia Reactor (Appendix C)

CASE STUDY: Control of a 9-state Ammonia Reactor

C1. Introduction

- **System Matrices**

$A =$

$$\begin{bmatrix} -4.019 & 5.12 & 0. & 0. & -2.082 & 0. & 0. & 0. & 0.870 \\ -0.346 & 0.986 & 0. & 0. & -2.340 & 0. & 0. & 0. & 0.970 \\ -7.909 & 15.407 & -4.069 & 0. & -6.450 & 0. & 0. & 0. & 2.680 \\ -21.816 & 35.606 & -0.339 & -3.870 & -17.800 & 0. & 0. & 0. & 7.390 \\ -60.196 & 98.188 & -7.907 & 0.340 & -53.008 & 0. & 0. & 0. & 20.400 \\ 0. & 0. & 0. & 0. & 94. & -147.200 & 0. & 53.200 & 0. \\ 0. & 0. & 0. & 0. & 0. & 94. & -147.200 & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 12.800 & 0. & -31.600 & 0. \\ 0. & 0. & 0. & 0. & 12.800 & 0. & 0. & 18.800 & -31.600 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.010 & 0.003 & 0.009 & 0.024 & 0.068 & 0. & 0. & 0. & 0. \\ -0.011 & -0.021 & -0.059 & -0.162 & -0.445 & 0. & 0. & 0. & 0. \\ -0.151 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \end{bmatrix}^T$$

$$C = \begin{bmatrix} 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. \end{bmatrix},$$

$$D = \begin{bmatrix} 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \end{bmatrix}.$$

C2. Testing the controllability via Reduction to Controller-Hessenberg Form

- MATCONTROL function **cntrlhs**

```
tol = 1e-13;
info = cntrlhs( A, B, tol)
info = 1
```

Conclusion: *The system is controllable.*

C3. Testing the Observability via Reduction to Observer-Hessenberg Form

- MATCONTROL function **obserhs**

```
info = obserhs( A, C, tol)
info = 1
```

Conclusion: The System is observable.

C4. Testing the stability by finding the Eigenvalues

- MATLAB function **eig** :

  $\{-147.2000\ ,\ -153.1189\ ,\ -56.0425\ ,\ -37.5446\ ,$
  $-15.5478\ ,\ -4.6610\ ,\ -3.3013\ ,\ -3.8592,\ -0.3047\}\ .$

  Conclusion: The system is asymptotically stable but it has a small eigenvalue $\lambda = -0.3047$ (relative to the other eigenvalues).

C5. Lyapunov Stabilization.

- MATCONTROL function **stablyapc**

beta = norm(A,'fro');
beta = 292.6085.
K_{-lyap} = stablyapc(A,B,beta)

The feedback matrix K_{-lyap}:

$$K_{-lyap} =$$

$$10^2 \begin{bmatrix} -3.3819 & -0.2283 & -56.4126 \\ 5118.1388 & 1207.4106 & 15424.1947 \\ -237858.9775 & -57713.9866 & -997148.5316 \\ -544.7145 & 220.0287 & 15199.8829 \\ 31495.6810 & 7491.5724 & 125946.2030 \\ -4510.0481 & 20403.0258 & -5516.3430 \\ 85.8840 & -495.7330 & 40.1441 \\ -39007.7960 & 182650.2401 & -43969.5212 \\ 38435.8476 & -150078.6710 & 61412.4200 \end{bmatrix}.$$

The eigenvalues of the corresponding closed-loop matrix are:

$$\{-292.6085 \pm 644.6016i, -292.6085 \pm 491.8461i,$$

$$-292.6085 \pm 145.4054i, -292.6085 \pm 49.3711i, -292.6085\}.$$

Note that these close-loop eigenvalues now are much further to the left of the complex plane than the open-loop ones.

C6. Pole-Placement Design. Move all the above nine eigenvalues to the negative real-axis with equal spacing in the interval $[-||A||_F/9, -||A||_F)]$.

- MATCONTROL function **polercm**

eig_{-rcm}=-[1:9]*beta/9;
K_{-rcm} = polercm (A,B, eig_{-rcm}).

