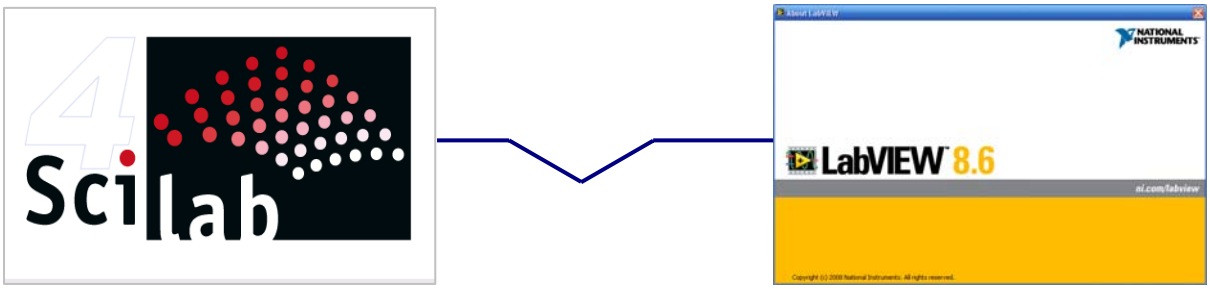
Introduction

SCILAB with LabVIEW GATEWAY



SCILAB เป็นโปรแกรมที่พัฒนาโดยกลุ่มของนักวิจัยของประเทศฝรั่งเศส โดยมีจุดมุ่งหมายเพื่อใช้ในการคำนวณเชิงตัวเลขและแสดงผลกราฟิกที่ซับซ้อนเช่นเดียวกับโปรแกรม MATLAB แต่โปรแกรม SCILAB เป็นโปรแกรมที่ให้ฟรี (freeware) ง่ายต่อการเรียนรู้และทำความเข้าใจ รวมทั้งมีประสิทธิภาพใกล้เคียงกับโปรแกรม MATLAB ซึ่งในปัจจุบันนี้จะพบว่า มหาวิทยาลัยหลายๆ แห่งในต่างประเทศได้เริ่มมีการนำโปรแกรม SCILAB ไปช่วยในการเรียนการสอนและทำงานวิจัย

LabVIEW ย่อมาจาก Laboratory Virtual Instrument Engineering Workbench เป็นโปรแกรมคอมพิวเตอร์ที่สร้างเพื่อนำมาใช้ในด้านการจัดการวัดและเครื่องมือวัดสำหรับงานทางวิศวกรรม ซึ่งเป็นโปรแกรมประเภท GUI (Graphic User Interface) นั้น คือผู้ใช้พัฒนาโปรแกรมไม่จำเป็นต้องเขียน code หรือคำสั่งใดๆ ทั้งสิ้น และภาษาที่ใช้ในโปรแกรมจะเรียกว่าเป็น ภาษารูปภาพ หรือเรียกอีกอย่างว่าภาษา G (Graphical Language) ซึ่งจะแทนการเขียนโปรแกรมเป็นบรรทัดภาษาพื้นฐาน เช่น C, BASIC หรือ FORTRAN ด้วยรูปภาพหรือสัญลักษณ์ทั้งหมด โดยจะช่วยอำนวยความสะดวกและสามารถลดเวลาในการเขียนโปรแกรมลงไปได้มาก โดยเฉพาะในงานเขียนโปรแกรมคอมพิวเตอร์ เพื่อเชื่อมต่อกับอุปกรณ์ภายนอก เช่น Port หรือ Card ต่างๆ รวมถึงการจัดวางตำแหน่งในหน่วยความจำเพื่อที่จะสามารถรวบรวมข้อมูลมาใช้ในการคำนวณและเก็บข้อมูลให้ได้ประโยชน์สูงสุด

สำหรับคู่มือเล่มนี้เหมาะสำหรับผู้ที่มีพื้นฐานโปรแกรม SCILAB และโปรแกรม LabVIEW มาบ้าง โดยผู้เขียนจะกล่าวถึงการใช้งานโปรแกรม LabVIEW 8.6 ร่วมกับโปรแกรม SCILAB 4.1.2 ซึ่งจะมีไลบรารีชื่อว่า ScriptScilab.dll ที่ทาง National Instruments และ INRIA จัดทำขึ้นมา และได้ให้ดาวน์โหลดตัวอย่างเพื่อติดตั้ง Scilab - LabVIEW Gateway เวอร์ชัน 1.1 แบบฟรี (Open source) โดยสามารถดาวน์โหลดได้ที่ http://www.scilab.org/products/other/labview_gateway

Introduction

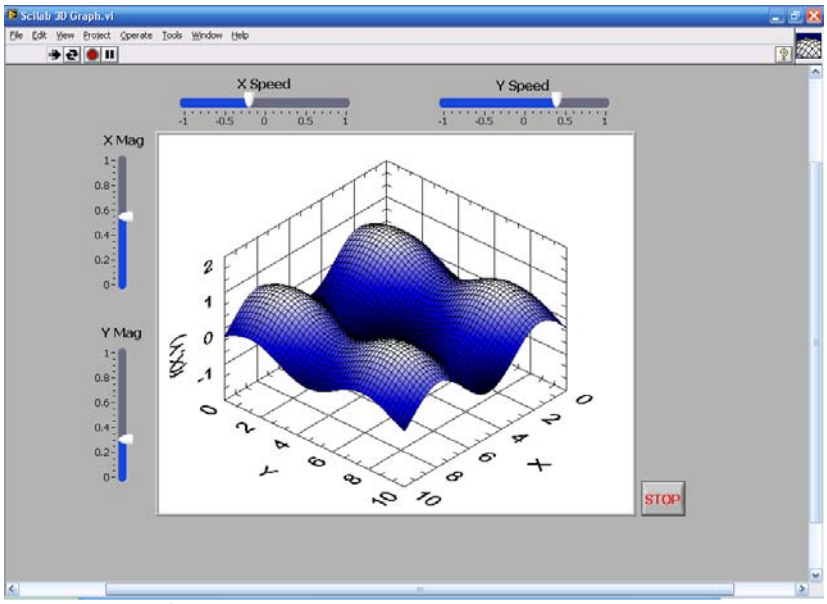
BASIC LabVIEW

สำหรับโปรแกรมที่พัฒนาขึ้นโดย LabVIEW จะเรียกว่า Virtual Instrument (VI) เพราะลักษณะที่ปรากฏทางจอภาพเมื่อเริ่มใช้งานจะเหมือนกับเครื่องมือหรืออุปกรณ์ทางวิศวกรรม ในขณะเดียวกันหลังฉากของอุปกรณ์เสมือนจริงเหล่านั้นจะเป็นการทำงานของฟังก์ชันต่างๆ ซึ่งในหนึ่ง Virtual Instrument (VI) จะประกอบด้วยส่วนประกอบ 3 ส่วนคือ

