โปรแกรมคำนวณ สเตทเสปซ ด้วย VB6

STATE SPACE COMPUTATIONS SOFTWARE BY VB6

โดย
นาย นพดล ตีรณะชัยดีกุล
นาย นพดล นันทสุขเกษม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขา วิศวกรรมระบบควบคุม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา **2544**

ภาควิชา **วิศวกรรมระบบควบคุม**

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **โปรแกรมคำนวณ สเตทสเปซ ด้วย VB6**

ผู้จัดทำ

1. **นาย นพดล ตีรณะชัยดีกุล** **41014201**
2. **นาย นพดล นันทสุขเกษม** **41014202**

..... อาจารย์ที่ปรึกษา

(รศ. วิพันธ์ ปรีชาพานิช)

โปรแกรมคำนวณ สเตทสเปซ ด้วย VB6

นพดล ตีรณะชัยดีกุล
นพดล นันทสุขเกษม
รศ. วิพันธ์ ปรีชาพานิช อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

การคำนวณในเรื่อง สเตทสเปซ (State Space) ส่วนใหญ่มักมีเรื่องเกี่ยวกับ การจัดการในเรื่องของเมตริกซ์ ค่าไอเก็น เวคเตอร์ไอเก็น เข้ามาเกี่ยวข้อง ซึ่งการคำนวณในเรื่องเหล่านี้เป็นงานที่ค่อนข้างน่าเบื่อ และผลการคำนวณที่ได้มักจะเกิดข้อผิดพลาด

ในโครงงานฉบับนี้จะนำเสนอการนำโปรแกรม Microsoft Visual Basic 6 มาเขียนเป็นโปรแกรมสำเร็จรูป ซึ่งประกอบไปด้วยเรื่อง การหาฟังก์ชันถ่ายโอน, การ Realization, การหาผลคำตอบจากสมการสภาวะเชิงเส้น, ความสามารถในการควบคุมได้ ฯลฯซึ่งแม้ว่าในปัจจุบันจะมีโปรแกรมสำเร็จรูปมากมายเช่น MATLAB, **Maple** แต่ผู้ใช้ต้องสามารถจำคำสั่งหรือฟังก์ชันต่าง ๆ ของโปรแกรมสำเร็จรูปนั้นได้จึงสามารถใช้งานได้แต่ถ้าใช้โปรแกรมสำเร็จรูปที่เขียนขึ้นด้วยสมบัติของVB6 คือ GUI ซึ่งทำให้ผู้ใช้งานเพียงแค่ป้อนค่าและกดปุ่มก็จะสามารถแสดงผลได้ทันที

State Space Computations By VB6.0

Noppadon Triranachaideekul
Noppadon Nuntasukasame
Associate Professor Vipan Prichapanich
2001

Abstract

The Majority of State Space Computations involves matrix manipulations, eigenvalue – eigenvector determinations, which are complex, tedious and error prone.

The purpose of this thesis to demonstrate how Microsoft Visual Basic 6.0 language can be used to develop State Space Computations package that consists of Transfer function calculation, solution of the linear state equation, Observability and Controllability systems, etc. Although there are many packages (such as MATLAB, **Maple**) that could calculable, user had to recognize the commands or functions of those packages. By using this project, it was found that users only supply data and press few buttons, then the results of computation will be appeared.

สารบัญ

สารบัญรูปภาพ

โปรแกรมคำนวณ สเตทเสปซ ด้วย VB6

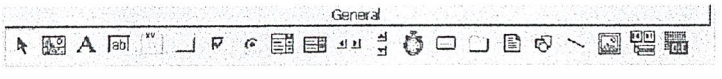
บทที่ 1 แนะนำโครงงาน

1.1 ความเป็นมาของโครงงาน

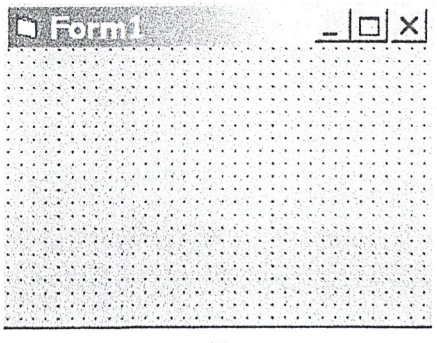
เนื่องจากการคำนวณในเรื่อง *สเตทเสปซ* (State Space) มักมีเรื่องการจัดการทางเมตริกซ์เข้ามาเกี่ยวข้อง ทำให้เกิดความยุ่งยาก ซึ่งถึงแม้ว่าในปัจจุบันจะมี โปรแกรมสำเร็จรูปเช่น MATLAB®, Maple®, Mathematica® ก็ตามแต่ Software สำเร็จรูปเหล่านี้เวลาจะใช้เราต้องรู้คำสั่งหรือจำฟังก์ชั่นต่างๆ นั้นได้ แต่หากว่าเราจำฟังก์ชั่นที่จะใช้งานนั้นไม่ได้ ก็ต้องเขียนฟังก์ชั่นขึ้นมาใช้งานเอง ซึ่งนับว่าไม่ใช่เรื่องที่ง่ายเลย ซึ่งจากจุดนี้เองทำให้เกิดแนวความคิดในการเขียนโปรแกรมสำเร็จรูปในเรื่อง สเตทเสปซ โดยไม่ต้องจำคำสั่งแต่จะเป็นการสร้างจอภาพเพื่อติดต่อกับผู้ใช้งาน (Graphic User Interface : GUI) ขึ้นมาแทน ซึ่งวิธีการนี้เป็นการอำนวยความสะดวกให้กับผู้ต้องการใช้งานได้มากทีเดียว

1.2 แนะนำ Microsoft Visual Basic 6.0

Microsoft Visual Basic 6.0 เป็นภาษาคอมพิวเตอร์ที่นิยมนำมาพัฒนาโปรแกรมบน Windows เนื่องจากเป็นภาษาคอมพิวเตอร์ที่ใช้เทคโนโลยีลักษณะทำให้มองเห็นภาพ (Visualize) โดยเพียงเลือก *คอนโทรล* (Control) ที่เหมาะสมแล้ววาดลงบน Form จากนั้นเราก็เขียน โปรแกรมเพื่อกำหนดการทำงานให้ คอนโทรลที่วางลงไปก็สามารถสร้างจอภาพเพื่อติดต่อกับผู้ใช้ได้แล้ว

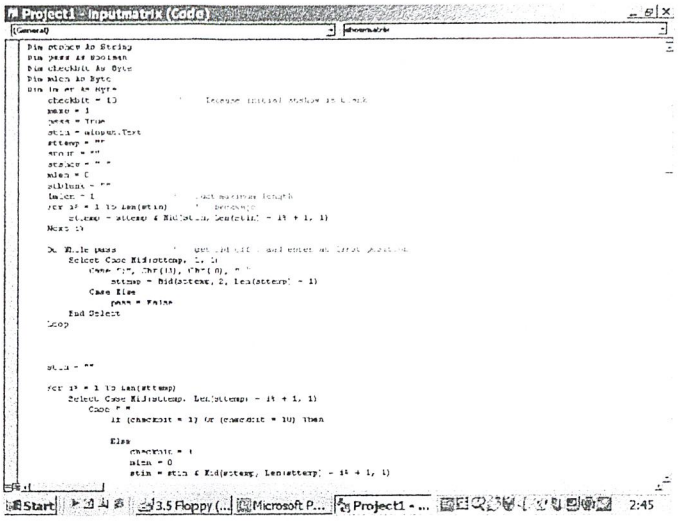


(ก)



(ข)

รูปที่ 1 (ก) แถบเครื่องมือที่ประกอบด้วยคอนโทรล ชนิดต่าง ๆ ซึ่งใช้ร่วมกับ ฟอร์ม เพื่อสร้างจอภาพของ โปรเจค
(ข) ฟอร์ม เป็นส่วนที่ใช้สำหรับสร้างจอภาพของโปรแกรมขึ้นใช้งาน



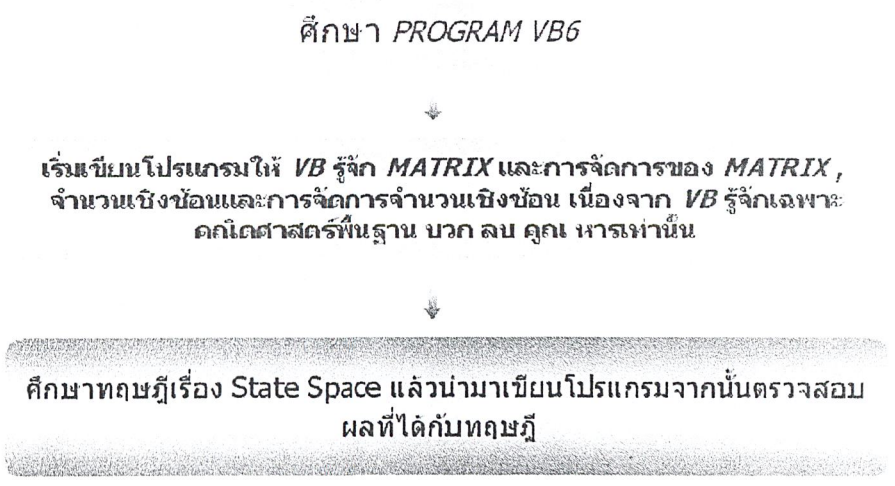
รูปที่ 2 แสดง ภาพการเขียนโปรแกรมเพื่อทำการกำหนดการทำงานให้กับ คอนโทรล ที่วางลงไป

1.3 ขั้นตอนการทำงาน

(1) ศึกษาการเขียนโปรแกรมด้วย Microsoft Visual Basic 6.0 (VB 6)

(2) เริ่มเขียนโปรแกรมให้ VB รู้จักตัว เมตริกซ์ คุณสมบัติ และ การจัดการเกี่ยวกับ เมตริกซ์ จำนวนเชิงซ้อน และ การจัดการเกี่ยวกับจำนวนเชิงซ้อน สาเหตุที่ต้องทำเนื่องมาจากว่าตัวของ VB สามารถคำนวณได้เฉพาะคณิตศาสตร์พื้นฐานเท่านั้น ซึ่งในขั้นตอนนี้ต้องใช้เวลามากพอสมควร

(3) ศึกษาทฤษฎีในเรื่อง สเตทสเปซ แล้วนำมาเขียนโปรแกรมจากนั้นตรวจสอบผลที่ได้กับทฤษฎีที่อ้างอิงถึง



รูปที่ 3 แสดงขั้นตอนในการเตรียมงานและทำโครงงาน

1.4 ขอบเขตเนื้อหางานของโครงงาน

- ความสัมพันธ์ระหว่างฟังก์ชันถ่ายโอนกับสมการสภาวะ
  - การคำนวณฟังก์ชันถ่ายโอนจากสมการสภาวะ
    - อัลกอริทึมของเลเวอเรีย (Leverrier ' s Algorithm)
- การ Realization ของระบบ LTI อินพุตเดียวเอาพุตเดียว
  - รูปแบบ Companion
    - รูปแบบ Controllable Canonical
    - รูปแบบ Observable Canonical
  - รูปแบบ Jordan Canonical
    - วิธีวนซ้ำ Householder/QR
      - Householder Reflection
      - การหา ค่า Eigen ด้วยวิธี QR Factorization
  - รูปแบบ Tridiagonal
- การคำนวณหาTransition Matrix e^{At} สำหรับระบบเชิงเส้น
  - การคำนวณหาTransition Matrix e^{At} ด้วยวิธีกระจายอนุกรมเทเลอร์
- การหาผลคำตอบของสมการสภาวะด้วยวิธี RK4
- โครงสร้างของระบบเชิงเส้น
  - ความสามารถในการควบคุมได้และความสามารถในการสังเกตได้ของระบบ LTI
  - Controllable Subspace
  - Unobservable Subspace
  - State - Variable Feedback
  - Asymptotic State Observers

บทที่ 2 การนำเสนอระบบเชิงเส้น

2.1 การนำเสนอ สเตทสเปซ (State space representation)

สมการสภาวะ เขียนเป็นรูปของกลุ่มของสมการเชิงอนุพันธ์อันดับที่ 1

(ก) ระบบเชิงเส้นที่เปลี่ยนแปลงตามเวลา (Time – variant linear system) [A(t),B(t),C(t),D(t)]

$$\dot{x}(t) = A(t)\,x(t) + B(t)\,u(t)$$
$$y(t) = C(t)\,x(t) + D(t)\,u(t) \qquad x(t_0) = x_0$$

(ข) ระบบเชิงเส้นที่ไม่เปลี่ยนแปลงตามเวลา (Time – invariant linear system) [A,B,C,D];

$$\dot{x}(t) = A\,x(t) + B\,u(t) \qquad x(t_0) = x_0$$
$$y(t) = C\,x(t) + D\,u(t)$$

เมื่อ A(t),B(t),C(t) และ D(t) (หรือ A,B,C และD) คือ เมตริกซ์ ขนาด nxn nxm pxn pxm ตามลำดับ

x(t) คือ เวคเตอร์สภาวะที่มีขนาดเท่ากับ n

y(t) คือ เวคเตอร์เอาท์พุตที่มีขนาดเท่ากับ p

u(t) คือ เวคเตอร์อินพุตที่มีขนาดเท่ากับ m

2.2 ความสัมพันธ์ระหว่าง ฟังก์ชันถ่ายโอน และ ตัวแปรสภาวะ

พิจารณา ฟังก์ชันถ่ายโอน ของ ระบบเชิงเส้นที่ไม่เปลี่ยนแปลงตามเวลา [A,B,C,D] ทำการแปลงลาปลาซ ระบบเชิงเส้นที่ไม่เปลี่ยนแปลงตามเวลา

$$s\,X(s) - x_0 = AX(s) + BU(s)$$
$$Y(s) = CX(s) + DU(s)$$

นำมาจัดรูปใหม่จะได้ว่า

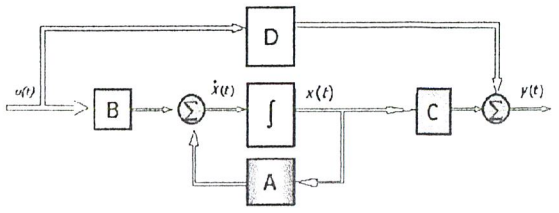
$$X(s) = (sI - A)^{-1} x_0 + (sI - A)^{-1} BU(s)$$
$$Y(s) = C(sI - A)^{-1} x_0 + [C(sI - A)^{-1} B + D]\,U(s)$$

เมื่อ I คือ เมตริกซ์เอกลักษณ์ (Identity matrix)

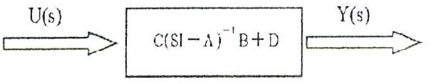
สำหรับ เงื่อนไขเริ่มต้นที่เท่ากับศูนย์ นั่นคือ $x_0 = 0$ จะได้ว่า

$$G(s) = \frac{Y(s)}{U(s)} = C(sI - A)^{-1} B + D \qquad (1)$$

ความสัมพันธ์ระหว่าง G(s) และ (A,B,C,D) แสดงได้ดังรูปที่ 4



(ก)



(ข)

รูปที่ 4 Block diagram ของ Linear Dynamical Syatem

(ก) State Variable Description

(ข) Input – Output Description (Tranfer Matrix Function)

2.3 การคำนวณ ฟังก์ชันถ่ายโอน จาก สมการสภาวะ

ในการคำนวณฟังก์ชันถ่ายโอนเราต้องการ เมตริกซ์ผกผัน ของ (sI - A) สำหรับการคิดคำนวณ $(sI - A)^{-1}$ เราสามารถหาได้จาก อัลกอริทึมของเลเวอเรีย (*Leverrier ' s Algorithm*)

Leverrier ' s Algorithm

$$\text{ให้ } (sI-A)^{-1} = \frac{\text{adj}(sI-A)}{\det(sI-A)} = \frac{P_{n-1}s^{n-1}+P_{n-2}s^{n-2}+\ldots+P_1 s+P_0}{s^n+a_{n-1}s^{n-1}+\ldots+a_1 s+a_0}$$

เมื่อ P_i คือ เมตริกซ์ที่มีขนาด nxn และ a_i คือ เลขจริง

โดย P_i และ a_i สามารถคำนวณด้วยวิธีดังนี้ คือ

$$P_{n-1} = I \qquad a_{n-1} = -\text{tr}(A)$$

$$P_{n-2} = P_{n-1}A + a_{n-1}I \qquad a_{n-2} = -\frac{1}{2}\text{tr}(P_{n-2}A)$$

$$\vdots \qquad \vdots$$

$$P_k = P_{k+1}A + a_{k+1}I \qquad a_k = -\frac{1}{n-k}\text{tr}(P_k A)$$

และ $P_i A + a_i I = \underline{0}$ เป็นเงื่อนไขสำหรับการตรวจสอบ

$\text{tr}(A) = \sum_{i=1}^{n} a_{ii}$ ผลรวมแต่สมาชิกในแนวเส้นทแยงมุมหลัก (trace of matrix)

2.4 แนวคิดในการเขียนโปรแกรมคำนวณ ฟังก์ชันถ่ายโอน จาก สมการสภาวะ

Input : A,B,C,D ที่มีขนาด nxn nxm pxn pxm ตามลำดับ

Output : อาเรย์ 3 มิติเก็บสัมประสิทธิ์ตัวตั้ง (Num) และ อาเรย์ 1 มิติเก็บตัวหาร (Den) ของ G(s)

1. ตรวจสอบขนาดของเมตริกซ์ A โดยดูจากตัวแปรที่เก็บ แถว หรือ สดมภ์ ของ A
2. กำหนดให้ *Num* เป็น อาเรย์ 3 มิติ มีขนาดดังนี้ ตัวชี้ (Index) ตัวแรก เท่ากับ n ตัวที่ 2 มีขนาดเท่ากับแถว ของ C ตัวที่ 3 มีขนาดเท่ากับสดมภ์ของ B และ *Den* จะเป็นอาเรย์ 1 มิติ ที่เก็บสัมประสิทธิ์ s

3. กำหนด ***P*** เป็น อาเรย์ 3 มิติ โดย ตัวแรกเก็บ ตัวชี้ (n) ตัวที่ 2 (n) และ 3 (n) เก็บ แถว สดมภ์ ตามลำดับ
4. กำหนด *a* เป็น อาเรย์ 1 มิติ (n)
5. กำหนด ***I*** เป็น เมตริกซ์เอกลักษณ์ที่มีขนาดเท่ากับ n x n
6. คำนวณค่า P และ a ดังต่อไปนี้

   a(1) = 1

   P(n-1) = ***I***

   a(n-2) = (-1) * (trace(**A**))

   For K% = n – 2 to 0 step -1 ' หมายเหตุ การวนลูพ ทำตามความเหมาะสมของโปรแกรม

   P(K%) = ***P***(K%+1)***A** + *a*(n+1)***I**

   A(K%) = (-1 /(n-K%)) trace(***P***(K%)***A**)

   Next K%
7. นำ **C** คูณทางซ้าย และนำ **B** คูณทางขวาของ **P** แต่ละตัว และนำผลลัพธ์เก็บไว้ใน **P** เหมือนเดิม
8. จากนั้นนำ เมตริกซ์ **D** คูณกับ อาเรย์ a ทุกตัวแล้วนำไปบวกกับ **P** ที่ ตัวชี้ เดียวกัน แล้วนำไปเก็บใน *Num* ที่ ตำแหน่งเดียวกัน และ นำ a ไปเก็บใน *Den*
9. วิธีอ่านค่าผล

   (1) ค่า *Den* จะเป็น อาเรย์ 1 มิติ ที่เก็บสัมประสิทธิ์ s แต่ละอันดับ โดยเรียงจากมากไปหาน้อย

   เช่น *Den* = [1 5 8 3 4] หมายถึง $s^4 + 5s^3 + 8s^2 + 3s + 4$

   (2) ค่า *Num* จะเป็น อาเรย์ 3 มิติ มีขนาดดังนี้ ตัวชี้ ตัวแรก เท่ากับ n ตัวที่ 2 มีขนาดเท่ากับ แถว ของ C ตัวที่ 3 มีขนาดเท่ากับ สดมภ์ ของ **B** เช่น

   $$Num = \left(\begin{bmatrix} 1 & 3 \\ 2 & 6 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 5 & -4 \end{bmatrix}, \begin{bmatrix} 7 & 9 \\ 0 & 5 \end{bmatrix}\right) \text{หมายถึง} \begin{bmatrix} s^2+7 & 3s^2+s+9 \\ 2s^2+5s & 6s^2-4s+5 \end{bmatrix}$$

บทที่ 3 Realization ของระบบเชิงเส้นที่ไม่เปลี่ยนแปลงตามเวลา SISO

พิจารณาระบบเชิงเส้นแบบ อินพุตเดียวเอาท์พุตเดียว และ G(s) ฟังก์ชันถ่ายโอน ถ้าเราสามารถหาระบบพลวัตเชิงเส้น (A, B, C, D) ที่มีขนาดจำกัดได้ และ สามารถหาฟังก์ชันถ่ายโอน G(s) ที่สอดคล้องกับระบบดังกล่าวได้ กล่าวว่า G(s) *Realizable* และ (A, B, C, D) เป็น *Realization* ของ G(s)

เมื่อกำหนด A, B, C และ D มา หา G(s) ในรูปฟังก์ชั่นตรรกยะ (Rational Function) ของ s

โดยที่ $G(s) = C(sI - A)^{-1}B + D$

- ถ้า D = 0; ฟังก์ชันถ่ายโอน $G(s) = C(sI - A)^{-1}B$ จะมีพหุนามตัวส่วนที่มีดีกรีมากกว่าตัวเศษอย่างน้อย 1 ระบบดังกล่าวเรียกว่า ***ระบบที่ถูกต้องอย่างสมบูรณ์*** *(Strictly Proper System)*
- ถ้า D เป็นค่าคงที่ $G(s) = C(sI - A)^{-1}B + D$ จะมีพหุนามตัวส่วนที่มีดีกรีมากกว่าหรือเท่ากับตัวเศษ ระบบดังกล่าวเรียกว่า ***ระบบที่ถูกต้อง*** *(Proper System)*

นั่นคือสรุปได้ว่า G(s) จะ Realizable ถ้าระบบเป็นระบบที่ถูกต้อง

ถ้า $G(s) = \dfrac{b_n s^n + b_{n-1}s^{n-1} + \ldots + b_1 s + b_0}{s^n + \alpha_{n-1}s^{n-1} + \ldots + \alpha_1 s + \alpha_0}$ (2)

$= \dfrac{\beta_{n-1}s^{n-1} + \beta_{n-2}s^{n-2} + \ldots + \beta_1 s + \beta_0}{s^n + \alpha_{n-1}s^{n-1} + \ldots + \alpha_1 s + \alpha_0} + \delta$ (3)

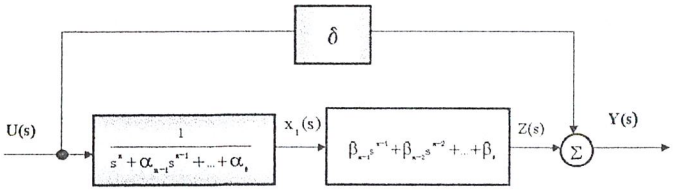
เมื่อ $\beta_i = b_i - b_n\alpha_i$ สำหรับ $i = 0, 1, \ldots, n-1$ และ $\delta = b_n$

การ Realization ที่จะนำเสนอมีทั้งหมด 4 รูปแบบ คือ รูปแบบ Controllable Canonical , รูปแบบ Observable Canonical , รูปแบบ Jordan Canonical และ รูปแบบ Tridiagonal

3.1 รูปแบบ Companion

3.1.1 รูปแบบ Controllable Canonical

จากสมการที่ (3) เราจะสามารถแยกออกเป็น 2 ส่วนได้ดังรูป



รูปที่ 5 การนำเสนอฟังก์ชันถ่ายโอน

จากรูปที่ 5 สามารถเขียนได้ว่า

$$Y(s) = Z(s) + \delta U(s)$$

เมื่อ $\dfrac{Z(s)}{U(s)} = \dfrac{\beta_{n-1}s^n + \beta_{n-2}s^{n-2} + \ldots + \beta_1 s + \beta_0}{s^n + \alpha_{n-1}s^{n-1} + \ldots + \alpha_1 s + \alpha_0}$

$$\frac{X_1(s)}{U(s)} = \frac{1}{s^n + \alpha_{n-1}s^{n-1} + \ldots + \alpha_1 s + \alpha_0}$$

$$\frac{Z(s)}{X_1(s)} = \beta_{n-1}s^{n-1} + \beta_{n-2}s^{n-2} + \ldots + \beta_0$$

ถ้า $x_1(t) = \mathcal{L}^{-1}\{X_1(s)\}$ จะได้ว่า $x_1^{(n)}(t) + \alpha_{n-1}x_1^{n-1}(t) + \ldots + \alpha_0 x_1(t) = u(t)$

เลือกให้

$$\left.\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_{n-1} &= x_n \end{aligned}\right\} \quad \text{จะได้} \quad \dot{x}_n = -\alpha_0 x_1 - \alpha_1 x_2 - \ldots - \alpha_{n-1}x_n + u(t) \qquad (4)$$

