สรุป Octave เบื้องต้น

เป็นเอกสารที่กล่าวถึงการทำงานอย่างไร การติดตั้ง และ port GNU Octave และ รายการ bug เป็นอย่างไร

GNU Octave

คือ ภาษาระดับสูง มีจุดมุ่งหมายหลัก ของการคำนวณตัวเลขคอมพิวเตอร์ เป็นคำสั่งที่สะดวกใช้ในการเชื่อมต่อสำหรับแก้ปัญหา linear และ nonlinear ของตัวเลข
การแบ่ง Software เป็นอิสระ คุณอาจจะแบ่ง หรือเปลี่ยนแปลง ภายใต้ term ของ GNU General Public License ซึ่ง published โดย หา Software อิสระ

เอกสารที่แสดงถึง Octave V.2.1x

Running Octave

ระบบจะมีการร้องขอ Octave คือ ใช้คำสั่ง “Octave ” Octave จะโชว์การเริ่มข้อความ prompt แสดงการอ่านข้อมูลเข้า และจะเริ่มใช้คำสั่งต่างๆ หลังคำๆ นี้ และถ้าเจอกับปัญหาต่างๆ เข้าก็จะสามารถใช้ Ctrl + c เพื่อหยุดงานระหว่างการทำงาน และใช้ quit หรือ exit เพื่อออกจากโปรแกรม
หรือใช้ Ctrl + z เพื่อหยุดการทำงานชั่วคราว

Simple Examples

Creating a Matrix

คือการคิดคำนวณและเก็บค่าในรูปแบบตัวแปร ซึ่งสามารถนำมาใช้อ้างอิง
ตามชนิดของคำสั่งได้ octave : 1 > a = [1, 1, 2 ,3 ,5 ,8 ,13, 21 ,24] แต่ถ้าเราต้องการเก็บค่าไว้ เราต้องใช้เครื่องหมาย ; (semicolon) octave : 2 > b = rand (3,2) ; จะทำให้เกิด matrix นี้มี 3 แถว 2 คอลัมน์ โดยที่ค่าของแต่ละตัวนั้น จะสุ่มมาจาก 0 ถึง 1

Matrix Arithmetic Octave
จะเป็นตัวดำเนินการที่ช่วยเพิ่มความสะดวกในการคำนวณเกี่ยวกับตรีโกณมิติ เช่น ซึ่งจะคูณ Matrix a ด้วย Scalar ค่าหนึ่งในคำสั่งดังนี้

Octave : 4 > 2 * a หรือ ให้ matrix a,b คูณกัน ใช้คำสั่งดังนี้

Octave : 5 > a * b หรือ ให้ transpose ของ a คูณกับ matrix a ใช้คำสั่งดังนี้

Octave : 6 > a′ * a

Solving Linear Equation (สมการเชิงเส้นอยู่ในรูปของ ax = b)

โดยใช้ตัวดำเนินการ “ \ ”

Octave : 7 > a \ b

Integrating Differential Equation

Octave จะสร้าง ฟังก์ชั่น ที่แก้ปัญหาของ nonlinear differential Equation ในรูปของ

$$\frac{dx}{dt} = f(x, t)$$

และเงื่อนไขเริ่มต้นที่

$$x(t = t0) = x0$$

สำหรับรูปแบบของ Integrating Equation อย่างแรกต้องแยกตัวกำหนดของ ฟังก์ชัน f (x,t) โดยการใส่ ฟังก์ชันบอดี้ เข้าไปโดยตรง สำหรับตัวอย่างนี้เราจะให้ฟังก์ชันอยู่ทางขวามือของสิ่งที่เราสนใจ ซึ่งเป็น nonlinear differential Equation ขณะที่ใส่ฟังชันเข้าไป Octave ก็จะตอบกลับ และจะรอให้ป้อน input เข้าไปจนเสร็จ

```
octave:8> function xdot = f (x, t)
>
>  r = 0.25;
>  k = 1.4;
>  a = 1.5;
>  b = 0.16;
>  c = 0.9;
>  d = 0.8;
>
```

```
> xdot(1) = r*x(1)*(1 - x(1)/k) - a*x(1)*x(2)/(1 + b*x(1));
> xdot(2) = c*a*x(1)*x(2)/(1 + b*x(1)) - d*x(2);
>
> endfunction
```

ให้เงื่อนไขเริ่มต้นเป็น

```
x0 = [1; 2];
```

แล้วจะให้ output เวลาเป็น column vector

```
t = linspace (0, 50, 200)';
```

จะทำให้การ Integrat ง่ายขึ้น จากการset differential Equation

```
x = lsode ("f", x0, t);
```

Conventions

ส่วนนี้จะอธิบายเกี่ยวกับการแปลงสัญญาณ

1. Fonts

Octave Code ที่ปรากฏในรูปของตัวอักษร หรือรูปแบบ svd(a) ชื่อที่นำเสนอ นี้จะขัดแย้งกับตัวแปรที่ปรากฏในรูปแบบ first- number คำสั่งจะขึ้นอยู่กับชนิดแต่ละชนิด บางทีก็ปรากฏในรูปของ “Octave –no –init-file” หรือ foo--bar--baz สำหรับคีย์เฉพาะที่ปรากฏบนคีย์บอร์ด จะอยู่ในรูปแบบ ANY

2.Evaluation notation

ผลจากการแสดงค่าจะระบุด้วย “$\Longrightarrow$” เช่น

Sqrt (2)

$\Longrightarrow$ 1.4142

ในบางกรณี ค่าของ matrix จะย้อนกลับโดยการหาค่าและแสดง เช่น

[1,2;3,4]==[1,3;2,4]

$\Longrightarrow$ [1,0;0,1]

กรณีอื่นๆ เช่น

Eye(3)

1 0 0

0 1 0

0 0 1

บางครั้งมันจะช่วยในการอธิบายการหาค่า และก็มีที่การหาค่าจะแสดงผลที่เหมือนกัน ซึ่งเราจะใช้ “= = ” เช่น

```
rot90 ([1, 2; 3, 4], -1)
==
rot90 ([1, 2; 3, 4], 3)
==
rot90 ([1, 2; 3, 4], 7)
```

Printing notation

ในการแสดงตัวอักษรเมื่อต้องการหาค่า เราจะใช้ ‘-1’ ซึ่งค่านี้ก็จะกลับมาแสดง โดยใช้ “⇒” เช่น

```
printf ("foo %s\n", "bar")
    -| foo bar
    => 1
```

Error Message

บางครั้งสัญญาณจะมีการผิดพลาดปกติการแสดงข้อความนี้จะอยู่ภายในและจะแสดงกับ error : เช่น

```
struct_elements ([1, 2; 3, 4])
error: struct_elements: wrong type argument `matrix'
```

Format of Descriptions

ฟังก์ชัน คำสั่ง และตัวแปร จะอธิบายในรูปแบบที่มีการกำหนดไว้ อย่างแรกจะอธิบายเกี่ยวกับการบรรจุชื่อ ของ item โดยมันจะขัดแย้งกัน ฟังก์ชัน ตัวแปร หรืออย่างไรก็ตามที่ปรากฏในตอนเริ่มต้น

A sample function description

Function: **foo** ***(x, y, ...)***

The function foo subtracts *x* from *y*, then adds the remaining arguments to the result. If *y* is not supplied, then the number 19 is used by default.

foo (1, [3, 5], 3, 9)

=> [14, 16]

foo (5)

=> 14

More generally,

foo (*w*, *x*, *y*, ...)

==

x - *w* + *y* + ...

1. A Sample Command Description

คำสั่งอธิบาย จะมีความคล้ายคลึงกับ ฟังก์ชัน อธิบาย สำหรับใน Octave จะใช้คำสั่ง cd เช่น

Command: **cd** ***dir***

Command: **chdir** ***dir***

2. **A Sample Variable Description**

เช่น สิ่งที่เราไม่รู้ คือ do_what_i_mean_not_what_i_say.

เราก็จะสร้างตัวแปร do_what_i_mean_not_what_i_say

Getting Started

ส่วนนี้จะอธิบายเกี่ยวกับ คุณลักษณะพื้นฐานของ Octave เช่น การแก้ไขคำสั่ง การเขียนโปรแกรม

1. Invoking Octave ปกติแล้ว การ run ของ Octave จะปราศจากข้อขัดแย้งใดๆ Octave จะอ่านคำสั่งมันไปเรื่อยๆ จนกว่าจะออกจากโปรแกรม และเราสามารถ ใช้ชื่อพิเศษจาก ไฟล์บนคำสั่ง และ Octave จะทำการอ่าน และประมวลผลคำสั่งจากชื่อไฟล์

1.1 Command line option

ตัวอย่างรายชื่อ Command line option

--debug

-d

--echo-commands

-x

--exec-path *path*

--help

-h

-?

--info-file *filename*

--info-program *program*

--interactive

-i

--no-history

-H

--no-init-file

--no-line-editing

--no-site-file

.

--norc

-f

--path *path*

-p *path*

--silent

--quiet

-q

.

--traditional

--braindead

```
PS1                    = ">> "
PS2                    = ""
beep_on_error          = true
crash_dumps_octave_core = false
default_save_format    = "mat-binary"
fixed_point_format     = true
```

page_screen_output = false

print_empty_dimensions = false

warn_function_name_clash = false

--verbose

-V

Turn on verbose output.

--version

-v

1.2 Startup Files

Octave จะเริ่มทำงานเมื่อ มีการป้อนคำสั่งเข้าไป คำสั่งต่างๆ นี้ คือฟังก์ชัน ที่มีการกำหนดไว้ในตัวของมันเอง

1.3 Quitting Octave

จะใช้คำสั่ง 2 อย่าง คือ exit , quit สำหรับ status

และใช้ atexit สำหรับฟังก์ชัน

Command for Gatting help

ความสมบูรณ์ของการลงมือ คือประโยชน์จาก Octave prompt ผ่านคำสั่ง help – i เพื่อ ในข้อมูล สำหรับเฉพาะคนที่เขียน ฟังก์ชัน และตัวแปร คือ ประโยชน์ที่ผ่าน help command คำสั่งที่จะใช้ในการอธิบาย การอ่าน ของผู้ลงมือ และข้อมูลสำหรับใส่และตัวแปร

Command: help

Octave's help command can be used to print brief usage-style messages, or to display information directly from an on-line version of the printed manual, using the GNU Info browser. If invoked without any arguments, help prints a list of all the available operators, functions, and built-in variables. If the first argument is -i, the help command searches the index of the on-line version of this manual for the given topics.

สำหรับคำสั่ง นี้จะช่วยในการเสนอข้อความสั้นๆ คำสั่ง help และ help – i ช่วยเริ่ม GNU : the on-line version ขอผู้ลงมือ

GNU Info Browser

การทำงานที่ช่วย command C-h.

Help command สามารถให้ข้อมูลเกี่ยวกับปฏิบัติ แต่ , และ ; เป็นคำสั่งที่ใช้แยกต่างหาก

Built-in Variable: INFO_FILE

The variable INFO_FILE names the location of the Octave info file. The default value is "*octave-home*/info/octave.info", in which *octave-home* is the directory where all of Octave is installed.

Built-in Variable: MAKEINFO_PROGRAM

The variable MAKEINFO_PROGRAM names the makeinfo program that Octave runs to format help text that contains Texinfo markup commands. Its default initial value is "makeinfo".

Command Line Editing

Octave ใช้ GNU อ่าน line library ถึงองค์ประกอบของ Command Line Editing และคุณลักษณะพื้นฐาน คืออธิบายการทำงาน

Octave จะเพิ่ม คุณลักษณะที่ cursor และ cursor forward

Command Line Editing มีหน้าที่ในการควบคุมคุณลักษณะ สำหรับตัวอย่าง คุณลักษณะของ Ctrl – a หรือ C- a จะใช้สำหรับย้าย cursor ไปยังจุดเริ่มต้น

1. **Cursor Motion**

   ตัวอย่างคำสั่งที่ใช้ในการย้ายตำแหน่ง cursor

C-b

Move back one character.

C-f

Move forward one character.

DEL

Delete the character to the left of the cursor.

C-d

Delete the character underneath the cursor.

M-f

Move forward a word.

M-b

Move backward a word.

C-a

Move to the start of the line.

C-e

Move to the end of the line.

C-l

Clear the screen, reprinting the current line at the top.

C-_

C-/

Undo the last thing that you did. You can undo all the way back to an empty line.

M-r

Undo all changes made to this line. This is like typing the `undo' command enough times to get back to the beginning.

จากตัวอย่างด้านบน จะอธิบายถึงพื้นฐาน ของการใช้คีย์ของคำสั่งชื่อที่ต้องการแก้ไข คุณสามารถใช้ลูกศร (>>) ใน C – f และ C – 6 ถึงการย้าย forward และ backward

2. Killing and Yanking

Killing text คือการลบ text จาก line แต่ต้องมีการ save ทุกครั้ง เพื่อใช้ในการดึงกลับ (Yanking)

รายการของคำสั่งสำหรับ Killing text

C-k

ลบ text จาก cursor ปัจจุบัน

M-d

ลบจากระหว่างคำ เมื่อจบคำต่อไป

M-DEL

ลบจาก cursor ถึงจุดเริ่มต้น

C-w

ลบจาก cursor ถึงจุดสิ้นสุด

3. Commands For Changing Text

ใช้สำหรับ การเข้าคุณสมบัติ

C-q

C-v

การเพิ่ม character

M-TAB

การ Insert tab character

C-t

การติดของ character ก่อน cursor จะส่ง character ที่ cursor ไปยังจุดที่สุด

M-t

การติดของคำที่อยู่ใกล้ cursor

4**. Letting Readline Type For You**

เป็นคำสั่งที่ยอมให้ Octave ใช้กับคำสั่ง complete และชื่อตัวแปร

Tab พยายามจะทำให้สมบูรณ์ บน text ก่อน cursor

M - ? รายการความเป็นไปได้

Built-in Variable: completion_append_char

The value of completion_append_char is used as the character to append to successful command-line completion attempts. The default value is " " (a single space).

Built-in Function: completion_matches *(hint)*

Generate possible completions given *hint*.

5. Commands For Manipulating The History

Octave แก้ไขการเก็บ track ของคำสั่ง ซึ่งคุณสามารถเรียกกลับเพื่อมาแก้ไข หรือกระทำได้อีกครั้ง เมื่อคุณออกจาก Octave คำสั่งที่คุณทำมาก่อนหน้านี้ จะมีการ up ตัวเลข ชนิดโดยตัวแปร history – size คือมีการ save file เมื่อ Octave ทำงานอีกครั้ง มันจะ โหลด รายการเริ่มต้นของคำสั่งจาก ไฟล์โดยตัวแปร history – file

C-p

Move `up' through the history list.

C-n

Move `down' through the history list.

M-<

Move to the first line in the history.

M->

Move to the end of the input history, i.e., the line you are entering!

C-r

Search backward starting at the current line and moving `up' through the history as necessary.

This is an incremental search.

C-s

Search forward starting at the current line and moving `down' through the history as necessary.

6. Customizing readline

Built-in Function: read_readline_init_file *(file)*

Read the readline library initialiazation file *file*. If *file* is omitted, read the default initialization file (normally `~/.inputrc'.

7. Customizing the Prompt

`\t'

The time.

`\d'

The date.

`\n'

Begins a new line by printing the equivalent of a carriage return followed by a line feed.

`\s'

The name of the program (usually just `octave').

`\w'

The current working directory.

`\W'

The basename of the current working directory.

`\u'

The username of the current user.

`\h'

The hostname, up to the first `.'.

`\H'

The hostname.

`\#'

The command number of this command, counting from when Octave starts.

`\!'

The history number of this command. This differs from `\#' by the number of commands in the history list when Octave starts.

`\$'

If the effective UID is 0, a `#', otherwise a `$'.

`\nnn'

The character whose character code in octal is *nnn*.

`\\'

A backslash.

Data Types

แบ่งเป็น 3 ประเภท

1. Built-in Data Types

1.1 Numeric Objects

ซึ่งประกอบด้วยจำนวนจริง และจำนวนเชิงซ้อนรวมทั้ง matrix ด้วย โดยที่ค่าจะอยู่ในช่วง2.225e-308ถึง1.1977e308โดยค่าที่แน่นอนประมาณ 2.22.4 e -16 โดยที่ความแน่นอนจะกำหนดโดยตัวแปร realmin , realmax and eps ส่วน Matrix เราสามารถเพิ่มได้ ทั้งรูปร่างขนาด ได้ตามต้องการซึ่งทำได้ไม่ยาก

1.2 Missing Data

Built-in Variable: NA

Missing value.

Mapping Function: isna *(x)*

คืนค่า 1 สำหรับค่าที่หายไป นอกนั้นเป็น 0

```
is_NA ([13, Inf, NA, NaN])
  => [ 0, 0, 1, 0 ]
```

Mapping Function: is_nan_or_na *(x)*

คืนค่า 1 สำหรับค่าที่หายไป นอกนั้นเป็น 0

is_NAN_or_NA ([13, Inf, NA, NaN])

=> [0, 0, 1, 1]

1.3 String Objects

ลักษณะของ string ใน Octave เป็นตัวอักษรตั้งแต่ 1 – 256 ตัวอักษร

1.4 Data Structure Objects

Data Structure ใน Octave จะช่วยให้เราดูความแตกต่างระหว่างข้อมูล

2. User-defined Data Types

บางครั้งที่เราจะอธิบายส่วนประกอบของ เชิงกล ของ Octave เราจะทำได้โดยการอ่าน code “or.h” “ops.h”

3. Object Sizes

สามารถกำหนดขนาดของตัวแปรหรืออธิบายฟังก์ชันต่างๆ ที่นิยามไว้ได้ทั้งหมด

Function File: columns *(a)*

คืนค่า จำนวน columns (a)

Function File: rows *(a)*

คืนค่า จำนวน rows (a)

Built-in Function: length *(a)*

คืนค่า length เมื่อ length คือ จำนวน columns หรือ rows

Built-in Function: size *(a, n)*

คืนค่า 1 เมื่อ a เป็น Matrix ว่าง และ 0 สำหรับค่าอื่น

Numeric Data Types

ค่าคงที่ ส่วนใหญ่จะเป็น Scalar , vector หรือ matrix ซึ่งจะมีจำนวนเชิงซ้อนเกี่ยวข้องด้วย ตัวอย่างที่ง่ายที่สุด ก็คือ ค่าคงที่ scalar ที่เป็นตัวเลข 1 หลัก แต่สามารถเขียนให้อยู่ได้ในหลายรูปแบบ เช่น

105

1.05e+2

1050e-1

จำนวนเชิงซ้อนก็สามารถจัดรูปแบบนี้ได้เช่นกัน

3 + 4i

3.0 + 4.0i

0.3e1 + 40e-1i

แต่ใน Octave ไม่ปรากฏค่า i แต่เราสามารถให้ "j" หรือ "J" แทนได้

1. Matrices

1.1 Empty Matrices

Empty Matrices จะนิยามได้ว่า matrix ที่มี 1 หรือ 2 dimensions เป็น 0 และดำเนินการบน matrix ที่ว่าง

ก็คือ ถ้าให้มี scalar s matrix M(mxn) , matrix[ ](mxn) จะได้ว่า

s * [](mxn) = [](mxn) * s = [](mxn)

[](mxn) + [](mxn) = [](mxn)

[](0xm) * M(mxn) = [](0xn)

M(mxn) * [](nx0) = [](mx0)

[](mx0) * [](0xn) = 0(mxn)

ถ้าเราสร้างตัวแปร print_empty_dimensions ถ้าค่าของมัน ≠ 0 มิติของ matrix ว่าง
จะมาจากที่มิติที่≠ 0 เช่น

zeros (3, 0)

จะได้

ans = [](3x0)

ถ้าสร้างตัวแปร warn_empty_list_elements ถ้าค่าของมัน ≠ 0 มันจะเตือนเมื่อ matrix ว่างถูกเจอใน matrix เช่น

a = [1, [], 3, [], 5]

default value เป็น 0.