1. Front Panel
2. Block Diagram
3. Icon และ Connector

ทั้ง 3 ส่วนนี้จะประกอบกันขึ้นมาเป็นอุปกรณ์เสมือนจริง ตามลักษณะและหน้าที่ของส่วนประกอบทั้ง 3 โดยมีรายละเอียดดังต่อไปนี้

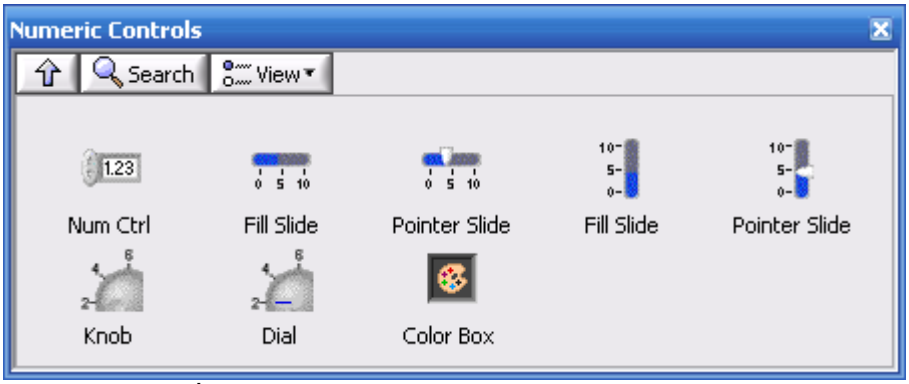
1. Front Panel หรือหน้าปัทม์ จะเป็นส่วนที่ใช้สื่อความกันระหว่างผู้ใช้กับโปรแกรม (หรือที่นิยมเรียก user interface) โดยทั่วไปจะมีลักษณะเหมือนกับหน้าปัทม์ของของเครื่องมือหรืออุปกรณ์ที่ใช้งานด้านการวัดทั่วๆ ไป เช่น มีสวิตซ์ปิดเปิด, ปุ่มบิด, ปุ่มกด จอแสดงผล ดังนั้น Front Panel นี้จึงเปรียบเสมือนเป็น GUI ของโปรแกรมหรือ VI นั่นเอง ลักษณะของ Front Panel แสดงดังภาพที่ 1.1



ภาพที่ 1.1 แสดง Front Panel ของโปรแกรม LabVIEW

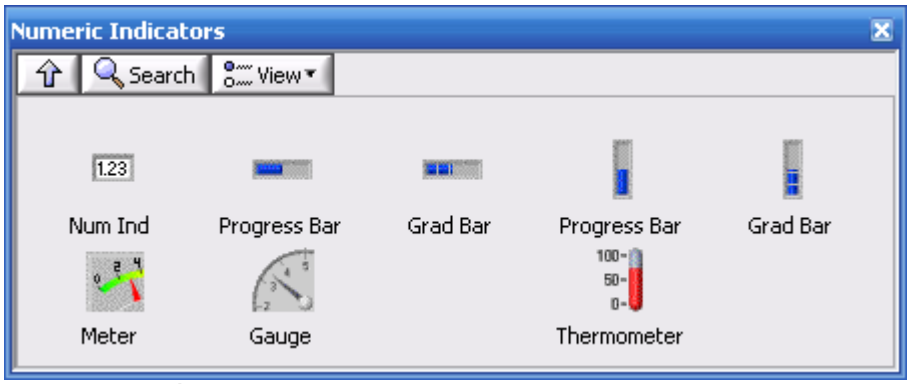
สำหรับหน้าปัทม์ (Front Panel) ของ LabVIEW จะมีส่วนประกอบที่สำคัญ 2 แบบ คือ ตัวควบคุม (Controls) และ ตัวแสดงผล (Indicator) ซึ่งส่วนประกอบทั้ง 2 จะมีการทำงานต่างกันและหน้าที่ตรงกันข้ามกัน ดังมีรายละเอียดต่อไปนี้

1.1 Controls มีหน้าที่เป็นตัวควบคุม คือการใส่ค่า (Input) จากผู้ใช้ ลักษณะของ Controls เช่น ปุ่มปรับค่า, สะพานปิด – เปิดไฟ, แท่งเลื่อนเพื่อปรับค่า, การให้ค่าด้วยตัวเลข Digital หรืออื่นๆ ดังนั้น Controls คือการกำหนดค่าหรือแหล่ง (source) ข้อมูล แสดงตัวอย่าง Controls ดังภาพที่ 1.2



ภาพที่ 1.2 แสดงตัวอย่าง Control ของโปรแกรม LabVIEW

1.2 Indicators มีหน้าที่เป็นตัวแสดงผลเพียงอย่างเดียวโดยจะรับค่าที่ได้จากแหล่งข้อมูลมาแสดงผลซึ่งอาจปรากฏในรูปของกราฟ, เข็มชี้, ระดับของเหลว หรืออื่นๆ Indicators นี้เปรียบเสมือน output เพื่อให้ผู้ใช้ได้ทราบค่าสิ่งที่เรากำลังวิเคราะห์อยู่ และผู้ใช้งานจะไม่สามารถปรับค่าต่างๆ บน Indicators ได้ โดยตรงแต่จะต้องมีแหล่งข้อมูลที่ส่งให้กับ Indicators เหล่านี้ แสดงตัวอย่าง Indicators ในภาพที่ 1.3

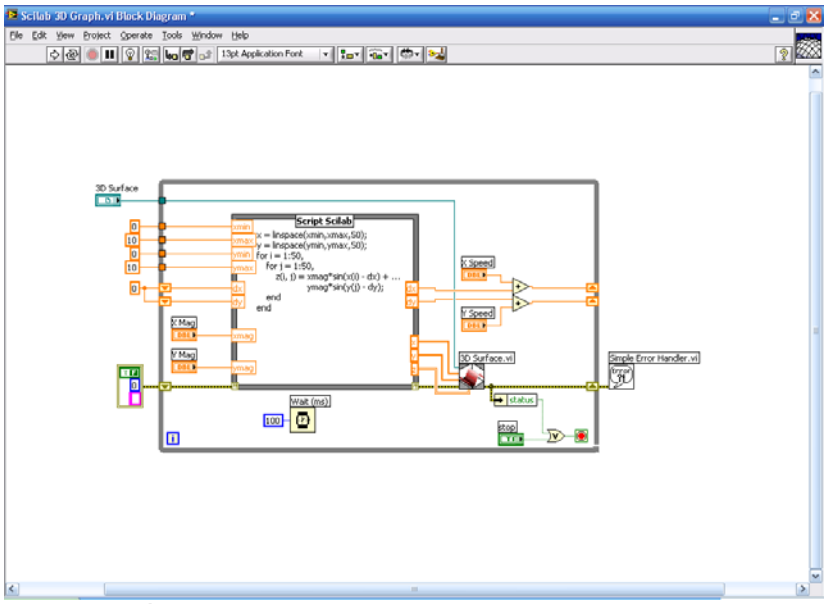


ภาพที่ 1.3 แสดงตัวอย่าง Indicators ของโปรแกรม LabVIEW

2. Block Diagram เป็นเสมือนกับ Source Code ของโปรแกรมที่พัฒนาขึ้นด้วย LabVIEW ซึ่งปรากฏอยู่ในรูปของภาษา G โดย Block Diagram นี้จะถือว่าเป็น Executable Program คือสามารถที่จะทำงานได้ทันที และข้อดีอีกประการหนึ่งก็คือ LabVIEW จะมีการตรวจสอบข้อผิดพลาดของโปรแกรมตลอดเวลา