และ $y(t) = z(t) + \delta\, u(t)$

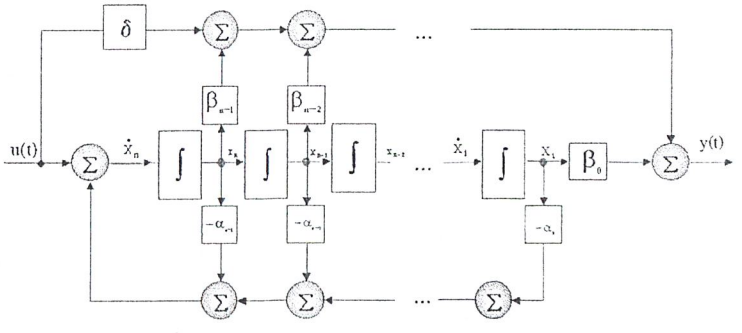
$$= \beta_0 x_1(t) + \beta_1 x_2(t) + \ldots + \beta_{n-1}x_n(t) + \delta u(t) \qquad (5)$$

จากสมการ (4) และ (5) เขียนได้ในรูป $\dot{x} = \mathbf{A}\,x(t) + \mathbf{B}\,u(t)$

$$y(t) = \mathbf{C}\,x(t) + \mathbf{D}\,x(t)$$

เมื่อ $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ -\alpha_0 & -\alpha_1 & -\alpha_2 & -\alpha_3 & \cdots & -\alpha_{n-1} \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$

$\mathbf{C} = \begin{bmatrix} \beta_0 & \beta_1 & \beta_2 & \beta_3 & \cdots & \beta_{n-1} \end{bmatrix}$ และ $\mathbf{D} = \delta$



รูปที่ 6 การ Realization ด้วยรูปแบบ Controllable Canonical

3.1.2 แนวคิดการเขียนโปรแกรมหารูปแบบ Controllable Canonical

Input : อาเรย์ 1 มิติ 2 ตัว คือ *Num* และ *Den* ซึ่งเป็นตัวเก็บสัมประสิทธิ์ของตัวตั้งและตัวหารของฟังก์ชันถ่ายโอนตามลำดับโดยมีขนาด n+1 ทั้งคู่

Output : อาเรย์ 2 มิติ จำนวน 2 ตัว คือ A โดยมีขนาด n x n

และ B ขนาด n x 1

อาเรย์ 1 มิติ จำนวน 1 ตัวคือ C ขนาด n

จำนวนเต็ม 1 ตัว คือ D

1. ให้ *Delta* = *Num*(n+1)
2. For K% = 1 to n

   Beta(K%) = *Num*(K%) – (*Num*(n+1)**Den*(K%))

   Next K%
3. การนำเอาค่ามาเรียงใน เมตริกซ์

   (3.1) A : ถ้า เมตริกซ์ A มีขนาด n ตัว

   กำหนด อาเรย์พลวัตร (Dynamic Array) *Temp* = $\underbrace{0\,0\ldots0}_{n-1\text{ ตัว}}\;\;1\;\;\underbrace{0\,0\ldots0}_{n-2\text{ ตัว}}$

   จะทำการตัด *Temp* เพื่อนำมาใส่ A ทีละแถว

   For K% = 1 to n-1

   A(K%,:) = ตัด *Temp* ตำแหน่งที่ n- K% ไปเป็นจำนวน n ตัว

   Next K%

   For K% = 1 to n

   A(n,K%) = (-1) * *Den*(K%)

   Next K%

   (3.2) B: For K% = 1 to n - 1

   B(K%,1) = 0

   Next K%

   B(n,1) = 1

   (3.3) C: For K% = 1 to n

   C(K%) = *Beta*(K%)

   Next K%

   (3.4) D: D = *Delta*

' หมายเหตุ การวนลูพ ทำตามความเหมาะสมของโปรแกรม

3.1.3 รูปแบบ Observable Canonical

ถ้าจาก รูปแบบ Controllable Canonical ของ (A, B, C, D) เป็น Realization ของ G(s) จะพบว่า (A^T, C^T, B^T, D^T) ก็จะเป็น Realization ของ G(s) ด้วย

แนวคิด

$$G(s) = C(sI - A)^{-1}B + D$$

$$B^T(sI - A^T)^{-1}C^T + D^T = B^T[(sI - A)^{-1}]^T C^T + D^T \quad ; \text{D เป็น เมตริกซ์ 1 x 1}$$

$$= [C[(sI - A)^{-1}]B]^T + D^T$$

$$= [G(s)]^T$$

$$= G(s) \quad ; \text{เป็นระบบ SISO}$$

ถ้า (A, B, C, D) เป็น Realization ของ G(s) โดย รูปแบบ Controllable Canonical แล้ว

$$\dot{x}(t) = \tilde{A}\,x(t) + \tilde{B}\,u(t)$$

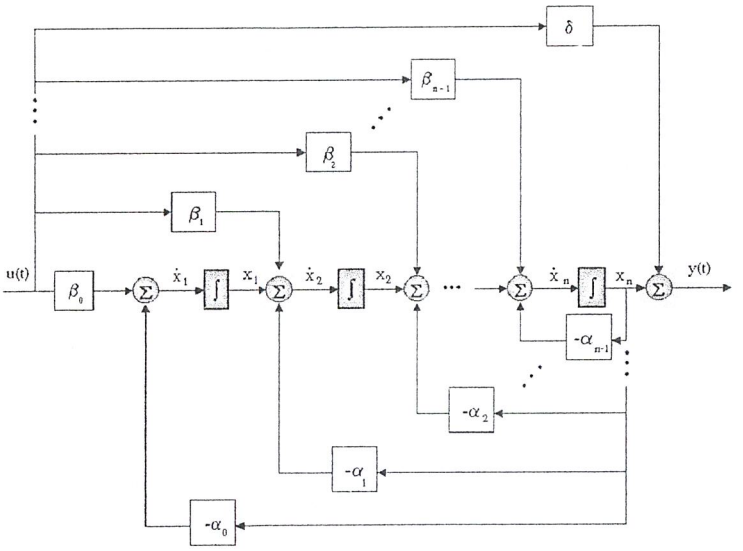
$$y(t) = \tilde{C}\,x(t) + \tilde{D}\,u(t)$$

$(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D})$ เป็น Realization ของ G(s) โดย *รูปแบบ Observable Canonical*

เมื่อ $\tilde{A} = A^T = \begin{bmatrix} 0 & 0 & \cdots & 0 & -\alpha_0 \\ 1 & 0 & \cdots & 0 & -\alpha_1 \\ 0 & 1 & \cdots & 0 & -\alpha_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -\alpha_{n-1} \end{bmatrix}$, $\tilde{B} = C^T = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{bmatrix}$

$$\tilde{C} = B^T = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \quad \tilde{D} = D = \delta$$

สามารถเขียนเป็น บล็อคไดอะแกรมแสดงการคำนวณ ได้ดังรูปที่ 7



รูปที่ 7 Realization ด้วยรูปแบบ Observable Canonical

3.2 รูปแบบ Jordan Canonical

กรณีที่ 1 ถ้า ฟังก์ชันถ่ายโอน G(s) จากสมการที่ (3) มีโพลไม่ซ้ำค่ากัน n ตัว $\lambda_1, \lambda_2, \lambda_3, \ldots \lambda_n$

G(s) สามารถกระจายด้วยเศษส่วนย่อยได้ดังนี้

$$G(s) = \frac{\gamma_1}{s-\lambda_1} + \frac{\gamma_2}{s-\lambda_2} + \cdots + \frac{\gamma_n}{s-\lambda_n} + \delta$$

สำหรับค่า $\gamma_i = \lim_{s \to \lambda_i} (s - \lambda_i)(G(s))$

กำหนด $\frac{X_K(s)}{U(s)} = \frac{1}{s - \lambda_K}$ สำหรับทุก ๆ ค่า k = 1, 2, 3, …, n

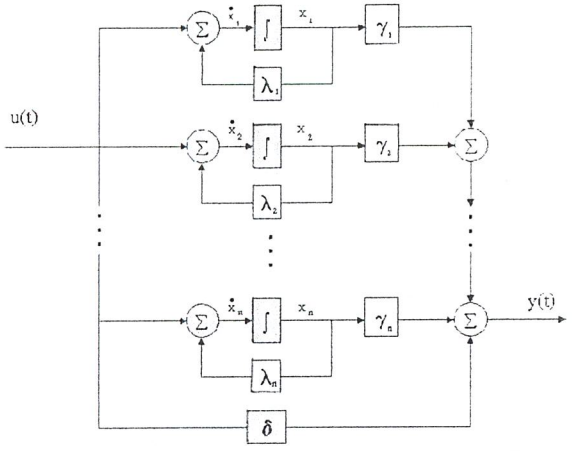
ดังนั้น $Y(s) = \gamma_1 X_1(s) + \gamma_2 X_2(s) + \gamma_3 X_3(s) + \ldots + \gamma_n X_n(s) + \delta\, U(s)$

นั้นคือเขียนในรูปสมการสภาวะ และ สมการเอาท์พุต ได้ว่า

$$\dot{x}(t) = A\,x(t) + B\,u(t)$$

$$y(t) = C\,x(t) + D\,u(t)$$

เมื่อ $A = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$, $C = \begin{bmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_n \end{bmatrix}$, $D = \delta$



รูปที่ 8 Realization ด้วยรูปแบบJordan Canonical (n) เมื่อค่าไอเก็นไม่ซ้ำกัน

กรณีที่ 2 ถ้า ฟังก์ชันถ่ายโอน G(s) จากสมการที่ (3) มีโพลซ้ำค่ากัน

สมมุติว่า G(s) สามารถกระจายด้วยเศษส่วนย่อยได้ดังนี้

$$G(s) = \frac{\gamma_{11}}{(s-\lambda_1)^3} + \frac{\gamma_{12}}{(s-\lambda_1)^2} + \frac{\gamma_{13}}{(s-\lambda_1)} + \frac{\gamma_4}{(s-\lambda_4)} + \cdots + \frac{\gamma_n}{(s-\lambda_n)} + \delta$$

เมื่อ $\gamma_{11} = \lim_{s\to\lambda_1}\left\{(s-\lambda_1)^3 G(s)\right\}$

$$\gamma_{12} = \lim_{s\to\lambda_1}\left\{\frac{\partial}{\partial s}(s-\lambda_1)^3 G(s)\right\}$$

$$\gamma_{13} = \lim_{s\to\lambda_1}\left\{\frac{\partial^2}{\partial s^2}(s-\lambda_1)^3 G(s)\right\}$$

$$\gamma_4 = \lim_{s\to\lambda_4}\left\{(s-\lambda_4)G(s)\right\}$$

กำหนด

$$\frac{X_3(s)}{U(s)} = \frac{1}{s-\lambda_1} \xrightarrow{\text{แปลงสู่ Time Domain}} \dot{x}_3(t) = \lambda_1 x_3(t) + u(t)$$

$$\frac{X_2(s)}{X_3(s)} = \frac{1}{s-\lambda_1} \xrightarrow{\text{แปลงสู่ Time Domain}} \dot{x}_2(t) = \lambda_1 x_2(t) + x_3(t)$$

$$\frac{X_1(s)}{X2(s)} = \frac{1}{s-\lambda_1} \xrightarrow{\text{แปลงสู่ Time Domain}} \dot{x}_1(t) = \lambda_1 x_2(t) + x_2(t)$$

$$\frac{X_4(s)}{U(s)} = \frac{1}{s-\lambda_4} \xrightarrow{\text{แปลงสู่ Time Domain}} \dot{x}_4(t) = \lambda_4 x_4(t) + u(t)$$

$$\vdots \qquad\qquad \vdots$$

$$\frac{X_n(s)}{U(s)} = \frac{1}{s-\lambda_n} \xrightarrow{\text{แปลงสู่ Time Domain}} \dot{x}_n(t) = \lambda_n x_n(t) + u(t)$$

$$y(t) = \gamma_{11}x_1(t) + \gamma_{12}x_2(t) + \gamma_{13}x_3(t) + \gamma_4 x_4(t) + \ldots + \gamma_n x_n(t) + \delta\, u(t)$$

นั่นคือเขียนในรูปสมการสภาวะ และ สมการเอาท์พุตได้ว่า

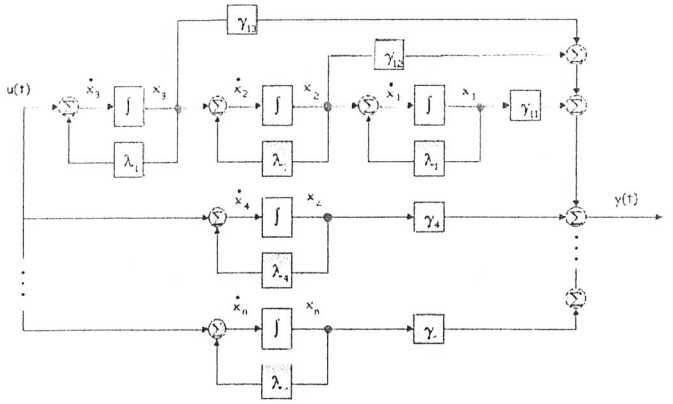
$$\dot{x}(t) = A\,x(t) + B\,u(t)$$

$$y(t) = C\,x(t) + D\,u(t)$$

เมื่อ

$$A = \begin{bmatrix} \lambda_1 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \lambda_4 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & \gamma_4 & \cdots & \gamma_n \end{bmatrix}, \quad D = \delta$$



รูปที่ 9 Realization ด้วยรูปแบบ Jordan Canonical (ข) เมื่อมีการซ้ำค่าไอเก็น

สำหรับวิธีการหาค่าไอเก็นจะเสนอ ***วิธีการวนซ้ำแบบ Householder/QR*** ซึ่งวิธีการดังกล่าวมีข้อดีคือ สามารถหาค่า ค่าไอเก็น ได้ทั้งค่าจริงและค่าเชิงซ้อนและสามารหาได้ทุกค่า

วิธีการหาค่า ค่าไอเก็น จะด้วยวิธีการวนซ้ำแบบ Householder/QR มี 2 ขั้นตอนย่อย ดังนี้

- Householder Reflection
- QR Factorization

3.3 Householder Reflection

Householder Reflection มักจะแสดงในรูปของพจน์ของการคูณโดย เมตริกซ์ Householder ซึ่งก็คือ เมตริกซ์ ที่อยู่ในรูปแบบ

$$H = I - 2ww^T$$

เมื่อ **w** คือ ยูนิตเวคเตอร์สดมภ์ (Unit (Column) Vector)

I คือ เมตริกซ์เอกลักษณ์ (Identity Matrix)

สำหรับการหา **w** จะแสดงใน อัลกอริทึม และ โฟลว์ชาร์ต

อัลกอริทึม สำหรับ เมตริกซ์ Householder

➢ กำหนด เวคเตอร์ x ขนาด n $[x_1,\ldots,x_n]^T$

เวคเตอร์ w ขนาด n $[w_1,\ldots,w_n]^T$

ตัวชี้ k โดยที่ $k \in [1, n]$

➢ จุดประสงค์การทำ หา เวคเตอร์ w ที่เป็นส่วนหนึ่งของ Householder Matrix $H = I - 2ww^T$ ในการทำให้สมาชิก เวคเตอร์ x ตำแหน่งที่ k + 1, ...,n มีค่า = 0 หรือ กล่าวว่า $Hx = [z_1,\ldots, z_k,0,0,\ldots,0]^T$

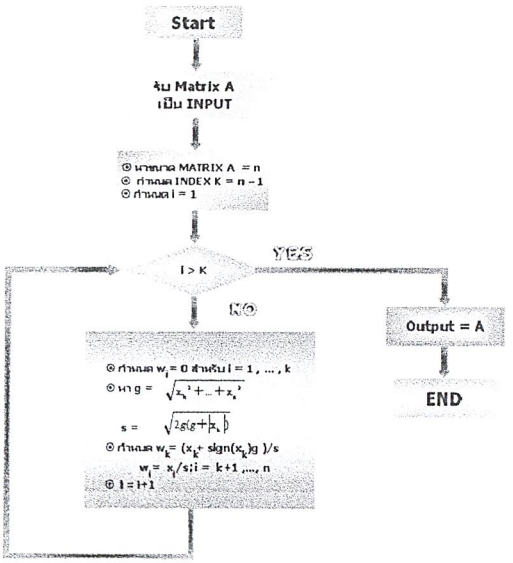
ขั้นตอนที่ 1 : กำหนด $w_i = 0$ สำหรับ i = 1 , ... , k-1

ขั้นตอนที่ 2 : หา $g = \sqrt{x_k^2 + \ldots + x_n^2}$

$s = \sqrt{2g(g + |x_k|)}$

ขั้นตอนที่ 3 : กำหนด $w_k = (x_k + \text{sign}(x_k)g)/s$

กำหนด $w_i = x_i / s$ สำหรับ $i = k+1 , \ldots , n$



รูปที่ 10 แสดง โฟลว์ชาร์ต การเขียนโปรแกรมหา เมตริกซ์ Householder

ข้อสังเกต : (1) เมตริกซ์ Householder จะนำมาคำนวณทุก ๆ แต่ละ สดมภ์ ของ **A** ยกเว้นครั้งสุดท้าย ทั้งนี้เพื่อต้องการลดสมาชิกใต้เส้นทแยงมุมหลักให้เป็นศูนย์

(2) Sign (x) = $\frac{x}{|x|}$ สำหรับ x ที่ไม่เท่ากับ 0 ถ้า x = 0 แล้ว Sign (x) = 0

(3) เมตริกซ์ **H** มีสมบัติเป็นทั้งเมตริกซ์ สมมาตรและเชิงตั้งฉาก ($H = H^{-1} = H^T$)

ตัวอย่างการนำ อัลกอริทึม ดังกล่าวมาเขียนเป็นโปรแกรม

```
Function OUTPUT = Householder(A)
    n% = length(A) ;   k% = n% - 1;
    for i% = 1 To k%,
        x = A(: , i%)   ;     w = zeros(n% , 1)  ;       g = norm(x(i% : n%));
            p = sign(x(i%))      ;      s = sqrt(2*g*(g + p*x(i%)));
            w(i%) = (x(i%) + p*g)/s ;       w(i% + 1 : n) = x(i% + 1 : n%) / s;
        H = eye(n%) - 2*w*w';
      A = H*A;
    Next i%
Output = A
```

3.4 QR Factorization

⊕ **แนวคิดการแยกตัวประกอบเมตริกซ์ A ไปเป็นรูปแบบ QR (Basic QR Factorization)**

เมตริกซ์ A สามารถแยกตัวประกอบไปเป็นรูปแบบ $A = QR$

เมื่อ Q คือ เมตริกซ์เชิงตั้งฉาก (Orthogonal Matrix) มีคุณสมบัติ คือ $Q^{-1} = Q^{T}$

R คือ เมตริกซ์สามเหลี่ยมด้านบน (Upper Triangular Matrix)

⊕ **อัลกอริทึม สำหรับ QR Factorization**

เงื่อนไข A เป็นเมตริกซ์ที่มีขนาด $n \times n$

(1) กำหนด $R^{(0)} = A$

(2) For k% = 1 To n% - 1

⊙ หา $H^{(k\%)}$ เพื่อที่จะลดตำแหน่ง k%+1 ... n% ของ สดมภ์ ที่ k ของ $R^{(k\%-1)}$ ให้เป็น 0

⊙ กำหนด $R^{(k\%)} = H^{(k\%)} R^{(k\%-1)}$

Next k%

(3) กำหนด $Q = I$

(4) For k% = n%-1 To 1 Step –1 ‘ ทำเพื่อ $Q = H^{(1)} H^{(2)} \ldots H^{(n-1)}$

$Q = H^{(k\%)} Q$

Next k%

(5) กำหนด $R = R^{(n\%-1)}$

จาก อัลกอริทึม ด้านบนสามารถเขียนเป็น โฟลว์ชาร์ต ได้ดังรูปที่ 11

3.5 แนวคิดการหา ค่าไอเก็น

(1) จาก เมตริกซ์ $B = P^{-1}AP$ จะมี ค่าไอเก็น เหมือนกับของ เมตริกซ์ A ถ้า P เป็น เมตริกซ์ไม่เอกฐาน ใด ๆ

(2) จุดมุ่งหมายในการทำ เพื่อที่จะแปลง เมตริกซ์ A ไปอยู่ในรูปที่สังเกต ค่าไอเก็นได้ง่าย

(3) ถ้า ค่าไอเก็น ของ A เป็นไปตามเงื่อนไข $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$ การวนซ้ำตาม อัลกอริทึม ที่จะกล่าวต่อไปจะลู่เข้าจนทำให้ เมตริกซ์ A กลายไปเป็นรูป เมตริกซ์ สามเหลี่ยมด้านบนที่ซึ่ง ค่า

ไอเก็น อยู่บนแนวเส้นทแยงมุม

(4) ถ้า ค่าไอเก็น ของ A เกิดค่าซ้ำกันหรือค่าเชิงซ้อน การวนซ้ำตาม อัลกอริทึม ที่จะกล่าวต่อไปจะลู่เข้าจนทำให้ เมตริกซ์ A ในแนวเส้นทแยงมุมเกิดบล็อคในแนวเส้นทแยงมุม เช่น

$$\begin{bmatrix} D & X & X & X & X & X & X \\ 0 & D & X & X & X & X & X \\ 0 & 0 & D & D & X & X & X \\ 0 & 0 & D & D & X & X & X \\ 0 & 0 & 0 & 0 & D & D & X \\ 0 & 0 & 0 & 0 & D & D & X \\ 0 & 0 & 0 & 0 & 0 & 0 & D \end{bmatrix}$$

เมื่อ D และ X คือ สมาชิกที่ไม่เท่ากับศูนย์

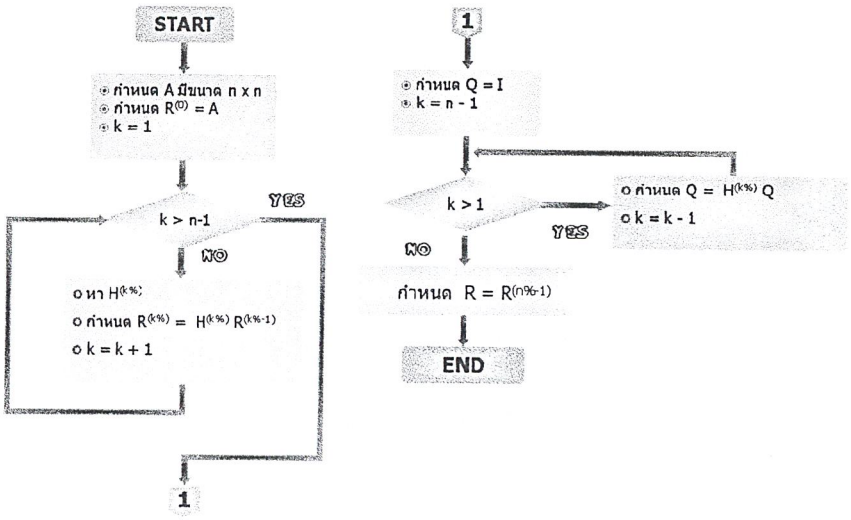
D อาจพิจารณาจาก 1×1 หรือ 2×2 บล็อคแนวทแยงมุม (Square Submatrices)

(5) ค่าไอเก็น จากข้อ (4) จะแบ่งเป็น 2 กรณี คือ

(5.1) ถ้าเป็น สมาชิกค่าเดี่ยวตามแนวเส้นทแยงมุมแล้วค่าไอเก็น เท่ากับตัวมันเอง

(5.2) ถ้าเป็น เมตริกซ์ย่อยตามแนวทแยงมุมขนาด 2×2 แสดงว่าเกิด ค่าไอเก็น ที่ซ้ำกัน หรือ ค่าไอเก็นที่เป็นคู่ จำนวนเชิงซ้อนสังยุค ซึ่งเวลาคำนวณก็คิดเหมือนปรกติคือ $|\lambda I - \tilde{D}| = 0$

เมื่อ $\tilde{D}$ คือ 2×2 Diagonal Submatrices จากนั้นหาค่า λ ออกมา



รูปที่ 11 แสดง โฟลว์ชาร์ต การเขียนโปรแกรมหา QR Factorization

⊕อัลกอริทึม ในการหา ค่าไอเก็น โดยการใช้ QR Factorization

เงื่อนไข A เป็นเมตริกซ์ที่มีขนาด $n \times n$

(1) กำหนด $A^{(1)} = A$

(2) For k = 1 to จำนวนครั้งที่ต้องการให้วนมากที่สุด (ยิ่งมากความละเอียดยิ่งสูง)