The feedback matrix K_{-rcm}

$$10^5 \begin{bmatrix} -0.1088 & 14.0002 & -1358.6004 & 17.6295 & 171.8716 & 1.2245 & 0.0034 & 4.8847 & -5.8828 \\ -0.0153 & 2.1371 & -207.6062 & 2.6939 & 26.2618 & 0.1865 & 0.0005 & 0.7408 & -0.8998 \\ -0.0357 & -0.5495 & -74.6029 & 1.3318 & 9.3685 & 0.0670 & 0.0002 & 0.2666 & -0.3206 \end{bmatrix}.$$

The eigenvalues of the corresponding closed-loop matrix are:

$\{-292.6085, -260.0965, -227.5844, -195.0724, -162.5603,$
$-130.0482, -97.5362, -65.0241$ and $-32.5121\}$.

C7. The LQR and LQG Designs
LQR Design

$$\text{Optimal control-law: } u^0(t) = K_{-lqr}x(t)$$

- MATLAB function **lqr** with $R = eye(3)$, $N = zeros(9,3)$, and $Q = eye(9)$ gives

$$\boxed{K_{-lqr} = lqr(A, B, Q, R, N)}$$

The optimal gain matrix K_{-lqr} is:

$$10^{-1}\begin{bmatrix} 0.1187 & 0.0728 & 0.0228 & 0.0012 & -0.0007 & 0.0018 & 0.0003 & 0.0042 & 0.0044 \\ 0.2443 & -0.3021 & 0.0084 & -0.0465 & -0.0673 & -0.0138 & -0.0023 & -0.0464 & -0.0439 \\ -2.8408 & -0.5942 & -0.4540 & 0.0855 & 0.2102 & 0.0061 & 0.0003 & 0.0496 & 0.0378 \end{bmatrix}.$$

The eigenvalues of the corresponding closed-loop system are:

$$\{-153.1201\ ,\ -147.1984,\ -56.0452,\ -37.5442,$$
$$-15.5463,\ -4.6789,\ -3.3090\ ,\ -3.8484,\ \text{and}\ -0.3366\ \}.$$

Note that these closed-loop eigenvalues are quite close to the open-loop ones. *Also,* $||K_{-lqr}||$ *is much smaller than that of* $||K_{-rcm}||$.

LQG Design

$$\dot{x}_e = Ax_e + Bu + L(y_m - Cx_e - Du)$$

- MATLAB functions **kalman** and **lqgreg** gives

```
sysA = ss(A,B,C,D);
Qn = 1E-3 *eye(3); Rn =
1E-3 *eye(3);
[K_est,          L]      =
kalman(sysA,Qn,Rn)
```

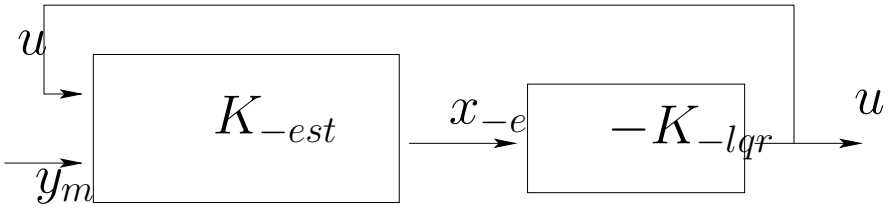
The filter gain matrix L is:

$$10^{-3} \begin{bmatrix} -0.0007 & 1.0703 & 1.6155 & 1.9729 & 2.8499 & 2.1636 & 1.3816 & 0.7990 & 1.5340 \\ 0.0176 & 0.6284 & 0.9780 & 1.1679 & 1.5765 & 1.2251 & 0.7990 & 0.4962 & 0.9469 \\ 0.0349 & 1.2163 & 1.8921 & 2.2676 & 3.0810 & 2.3701 & 1.5340 & 0.9469 & 1.8112 \end{bmatrix}^T$$

Using the matrices K_{-est} and L, the LQG regulator can now be designed. The MATLAB command for finding an LQG regulator is **lqgreg**.

$$\boxed{RLQG = lqgreg(K_{-est}, K_{-lqr})}$$

The resulting regulator $RLQG$ has input y_m and the output $u = -K_{-lqr}x_{-e}$ as shown below:



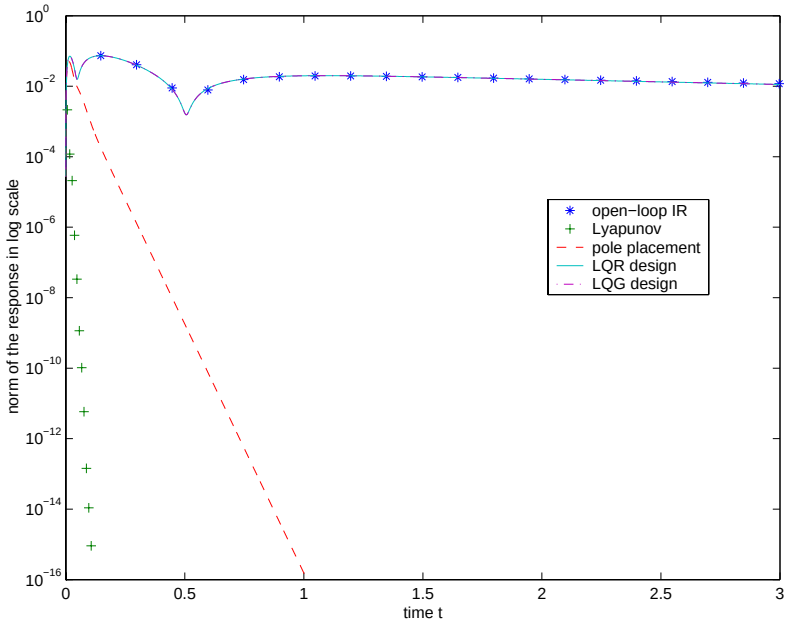


Figure 1: Comparison of the Impulse Responses.

C8. State-Estimation (Observer): Kalman Estimator vs. Sylvester Equation Estimator

MATCONTROL Function: sylvobsmb

sylvobsmb implements Algorithm 12.7.2 (A Recursive Block Triangular Algorithm) for this purpose, is used here. The observer eigenvalues: $ev = [-2, -4 \pm 2i, -5, -6, -7]^T$.

$$\boxed{[X, F, G] = sylvobsmb(A, C, ev)}$$

$$X = \begin{bmatrix} -5734.5147 & 5470.8582 & -1106.8206 & 52.2506 & 287.3781 & 0. & 0. & 0. & 727.5156 \\ -8.4146 & 13.4543 & -0.0962 & -0.0139 & -0.5931 & 0. & 0. & 0. & -1.2949 \\ 6.8075 & -9.1950 & 0.8500 & -0.0354 & 0.2121 & 0. & 0. & 0. & 0.5729 \\ 0.5214 & -0.8505 & 0.0685 & -0.0029 & 0.0782 & 0. & 0. & 0. & 0.2327 \\ 0. & 0. & 0. & 0. & 1. & 0. & -0.0213 & 0. & 3.1234 \\ 0. & 0. & 0. & 0. & 0. & 1. & 1.5234 & 0. & -7.1875 \end{bmatrix}$$

$$F = \begin{bmatrix} -2. & 0. & 0. & 0. & 0. & 0. \\ -0.0042 & -5. & 0. & 0. & 0. & 0. \\ 0. & 0.2435 & -6. & 0. & 0. & 0. \\ 0. & 0. & -0.4901 & -7. & 0. & 0. \\ 0. & 0. & 0. & -115.4430 & -4. & -2. \\ 0. & 0. & 0. & 0. & 2. & -4. \end{bmatrix}$$

$$G = 10^1 \begin{bmatrix} 0. & 0. & 0. & 0. & 0.6094 & -21.8109 \\ 1367.7293 & -2.4345 & 1.0771 & 0.4375 & 5.8721 & -8.1925 \\ -1793.4391 & 3.0741 & -1.1006 & -0.4058 & -5.3316 & 19.2128 \end{bmatrix}^T$$

Error X: $\|XA - FX - GC\|_F = 1.2246 \cdot 10^{-11}$.

Figure 2 shows the comparison of relative errors, between actual and estimated states in tow cases: *Kalman estimator and Sylvester-equation estimator.* The quantity plotted is

$$r(t) = \frac{||x(t) - \hat{x}(t)||}{||x(t)||}$$

where $\hat{x}(t)$ is the estimate given by the estimator in each case.