2. Ranges

จะข้ามการเขียนให้ row vector ง่ายขึ้น เราจะอธิบาย range โดยกำหนดให้ค่าแรกใน range เพื่อขึ้น ระหว่างค่าต่างๆ และค่าสูงสุดของแต่ละตัวจะไม่มากกว่า range

3. Logical Values

สร้างตัวแปร true สำหรับ Logical ที่เป็นจริง

สร้างตัวแปร false สำหรับ Logical ที่เป็นเท็จ

4. Predicates for Numeric Objects

Built-in Function: isnumeric *(x)*
สำหรับ return nonzero ถ้า x เป็น numeric object

Built-in Function: isreal *(x)*
สำหรับ return ค่า true ถ้า x เป็น ค่าจริง

Built-in Function: iscomplex *(x)*

สำหรับ return ค่า true ถ้า x เป็น จำนวนเชิงซ้อน

Built-in Function: ismatrix *(a)*

สำหรับ return 1 ถ้า a เป็น matrix อื่นๆ เช่น 0.

Function File: isvector *(a)*

สำหรับ return 1 ถ้า a เป็น vector อื่นๆ เช่น 0

Function File: isscalar *(a)*

สำหรับ return 1 ถ้า a เป็น scalar อื่นๆ เช่น 0

Function File: issquare *(x)*

ถ้า x เป็น square matrix เมื่อ return มิติของ x อื่นๆ เป็น 0

Function File: issymmetric *(x, tol)*

ถ้า x เป็น symmetric ภายใน tolerance พิเศษ โดย *tol*, return มิติ ของ x อื่นๆ เป็น 0 ถ้าข้าม *tol* ไป

Built-in Functio: isbool *(x)* Return true ถ้า *x* เป็น boolean

Statement

คำสั่งอาจจะแสดงคงที่หรือมีรายการประกอบเข้าด้วยกัน ซึ่งเป็นบ่อเกิดการ loops และเงื่อนไข คำสั่งที่ควบคุม เช่น if,while ดำส่งควบคุมทั้งหมดจะเริ่มด้นด้วย keyboard เช่น if,while ซึ่งจะแยกกัน

คำสั่งการควบคุมจะมีลักษณะเดียวกัน กับคำสั่งจบ เช่น

Keyword if คำสั่งจบคือ endif , keyword while คำสั่งจบคือ endwhile

The if statement

คำสั่ง if คือ Octave ตัดสินใจทำคำสั่ง มี 3 รูปแบบ

รูปแบบที่ 1

if (*condition*)

then-body

endif

Condition คือ ส่วนที่แสดงการควบคุมอะไรที่จะทำ then-body คือส่วนที่แสดง Condition เป็นจริง เงื่อนไขในคำสั่ง if คือพิจารณาเป็นจริง ไม่เท่ากับ 0 และเท็จ แต่ถ้าเงื่อนไขแสดงในคำสั่ง if เป็น vector หรือ matrix คือ พิจารณาเป็นจริงเท่านั้น ถ้าทุกส่วนไม่เท่ากับ 0

รูปแบบที่ 2

if (*condition*)

then-body

else

else-body

endif

ถ้าเงื่อนไขเป็นจริง , then-body คือแสดง 1 else-body จะเป็น 0

รูปแบบที่ 3

if (*condition*)

then-body

elseif (*condition*)

elseif-body

else

else-body

endif

ตัวเลขมากมายของ else,if ให้แสดง เงื่อนไข คือ ทดสอบในการย้อนกลับและถ้าเป็นจริง มันจะแสดงในทำนองเดียวกับใน body ถ้าไม่เป็นจริงจะไปยัง else

The switch statement

คำสั่ง switch จะพิจารณาการทดลองและรายละเอียดของเครื่องมือ อาจมีการเปลี่ยนแปลงเล็กน้อยในอนาคต

รูปแบบของคำส่ง switch คือ

```
switch expression
  case label
    command_list
  case label
    command_list
  ...

  otherwise
    command_list
endswitch
```

การกำหนด switch , case , otherwise และ endswitch คือ keyword

Label อาจจะมีหลาย expresstion

สามารถมี case label , command_list ได้หลายอัน

ค่าของ label จะไม่สามารถตรวจพบได้ เช่นเดียวกับ command_list

Otherwise command list คือ option

Built-in Variable: warn_variable_switch_label

ถ้าค่าตัวแปรไม่เท่ากับ 0 Octave จะเตือน ถ้า switch lebel ไม่คงที่

3 The while Statement

loops คือส่วนของโปรแกรมที่ทำงาน 2หรือมากกว่า 2 ติดต่อกัน

คำส่ง while คือการวน loop ของคำสั่งใน Otave มันจะทำซ้ำเมื่อเงื่อนไขเป็นจริง

เงื่อนไขคำสั่งของ if,while คือ จะพิจารณาเป็นจริง เมื่อมีค่าไม่เท่ากับ 0 และเป็นเท็จเมื่อเท่ากับ 0 ถ้าค่าของเงื่อนไขแสดงคำส่งใน while คือ vector หรือ matrix มันพิจารณาเป็นจริงเท่านั้นเมื่อทุกส่วนไม่เท่ากับ 0

while (*condition*)

body

endwhile

Body คือคำสั่งหรือรายการของคำสั่ง

Condition คือ expression ซึ่งควบคุมจำนวน loop ที่เก็บในการทำงานสิ่งแรกที่คำสั่ง while จะทำคือ ทดสอบ condition ถ้า condition เป็น false ตั้งแต่ต้น body จะเกิดการ loop การทำงาจะไม่จบ

```
fib = ones (1, 10);
i = 3;
while (i <= 10)
  fib (i) = fib (i-1) + fib (i-2);
  i++;
endwhile
```

4.The do-until Statement

คำสั่ง do-until เหมือนกับคำสั่ง while มันจะกระทำซ้ำเมื่อเงื่อนไขเป็นจริง และทดสอบเงื่อนไข loop ซึ่งถ้าbody ของ loop คือทั้งหมดของกากระทำ เงื่อนไขใน do-until จะพิจารณาเป็นจริงถ้าค่าไม่เท่ากับ 0 และเป็นเท็จเมื่อ เท่ากับ 0 ถ้าค่าของเงื่อนไขแสดงในคำสั่ง do-until คือ vector หรือ matrix จะพิจารณาเป็นจริงเท่านั้นถ้าทุกส่วนไม่เท่ากับ 0

do

body

until (*condition*)

Body คือ คำสั่งหรือรายการคำสั่ง ,condition คือ expression ซึ่งควบคุมจำวน loop ที่เก็บการทำงาน

```
fib = ones (1, 10);
i = 2;
do
  i++;
```

```
  fib (i) = fib (i-1) + fib (i-2);
until (i == 10)
```

5 The for Statement

คำสั่ง for จะสะดวกมากสำหรับการนำจำนวน loop รูปแบบคำสั่งfor มีหลายรูปแบบ

```
for var = expression
  body
endfor
```

Body สำหรับคำสั่งหรือรายการของคำสั่ง expression คือ ใช้ในการแสดง และ var อาจมีหลายรูปแบบ

```
fib = ones (1, 10);
for i = 3:10
  fib (i) = fib (i-1) + fib (i-2);
endfor
```

C-Style I/O Functions

ฟังก์ชันของ Octave ส่วนใหญ่มาจากภาษา C จะแตกต่างกันบ้านก็ตรงที่ input ที่ป้อนเข้าไปในโปรแกรม

ต่อจากนี้ ไฟล์ อ้างอิงจาก file name และ fid อ้างอิงที่ integer file name เป็นการ reture โดย fopen

มี 3 file ที่ always available

Built-in Variable: stdin

The standard input stream (file id 0). When Octave is used interactively, this is filtered through the command line editing functions.

Built-in Variable: stdout

The standard output stream (file id 1). Data written to the standard output is normally filtered through the pager.

Built-in Variable: stderr

The standard error stream (file id 2). Even if paging is turned on, the standard error is not sent to the pager. It is useful for error messages and prompts.

Opening and Closing Files

Built-in Function: [*fid*, *msg*] = fopen *(name, mode, arch)*

Built-in Function: *fid_list* = fopen *("all")*

Built-in Function: *file* = fopen *(fid)*

รูปแบบแรกของ fopen ฟังก์ชัน จะเปิดชื่อไฟล์ไว้กับ โหมดพิเศษ พร้อมกับแปลความหมายและ reture ค่า integer ที่อาจจะนำมาใช้อ้างอิงที่หลัง
รูปแบบที่สอง fopen ฟังก์ชัน จะ reture vector field
รูปแบบที่สาม fopen ฟังก์ชัน ชื่อของ currently ให้กับข้อมูล

Built-in Function: fclose *(fid)*

ปิดไฟล์เฉพาะ ถ้าทำสำเร็จ fclose จะเป็น 0 นอกจากนั้นจะเป็น 1

Simple Output

สร้างฟังก์ชัน

Built-in Function: fputs *(fid, string)*

เขียน string เข้าไปใน โดยที่จะไม่มีรูปแบบ

Built-in Function: puts *(string)*

เขียน string ส่งไปที่ output โดยที่จะไม่มีรูปแบบ

Line-Oriented Input

Built-in Function: fgetl ***(fid, len)***

อ่านคุณลักษณะจากไฟล์ และหยุดหลังจากขึ้นบรรทัดใหม่ โดยจะมีคุณลักษณะที่จำกัดเฉพาะสิ่งที่เป็นไปได้

Built-in Function: fgets ***(fid, len)***

อ่านคุณลักษณะจากไฟล์ และหยุดหลังจากขึ้นบรรทัดใหม่ โดยจะมีคุณลักษณะประกอบด้วยสิ่งที่เป็นไปได้

Formatted Output

ส่วนนี้จะอธิบายเกี่ยวกับ วิธีการใช้ printf และฟังชันที่เกี่ยวข้อง โดยที่จะกำหนดตัวแปรขึ้นมาแล้วหาค่าที่ output ซึ่งจะใช้ฟังก์ชันเหมือนภาษา c ส่วนใหญ่ แต่จะแตกต่างกันตรงที่ การแสดงจำพวก vector และ matrix

Built-in Function: printf ***(fid, template, ...)***

การแสดงส่วนประกอบของเหตุผลภายใต้การควบคุม รูปแบบ string จะกลายเป็น stdout

Built-in Function: fprintf ***(fid, template, ...)***

ฟังก์ชันที่จะเหมือนกับ printf จะแตกต่างตรงที่ output ที่ถูกเขียนขึ้น จะไหลไปแทนที่ stdout.

Built-in Function: sprintf ***(template, ...)***

ฟังก์ชันที่จะเหมือนกับ printf จะแตกต่างตรงที่ output จะ reture เหมือน string

Output Conversion for Matrices

เมื่อกำหนดค่าของ Matrix จะทำงานเป็นวงกลม วนไปเรื่อยๆ จนกระทั่ง ค่าทั้งหมดใน matrix แสดงออกมาหมดแล้ว เช่น

printf ("%4.2f %10.2e %8.4g\n", hilb (3));

-| 1.00 5.00e-01 0.3333

-| 0.50 3.33e-01 0.25

-| 0.33 2.50e-01 0.2

ถ้ามีค่าที่แสดงออกมามากกว่า 1 ค่า output จะไม่ กลับไปยังจุดเริ่มต้น จนกระทั่งเราได้นำเอาค่า หนึ่งออกไป ถ้าค่าใน matrix ไม่แน่นอน output จะไม่เกิดขึ้น(ปฏิเสธ output)

Output Conversion Syntax

จะพูดถึงการจัดการเกี่ยวกับโครงสร้างที่แน่นอนของคำสั่งพิเศษที่ปรากฏให้ printf โดยจะมีรูปแบบทั่วไป

% *flags width* [. *precision*] *type conversion*

Table of Output Conversions

ตัวอย่างของ different conversions

`%d', `%i'

Print an integer as a signed decimal number. See section Integer Conversions, for details. `%d' and `%i' are synonymous for output, but are different when used with scanf for input (see section Table of Input Conversions).

`%o'

Print an integer as an unsigned octal number. See section Integer Conversions, for details.

`%u'

Print an integer as an unsigned decimal number. See section Integer Conversions, for details.

`%x', `%X'

Print an integer as an unsigned hexadecimal number. `%x' uses lower-case letters and `%X' uses upper-case. See section Integer Conversions, for details.

`%f'

Print a floating-point number in normal (fixed-point) notation. See section Floating-Point Conversions, for details.

`%e', `%E'

Print a floating-point number in exponential notation. `%e' uses lower-case letters and `%E' uses upper-case. See section Floating-Point Conversions, for details.

`%g', `%G'

Print a floating-point number in either normal (fixed-point) or exponential notation, whichever is more appropriate for its magnitude. `%g' uses lower-case letters and `%G' uses upper-case. See section Floating-Point Conversions, for details.

`%c'

Print a single character. See section Other Output Conversions.

`%s'

Print a string. See section Other Output Conversions.

`%%'

Print a literal `%' character. See section Other Output Conversions.

Integer Conversions

อธิบายเกี่ยวกับส่วนประกอบของ `%d', `%i', `%o', `%u', `%x', และ `%X' อธิบายได้ดังนี้

The `%d' and `%i' conversion specifications both print an numeric argument as a signed decimal number; while `%o', `%u', and `%x' print the argument as an unsigned octal, decimal, or hexadecimal number (respectively). The `%X' conversion specification is just like `%x' except that it uses the characters `ABCDEF' as digits instead of `abcdef'.

The following flags are meaningful:

`-'

Left-justify the result in the field (instead of the normal right-justification).

`+'

For the signed `%d' and `%i' conversions, print a plus sign if the value is positive.

` '

For the signed `%d' and `%i' conversions, if the result doesn't start with a plus or minus sign, prefix it with a space character instead. Since the `+' flag ensures that the result includes a sign, this flag is ignored if you supply both of them.

`#'

For the `%o' conversion, this forces the leading digit to be `0', as if by increasing the precision. For `%x' or `%X', this prefixes a leading `0x' or `0X' (respectively) to the result. This doesn't do anything useful for the `%d', `%i', or `%u' conversions.

`0'

Pad the field with zeros instead of spaces. The zeros are placed after any indication of sign or base. This flag is ignored if the `-' flag is also specified, or if a precision is specified.

Floating-Point Conversions

อธิบายถึงความพิเศษของ **floating-point numbers(`%f', `%e', `%E', `%g', and `%G')**

`%f' จะเป็นรูปแบบของ [-]*ddd.ddd*

`%e' จะเป็นรูปแบบของ exponential [-]*d.ddd*e[+|-]*dd*

**`%E'** จะคล้ายกับ` e' ซึ่งบางครั้งก็สามารถใช้แทนกันได้

`%g' และ `%G' คล้ายกับ `%e' or `%E' ถ้าเป็น exponential ถ้าน้อยกว่า – 4 หรือมากกว่าหรือ เท่ากันค่าอื่นๆจะใช้ `%f'

ซึ่งสามารถดัดแปลงได้ดังนี้

`-'

Left-justify the result in the field. Normally the result is right-justified.

`+'

Always include a plus or minus sign in the result.

` '

If the result doesn't start with a plus or minus sign, prefix it with a space instead. Since the `+' flag ensures that the result includes a sign, this flag is ignored if you supply both of them.

`#'

Specifies that the result should always include a decimal point, even if no digits follow it. For the `%g' and `%G' conversions, this also forces trailing zeros after the decimal point to be left in place where they would otherwise be removed.

`0'

Pad the field with zeros instead of spaces; the zeros are placed after any sign. This flag is ignored if the `-' flag is also specified.

Other Output Conversions

อธิบายถึงส่วนต่างๆ ที่ใช้ใน printf

`%c' เป็นลักษณะเดี่ยว

```
printf ("%c%c%c%c%c", "h", "e", "l", "l", "o");
```

`%s' string **(**ตัวอักษร**)**

```
printf ("%3s%-6s", "no", "where");
```

Formatted Input

Octave จะแบ่งเป็น scanf, fscanf และ sscanf

Built-in Function: **[***val***,** *count***] = fscanf** ***(****fid, template, size)***

Built-in Function: **[***v1***,** *v2***, ...,** *count***] = fscanf** ***(****fid, template, "C")***

จำนวนข้อมูลที่อ่านขึ้นอยู่กับ Inf

Inf

Read as much as possible, returning a column vector.

nr

Read up to *nr* elements, returning a column vector.

[*nr*, Inf]

Read as much as possible, returning a matrix with *nr* rows. If the number of elements read is not an exact multiple of *nr*, the last column is padded with zeros.

[*nr*, *nc*]

Read up to *nr* * *nc* elements, returning a matrix with *nr* rows. If the number of elements read is not an exact multiple of *nr*, the last column is padded with zeros.

Built-in Function: [*val***,** *count***] = sscanf** ***(****string***, *template*, *size)***

Built-in Function: [*v1***,** *v2***, ...,** *count***] = sscanf** ***(****string***, *template*, *"C")***

Input Conversion Syntax

scanf รูปแบบของ โดยปกติแล้วจะมีรูปแบบที่กว้าง และทำงานได้หลากหลาย โดยจะเริ่มใช้พร้อมกับ `%' รูปแบบทั่วไปของ scanf คือ

% *flags width type conversion*

String Input Conversions

การแปลงสิ่งที่นำเข้าข้อความ ส่วนนี้อ้างถึง scanf การแปลงสิ่งที่นำเข้าสำหรับการอ่านข้อความและค่าตัวอักษร %s และ %c

%c มันจับคู่ที่กำหนดจำนวนของตัวอักษรง่ายที่สุดเสมอ ค่ามากสุดจะขึ้นอยู่กับลักษณะของการอ่าน ถ้าคุณไม่เจาะจงค่าสูงสุด default จะเท่ากับ 1

%s การแปลงจับคู่ข้อความของ nonwhitespace ตัวอักษรมันข้ามและ discards อักษรย่อ whitespace หลังจากการได้อ่านบางสิ่งสำหรับ ตัวอย่าง

hello, world

สำหรับการแปลง `%10c' กลายเป็น " hello, wo" แต่การอ่านสิ่งที่นำเข้าที่เหมือนกับการแปลง `%10s' กลายเป็น "hello"

Binary I/O

Oct e สามารถอ่านและเขียนข้อมูลไบนารี่ ใช้หน้าที่ fread และ fwrite ซึ่งจะทำงานเหมือนกับฟังก์ชันในภาษา av C สามารถที่จะทำงานอย่างอัตโนมัติแลกคำสั่งของข้อมูลจำนวนเต็มและแปลง ths ที่สนับสนุน floating รูปแบบที่เหมือนข้อมูลที่อ่าน

Built-in Function: **[*val*, *count*] = fread (*fid*, *size*, *precision*, *skip*, *arch*)**

Read binary data of type *precision* from the specified file ID *fid*.

