

การสร้างฟังก์ชัน ทดสอบการใช้งานฟังก์ชัน และวิเคราะห์ผลการใช้งานฟังก์ชัน

จัดทำโดย

นาย บัณฑิต สีดาว 59010759 sec.1

เสนอ

อาจารย์จิระศักดิ์ สิทธิกร

รายงานนี้เป็นส่วนหนึ่งของวิชา

วิชา Engineering Mathematics 3

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 1 ปีการศึกษา 2560

**สารบัญ**

**เรื่อง** **หน้า**

คำนำ 3

ที่มาและความสำคัญ 4

ประโยชน์ที่ได้รับ 4

การสร้าง ฟังก์ชัน และ ทดสอบการใช้งานฟังก์ชัน และวิเคราะห์ผลการใช้งานฟังก์ชัน 5

1. ระหว่างฟังก์ชันใน Homework 1.5.4.1 และ Homework 1.6.6.1
   1. Source Code ของ Homework 1.5.4.1 5
   2. Result ของ Homework 1.5.4.1 8
   3. Source Code ของ Homework 16.6.1 9
   4. Result ของ Homework 16.6.1 11
   5. วิเคราะห์ผลการใช้งานระหว่าง Homework 1.5.4.1 กับ Homework 16.6.1 11
2. ระหว่างฟังก์ชันใน Homework 1.5.5.1 และ Homework 1.6.3.1
   1. Source Code ของ Homework 1.5.5.1  12
   2. Result ของ Homework 1.5.5.1  16
   3. Source Code ของ Homework 1.6.3.1 18
   4. Result ของ Homework 1.6.3.1 20
   5. วิเคราะห์ผลการใช้งานระหว่าง Homework 1.5.4.1 กับ Homework 16.6.1 20
3. ระหว่างฟังก์ชันใน Homework 4.3.2.5 และ ฟังก์ชันที่สร้างขึ้นเอง
   1. Source Code ของ Homework 4.3.2.5  21
   2. Result ของ Homework 4.3.2.5  25
   3. Source Code ของฟังก์ชั่นที่สร้างขึ้นเอง 27
   4. Result ของฟังก์ชั่นที่สร้างขึ้นเอง 28
   5. วิเคราะห์ผลการใช้งานระหว่าง Homework 1.5.4.1 กับฟังก์ชั่นที่สร้างขึ้นเอง 29

**คำนำ**

รายงานเล่มนี้่่ จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของวิชา 01006003 Engineering Mathematics 3 เพื่อให้ได้ศึกษาหาความรู้เกี่ยวกับการการสร้างฟังก์ชัน ทดสอบการใช้งานฟังก์ชัน และวิเคราะห์ผลการใช้งานฟังก์ชันต่างๆ โดยใช้โปรแกรมในการคำนวณพีชคณิตเชิงเส้น (Linear Algebra) ได้แก่ MATLAB และโปรแกรม Spark เพื่อเป็นประโยชน์กับการเรียน

ผู้จัดทำหวังว่า รายงานเล่มนี้จะเป็นประโยชน์กับผู้อ่าน หรือนักเรียน นักศึกษา ที่กำลังหาข้อมูลเรื่องนี้ หากมีข้อเสนอและข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้ด้วยประการทั้งปวง

บัณฑิต สีดาว

 วันที่ 9 ธันวาคม 2560

**ที่มาและความสำคัญ**

รายงานนี้เกิดจากทางผู้สอนอยากให้นักศึกษาได้นำความรู้ด้านการเขียนโปรแกรมมาผสานกับความรู้ทางคณิตศาสตร์จึงเกิดรายงานเล่มนี้ขึ้นมา

**ประโยชน์ที่ได้รับ**

1. ได้ฝึกทักษะการเขียนโปรแกรมที่เกี่ยวข้องกับความรู้ทางคณิตศาสตร์
2. ใช้เวลาว่างให้เกิดประโยชน์

**สร้าง ฟังก์ชัน และ ทดสอบการใช้งานฟังก์ชัน และวิเคราะห์ผลการใช้งานฟังก์ชัน**

1. ระหว่างฟังก์ชันใน Homework 1.5.4.1 และ Homework 1.6.6.1
   1. Source Code ของ Homework 1.5.4.1

laff\_axpy.m

function [y\_out] = laff\_axpy(a, x, y)