◉ แยก $A^{(k)} = Q^{(k)} R^{(k)}$ ‘ จาก อัลกอริทึม QR Factorization

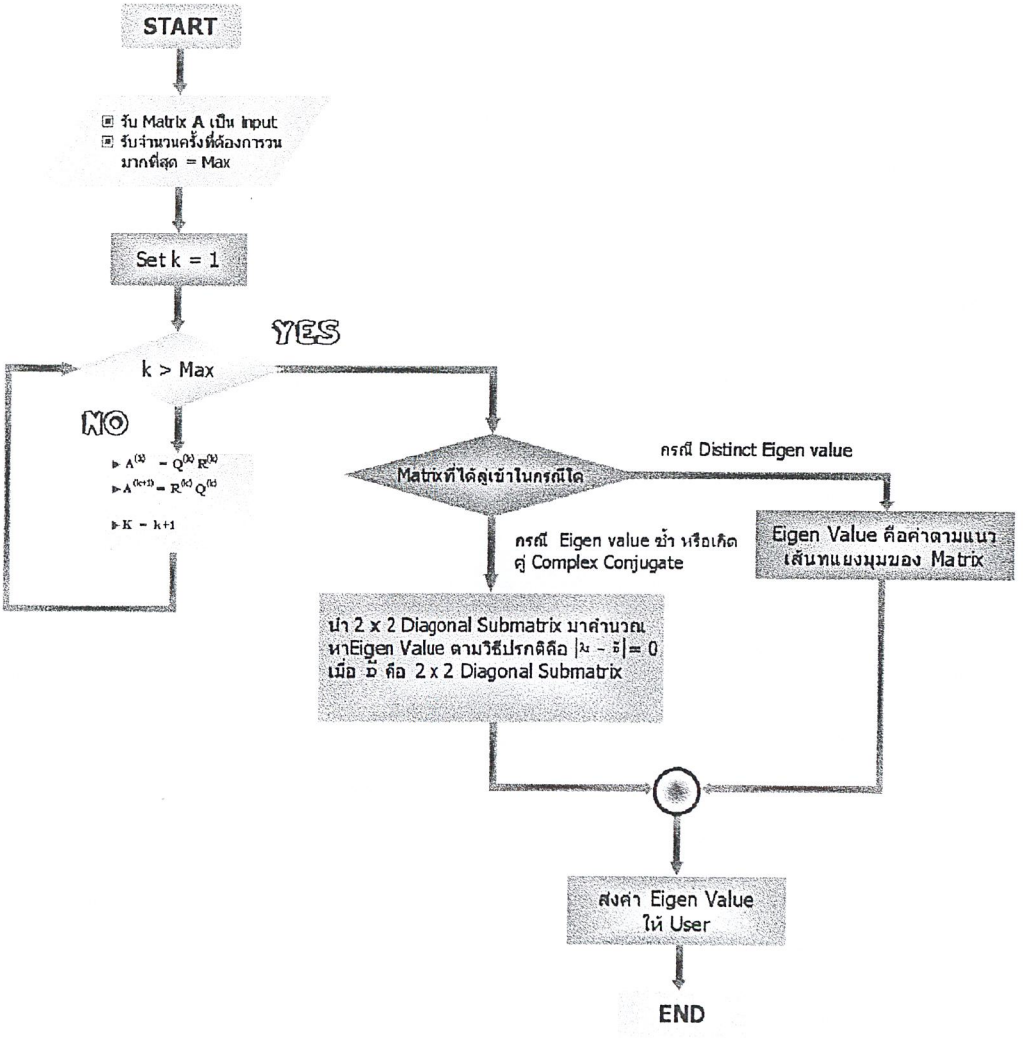
‘ เนื่องจาก Q เป็นเมตริกซ์เชิงตั้งฉาก ดังนั้น $Q^T = Q^{-1}$; $[Q^{(k)}]^T A^{(k)} = R^{(k)}$

◉ กำหนด $A^{(k+1)} = R^{(k)} Q^{(k)}$ ‘ เป็นการแสดงให้เห็นว่า ค่าไอเก็น ไม่เปลี่ยนแปลง

‘ เนื่องจาก $A^{(k+1)} = [Q^{(k)}]^T A^{(k)} Q^{(k)}$

Next k

สรุปแนวคิดทั้งหมดของการหา ค่าไอเก็น เป็น โฟลว์ชาร์ต ดังรูปที่12



รูปที่ 12 แสดง โฟลว์ชาร์ต การเขียนโปรแกรมหา ค่าไอเก็น

ตัวอย่างการนำ อัลกอริทึม ดังกล่าวมาเขียนเป็นโปรแกรม

```
function [B] = test_Eigenvalue(A,max)
n = length(A); k = n-1;
for z = 1 : max
    R = A;   Q = eye(n);
        for i = 1 : k,
            x = A(:,i);   w = zeros(n,1); g = norm(x(i:n)); p = sign(x(i));
            s = sqrt(2*g*(g+p*x(i))); w(i)=(x(i)+p*g)/s; w(i+1:n)=x(i+1:n)/s;
            H = eye(n)- 2*w*w'; A = H*A; R = H*R; Q = Q*H;
        End
A = R*Q;
end
B = A;
```

ตัวอย่างผลที่ได้จากแนวคิดการหา ค่าไอเก็น ด้วยวิธีนี้

$$A = \begin{bmatrix} 5.3 & 2.3 & 4.6 & 2.7 & 1.6 & 2.2 \\ 2.4 & 7.8 & 5.7 & 8.4 & 3.4 & 4.2 \\ 3.4 & 5.6 & 2.4 & 1.7 & 7.4 & 3.9 \\ 8.3 & 7.5 & 9.2 & 6.1 & 5.2 & 7.9 \\ 4.3 & 5.9 & 7.2 & 2.6 & 4.9 & 0.8 \\ 0.9 & 2.7 & 4.9 & 4.8 & 6.7 & 4.8 \end{bmatrix}$$

ถ้าจากการวนด้วย อัลกอริทึม ด้านบน 1500 ครั้งจะได้ผลดังนี้

$$\tilde{A} = \begin{bmatrix} 28.3953 & 1.0604 & 5.2207 & -3.7265 & -0.0031 & 5.2596 \\ 0.0000 & 2.4765 & 3.7496 & 1.0038 & 2.1814 & 0.3748 \\ -0.0000 & -3.3680 & 3.4624 & 0.3205 & 0.9822 & 0.1544 \\ -0.0000 & 0 & 0.0000 & -2.3834 & -1.7853 & 4.8820 \\ -0.0000 & 0.0000 & 0 & 0.7362 & -3.7670 & -1.3417 \\ -0.0000 & -0.0000 & 0.0000 & -0.0000 & -0.0000 & 3.1161 \end{bmatrix}$$

ถ้าใช้ Function eig(A) ของ MATLAB® จะได้คำตอบเป็น 28.3953 , -3.0752 $\pm$ 0.9142i , 3.1161 และ 2.9695 $\pm$ 3.5193i ซึ่งจากผลที่ได้จาก อัลกอริทึม มีค่าตรงกัน

3.6 รูปแบบ Tridiagonal

จาก G(s) สมการที่ (3) $G(s) = \dfrac{\beta_{n-1}s^{n-1} + \beta_{n-2}s^{n-2} + \ldots + \beta_1 s + \beta_0}{s^n + \alpha_{n-1}s^{n-1} + \ldots + \alpha_1 s + \alpha_0} + \delta$

$$= \delta + \cfrac{1}{b_1 + a_1 s + \cfrac{1}{b_2 + a_2 s + \cfrac{1}{\ddots \quad \ddots} \\ \qquad\qquad \cfrac{1}{b_{n-1} + a_{n-1}s + \cfrac{1}{b_n + a_n s}}}}$$

$\forall a_i \neq 0$ สำหรับ $i = 1, 2, \ldots, n$

กำหนด ตัวแปรสภาวะ

$$\frac{X_n(s)}{X_{n-1}(s)} = \frac{1}{b_n + a_n s}$$

$$\frac{X_{n-1}(s)}{X_{n-2}(s)} = \frac{1}{b_{n-1} + a_{n-1}s + \dfrac{X_n(s)}{X_{n-1}(s)}}$$

$$\vdots \qquad\qquad \vdots$$

$$\frac{X_2(s)}{X_1(s)} = \frac{1}{b_2 + a_2 s + \dfrac{X_3(s)}{X_2(s)}}$$

$$\frac{X_1(s)}{U(s)} = \frac{1}{b_1 + a_1 s + \dfrac{X_2(s)}{X_1(s)}}$$

$$Y(s) = X_1(s) + \delta U(s)$$

เขียนในรูปสมการสภาวะ

$$\dot{x}_n(t) = \frac{1}{a_n} x_{n-1}(t) - \frac{b_n}{a_n} x_n(t)$$

$$\dot{x}_{n-1}(t) = \frac{1}{a_{n-1}} x_{n-2}(t) - \frac{b_{n-1}}{a_{n-1}} x_{n-1}(t) - \frac{1}{a_{n-1}} x_n(t)$$

$$\vdots$$

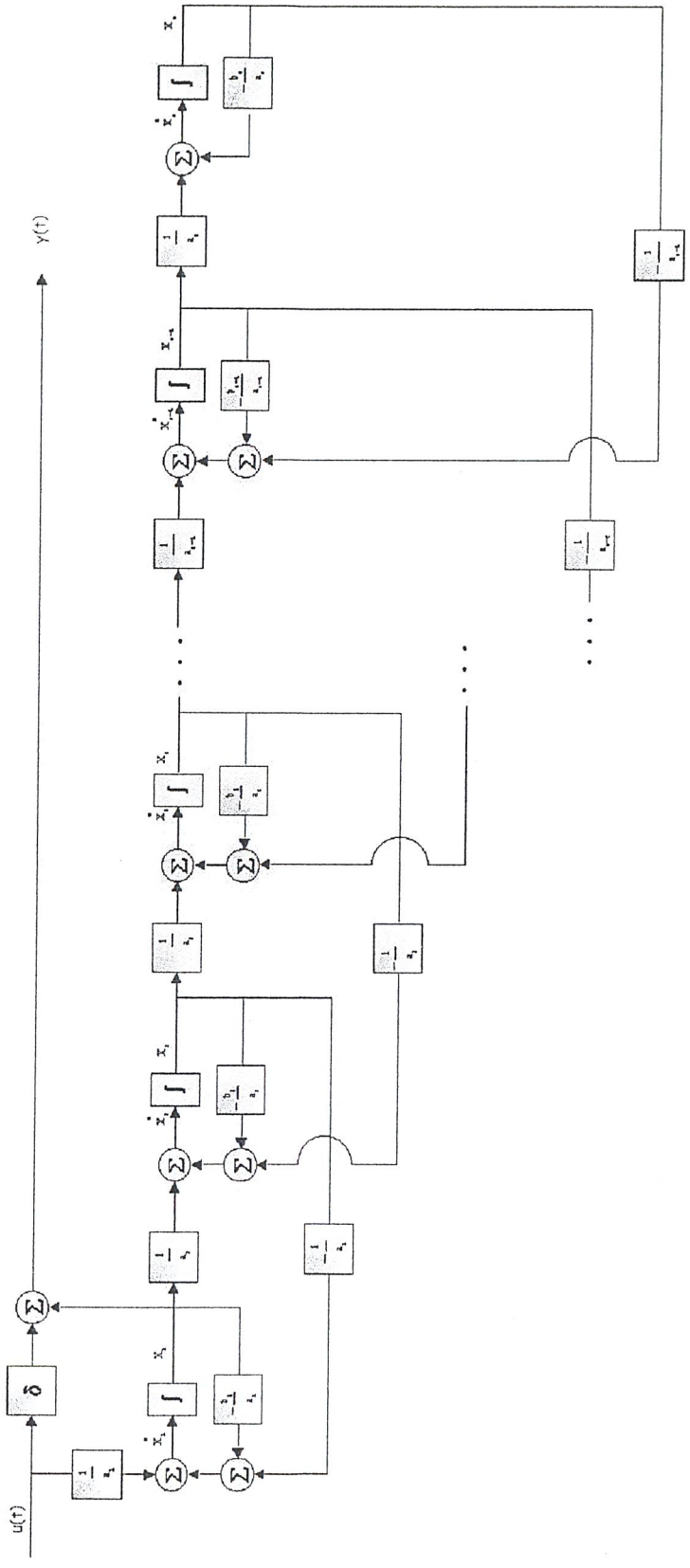
$$\dot{x}_2(t) = \frac{1}{a_2} x_1(t) - \frac{b_2}{a_2} x_2(t) - \frac{1}{a_2} x_3(t)$$

$$\dot{x}_1(t) = \frac{-b_1}{a_1} x_1(t) - \frac{1}{a_1} x_2(t) - \frac{1}{a_1} u(t)$$

$$A = \begin{bmatrix} -\frac{b_1}{a_1} & -\frac{1}{a_1} & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{a_2} & -\frac{b_2}{a_2} & -\frac{1}{a_2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{a_{n-1}} & -\frac{b_{n-1}}{a_{n-1}} & -\frac{1}{a_{n-1}} \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{a_n} & -\frac{b_n}{a_n} \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{a_1} \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}, \quad D = \delta$$

ซึ่งสามารถแสดงเป็น บล็อคไดอะแกรมแสดงการคำนวณในรูปแบบ Tridiagonal ได้ดังรูปที่ 13



รูปที่ 13 Realization ด้วยรูปแบบ Tridiagonal

บทที่ 4 ผลคำตอบของสมการสภาวะเชิงเส้นที่ไม่เปลี่ยนแปลงตามเวลา

4.1 การคำนวณหาTransition Matrix e^{At} สำหรับระบบเชิงเส้น

จากการพิสูจน์พบว่า เมตริกซ์เอกซ์โพแนนเชียล ของ เมตริกซ์ A ขนาด n×n

$$e^{At} \triangleq \sum_{k=0}^{\infty} \frac{A^k t^k}{k!}$$

และยิ่งไปกว่านั้นหากมีเวลาที่แน่นอน เมตริกซ์ e^{At} จะเป็น เมตริกซ์ ที่ลู่เข้าซึ่งสำหรับในหัวข้อนี้จะนำเสนอการคำนวณ เมตริกซ์ e^{At} ด้วยวิธีการกระจายในรูป อนุกรมเทย์เลอร์

การคำนวณ e^{At} ด้วยวิธีกระจายในรูปของอนุกรมเทย์เลอร์

แนวคิด

จากการกระจาย e^{At} ในรูปอนุกรมเทย์เลอร์ได้ผลดังนี้

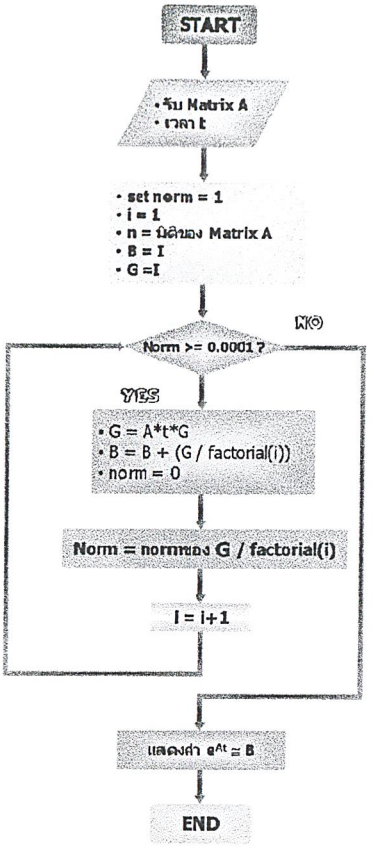
$$e^{At} = I + At + \frac{A^2t^2}{2!} + \frac{A^3t^3}{3!} + \ldots + \frac{A^nt^n}{n!} + \ldots$$

$$= I + (At) + \frac{At}{2}\left(\frac{At}{1!}\right) + \frac{At}{3}\left(\frac{A^2t^2}{2!}\right) + \ldots + \frac{At}{n}\left(\frac{A^{n-1}t^{n-1}}{(n-1)!}\right) + \ldots$$

ซึ่งเป็นที่สังเกตได้ว่าพจน์ที่อยู่ในวงเล็บของสมการด้านบน คือ พจน์ที่อยู่ก่อนหน้า ซึ่งจากข้อสังเกตดังกล่าวเราจะนำเอาไปเป็นส่วนหนึ่งของแนวการเขียนโปรแกรม สำหรับขอบเขตของการคำนวณด้วยวิธีนี้คือ เราจะใช้ ***นอร์มของเมตริกซ์*** เป็นตัวบ่งชี้ว่าเมื่อใดจะหยุดการคำนวณ

หมายเหตุ $\text{Norm of M} = \| M \| = \sum_{i=1}^{n} \left| m_{ij} \right|$

```
function [B,i,norm] = trantaylor(A,t)
n = length(A);
B = eye(n);
G = eye(n);
H = eye(n);
norm = 1;
i = 1;
while   norm >= 0.00001;
   G = (A*t)*G;
   H = G/fac(i);
   B = B+H;
   norm = 0;
   for   k = 1:n;
      for w = 1:n;
         norm = norm + abs(H(k,w));
      end
   end
   i = i+1;
end
i = i-1;
```



รูปที่ 14 โฟลว์ชาร์ตแสดงการเขียนโปรแกรมเพื่อหา e^{At}

4.2 การหาผลคำตอบของสมการสภาวะเชิงเส้นที่ไม่เปลี่ยนแปลงตามเวลา

จากสมการสภาวะ $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t)$ (6)

เมื่อ $\mathbf{x}(t)$: เวคเตอร์สภาวะที่มีขนาดเท่ากับ n

$u(t)$: เวคเตอร์อินพุตที่มีขนาดเท่ากับ m

$\mathbf{A}$: เมตริกซ์ค่าคงที่ขนาดท่ากับ $n \times n$

$\mathbf{B}$: เมตริกซ์ค่าคงที่ขนาดท่ากับ $n \times m$

เมื่อนำสมการที่ (6) มาจัดรูปใหม่จะได้ว่า ;

$$\dot{\mathbf{x}}(t) - \mathbf{A}\mathbf{x}(t) = \mathbf{B}u(t)$$

$$e^{-\mathbf{A}t}[\dot{\mathbf{x}}(t) - \mathbf{A}\mathbf{x}(t)] = e^{-\mathbf{A}t}\mathbf{B}u(t)$$

$$\frac{d}{dt}[e^{-\mathbf{A}t}\mathbf{x}(t)] = e^{-\mathbf{A}t}\mathbf{B}u(t)$$

$$e^{-\mathbf{A}t}\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t e^{-\mathbf{A}\tau}\mathbf{B}u(\tau)d\tau$$

หรือ $$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}u(\tau)d\tau \quad (7)$$

จากสมการที่ (7) ถ้ากำหนด เวลาเริ่มต้นเป็น t_0 , ผลคำตอบ สมการที่ (7) จะปรับเปลี่ยนเป็น

$$x(t) = e^{-A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{-A(t-\tau)}Bu(\tau)d\tau \qquad (8)$$

การคำนวณผลคำตอบของสมการสภาวะ ด้วยวิธี รุงเง- กุตตา อันดับที่ 4

จาก การแก้สมการเชิงอนุพันธ์ ในรูป $\frac{dx}{dt} = f(t,x)$ เมื่อ $x(0) = x_0$

สำหรับผลคำตอบเมื่อใช้วิธี วิธี รุงเง- กุตตา อันดับที่ 4 จะได้ว่า

$$\boxed{x_{i+1} = x_i + \frac{1}{6}(k_1 + k_2 + k_3 + k_4)} \qquad (9)$$

ถ้ากำหนด ช่วงการเพิ่มของเวลา (Incremental time) $t_{i+1} - t_i = h$ แล้ว

$$k_1 = hf(t_i, x_i) \qquad k_3 = hf(t_i + \frac{1}{2}h, x_i + \frac{1}{2}k_2)$$

$$k_2 = hf(t_i + \frac{1}{2}h, x_i + \frac{1}{2}k_1) \qquad k_4 = hf(t_i + h, x_i + k_3)$$

ตัวอย่างการเขียนโปรแกรม หาผลคำตอบของ สมการสภาวะ ด้วยวิธี RK4

```
function [x] = solution(A,B,x0,chooseu,k);
% [x] = solution(A,B,x0,chooseu,k)
% solve by Forth-Order Runge-Kutta methods
% u = 1 mean step function
% u = 2 mean ramp
% u = 3 mean parabolic input
% k = coeff of u(t)
t= [0:0.1:10];
switch chooseu
case 1
      for i = 1:101
            u(i,1) = k*1;
      end;
case 2
      for i = 1:101
            u(i,1) = k *t(i);
      end;
case 3
      for i = 1:101
            u(i,1) = 0.5*(t(i)*t(i))*k;
      end;
otherwise
      display('please read comment');
end;
n = length(A)
for i = 1:n
      x(i,1) = 0;
end;
h = 0.1;
for i = 1:100
      k1 = h*(A*x(:,i)+B*u(i));
      k2 = h*(A*(x(:,i)+0.5*k1)+B*u(i));
      k3 = h*(A*(x(:,i)+0.5*k2)+B*u(i));
      k4 = h*(A*(x(:,i)+k3)+B*u(i));
      x(:,i+1) = x(:,i) + (1/6)*(k1 + 2*k2 + 2*k3 + k4 );
end;
```

<u>หมายเหตุ</u> - เพื่อความสะดวกของผู้ใช้ในโปรแกรมที่เขียนจะมีการให้ผู้ใช้เลือก อินพุตทดสอบมาตรฐาน
ได้แก่ (ก) <u>อินพุตขั้นบันได</u>

$u(t) = k \quad \text{for } t > 0$

$u(t) = 0 \quad \text{for } t < 0$

(ข) <u>อินพุตแรมป์</u>

$u(t) = kt \quad \text{for } t > 0$

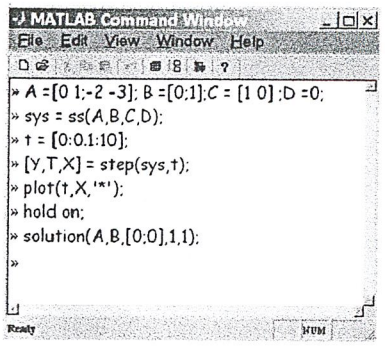
$u(t) = 0 \quad \text{for } t < 0$

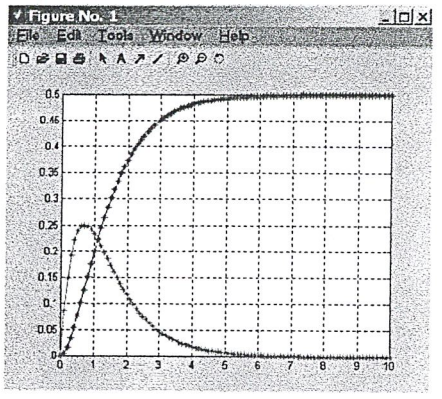
(ค) <u>อินพุตพาราโบลิค</u>

$u(t) = k \times \frac{1}{2} \times t^2 \quad \text{for } t > 0$

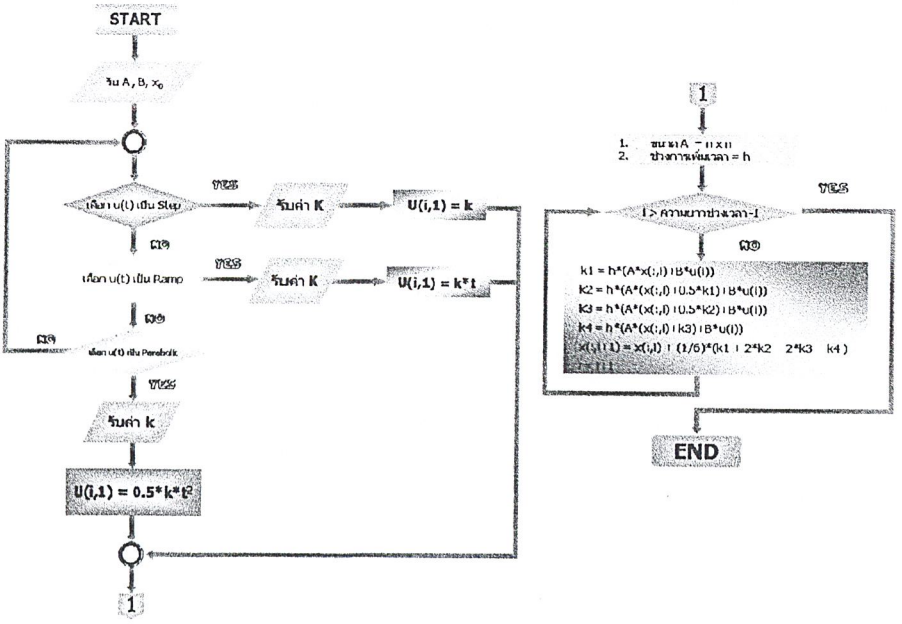
$u(t) = 0 \quad \text{for } t < 0$

ตรวจสอบผลการเขียน อัลกอริทึม ดังกล่าวกับคำสั่ง [Y,T,X] = STEP(SYS) ของ MATLAB®





สังเกตได้ว่ากราฟของคำสั่ง STEP กับที่เขียนขึ้นเองทับกันสนิทนั่นคือ อัลกอริทึมที่เขียนขึ้นนั้นถูกต้อง (กราฟ ที่เป็นเส้นทึบ คือ อัลกอริทึม ที่เขียนขึ้นเอง ส่วนกราฟที่เป็นลาย * จะเป็นกราฟที่ใช้จากคำสั่ง STEP ของ MATLAB®)



รูปที่ 15 โฟลว์ชาร์ต แสดงการเขียนโปรแกรมเพื่อหา ผลคำตอบของ สมการสภาวะ โดย RK4

บทที่ 5 โครงสร้างของระบบเชิงเส้น

5.1 Controllability and Observability of time-invariant systems

5.1.1 ความสามารถในการควบคุมได้ (Controllability)

พิจารณาระบบที่ไม่เปลี่ยนแปลงตามเวลา

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{10}$$

เมื่อ $\mathbf{x}$, $\mathbf{y}$, $\mathbf{u}$ คือ เวคเตอร์ ขนาด n , m และp ตามลำดับ