The optional argument *size* specifies the amount of data to read and may be one of

Inf

Read as much as possible, returning a column vector.

nr

Read up to *nr* elements, returning a column vector.

[*nr*, Inf]

Read as much as possible, returning a matrix with *nr* rows. If the number of elements read is not an exact multiple of *nr*, the last column is padded with zeros.

[*nr*, *nc*]

Read up to *nr* * *nc* elements, returning a matrix with *nr* rows. If the number of elements read is not an exact multiple of *nr*, the last column is padded with zeros.

If *size* is omitted, a value of Inf is assumed.

The optional argument *precision* is a string specifying the type of data to read and may be one of

"char"

"char*1"

"integer*1"

"int8"

Single character.

"signed char"

"schar"

Signed character.

"unsigned char"

"uchar"

"uint8"

Unsigned character.

"short"

Short integer.

"unsigned short"

"ushort"

Unsigned short integer.

"int"

Integer.

"unsigned int"

"uint"

Unsigned integer.

"long"

Long integer.

"unsigned long"

"ulong"

Unsigned long integer.

"float"

"float32"

"real*4"

Single precision float.

"double"

"float64"

"real*8"

Double precision float.

"integer*2"

"int16"

Two byte signed integer.

"integer*4"

"int32"

Four byte signed integer.

"uint16"

Two byte unsigned integer.

"uint32"

Four byte unsigned integer.

The default precision is "uchar".

The optional argument *skip* specifies the number of bytes to skip after each element is read. If it is not specified, a value of 0 is assumed.

The optional argument *arch* is a string specifying the data format for the file. Valid values are

"native"

The format of the current machine.

"ieee-be"

IEEE big endian.

"ieee-le"

IEEE little endian.

"vaxd"

VAX D floating format.

"vaxg"

VAX G floating format.

"cray"

Cray floating format.

Conversions are currently only supported for "ieee-be" and "ieee-le" formats.

The data read from the file is returned in *val*, and the number of values read is returned in count

Built-in Function: *count* **= fwrite (***fid***,** *data***,** *precision***,** *skip***,** *arch***)**

การเขียนข้อมูลที่อยู่ในแบบฟอร์มไบนารี่ ของ ความเที่ยงตรงพิมพ์ที่เจาะจงแฟ้ม ID fid การคืนกลับของค่าจำนวนมาก successfully เขียนที่แฟ้ม

ข้อมูลการอภิปรายคือแม่แบบ ของ ค่าที่เขียนที่แฟ้มค่าที่ถูกขยายอยู่ในคำสั่งคอลัมน์ใหญ่

precision, skip, and arch คือข้อกำหนดเพิ่มเติม และถูกอธิบายเหมือนอ้างถึงสำหรับ fread พฤติกรรมของ fwrite ที่ไม่ถูกกำหนด ถ้าค่าอยู่ในข้อมูลใหญ่เกินไป จะทำให้ความเที่ยงตรงเปลี่ยน

End-of-file and Error

Built-in Function: feof ***(fid)***

ถ้าเงื่อนไขเป็นแบบ end-of-file จะคืนค่า 1 นอกจากนั้นจะเป็น 0

Built-in Function: ferror ***(fid)***

ถ้าเงื่อนไข error จะคืนค่า 1 นอกจากนั้นจะเป็น 0

Built-in Function: freport ***()***

ถ้ารายชื่อแฟ้มของตัวไหนเปิด และไม่ว่าจะเปิดสำหรับการอ่าน การเขียน หรือทั้งคู่ เช่น

freport ()

-| number mode name
-|
-| 0 r stdin
-| 1 w stdout
-| 2 w stderr
-| 3 r myfile

File Positioning

3 ฟังก์ชันที่เป็นตัวแปรสำหรับการตั้งค่า และกำหนด ตำแหน่ง ของตัวชี้แฟ้มสำหรับแฟ้มที่ให้

Built-in Function: ftell *(fid)*

การคืนกลับตำแหน่งของ ตัวชี้แฟ้มนับแต่ตัวอักษรจำนวนมาก การเริ่มต้นของแฟ้ม fid

Built-in Function: fseek *(fid, offset, origin)*

การวางตัวแฟ้มที่ตั้งใดๆ ภายในแฟ้ม fid ตัวชี้ถูกตำแหน่ง offset ซึ่งมาจากจุดกำเนิด เป็นไปได้ที่ตัวแปรอาจถูกกำหนดมาก่อน SEEK_CUR (current position), SEEK_SET (beginning), or SEEK_END (end of file)

Built-in Variable: SEEK_SET

Built-in Variable: SEEK_CUR

Built-in Variable: SEEK_END

ตัวแปรเหล่านี้อาจจะถูกใช้เหมือนข้อกำหนดเพิ่มเติมการอภิปรายทั้ง 3 อย่าง สำหรับหน้าที่ของ fseek

Plotting

ทั้งหมดของ Octave plotting หน้าที่ใช้ gnuplot เพื่อที่จะควบคุมกราฟฟิคตามความเป็นจริง มีระดับต่ำ 2 ระดับทำงาน gplot และ gsplot นั้น เกือบจะอย่างแน่นอนเหมือนกับ ฟังก์ชัน gnuplot plot และ splot จำนวนของระดับที่สูงกว่า ฟังก์ชัน plotting อื่นๆ กราฟฟิคที่ค้นพบใน matlab ระดับสูงกว่าเหล่านี้ทำงานถูกเพิ่มทั้งหมดอยู่ในข้อตกลง ของระดับต่ำ 2 ระดับ ของฟังก์ชันplotting

Generate a 2-dimensional plot

The ranges, using, title, and style คือข้อกำหนดเพิ่มเติมและ using, title, and style ปรากฏอยู่ในคำสั่งใดๆ หลังจากเครื่องแสดง เมื่อคุณต้องการ plot หลายๆ ครั้งโดยใช้คำสั่งเดียว เราจะใช้ ranges

[x_lo : x_up] [y_lo : y_up]

Function File: plot *(args)*

This function produces two-dimensional plots. Many different combinations of arguments are possible. The simplest form is

plot (*y*)

where the argument is taken as the set of *y* coordinates and the *x* coordinates are taken to be the indices of the elements, starting with 1.

If more than one argument is given, they are interpreted as

plot (*x*, *y*, *fmt* ...)

where *y* and *fmt* are optional, and any number of argument sets may appear. The *x* and *y* values are interpreted as follows:

If a single data argument is supplied, it is taken as the set of *y* coordinates and the *x* coordinates are taken to be the indices of the elements, starting with 1.

If the first argument is a vector and the second is a matrix, the the vector is plotted versus the columns (or rows) of the matrix. (using whichever combination matches, with columns tried first.)

If the first argument is a matrix and the second is a vector, the the columns (or rows) of the matrix are plotted versus the vector. (using whichever combination matches, with columns tried first.)

If both arguments are vectors, the elements of *y* are plotted versus the elements of *x*.

If both arguments are matrices, the columns of *y* are plotted versus the columns of *x*. In this case, both matrices must have the same number of rows and columns and no attempt is made to transpose the arguments to make the number of rows match.

If both arguments are scalars, a single point is plotted.

If the *fmt* argument is supplied, it is interpreted as follows. If *fmt* is missing, the default gnuplot line style is assumed.

`-'

Set lines plot style (default).

`.'

Set dots plot style.

`@'

Set points plot style.

`-@'

Set linespoints plot style.

`^'

Set impulses plot style.

`L'

Set steps plot style.

`*n*'

Interpreted as the plot color if *n* is an integer in the range 1 to 6.

`*nm*'

If *nm* is a two digit integer and *m* is an integer in the range 1 to 6, *m* is interpreted as the point style. This is only valid in combination with the @ or -@ specifiers.

`*c*'

If *c* is one of "r", "g", "b", "m", "c", or "w", it is interpreted as the plot color (red, green, blue, magenta, cyan, or white).

`"';title;"'

Here "title" is the label for the key.

`+'

`*'

`o'

`x'

Used in combination with the points or linespoints styles, set the point style.

The color line styles have the following meanings on terminals that support color.

Number	Gnuplot colors	(lines)points style
1	red	*
2	green	+
3	blue	o
4	magenta	x
5	cyan	house
6	brown	there exists

The *fmt* argument can also be used to assign key titles. To do so, include the desired title between semi-colons after the formatting sequence described above, e.g. "+3;Key Title;" Note that the last semi-colon is required and will generate an error if it is left out.

Here are some plot examples:

```
plot (x, y, "@12", x, y2, x, y3, "4", x, y4, "+")
```

This command will plot y with points of type 2 (displayed as `+') and color 1 (red), y2 with lines, y3 with lines of color 4 (magenta) and y4 with points displayed as `+'.

```
plot (b, "*")
```

This command will plot the data in the variable b will be plotted with points displayed as `*'.

```
t = 0:0.1:6.3;
plot (t, cos(t), "-;cos(t);", t, sin(t), "+3;sin(t);");
```

This will plot the cosine and sine functions and label them accordingly in the key.

Function File: axis *(limits)*

กำหนดแกนสำหรับ plots

เลือก vector มา 2 ,4 หรือ 6 vector ครั้งที่ 1 และ 2 จะเจาะจงที่ขีดจำกัดบนและล่างของแกน x ครั้งที่ 3 , 4 สำหรับแกน y ครั้งที่ 5 ,6 สำหรับแกน z ถ้าคุณเริ่ม plot บางครั้งคุณต้องการ replot ก่อนจึงจะได้ผล

คุณสามารถได้สิ่งนี้ที่จะเกิดขึ้นโดยอัตโนมัติโดยการตั้งค่าตัวแปรที่อยู่ภายใน automatic_replot ที่ค่าไม่เท่ากับ 0

```
axis ([1, 2, 3, 4], "square");
```

forces a square aspect ratio, and

```
axis ("labely", "tic");
```

turns tic marks on for all axes and tic mark labels on for the y-axis only.

The following options control the aspect ratio of the axes.

"square"

Force a square aspect ratio.

"equal"

Force x distance to equal y-distance.

"normal"

Restore the balance.

The following options control the way axis limits are interpreted.

"auto"

Set the specified axes to have nice limits around the data or all if no axes are specified.

"manual"

Fix the current axes limits.

"tight"

Fix axes to the limits of the data (not implemented).

The option "image" is equivalent to "tight" and "equal".

The following options affect the appearance of tic marks.

"on"

Turn tic marks and labels on for all axes.

"off"

Turn tic marks off for all axes.

"tic[xyz]"

Turn tic marks on for all axes, or turn them on for the specified axes and off for the remainder.

"label[xyz]"

Turn tic labels on for all axes, or turn them on for the specified axes and off for the remainder.

"nolabel"

Turn tic labels off for all axes.

Note, if there are no tic marks for an axis, there can be no labels.

The following options affect the direction of increasing values on the axes.

"ij"

Reverse y-axis, so lower values are nearer the top.

"xy"

Restore y-axis, so higher values are nearer the top.

Specialized Two-Dimention Plots

Function File: contour *(z, n)*

Function File: contour *(x, y, z, n)*

Make a contour plot of the three-dimensional surface described by *z*. Someone needs to improve gnuplot's contour routines before this will be very useful.

Function File: loglog ***(****args****)***

Make a two-dimensional plot using log scales for both axes. See the description of plot for a description of the arguments that loglog will accept.

Function File: polar ***(****theta****,*** *rho****,*** *fmt****)***

Make a two-dimensional plot given polar the coordinates *theta* and *rho*.

The optional third argument specifies the line type

Function File: semilogx ***(****args****)***

Make a two-dimensional plot using a log scale for the *x* axis. See the description of plot for a description of the arguments that semilogx will accept.

Function File: semilogy ***(****args****)***

Make a two-dimensional plot using a log scale for the *y* axis. See the description of plot for a description of the arguments that semilogy will accept.

Plot Annotations

Function File: grid ***(****arg****)***

For two-dimensional plotting, force the display of a grid on the plot. The argument may be either "on" or "off". If it is omitted, "on" is assumed.

Function File: title ***(****string****)***

Specify a title for a plot. If you already have a plot displayed, use the command replot to redisplay it with the new title.

Function File: bottom_title ***(****string****)***

See top_title.

Function File: xlabel ***(****string****)***

Function File: ylabel ***(****string****)***

Function File: zlabel ***(****string****)***

Specify x, y, and z axis labels for the plot. If you already have a plot displayed, use the command replot to redisplay it with the new labels.

Multiple Plots on One Page

ฟังก์ชันทั้งหมดต้องการ gnuplot ใน เวอร์ชัน ที่สนับสนุน ลักษณะที่ plot ได้หลายค่า

Function File: mplot *(x, y)*

Function File: mplot *(x, y, fmt)*

Function File: oneplot *()*

If in multiplot mode, switches to single plot mode.

Function File: plot_border *(...)*

Multiple arguments allowed to specify the sides on which the border is shown. Allowed arguments include:

"blank"

No borders displayed.

"all"

All borders displayed

"north"

North Border

"south"

South Border

"east"

East Border

"west"

West Border

Function File: subplot *(rows, cols, index)*

Function File: subplot *(rcn)*

กำหนดค่า gnuplot ใน multiplotโหมดและ plots อยู่ในที่ตั้งให้โดยดัชนีตัวแปรครอบคลุม ทั่ว_multiplot_scale_จะเป็น vector 2ค่า ครั้งแรกให้เท่ากับ xsize และครั้งที่ 2 เท่ากับ ysize

Input:

rows

Number of rows in subplot grid.

columns

Number of columns in subplot grid.

index

Index of subplot where to make the next plot.

มีเหตุผลเดียวที่สนับสนุน เมื่อมันมีค่าของตัวเลขเป็น 3 หลัก โดย 1 เป็น แถว 2 เป็นหลัก และ 3 เป็นค่าในการ plot เช่นตัวอย่าง จะ plot แบบ 4 และ 2 กริด ๆได้ดังนี้

```
+-----+-----+-----+-----+
| 1 | 2 | 3 | 4 |
+-----+-----+-----+-----+
| 5 | 6 | 7 | 8 |
+-----+-----+-----+-----+
```

Function File: **top_title** ***(string)***

Function File: **bottom_title** ***(string)***

Makes a title with text *string* at the top (bottom) of the plot.

Interaction with gnuplot

Built-in Variable: gnuplot_binary

ชื่อโปรแกรม ทำรายการโดยทำสั่ง plot ค่าที่ผิดพลาดคือ gnuplot

Built-in Variable: gnuplot_has_frames

ถ้าค่าของตัวแปรไม่เท่ากับ 0 Octave จะสมมติคุณก๊อปปี้ gnuplot ได้สนับสนุนสำหรับหลายเฟรมที่ถูกรวมถึงอยู่ในรุ่นที่ 3.6 beta

เมื่อเร็วๆ นี้ มันคือค่าแรก determined โดยแก้ไขแต่มันสามารถเป็นเปลี่ยนอยู่ในสคริปต์การเริ่มต้นของคุนหรือ คำสั่งในกรณีที่แก้ไข หรือถ้าคุณเปลี่ยน gnuplot ในการติดตั้ง

Built-in Function: graw ***(string)***

Send *string* directly to gnuplot subprocess.

Built-in Variable: gnuplot_command_plot

Built-in Variable: gnuplot_command_replot

Built-in Variable: gnuplot_command_splot

Built-in Variable: gnuplot_command_using

Built-in Variable: gnuplot_command_with

Built-in Variable: gnuplot_command_axes

Built-in Variable: gnuplot_command_title

Built-in Variable: gnuplot_command_end

การคืนค่าปัจจุบันของ seed

Loadable Function: randn *(x)*

Loadable Function: randn *(n, m)*

Loadable Function: randn *("seed", x)*

Matrixจะกลับคืนโดยการสุ่มแบบปกติ ค่าจะถูกจัดการเหมือนกับค่าที่เห็น ในตอนนี้คุณสามารถจัด seed สำหรับจำนวนโดยการสุมตัวเลขโดยใช้รูปแบบ

randn ("seed", *x*)

เมื่อ x เป็นค่าตัวเ**ลขเมื่อใช้**

randn ("seed")

randn จะคืนค่าปัจจุบันของ seed

rand และ randnเป็นการแบ่งแยกโดย

```
rand ("seed", 13);
randn ("seed", 13);
u = rand (100, 1);
n = randn (100, 1);
```

and

```
rand ("seed", 13);
randn ("seed", 13);
```

```
u = zeros (100, 1);
n = zeros (100, 1);
for i = 1:100
  u(i) = rand ();
  n(i) = randn ();
end
```

ผลลัพธ์จะเท่ากัน

ตามปกติ rand และ randn จะรับค่า seed จากระบบนาฬิกา เพื่อที่ผลที่ตามกันมาของจำนวนโดยการสุ่มไม่เป็นของเหมือนกันแต่ละเวลาซึ่งคุณ run octave แล้ว ถ้าคุณความจำเป็นสำหรับเพื่อจำลองผลที่ตามกันมาของจำนวนพอดี คุณสามารถจัดseedต่อค่าซึ่งเจาะจง.

ถ้ามันถูกเรียกไม่มี argument, rand และrandn ปัจจัยสำคัญเดี่ยวการกลับคืนของผลที่ตามกันมาโดยการสุ่ม.

Function File: randperm ***(n)***

เวกเตอร์แถวการกลับคืนการเปลี่ยนลำดับโดยการสุ่มของจำนวนเต็มตั้งแต่ 1 ถึงn .

Built-in Function: diag ***(v, k)***

แม่แบบที่ทแยงมุมกับเวกเตอร์v บนแถวทแยงk .argumentที่สองในการเลือก. ถ้ามันเป็นประโยชน์ ,เวกเตอร์ถูกวางบนk-th super-diagonal . ถ้ามันที่เป็นลบ,มันถูกวางบน-k-th sub-diagonal . ค่าการผิดสัญญาเกี่ยวกับk เป็น 0 ,และเวกเตอร์ถูกวางบนแถวทแยงซึ่งสำคัญ. สำหรับตัวอย่าง,.

```
diag ([1, 2, 3], 1)
   => 0 1 0 0
      0 0 2 0
      0 0 0 3
      0 0 0 0
```

หน้าที่linspace และlogspace ทำมันอย่างง่ายมากเพื่อสร้างเวกเตอร์ด้วยอย่างราบเรียบหรือ logarithmical .