ทำให้โปรแกรมจะทำงานได้ก็ต่อเมื่อไม่มีข้อผิดพลาดในโปรแกรมเท่านั้นโดยผู้เขียนโปรแกรมสามารถที่จะดูรายละเอียดของข้อผิดพลาดแสดงให้เห็นได้ตลอดเวลา ทำให้การเขียนโปรแกรมนั้นง่ายขึ้น

ส่วนประกอบภายใน Block Diagram จะประกอบด้วย ฟังก์ชัน ค่าคงที่ โปรแกรมควบคุมการทำงานหรือโครงสร้าง จากนั้นในแต่ละส่วนเหล่านี้ ซึ่งจะปรากฏอยู่ในรูปของ Block เราจะได้รับการต่อสาย (wire) สำหรับ Block ที่เหมาะสมเข้าด้วยกัน เพื่อกำหนดลักษณะการไหลของข้อมูลระหว่าง block เหล่านั้น ทำให้ข้อมูลได้รับการประมวลผลตามที่ต้องการ แสดง Block Diagram ดังภาพที่ 1.4



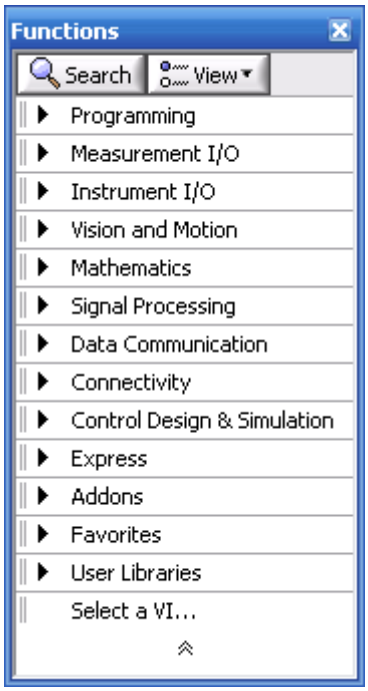
ภาพที่ 1.4 แสดง Block Diagram ของโปรแกรม LabVIEW

สำหรับ Block Diagram จะมีส่วนที่ทำหน้าที่หลักคือการควบคุมการส่งผ่านหรือเราอาจเรียกว่าการไหลของข้อมูล (Data Flow) และกำหนดถึงวิธีการประมวลผลข้อมูลมี 4 ส่วนดังนี้

2.1 Terminal ทุกครั้งที่เราสร้าง Control หรือ Indicator บน Front Panel ใน Window ของ Block Diagram จะปรากฏ Terminal ขึ้น Terminal ก็คล้ายกับสถานีของข้อมูลคือจะเป็นทั้งสถานีต้นทางของข้อมูลถ้า Terminal นั้นเป็น Terminal ของ Controls และขณะเดียวกันจะเป็นทั้งสถานีปลายทางของข้อมูลถ้า Terminal นั้นเป็น Terminal ของ Indicator แสดงตัวอย่างดังภาพที่ 1.6

2.2 Node เป็นจุดต่อบนบล็อกไดอะแกรม แสดงเป็นแบบอินพุต และ/หรือ เอาท์พุต จะทำงานเมื่อสั่ง RUN VI แสดงตัวอย่าง Node ดังภาพที่ 1.6

2.3 Functions คือตัวดำเนินการต่างๆ ที่สำเร็จรูปเช่น sine, cos, tan, log เป็นต้น ซึ่งสามารถเรียกแสดง Functions Palette นี้ได้โดยการคลิกขวาพื้นที่วาง ๆ ในฝั่ง Block Diagram จะแสดง Functions Palette ดังภาพที่ 1.5 และภาพที่ 1.6