[m\_a, n\_a] = size(a);

[m\_x, n\_x] = size(x);

[m\_y, n\_y] = size(y);

if (m\_a ~= 1 & n\_a ~= 1)

y\_out = 'FAILED';

return

end

if ( m\_x \* n\_x ~= m\_y \* n\_y )

y\_out = 'FAILED';

return

end

if (n\_x == 1)

if (n\_y == 1)

for i=1:m\_x

y(i, 1) = a\*x(i, 1) + y(i, 1);

end

elseif (m\_y == 1)

for i=1:m\_x

y(1, i) = a\*x(i, 1) + y(1, i);

end

end

elseif (m\_x == 1)

if (m\_y == 1)

for i=1:n\_x

y(1, i) = a\*x(1, i) + y(1, i);

end

elseif (n\_y == 1)

for i=1:n\_x

y(i, 1) = a\*x(1, i) + y(i, 1);

end

end

end

y\_out = y;

end

test\_laff\_axpy.m

a = randi( [ -2, 2 ], 1 );

x = randi( [ -2, 2 ], 4, 1 );

y = randi( [ -2, 2 ], 4, 1 );

z = randi( [ -2, 2 ], 3, 1 );

disp( 'column - column axpy' )

if ( isequal( laff\_axpy\_59010759( a, x, y ), a \* x + y ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'column - row axpy' )

if ( isequal( laff\_axpy\_59010759( a, x, y' ), a \* x' + y' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'row - column axpy' )

if ( isequal( laff\_axpy\_59010759( a, x', y ), a \* x + y ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'row -> row axpy' )

if ( isequal( laff\_axpy\_59010759( a, x', y' ), a \* x' + y' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'column - column axpy (wrong size)' )

if ( isequal( laff\_axpy\_59010759( a, x, z ), 'FAILED' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'column - row axpy (wrong size)' )

if ( isequal( laff\_axpy\_59010759( a, x, z' ), 'FAILED' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'row -> column (wrong size)' )

if ( isequal( laff\_axpy\_59010759( a, x', z ), 'FAILED' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'row -> row axpy (wrong size)' )

if ( isequal( laff\_axpy\_59010759( a, x', z' ), 'FAILED' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

test\_time\_laff\_axpy.m

limit = 1000;

size = 10;

while size <= limit

tic

for time=1:limit

a = randi( [ -2, 2 ], 1 );

x = randi( [ -2, 2 ], size, 1 );

y = randi( [ -2, 2 ], size, 1 );

laff\_axpy\_59010759( a, x, y );

end

fprintf('Size : %5d, 1,000 times : ', size);

toc

size = 10\*size;

end

* 1. Result ของ Homework 1.5.4.1

ผลการทดสอบของความถูกต้องของฟังก์ชั่น

>> test\_axpy\_59010759

column - column axpy

TEST PASSED

column - row axpy

TEST PASSED

row - column axpy

TEST PASSED

row -> row axpy

TEST PASSED

column - column axpy (wrong size)

TEST PASSED

column - row axpy (wrong size)

TEST PASSED

row -> column (wrong size)

TEST PASSED

row -> row axpy (wrong size)

TEST PASSED

ผลการทดสอบประสิทธิภาพของฟังก์ชั่น

>> test\_time\_laff\_axpy\_59010759

Size : 10, 1,000 times : Elapsed time is 0.199105 seconds.

Size : 100, 1,000 times : Elapsed time is 0.137768 seconds.

Size : 1000, 1,000 times : Elapsed time is 0.234531 seconds.

* 1. Source Code ของ Homework 16.6.1

Axpy\_unb.m

% Copyright 2017 The University of Texas at Austin

%

% For licensing information see

% http://www.cs.utexas.edu/users/flame/license.html

%

% Programmed by: Name of author

% Email of author

function [ y\_out ] = Axpy\_unb( alpha, x, y )

[ xT, ...

xB ] = FLA\_Part\_2x1( x, ...

0, 'FLA\_TOP' );

[ yT, ...

yB ] = FLA\_Part\_2x1( y, ...

0, 'FLA\_TOP' );

while ( size( xT, 1 ) < size( x, 1 ) )

[ x0, ...

chi1, ...

x2 ] = FLA\_Repart\_2x1\_to\_3x1( xT, ...

xB, ...