Built-in Function: linspace ***(base, limit, n)***

เวกเตอร์แถวการกลับคืนกับn linearly เว้นช่องปัจจัยสำคัญระหว่างฐานและขอบเขต. จำนวนปัจจัยสำคัญ,n ,ต้องใหญ่กว่า 1 ฐานและขอบเขตรวมอยู่แล้วในขอบเขตเสมอ. ถ้าฐานยิ่งใหญ่กว่าขอบเขต ,ปัจจัยสำคัญถูกเก็บในการลดคำสั่ง. . ถ้าจำนวนจุดไม่ถูกระบุ,ค่าของ 100 ถูกใช้.
linspace หน้าที่เวกเตอร์แถวการกลับคืนเสมอ.

Function File: logspace ***(base, limit, n)***

คล้ายคลึงกันต่อlinspace ยกเว้นซึ่งค่าเป็นlogarithmically เว้นช่องตั้งแต่10base ถึง10limit . ถ้าขอบเขตเท่ากับpi ,จุดอยู่ระหว่าง10base และpi ,ไม่10base และ10pi .เพื่อจะเป็นเข้ากันได้กับ MATLABซึ่งตรงกันหน้าที่.

Built-in Variable: warn_neg_dim_as_zero

ถ้าค่าเกี่ยวกับwarn_neg_dim_as_zero เป็นคำเตือน nonzero ,print สำหรับสิ่งที่เหมือนกันการแสดงออก.

eye (-1)

ค่าเป็น 0 .

Built-in Variable: warn_imag_to_real

ถ้าค่าเกี่ยวกับwarn_imag_to_real เป็นnonzero . คำเตือนถูกพิมพ์สำหรับการเปลี่ยนแปลงซึ่งให้เห็นของจำนวนซับซ้อนต่อจำนวนจริง เมื่อค่าเป็น 0

Arithmetic

1. Utility Functions

หน้าที่ซึ่งตามมาเป็นพร้อมสำหรับการทำงานกับจำนวนเชิงซ้อน. แต่จะคาดว่าเป็นargumentเดี่ยว . คำสั่ง ประยุกต์หน้าที่ซึ่งมอบให้ไว้ต่อแต่ละปัจจัยสำคัญของmatrix

Mapping Function: ceil *(x)*

คืนจำนวนเต็มเล็กสุดไม่น้อยกว่าx. ถ้าเอ็กซ์ซับซ้อน. return ceil (real (*x*)) + ceil (imag (*x*)) * I.

Mapping Function: exp *(x)*

คำนวณexponential ของx. เพื่อคำนวณmatrixexponential .

Mapping Function: fix *(x)*

ให้เอ็กซ์ไปยังศูนย์. ถ้าเอ็กซ์ซับซ้อน return fix (real (*x*)) + fix (imag (*x*)) * I.

. Mapping Function: floor *(x)*

คืนจำนวนเต็มใหญ่สุดไม่ยิ่งใหญ่กว่าx. ถ้าเอ็กซ์เป็นจำนวนเชิงซ้อน., return floor (real (*x*)) + floor (imag (*x*)) * I.

Mapping Function: gcd *(x, ...)*

คำนวณตัวหารสามัญซึ่งใหญ่สุดของปัจจัยสำคัญของเอ็กซ์,หรือรายการของทั้งหมดargument. เช่น ตัวอย่าง

gcd (a1, ..., ak)

เหมือนกับ

gcd ([a1, ..., ak])

ค่ากลับที่สองซึ่งสิทธิในการเลือก,v บรรจุนั่นสิ่งนั้นเวกเตอร์จำนวนเต็ม.

..

g = v(1) * a(k) + ... + v(k) * a(k)

Mapping Function: lcm ***(x, ...)***

คำนวณผลคูณสามัญซึ่งส่วนใหญ่ของปัจจัยสำคัญปัจจัยสำคัญของเอ็กซ์,หรือรายการของทั้งหมด argument. เช่นตัวอย่าง

lcm (a1, ..., ak)

Mapping Function: log ***(x)***

คำนวณเลขลอกอริทึมโดยธรรมชาติสำหรับแต่ละปัจจัยสำคัญของx. เพื่อคำนวณmatrixเลขลอกอริทึม,.

Mapping Function: log10 ***(x)***

คำนวณbase-10 เลขลอกอริทึมสำหรับแต่ละปัจจัยสำคัญของเอ็กซ์.

Mapping Function: log2 ***(x)***

Mapping Function: [*f*, *e*] log2 ***(x)***

คำนวณbase-2 เลขลอกอริทึมเกี่ยวกับx. กับoutput returns *f* and *e* such that 1/2 <= abs(f) < 1 and x = f * 2^e.

Mapping Function: max ***(x, y)***

Mapping Function: [*w*, *iw*] = max ***(x)***

สำหรับเวกเตอร์argument,คืนค่าสูงที่สุด. สำหรับmatrix argument. คืนค่าสูงที่สุดจากแต่ละหลัก เช่นเวกเตอร์แถว. สำหรับสองmatrices (หรือmatrixและscalar),คืนpair-wise ค่าสูงสุด. ดังนั้น,.

max (max (*x*))

คืนปัจจัยสำคัญใหญ่สุดของเอ็กซ์,และ.

max (2:5, pi)

=> 3.1416 3.1416 4.0000 5.0000

เปรียบเทียบแต่ละปัจจัยสำคัญของขอบเขต 2 : 5 กับpi ,และเวกเตอร์แถวการกลับคืนของค่าสูงที่สุด.

สำหรับจำนวนเชิงซ้อน argument,ขนาดของปัจจัยสำคัญถูกใช้สำหรับการเปรียบเทียบ.
ถ้าเรียกกับอินพุทหนึ่งและการโต้เถียงสองสิ่งที่ออกมา,max คืนดัชนีแรกของค่าสูงสุดv ด้วย.

[x, ix] = max ([1, 3, 5, 2, 5])

=> x = 5

ix = 3

Function File: nextpow2 ***(x)***

ถ้าเอ็กซ์เป็นscalar ,คืนจำนวนเต็มแรกn นั่นสิ่งนั้น. 2^n >= abs (x).

ถ้า x เป็น ค่าตัวเลข, return nextpow2 (length (*x*)).

Mapping Function: pow2 ***(x)***

Mapping Function: pow2 ***(f, e)***

ด้วยargumentหนึ่ง,คำนวณ 2 . เอ็กซ์ สำหรับแต่ละปัจจัยสำคัญเกี่ยวกับx. กับสองargument. **returns f .* (2 .^ e).**

Mapping Function: rem ***(x, y)***

คืนสิ่งที่เหลืออยู่ของ x/ y ,คำนวณกำลังใช้การแสดงออก.

x - y .* fix (x ./ y)

ข้อความผิดถูกพิมพ์ถ้ามิติของargumentไม่เห็นด้วย,หรือถ้าเพียงของargumentเป็นจำนวนเชิงซ้อน.

Mapping Function: round ***(x)***

คืนจำนวนเต็มใกล้ต่อx. ถ้าเอ็กซ์เป็นจำนวนเชิงซ้อน. return round (real (*x*)) + round (imag (*x*)) * I.

Mapping Function: sign ***(x)***

คำนวณsignum

$$\text{sign}(x) = \begin{cases} -1, & x < 0; \\ 0, & x = 0; \\ 1, & x > 0. \end{cases}$$

Mapping Function: sqrt ***(x)***

คำนวณรากสองของx. ถ้าเอ็กซ์ที่เป็นลบ,จำนวนเชิงซ้อนผลลัพท์ถูกคืน. การคำนวณmatrixรากที่สอง.

2 . Complex Arithmetic

หน้าที่ซึ่งตามมาเป็นพร้อมสำหรับการทำงานกับจำนวนเชิงซ้อนจำนวน. แต่ละค่าเดี่ยว

Mapping Function: abs ***(z)***

ใช้หาขนาดโดย , defined as |*z*| = sqrt (x^2 + y^2). เช่น

abs (3 + 4i)

=> 5

Mapping Function: arg *(z)*

Mapping Function: angle *(z)*

ใช้หามุมโดย as *theta* = atan (*y*/*x*). เช่น

arg (3 + 4i)

=> 0.92730

Mapping Function: conj *(z)*

ใช้หาconjugate conj (*z*) = *x* - *iy*.

Mapping Function: imag *(z)*

ใช้หาจำนวนจินตภาพเมื่อ z เท่ากับจำนวนจริง

Mapping Function: real *(z)*

ใช้หาค่าจริงของ z

3. Trigonometry

Mapping Function: sin *(x)*

ใช้หาค่า sine x

Mapping Function: cos *(x)*

ใช้หาค่า cosine *x*.

Mapping Function: tan *(z)*

ใช้หาค่า tanget *x*.

Mapping Function: sec *(x)*

ใช้หาค่า secant *x*.

Mapping Function: csc *(x)*

ใช้หาค่า cosecant *x*.

Mapping Function: cot *(x)*

ใช้หาค่า cotangent *x*.

Mapping Function: asin *(x)*

ใช้หามุม sine *x*.

Mapping Function: acos *(x)*

ใช้หามุม cosine *x*.

Mapping Function: atan *(x)*

ใช้หามุม tangent *x*.

Mapping Function: asec *(x)*

ใช้หามุม secant *x*.

Mapping Function: acsc *(x)*

ใช้หามุม cosecant *x*.

Mapping Function: acot *(x)*

ใช้หามุม cotangent *x*.

Mapping Function: sinh *(x)*

ใช้หา hyperbolic sin *x*.

Mapping Function: cosh *(x)*

ใช้หา hyperbolic cosine *x*.

Mapping Function: tanh *(x)*

ใช้หา hyperbolic tangent *x*.

Mapping Function: sech *(x)*

ใช้หา hyperbolic secant *x*.

Mapping Function: csch *(x)*

ใช้หา hyperbolic cosecant *x*.

Mapping Function: coth *(x)*

ใช้หา hyperbolic cotangent *x*.

Mapping Function: asinh *(x)*

ใช้หามุม hyperbolic sine *x*.

Mapping Function: acosh *(x)*

ใช้หามุม hyperbolic cosine *x*.

Mapping Function: atanh *(x)*

ใช้หามุม hyperbolic tanget *x*.

Mapping Function: asech *(x)*

ใช้หามุม hyperbolic secant *x*.

Mapping Function: acsch *(x)*

ใช้หามุม hyperbolic cosecant *x*.

Mapping Function: acoth (*x*)

ใช้หามุม hyperbolic cotangent *x*.

19.5 Special Functions

Mapping Function: bincoeff *(n, k)*

ใช้หาค่า binomial coefficient of *n* and *k*, โดย

```
/   \
| n |    n (n-1) (n-2) ... (n-k+1)
|   |  = -------------------------
| k |              k!
\   /
```

Mapping Function: erf *(z)*

ใช้หาค่า error function

```
                    z
                    /
erf (z) = (2/sqrt (pi)) | e^(-t^2) dt
                    /
                  t=0
```

Function File: cross *(x, y)*

ใช้หา cross product ของ vector

cross ([1,1,0], [0,1,1])

=> [1; -1; 1]

Coordinate Transformations

Function File: [*theta*, *r*] = cart2pol (*x*, *y*)

Function File: [*theta*, *r*, *z*] = cart2pol (*x*, *y*, *z*)

ใช้หาความสัมพันธ์ระหว่าง พิกัดทรงกระบอกกับแกน x,y

Function File: [*x*, *y*] = **pol2cart (*theta*, *r*)**

Function File: [*x*, *y*, *z*] = **pol2cart (*theta*, *r*, *z*)**

ใช้หาความสัมพันธ์ระหว่างแกน x,yกับ พิกัดทรงกระบอก

Function File: [*theta*, *phi*, *r*] = **cart2sph (*x*, *y*, *z*)**

ใช้หาความสัมพันธ์ระหว่าง พิกัดทรงกระบอกกับพิกัดทรงกลม

Function File: [*x*, *y*, *z*] = **sph2cart (*theta*, *phi*, *r*)**

ใช้หาความสัมพันธ์ระหว่างพิกัดทรงกลมกับพิกัดทรงกระบอก

Mathematical Constants

Built-in Variable: I

Built-in Variable: J

Built-in Variable: i

Built-in Variable: j

เป็นค่ารากที่สองของ -1

Built-in Variable: pi

อัตราส่วนของเส้นรอบวงของวงกลมต่อเส้นผ่าศูนย์กลางของมัน. ภายใน,pi ถูกคำนวณโดย `4.0 * atan (1.0)'.

Built-in Variable: e

ค่า log ฐานธรรมชาติ log(e) = 1

Linear Algebra

1. Basic Matrix Functions

Loadable Function: *aa* = balance *(a, opt)*

Loadable Function: [*dd*, *aa*] = balance *(a, opt)*

Loadable Function: [*cc*, *dd*, *aa*, *bb*] = balance *(a, b, opt)*

[dd, aa] = balance (a) returns aa = dd \ a * dd. aaเป็นmatrixของแถวและหลักเท่ากันอย่างในขนาด. และ dd = p * d, การเปลี่ยนลำดับmatrixและd เป็นmatrixที่ทแยงมุมของกำลังสอง นี่อนุญาติ equilibration เพื่อถูกคำนวณไม่มีroundoff . ผลลัพท์เกี่ยวกับeigenvalue การคำนวณถูกทำให้ดีขึ้น ใกล้สมดุลอันดับแรกอย่างเป็นแบบอย่าง .

[cc, dd, aa, bb] = balance (a, b) returns aa = cc*a*dd and bb = cc*b*dd)
ที่aa และbb มีnon-zero ปัจจัยสำคัญของอย่างใกล้เคียงขนาดซึ่งเหมือนกัน. และcc และdd ถูก เรียงลำดับแถวทแยงmatrices เช่นในdd สำหรับเกี่ยวกับพีชคณิตeigenvalue ปัญหา.

igenvalue กำลังทำให้ทางเลือกสมดุลopt ถูกเลือกอย่างที่ติดตาม:

"N", "n"

ไม่มีกำลังสมดุล;argumentคัดลอก,การแปรสภาพ(s)เริ่มต้นเอกลักษณ์.

.. "P", "p"

เรียงลำดับargument(s)เพื่อแยกออกeigenvalues ที่ไหนเป็นไปได้.

"S", "s"

สเกลเพื่อทำให้ความแม่นยำดีขึ้นของคำนวณแล้วeigenvalues .

"B", "b"

เรียงลำดับและสเกล,ในซึ่งที่ออกคำสั่ง. Rows/columns ของ(และb)ซึ่งถูกแยกออกโดยการเปลี่ยนลำดับไม่มี.

นี่เป็นความประพฤติการผิดสัญญา.

เกี่ยวกับพีชคณิตeigenvalue ทำให้พวกเราสมดุลLAPACKมาตรฐานงานประจำ.

Function File: cond *(a)*

คำนวณ(two-norm)กำหนดเงื่อนไขจำนวนmatrix. cond ()เป็นnorm (a) * norm (inv (a)),

Loadable Function: [*d*, *rcond*] = det *(a)*

ใช้คำนวนหา det

Function File: dot *(x, y)*

ใช้คำนวนหา dot product ของ2เวคเตอร์

Loadable Function: [*x*, *rcond*] = inv *(a)*

Loadable Function: [*x*, *rcond*] = inverse *(a)*

ใช้หา inverse ของmatrix 2 matrix

2. Matrix Factorizations

Loadable Function: chol *(a)*

คำนวณCholesky factor,r ,ของsymmetric matrixซึ่งแน่นอนซึ่งเป็นประโยชน์

r' * r = a.

Loadable Function: *h* = hess *(a)*

Loadable Function: [*p*, *h*] = hess *(a)*

คำนวณHessenberg ของmatrix.

3. Functions of a Matrix

Loadable Function: expm *(a)*

exponential ของmatrix,นิยามว่าคืออนุกรมอนันต์ของ taylor

expm(a) = I + a + a^2/2! + a^3/3! + ...

Function File: logm ***(a)***

คำนวณmatrixเลขลอกอริทึมของmatrixจัตุรัส.

1. Nonlinear Equations

octave สามารถแก้โจทย์ของสมการไม่เป็นเส้นตรงได้

F (x) = 0

ใช้ฟังก์ชันfsolve ,แก้โดยอาศัยพื้นฐานMINPACKsubroutine hybrd .

Loadable Function: [*x*, *info*, *msg*] = fsolve ***(fcn, x0)***

ให้fcn ,ชื่อของฟังก์ชันของรูปแบบf (เอ็กซ์)และจุดการเริ่มแรกเริ่มx0 .

fsolve แก้ไขการจัดตั้งของนั่นสิ่งนั้นสมการf (เอ็กซ์)== 0 .

ถ้าfcn เป็นtwo-element ขบวนเชือก,ปัจจัยสำคัญแรกตั้งชื่อฟังก์ชัน. และชื่อปัจจัยสำคัญที่สอง ฟังก์ชันของรูปแบบj (เอ็กซ์)เพื่อคำนวณJacobian matrixด้วยelement.

$$jac(i,j) = \frac{df\_i}{dx\_j}$$

คุณสามารถใช้ฟังก์ชันfsolve_options เพื่อจัดปัจจัยกำหนดซึ่งสิทธิในการเลือกสำหรับfsolve .

Loadable Function: fsolve_options ***(opt, val)***

เมื่อไรเรียกกับสองargument,ฟังก์ชันนี้อนุญาติปัจจัยกำหนดคุณทางเลือกกำหนดแล้วสำหรับ ฟังก์ชัน. fsolveให้argumentหนึ่ง,fsolve_options คืนค่าของทางเลือกซึ่งตรงกัน.

ทางเลือกประกอบด้วย.

"tolerance"

ที่นี่เป็นตัวอย่างสมบูรณ์. ต่อแก้ปัญหาของสมการ.

-2x^2 + 3xy + 4 sin(y) = 6

3x^2 - 2xy^2 + 3 cos(x) = -4

คุณอันดับแรกจำเป็นจะเขียนฟังก์ชันเพื่อคำนวณค่าของฟังก์ชันซึ่งมอบให้ไว้. สำหรับตัวอย่าง:

```
function y = f (x)
  y(1) = -2*x(1)^2 + 3*x(1)*x(2)   + 4*sin(x(2)) - 6;
  y(2) =  3*x(1)^2 - 2*x(1)*x(2)^2 + 3*cos(x(1)) + 4;
endfunction
```

ดังนั้น,เรียกfsolve ด้วยระบุเงื่อนไขแรกเริ่มเพื่อพบรากของระบบของสมการ.

สำหรับตัวอย่าง,ให้ฟังก์ชันf

[x, info] = fsolve ("f", [1; 2])

ผลลัพท์ในการแก้ปัญหา.

x =

0.57983

2.54621

info = 1

ค่าของข้อมูล= 1 ชี้บอกซึ่งที่การแก้ปัญหาลู่เข้า.

ฟังก์ชันperror เคยพิมพ์รหัสข่าวสารอังกฤษกำลังสอดคล้องกับnumeric ความผิด.

เช่นตัวอย่าง

perror ("fsolve", 1)

-| solution converged to requested tolerance

Differential Equations

Octaveมีสองหน้าที่รวมอยู่ด้วยสำหรับการแก้ไขสมการซึ่งแตกต่างกัน.