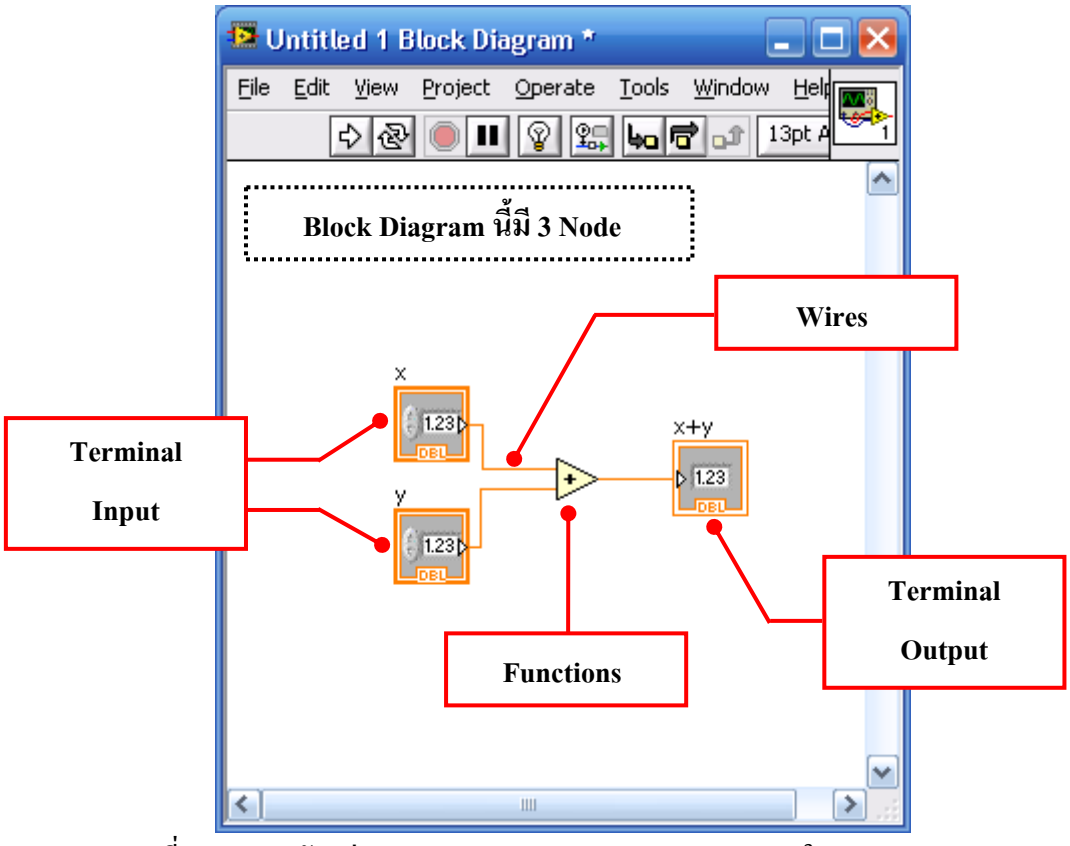


ภาพที่ 1.5 แสดง Functions Palette ของทางฝั่ง Block Diagram

2.4 Wires คือการเชื่อมต่อการรับ - ส่งข้อมูลระหว่าง terminal หรือ node ต่างๆ ที่มีใน Block Diagram นี้เข้าด้วยกัน โดย wires นี้จะเป็นการกำหนดเส้นทางของข้อมูลว่าเมื่อออกจาก terminal หนึ่งแล้วจะกำหนดการไหลข้อมูลไปที่ node ใดบ้าง มีลำดับเป็นอย่างไร และสุดท้ายจะให้แสดงผลลัพธ์ที่ terminal ใด ซึ่งการเชื่อมต่อสายนี้จะทำให้เราเข้าใจถึงหลักการของ Data Flow Programming ได้ดีขึ้น ซึ่งลักษณะต่าง ๆ ของเส้นจะมีรูปแบบและสีที่แตกต่างกันไปกันขึ้นอยู่ชนิดของข้อมูล แสดงดังตารางที่ 1.1

ตารางที่ 1.1 แสดงลักษณะของเส้น Wires แบบต่างๆ

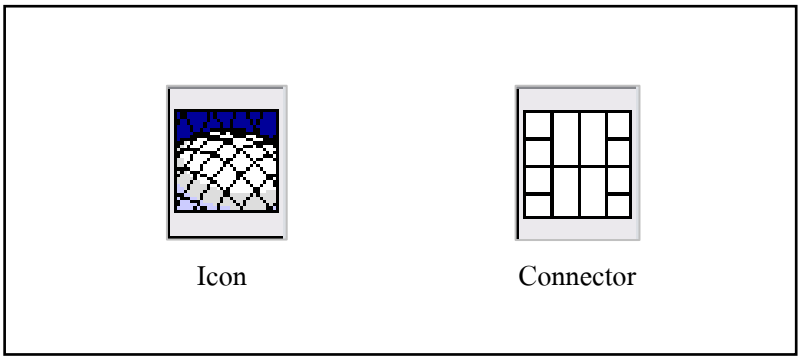
Type	Scalars	1-D Array	2-D Array	Color
เลขทศนิยม				ส้ม
เลขจำนวนเต็ม				น้ำเงิน
Boolean				เขียว
ตัวอักษร				ชมพู



ภาพที่ 1.6 แสดงตัวอย่าง Terminal, Node, Function และ wire ใน Block Diagram

3. Icon และ Connector เปรียบเสมือนโปรแกรมย่อย Subroutine ในโปรแกรมปกติทั่วไป icon จะหมายถึง block diagram ตัวหนึ่งที่มีการส่งข้อมูลเข้าและออกผ่านทาง Connector ซึ่งใน LabVIEW จะเรียก Subroutine นี้ว่า subVI ข้อดีของการเขียนโปรแกรมด้วยภาษา G นี้ก็คือเราสามารถสร้าง VI ทีละส่วนขึ้นมาให้ทำงานด้วยตัวเองได้อย่างอิสระ จากนั้นเราก็สามารถเขียนโปรแกรมอื่นเพื่อเรียกใช้งาน VI ที่เราเคยสร้างขึ้นก่อนหน้านี้ทีละตัว ซึ่งทำให้ VI ที่เราเขียนขึ้นก่อนกลายเป็น subVI ไป การเขียนในลักษณะนี้เราเรียกว่า module

สำหรับลักษณะทั่วไปของ Icon จะแสดงดังภาพที่ 1.7 (ซ้ายมือ) และ Connector นั้นจะแสดงพบว่ามีช่องต่อข้อมูลหรือที่เรียกว่า Terminal แสดงดังภาพที่ 1.7 (ขวามือ)



ภาพที่ 1.7 แสดง Icon และ Connector ของโปรแกรม LabVIEW

สำหรับคำศัพท์ต่างๆ ที่ใช้กันใน LabVIEW นี้ออกจะแตกต่างจากที่เราใช้กันในการเขียนโปรแกรมภาษาทั่วไป ดังแสดงตารางที่ 1.2 คำศัพท์ที่ควรรู้ในโปรแกรม LabVIEW ดังนี้

ตารางที่ 1.2 คำศัพท์ที่ควรรู้ในโปรแกรม LabVIEW

LabVIEW	**โปรแกรมพื้นฐาน**	**รายละเอียด**
VI	Program	ตัวโปรแกรมหลัก
Function	function	ฟังก์ชันสำเร็จรูปที่สร้างขึ้นกับโปรแกรมนั้นๆ เช่น sin, log เป็นต้น
SubVI	Subroutine	โปรแกรมย่อยที่ถูกเรียกใช้โดยโปรแกรมหลัก
Front Panel	user interface	ส่วนที่ติดต่อกับผู้ใช้
Block Diagram	Program code	การเขียนตามขั้นตอนของที่แต่ละโปรแกรมกำหนดขึ้น

CHAPTER 1

SCILAB with LabVIEW Gateway

สำหรับการติดต่อโปรแกรม SCILAB กับโปรแกรม LabVIEW นั้นจะมีไลบรารีที่ชื่อว่า ScriptScilab.dll ที่ทาง National Instruments และ INRIA จัดทำขึ้นมา และได้ให้ดาวน์โหลดตัวอย่างเพื่อติดตั้งใช้งาน Scilab - LabVIEW Gateway เวอร์ชัน 1.1 แบบฟรี (Open source) โดยสามารถดาวน์โหลดได้ที่ http://www.scilab.org/products/other/labview_gateway สำหรับขั้นตอนในการติดตั้งมีดังนี้

ระบบที่ต้องการ :

Windows 2000/XP/Vista

Scilab 4.1.1 release or higher

LabVIEW 8.0 or 8.2.x

วิธีการติดตั้งไลบรารี Scilab - LabVIEW Gateway

1. ขั้นตอนในการติดตั้งซอฟแวร์ไลบรารี Scilab - LabVIEW Gateway 1.1 นั้นจะต้องทำการติดตั้งโปรแกรม Scilab 4.1.1 หรือสูงกว่า และโปรแกรม LabVIEW 8.0 หรือ 8.2.x ขึ้นไป ก่อนจะติดตั้งไลบรารีตัวนี้ โดยเมื่อขยายไฟล์ที่ได้ (UNZIP) จะมีโฟลเดอร์ และไฟล์ต่างๆ ดังนี้ภาพที่ 1.8