1, 'FLA\_BOTTOM' );

[ y0, ...

psi1, ...

y2 ] = FLA\_Repart\_2x1\_to\_3x1( yT, ...

yB, ...

1, 'FLA\_BOTTOM' );

%------------------------------------------------------------%

psil = alpha \* chi1 + psi1;

%------------------------------------------------------------%

[ xT, ...

xB ] = FLA\_Cont\_with\_3x1\_to\_2x1( x0, ...

chi1, ...

x2, ...

'FLA\_TOP' );

[ yT, ...

yB ] = FLA\_Cont\_with\_3x1\_to\_2x1( y0, ...

psi1, ...

y2, ...

'FLA\_TOP' );

end

y\_out = [ yT

yB ];

return

test\_Axpy\_unb.m

a = randi( [ -2, 2 ], 1 );

x = randi( [ -2, 2 ], 4, 1 );

y = randi( [ -2, 2 ], 4, 1 );

disp( 'axpy' )

if ( isequal( Axpy\_unb\_59010759( a, x, y ), a \* x + y ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

test\_time\_\_Axpy\_unb.m

limit = 1000;

size = 10;

while size <= limit

tic

for time=1:limit

a = randi( [ -2, 2 ], 1 );

x = randi( [ -2, 2 ], size, 1 );

y = randi( [ -2, 2 ], size, 1 );

Axpy\_unb\_59010759( a, x, y );

end

fprintf('Size : %5d, 1,000 times : ', size);

toc

size = 10\*size;

end

* 1. Result ของ Homework 16.6.1

ผลการทดสอบของความถูกต้องของฟังก์ชั่น

>> test\_Axpy\_unb\_59010759

axpy

TEST PASSED

ผลการทดสอบประสิทธิภาพของฟังก์ชั่น

>> test\_time\_Axpy\_unb\_59010759

Size : 10, 1,000 times : Elapsed time is 6.398407 seconds.

Size : 100, 1,000 times : Elapsed time is 28.132483 seconds.

* 1. วิเคราะห์ผลการใช้งานระหว่าง Homework 1.5.4.1 กับ Homework 16.6.1

ฟังก์ชัน laff\_axpy ใช้เวลาน้อยกว่าฟังก์ชัน Axpy\_unb เมื่อ row เพิ่มขึ้นเรื่อยๆ อันเนื่องจาการ Axpy\_unb นั้นต้องใช้เวลาในการแบ่งเมทริกซ์ ยิ่งมีขนาดมากจะยิ่งแบ่งเยอะขึ้น

1. ระหว่างฟังก์ชันใน Homework 1.5.5.1 และ Homework 1.6.3.1
   1. Source Code ของ Homework 1.5.5.1

laff\_dot.m

function [a] = laff\_dot(x, y)

[m\_x, n\_x] = size(x);

[m\_y, n\_y] = size(y);

if (m\_x ~= 1 & n\_x ~= 1)

a = 'FAILED';

return

end

if ( m\_x \* n\_x ~= m\_y \* n\_y )

a = 'FAILED';

return

end

a = 0;

if (n\_x == 1)

if (n\_y == 1)

for i=1:m\_x

a = a + x(i, 1) \* y(i, 1);

end

else

for i=1:m\_x

a = a + x(i, 1) \* y(1, i);

end

end

else

if (n\_y == 1)

for i=1:n\_x

a = a + x(1, i) \* y(i, 1);

end

else

for i=1:n\_x

a = a + x(1, i) \* y(1, i);

end

end

end

end

test\_ laff\_dot.m

x = randi( [ -2, 2 ], 4, 1 );

y = randi( [ -2, 2 ], 4, 1 );

a = randi( [ -2, 2 ], 1 );

z = randi( [ -2, 2 ], 3, 1 );

disp( 'dot product of column vector with column vector' )

if ( isequal( laff\_dot\_59010759( x, y ), x' \* y ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'dot product of column vector with row vector' )

if ( isequal( laff\_dot\_59010759( x, y' ), x' \* y ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'dot product of row vector with column vector' )

if ( isequal( laff\_dot\_59010759( x', y ), x' \* y ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'dot product of row vector with row vector' )

if ( isequal( laff\_dot\_59010759( x', y' ), x' \* y ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'dot product of column vector with column vector (wrong size)' )