Ordinary Differential Equations

ฟังก์ชันlsode สามารถเคย แก้ปัญหา ODEsของรูปแบบ.

dx

-- = f (x, t)

dt

.

Loadable Function: **[*x*, *istate*, *msg*] lsode** ***(fcn, x_0, t, t_crit)***

ใช้แก้สมการ differential equation

dx

-- = f(x, t)

dt

กับ

x(t_0) = x_0

การแก้ปัญหาถูกคืนในmatrix เอ็กซ์,กับแต่ละแถวกำลังสอดคล้องกับปัจจัยสำคัญของเวกเตอร์t .
ปัจจัยสำคัญแรกเกี่ยวกับt ควรเป็นและควรสอดคล้องกับสภาพแรกเริ่มของระบบx_0 .
พื่อที่แถวแรกของสิ่งที่ออกมาเป็นx_0 .
argumentแรก,fcn ,เป็นอักษรนั่นตั้งชื่อฟังก์ชันเพื่อคำนวณเวกเตอร์
ฟังก์ชันต้องมีรูปแบบ.

xdot = f (*x*, *t*)

ในซึ่งการxdot และเอ็กซ์เป็นเวกเตอร์และt เป็นscalar .
.. ถ้าfcn เป็นtwo-element แถวอักษร,ปัจจัยสำคัญแรกตั้งชื่อฟังก์ชัน
และชื่อปัจจัยสำคัญที่สองฟังก์ชันเพื่อคำนวณJacobian เกี่ยวกับ. Jacobian ฟังก์ชันต้องมี. รูปแบบ
ดังนี้

jac = j (*x*, *t*)

ในซึ่งjac เป็นmatrixของอนุพันธ์เพียงบางส่วน.

| df_1 df_1   df_1 |

| ---- ---- ... ---- |

| dx_1 dx_2   dx_N |

| |

| df_2 df_2   df_2 |

| ---- ---- ... ---- |

df_i | dx_1 dx_2   dx_N |

jac = ---- = | |

dx_j | .   .   .   . |

| .   .   .   . |

| .   .   .   . |

| |

| df_N df_N   df_N |

| ---- ---- ... ---- |

| dx_1 dx_2   dx_N |

argumentที่สองและสามระบุintial สภาพของระบบ,,และค่าแรกเริ่มเกี่ยวกับตัวแปรอิสระ.

Loadable Function: lsode_options ***(opt, val)***

เมื่อไรเรียกกับสองargument,ฟังก์ชันนี้อนุญาติปัจจัยกำหนดคุณทางเลือกกำหนดแล้วสำหรับ ฟังก์ชันlso ห้argumentหนึ่ง,lsode_options คืนค่าของทางเลือกซึ่งตรงกัน.

ถ้าไม่มีargumentถูกจัดหา,ชื่อของทางเลือกซึ่งสามารถใช้ได้ทั้งหมดและค่าที่แสดง.

ทางเลือกประกอบด้วย.

.. "absolute tolerance"

สมบูรณ์tolerance . อาจจะเวกเตอร์เพียงหรือscalar . ถ้าเวกเตอร์,มันต้องเหมาะสมมิติของ.เวคเตอร์ตัวตั้ง

"relative tolerance"

tolerance ปัจจัยกำหนด. ไม่เหมือนสมบูรณ์tolerance ,ปัจจัยกำหนดนี้อาจจะเป็นscalar เท่านั้น.

การทดสอบความผิดพลาดของการอินทิเกรตคือ

abs (local error in x(i)) <= rtol * abs (y(i)) + atol(i)

ที่นี่เป็นตัวอย่างเกี่ยวกับการแก้ไขการจัดตั้งของสามสมการซึ่งแตกต่างกันกำลังใช้lsode . ให้ฟังก์ชัน.

```
function xdot = f (x, t)

  xdot = zeros (3,1);

  xdot(1) = 77.27 * (x(2) - x(1)*x(2) + x(1) \
          - 8.375e-06*x(1)^2);
  xdot(2) = (x(3) - x(1)*x(2) - x(2)) / 77.27;
  xdot(3) = 0.161*(x(1) - x(3));

endfunction
```

และเงื่อนไขแรกเริ่มx0 = [4 ; 1 . 1 ; 4] ,การจัดตั้งของสมการถูกทำให้รวมกันสามารถกำลังใช้คำสั่ง

```
t = linspace (0, 500, 1000);

y = lsode ("f", x0, t);
```

ถ้าคุณพยายามนี่,คุณจะพบซึ่งค่าของการเปลี่ยนแปลงผลลัพท์เกี่ยวกับความสัมพันธ์ระหว่างt . = 0 และ 5 ,และรอบt = 305 อีกครั้ง. การจัดตั้งซึ่งประสิทธิภาพเพิ่มเติมกว่าของจุดสิ่งที่ออกมาอาจจะเป็น.

```
..
t = [0, logspace (-1, log10(303), 150), \
      logspace (log10(304), log10(500), 150)];
```

Loadable Function: [*x*, *sx*, *istate*, *msg*] odessa (*fcn*, *x_0*, *p*, *sx_0*, *t*, *t_crit*)

ใช้หา set ของ differential equation

$$\frac{dx}{dt} = f(x, t; p)$$

กับ

x(t_0) = x_0

และคำนวณโดยพร้อมกันfirst-order เลขคูณความไวต่อสิ่งกระตุ้นให้ใกล้. โดย

s'(t) = j(t)*s(t) + df/dp

ซึ่ง

s(t) = dx(t)/dp (sensitivity functions)

s'(t) = d(dx(t)/dp)/dt

j(t) = df(x,t;p)/dx(t) (Jacobian matrix)

df/dp = df(x,t;p)/dp (inhomogeneity matrix)

การแก้ปัญหาถูกคืนในmatrix เอ็กซ์,กับแต่ละแถวกำลังสอดคล้องกับปัจจัยสำคัญของเวกเตอร์t .
ปัจจัยสำคัญแรกเกี่ยวกับt ควรเป็นและควรสอดคล้องกับสภาพแรกเริ่มของระบบx_0 ,.
เพื่อที่แถวแรกของสิ่งที่ออกมาเป็นx_0 .
.. ความไวต่อสิ่งกระตุ้นถูกคืนในรายการของmatrices ,sx ,.
ด้วยแต่ละปัจจัยสำคัญของรายการกำลังสอดคล้องกับปัจจัยสำคัญของเวกเตอร์t .
.. rgumentแรก,fcn ,เป็นเชือกนั่นตั้งชื่อฟังก์ชันเพื่อหาคำนวณเวกเตอร์ทางขวามือของ set สมการ
ฟังก์ชันต้องมีรูปแบบ.

xdot = f (*x*, *t*, *p*)

ในซึ่งการxdot และเอ็กซ์เป็นเวกเตอร์และt เป็นscalar .
ตัวแปรp เป็นเวกเตอร์ของปัจจัยกำหนด.
fcn argumentอาจจะเป็นแถวอักษรด้วย.

["f"; "j"; "b"]

ในซึ่งปัจจัยสำคัญแรกตั้งชื่อฟังก์ชันบรรยายเหนือ,ชื่อปัจจัยสำคัญที่สองฟังก์ชัน.การคำนวณ
Jacobian เกี่ยวกับ,และชื่อปัจจัยสำคัญที่สามฟังก์ชันเพื่อคำนวณinhomogeneity . matrix
Jacobian ฟังก์ชันต้องมีรูปแบบ.

jac = j (*x*, *t*, *p*)

ในซึ่งเอ็กซ์,t ,และp มีความหมายซึ่งเหมือนกันเช่นเหนือสำหรับฟังก์ชันf ,และjac เป็นmatrixของ
อนุพันธ์เพียงบางส่วน.

$$jac = \frac{df\_i}{dx\_j}$$

ฟังก์ชันb ต้องมีรูปแบบ.

dfdp = b (*x*, *t*, *p*, *c*)

Control Theory

tf format variables

num

numerator เลขคูณ(เวกเตอร์).

den

เลขคูณตัวหาร(เวกเตอร์).

zp format variables

zer

ระบบ เป็น 0 (vector)

Pole

ระบบ เป็น pole (vector)

k

การนำเลขคูณ(scalar).

29.1.4 ss format variables

a

b

c

d

เป็นประจำstate-space matrices . ถ้าระบบมีทั้งคู่ต่อเนื่องกันและdiscrete สภาพพวกเขาถูกแบ่งกลุ่มเพื่อที่สภาพซึ่งต่อเนื่องกันมาอันดับแรก,ดังนั้นdiscrete สภาพ.

บันทึกบางหน้าที่(e. g. ,เป็นลาง,hinfsyn)จะไม่ยอมรับระบบด้วยทั้งคู่discrete และต่อเนื่องกันของ state output.

stname

ชื่อของระบบ(อัก ษ ร).

System Construction and Interface Functions

Finite impulse response system interface functions

Function File: fir2sys (*num*, *tsam*, *inname*, *outname*)

โครงสร้างข้อมูลระบบจากFIR

Inputs:

num

เวกเตอร์ของเลขคูณของSISO FIRการย้ายฟังก์ชัน.

C(z) = c0 + c1*z^{-1} + c2*z^{-2} + ... + znz^{-n}

tsam

sampling time (default: 1)

inname

ชื่อของสัญญาณอินพุท;อาจจะเป็นอักษรหรือรายการด้วยทางเข้าเดี่ยว.

outname

Outputs *sys* (system data structure)

Example

```
octave:1> sys = fir2sys([1 -1 2 4],0.342,"A/D input","filter output");
octave:2> sysout(sys)
Input(s)
     1: A/D input

Output(s):
     1: filter output (discrete)

Sampling interval: 0.342
transfer function form:
1*z^3 - 1*z^2 + 2*z^1 + 4
```

1*z^3 + 0*z^2 + 0*z^1 + 0

Function File: **[*c*, *tsam*, *input*, *output*] = sys2fir *(sys)***

สิ่งที่ดึงออกFIRข้อมูลจากโครงสร้างข้อมูลระบบ;พบfir2sys สำหรับการพรรณาปัจจัยกำหนด.

State space system interface functions

Function File: **ss2sys (*a*, *b*, *c*, *d*, *tsam*, *n*, *nz*, *stname*, *inname*, *outname*, *outlist*)**

สร้างโครงสร้างระบบจากstate-space ข้อมูล. อาจจะเป็นcontinous ,discrete ,หรือผสม(sampeled-data).

Inputs

a

b

c

d

usual state space matrices.

default: *d* = zero matrix

tsam

sampling rate. Default: (continuous system)

n

nz

number of continuous, discrete states in the system

If *tsam* is 0, , .

If *tsam* is greater than zero, ,

see below for system partitioning

stname

list of strings of state signal names

default (*stname*=[] on input): x_n for continuous states, xd_n for discrete states

inname

list of strings of input signal names

default (*inname* = [] on input): u_n

outname

list of strings of input signal names

default (*outname* = [] on input): y_n

outlist

list of indices of outputs y that are sampled

If *tsam* is 0, .

If *tsam* is greater than 0, .

ไม่เหมือนสภาพ,discrete/continous สิ่งที่ออกมาอาจจะปรากฏในคำสั่งอื่น
Note sys2ss เวกเตอร์การกลับคืนyd ที่ไหนyd (outlist)= 1 ;ทางเข้าอื่นทั้งหมดเกี่ยวกับyd เป็น 0 .

Outputs ***outsys* = system data structure**

System partitioning

คาดคะเนสำหรับนั่นความเรียบง่ายoutlist ระบุซึ่งที่หลายสิ่งที่ออกมาแรกเป็น.ต่อเนื่องกันและสิ่งที่ออกมายังอยู่เป็นdiscrete . ดังนั้นระบบถูกแบ่งกั้นอย่างที่.

.. x = [xc] (n x 1)

[xd] (nz x 1 discrete states)

a = [acc acd] b = [bc]

   [adc add]   [bd]

c = [ccc ccd] d = [dc]

   [cdc cdd]   [dd]

   (cdc = c(outlist,1:n), etc.)

        with dynamic equations:

        Signal partitions

	continuous	discrete
states	stname(1:n,:)	stname((n+1):(n+nz),:)
outputs	outname(cout,:)	outname(outlist,:)

รายการของใน 1 เป็นที่ไหน:แถว(p)ซึ่งไม่ถูกบรรจุในoutlist .
(Discrete/continuous สิ่งที่ออกมาอาจจะถูกใส่เข้าในคำสั่งอื่นโดยผู้ใช้). .
..

        Example

```
octave:1> a = [1 2 3; 4 5 6; 7 8 10];
octave:2> b = [0 0 ; 0 1 ; 1 0];
octave:3> c = eye(3);
octave:4> sys = ss2sys(a,b,c,[],0,3,0,list("volts","amps","joules"));
octave:5> sysout(sys);
Input(s)
      1: u_1
      2: u_2
```

Output(s):

1: y_1

2: y_2

3: y_3

state-space form:

3 continuous states, 0 discrete states

State(s):

1: volts

2: amps

3: joules

A matrix: 3 x 3

1 2 3

4 5 6

7 8 10

B matrix: 3 x 2

0 0

0 1

1 0

C matrix: 3 x 3

1 0 0

0 1 0

0 0 1

D matrix: 3 x 3

0 0

0 0

0 0

ข้อสังเกตุซึ่งmatrixเป็นสร้างโดยการผิดสัญญาต่อข้อสังเกตุซึ่งmatrixถูกสร้างโดยd .

Function File: **[*a*, *b*, *c*, *d*, *tsam*, *n*, *nz*, *stname*, *inname*, *outname*, *yd*] = sys2ss *(sys)***

การแสดงช่องว่างหน้าที่สิ่งที่ดึงออกจากโครงสร้างข้อมูลระบบ.

Inputs *sys* system data structure

Outputs

a

b

c

d

state space matrices for sys

tsam

sampling time of sys (0 if continuous)

n

nz

number of continuous, discrete states (discrete states come last in state vector *x*)

stname

inname

outname

signal names (lists of strings); names of states, inputs, and outputs, respectively

yd

binary vector; *yd*(*ii*) is 1 if output *y*(*ii*)$ is discrete (sampled); otherwise *yd*(*ii*) 0.

คำเตือนถูกพิมพ์ถ้าระบบเป็นต่อเนื่องกันผสมรวมๆและdiscrete

Example

```
octave:1> sys=tf2sys([1 2],[3 4 5]);
octave:2> [a,b,c,d] = sys2ss(sys)
a =
   0.00000   1.00000
  -1.66667  -1.33333
b =
  0
  1
c = 0.66667  0.33333
```

d = 0

Transfer function system interface functions

Function File: tf2sys (*num*, *den*, *tsam*, *inname*, *outname*)

สร้างโครงสร้างข้อมูลระบบจากข้อมูลการย้ายฟังก์ชันรูปแบบ.

Inputs

num

den

coefficients of numerator/denominator polynomials

tsam

sampling interval. default: 0 (continuous time)

inname

outname

input/output signal names; may be a string or list with a single string entry.

Outputs *sys* = system data structure

Example

```
octave:1> sys=tf2sys([2 1],[1 2 1],0.1);
octave:2> sysout(sys)
Input(s)
    1: u_1
Output(s):
    1: y_1 (discrete)
Sampling interval: 0.1
transfer function form:
2*z^1 + 1
----------------
1*z^2 + 2*z^1 + 1
```

2*z^1 + 1

1*z^2 + 2*z^1 + 1

Function File: **[*num*, *den*, *tsam*, *inname*, *outname*] = sys2tf *(sys)***

การย้ายสิ่งที่ดึงออกฟังก์ชันข้อมูลจากโครงสร้างข้อมูลระบบ.

พบtf2sys สำหรับการพรรณาปัจจัยกำหนด.

Example

```
octave:1> sys=ss2sys([1 -2; -1.1,-2.1],[0;1],[1 1]);
octave:2> [num,den] = sys2tf(sys)
num = 1.0000  -3.0000
den = 1.0000   1.1000  -4.3000
```

Zero-pole system interface functions

Function File: **zp2sys (*zer*, *pol*, *k*, *tsam*, *inname*, *outname*)**

สร้างโครงสร้างข้อมูลระบบจากzero-pole ข้อมูล.

Inputs

zer

vector of system zeros

pol

vector of system poles

k

scalar leading coefficient

tsam

sampling period. default: 0 (continuous system)

inname

outname

input/output signal names (lists of strings)

Outputs sys: system data structure

Example

```
octave:1> sys=zp2sys([1 -1],[-2 -2 0],1);
octave:2> sysout(sys)
Input(s)
      1: u_1
Output(s):
      1: y_1
zero-pole form:
1 (s - 1) (s + 1)
-----------------
s (s + 2) (s + 2)
```

Function File: **[*zer*, *pol*, *k*, *tsam*, *inname*, *outname*] = sys2zp** ***(sys)***

สิ่งที่ดึงออกzero/pole/leading ข่าวสารเลขคูณจากโครงสร้างข้อมูลระบบ.

.. **Example**

```
octave:1> sys=ss2sys([1 -2; -1.1,-2.1],[0;1],[1 1]);
octave:2> [zer,pol,k] = sys2zp(sys)
zer = 3.0000
pol =
  -2.6953
   1.5953
k = 1
```

Data structure access functions

Function File: **syschnames** ***(sys, opt, list, names)***

แทนที่โดยsyssetsignals .

..

Function File: **syschtsam** ***(sys, tsam)***

ฟังก์ชันนี้เปลี่ยนเวลาการสุ่มตัวอย่าง(tsam)ของระบบ. ทางออกกับความผิดถ้าsys เป็นซีอย่างบริสุทธิ์.

Function File: **[*n*, *nz*, *m*, *p*, *yd*] = sysdimensions** ***(sys, opt)***

คืนจำนวนสภาพ,อินพุท,and/or สิ่งที่ออกมาในระบบsys

Inputs

sys

system data structure

opt

String indicating which dimensions are desired. Values:

"all"

(default) return all parameters as specified under Outputs below.

"cst"

return *n*= number of continuous states

"dst"

return *n*= number of discrete states

"in"

return *n*= number of inputs

"out"

return *n*= number of outputs

Outputs

n

number of continuous states (or individual requested dimension as specified by *opt*).

nz

number of discrete states

m

number of system inputs

p

number of system outputs

yd

binary vector; *yd*(*ii*) is nonzero if output *ii* is discrete. if output *ii* is continous

Function File: [*stname*, *inname*, *outname*, *yd*] = sysgetsignals *(sys)*

@deftypefnx{Function File}: *siglist* = sysgetsignals *(sys, sigid)*

@deftypefnx{Function File}: *signame* = sysgetsignals *(sys, sigid, signum, strflg)*

Get signal names from a system

Inputs

sys

system data structure for the state space system

sigid

signal id. String. Must be one of

"in"

input signals

"out"

output signals

"st"

stage signals

"yd"

value of logical vector *yd*

signum

index(indices) or name(s) or signals; see sysidx

strflg

flag to return a string instead of a list; Values:

0

(default) return a list (even if signum specifies an individual signal)

1

return a string. Exits with an error if signum does not specify an individual signal.