จากระบบ (10) หากมี อินพุต u(t) ที่ซึ่งสามารถเคลื่อนสภาวะ เริ่มต้น $\mathbf{x}(t_0) = \mathbf{x}_0$ ไปยัง สภาวะสุดท้าย $\mathbf{x}(t_f)$ บนช่วงเวลาจำกัด $t_f - t_0$, State $\mathbf{x}_0$ กล่าวว่า ***สามารถควบคุมได้ (Controllable)*** และถ้าทุก ๆ สภาวะ เริ่มต้น สามารถควบคุมได้ ระบบจะกล่าวว่า ***สามารถควบคุมได้อย่างสมบูรณ์***

◉ เงื่อนไขของการตรวจสอบ (A,B) ว่าสามารถควบคุมได้

ระบบ (10) จะสามารถควบคุมได้อย่างสมบูรณ์ถ้าแร็งค์ของ ***เมตริกซ์ Controllability*** ที่นิยามโดย

$$\mathcal{C} \triangleq [\,B \quad AB \quad A^2B \quad \dots \quad A^{n-1}B\,]$$

มีค่าเท่ากับ n

5.1.2 ความสามารถในการสังเกตได้ (Observability)

ถ้าเอาท์พุตที่วัดได้จากระบบในช่วงเวลาที่จำกัด สามารถนำไปกำหนดสภาวะเริ่มต้น x_0 ของระบบได้, สภาวะ $\mathbf{x}_0$ กล่าวว่า ***สามารถสังเกตได้ (Observable)*** และถ้าทุก ๆ State ของระบบ Observable ระบบจะกล่าวว่า ***สามารถสังเกตได้อย่างสมบูรณ์***

◉ เงื่อนไขของการตรวจสอบ (A,C) ว่าสามารถสังเกตได้

ระบบเชิงเส้นระบบหนึ่งจะกล่าวว่า สามารถสังเกตได้อย่างสมบูรณ์ ถ้า แรงค์ ของ ***เมตริกซ์ Observability*** ที่นิยามโดย

$$\mathcal{O} \triangleq \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

มีค่าเท่ากับ n

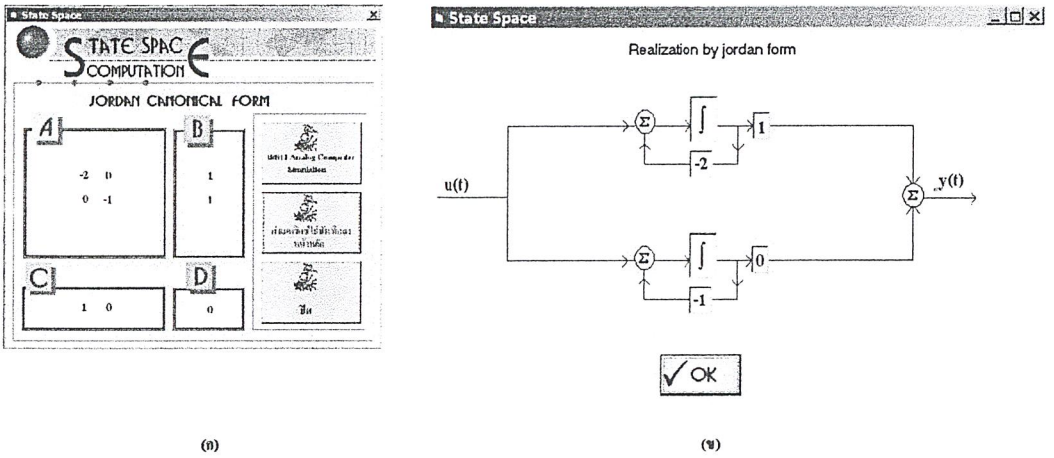
จากบทพิสูจน์บทหนึ่งกล่าวว่าระบบเชิงเส้นอินพุตเดียวเอาท์พุตเดียวที่ไม่เปลี่ยนแปลงตามเวลา จะทั้งสามารถควบคุมได้ และ สามารถสังเกตได้ก็ต่อเมื่อพหุนามตัวเศษและพหุนามตัวส่วนของฟังก์ชันถ่ายโอนไม่มีตัวประกอบ ร่วมกันซึ่งจากบทพิสูจน์ดังกล่าวแสดงได้ด้วยการ Realization ไปสู่ รูปแบบ Jordan Canonical ซึ่งหากเกิดกรณี ตัวประกอบร่วมกัน (Pole –zero cancellations)

เมื่อ Realization ไปสู่ รูปแบบ Jordan Canonical แล้วจะทำให้สภาวะไม่สามารถสังเกตได้

เช่น

$$G(s) = \frac{Y(s)}{U(s)} = \frac{s+1}{s^2+3s+2} = \frac{s+1}{(s+1)(s+2)} = \frac{1}{s+2}$$

เกิดการ Cancellation ของ Factor (s+1) , นั่นคือ Mode ที่ s = – 1 ที่เป็นตัวที่ให้เกิดพจน์ Ke^{-t} ซึ่งถึงแม้ว่าจะมีอยู่ในระบบก็ตามแต่กลับไม่ไปปรากฏที่ Output ดังแสดงได้โดยการใช้ Software ที่เขียนขึ้น



(ก) (ข)

รูปที่ 16 (ก) – (ข) แสดงการRealization JCF เพื่อแสดงว่าเมื่อเกิดการ pole-zero Cancellation แล้ว ใน JCF จะUnobservable

ซึ่งจาก 16(ก) จะได้ว่า เมตริกซ์ Observability $\mathcal{O} = \begin{bmatrix} 1 & 0 \\ -2 & 0 \end{bmatrix}$ ซึ่งสังเกตได้ชัดว่า Rank($\mathcal{O}$) = 1 ซึ่งไม่ Full rank นั่นคือ ไม่สามารถสังเกตได้

โดยทั่วไปแล้วถ้าตัวเศษและตัวส่วนของฟังก์ชันถ่ายโอนมีตัวประกอบร่วมกันเราสามารถเขียนสมการสภาวะ ได้ทั้งใน รูปแบบ Controllable Canonical และ รูปแบบ Observable Canonical ซึ่งถ้าเขียนสมการสภาวะใน รูปแบบ Controllable Canonical จะไม่สามารถสังเกตได้อย่างสมบูรณ์ และในทำนองเดียวกันถ้าเขียนสมการสภาวะในรูปแบบ Observable Canonical จะไม่สามารถควบคุมได้อย่างสมบูรณ์ (แสดงแนวคิดนี้ในส่วนของการใช้งานของ Software)

นั่นคือ สมการสภาวะ จะทั้ง ควบคุมได้ และ สังเกตได้ ถ้าไม่เกิด การตัดกันของโพลและซีโร่ ในกรณีดังกล่าวนี้ สมการสภาวะ กล่าวว่าเป็น **Realization ที่ต่ำที่สุด** ของ ฟังก์ชันถ่ายโอน แล้ว

5.1.3 ตัวอย่างการเขียนโปรแกรมเพื่อตรวจสอบ Controllability และ Observability

โปรแกรมการหา Rank(A)

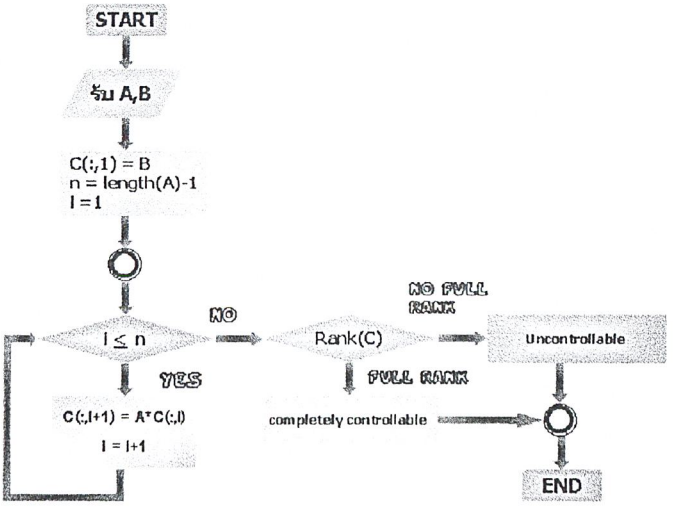
```
function [output] = myrank(A);

i = 0;
pass = 1;
B = A;
ans = length(A)+1;
while (i ~= length(A)) & (pass)
for j = 1:i+1
for k = 1:i+1
B = A(j:length(A)-i+j-1,k:length(A)-i+k-1);
if det(B) ~= 0
pass = 0;
end;
end;
end;
i = i+1;
ans = ans-1;
end;
if pass
ans = 0;
end;
output = ans;
```

โปรแกรมการตรวจสอบ Controllability

```
function [C,rank _C,output]= control _test(A,B);
C(:,1) = B;
for i =1:length(A)-1;
      C(:,i+1)= A*C(:,i);
end;
rank_ C = myrank(C);
if rank_ C ~= length(A)
      output = 'Uncontrollable';
else
      output = 'completely controllable';
end
```

โปรแกรมการตรวจสอบ Observability

```
function [O,rank_O,output]= observe_test(A,C)
O(1,:)= C;
for i =1:length(A)-1;
      O(i+1,:)=     O(i,:)*A;
end;
rank_O = myrank(O);
if rank_O     ~= length(A)
      output = 'Unobservable';
else
      output = 'completely observable';
end
```



รูปที่ 17 แสดง โฟลว์ชาร์ตโปรแกรม การตรวจสอบความสามารถในการควบคุมได้

5.2 Controllable Subspace

จาก ระบบเชิงเส้นที่ไม่เปลี่ยนแปลงตามเวลา

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t) \qquad (11)$$

เมื่อ $\mathbf{x}$, $\mathbf{y}$, $\mathbf{u}$ คือ เวคเตอร์ ขนาด n , m และp ตามลำดับ

ถ้าจากระบบดังกล่าว สมมุติว่าไม่สามารถควบคุมได้อย่างสมบูรณ์ นั่นคือ แม้ว่าจะไม่ใช่ทุก ๆ สภาวะ จะสามารถควบคุมได้ แต่ก็ยังมีบางสภาวะ ที่ยังสามารถควบคุมได้ อยู่

พิจารณาผลคำตอบของระบบที่ (11)

$$x(t) = e^{-At}x_0 + \int_0^t e^{-A(t-\tau)}Bu(\tau)d\tau \qquad (12)$$

ถ้าระบบสามารถควบคุมได้ จะต้องมีอินพุต ที่ซึ่งทำให้ $x(t_1) = x_1 = 0$ ที่เวลาจำกัด $t - t_1$ นั่นก็คือ

$$-x_0 = \int_0^{t_1} e^{-A\tau}Bu(\tau)d\tau$$

ซึ่ง x_0 นี้เราถือว่าเป็น *สภาวะที่สามารถควบคุมได้*

ถ้ากำหนด Set ของ สภาวะที่สามารถควบคุมได้ $\mathcal{X}_c$ แล้วให้ชื่อว่า ***Controllable Subspace***

$$\mathcal{X}_c = \left\{ x_0 \;\middle|\; x_0 = -\int_0^t e^{-A\tau}Bu(\tau)d\tau \;, \forall u \right\}$$

สำหรับระบบที่ไม่สามารถควบคุมได้อย่างสมบูรณ์, Rank($\mathcal{C}$) = $n_c < n$, จะได้ Basis เวคเตอร์ ของ $\mathcal{X}_c$ คือ $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{nc}$ ซึ่งได้มาจากการเลือก เวคเตอร์ที่อิสระเชิงเส้น จาก $\mathcal{C} = [\, B \quad AB \quad A^2B \quad \ldots \quad A^{n-1}B \,]$ นั่นคือจะมี $x_c \in \mathcal{X}_c$ เมื่อ

$$x_c = \sum_{i=1}^{n_c} v_i \tilde{x}_i \quad , \; \tilde{x}_i \text{ คือ } Scalar$$

นอกจาก $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{nc}$ แล้วจะยังต้องมี เวคเตอร์ที่อิสระเชิงเส้น $\mathbf{v}_{nc+1}, \mathbf{v}_{nc+2}, \ldots, \mathbf{v}_n$ เพื่อที่ว่า $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ จะได้สร้าง Basis สำหรับ เวคเตอร์สเปซ n มิติ ขึ้นมา

$$x = \sum_{i=1}^{n} v_i \tilde{x}_i = \begin{bmatrix} v_1, v_2, \ldots, v_n \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_n \end{bmatrix}$$

$$= \begin{bmatrix} v_1, v_2, \ldots, v_{nc} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_{nc} \end{bmatrix} + \begin{bmatrix} v_{nc+1}, v_{nc+2}, \ldots, v_n \end{bmatrix} \begin{bmatrix} \tilde{x}_{nc+1} \\ \tilde{x}_{nc+2} \\ \vdots \\ \tilde{x}_n \end{bmatrix}$$

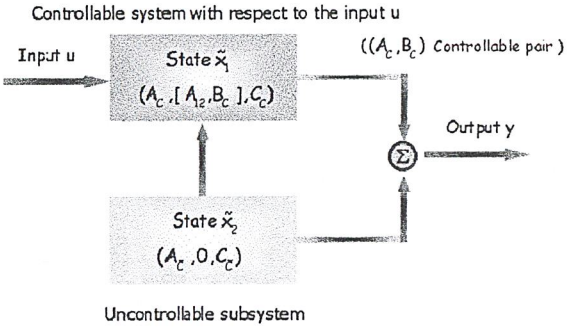
กำหนด การแปลง T = [v_1 , v_2 , ..., v_n] ; หากใช้ ตัวแปลง T แปลงระบบที่ไม่สามารถควบคุมได้อย่างสมบูรณ์ จะได้ระบบที่สมมูลกันอยู่ในรูปแบบดังนี้

$$\begin{bmatrix} \dot{\tilde{x}}_1 \\ \dot{\tilde{x}}_1 \end{bmatrix} = \left[\begin{array}{c|c} A_c & A_{12} \\ \hline O & A_{\bar{c}} \end{array}\right] \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_1 \end{bmatrix} + \left[\begin{array}{c} B_c \\ \hline O \end{array}\right] u$$

$$y = \begin{bmatrix} C_c & C_{\bar{c}} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_1 \end{bmatrix} + Du$$

เมื่อ (A_c, B_c) สามารถควบคุมได้

จากการแปลงดังกล่าวแสดงให้เห็นว่าระบบที่ไม่สามารถควบคุมได้อย่างสมบูรณ์จะสามารถแยกได้เป็น 2 ระบบ ย่อย ๆคือ ระบบย่อยที่สามารถควบคุมได้ และ ระบบย่อยที่ไม่สามารถควบคุมได้ ดังรูปที่ 18



รูปที่ 18 แสดง การแยกระบบที่ไม่ Completely Controllable แยกออกเป็น Controllable subsystem Uncontrollable Subsystem

จากรูปที่ 18 มีระบบย่อยอิสระ $\dot{\tilde{x}}_2 = A_{\bar{c}}\tilde{x}_2$ ที่ไม่ขึ้นต่ออินพุตระบบย่อยนี้เอง คือ ระบบย่อยที่ไม่สามารถควบคุมได้

การเขียนโปรแกรมเพื่อทำการแยกระบบที่ไม่สามารถควบคุมได้อย่างสมบูรณ์

ขั้นที่ 1: หา Controllability Matrix $\mathcal{C}$

ขั้นที่ 2: Rank($\mathcal{C}$) = n หรือไม่ ถ้า Rank($\mathcal{C}$) = n แยกระบบออกไม่ได้แล้ว(จบ) แต่ถ้า Rank($\mathcal{C}$) = n_c < n ให้ทำต่อใน ขั้นที่ 3

ขั้นที่ 3: เลือก v_1 , v_2 , ..., v_{nc} จาก เวคเตอร์อิสระเชิงเส้น จาก $\mathcal{C}$ โดยเราจะเลือกเวคเตอร์ ที่อิสระเชิงเส้นต่อกันได้โดยใช้ความรู้เรื่องของ แร็งค์ ตัวอย่างเช่น ถ้า v_1= [1;2;3] , v_2= [2;4;6] , v_1= [5;8;0] เนื่องจาก ถ้ากำหนด G_1 = [v_1 , v_2] , G_2 = [v_1 , v_3] แล้ว Rank(G_1) = 1 แต่ Rank(G_2) = 2 ดังนั้น v_1 v_2 ไม่อิสระเชิงเส้นต่อกันแต่ v_1 , v_3 อิสระเชิงเส้นต่อกัน

ขั้นที่ 4: เลือก เวคเตอร์ v_{nc+1} , v_{nc+2} , ..., v_n อีกโดย v_{nc+1} , v_{nc+2} , ..., v_nที่เลือกให้แต่ละตัวเป็นอิสระเชิงเส้นกับ v_1 , v_2 , ..., v_{nc} และ เวคเตอร์ ที่เลือกไปแล้วใน ขั้นที่ 4

เช่น ถ้า Rank($\mathcal{C}$) = 2 < 4 ; และสมมุติว่า $\mathcal{C}$ = [1 2 3 1;2 4 6 1;3 6 9 1;4 8 12 1] ดังนั้นจาก

ขั้นที่ 3 อาจเลือก v_1 = [1 ;2 ;3 ;4] แต่เลือก v_2 = [2;4; 6; 8] ไม่ได้เนื่องจาก Rank[1 2;2 4;3 6;4 8] = 1 และเลือก v_2 = [3;6;9;12] ไม่ได้อีกเช่นกันเนื่องจากเหตุผลดังกล่าว ดังนั้นจึงเลือก v_1 = [1;1;1;1] จากนั้นต้องเลือก เวคเตอร์ อีก 2 ตัว (เนื่องจาก n = 4) ถ้าเราจะเลือก เวคเตอร์ จากรูปแบบ [1;1;1;...;1] , [1;0;1;....;1] , [0;1;...;1] ซึ่งทุกตำแหน่งสลับ1,0 ได้เราจะได้ทางเลือกสำหรับการเลือก เวคเตอร์ ที่เหลือถึง $2^n - 1$ (ไม่นับ[0;0;...;0]) จากนั้นนำ เวคเตอร์ ที่เลือกมาทำการตรวจสอบความเป็นอิสระเชิงเส้นเช่นเดียวกับ ขั้นที่ 3 ซึ่งจากตัวอย่างดังกล่าว เราพบว่าไม่สามารถเลือก v_3 = [1;1;1;1] เนื่องจากRank[v_2, v_3] = 1 แต่เราสามารถใช้ v_3 = [1;0;1;0] ได้เนื่องจาก Rank[v_1, v_3] = 2 และ Rank[v_2, v_3] = 2 จากเหตุผลเดียวกันเลือก v_4 = [1 ;1 ;0 ;1] จากนั้นไปต่อ ขั้นที่ 5

ขั้นที่ 5: $T = [v_1, v_2, \ldots, v_n]$ และ $\tilde{A} = T^{-1}AT$, $\tilde{B} = T^{-1}B$, $\tilde{C} = CT$

เช่น

$$A = \begin{bmatrix} 0 & 0 & 0 & -24 \\ 1 & 0 & 0 & -50 \\ 0 & 1 & 0 & -35 \\ 0 & 0 & 1 & -10 \end{bmatrix}, \quad B = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 0 \end{bmatrix}, \quad C = [0 \quad 0 \quad 0 \quad 1]$$

$$\mathcal{C} = \begin{bmatrix} 2 & 0 & -24 & 168 \\ 3 & 2 & -50 & 326 \\ 1 & 3 & -33 & 195 \\ 0 & 1 & -7 & 37 \end{bmatrix}, \quad T = \begin{bmatrix} 2 & 0 & 1 & 1 \\ 3 & 2 & 1 & 0 \\ 1 & 3 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$\tilde{A} = \left[\begin{array}{cc:cc} 0 & -12 & -11 & 0 \\ 1 & -7 & -7 & 0 \\ \hdashline 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & 1 \end{array}\right], \quad \tilde{B} = \left[\begin{array}{c} 1 \\ 0 \\ \hdashline 0 \\ 0 \end{array}\right], \quad \tilde{C} = [0 \quad 1 \,\vdots\, 1 \quad 0]$$

5.3 Unobservable Subspace

ถ้าสมมุติว่า LTI system (11) ไม่สามารถสังเกตได้อย่างสมบูรณ์ จะได้ว่า Rank($\mathcal{O}$) = n_o < n เมื่อ $\mathcal{O}$ คือ Observability Matrix

ดังนั้น LTI system (11) สามารถแยกออกได้เป็นระบบย่อยที่สามารถสังเกตได้ และระบบย่อยที่ไม่สามารถสังเกตได้ ซึ่งในการหาการแปลงระบบมีขั้นตอนดังต่อไปนี้

① หาแถวที่เป็นอิสระเชิงเส้น จำนวน n_o ตัวจาก $\mathcal{O}$

② หาแถวที่เป็นอิสระเชิงเส้น เพิ่มเติมอีก n - n_o ตัวเพื่อที่จะสร้างตัวดำเนินการ เมตริกซ์ที่ไม่เอกฐาน

$$\tilde{T} = \begin{bmatrix} u_1 \\ \vdots \\ u_{n_o} \\ \hline u_{n_o+1} \\ \vdots \\ u_n \end{bmatrix}^{-1}$$

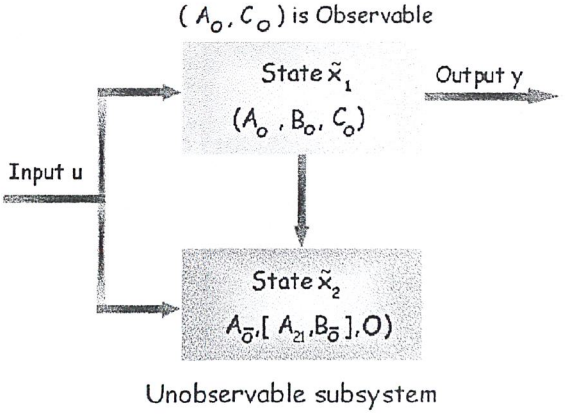
ถ้ากำหนดสภาวะ ใหม่ $x = \tilde{T}^{-1}\tilde{x}$ จะได้ระบบใหม่ที่อยู่ในรูปแบบของ

$$\begin{bmatrix} \dot{\tilde{x}}_1 \\ \dot{\tilde{x}}_1 \end{bmatrix} = \left[\begin{array}{c|c} A_o & O \\ \hline A_{21} & A_{\bar{o}} \end{array}\right] \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_1 \end{bmatrix} + \begin{bmatrix} B_o \\ B_{\bar{o}} \end{bmatrix} u$$

$$y = \left[\begin{array}{c|c} C_o & O \end{array}\right] \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_1 \end{bmatrix} + Du$$

เมื่อ (A_o, C_o) สามารถสังเกตได้

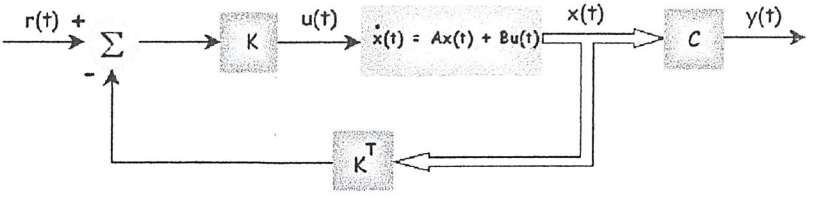
จากการแปลง ดังกล่าวแสดงให้เห็นว่าไม่สามารถสังเกตได้อย่างสมบูรณ์จะสามารถแยกได้เป็น 2 ระบบ ย่อย ๆคือ ระบบย่อยที่ สามารถสังเกตได้ และ ระบบย่อยที่ ไม่สามารถสังเกตได้ ดังรูปที่ 19



รูปที่ 19 แสดง การแยกระบบที่ไม่ Completely Observable แยกออกเป็น Observable subsystem Unobservable Subsystem

5.4 State Variable Feedback

พิจารณา Block Diagram ดังรูปที่ 20



รูปที่ 20 แสดง System กับ State-variable Feedback

ถ้าเรามีระบบเชิงเส้นที่แสดงด้วยสมการ

$$\dot{x}(t) = A\,x(t) + Bu(t) \qquad (13)$$
$$y(t) = Cx(t)$$

ถ้าระบบนี้เราสมมุติว่าเป็น อินพุตเดียวเอาท์พุตเดียว (SISO) และถ้าตัวแปรสภาวะถูกป้อนกลับด้วย

$$u(t) = K\,(\,r - k^T\,) \qquad (14)$$

เมื่อ $k^T \triangleq [\,k_1\, k_2 \ldots k_n\,]$

แทน (14) ลง ใน (13) จะได้ว่า

$$\dot{x}(t) = A_f\,x(t) + KBr(t) \qquad (15)$$
$$y(t) = Cx(t)$$

เมื่อ $A_f = A - BKk^T$

สมการที่ (15) คือสมการสภาวะสำหรับระบบวงปิดซึ่งสังเกตได้ว่า C ยังคงไม่เปลี่ยนแปลง

จากทฤษฎีบทกล่าวว่า สำหรับระบบที่สามารถควบคุมได้อย่างสมบูรณ์เราสามารถหา Kk^T ได้เสมอ

เพื่อที่จะวางโพลของระบบวงปิด (หรือ ค่าไอเก็น ของ A_f) ในตำแหน่งใด ๆ ที่ต้องการได้