if ( isequal( laff\_dot\_59010759( x, z ), 'FAILED' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'dot product of column vector with row vector (wrong size)' )

if ( isequal( laff\_dot\_59010759( x, z' ), 'FAILED' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'dot product of row vector with column vector (wrong size)' )

if ( isequal( laff\_dot\_59010759( x', z ), 'FAILED' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'dot product of row vector with row vector (wrong size)' )

if ( isequal( laff\_dot\_59010759( x', z' ), 'FAILED' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'dot product of column vector with alpha (wrong size)' )

if ( isequal( laff\_dot\_59010759( x, a ), 'FAILED' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'dot product of alpha with column vector (wrong size)' )

if ( isequal( laff\_dot\_59010759( a, x ), 'FAILED' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'dot product of row vector with alpha (wrong size)' )

if ( isequal( laff\_dot\_59010759( x', a ), 'FAILED' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

disp( ' ' )

disp( 'dot product of alpha with row vector (wrong size)' )

if ( isequal( laff\_dot\_59010759( a, x' ), 'FAILED' ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

test\_time\_ laff\_dot.m

limit = 1000;

size = 10;

while size <= limit

tic

for time=1:limit

x = randi( [ -2, 2 ], size, 1 );

y = randi( [ -2, 2 ], size, 1 );

laff\_dot\_59010759( x, y );

end

fprintf('Size : %5d, 1,000 times : ', size);

toc

size = 10\*size;

end

* 1. Result ของ Homework 1.5.5.1

ผลการทดสอบของความถูกต้องของฟังก์ชั่น

>> test\_laff\_dot\_59010759

dot product of column vector with column vector

TEST PASSED

dot product of column vector with row vector

TEST PASSED

dot product of row vector with column vector

TEST PASSED

dot product of row vector with row vector

TEST PASSED

dot product of column vector with column vector (wrong size)

TEST PASSED

dot product of column vector with row vector (wrong size)

TEST PASSED

dot product of row vector with column vector (wrong size)

TEST PASSED

dot product of row vector with row vector (wrong size)

TEST PASSED

dot product of column vector with alpha (wrong size)

TEST PASSED

dot product of alpha with column vector (wrong size)

TEST PASSED

dot product of row vector with alpha (wrong size)

TEST PASSED

dot product of alpha with row vector (wrong size)

TEST PASSED

ผลการทดสอบประสิทธิภาพของฟังก์ชั่น

>> test\_time\_laff\_dot\_59010759

Size : 10, 1,000 times : Elapsed time is 0.161985 seconds.

Size : 100, 1,000 times : Elapsed time is 0.101619 seconds.

Size : 1000, 1,000 times : Elapsed time is 0.218669 seconds.

* 1. Source Code ของ Homework 1.6.3.1

Dot\_unb.m

% Copyright 2017 The University of Texas at Austin

%

% For licensing information see

% http://www.cs.utexas.edu/users/flame/license.html

%

% Programmed by: Name of author

% Email of author

function [ alpha\_out ] = Dot\_unb( alpha, x, y )

[ xT, ...

xB ] = FLA\_Part\_2x1( x, ...

0, 'FLA\_TOP' );

[ yT, ...

yB ] = FLA\_Part\_2x1( y, ...

0, 'FLA\_TOP' );

while ( size( xT, 1 ) < size( x, 1 ) )

[ x0, ...

chi1, ...

x2 ] = FLA\_Repart\_2x1\_to\_3x1( xT, ...

xB, ...

1, 'FLA\_BOTTOM' );

[ y0, ...

psi1, ...

y2 ] = FLA\_Repart\_2x1\_to\_3x1( yT, ...

yB, ...

1, 'FLA\_BOTTOM' );

%------------------------------------------------------------%

alpha = chi1 \* psi1 + alpha;

%------------------------------------------------------------%

[ xT, ...

xB ] = FLA\_Cont\_with\_3x1\_to\_2x1( x0, ...

chi1, ...

x2, ...

'FLA\_TOP' );

[ yT, ...

yB ] = FLA\_Cont\_with\_3x1\_to\_2x1( y0, ...

psi1, ...

y2, ...