Outputs

names @bullet{If *sigid* is not specified}

stname

inname

outname

signal names (lists of strings); names of states, inputs, and outputs, respectively

yd

binary vector; *yd*(*ii*) is nonzero if output *ii* is discrete.

@bullet{If *sigid* is specified but *signum* is not specified, then}

sigid="in"

siglist is set to the list of input names

sigid="out"

siglist is set to the list of output

sigid="st"

siglist is set to the list of state names

stage signals

sigid="yd"

siglist is set to logical vector indicating discrete outputs; *siglist*(*ii*) = 0 indicates that output *ii* is continuous (unsampled), otherwise it is discrete.

@bullet{if the first three input arguments are specified, then *signame* is}

a list of the specified signal names (*sigid* is "in", "out", or "st"), or else the logical flag indicating whether output(s) *signum* is(are) discrete (*sigval*=1) or continuous (*sigval*=0).

Examples (From sysrepdemo)

```
octave> sys=ss2sys(rand(4),rand(4,2),rand(3,4));
octave> [Ast,Ain,Aout,Ayd] = sysgetsignals(sys) i  # get all signal names
Ast =
(
  [1] = x_1
  [2] = x_2
  [3] = x_3
  [4] = x_4
```

```
)
Ain =
(
  [1] = u_1
  [2] = u_2
)
Aout =
(
  [1] = y_1
  [2] = y_2
  [3] = y_3
)
Ayd =

  0  0  0
octave> Ain = sysgetsignals(sys,"in")   # get only input signal names
Ain =
(
  [1] = u_1
  [2] = u_2
)
octave> Aout = sysgetsignals(sys,"out",2)   # get name of output 2 (in list)
Aout =
(
  [1] = y_2
)
octave> Aout = sysgetsignals(sys,"out",2,1)  # get name of output 2 (as string)
Aout = y_2
```

Function File: **sysgettype** ***(sys)***

คืนประเภทระบบแรกเริ่มของระบบ.

Inputs *sys*: system data structure

Outputs *systype*: อักษรชี้บอกวิธีที่โครงสร้างถูกสร้างตั้งแต่ดั้งเดิม:

: values: "ss", "zp", or "tf"

โน้ต FIRตั้งค่าการกลับคืนระบบsystype= "tf ".

Function File: syssetsignals ***(sys, opt, names, sig_idx)***

เปลี่ยนชื่อของอินพุทซึ่งถูกเลือกแล้ว,สิ่งที่ออกมาและสภาพ.

. **Inputs**

sys

system data structure

opt

change default name (output)

"out"

change selected output names

"in"

change selected input names

"st"

change selected state names

"yd"

change selected outputs from discrete to continuous or from continuous to discrete.

names

opt = "out", "in", or "st"

string or string array containing desired signal names or values.

opt = "yd"

To desired output continuous/discrete flag. Set name to 0 for continuous, or 1 for discrete.

sig_idx

indices or names of outputs, yd, inputs, or states whose respective names/values should be changed.

Default: replace entire list of names/entire yd vector.

Outputs ***retsys***=*sys* with appropriate signal names changed (or yd values, where appropriate)

Example

```
octave:1> sys=ss2sys([1 2; 3 4],[5;6],[7 8]);
octave:2> sys = syssetsignals(sys,"st",str2mat("Posx","Velx"));
octave:3> sysout(sys)
Input(s)
        1: u_1
Output(s):
        1: y_1
state-space form:
2 continuous states, 0 discrete states
State(s):
        1: Posx
        2: Velx
A matrix: 2 x 2
  1  2
  3  4
B matrix: 2 x 1
  5
  6
C matrix: 1 x 2
  7  8
D matrix: 1 x 1
  0
```

Function File: **sysupdate** ***(sys, opt)***

ปรับปรุงการแสดงภายในของระบบ.

Inputs

sys:

system data structure

opt

string:

"tf"

update transfer function form

"zp"

update zero-pole form

"ss"

update state space form

"all"

all of the above

Outputs *retsys* บรรจุสหภาพของข้อมูลในsys และร้องขอข้อมูล. ถ้าข้อมูลซึ่งร้องขอแล้วในsys อยู่บนแล้ว. ดังนั้นretsys=sys .

การเปลี่ยนแปลงต่อtf หรือzp ทางออกกับความผิดถ้าระบบถูกผสมcontinuous/digital **.**

function [systype, nout, nin, ncstates, ndstates] = minfo(inmat)

MINFO:ตัดสินใจประเภทของระบบmatrix. INMATเป็นการเปลี่ยนแปลง(*),ระบบ,จำนวนคงที่, สามารถและmatrixว่างเปล่า.

Returns: systype สามารถหนึ่ง:การเปลี่ยนแปลง,ระบบ,จำนวนคงที่,และทำให้ว่างเปล่าnout . เป็นจำนวนสิ่งที่ออกมาของระบบnin เป็นจำนวนอินพุทของระบบncstates เป็น. จำนวนอินพุทของระบบncstates เป็นจำนวนสภาพซึ่งต่อเนื่องกันเกี่ยวกับ.สภาพซึ่งต่อเนื่องกันของระบบndstates เป็นจำนวนdiscrete สภาพของระบบ

Function File: sysgettsam ***(sys)***

กลับไปยังการสุ่มตัวอย่างของระบบsys .

..

System display functions.

Function File: sysout ***(sys, opt)***

zero-pole form

"all" แสดงโครงสร้างข้อมูลระบบในรูปแบบที่ต้องการ

sys

system data structure

opt

Display option

[]

primary system form (default)

"ss"

state space form

"tf"

transfer function form

"zp"

all of the above

Function File: **tfout** ***(num, denom, x)***

แสดงรูปแบบการย้ายฟังก์ชันต่อหน้าจอ. เอ็กซ์การผิดสัญญาต่ออักษร"s ".

Function File: **zpout** ***(zer, pol, k, x)***

แสดงรูปแบบzero-pole รูปแบบต่อหน้าจอ. เอ็กซ์การผิดสัญญาต่ออักษร"s ".

Block Diagram Manipulations

Function File: **bddemo** ***(inputs)***

เดโมกล่องเครื่องมือOctaveการควบคุม:เดโมการจัดการแผนผังบล็อก

Function File: **buildssic (*clst*, *ulst*, *olst*, *ilst*, *s1*, *s2*, *s3*, *s4*, *s5*, *s6*, *s7*, *s8*)**

รูปแบบจำนวนเชิงซ้อนซึ่งโดยพลการ(เปิดออกหรือรูซึ่งปิดแล้ว)ระบบในstate-space รูปแบบจากหลายระบบ.

buildssic "อย่างง่ายดายสามารถ(แม้ว่ามันเป็นซินแท็กซึ่งลับ)ทำให้การย้ายรวมกันหน้าที่จากจำนวนเชิงซ้อน.

แผนผังบล็อกในระบบเดี่ยวกับหนึ่งเรียก. ฟังก์ชันนี้เป็นประโยชน์โดยเฉพาะอย่างยิ่งสำหรับการสร้างเปิดการเชื่อมต่อระหว่างกันรูสำหรับH_infinity และH2 การออกแบบหรือ.สำหรับการปิดรูกับตัวควบคุมเหล่านี้.

ปัจจัยกำหนดประกอบด้วย 4 รายการซึ่งบรรยายสิ่งที่ออกมาการเชื่อมต่อ สิ่งที่ออกมาและอินพุทและบนต่อ 8 ระบบs1-s8 . รูปแบบของรายการ:

..

clst

ายการการเชื่อมต่อ,บรรยายสัญญาณอินพุทของแต่ละระบบ.

ค่าสูงสุดจำนวนแถวเกี่ยวกับClst เท่ากับผลรวมของอินพุททั้งหมดของs1-s8 .

Example: [1 2 -1; 2 1 0] ==> new input 1 is old inpout 1 + output 2 - output 1, new input 2 is old input 2 + output 1. The order of rows is arbitrary.

ulst

ถ้าไม่ว่างเปล่าอินพุทเก่า/แก่ในเวกเตอร์Ulst จะถูกเพิ่มเติมต่อสิ่งที่ออกมา.
คุณจำเป็นนี่ถ้าคุณต้องการที่จะ"ถอนออก"อินพุทของระบบ.

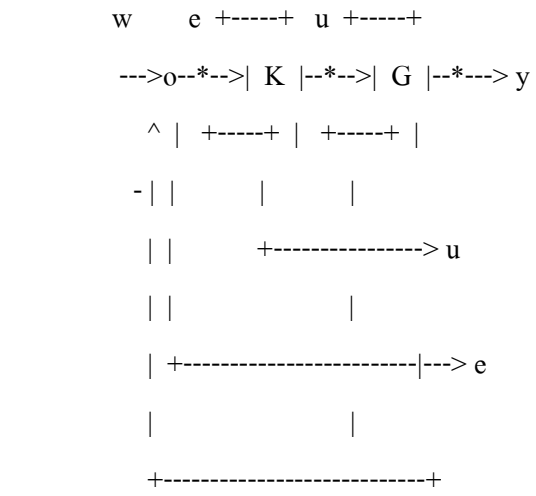
olst

outputรายการ,specifiy สิ่งที่ออกมาของระบบเป็นผล.
ปัจจัยสำคัญเป็นoutputจำนวนs1-s8 . จำนวนเป็นalowed เพื่อเป็นการปฏิเสธและอาจจะปรากฏในใครก็ได้.

ilst

รายการอินพุท,specifiy อินพุทของระบบเป็นผล. จำนวนเป็นalowed เพื่อเป็นการปฏิเสธและอาจจะปรากฏในคำสั่งอื่น. matrixว่างเปล่าทั้งหมดในอินพุท.

. Example: Very simple closed loop system.



The closed loop system GW can be optained by

```
GW = buildssic([1 2; 2 -1], 2, [1 2 3], 2, G, K);
```

clst

(1. row) connect input 1 (G) with output 2 (K). (2. row) connect input 2 (K) with neg. output 1 (G).

ulst

append input of (2) K to the number of outputs.

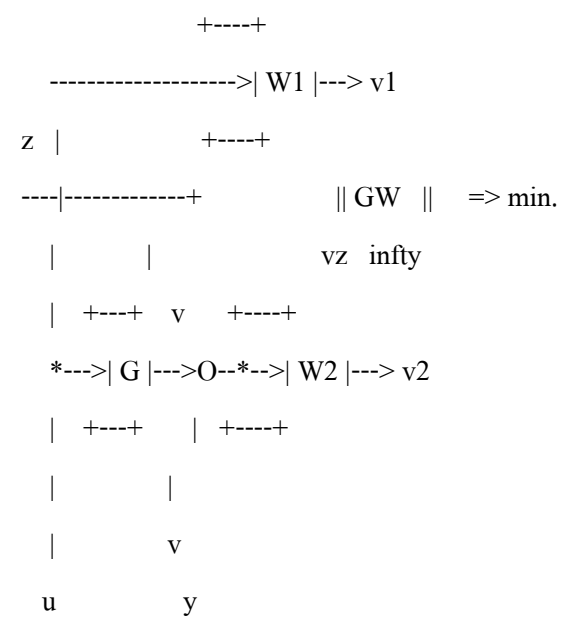
olst

Outputs are output of 1 (G), 2 (K) and appended output 3 (from Ulst).

ilst

the only input is 2 (K).

Here is a real example:



The closed loop system GW from [z; u]' to [v1; v2; y]' can be obtained by (all SISO systems):

```
GW = buildssic([1, 4; 2, 4; 3, 1], 3, [2, 3, 5],
               [3, 4], G, W1, W2, One);
```

"one"เป็นunity สิ่งที่ได้รับ(auxillary)ฟังก์ชันกับคำสั่ง 0 . (e. g. one= ugain (1));.

Function File: jet707 *()*

สร้างlinearized รูปแบบช่องว่างหน้าที่ของBoeing 707-321 aircraft v=80m/s. (M = 0.26, Ga0 = -3 deg, alpha0 = 4 deg, kappa = 50 deg) System inputs: (1) thrust and (2) elevator angle System

outputs: (1) airspeed and (2) pitch angle Ref: R. Brockhaus: Flugregelung (Flight Control), Springer, 1994

Function File: ord2 ***(nfreq, damp, gain)***

สร้างต่อเนื่องกัน2nd ระบบคำสั่งกับปัจจัยกำหนด:

..

Inputs

nfreq

natural frequency [Hz]. (not in rad/s)

damp

damping coefficient

gain

dc-gain This is steady state value only for damp > 0. gain is assumed to be 1.0 if ommitted.

Outputs *outsys* system data structure has representation with :

```
                /                                  \
                | / -2w*damp -w \  / w \                 |
          G = | |              |, |   |, [ 0  gain ], 0 |
                | \   w       0 /  \ 0 /                 |
                \                                  /
```

See also jet707 (MIMO example, Boeing 707-321 aircraft model)

Function File: sysadd ***(gsys, hsys)***

returns *sys* = *gsys* + *hsys*.

..Exitsกับความผิดถ้าgsys และhsys ไม่เป็นcompatibly dimensioned .

..ข้อความจะเตือนผลที่เป็นสภาพระบบมีชื่อเหมือนกัน. duplicate ชื่อถูกให้คำเสริมท้ายเพื่อให้ค่าไม่เหมือนค่าใด

..sys input/output ชื่อถูกเอาจากgsys .

```
                 _________
              ---|  gsys  |---
          u   |   ----------  +|
          -----               (_)----> y
            |    _________    +|
              ---|  hsys  |---
```

Function File: **sysappend (*sys*, *b*, *c*, *d*, *outname*, *inname*, *yd*)**

เพิ่มเติมอินพุทใหม่and/or สิ่งที่ออกมาต่อระบบ.

Inputs

sys

system data structure

b

matrix to be appended to sys "B" matrix (empty if none)

c

matrix to be appended to sys "C" matrix (empty if none)

d

revised sys d matrix (can be passed as [] if the revised d is all zeros)

outname

list of names for new outputs

inname

list of names for new inputs

yd

binary vector; indicates a continuous output; indicates a discrete output.

Outputs *sys*

```
sys.b := [sys.b , b]
sys.c := [sys.c  ]
         [ c     ]
sys.d := [sys.d | D12 ]
         [D21   | D22 ]
```

เป็นเหมาะสมdimensioned บล็อกของปัจจัยกำหนดอินพุทd .

..บล็อกซึ่งสำคัญเกี่ยวกับd ถูกละเลย.

..ถ้าinname และoutname ถูกให้อย่างที่argument. อินพุทใหม่และเอาท์พุทเป็นถูกมอบหมายชื่อการผิดสัญญา.

yd เป็นเวกเตอร์เลขคู่ของแถวความยาว(ซี)ซึ่งชี้บอกcontinuous/sampled output. ค่าสำหรับ yd เป็น

sys = continuous or mixed *yd* = zeros(1,rows(c))

sys = discrete *yd* = ones(1,rows(c))

Function File: sysconnect (*sys*, *out_idx*, *in_idx*, *order*, *tol*)

ปิดloopตั้งแต่ระบุoutputถึงของแต่ละอินพุท.

Inputs

sys

system data structure

out_idx

in_idx

names or indices of signals to connect (see sysidx). The output specified by is connected to the input specified by .

order

logical flag (default = 0)

0

leave inputs and outputs in their original order

1

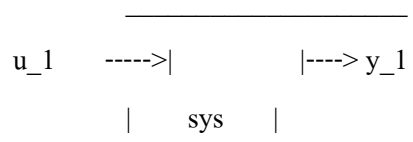
permute inputs and outputs to the order shown in the diagram below

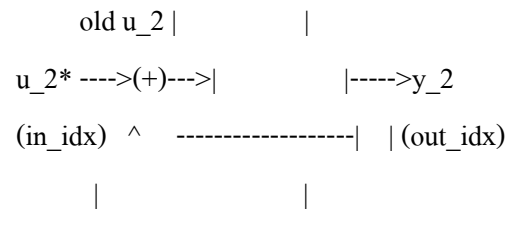
tol

tolerance for singularities in algebraic loops default: 200*eps*

Outputs *sys*: resulting closed loop system.

วิธีการsysconnect เรียงลำดับอินพุทซึ่งถูกเลือกแล้วภายใน. outputเช่นแสดงต่ำกว่า,ปิดloop. เรียงลำดับอินพุทครั้นแล้วและoutputหลังต่อคำสั่ง.





อินพุทซึ่งมีการเชื่อมต่อรวมยอดเพิ่มเติมมันมี. * เพิ่มเติมตอนจบของชื่ออินพุท.

Function File: **[*csys*, *acd*, *ccd*] = syscont *(sys)***

สกัดระบบย่อยซึ่งต่อเนื่องกันของระบบอินพุท.

Inputs *sys* is a system data structure

Outputs

csys

is the purely continuous input/output connections of *sys*

acd

ccd

connections from discrete states to continuous states, discrete states to continuous outputs, respectively.

returns *csys* empty if no continuous/continous path exists

Function File: **[*dsys*, *adc*, *cdc*] = sysdisc *(sys)***

Inputs *sys* = system data structure

Outputs

dsys

discrete ส่วนแบ่งเกี่ยวกับsys .

(คืนทำให้ว่างเปล่าถ้าdiscrete ทางตั้งแต่อินพุทถึงoutput).

adc

cdc

การเชื่อมต่อตั้งแต่สภาพซึ่งต่อเนื่องกันถึงdiscrete สภาพและdiscrete output,ตามลำดับ.

Function File: **sysdup** ***(asys, out_idx, in_idx)***

สำเนาระบุinput/output การเชื่อมต่อของระบบ.

Inputs

asys

system data structure

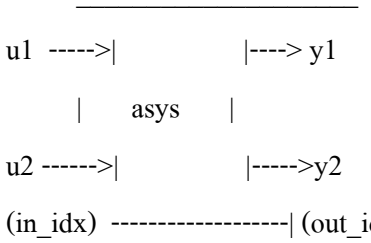
out_idx

in_idx

indices or names of desired signals (see sigidx). duplicates are made of y(out_idx(ii)) and u(in_idx(ii)).

Outputs *retsys*: resulting closed loop system: duplicated i/o names are appended with a "+" suffix.