แนวคิด

ถ้าสมมุติว่าระบบเป็นระบบที่สามารถควบคุมได้อย่างสมบูรณ์ เราจะสามารถหาตัวแปลงที่เป็นเมตริกซ์ไม่เอกฐาน P โดยที่ $\det(sI - A) = s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0$

แล้ว $P \triangleq \mathcal{C}W$ ซึ่ง P คือ เมตริกซ์การแปลง ที่แปลงระบบใด ๆ ที่ สามารถควบคุมได้อย่างสมบูรณ์ไปสู่รูปแบบ Controllable Canonical

เมื่อ $\mathcal{C}$ คือ Controllability Matrix

$$W \triangleq \begin{bmatrix} a_1 & a_2 & \cdots & a_{n-1} & 1 \\ a_2 & a_3 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1} & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

นั่นคือ

$$\tilde{A} = P^{-1}AP = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix}, \quad \tilde{B} = P^{-1}B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

เนื่องจากว่า $\tilde{A} = P^{-1}AP$ ค่าไอเก็น ของ $\tilde{A}$ ยังคงเท่ากับ ค่าไอเก็น ของ A ถ้า P เป็น ตัวแปลงที่ไม่เอกฐาน ดังนั้นเราจะพิจารณาที่รูปแบบ Controllable Canonical ก่อน

เนื่องจาก $\tilde{B}K = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ K \end{bmatrix}$ และ $\tilde{B}K\tilde{k}^T = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \\ K\tilde{k}_1 & K\tilde{k}_2 & K\tilde{k}_3 & \cdots & K\tilde{k}_n \end{bmatrix}$

$$\tilde{A}_f = \tilde{A} - \tilde{B}K\tilde{k}^T = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ -a_0 - K\tilde{k}_1 & -a_1 - K\tilde{k}_1 & \cdots & -a_{n-1} - K\tilde{k}_{n-1} \end{bmatrix}$$

ถ้าต้องการตำแหน่งโพล $\mu_1, \mu_2, \ldots, \mu_n$ ซึ่งทำให้ได้ พหุนามคุณลักษณะ ของ $\tilde{A}_f$ เท่ากับ $(s-\mu_1)(s-\mu_2)\ldots(s-\mu_n) = s^n + \alpha_{n-1}s^{n-1} + \ldots + \alpha_1 s + \alpha_0 = \det(sI - \tilde{A}_f)$ จากนั้นเปรียบเทียบแถวสุดท้ายของ $\tilde{A}_f$ จะได้

$$\begin{aligned} a_0 + K\tilde{k}_1 &= \alpha_0 & K\tilde{k}_1 &= \alpha_0 - a_0 \\ a_1 + K\tilde{k}_2 &= \alpha_1 & K\tilde{k}_2 &= \alpha_1 - a_1 \\ &\vdots & &\vdots \\ a_{n-1} + K\tilde{k}_n &= \alpha_{n-1} & K\tilde{k}_n &= \alpha_{n-1} - a_{n-1} \end{aligned}$$

ถ้าเลือก ตัวขยายคงที่ K เพื่อทำให้ ตัวขยาย dc วงปิด ฟังก์ชันถ่ายโอน = 1 จะทำให้เหลือเฉพาะแถว $\tilde{k}^T$ เนื่องจาก $\tilde{k}^T\tilde{x} = \tilde{k}^T P^{-1}x = k^T x$ ดังนั้น $k^T = \tilde{k}^T P^{-1}$ ซึ่งก็คือ เมตริกซ์ป้อนกลับก่อนการ แปลง

ตัวอย่างการเขียนโปรแกรมหา เวคเตอร์ State Feedback k^T

```
function KT = statefeedback(A,B,Pole,Gain);
if rank(ctrb(A,B)) ~= length(A);
      disp('End');
else
      uhu = eig(A);
      delta1 = [1];
      delta2 = [1];
      for i =1:length(A);
            delta1 = conv(delta1,[1 -uhu(i,1)]);
            delta2 = conv(delta2,[1 -Pole(i)]);
      end;
      ohmygod = 1;
      for j = 1:length(A);
            ohmygod(j) = delta1(length(delta1)-j);
      end;
      W(1,:)= ohmygod;
      for i = 2:length(A);
            ohmygod = [ohmygod(1,2:length(A)) 0];
            W(i,:) = ohmygod;
      end;
      P = ctrb(A,B)*W;
      for i = 1:length(A);
            KT(i) = delta2(length(A)+2-i)- delta1(length(A)+2-i);
      end;
      KT = KT*inv(P)*1/Gain;
end;
```

5.5 Asymptotic State Observer

ในการทำ State feedback ในหัวข้อที่แล้วทำให้เราสามารถวางตำแหน่งของ โพล ของระบบวงปิดในตำแหน่งใด ๆ ที่ต้องการได้ แต่ต้องอยู่ภายใต้เงื่อนไขที่ว่าระบบนั้นต้อง ควบคุมได้อย่างสมบูรณ์ แต่ในทางปฏิบัติ สภาวะ x(t) ของระบบไม่สามารถตรวจจับค่ามา ป้อนกลับ ได้ (x(t) undetectable) , ดังนั้นเราจึงต้องสร้างสภาวะตัวใหม่จากอินพุต , เอาท์พุต ของระบบมาทำการ ป้อนกลับแต่สำหรับการสร้างสภาวะขึ้นมาได้นั้นต้องอยู่ภายใต้เงื่อนไขสามารถสังเกตได้อย่างสมบูรณ์

ในการแก้ปัญหาการสร้างสภาวะใหม่จะใช้วิธีการป้อนอินพุตตัวเดียวกับระบบเข้าไปในแบบจำลองที่กำหนดขึ้น

ถ้ามีระบบเชิงเส้น

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathrm{A}\,\mathbf{x}(t) + \mathrm{B}u(t) \\ y(t) &= \mathrm{C}\mathbf{x}(t) \end{aligned} \tag{16}$$

ถ้าเราจะกำหนดแบบจำลองขึ้นมาใหม่โดย

$$\begin{aligned} \dot{\hat{\mathbf{x}}} &= \mathrm{A}\hat{\mathbf{x}} + \mathrm{Bu} + \mathrm{L}(y - \hat{y}) \\ \mathrm{y} &= \mathrm{C}\hat{\mathbf{x}} \end{aligned} \tag{17}$$

เมื่อ $\hat{\mathbf{x}}(t)$ คือ เวคเตอร์สภาวะสำหรับแบบจำลองนี้

L คือ เวคเตอร์ตัวขยายโดยมีมิติเท่ากับ x

ถ้านิยามค่าผิดพลาดในการสร้างสภาวะขึ้นมาใหม่

$$\tilde{x}(t) \triangleq x(t) - \hat{x}(t) \qquad (18)$$

หาอนุพันธ์ของสมการที่ (18) ทั้งสองข้างและแทน $x(t)$ จาก (16) และ $\hat{x}(t)$ จาก (17) จะได้

$$\dot{\tilde{x}}(t) = A(x-\hat{x}) - L(y-\hat{y}) = A(x-\hat{x}) - LC(x-\hat{x}) = (A-LC)(x-\hat{x}) = (A-LC)\tilde{x}(t)$$

หรือจะได้ว่า ผลคำตอบ $\tilde{x}(t) = e^{(A-LC)t}\tilde{x}(0)$ เมื่อ $\tilde{x}(0) = x(0) - \hat{x}(0)$

จากสมการจะพบว่าถึงแม้ว่าในตอนต้นค่าผิดพลาดในการสร้าง สภาวะขึ้นมาใหม่จะยังไม่เท่ากับศูนย์แต่เราก็สามารถลดมันลงได้โดยการเลือก L ที่เหมาะสมเพื่อที่จะทำให้ ค่าไอเก็น ของ A - LC มีเฉพาะส่วนจริงที่เป็นลบ

จากทฤษฎีบทกล่าวว่า สำหรับระบบที่ สามารถสังเกตได้อย่างสมบูรณ์เราสามารถหาเวคเตอร์ตัวขยาย L ได้เสมอเพื่อที่จะวาง ค่าไอเก็น ของ A - LC ในตำแหน่งใด ๆ ที่ต้องการได้

แนวคิด

ถ้าสมมุติว่าระบบที่สามารถสังเกตได้อย่างสมบูรณ์เราจะสามารถหาตัวแปลงที่ไม่เอกฐาน P โดยที่ $\det(sI - A) = s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0$

แล้ว $P \triangleq [W\mathcal{O}]^{-1}$ ซึ่ง P คือ เมตริกซ์ตัวแปลง ที่แปลงระบบใด ๆ ที่ สังเกตได้อย่างสมบูรณ์ ไปสู่รูปแบบ Observable Canonical

เมื่อ $\mathcal{O}$ คือ Observability Matrix

$$W \triangleq \begin{bmatrix} a_1 & a_2 & \cdots & a_{n-1} & 1 \\ a_2 & a_3 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1} & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\bar{A} = P^{-1}AP = \begin{bmatrix} 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & \cdots & 0 & -a_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & -a_{n-2} \\ 0 & 0 & \cdots & 0 & -a_{n-1} \end{bmatrix}, \quad \bar{C} = CP = [0 \;\; 0 \;\; \ldots \;\; 0 \;\; 1]$$

$$\bar{L} = \begin{bmatrix} l_0 \\ l_1 \\ \vdots \\ l_{n-1} \end{bmatrix} \quad \text{ถ้า} \quad L = P\bar{L}, \quad \bar{A} - \bar{L}\bar{C} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -(a_0 + l_0) \\ 1 & 0 & \cdots & 0 & -(a_1 + l_1) \\ 0 & 1 & \cdots & 0 & -(a_2 + l_2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -(a_{n-2} + l_{n-2}) \\ 0 & 0 & \cdots & 1 & -(a_{n-1} + l_{n-1}) \end{bmatrix}$$

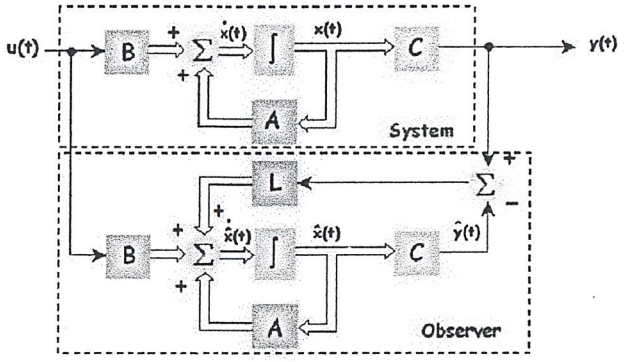
ถ้าต้องการตำแหน่ง Pole $\gamma_1, \gamma_2, \ldots, \gamma_n$ ซึ่งทำให้ได้ พหุนามคุณลักษณะ ของ $\bar{A} - \bar{L}\bar{C}$ เท่ากับ $(s-\gamma_1)(s-\gamma_2)\ldots(s-\gamma_n) = s^n + d_{n-1}s^{n-1} + \ldots + d_1 s + d_0 = \det(sI - (\bar{A} - \bar{L}\bar{C}))$

จากนั้นเปรียบเทียบสดมภ์สุดท้ายของ $\bar{A} - \bar{L}\bar{C}$ จะได้

$$-(a_0 + l_0) = -d_0 \qquad l_0 = d_0 - a_0$$
$$\vdots \qquad\qquad \vdots$$
$$-(a_{n-1} + l_{n-1}) = -d_{n-1} \qquad l_{n-1} = d_{n-1} - a_{n-1}$$

$$\bar{L} = \begin{bmatrix} d_0 - a_0 \\ \vdots \\ d_{n-1} - a_{n-1} \end{bmatrix} \quad \text{จะได้ } L = P\bar{L}$$

สำหรับบล็อคไดอะแกรม ของ Observer แสดงดังรูปที่ 21



รูปที่ 21 แสดง Asymptotic Observer

ตัวอย่างการเขียนโปรแกรม Design Asymptotic Observer

```
function L = asymobserver(A,C,Pole)
if rank(obsv(A,C)) ~= length(A);
    disp('End');
else
  uhu = eig(A);
    delta1 = [1];
    delta2 = [1];
    for i =1:length(A);
        delta1 = conv(delta1,[1 -uhu(i,1)]);
        delta2 = conv(delta2,[1 -Pole(i)]);
    end;
    ohmygod = 1;
    for j = 1:length(A);
        ohmygod(j) = delta1(length(delta1)-j);
    end;
```

```
W(1,:)= ohmygod;
    for i = 2:length(A);
        ohmygod = [ohmygod(1,2:length(A)) 0];
        W(i,:) = ohmygod;
    end;
    P = inv(W*obsv(A,C));
    for i = 1:length(A);
        L(i,1) = delta2(length(A)+2-i)- delta1(length(A)+2-i);
    end;
  L = P*L;
end;
```

ภาคผนวก (ก)

ตัวอย่างการใช้งาน SOFTWARE

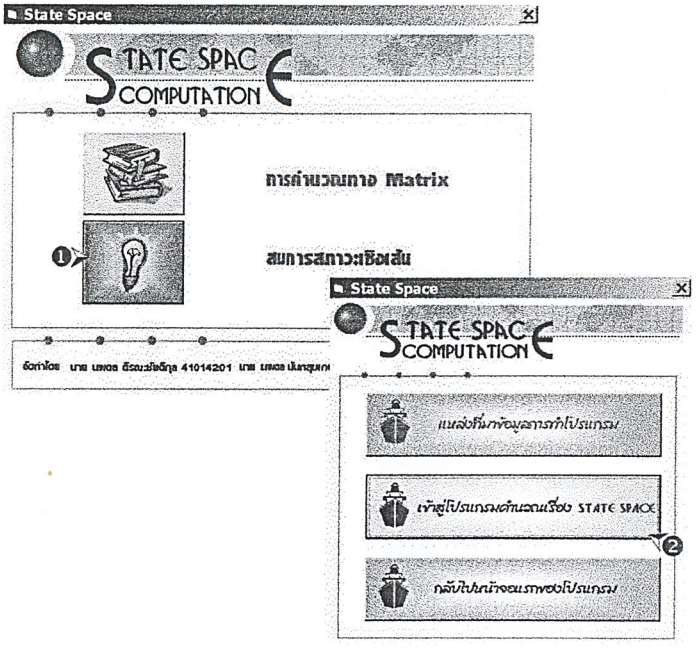
ตัวอย่างการใช้งาน Software ที่เขียนขึ้น

1. วิธีเข้าหน้าจอ การคำนวณเรื่อง สเตทสเปซ

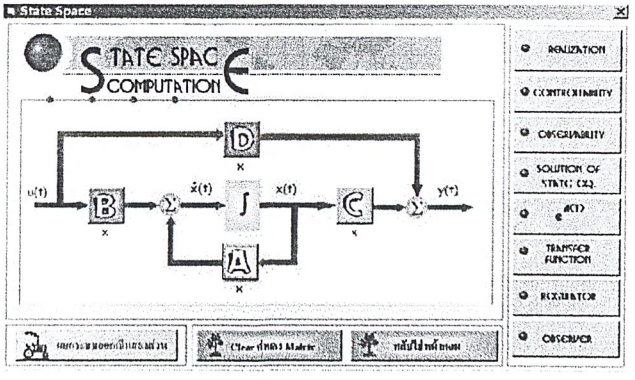
เนื่องจากโปรแกรมที่เขียนขึ้นส่วนใหญ่จะเกี่ยวข้องกับเมตริกซ์ ดังนั้นเราจึงมีส่วนของการคำนวณ เมตริกซ์ อยู่ด้วยแต่ในที่นี้จะกล่าวแต่เพียงการใช้งานเรื่องของ สเตทสเปซ เท่านั้น

วิธีการเข้าหน้าจอ การคำนวณเรื่อง สเตทสเปซ ทำได้ดังนี้

1. กดปุ่มสมการสภาวะเชิงเส้น
2. หน้าจอถัดมาจะพบว่ามี 3 ปุ่ม ให้เลือก ปุ่ม *เข้าสู่โปรแกรมคำนวณเรื่อง สเตทสเปซ*



3. หน้าจอถัดมาจะพบหน้าจอการคำนวณเรื่อง สเตทสเปซ



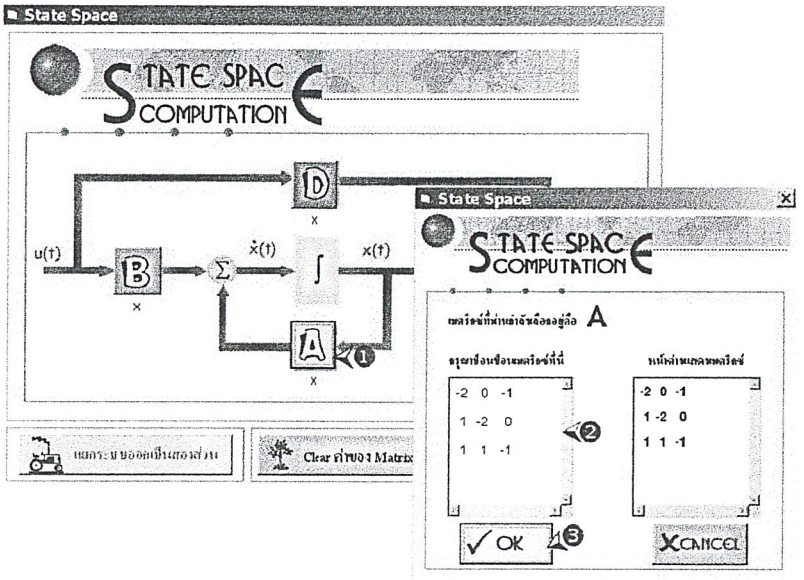
2. การหาฟังก์ชันถ่ายโอนจากสมการสภาวะ

ตัวอย่างที่ 2.1 หา ฟังก์ชันถ่ายโอน จากกรณีต่อไปนี้

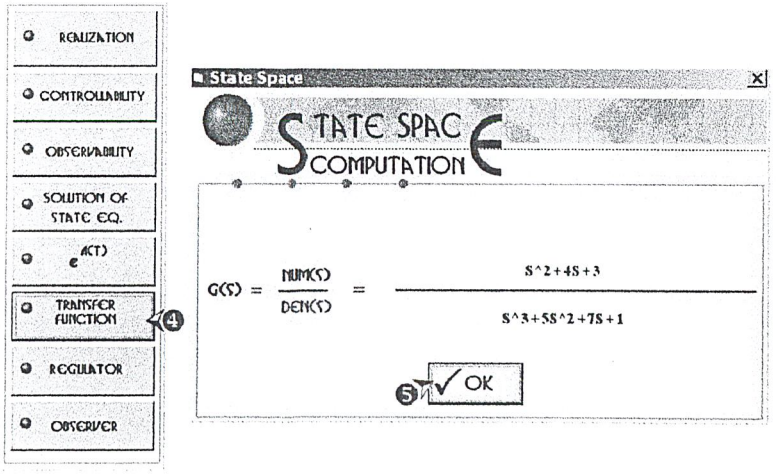
$$A = \begin{bmatrix} -2 & 0 & 1 \\ 1 & -2 & 0 \\ 1 & 1 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \qquad C = [\ 2 \quad 1 \quad -1\ ]$$

วิธีทำ

1. เลือก เมตริกซ์ ที่ต้องการป้อนค่า เช่นถ้าต้องการป้อน เมตริกซ์ A
2. เข้าหน้าจอการป้อนค่า ให้ป้อนค่า เมตริกซ์ ให้ถูกต้อง
3. จากนั้นกดปุ่ม OKเพื่อกลับมาที่หน้าจอ สเตทสเปซ



4. เมื่อกลับมาที่หน้าจอ สเตทสเปซ แล้ว ให้กดปุ่ม *Transfer Function* เพื่อหาฟังก์ชันถ่ายโอน ของระบบ
5. จะพบหน้าจอแสดงผล ฟังก์ชันถ่ายโอน จากนั้นกดปุ่ม OK เพื่อกลับหน้าจอสเตทสเปซ



ลองตรวจสอบผลที่ได้กับคำสั่ง [NUM,DEN] = SS2TF(A,B,C,D,iu) ของ MATLAB®

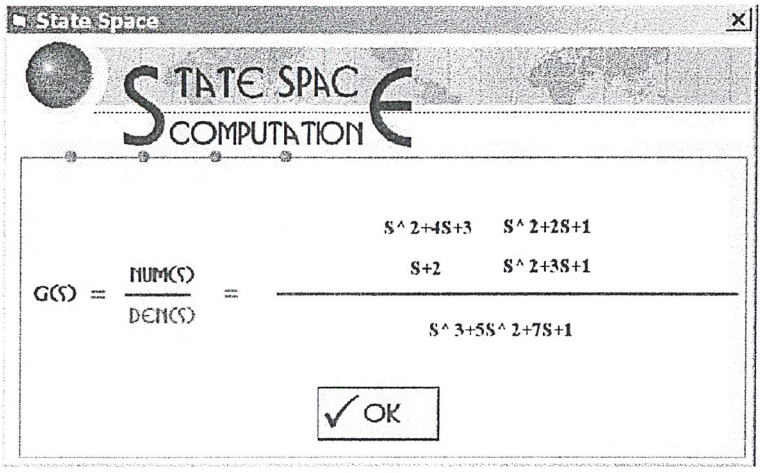
```
A =[-2 0 1;1 -2 0;1 1 -1]; B = [1;0;1] ; C =[2 1 -1];D =0;
[Num,Den] = ss2tf(A,B,C,D)
Num =
         0    1.0000    4.0000    3.0000
Den =
    1.0000    5.0000    7.0000    1.0000
```

ตัวอย่างที่ 2.2 หา ฟังก์ชันถ่ายโอน จากกรณีต่อไปนี้

$$A = \begin{bmatrix} -2 & 0 & 1 \\ 1 & -2 & 0 \\ 1 & 1 & -1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 2 & 1 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

วิธีทำ

ทำเหมือนตัวอย่างที่ 2.1 ตั้งแต่ขั้นตอนที่ 1 – 5 จะได้ผลดังนี้



ลองตรวจสอบผลที่ได้กับคำสั่ง [NUM,DEN] = SS2TF(A,B,C,D,iu) ของ MATLAB®

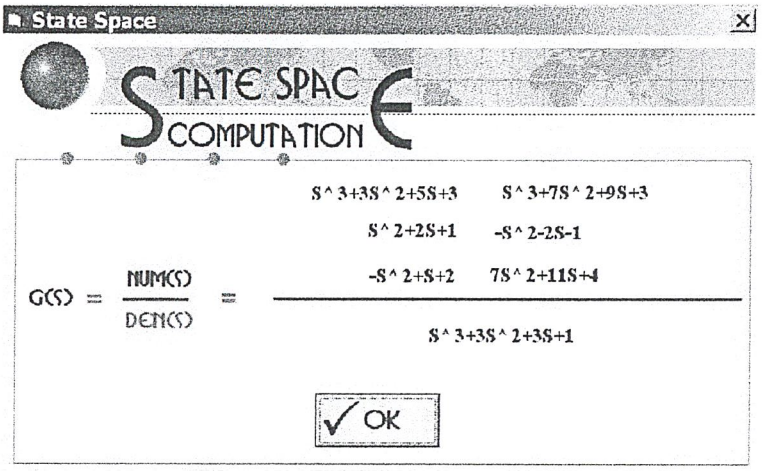
```
A =[-2 0 1;1 -2 0;1 1 -1];B = [1 0;0 1;1 0];C = [2 1 -1;0 1 0]; D =[0 0;0 0];
[Num1,Den1] = ss2tf(A,B,C,D,1)
Num1 =
         0    1.0000    4.0000    3.0000
         0    0.0000    1.0000    2.0000
Den1 =
    1.0000    5.0000    7.0000    1.0000

[Num2,Den2] = ss2tf(A,B,C,D,2)
Num2 =
         0    1.0000    2.0000    1.0000
         0    1.0000    3.0000    1.0000
Den2 =
    1.0000    5.0000    7.0000    1.0000
```

ตัวอย่างที่ 2.3 หา ฟังก์ชันถ่ายโอน จากกรณีต่อไปนี้

$$A = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad B = \begin{bmatrix} 2 & 0 \\ 1 & -1 \\ -1 & 1 \end{bmatrix} \quad C = \begin{bmatrix} 2 & -1 & 3 \\ 0 & 1 & 0 \\ 3 & -1 & 6 \end{bmatrix} \quad D = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

วิธีทำ

ทำเหมือนตัวอย่างที่ 2.1 ตั้งแต่ขั้นตอนที่ 1 – 5 จะได้ผลดังนี้



ลองตรวจสอบผลที่ได้กับคำสั่ง [NUM,DEN] = SS2TF(A,B,C,D,iu) ของ MATLAB®

```
A =[-1 1 0;0 -1 0;0 0 -1];B =[ 2 0;1 -1;-1 1];
C =[2 -1 3;0 1 0;3 -1 6];D = [1 1;0 0;0 0];
[Num1,Den1] = ss2tf(A,B,C,D,1)
Num1 =
    1.0000    3.0000    5.0000    3.0000
         0    1.0000    2.0000    1.0000
         0   -1.0000    1.0000    2.0000
Den1 =
     1     3     3     1
[Num2,Den2] = ss2tf(A,B,C,D,2)
Num2 =
    1.0000    7.0000    9.0000    3.0000
         0   -1.0000   -2.0000   -1.0000
         0    7.0000   11.0000    4.0000
Den2 =
     1     3     3     1
```