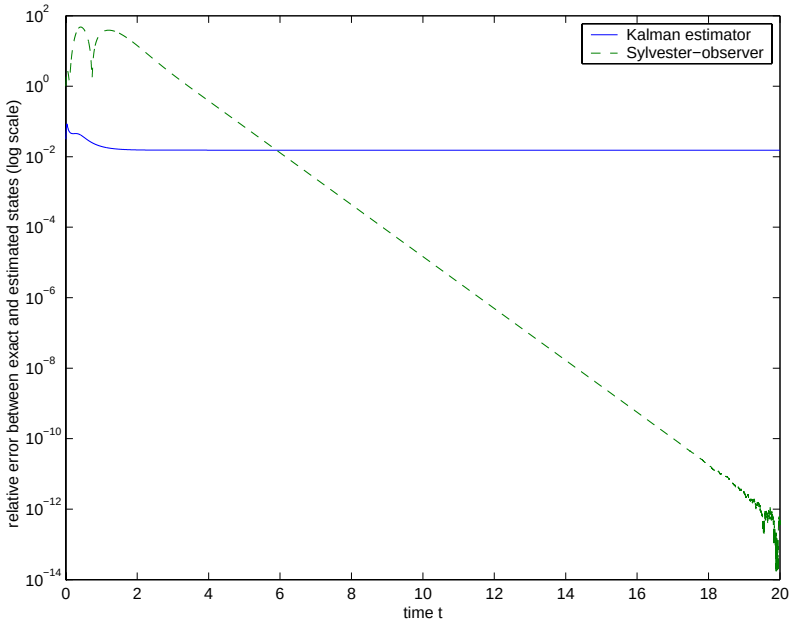


Figure 2: Comparison between Kalman and Sylvester-observer Estimations.

The plot show that error in the Sylvester-observer estimator approaches to zero faster than the Kalman estimator as the time increases.

C9. System Identification and Model Reduction

- Transfer Function:

$$H(s) = C(sI - A)^{-1}B = \sum_{i=1}^{\infty} \frac{CA^iB}{s^i}.$$

  - Markov parameters:
    $H_i = CA^iB$, $i = 1, 2, 3, \ldots$
  - The frequency response function: $G(j\omega) = H(j\omega)$ where ω is a nonnegative real number and $j = \sqrt{-1}$.
  - *MATCONTROL* functions **minresvd** and **minremsvd**: [A_s,B_s,C_s] = minresvd(4,[H1 H2 H3 H4 H5 H6 H7 H8 H9],1e-8);

N = 4, tol = 1e-13 $[A_{-s}, B_{-s}, C_{-s}]$ = minresvd (N, H_{-i}, tol); $[A_{-r}, B_{-r}, C_{-r}]$ = modreds $(A_{-s}, B_{-s}, C_{-s}, 9)$

Comparisons of Frequency Response Functions

- Original Model
- Model by SVD Algorithm **(Algorithm 9.3.1)**
- Model by Modified SVD Algorithm **(Algorihtm 9.3.2)**
- SVD Model followed by Model Reduction

```
omega=1:.1:100;
G = freqresh(A,B,C,omega);
G_s = freqresh(A_s,B_s,C_s,omega);
G_sm = freqresh(A_sm,B_sm,C_sm,omega);
G_r = freqresh(A_r,B_r,C_r,omega)
```

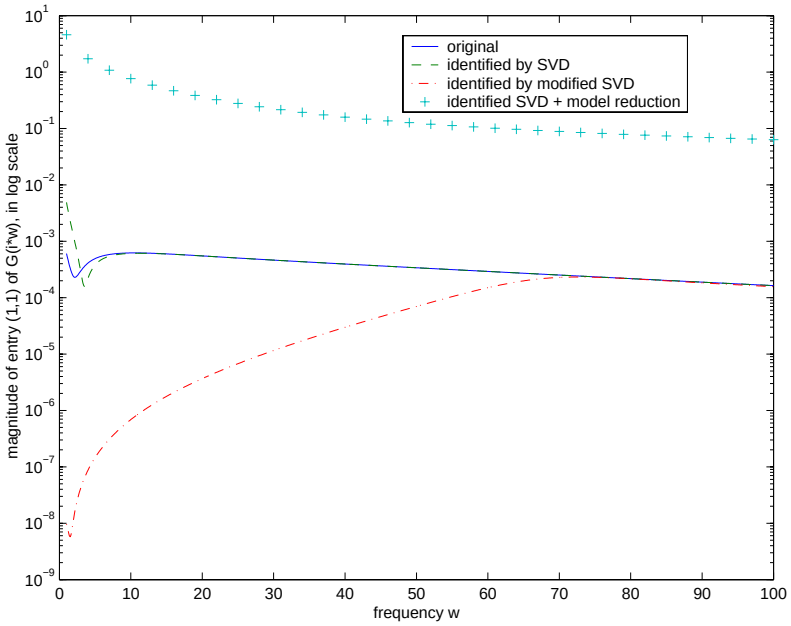


Figure 3: C.3 Comparison Between Frequency Responses.

Bibliography:

1. P. Benner, A. Laub and V. Mehrmann, *A Collection of benchmark examples for the numerical solution of algebraic Riccati equations I: continuous-time case. Technische Universität Chemnitz-Zwickau*, SPC Report 95-22, 1995.