วิธีการsysdup สร้างสำเนาของอินพุทซึ่งถูกเลือกแล้วและoutput. เช่นแสดงเบื้องล่าง. u1/y1 เป็นการจัดตั้งของของแท้inputs/output . และu2 ,y2 เป็นการจัดตั้งของทำซ้ำinputs/outputs ในคำสั่งระบุใน in_idx ,out_idx ,. ตามลำดับ

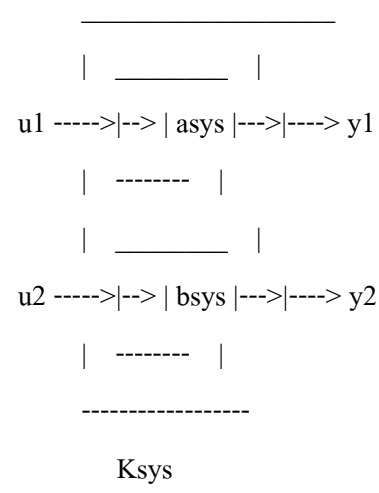


Function File: **sysgroup** ***(asys, bsys)***

รวม2ระบบของระบบเดี่ยว

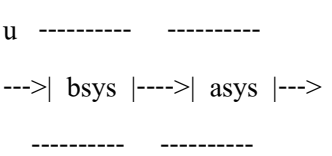
Inputs ***asys*****,** ***bsys*****: system data structures**

Outputs



Function File: **sysmult** ***(asys, bsys)***

คำนวณกราต่อแบบอนุกรม



Function File: **sysprune** ***(asys, out_idx, in_idx)***

หาค่าของอินพุทและเอาท์พุทของระบบ

Inputs

asys

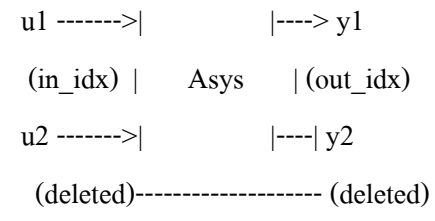
system data structure

out_idx

in_idx

```
retsys = sysprune(Asys,[1:3,4],"u_1");
retsys = sysprune(Asys,list("tx","ty","tz"), 4);
```

Outputs *retsys*: resulting system



Function File: **sysreorder** ***(vlen, list)***

Inputs *vlen*=vector length, *list*= a subset of [1:vlen],

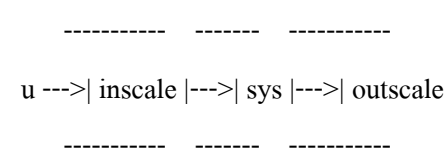
Outputs *pv*: a permutation vector to order elements of [1:vlen] in list to the end of a vector.

Function File: **sysscale (*sys*, *outscale*, *inscale*, *outname*, *inname*)**

หาสเกลอินพุทและเอาท์พุทของระบบ

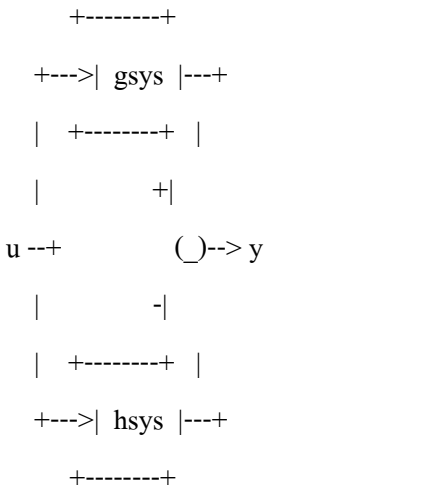
input **sys :**สร้างระบบoutscale ,inscale . จำนวนคงที่matrices ของมิติซึ่งเหมาะสม.

Outputs *sys*: resulting open loop system:



Function File: **syssub** ***(gsys, hsys)***

วิธีการ:gsys และhsys ถูกเชื่อมต่อเวกเตอร์ในทำให้อินพุทขนานถูกเชื่อมต่อต่อทั้งคู่ระบบ. outputถูกลบออก. ชื่อระบบซึ่งคืนแล้วเหล่านั้นเกี่ยวกับgsys .



Function File: **parallel** ***(asys, bsys)***

สร้างการเชื่อมต่อแบบขนานเป็นรูปร่างของสองระบบ

_____ | _____ | u ----->|----> | asys |--->|----> y1 | | ------ -- | | | _____ | |--->|----> | bsys |--->|----> y2 | -------- | ---------- ksys

System Analysis-Properties

Function File: **analdemo *()*System Analysis-Properties**

กล่องเครื่องมือของ octave

Function File: **[*n*, *m*, *p*] = abcddim *(a, b, c, d)***

การตรวจสอบสำหรับความเข้ากันได้ของมิติของmatrices กำลังจำกัดความระบบตามระยะยาว.

dx/dt = a x + b u

y = c x + d u

Function File: **ctrb *(sys, b)***

Function File: ctrb *(a, b)*

สร้างความสามารถควบคุมmatrix.

2 n-1

Qs = [B AB A B ... A B]

Function File: **obsv *(sys, c)***

สร้างobservability matrix.

| C |

| CA |

Qb = | CA^2 |

| ... |

| CA^(n-1) |

ของโครงสร้างข้อมูลระบบหรือคู่(,ซี).

Function File: **[*zer*, *pol*]= pzmap *(sys)***

กำหนดzeros และpoleของระบบในจำนวนเชิงซ้อนระดับ.

Inputs sys โครงสร้างข้อมูลระบบ**.**

Outputs ถ้าละเว้น,polesและzeros ถูกกำหนดบนหน้าจอ. มิฉะนั้น. pol ,zer ถูกคืนเช่นระบบpoles และzeros .

Function File: [*retval*, *u*] = is_controllable *(sys, tol)*

Function File: [*retval*, *u*] = is_controllable *(a, b, tol)*

การตรวจสอบซึ่งมีความน่าจะเป็นสำหรับความสามารถควบคุมระบบ

Inputs

sys

system data structure

a

b

n by *n*, *n* by *m* matrices, respectively

tol

optional roundoff paramter. default value: 10*eps

Outputs

retval

Logical flag:คืนความจริง(1)ถ้าระบบsys หรือคู่(,b)เป็นควบคุมได้,อันไหนก็ตามถูกผ่านอย่างที่ **input .**

U

Uเป็นorthogonal หลักสำคัญควบคุมได้subspace .

Function File: [*retval*, *dgkf_struct*] = is_dgkf (*asys*, *nu*, *ny*, *tol*)

ตัดสินใจไม่ว่าที่ระบบช่องว่างหน้าที่เวลาซึ่งต่อเนื่องกันพบการสันนิษฐานของDGKF อัลกอลิทึ่ม. ระบบพาร์ติชั่นใน:

[dx/dt] = [A | Bw Bu][w]

[z] [Cz | Dzw Dzu][u]

[y] [Cy | Dyw Dyu]

Inputs

asys

system data structure

nu

number of controlled inputs

ny

number of measured outputs

tol

threshhold for 0. Default: 200*eps*

Outputs

retval

true(1) if system passes check, false(0) otherwise

dgkf_struct

data structure of is_dgkf results. Entries:

nw

nz

dimensions of *w*, *z*

a

system matrix

bw

(*n* x *nw*) *qw*-transformed disturbance input matrix

bu

(*n* x *nu*) *ru*-transformed controlled input matrix;

Note

cz

(*nz* x *n*) Qz-transformed error output matrix

cy

(*ny* x *n*) *ry*-transformed measured output matrix

Note

dzu

dyw

off-diagonal blocks of transformed system matrix that enter *z*, *y* from *u*, *w* respectively

ru

controlled input transformation matrix

ry

observed output transformation matrix

dyu_nz

nonzero if the *dyu* block is nonzero.

dyu

untransformed *dyu* block

dflg

nonzero if the system is discrete-time

is_dgkf exits with an error if the system is mixed discrete/continuous

References

[1]

Doyle, Glover, Khargonekar, Francis, "State Space Solutions to Standard H2 and Hinf Control Problems," IEEE TAC August 1989

[2]

Maciejowksi, J.M.: "Multivariable feedback design,"

Function File: is_stable *(a, tol, dflg)*

Function File: is_stable *(sys, tol)*

การกลับคืน 1 ถ้าmatrixหรือระบบsys มีเสถียรภาพ,หรือ 0 ถ้าไม่.

Inputs

tol

is a roundoff paramter, set to 200**eps* if omitted.

dflg

Digital system flag (not required for system data structure):

dflg != 0

stable if eig(a) in unit circle

dflg == 0

stable if eig(a) in open LHP (default)

System Analysis-Time Domain

Function File: **c2d** ***(sys, opt, t)***

Function File: **c2d** ***(sys, t)***

Inputs

sys

system data structure (may have both continuous time and discrete time subsystems)

opt

string argument; conversion option (optional argument; may be omitted as shown above)

"ex"

use the matrix exponential (default)

"bi"

use the bilinear transformation

$$s = \frac{2(z-1)}{T(z+1)}$$

FIXME:ทางออกทางเลือกนี้กับความผิดถ้าsys ไม่ต่อเนื่องกัน

Note ถ้า2nd argumentไม่เป็นเชือก,c2d สันนิษฐานซึ่งที่2nd argumentเป็นt และดำเนินการจัดสรรไว้ของการตรวจสอบ argument

Outputs *dsys* discrete จับเวลาสิ่งที่เท่ากันผ่านzero-order ถือ,ทดลองแต่ละt วินาที.
เปลี่ยนโครงสร้างข้อมูลระบบกำลังบรรยาย.

..

$$\dot{x} = Ac\,x + Bc\,u$$

discrete จับเวลารูปแบบเท่ากัน.

$$x[n+1] = Ad\,x[n] + Bd\,u[n]$$

Function File: **d2c** ***(sys, tol)***

Function File: **d2c** ***(sys, opt)***

เปลี่ยนdiscrete (ตัวแทน)ระบบต่อระบบซึ่งต่อเนื่องกัน. ใช้.เป็น

sysgettsam(*sys*)

Inputs

sys

system data structure with discrete components

tol

Scalar value. tolerance for convergence of default "log" option (see below)

opt

conversion option. Choose from:

"log"

ดำเนินการอยู่ผ่านmatrixเลขลอกอริทึม. เพราะว่าบางปัญหากับการคำนวณมันถูกติดตามโดยอัลกอลิที่มการตกลงมาซึ่งสูงสุดเพื่อแสดงตัวเวลาซึ่งต่อเนื่องกัน,b ,เพื่อได้สิ่งที่ดีกว่าเหมาะสมต่อข้อมูลแท้.

"bi"

การเปลี่ยนแปลงถูกดำเนินการผ่านbilinear เปลี่ยนรูปที่ไหนที่เป็นระบบกำลังทดลองเวลา.

Outputs *csys* continuous time system (same dimensions and signal names as in *sys*).

Function File: [*dsys*, *fidx*] = dmr2d (*sys*, *idx*, *sprefix*, *ts2*, *cuflg*)

ผู้เปลี่ยนแปลงmultirate ระบบซึ่งดิจิตอลต่อสภาพระบบซึ่งดิจิตอลอัตราเดี่ยวระบุโดยidx . sprefix ถูกทดลองที่ผู้อื่นทั้งหมดts2 ,ถูกสันนิษฐานทดลองที่.

ts1 = sysgettsam (*sys*).

Inputs

sys

discrete time system; dmr2d exits with an error if *sys* is not discrete

idx

indices or names of states with sampling time sysgettsam(*sys*) (may be empty); see listidx

sprefix

list of string prefixes of states with sampling time sysgettsam(*sys*) (may be empty)

ts2

sampling time of states not specified by *idx*, *sprefix* must be an integer multiple of

sysgettsam(*sys*)

cuflg

"constant u flag"ถ้าcuflg เป็นnonzero ดังนั้นอินพุทระบบถูกสันนิษฐานเป็นจำนวนคงที่เหนือการตรวจแก้กำลังทดลองใน".มิฉะนั้น,ตั้งแต่อินพุทสามารถการเปลี่ยนแปลงระหว่างเวลาระหว่างt ใน, การจัดตั้งเพิ่มขึ้นของอินพุทเป็น.ประกอบด้วยอยู่ในการตรวจแก้B matrixเพื่อที่สิ่งเหล่านี้ intersample อินพุทอาจจะรวมอยู่แล้วใน.ระบบsingle-rate . default *cuflg* = 1.

Outputs

dsys

equivalent discrete time system with sampling time *ts2*.

The sampling time of sys is updated to *ts2*.

if *cuflg*=0 then a set of additional inputs is added to the system with suffixes _d1, ..., _dn to indicate their delay from the starting time k *ts2*, i.e. u = [u_1; u_1_d1; ..., u_1_dn] where u_1_dk is the input k*ts1 units of time after u_1 is sampled. (*ts1* is the original sampling time of the discrete time system and *ts2* = (n+1)*ts1)

fidx

indices of "formerly fast" states specified by *idx* and *sprefix*; these states are updated to the new (slower) sampling interval *ts2*.

Function File: damp ***(p, tsam)***

แสดงeigenvalues ,ความถี่โดยธรรมชาติและการทำให้อัตราส่วนdampingเกี่ยวกับeigenvalues ของ matrix p .

ถ้าp เป็นmatrixและtsam ถูกระบุ,eigenvalues เกี่ยวกับp ถูกสันนิษฐานเป็นในz-domain .

Function File: dcgain ***(sys, tol)***

การกลับคืนdc-gain matrix. ถ้าdc-gain ไม่สิ้นสุดmatrixว่างเปล่าถูกคืน.

Function File: [*y*, *t*] = impulse ***(sys, inp, tstop, n)***

คำตอบสำหรับระบบตามระยะยาว. ระบบสามารถdiscrete หรือสามารถmultivariable ได้.

ถ้าไม่มีoutput argumentถูกระบุ,แผนผังผลหรือข้อมูลคำตอบแสำหรับsystem

Inputs

sys

System data structure.

inp

Index of input being excited

tstop

The argument *tstop* (scalar value) denotes the time when the simulation should end.

n

the number of data values.

Both parameters *tstop* and *n* can be omitted and will be computed from the eigenvalues of the A-Matrix.

Outputs *y*, *t*: impulse response

Function File: [*y*, *t*] = step *(sys, inp, tstop, n)*

คำตอบstepสำหรับระบบเส้นตรง. ระบบสามารถdiscrete หรือสามารถmultivariable ได้.

Inputs

sys

System data structure.

inp

Index of input being excited

tstop

The argument *tstop* (scalar value) denotes the time when the simulation should end.

n

the number of data values.

Both parameters *tstop* and *n* can be omitted and will be computed from the eigenvalues of the A-Matrix.

Outputs *y*, *t*: impulse response

System Analysis-Frequency Domain

Function File: frdemo *()*

. กล่องเครื่องมือของ octave ของ freg.response

Function File: [*mag*, *phase*, *w*] = bode (*sys*, *w*, *out_idx*, *in_idx*)

.ใช้หากราฟของ bode

Inputs

sys

a system data structure (must be either purely continuous or discrete; see is_digital)

w

frequency values for evaluation.

ถ้าsys เป็นต่อเนื่องกัน,ดังนั้นbodeประเมินค่าที่ไหนที่เป็นการย้ายระบบ

ถ้าsys เป็นdiscrete ,ดังนั้นbodeประเมินค่าG (exp (jwT)). G(exp(jwT))

...เป็นระบบกำลังทดลองเวลา.

...เป็นการย้ายระบบฟังก์ชัน.

Function File: [*wmin*, *wmax*] = bode_bounds *(zer, pol, dflg, tsam)*

ได้ขอบเขตการผิดสัญญาของความถี่การตัดออกซึ่งอาศัยพื้นฐานแล้วความถี่ของระบบpolesและ zeros .

ขอบเขตความถี่เป็นเวลาระหว่าง[10wmin ,10wmax] .

Function File: ltifr *(a, b, w)*

Function File: ltifr *(sys, w)*

ตามระยะยาวจับเวลาคำตอบความถี่ไม่เปลี่ยนแปลงของระบบอินพุทเดี่ยว.

Inputs

a

b

coefficient matrices of

sys

system data structure

w

vector of frequencies

Outputs *out*

$$G(s) = (jw\ I - A)^{-1} B$$

Function File: [*realp*, *imagp*, *w*] = nyquist (*sys*, *w*, *out_idx*, *in_idx*, *atol*)

Function File: nyquist (*sys*, *w*, *out_idx*, *in_idx*, *atol*)

Nyquist plot แผนผังของระบบ;ถ้าไม่มีoutput argumentถูกให้,Nyquist แผนผังถูกพิมพ์ต่อ.

Function File: tzero *(a, b, c, d, opt)*

Function File: tzero *(sys, opt)*

คำนวณการส่งมาzeros ของต่อเนื่องกัน.

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

หรือ

$$x(k+1) = A\,x(k) + B\,u(k)$$

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

Outputs

zer

การส่งมาzeros ของระบบ.

.. gain

การนำเลขคูณ(pole-zero รูปแบบ)ของSISOการย้ายฟังก์ชันการกลับคืนgain=0 ถ้าระบบเป็น multivariable

Function File: tzero2 *(a, b, c, d, bal)*

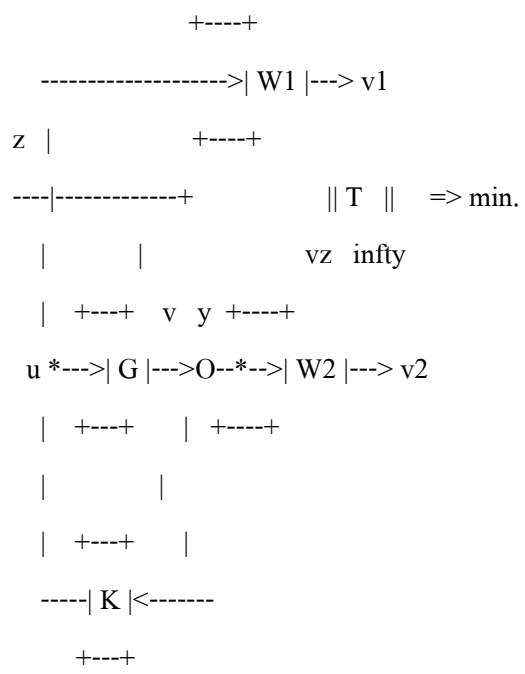
คำนวณการส่งมาzeros ของ a,b ,c,d .

Controller Design

H_infinity การออกแบบdemos สำหรับSISOซึ่งต่อเนื่องกันและMIMOระบบและdiscrete ระบบ SISOระบบยากควบคุมเพราะว่ามันเป็นไม่ทำให้เป็นขั้นตอนน้อยที่สุดและไม่เสถียร.