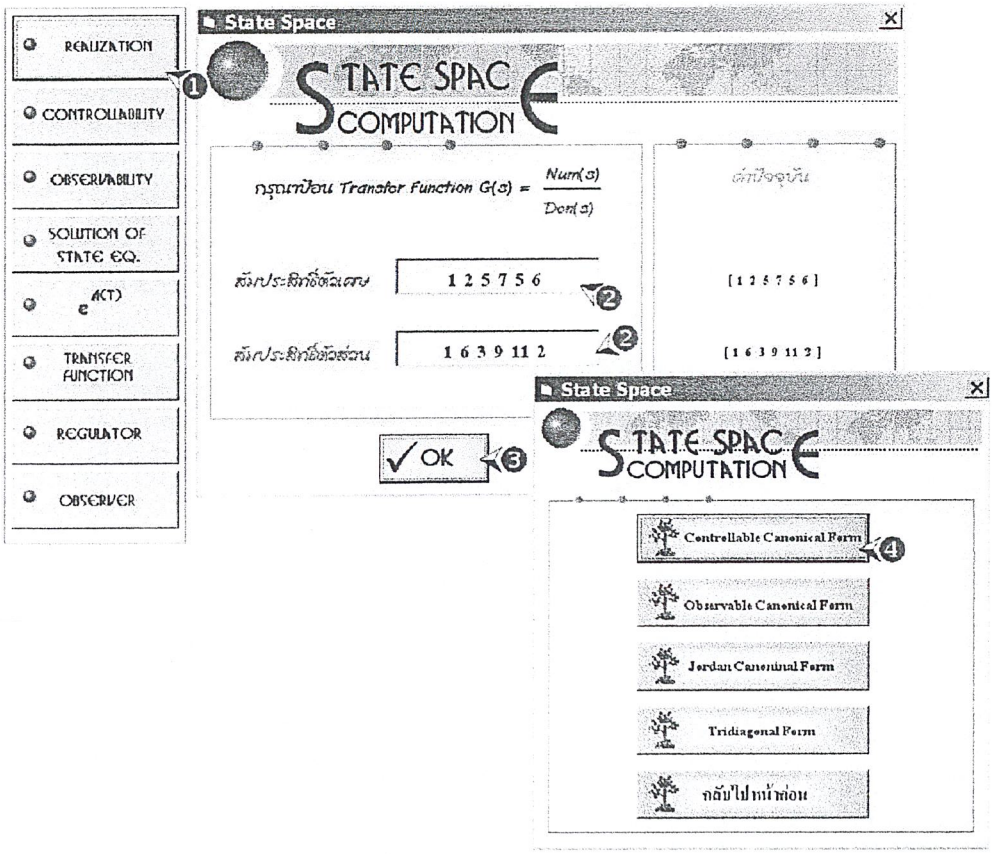
3. การ Realization ของระบบเชิงเส้นอินพุตเดียวเอาท์พุตเดียว

ตัวอย่างที่ 3.1 หา Realization ของ G(s) ในรูปแบบ Controllable Canonical

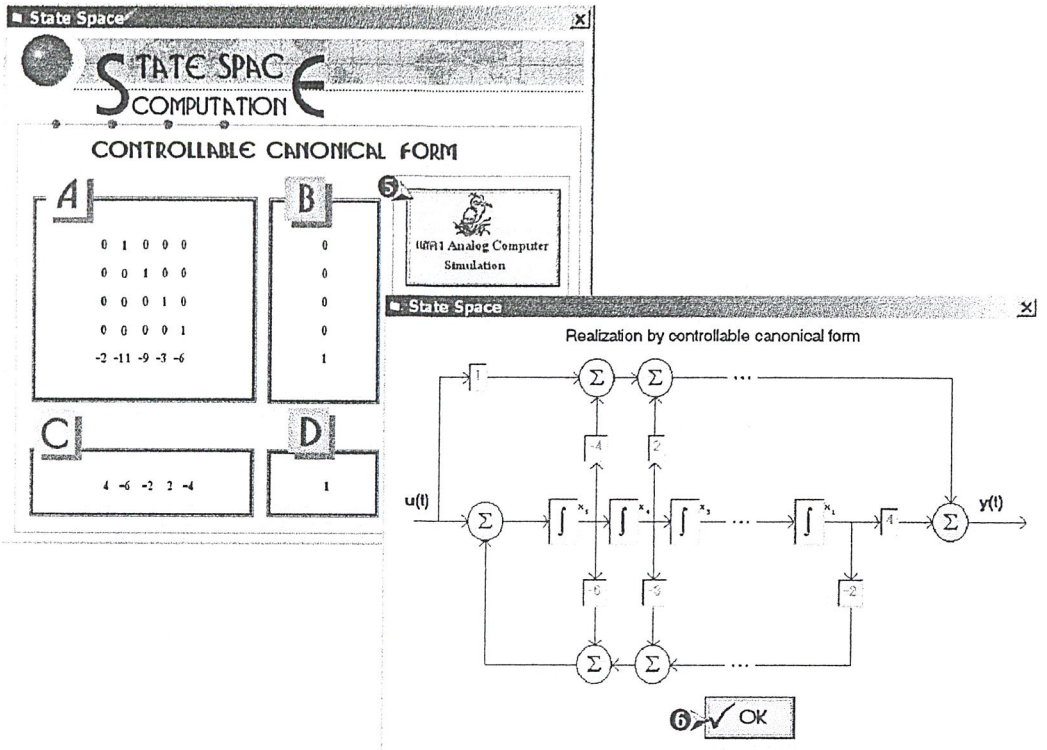
$$\text{เมื่อ } G(s) = \frac{s^5+2s^4+5s^3+7s^2+5s+6}{s^5+6s^4+3s^3+9s^2+11s+2}$$

วิธีทำ

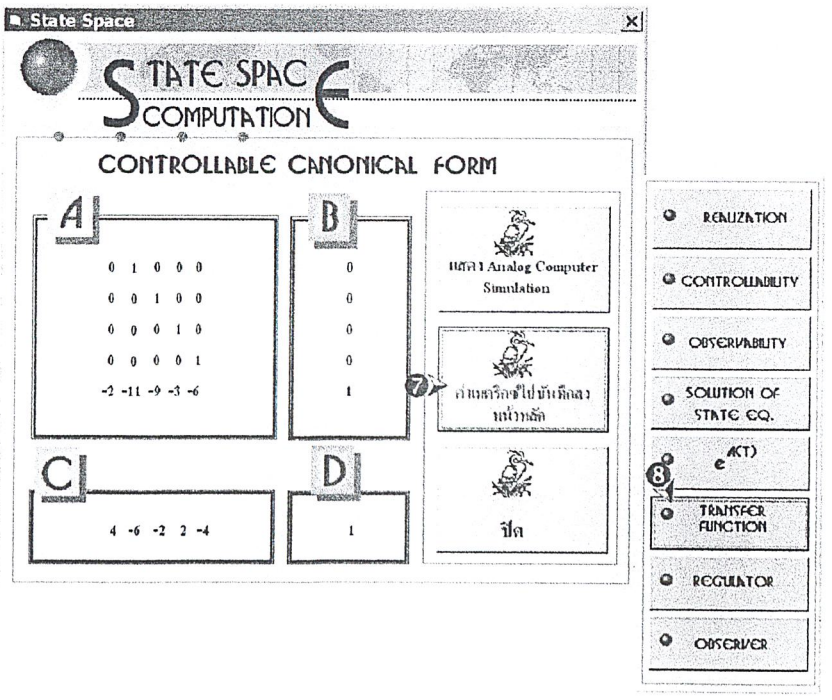
1. กดปุ่ม *Realization* (ปุ่มขวาบนสุดของหน้าจอ สเตทสเปซ) เพื่อเข้าสู่หน้าจอ การป้อน ฟังก์ชันถ่ายโอน
2. ป้อนสัมประสิทธิ์ของตัวเศษ และ ตัวส่วนของ ฟังก์ชันถ่ายโอน ที่โจทย์กำหนด
3. กดปุ่ม OK เพื่อนำค่าดังกล่าวไปประมวลผล
4. เลือกการ Realization ไปสู่ *Controllable Canonical Form*



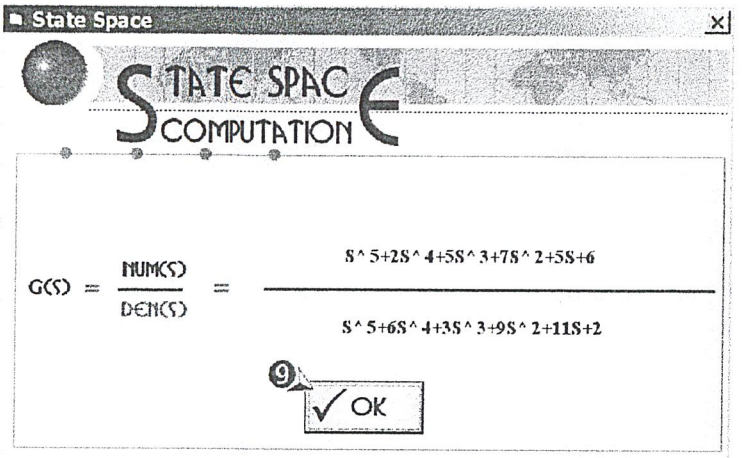
5. จะแสดงผลการ Realization ซึ่งหากต้องการดู *Analog Computer Simulation* ให้กดปุ่มทางด้านขวาปุ่มบนสุด
6. จะเปิดหน้าจอแสดง *Analog Computer Simulation* จากนั้นกดปุ่ม OK เพื่อกลับสู่หน้าจอแสดงผลการ Realization



7. ถ้าต้องการตรวจสอบผลการคำนวณให้กดปุ่ม *ค่า เมตริกซ์ไปบันทึกลงหน้าหลัก* ซึ่งเมื่อกดปุ่มนี้แล้วค่า เมตริกซ์ A B C และ D ในหน้าจอนี้จะถูกนำไปเก็บในหน้าจอสเตทสเปซ
8. กดปุ่ม *Transfer Function* เพื่อหา ฟังก์ชันถ่ายโอน ของระบบ



9. แสดงผลการหา ฟังก์ชันถ่ายโอน ซึ่งพบว่า ฟังก์ชันถ่ายโอน ที่ได้มีค่าตรงกับที่ป้อนเข้าในขั้นตอนที่ 2 จากนั้นกดปุ่ม OK เพื่อเข้าหน้าจอ สเตทสเปซ



ตัวอย่างที่ 3.2 หา Realization ของ G(s) ใน

(a) รูปแบบ Observable Canonical

(b) รูปแบบ Jordan Canonical

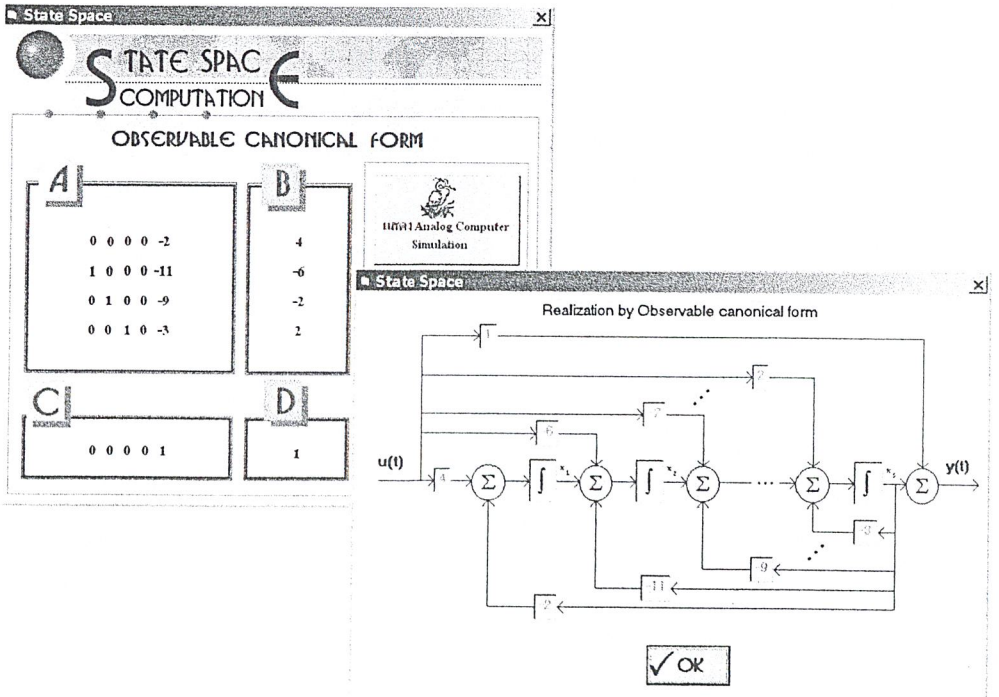
(c) รูปแบบ Tridiagonal

โดยใช้ G(s) ของตัวอย่างที่ 3.1

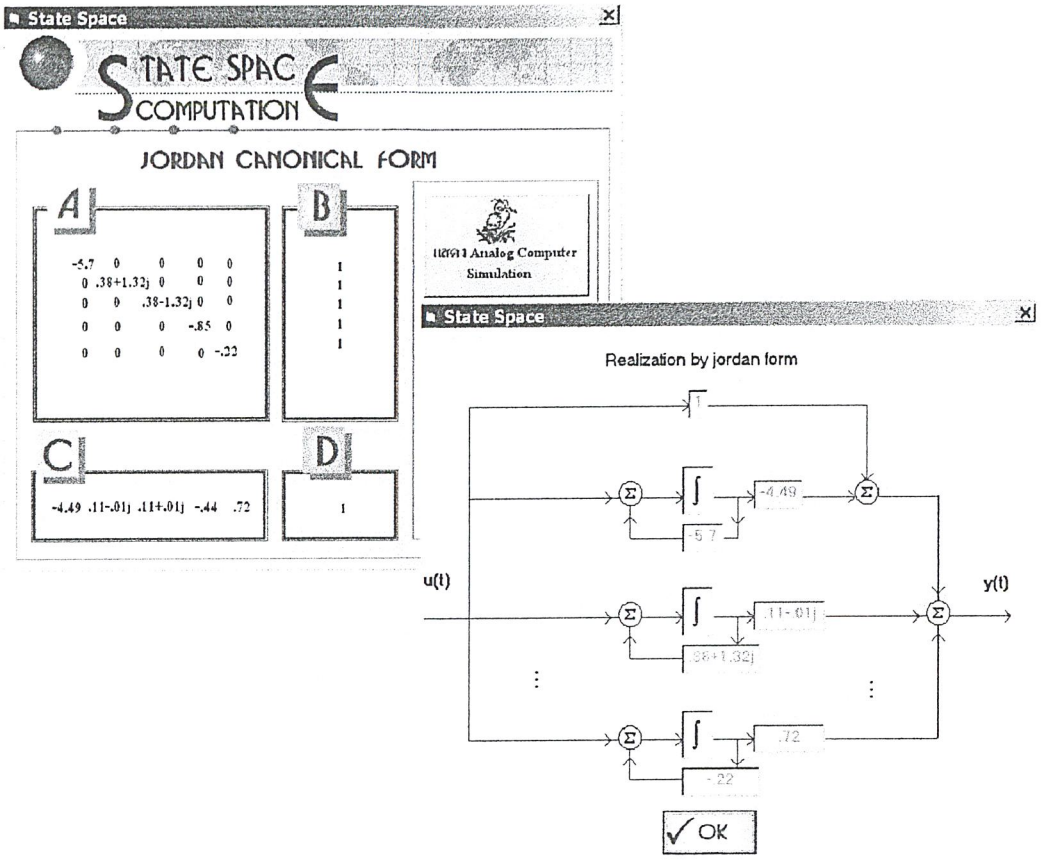
วิธีทำ

ทำเหมือนตัวอย่างที่ 3.1 ตั้งแต่ขั้นตอนที่ 1 – 6 จะได้ผลดังนี้

(a)



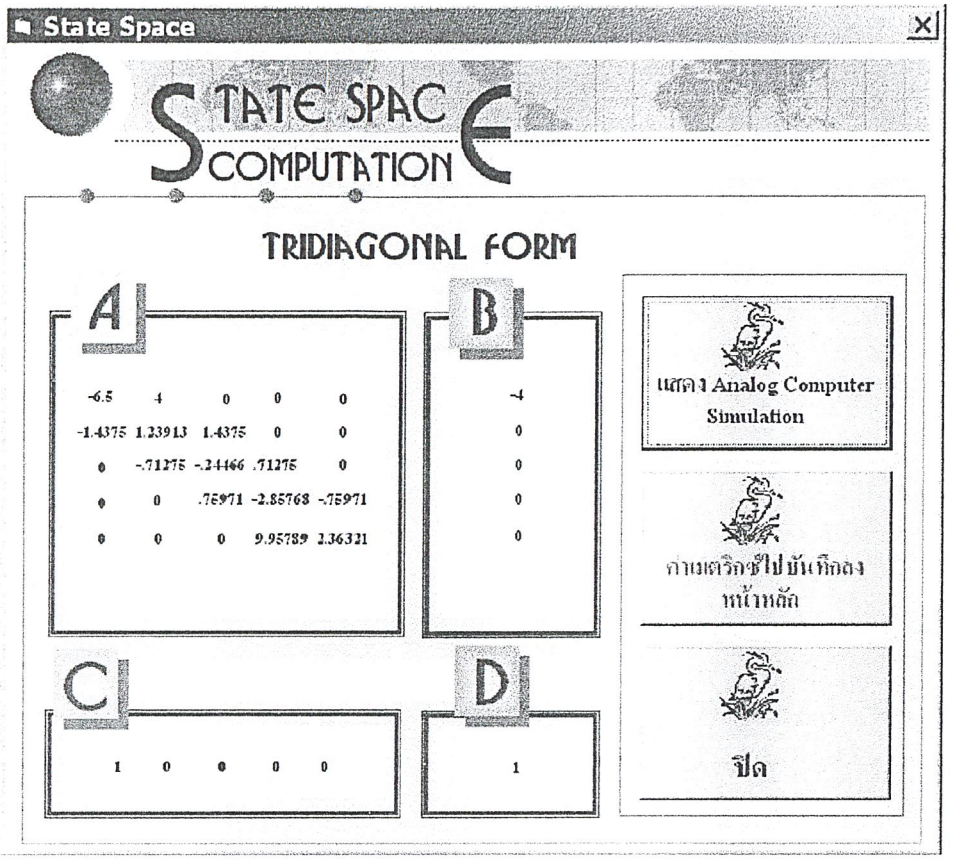
(b)

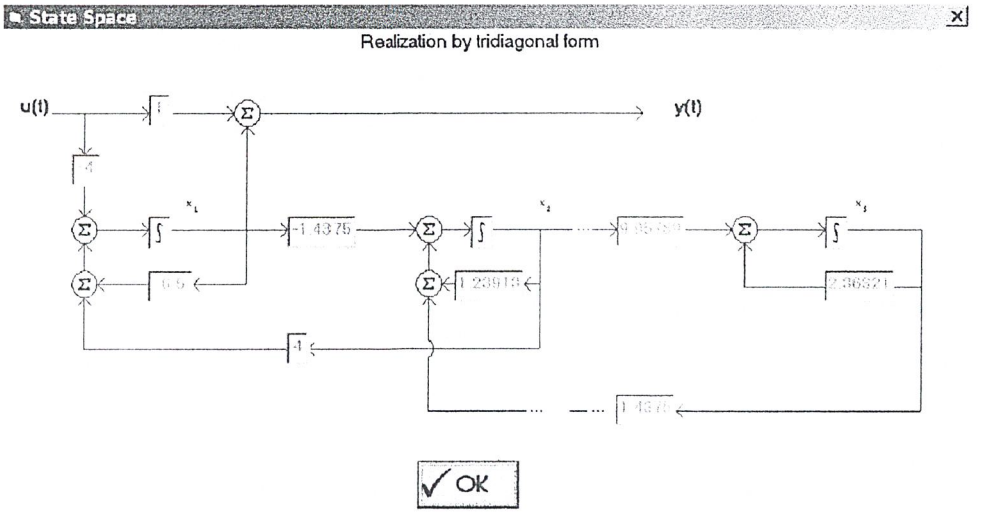


ซึ่งในการ Realization สู่ รูปแบบ Jordan Canonical สามารถตรวจสอบได้ด้วยคำสั่ง [R,P,K] = RESIDUE(Num,Den) ของ MATLAB®

```
Num = [1 2 5 7 5 6]; Den = [1 6 3 9 11 2];
[R,P,K] = residue(Num,Den)
R =
      -4.4893
       0.1044 - 0.0565i
      0.1044 + 0.0565i
      -0.4366
      0.7171
P =
      -5.6930
      0.3814 + 1.3176i
      0.3814 - 1.3176i
      -0.8502
      -0.2196
K =
       1
```

(c)


State Space
STATE SPACE COMPUTATION
TRIDIAGONAL FORM
A
-6.5 4 0 0 0
-1.4375 1.23913 1.4375 0 0
0 -.71275 -.24466 .71275 0
0 0 .75971 -2.85768 -.75971
0 0 0 9.95789 2.36321
B
-4
0
0
0
0
แสดง Analog Computer Simulation
คำนวณข้อมูลบันทึกลงหน้าหลัก
C
1 0 0 0 0
D
1
ปิด


State Space
Realization by tridiagonal form
u(t)
y(t)
-1.4375
1.23913
OK

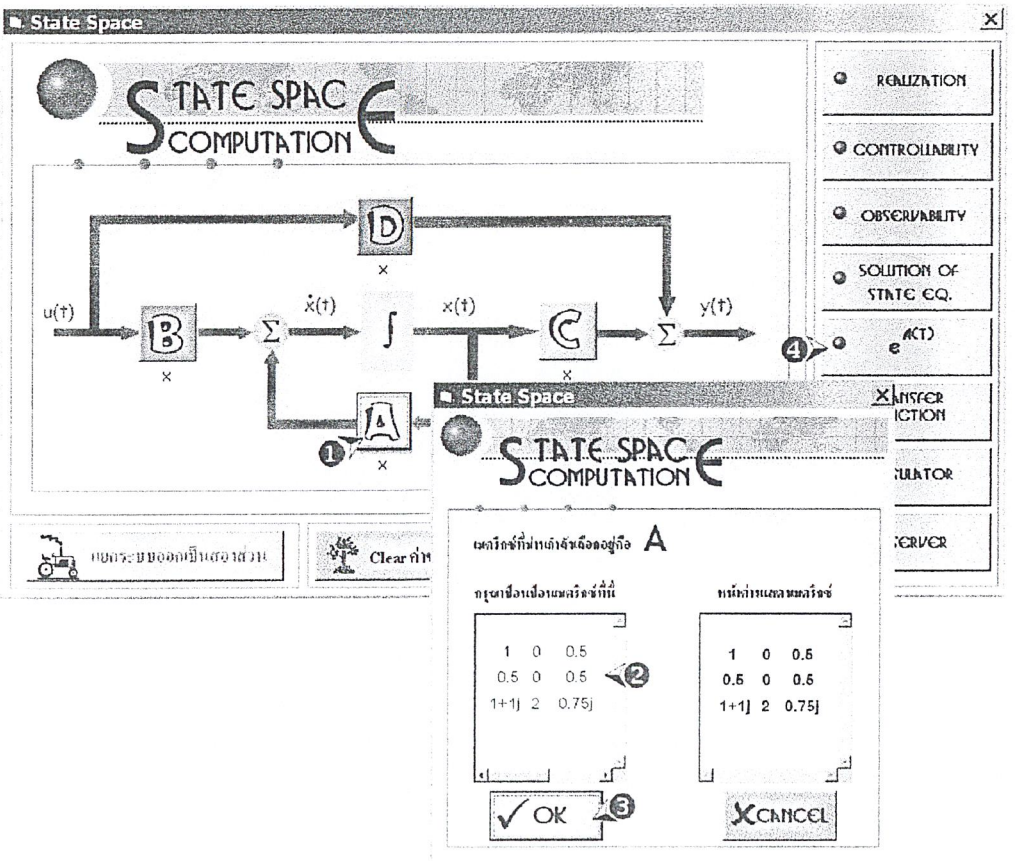
4. การคำนวณหาค่า เมตริกซ์ Transition

ตัวอย่างที่ 4 จงหาค่าของ e^{At} เมื่อ $t = 1$ และ

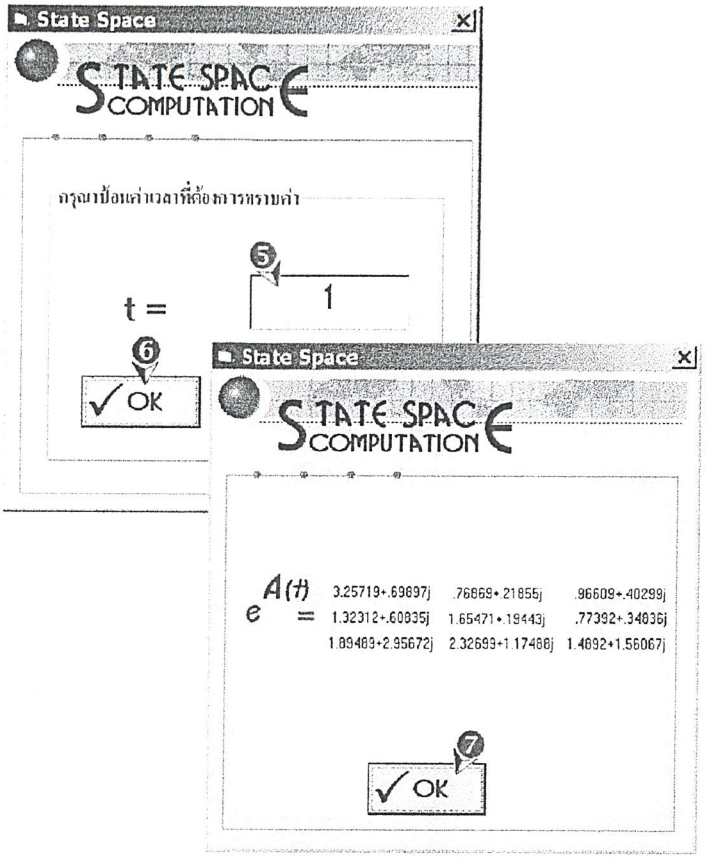
$$A = \begin{bmatrix} 1 & 0 & 0.5 \\ 0.5 & 0 & 0.5 \\ 1+j & 2 & 0.75j \end{bmatrix}$$