'FLA\_TOP' );

end

alpha\_out = alpha;

return

test\_Dot\_unb.m

a = randi( [ -2, 2 ], 1 );

x = randi( [ -2, 2 ], 4, 1 );

y = randi( [ -2, 2 ], 4, 1 );

disp( 'dot product' )

if ( isequal( Dot\_unb\_59010759( a, x, y ), a + x' \* y ) )

disp( 'TEST PASSED' )

else

disp( 'TEST FAILED' )

end

test\_time\_Dot\_unb.m

limit = 1000;

size = 10;

while size <= limit

tic

for time=1:limit

x = randi( [ -2, 2 ], size, 1 );

y = randi( [ -2, 2 ], size, 1 );

Dot\_unb\_59010759( 0, x, y );

end

fprintf('Size : %5d, 1,000 times : ', size);

toc

size = 10\*size;

end

* 1. Result ของ Homework 1.6.3.1

ผลการทดสอบของความถูกต้องของฟังก์ชั่น

test\_Dot\_unb\_59010759

dot product

TEST PASSED

ผลการทดสอบประสิทธิภาพของฟังก์ชั่น

>> test\_time\_Dot\_unb\_59010759

Size : 10, 1,000 times : Elapsed time is 3.266834 seconds.

Size : 100, 1,000 times : Elapsed time is 27.839730 seconds.

* 1. วิเคราะห์ผลการใช้งานระหว่าง Homework 1.5.4.1 กับ Homework 16.6.1

ฟังก์ชัน laff\_dot ใช้เวลาน้อยกว่าฟังก์ชัน Dot\_unb เมื่อ row ของ vector เพิ่มขึ้นเรื่อยๆ อันเนื่องจาการ Dot\_unb นั้นต้องใช้เวลาในการแบ่งเมทริกซ์ ยิ่งมีขนาดมากจะยิ่งแบ่งเยอะขึ้น

1. ระหว่างฟังก์ชันใน Homework 4.3.2.3 และฟังก์ชั่นที่สร้างเอง
   1. Source Code ของ Homework 4.3.2.5

Trmvp\_ln\_unb\_var1.m

function [ y\_out ] = Trmvp\_ln\_unb\_var1( L, x, y )

[ LTL, LTR, ...

LBL, LBR ] = FLA\_Part\_2x2( L, ...

0, 0, 'FLA\_TL' );

[ xT, ...

xB ] = FLA\_Part\_2x1( x, ...

0, 'FLA\_TOP' );

[ yT, ...

yB ] = FLA\_Part\_2x1( y, ...

0, 'FLA\_TOP' );

while ( size( LTL, 1 ) < size( L, 1 ) )

[ L00, l01, L02, ...

l10t, lambda11, l12t, ...

L20, l21, L22 ] = FLA\_Repart\_2x2\_to\_3x3( LTL, LTR, ...

LBL, LBR, ...

1, 1, 'FLA\_BR' );

[ x0, ...

chi1, ...

x2 ] = FLA\_Repart\_2x1\_to\_3x1( xT, ...

xB, ...

1, 'FLA\_BOTTOM' );

[ y0, ...

psi1, ...

y2 ] = FLA\_Repart\_2x1\_to\_3x1( yT, ...

yB, ...

1, 'FLA\_BOTTOM' );

%------------------------------------------------------------%

psi1 = laff\_dot\_59010759( l10t, x0 ) + psi1;

psi1 = laff\_dot\_59010759( lambda11, chi1 ) + psi1;

%------------------------------------------------------------%

[ LTL, LTR, ...

LBL, LBR ] = FLA\_Cont\_with\_3x3\_to\_2x2( L00, l01, L02, ...

l10t, lambda11, l12t, ...

L20, l21, L22, ...

'FLA\_TL' );

[ xT, ...

xB ] = FLA\_Cont\_with\_3x1\_to\_2x1( x0, ...

chi1, ...

x2, ...

'FLA\_TOP' );

[ yT, ...

yB ] = FLA\_Cont\_with\_3x1\_to\_2x1( y0, ...

psi1, ...

y2, ...