SISO plant

$$G(s) = \frac{s - 2}{(s + 2)(s - 1)}$$

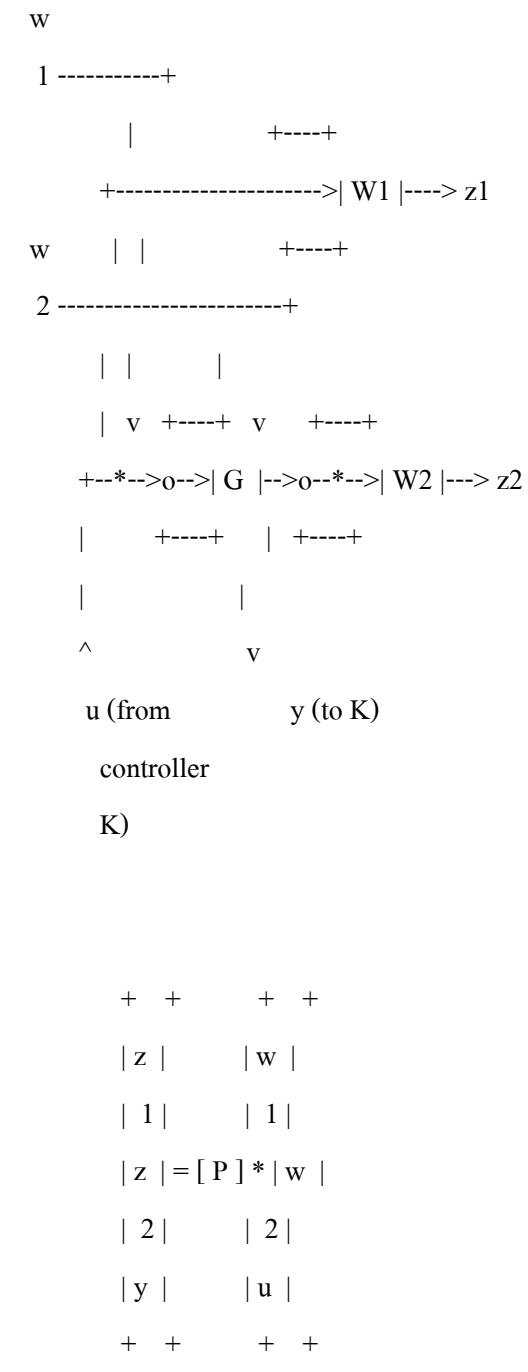


W1 und W2 เป็นความแข็งแรงและการกระทำกำลังเพิ่มความสำคัญฟังก์ชัน.

MIMO plant

w

1



$$\begin{bmatrix} z \\ 1 \\ z \\ 2 \\ y \end{bmatrix} = [P] * \begin{bmatrix} w \\ 1 \\ w \\ 2 \\ u \end{bmatrix}$$

Function File: hinfdemo *()*

DISCRETE SYSTEM

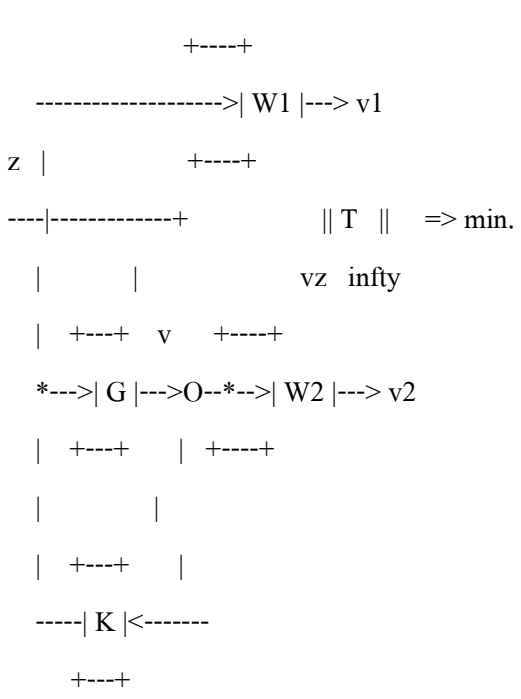
. นี่ไม่เป็นความจริงdiscrete การออกแบบ.การออกแบบถูกทำให้สำเร็จในเวลาซึ่งต่อเนื่องกันขณะที่ผลกระทบเกี่ยวกับการทดลองเป็น.บรรยายโดยbilinear การแปรสภาพของระบบซึ่งทดลอง.
โรงงานวิธีการนี้ทีเดียวบ่อน้ำถ้า.
ถ้าระยะเวลาการสุ่มตัวอย่างเป็น"สิ่งที่เล็ก"เปรียบเทียบกับพืชมั่นคงเวลา.

The continuous plant

$$G(s) = \frac{1}{k \quad (s+2)(s+1)}$$

is discretised with a ZOH (Sampling period = Ts = 1 second):

$$G(s) = \frac{0.199788z + 0.073498}{(z - 0.36788)(z - 0.13534)}$$



Function File: [*l*, *m*, *p*, *e*] = dlqe *(a, g, c, sigw, sigv, z)*

สร้างตามระยะยาวquadratic estimator (Kalman ตัวกรอง)สำหรับdiscrete ระบบเวลา.

$$x[k+1] = A\,x[k] + B\,u[k] + G\,w[k]$$

$$y[k] = C\,x[k] + D\,u[k] + v[k]$$

ที่w ,v เป็นกระบวนการzero-mean gaussian เสียงรบกวนกับความเข้มข้นของแต่ละตัว

sigw = cov (*w*, *w*) and *sigv* = cov (*v*, *v*).

ถ้า specified, *z* is cov (*w*, *v*). Otherwise cov (*w*, *v*) = 0.

The observer structure is

z[k|k] = z[k|k-1] + L (y[k] - C z[k|k-1] - D u[k])

z[k+1|k] = A z[k|k] + B u[k]

The following values are returned:

l

The observer gain, (*a* - *alc*). is stable.

m

The Riccati equation solution.

p

The estimate error covariance after the measurement update.

e

The closed loop poles of (*a* - *alc*).

Function File: [*k*, *p*, *e*] = dlqr *(a, b, q, r, z)*

สร้างตามระยะยาวquadratic ที่บังคับน้ำสำหรับdiscrete ระบบเวลา.

x[k+1] = A x[k] + B u[k]

to minimize the cost functional

J = Sum (x' Q x + u' R u)

z omitted or

J = Sum (x' Q x + u' R u + 2 x' Z u)

z included.

The following values are returned:

k

The state feedback gain, (*a* - *bk*) is stable.

p

The solution of algebraic Riccati equation.

e

The closed loop poles of (*a* - *bk*).

Function File: [*Lp*, *Lf*, *P*, *Z*] = dkalman *(A, G, C, Qw, Rv, S)*

สร้างตามระยะยาวquadratic estimator (Kalman predictor)สำหรับdiscrete ระบบเวลา.

x[k+1] = A x[k] + B u[k] + G w[k]

y[k] = C x[k] + D u[k] + v[k]

ที่w ,v เป็นกระบวนการzero-mean gaussian เสียงรบกวนกับความเข้มข้นของแต่ละตัว

Qw = cov (*w*, *w*) and *Rv* = cov (*v*, *v*).

,ถ้า *S* is cov (*w*, *v*). Otherwise cov (*w*, *v*) = 0..

The observer structure is

x[k+1|k] = A x[k|k-1] + B u[k] + LP (y[k] - C x[k|k-1] - D u[k])

x[k|k] = x[k|k-1] + LF (y[k] - C x[k|k-1] - D u[k])

The following values are returned:

Lp

The predictor gain, (*A* - *Lp C*) is stable.

Lf

The filter gain.

P

The Riccati solution.

P = E [(x - x[n|n-1])(x - x[n|n-1])']

Z

The updated error covariance matrix.

Z = E [(x - x[n|n])(x - x[n|n])']

Function File: {[K}, *gain*, *kc*, *kf*, *pc*, *pf*] = h2syn (*asys*, *nu*, *ny*, *tol*)

การออกแบบH2 ตัวควบคุมดีที่สุดต่อขบวนการในDoyle ,Glover ,Khargonekar ,Francis ,.

Inputs input system is passed as either

asys

system data structure (see ss2sys, sys2ss)

controller is implemented for continuous time systems

controller is NOT implemented for discrete time systems

nu

number of controlled inputs

ny

number of measured outputs

tol

threshhold for 0. Default: 200*eps

Outputs

k

system controller

gain

optimal closed loop gain

kc

full information control (packed)

kf

state estimator (packed)

pc

ARE solution matrix for regulator subproblem

pf

ARE solution matrix for filter subproblem

Function File: hinf_ctr ***(dgs, f, h, z, g)***

เรียกโดยhinfsyn เพื่อคำนวณH_inf ตัวควบคุมดีที่สุด.

Inputs

dgs

data structure returned by is_dgkf

f

h

feedback and filter gain (not partitioned)

g

final gamma value

Outputs controller (system data structure)

ไม่พยายามใช้นี่ที่บ้าน;ไม่มีargumentกำลังตรวจสอบดำเนินการ.

Function File: [*k*, *g*, *gw*, *xinf*, *yinf*] = hinfsyn (*asys*, *nu*, *ny*, *gmin*, *gmax*, *gtol*, *ptol*, *tol*)

Inputs input system is passed as either

asys

system data structure (see ss2sys, sys2ss)

controller ถูกทำให้สำเร็จสำหรับระบบเวลาซึ่งต่อเนื่องกัน.

controller เป็นNOTทำให้สำเร็จสำหรับdiscrete ระบบเวลา.

nu

number of controlled inputs

ny

number of measured outputs

gmin

initial lower bound on H-infinity optimal gain

gmax

initial upper bound on H-infinity optimal gain

gtol

gain threshhold. Routine quits when gmax/gmin < 1+tol

ptol

poles with abs(real(pole)) < ptol*||H|| (H is appropriate Hamiltonian) are considered to be on the imaginary axis. Default: 1e-9

tol

threshhold for 0. Default: 200*eps

gmax, *min*, *tol*, and *tol* must all be postive scalars.

Outputs

k

system controller

g

designed gain value

gw

closed loop system

xinf

ARE solution matrix for regulator subproblem

yinf

ARE solution matrix for filter subproblem

Function File: [*retval*, *pc*, *pf*] = hinfsyn_chk (*a*, *b1*, *b2*, *c1*, *c2*, *d12*, *d21*, *g*, *ptol*)

เรียกโดยhinfsyn เพื่อพบถ้าสิ่งที่ได้รับg ทำให้เงื่อนไขพอใจในทฤษฎีบท 3 ในDoyle ,Glover ,Khargonekar ,.

การแก้ปัญหาช่องว่างหน้าที่ต่อมาตรฐานH2 และHinf ปัญหาการควบคุม.

Inputs as returned by is_dgkf, except for:

g

candidate gain level

ptol

as in hinfsyn

Outputs

retval

Function File: [*xinf*, *x_ha_err*] = hinfsyn_ric (*a*, *bb*, *c1*, *d1dot*, *r*, *ptol*)

Forms

```
xx = ([BB; -C1'*d1dot]/R) * [d1dot'*C1 BB'];
Ha = [A 0*A; -C1'*C1 -A'] - xx;
```

และแก้ไขเข้าร่วมRiccati สมการ. รหัสความผิดx_ha_err ชี้บอกหนึ่งในผู้ติดตามcondition

0

successful

1

xinf has imaginary eigenvalues

2

hx not Hamiltonian

3

xinf has infinite eigenvalues (numerical overflow)

4

xinf not symmetric

5

xinf not positive definite

6

r is singular

Function File: **[*k*, *p*, *e*] = lqe** ***(a, g, c, sigw, sigv, z)***

Function File: **[*k*, *p*, *e*] = lqe** ***(a, g, c, sigw, sigv, z)***

.

สร้างตามระยะยาวquadratic estimator (Kalman ตัวกรอง)สำหรับระบบเวลาซึ่งต่อเนื่องกัน.

```
dx
-- = a x + b u
dt

y = c x + d u
```

ที่w และv เป็นกระบวนการzero-mean gaussian เสียงรบกวนกับความเข้มข้นของแต่ละคน.

sigw = cov (w, w)

sigv = cov (v, v)

argumentซึ่งสิทธิในการเลือกz เป็นcross-covariance cov (w ,v). ถ้ามันถูกละเว้น,cov (w ,v)= 0 เป็น .

Observer structure is dz/dt = A z + B u + k (y - C z - D u)

The following values are returned:

k

The observer gain, (*a* - *kc*) is stable.

p

The solution of algebraic Riccati equation.

e

The vector of closed loop poles of (*a* - *kc*).

Function File: [*k*, *q1*, *p1*, *ee*, *er*] = lqg (*sys*, *sigw*, *sigv*, *q*, *r*, *in_idx*)

การออกแบบlinear-quadratic-gaussian ตัวควบคุมดีที่สุดสำหรับระบบ.

dx/dt = A x + B u + G w [w]=N(0,[Sigw 0])

y = C x + v [v] (0 Sigv])

หรือ

x(k+1) = A x(k) + B u(k) + G w(k) [w]=N(0,[Sigw 0])

y(k) = C x(k) + v(k) [v] (0 Sigv])

Inputs

sys

system data structure

sigw

sigv

intensities of independent Gaussian noise processes (as above)

q

r

state, control weighting respectively. Control ARE is

in_idx

names or indices of controlled inputs (see sysidx, listidx)

default: last dim(R) inputs are assumed to be controlled inputs, all others are assumed to be noise inputs.

Outputs

k

system data structure format LQG optimal controller (Obtain A,B,C matrices with sys2ss, sys2tf, or sys2zp as appropriate)

p1

Solution of control (state feedback) algebraic Riccati equation

q1

Solution of estimation algebraic Riccati equation

ee

estimator poles

es

controller poles

Function File: [*k*, *p*, *e*] = lqr *(a, b, q, r, z)*

สร้างตามระยะยาวquadratic ที่บังคับน้ำสำหรับระบบเวลาซึ่งต่อเนื่องกัน.

dx

-- = A x + B u

dt

ทำให้ค่าน้อยที่สุดงาน.

infinity

/

J = | x' Q x + u' R u

/

t=0

ละเว้น z หรือ

infinity

/

J = | x' Q x + u' R u + 2 x' Z u

/

t=0

z ประกอบด้วย.

ค่าซึ่งตามมาถูกคืน:

.. *k*

สิ่งที่ได้รับผลย้อนกลับหน้าที่,(- bk)มีเสถียรภาพและทำให้ค่าน้อยที่สุดงาน.

p

การแก้ปัญหาเสถียรของเกี่ยวกับพีชคณิตเหมาะสมRiccati สมการ.

e

เวกเตอร์ของloop polesซึ่งปิดแล้วของ(- bk).

Function File: **lsim** ***(sys, u, t, x0)***

ผลิตoutputสำหรับการทำตามระบบlinear.

Function File: place *(sys, p)*

คำนวณmatrixK นั่นสิ่งนั้นถ้าสภาพเป็นผลย้อนกลับกับสิ่งที่ได้รับK ,ดังนั้นeigenvalues . ของloop ซึ่งปิดแล้วระบบ(ฉัน). e. A-BK เหล่านั้นระบุในเวกเตอร์p .

Miscellaneous Functions (Not yet properly filed/documented)

@deftypefn{Function File}: axis2dlim *(axdata)*

ตัดสินใจขอบเขตเส้นศูนย์กลางสำหรับ2-d ข้อมูล(เวกเตอร์หลัก);ให้10% ขอบรอบแผนผัง.

อดขอบเข้าเกี่ยวกับ+/- 0 . 1 ถ้าข้อมูลเกี่ยวกับขนาดหนึ่ง(หรือจุดเดี่ยว).

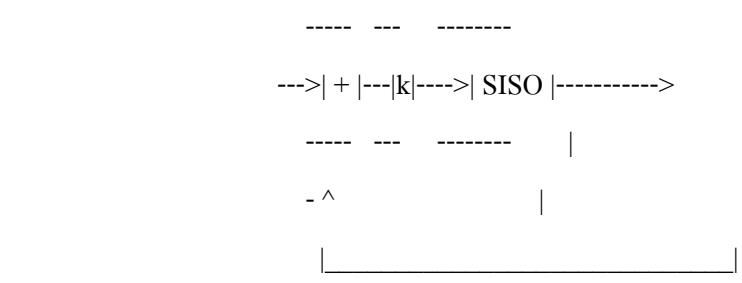
Inputs *axdata* nx2 matrix of data [x,y]

Outputs *axvec* vector of axis limits appropriate for call to axis() function

Function File: **rlocus** ***(inputs)***

[rldata, k] = rlocus(sys[,increment,min_k,max_k])

Displays root locus plot of the specified SISO system.



inputs: sys = system data structure

min_k, max_k,increment: minimum, maximum values of k and

the increment used in computing gain values

Outputs: plots the root locus to the screen.

rldata: Data points plotted column 1: real values, column 2: imaginary values)

k: gains for real axis break points.

Function File: sortcom ***(inputs)***

[yy,idx] = sortcom(xx[,opt]): sort a complex vector

xx: complex vector

opt: sorting option:

"re": real part (default)

"mag": by magnitude

"im": by imaginary part

if opt != "im" then complex conjugate pairs are grouped together,

a - jb followed by a + jb.

yy: sorted values

idx: permutation vector: yy = xx(idx)

Function File: ss2tf ***(inputs)***

[num,den] = ss2tf(a,b,c,d)

Conversion from tranfer function to state-space.

The state space system

.

x = Ax + Bu

y = Cx + Du

is converted to a transfer function

$$G(s)=\frac{num(s)}{den(s)}$$

used internally in system data structure format manipulations

Function File: **ss2zp** ***(inputs)***

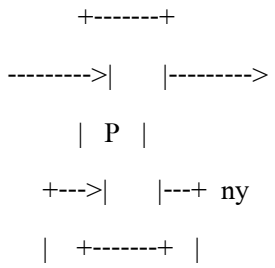
Converts a state space representation to a set of poles and zeros.

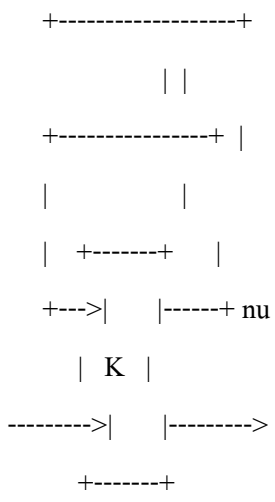
[pol,zer,k] = ss2zp(a,b,c,d) returns the poles and zeros of the state space system (a,b,c,d). K is a gain associated with the zeros.

used internally in system data structure format manipulations

Function File: **starp** ***(P, K, ny, nu)***

Redheffer star product or upper/lower LFT, respectively.





If ny and nu "consume" all inputs and outputs of K then the result is a lower fractional transformation. If ny and nu "consume" all inputs and outputs of P then the result is an upper fractional transformation.

ny and/or nu may be negative (= negative feedback)

Function File: **tf2ss** ***(inputs)***

Conversion from tranfer function to state-space.

The state space system

.

x = Ax + Bu

y = Cx + Du

is obtained from a transfer function

$$G(s)=\frac{num(s)}{den(s)}$$

via the function call [a,b,c,d] = tf2ss(num,den).

The vector 'den' must contain only one row, whereas the vector 'num'

may contain as many rows as there are outputs of the system 'y'.

The state space system matrices obtained from this function will be

in controllable canonical form as described in "Modern Control Theory",

[Brogan, 1991].

Function File: **tf2zp** ***(inputs)***

เปลี่ยนtransfer functions เป็น poles / zeros.

[zer,pol,k] = tf2zp(num,den)

Function File: **[*a*, *b*, *c*, *d*] = zp2ss** ***(zer, pol, k)***

เปลี่ยน from zero / pole เป็น state space

Function File: **[*num*, *den*] = zp2tf** ***(zer, pol, k)***

เปลี่ยน zeros / poles เป็น a transfer function.

Inputs

zer

pol

vectors of (possibly complex) poles and zeros of a transfer function. Complex values should appear in conjugate pairs

k

real scalar (leading coefficient)

[num,den] = zp2tf(zer,pol,k) forms the transfer function num/den from the vectors of poles and zeros.