วิธีทำ

1. เลือกป้อน เมตริกซ์ A จากหน้าจอ สเตทสเปซ
2. เมื่อเข้าสู่หน้าจอป้อน เมตริกซ์ ให้ป้อน เมตริกซ์ A ตามโจทย์กำหนด
3. กดปุ่ม OK เพื่อให้โปรแกรมเก็บค่าไว้และกลับมายังหน้าจอ สเตทสเปซ
4. เลือกปุ่ม e^{At} ที่หน้าจอ สเตทสเปซ



5. จากนั้นให้ป้อนค่าตำแหน่งเวลาที่ต้องการทราบค่า
6. กดปุ่ม OK เพื่อให้โปรแกรมคำนวณค่า
7. แสดงค่าผลลัพธ์ที่ต้องการจากนั้น กดปุ่ม OK เพื่อกลับสู่หน้าจอ สเตทสเปซ



ตรวจสอบผลที่ได้ด้วยคำสั่ง expm(A) ของ MATLAB®

```
>> A = [1 0 0.5 ; 0.5 0 0.5 ; 1+j 2 0.75j]
A =
      1.0000                  0                  0.5000
      0.5000                  0                  0.5000
      1.0000 + 1.0000i   2.0000                  0 + 0.7500i
>> expm(A)
ans =
   3.2572 + 0.6990i   0.7687 + 0.2185i   0.9661 + 0.4030i
   1.3231 + 0.6084i   1.6547 + 0.1944i   0.7739 + 0.3484i
   1.8949 + 2.9567i   2.3270 + 1.1749i   1.4892 + 1.5607i
```

5. การหา Solution of the Linear Time – invariant State Equations

ตัวอย่างที่ 5 จงหาผลคำตอบของสมการสภาวะ

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(t)$$

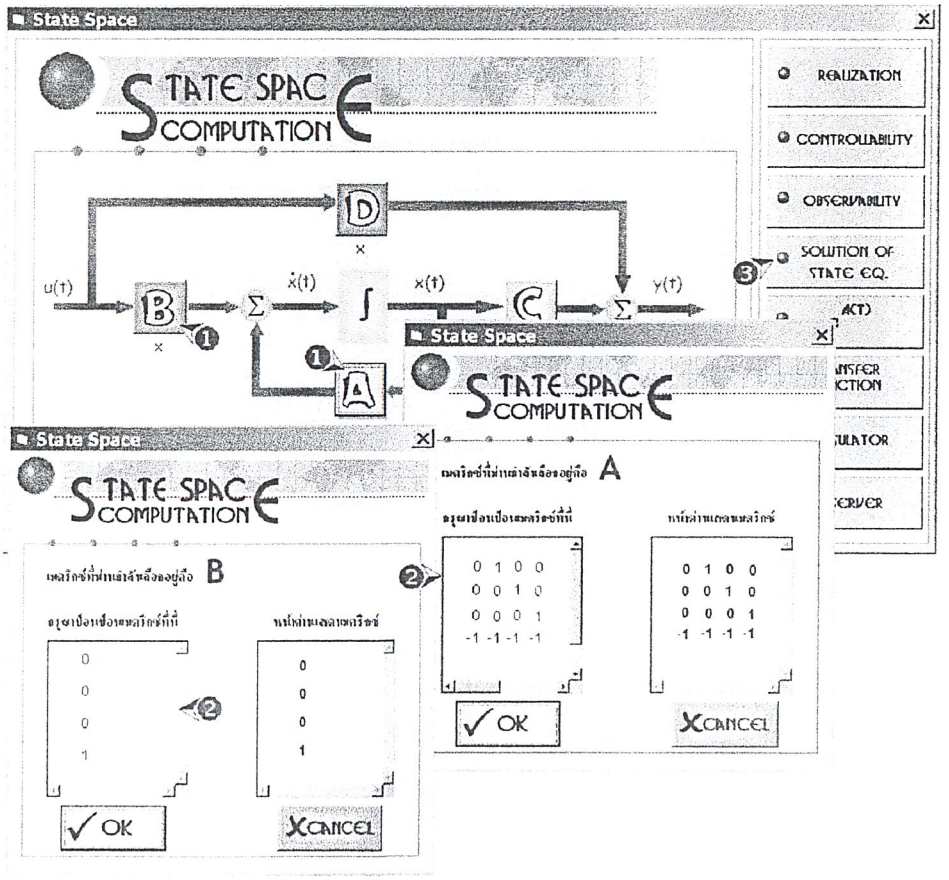
เมื่อ (a) $u(t) = 1$

(b) $u(t) = 5\,t$

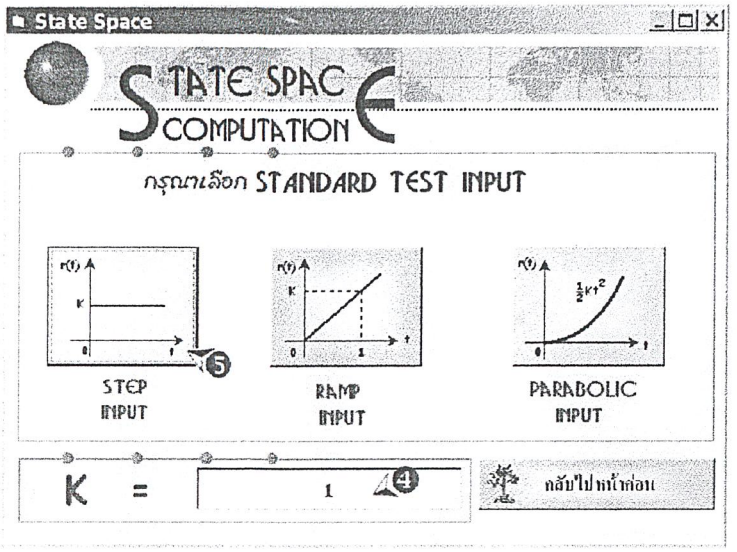
(c) $u(t) = t^2$

วิธีทำ

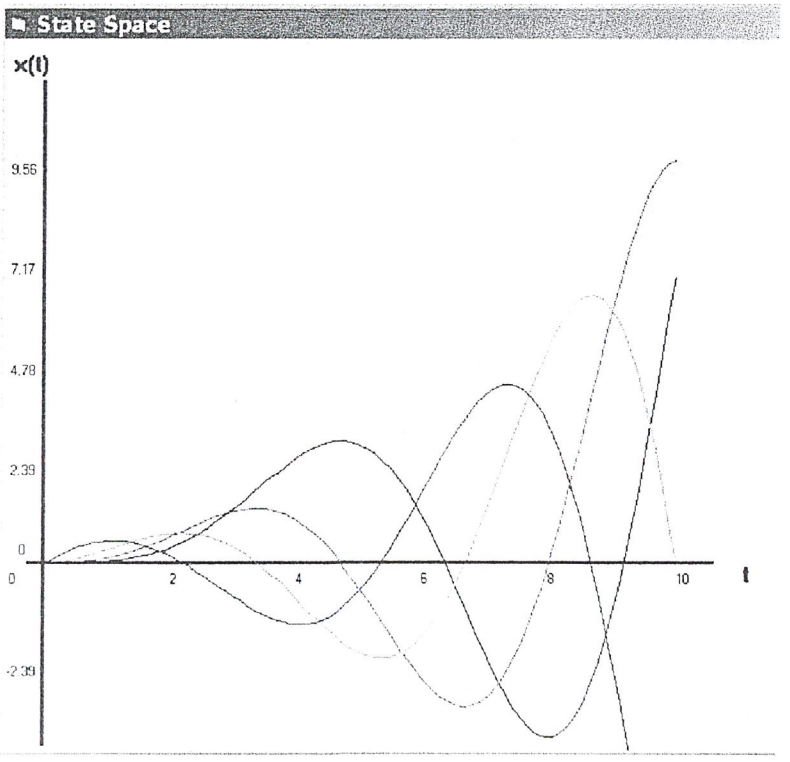
1. เลือกป้อน เมตริกซ์ A และ B ที่หน้าจอ สเตทสเปซ
2. ป้อนค่า เมตริกซ์ A และ B ตามที่โจทย์กำหนด
3. กดปุ่ม *Solution of State Eq.*



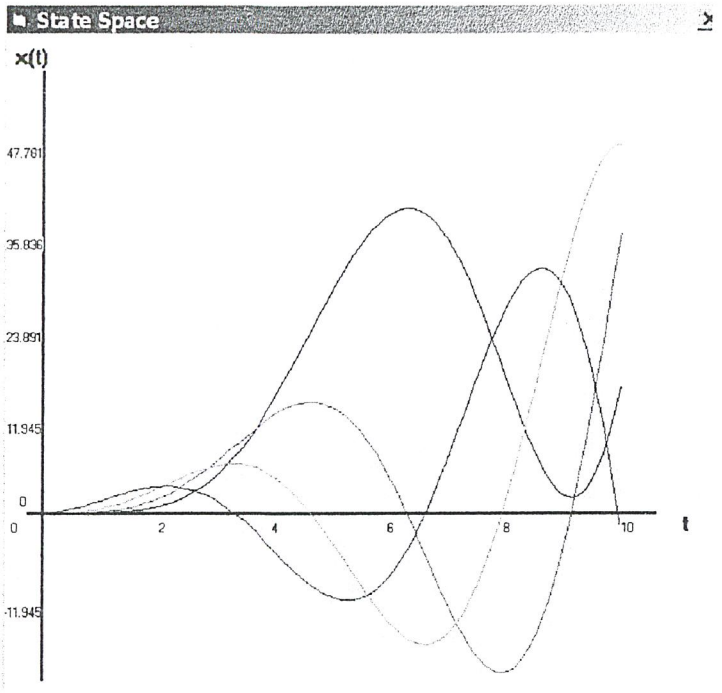
4. ป้อนค้าขนาดของอินพุตตามที่โจทย์กำหนด
5. เลือก *ชนิดอินพุต* ตามที่โจทย์กำหนดซึ่งเมื่อกดปุ่มนี้แล้วจะแสดงกราฟเปรียบเทียบสภาวะแต่ละตัวซึ่งมีจำนวนเท่ากับมิติของ A



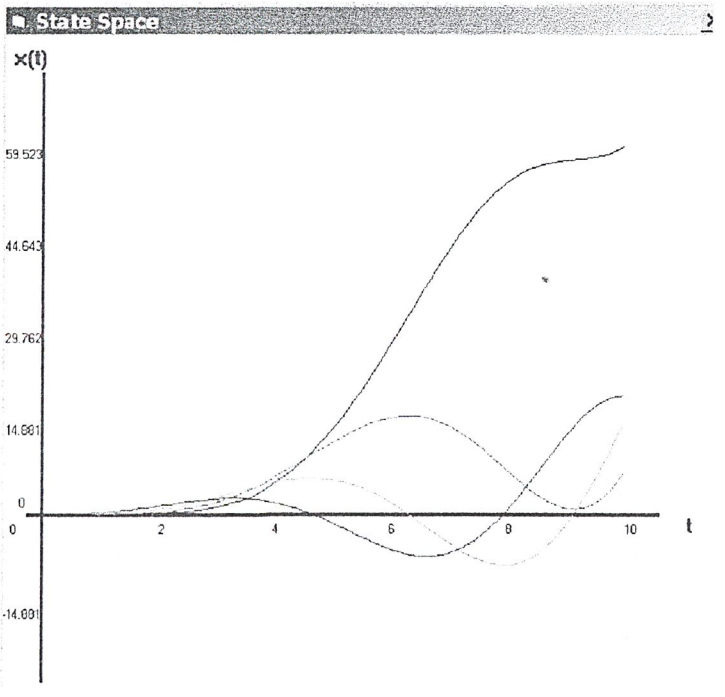
(a) เมื่อทำตามขั้นตอน 1 – 5 โดยป้อน K = 1 เลือก *Step Input* แล้วจะได้ผลดังนี้



(b) เมื่อทำตามขั้นตอน 1 – 5 โดยป้อน K = 5 เลือก *Ramp Input* แล้วจะได้ผลดังนี้

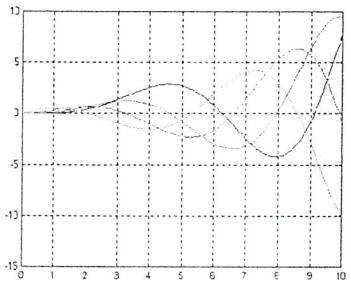


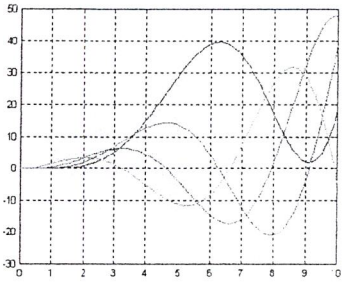
(c) เมื่อทำตามขั้นตอน 1 – 5 โดยป้อน K = 2 เลือก *Parabolic Input* แล้วจะได้ผลดังนี้

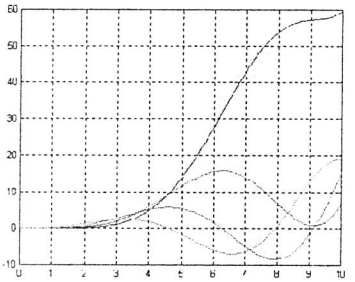


สำหรับการตรวจสอบคำตอบนี้เราจะใช้คำสั่ง [YS,TS,XS] = LSIM(SYS,U,T,X0) , [Y,T,X] = STEP(SYS) , SYS = SS(A,B,C,D) และ PLOT(X,Y) ของ MATLAB®

```
A =[0 1 0 0;0 0 1 0;0 0 0 1;-1 -1 -1 -1]; B = [0;0;0;1];
C = [0 0 0 1]; D = [0];
sys = ss(A,B,C,D);
t = [0:0.1:10];
for i = 1:101
u2(i) = 5*t(i);
u3(i) = t(i)*t(i);
end;
x0 = [0;0;0;0];
[Y,T,X1] = STEP(sys,t);
plot(t,X1);
[Y2,T2,X2] = LSIM(sys,u2,t,x0);
plot(t,X2);
[Y3,T3,X3] = LSIM(sys,u3,t,x0);
plot(t,X3);
```







6. ความสามารถในการควบคุมได้และสังเกตได้

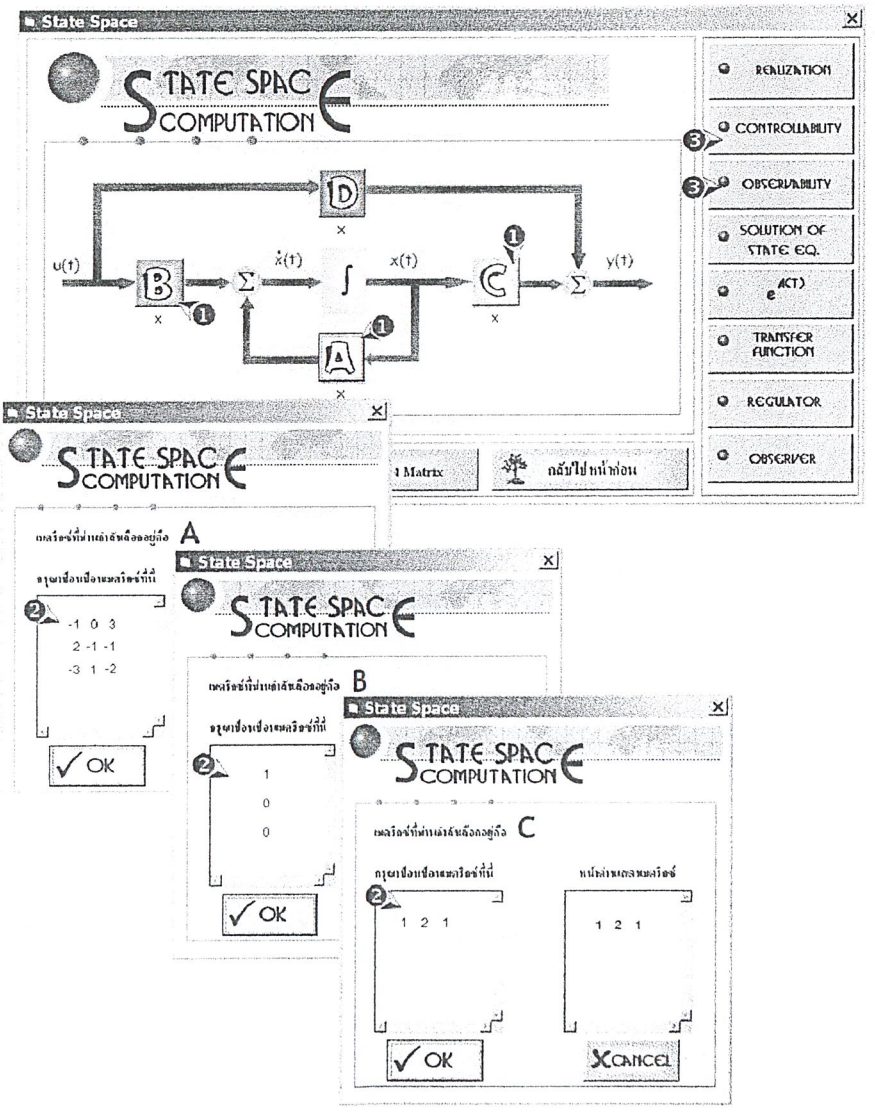
ตัวอย่างที่ 6.1 พิจารณาระบบด้านล่างที่กำหนดให้ว่า ควบคุมได้ หรือ สังเกตได้ หรือไม่

$$\dot{x}(t) = \begin{bmatrix} -1 & 0 & 3 \\ 2 & -1 & -1 \\ -3 & 1 & -2 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(t)$$

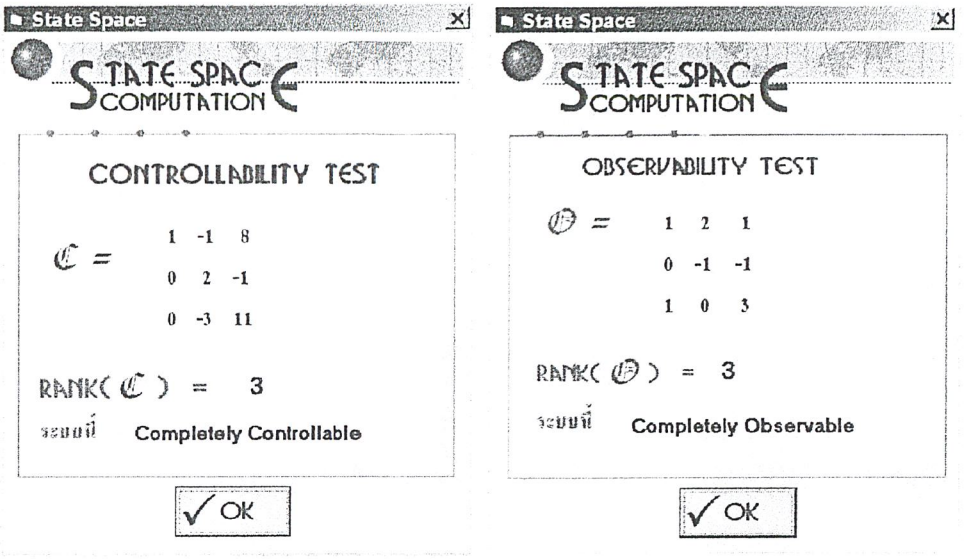
$$y(t) = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} x(t)$$

วิธีทำ

1. เลือกป้อน เมตริกซ์ A , Bและ C ที่หน้าจอ สเตทสเปซ
2. ป้อนค่า เมตริกซ์ A , B และ C ตามที่โจทย์กำหนด
3. กดปุ่ม *Controllability* และ *Observability*



ซึ่งจะได้ผลดังต่อไปนี้



สำหรับการตรวจสอบคำตอบนี้เราจะใช้คำสั่ง CO = CTRB(A,B) , OB = OBSV(A,C) และ RANK(A) ของ MATLAB®

```
A = [-1 0 3;2 -1 -1;-3 1 -2]; B = [1;0;0]; C = [1 2 1];
CO = ctrb(A,B)
CO =
          1     -1     -8
          0      2     -1
          0     -3     11
r1 = rank(CO)
r1 =
          3
B = obsv(A,C)
OB =
          1      2      1
          0     -1     -1
          1      0      3
r2 = rank(OB)
r2 =
          3
```

ตัวอย่างที่ 6.2 พิจารณา ฟังก์ชันถ่ายโอน ต่อไปนี้

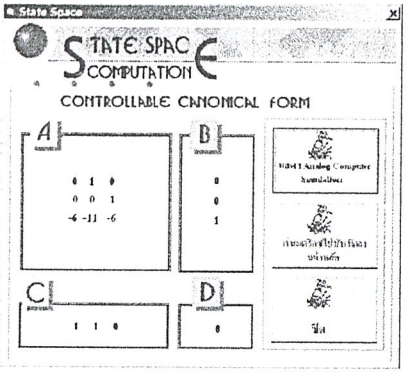
$$G(s) = \frac{s+1}{s^3 + 6s^2 + 11s + 6}$$

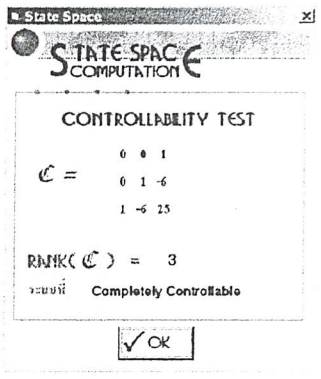
ให้ Realization ไปใน รูปแบบ Controllable Canonical, รูปแบบ Observable Canonical และ รูปแบบ Jordan Canonical จากนั้นตรวจสอบ ความสามารถในการควบคุมได้และสังเกตได้ ใน