'FLA\_TOP' );

end

y\_out = [ yT

yB ];

return

test\_Trmvp\_ln\_unb\_var1.m

A = randi( [ -2, 2 ], 4, 4 )

x = randi( [ -2, 2 ], 4, 1 )

y = randi( [ -2, 2 ], 4, 1 )

if ( isequal( Trmvp\_ln\_unb\_var1\_59010759( A, x, y ), tril( A ) \* x + y ) )

disp( 'Trmvp\_ln\_unb\_var1 appears to be correct' )

else

disp( 'Trmvp\_ln\_unb\_var1 has a problem' )

end

test\_time\_ Trmvp\_ln\_unb\_var1.m

limit = 1000;

size = 10;

while size <= limit

tic

for time=1:limit

A = randi( [ -2, 2 ], size, size );

x = randi( [ -2, 2 ], size, 1 );

y = randi( [ -2, 2 ], size, 1 );

Trmvp\_ln\_unb\_var1\_59010759( A, x, y );

end

fprintf('Size : %5d, 1,000 times : ', size);

toc

size = 10\*size;

end

Trmvp\_ln\_unb\_var2.m

function [ y\_out ] = Trmvp\_ln\_unb\_var2( L, x, y )

[ LTL, LTR, ...

LBL, LBR ] = FLA\_Part\_2x2( L, ...

0, 0, 'FLA\_TL' );

[ xT, ...

xB ] = FLA\_Part\_2x1( x, ...

0, 'FLA\_TOP' );

[ yT, ...

yB ] = FLA\_Part\_2x1( y, ...

0, 'FLA\_TOP' );

while ( size( LTL, 1 ) < size( L, 1 ) )

[ L00, l01, L02, ...

l10t, lambda11, l12t, ...

L20, l21, L22 ] = FLA\_Repart\_2x2\_to\_3x3( LTL, LTR, ...

LBL, LBR, ...

1, 1, 'FLA\_BR' );

[ x0, ...

chi1, ...

x2 ] = FLA\_Repart\_2x1\_to\_3x1( xT, ...

xB, ...

1, 'FLA\_BOTTOM' );

[ y0, ...

psi1, ...

y2 ] = FLA\_Repart\_2x1\_to\_3x1( yT, ...

yB, ...

1, 'FLA\_BOTTOM' );

%------------------------------------------------------------%

psi1 = laff\_axpy\_59010759( chi1 , lambda11 , psi1 );

y2 = laff\_axpy\_59010759( chi1 , l21 , y2 );

%------------------------------------------------------------%

[ LTL, LTR, ...

LBL, LBR ] = FLA\_Cont\_with\_3x3\_to\_2x2( L00, l01, L02, ...

l10t, lambda11, l12t, ...

L20, l21, L22, ...

'FLA\_TL' );

[ xT, ...

xB ] = FLA\_Cont\_with\_3x1\_to\_2x1( x0, ...

chi1, ...

x2, ...

'FLA\_TOP' );

[ yT, ...

yB ] = FLA\_Cont\_with\_3x1\_to\_2x1( y0, ...

psi1, ...

y2, ...

'FLA\_TOP' );

end

y\_out = [ yT

yB ];

return

test\_Trmvp\_ln\_unb\_var2.m

A = randi( [ -2, 2 ], 4, 4 )

x = randi( [ -2, 2 ], 4, 1 )

y = randi( [ -2, 2 ], 4, 1 )

if ( isequal( Trmvp\_ln\_unb\_var2\_59010759( A, x, y ), tril( A ) \* x + y ) )

disp( 'Trmvp\_ln\_unb\_var2 appears to be correct' )

else

disp( 'Trmvp\_ln\_unb\_var2 has a problem' )

end

test\_time\_ Trmvp\_ln\_unb\_var2.m

limit = 1000;

size = 10;

while size <= limit

tic

for time=1:limit

A = randi( [ -2, 2 ], size, size );

x = randi( [ -2, 2 ], size, 1 );

y = randi( [ -2, 2 ], size, 1 );

Trmvp\_ln\_unb\_var2\_59010759( A, x, y );

end

fprintf('Size : %5d, 1,000 times : ', size);

toc

size = 10\*size;

end

* 1. Result ของ Homework 4.3.2.5

ผลการทดสอบของความถูกต้องของฟังก์ชั่น

>> test\_Trmvp\_ln\_unb\_var1\_59010759

A =

-2 2 -2 -1

-2 -2 -1 0

0 0 -2 -2

1 0 1 1

x =

-1

1

1

1

y =

0

-2

-1

2

Trmvp\_ln\_unb\_var1 appears to be correct

>> test\_Trmvp\_ln\_unb\_var2\_59010759

A =

2 -2 -2 -2

0 -2 2 -1

0 -2 1 0

-1 0 -2 -2

x =

-2

1

-1

1

y =

0

0

-2

-2

Trmvp\_ln\_unb\_var2 appears to be correct

ผลการทดสอบประสิทธิภาพของฟังก์ชั่น

>> test\_time\_Trmvp\_ln\_unb\_var1\_59010759

Size : 10, 1,000 times : Elapsed time is 7.219494 seconds.