ภาพที่ 1.8 ไฟล์และโฟลเดอร์ที่อยู่ใน Scilab - LabVIEW Gateway 1.1

หมายเหตุ สำหรับการทดลองในที่นี้ ทางผู้จัดทำเอกสารได้ทำการติดตั้งโปรแกรม Scilab เวอร์ชัน 4.1.1 และโปรแกรม LabVIEW เวอร์ชัน 8.6

2. จากนั้นทำการคัดลอกไฟล์ต่าง ๆ ลงใน Folder ของโปรแกรม LabVIEW ดังนี้

<LabVIEW Installation Root>\examples\Scilab\Scilab 3D Graph.vi

<LabVIEW Installation Root>\examples\Scilab\Scilab Data Acquisition (lowlevel).vi

<LabVIEW Installation Root>\examples\Scilab\Scilab Data Acquisition.vi

<LabVIEW Installation Root>\examples\Scilab\Scilab Fourier Series.vi

<LabVIEW Installation Root>\examples\Scilab\Scilab Life.vi

<LabVIEW Installation Root>\examples\Scilab\Scilab Script Node Examples.aliases

<LabVIEW Installation Root>\examples\Scilab\Scilab Script Node Examples.lvproj

<LabVIEW Installation Root>\examples\Scilab\Scilab SignalGen-Analysis.vi

<LabVIEW Installation Root>\menus\Categories\Mathematics_scriptsAndFormulas\MoreScriptNodes\scilab.mnu

<LabVIEW Installation Root>\resource\script\ScriptScilab.dll

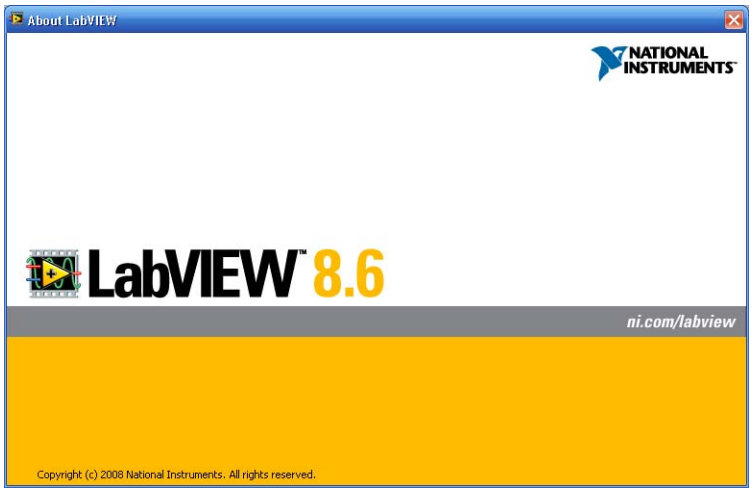
<LabVIEW Installation Root>\vi.lib\Scilab\ScilabScript.vi

<LabVIEW Installation Root>\Scilab - LabVIEW readme.pdf

สำหรับตัวอย่างหรือ Examples ที่ดาวน์โหลดมาจะต้องใช้งานโปรแกรม LabVIEW เวอร์ชัน 8.x ขึ้นไปจึงสามารถใช้งานได้

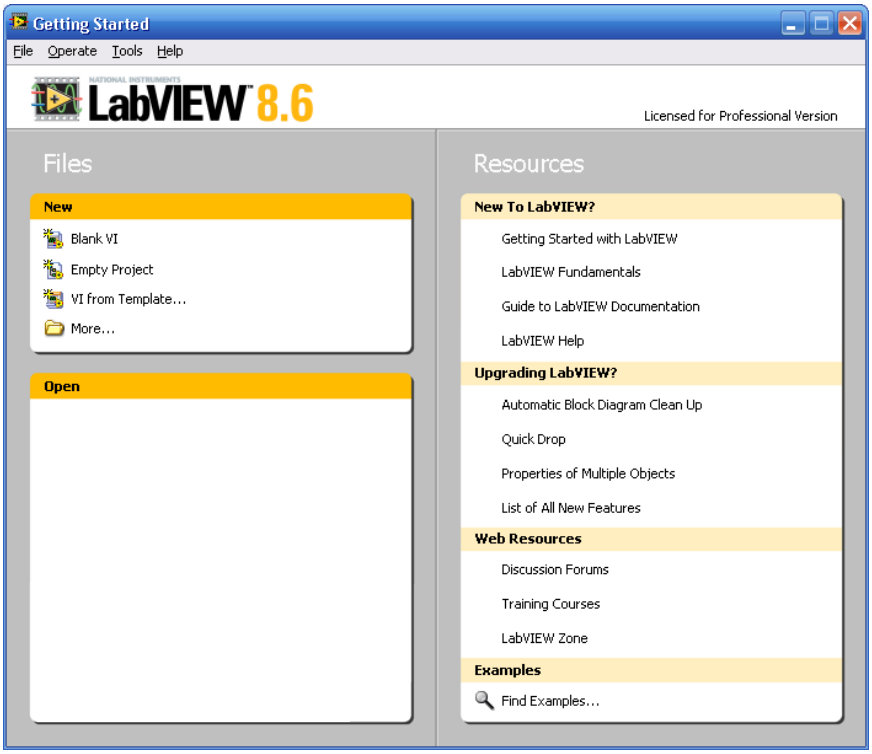
3. ขั้นตอนการเรียกใช้งาน Scilab Script มีดังนี้

3.1 หลังจากที่ได้ทำการติดตั้ง Scilab - LabVIEW Gateway 1.1 ไปแล้วนั้น หลังจากนั้นจะเป็นการเรียกใช้ Scilab Scripts ที่ทำการติดตั้งไปแล้ว โดยอันดับแรกเปิดโปรแกรม LabVIEW 8.6 ขึ้นมาจะแสดงหน้าต่างการโหลดค่าเริ่มต้นของโปรแกรม และ Functions ต่างๆ ขึ้นมาดังภาพที่ 1.2



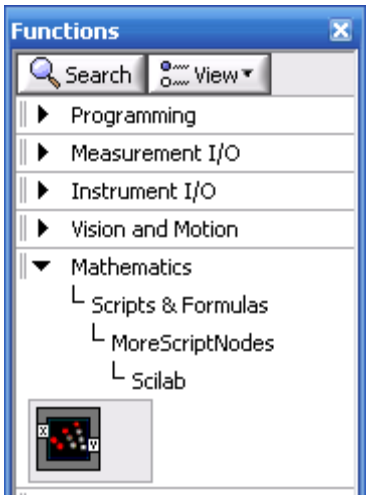
ภาพที่ 1.2 แสดงหน้าต่างโหลดค่าเริ่มต้นของโปรแกรม LabVIEW