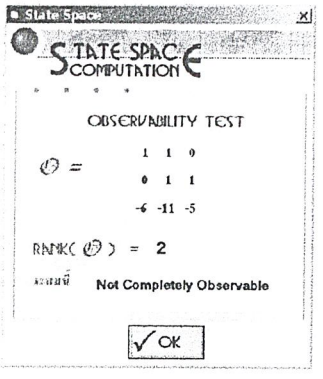
แต่ละ รูปแบบ

วิธีทำ

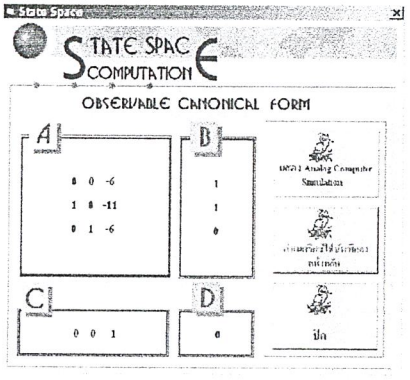
(a) Controllable Canonical Form

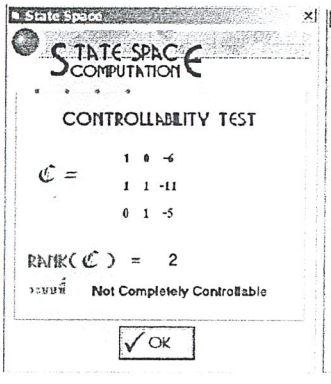


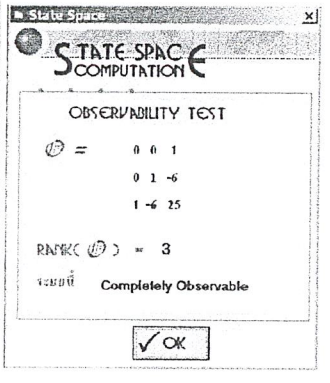




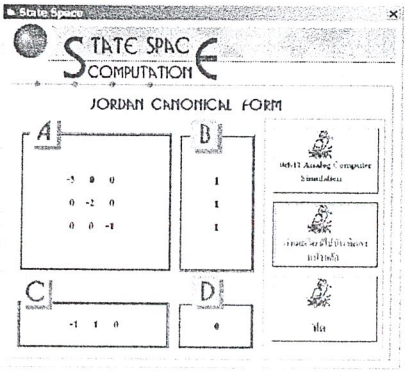
(b) Observable Canonical Form

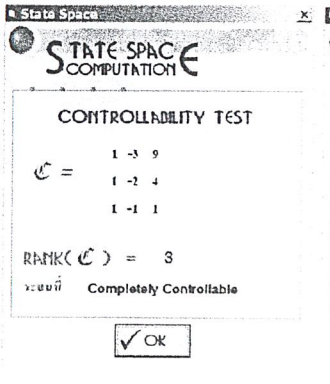


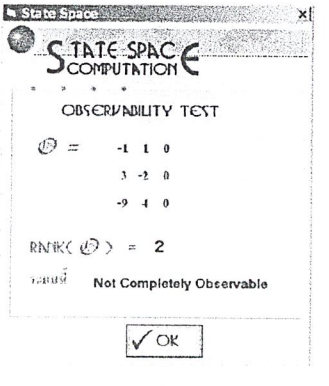


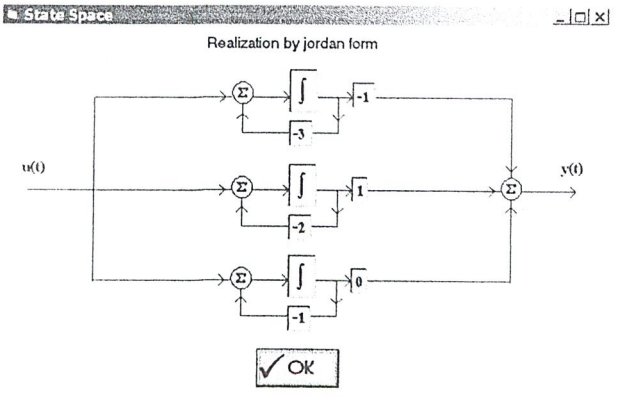


(c) Jordan Canonical Form







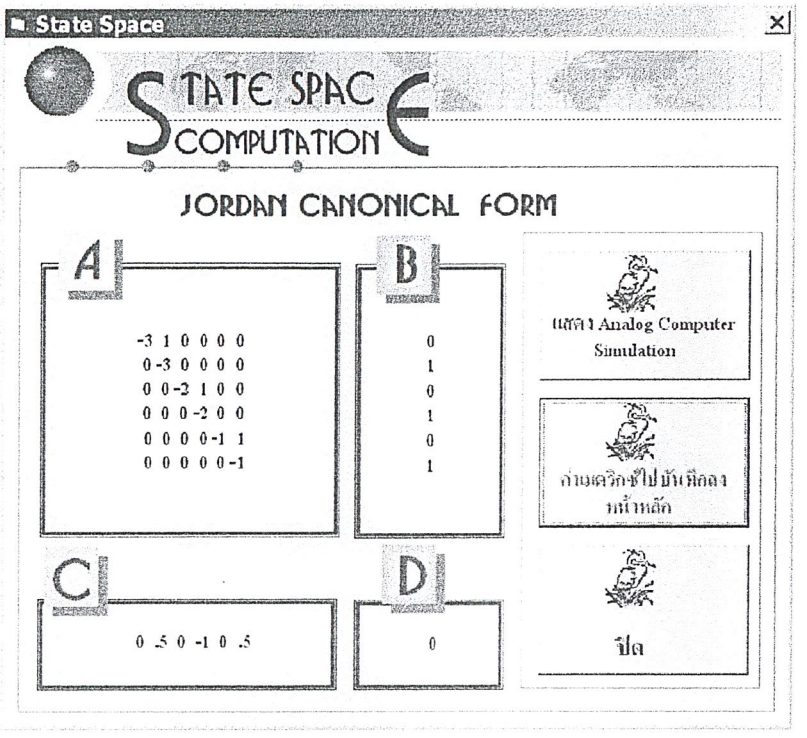


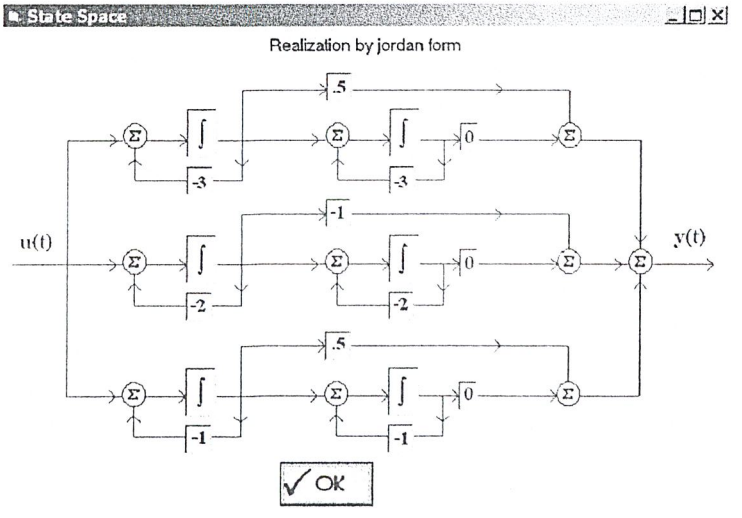
ตัวอย่างที่ 6.2 พิจารณา ฟังก์ชันถ่ายโอน ต่อไปนี้

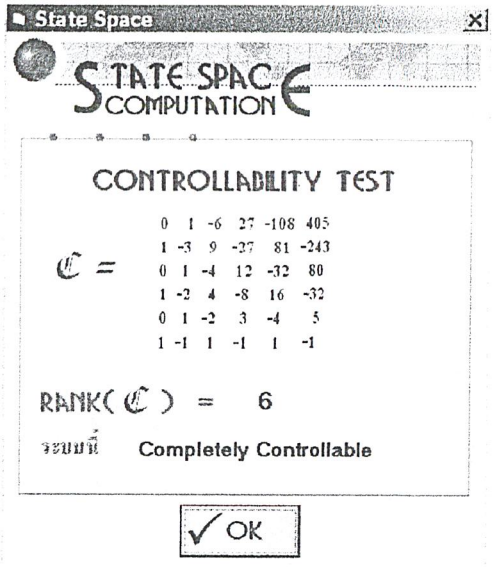
$$G(s) = \frac{s^3 + 6s^2 + 11s + 6}{s^6 + 12s^5 + 58s^4 + 144s^3 + 193s^2 + 132s + 36}$$

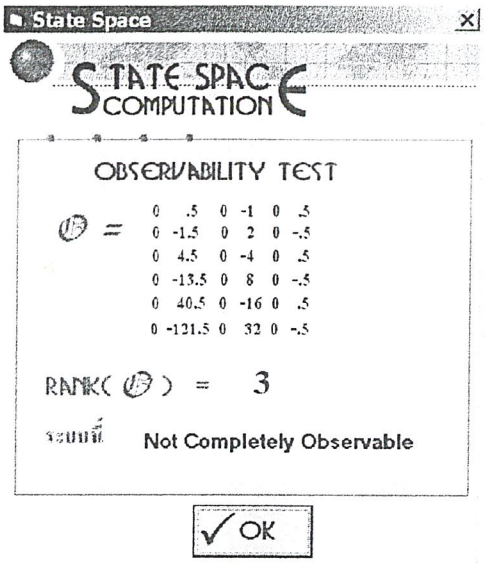
ให้ Realization ไปใน รูปแบบ Jordan Canonical จากนั้นตรวจสอบความสามารถในการควบคุมได้ และความสามารถในการสังเกตได้

วิธีทำ จะได้ผลดังต่อไปนี้






State Space
STATE SPACE COMPUTATION
CONTROLLABILITY TEST
ℂ =
0 1 -6 27 -108 405
1 -3 9 -27 81 -243
0 1 -4 12 -32 80
1 -2 4 -8 16 -32
0 1 -2 3 -4 5
1 -1 1 -1 1 -1
RANK(ℂ) = 6
ระบบนี้ Completely Controllable
OK


State Space
STATE SPACE COMPUTATION
OBSERVABILITY TEST
𝒪 =
0 .5 0 -1 0 .5
0 -1.5 0 2 0 -.5
0 4.5 0 -4 0 .5
0 -13.5 0 8 0 -.5
0 40.5 0 -16 0 .5
0 -121.5 0 32 0 -.5
RANK(𝒪) = 3
ระบบนี้ Not Completely Observable
OK

7. Decompose systems

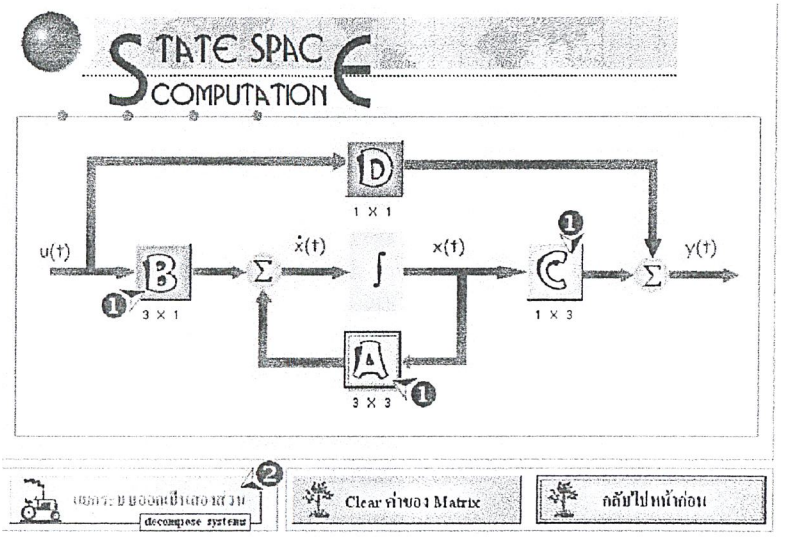
ตัวอย่างที่ 7.1 แยกระบบด้านล่างนี้ออกเป็น Controllable และ Uncontrollable Subspace

$$\dot{x}(t) = \begin{bmatrix} 0 & 0 & -6 \\ 1 & 0 & -11 \\ 0 & 1 & -6 \end{bmatrix} x(t) + \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} u(t)$$

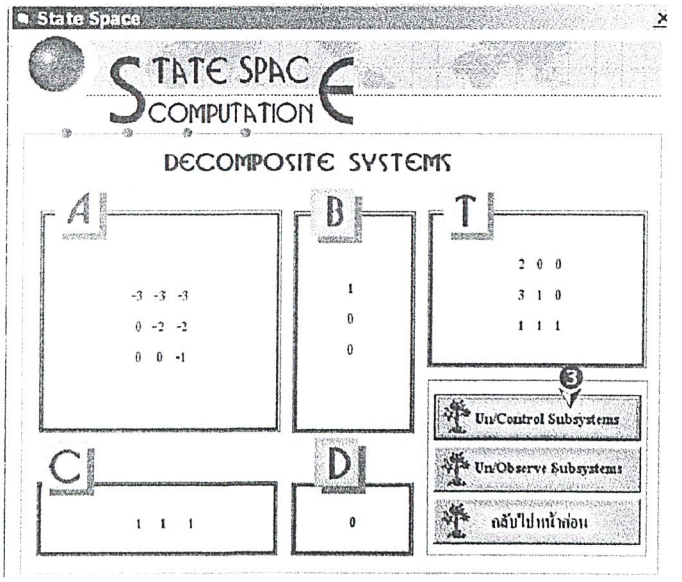
$$y(t) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} x(t)$$

วิธีทำ

1. ป้อนค่า เมตริกซ์ A B และ C ตามที่โจทย์กำหนดลงที่หน้าจอ สเตทสเปซ
2. กดปุ่ม *แยกระบบออกเป็นสองส่วน* อยู่ด้านมุมซ้ายล่างของหน้าจอ สเตทสเปซ



3. กดปุ่ม *Un/Controllable Subsystems*



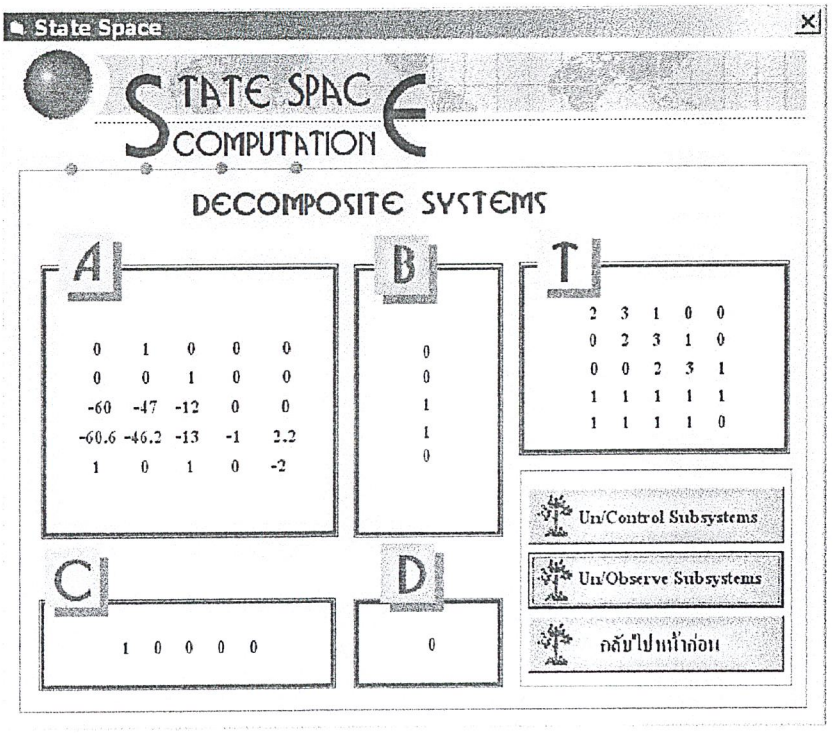
ตัวอย่างที่ 7.2 แยกระบบด้านล่างนี้ออกเป็น Observable และ Unobservable Subspace

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ -120 & -274 & -225 & -85 & -15 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 2 & 3 & 1 & 0 & 0 \end{bmatrix} x(t)$$

วิธีทำ

ทำเช่นเดียวกับตัวอย่างที่ 7.1 แต่ในขั้นตอนที่ 3 เลือก *Un/Observable Subsystems*ซึ่งจะผลดังต่อไปนี้



ตรวจสอบด้วยการใช้คำสั่งหาค่า ค่าไอเก็น E = EIG(A) ของ MATLAB®

```
A = [  0     1     0     0     0
       0     0     1     0     0
       0     0     0     1     0
       0     0     0     0     1
    -120   -274   -225   -85   -15 ];
Abar = [  0     1     0    0    0
          0     0     1    0    0
        -60   -47   -12    0    0
       -60.6  -46.2 -13   -1   2.2
         1     0     1    0   -2   ];
```

```
E1 = eig(A)
E1 =
      -5.0000
      -4.0000
      -3.0000
      -2.0000
      -1.0000
```

```
E2 = eig(Abar)
E2 =
      -1.0000
      -2.0000
      -5.0000
      -4.0000
      -3.0000
```

8. State - Variable Feedback

ตัวอย่างที่ 8 หา เมตริกซ์ อัตราขยายสภาวะป้อนกลับ ที่จะวางตำแหน่ง ค่าไอเก็น ใหม่เป็น

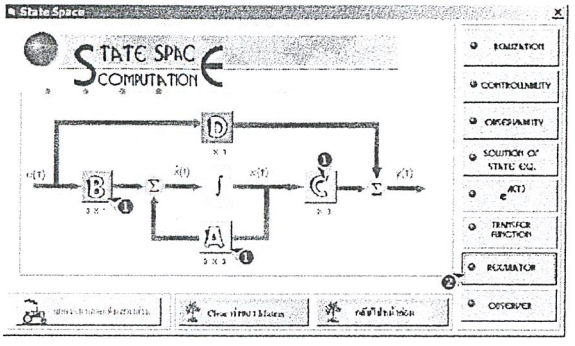
$$-4\,,\,-4\pm 3.7417\,j$$

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -20 & -6 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t)$$

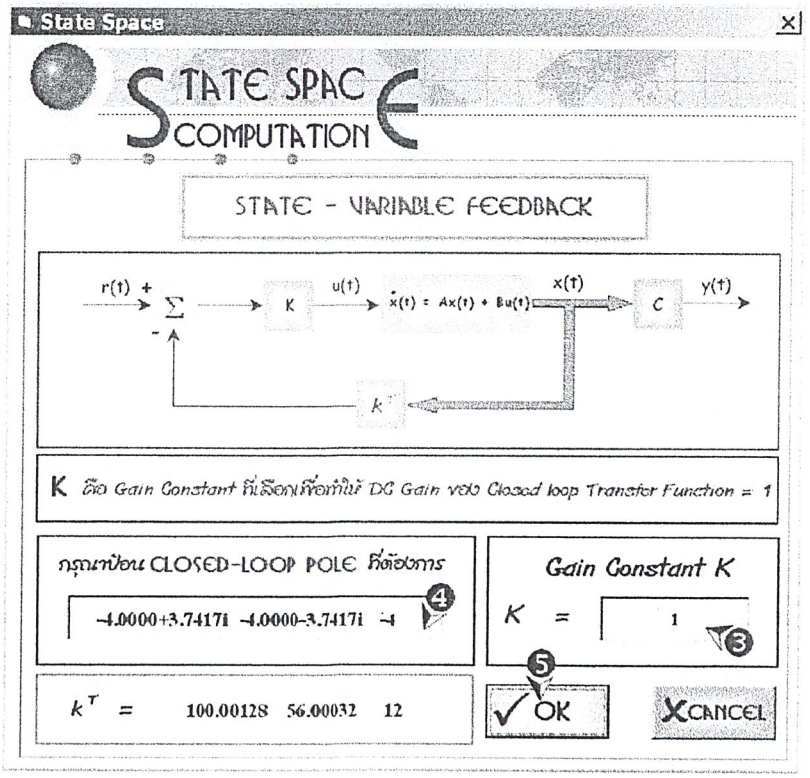
$$y(t) = \begin{bmatrix} 3 & 1 & 0 \end{bmatrix} x(t)$$

วิธีทำ

1. ป้อนค่า เมตริกซ์ A B และ C ตามที่โจทย์กำหนดลงที่หน้าจอ สเตทสเปซ
2. กดปุ่ม *Regulator* อยู่ด้านมุมขวาล่างของหน้าจอ สเตทสเปซ



3. ป้อนค่า อัตราขยายคงที่
4. ป้อนค่าตำแหน่ง ค่าไอเก็น ใหม่
5. กด ปุ่ม OK



ตรวจสอบด้วยการใช้คำสั่ง K = PLACE(A,B,P) ของ MATLAB®

```
A = [0  1  0 ;0  0  1 ;-20  -6  0 ];
B = [0  ;  0 ; 1 ];
C = [3  1  0];
P =[ [-4.0000 + 3.7417i;-4.0000- 3.7417i;-4]];
place(A,B,P)
place: ndigits=

ans =

   100.0013    56.0003    12.0000
```

9. Asymptotic State Observer

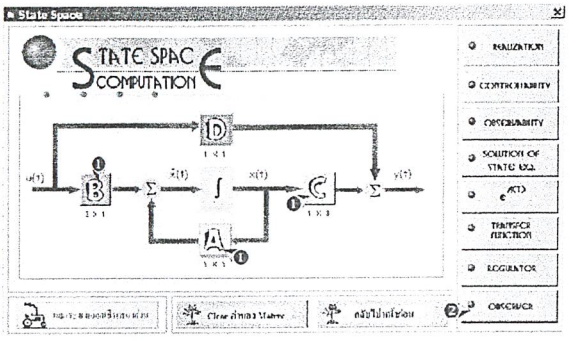
ตัวอย่างที่ 9 จงออกแบบ Observer สำหรับระบบด้านล่าง โดยให้ Observer มี ค่าไอเก็น ที่ $-4, -4 \pm 2j$

$$\dot{x}(t) = \begin{bmatrix} -1 & 0 & 3 \\ 2 & -1 & -1 \\ -3 & 1 & -2 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t)$$

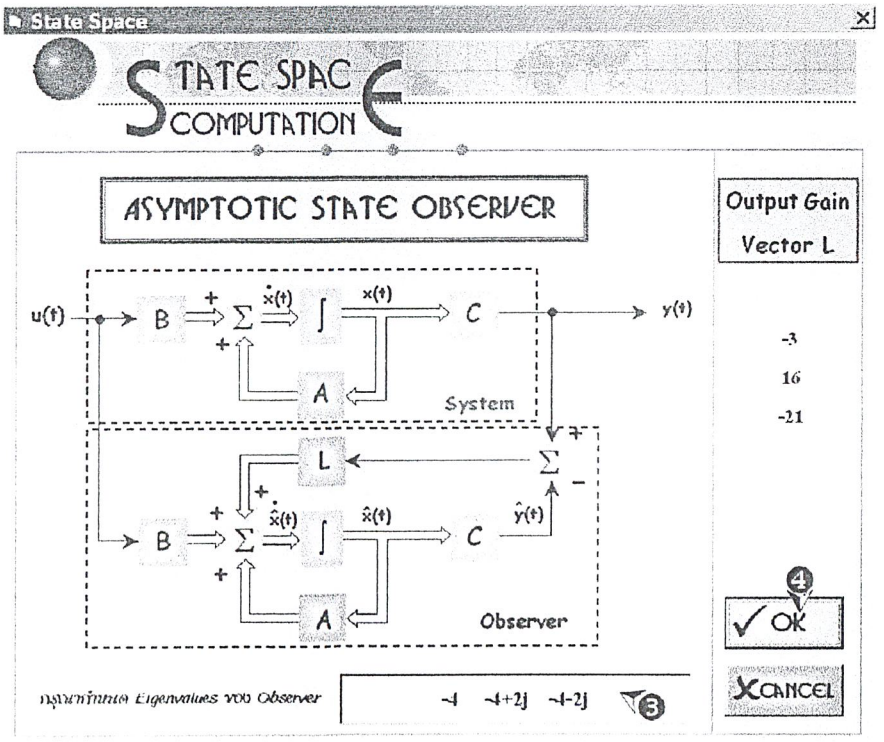
$$y(t) = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} x(t)$$

วิธีทำ

1. ป้อนค่า เมตริกซ์ A B และ C ตามที่โจทย์กำหนดลงที่หน้าจอ สเตทสเปซ
2. กดปุ่ม *Observer* อยู่ด้านมุมขวาล่างของหน้าจอ สเตทสเปซ



3. ป้อน ค่าไอเก็น ของ Observer
4. กดปุ่ม OK



ภาคผนวก (ข)

หนังสืออ้างอิง

หนังสืออ้างอิง

1. Atkinson, Kendall E., *An Introduction to numerical analysis*, 2nd ed., John Wiley & Sons. 1989. (ISBN 0 – 471 – 50023 – 2)
2. Bay, John S., *Fundamental of Linear State Space Systems*, McGraw – Hill, 1999. (ISBN 0 – 256 – 24639 – 4)
3. Borse, G.J. (Garold J.), *Numerical method with* MATLAB®, PWS Publishing Company, 1997. (ISBN 0 – 534 – 93822 – 1)
4. Brogan, William L., *Modern Control Theory*, 3rd ed., Prentice – Hall, 1991. (ISBN 0 – 13 – 589763 –7)
5. Burden, Richard L., *Numerical Analysis*, PWS Publishing Company, 1985. (ISBN 0 – 53494 – 585 – 2)
6. Chen, Chi-Tsong. *Linear system theory and design*, Holt Rinechart and Winston , Inc. , 1970 (ISBN 0 –03 – 060289 – 0)
7. Fausett, Laurence V., *Applied Numerical Analysis Using* MATLAB®, Prentice – Hall, 1999. (ISBN 0 – 13 – 319849 – 9)
8. Fröberg Carl Erik, *Introduction to numerical analysis*, The Benjamin/Cummings Publishing Company, 1985. (ISBN 0 – 8053 – 2530 – 1)
9. Furuta, Katsuhisa, *State Variable Methods in Automatic Control*, Corona Publishing, 1940. (ISBN 0 – 471 – 91877 – 6)
10. Gerald, Curtis F., *Applied Numerical Analysis*, 6th ed., Addison – Wesley. 1999. (ISBN 0 – 201 – 47435 – 2)
11. Gisela Engeln – Müllges, *Numerical Algorithm with C*, Springer, 1996. (ISBN 3 – 540 – 60530 – 4)
12. Grantham, Walter J., *Modern control systems: analysis and design*, John Wiley & Sons, 1993. (ISBN 0 –471 – 93276 –1)
13. Kaczorek, T., *Linear control systems*, Research studied press Ltd, 1992. (ISBN 0 – 86380 – 126 – 9)
14. Nakamura, Shoichiro, *Applied Numerical Methods with Software*, Prentice – Hall, 1991. (ISBN 0 – 13 – 041047 – 0)
15. Ogata, Katsuhiko, *Discrete – time Control Systems*, Prentice – Hall, 1987. (ISBN 0 – 13 – 216227 – 5)

16. Petkov, P.Hr., *Computational methods for linear control systems*, Prentice Hall, 1991. (ISBN 0 – 13 – 161803 – 2)
17. Robert W. Horn beck, *Matrix Eigenvalue Problems*, Quantum Publisher, 1975. (ISBN 0 – 13 – 626614 – 2)
18. Rohrs, Charles E., *Linear Control System*, McGraw – Hill, 1993. (ISBN 0 – 07 –112847 – 6)
19. Rubio, J.E., *The Theory of Linear Systems*, Academic Press, 1971. (ISBN 0 – 12 – 601650 – 7)
20. Sinha, N.K., Linear Systems,John Wiley $ Sons., 1991. (ISBN 0 – 47 – 154451 – 7)
21. Sinha, N.K., *Control system*, CBS College Publishing, 1996. (ISBN 0 – 246 – 76259 – 3)