2. L. Patnaik, N. Viswanadham and I. Sarma, *Computer control algorithms for a tubular ammonia reactor*. IEEE Trans. Automat. Control, AC-25, pp. 642-651, 1980.

H_∞-Control

Goal of H_∞-Control: Stabilize a Perturbed version of a system, assuming certain bounds for perturbations.

Problem Statement

Given

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B_1 w(t) + B_2 u(t)_1 \\ z(t) &= C_1 x(t) + D_{12} u(t)_2 \\ y(t) &= C_2 x(t) + D_{21} w(t)_3 \end{aligned}$$

- $x(t)$ - The state vector
- $w(t)$ - The disturbance signal
- $u(t)$ - The control input
- $z(t)$ - The controlled output
- $y(t)$ - The measured output

Find a controller $K(s)$ such that $||T_{zw}(s)|| < \gamma$, for a give positive number γ.

- $T_{zw}(s)$ = Transfor Function from the disturbance w to the output z

$$= G_{11} + G_{12}K(I - G_{22}K)^{-1}G_{21}$$

- $G(s) = \begin{pmatrix} 0 & D_{12} \\ D_{21} & 0 \end{pmatrix} + \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} (SI - A)^{-1}(B_1, B_2)$

  $= \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix}.$

Assumptions

- (A, B_1) is stabilizable and (A, C_1) is detectable
- (A, B_2) is stabilizable and (A, C_2) is detectable
- $D_{12}^T(C_{11}D_{12}) = (0, I)$
- $\begin{pmatrix} B_1 \\ D_{21} \end{pmatrix} D_1^T = \begin{pmatrix} 0 \\ I \end{pmatrix}$

H_∞-Theorem: A solution exists if and only if there exist symmetric positive semidefinite stabilizing solutions X and Y, respectively to the pair of $ARES$:

$$XA + A^TX - X\left(B_2B_2^T - \frac{1}{\gamma^2}B_1B_1^T\right)X + C_1C_1^T = 0$$

$$AY + YA^T - Y\left(C_2^TC_2 - \frac{1}{\gamma^2}C_1^TC_1\right)Y + B_1B_1^T = 0$$

- A Controller is given by

$$K(s) = -F(sI - \hat{A})^{-1}ZL$$

where

- $\hat{A} = A + \frac{1}{\gamma^2}B_1B_1^TX + B_2F + ZLC_2$
- $F = -B_2^TX$
- $Z = (I - \frac{1}{\gamma^2}YX)^{-1}$.

MATLAB Implementation

The function care can be used.
Write

$$A^T X + XA - X(B_2 B_2^T - \frac{1}{\gamma^2} B_1 B_1^T)X + C_1^T C_1 = 0$$

in the form

$$A^T X + XA - X(B_1, B_2) \begin{pmatrix} -\gamma^{-2} I & 0 \\ 0 & I \end{pmatrix}^{-1} \begin{pmatrix} B_1^T \\ B_2^T \end{pmatrix} X + C_1^T C_1 = 0.$$

Example

$$A = a,\ B_1 = (1, 0), B_2 = b_2$$

$$C_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix},\ D_{12} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$C_2 = c_2,\ D_{21} = (0, 1).$$

- Assumptions are satisfied.

  Take $a = -1, b_2 = c_2 = 1, \gamma = 2$.

- $T_{zw} = \begin{pmatrix} -1.7321 & 1 & -0.7321 \\ 1 & 0 & 0 \\ -0.7321 & 0 & -0.7321 \end{pmatrix}$.

- $||T_{zw}||_\infty = 0.7321 < \gamma = 2$.

H_∞ - Norm Algorithms.

- A Bisection Algorithm (**Algorithm 10.6.2**) (Boyd, et al. (1989)).
- Two-step Algorithm (MATLAB Command: **norm** (sys, inf))

Bisection Algorithm (Algorithm 10.6.1)

$$M_\gamma = \begin{pmatrix} A + BR^{-1}D^TC & BR^{-1}B^T \\ -C^T(I + DR^{-1}D^T)C & -(A + BR^{-1}D^TC)^T \end{pmatrix}$$

$$R = \gamma^2 I - D^T D.$$

Theorem. Let $G(s)$ be the stable transfer function. Then $||G(s)||_s < \gamma$ if and only if $\sigma_{\max}(D) < \gamma$ and M_γ has no imaginary eigenvalues.

MATCONTROL Function: hinfnrm.