3.2 เมื่อทำการโหลดค่าเริ่มต้นต่าง ๆ เสร็จแล้วโปรแกรมจะแสดงหน้าต่าง Getting Started ขึ้นมาดังภาพที่ 1.3 โดยในที่นี้ผู้อ่านทำการคลิกเลือกเมนูที่ Blank VI เพื่อทำการสร้างโปรแกรมย่อยๆ ขึ้นมา สำหรับรายละเอียดอื่นๆ สามารถหาศึกษาได้จากเว็บ www.ni.com ซึ่งจะมีการรวบรวมตัวอย่างและวิธีการใช้งานโปรแกรมไว้มากมายที่นี้



ภาพที่ 1.3 แสดงหน้าต่าง Getting Started

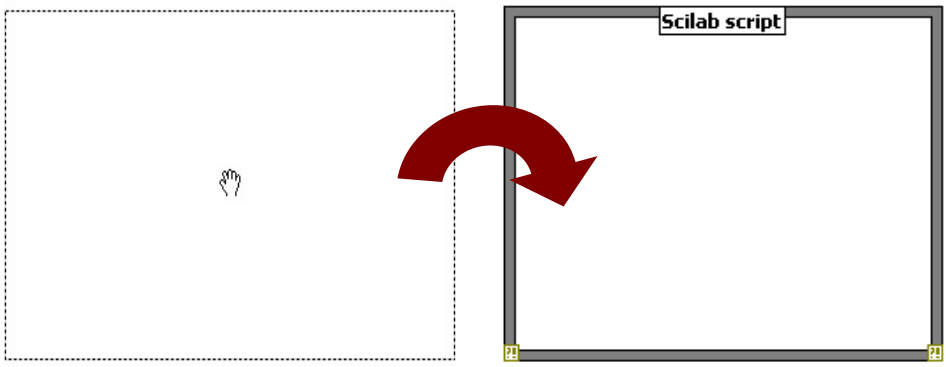
3.3 เริ่มทำการเรียกใช้งานฟังก์ชัน Scilab Script คลิกขวาที่ทางฝั่ง Block Diagram จะแสดง Functions ทั้งหมดของโปรแกรม LabVIEW ขึ้นมาเลือกคำสั่งดังนี้ **Mathematics >> Scripts & Formulas >> MoreScriptNodes >> Scilab >> Scilab Script** จะแสดงสัญภาพ (Icon) ดังภาพที่ 1.4



ภาพที่ 1.4 แสดงการเรียกใช้งาน Function Scilab

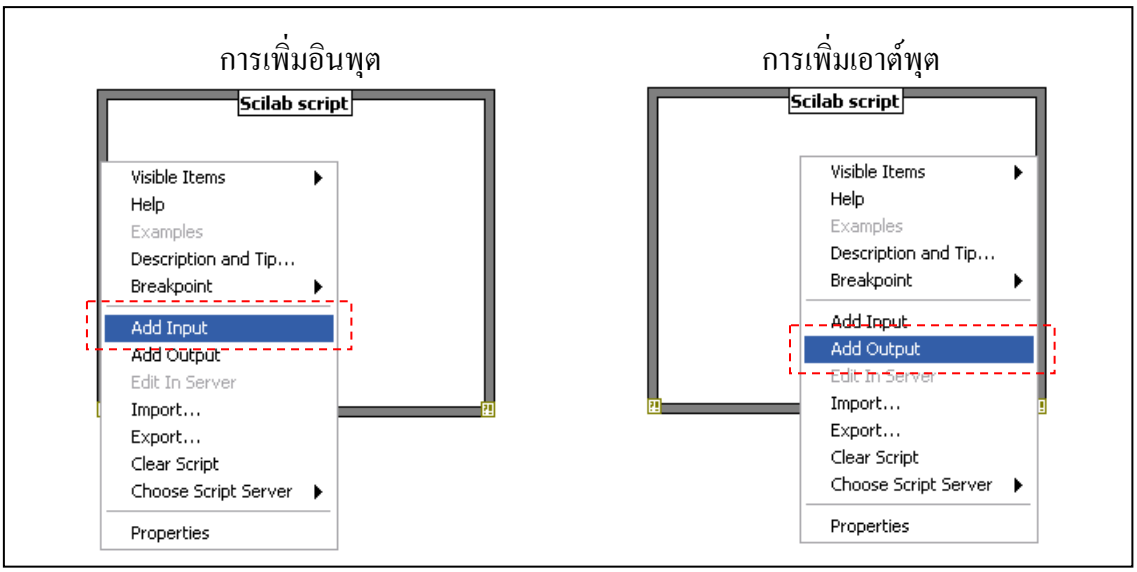
สำหรับการใช้ฟังก์ชัน Scilab Script นั้นจะต้องมีตัวโปรแกรม Scilab ติดตั้งไว้ด้วยเนื่องจากตัวไลบรารีได้ทำการติดต่อกับตัวโปรแกรม Scilab โดยตรงดังนั้นถ้าไม่ได้ติดตั้งโปรแกรม Scilab ไว้ก่อนจะทำให้เกิดการ Error ขึ้นมาได้

จากนั้นการเรียกใช้งาน Scilab Script ให้ทำการคลิกที่สัญรูป (Icon) 1 ครั้ง ลักษณะของ cursor จะเปลี่ยนเป็นรูปมือ (hand) แล้วคลิกที่ Block Diagram 1 ครั้งจะได้ตามภาพที่ 1.5



ภาพที่ 1.5 แสดงขั้นตอนการใช้งาน Scilab script

สำหรับขั้นตอนการเขียนโค้ดของฟังก์ชัน Scilab Script นั้นจะเหมือนกับการเขียนโค้ดโปรแกรม Scilab อาจจะแตกต่างกันบ้าง ขึ้นอยู่กับอัลกอริทึม (algorithm) ที่ต้องการของโปรแกรมนั้นๆ ส่วนการรับข้อมูลและแสดงผลลัพธ์จะมีคำสั่ง Add Input และ Add Out โดยคลิกขวาที่กรอบข้าง ๆ ของ Scilab Script จะแสดงตัวเลือกดังภาพที่ 1.6