Size : 100, 1,000 times : Elapsed time is 96.388200 seconds.

>> test\_time\_Trmvp\_ln\_unb\_var2\_59010759

Size : 10, 1,000 times : Elapsed time is 9.325294 seconds.

Size : 100, 1,000 times : Elapsed time is 76.779085 seconds.

* 1. Source Code ของฟังก์ชั่นที่สร้างขึ้นเอง

test\_Trmvp\_ln\_unb\_var2.m

function [ y\_out ] = Trmvp\_ln( L, x, y )

[m\_L, n\_L] = size(L);

[m\_x, n\_x] = size(x);

[m\_y, n\_y] = size(y);

if (m\_L ~= n\_L) & ( m\_x \* n\_x ~= m\_y \* n\_y )

y\_out = 'FAILED';

return

end

for i=1:m\_L

for j=1:i

y(i) = L(i, j) \* x(j) + y(i);

end

end

y\_out = y

end

test\_Trmvp\_ln.m

A = randi( [ -2, 2 ], 4, 4 );

x = randi( [ -2, 2 ], 4, 1 );

y = randi( [ -2, 2 ], 4, 1 );

if ( isequal( Trmvp\_ln\_59010759( A, x, y ), tril( A ) \* x + y ) )

disp( 'Trmvp\_ln appears to be correct' )

else

disp( 'Trmvp\_ln has a problem' )

end

test\_time\_ Trmvp\_ln.m

limit = 1000;

size = 10;

while size <= limit

tic

for time=1:limit

A = randi( [ -2, 2 ], size, size );

x = randi( [ -2, 2 ], size, 1 );

y = randi( [ -2, 2 ], size, 1 );

Trmvp\_ln\_59010759( A, x, y );

end

fprintf('Size : %5d, 1,000 times : ', size);

toc

size = 10\*size;

end

* 1. Result ของฟังก์ชั่นที่สร้างขึ้นเอง

ผลการทดสอบของความถูกต้องของฟังก์ชั่น

>> test\_Trmvp\_ln\_59010759

A =

-1 2 -1 -2

0 2 -1 -2

-2 0 2 1

-2 -2 -2 1

x =

1

0

0

-1

y =

1

-2

1

-2

Trmvp\_ln appears to be correct

ผลการทดสอบประสิทธิภาพของฟังก์ชั่น

>> test\_time\_Trmvp\_ln\_59010759

Size : 10, 1,000 times : Elapsed time is 0.208559 seconds.

Size : 100, 1,000 times : Elapsed time is 0.985248 seconds.

* 1. วิเคราะห์ผลการใช้งานระหว่าง Homework 1.5.4.1 กับ Homework 16.6.1

ในตอนแรกฟังก์ชั่น Trmvp\_ln\_unb\_var1 จะเร็วกว่า Trmvp\_ln\_unb\_var2 เมื่อข้อมูลอยู่ในช่วงท่น้อยกว่า 20 แถวโดยประมาณ แล้วหลังจากนั้น Trmvp\_ln\_unb\_var2 จะเร็วกว่าอย่างเห็นได้ชัดเนื่องจาก ฟังก์ชั่น laff\_axpy ใน Trmvp\_ln\_unb\_var2 นั้นทำงานเร็วกว่าฟังก์ชั่น laff\_dot ใน Trmvp\_ln\_unb\_var1

ส่วนฟังก์ชั่นที่สร้างขึ้นมาเองนั้นเร็วกว่า Trmvp\_ln\_unb\_var1 และ Trmvp\_ln\_unb\_var2 เนื่องจากไม่ต้องเสียเวลาในการแบ่งเมทริกซ์ และการใช้ loop วนนั้นทำได้เร็วกว่า