ภาพที่ 1.6 แสดงการเพิ่มอินพุตและเอาต์พุต

CHAPTER 2

Examples of applications

สำหรับในบทเรียนนี้ผู้จัดทำได้เขียนขึ้นตัวอย่างจำนวน 3 รายการดังนี้

1. การหาผลลัพธ์ของ Log ฐาน 2
2. การวาดรูปกราฟ 2 มิติ มีตัวอย่างการวาดรูปสัญญาณต่างๆ ดังนี้
    - 2.1 รูปสัญญาณไซนูซอยด์ (sinusoid waveform)
    - 2.2 รูปสัญญาณพังก์ชันซิงก์ (sinc function)
    - 2.3 รูปสัญญาณพังก์ชันซิงก์แบบตัดข้อมูลบางส่วนของกราฟ (sinc function)
    - 2.4 รูปสัญญาณพังก์ชันซิงก์แบบตัดข้อมูลบางส่วนของกราฟแบบที่ 2 (sinc function)
3. การวาดรูปกราฟ 3 มิติ มีตัวอย่างการวาดรูปสัญญาณต่างๆ ดังนี้
    - 3.1 รูปกราฟสามมิติจากสมการ $z = | 0.5*\cos(2\pi x) * \cos(2\pi y) |$
    - 3.2 รูปกราฟสามมิติพื้นผิวแบบมีสีของรูปทรงห่วงยาง (torus)
    - 3.3 รูปสัญญาณพังก์ชันซิงก์แบบ 3 มิติ (sinc function)

1. การหาผลลัพธ์ของ Log ฐาน 2 ดังนี้

ในที่นี้ผู้จัดทำจะยกตัวอย่างการใช้งานฟังก์ชัน Log2 ของโปรแกรม Scilab นั้นโดย ลอการิทึม เป็นการดำเนินการทางคณิตศาสตร์ ที่เป็นฟังก์ชันผกผันของ ฟังก์ชันเอ็กซ์โพเนนเชียล (ใช้ค่าคงตัว หรือ "ฐาน" เป็นเลขยกกำลัง) ลอการิทึมของจำนวน x ที่มีฐาน b คือจำนวน n นั่นคือ $x = b^n$ เขียนได้ดังนี้

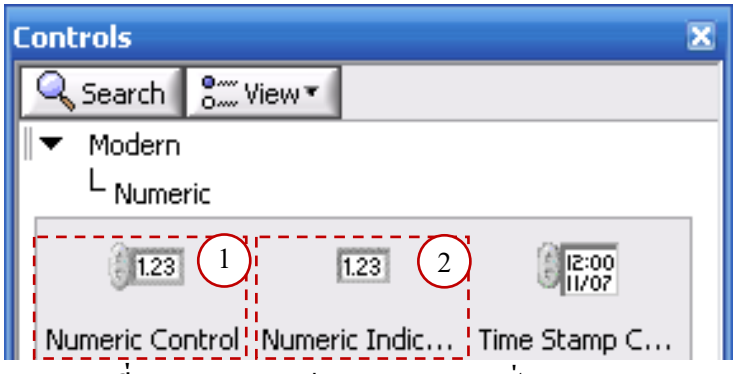
$$\log_b(x) = n$$

ตัวอย่างเช่น

$$\text{Log}_2(128) = 7$$

เพราะว่า 2 x 2 x 2 x 2 x 2 x 2 x 2 = 128

1.1 สำหรับส่วนของ Front Panel ให้เลือกตัวควบคุมสำหรับแสดงผลสัญญาณการวาดรูปกราฟ 2 มิติดังนี้ **Modern → Numeric → Numeric Control** นำมาวาง 1 ตัว และ **Modern → Numeric → Numeric Indicator** ดังภาพที่ 2.1



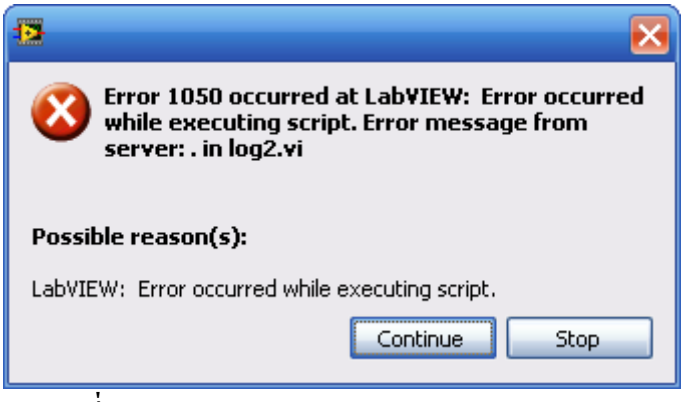
ภาพที่ 2.1 แสดงการเรียก Controls ทางฝั่ง Front Panel

1.2 จากนั้นในส่วนของ Block Diagram ให้เลือกฟังก์ชัน Scilab script ขึ้นมาและเขียน code Scilab ดังนี้

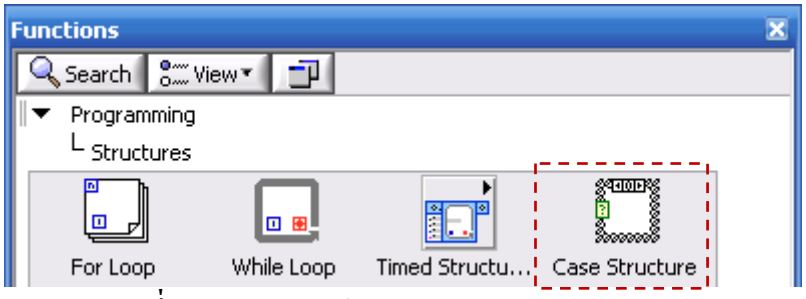
```
z = log2(x)
```

1.3 จากนั้นให้ Add Input ที่มีชื่อว่า x และ Add Output ชื่อว่า z (สามารถดูรายละเอียดวิธีการ Add Output ได้ใน CHAPTER 1) สำหรับการหาค่า Log นั้น ค่าเริ่มต้นของโปรแกรมจะเริ่มที่ 0 ซึ่งจะทำให้หาค่า Log ไม่ได้ แสดงดังภาพที่ 2.2 ดังนั้นจึงต้องเขียน Code เพื่อตรวจสอบ Input ที่รับเข้ามาว่ามีค่าเป็น 0 จริงหรือไม่ โดยในที่นี้จะใช้ Function Case Structure โดยเลือก

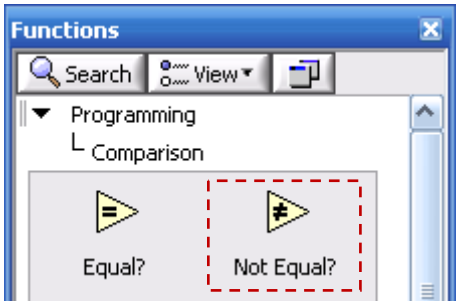
Programming → **Structures** → **Case Structure** ดังภาพที่ 2.3 และ Function Case Structure โดยเลือก **Programming** → **Comparison** → **Not Equal?** ดังภาพที่ 2.4



ภาพที่ 2.2 แสดง Error ของ Function Scilab Script

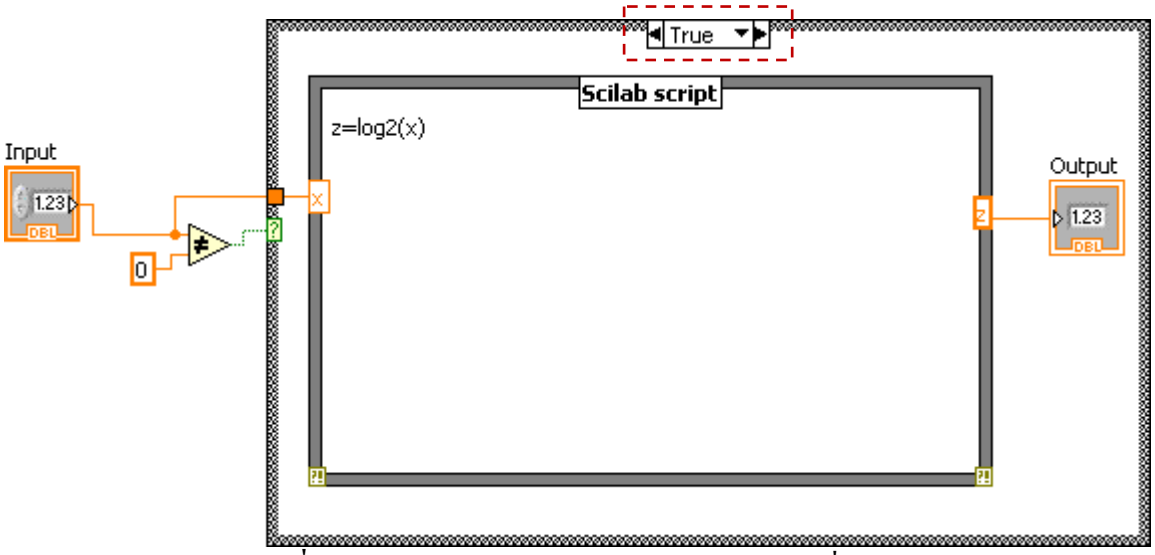


ภาพที่ 2.3 แสดงการเรียก Function Case Structure

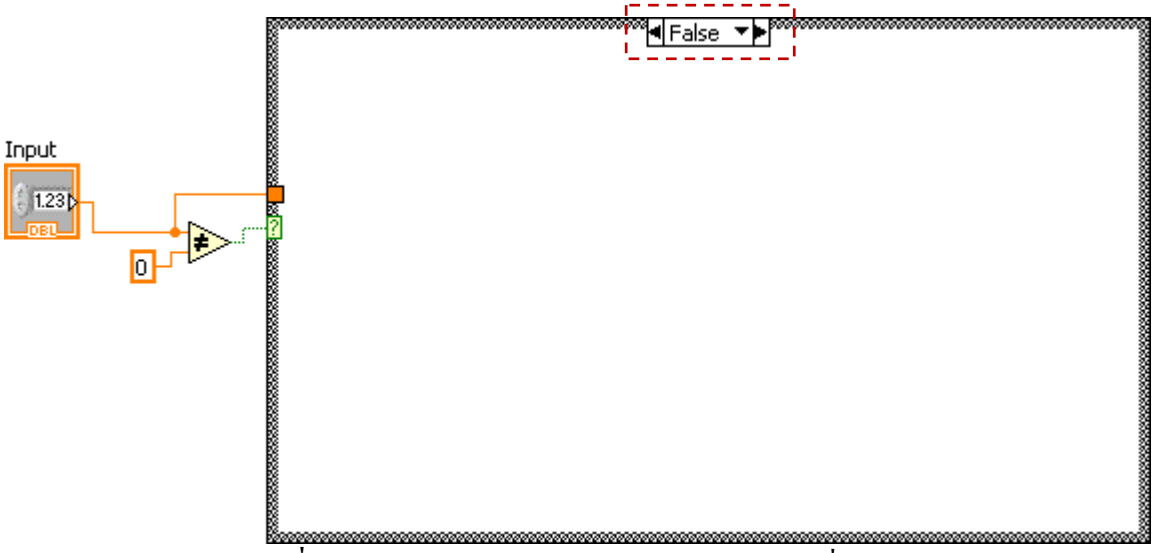


ภาพที่ 2.4 แสดงการเรียก Function Case Structure

1.4 จากนั้นเขียน code ถ้าเงื่อนไขเป็นจริง (True) ให้เข้ามาทำงานใน Function Scilab script ดังภาพที่ 2.5 แต่ถ้าเงื่อนไขเป็นเท็จ (False) ไม่ต้องเขียน Code ใด ๆ ดังภาพที่ 2.6 และเชื่อมต่อ Wires แต่ละ Node เข้าด้วยกันดังภาพที่ 2.6 และ 2.7 ตามลำดับ

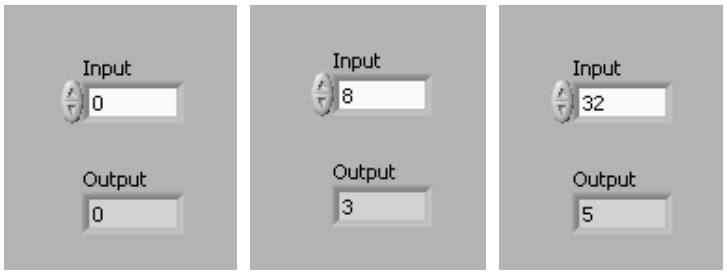


ภาพที่ 2.5 แสดงการเขียน Function Case Structure เงื่อนไขเป็นจริง (True)



ภาพที่ 2.6 แสดงการเขียน Function Case Structure เงื่อนไขเป็นเท็จ (False)

1.4 จากนั้นทดสอบการหาผลลัพธ์ของ Log ฐาน 2 โดยการคลิกที่เมนูRun Continuously  จะแสดงผลลัพธ์ ดังภาพที่ 2.7

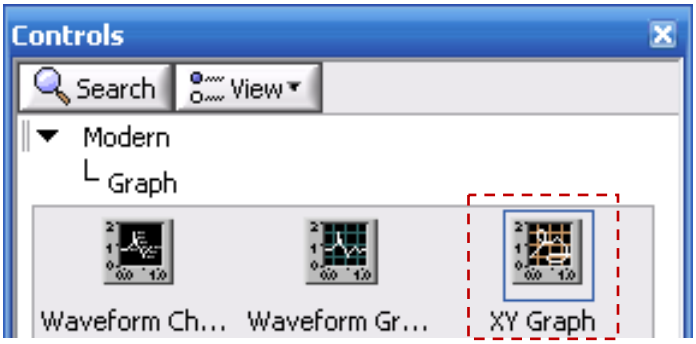


ภาพที่ 2.7 แสดงผลลัพธ์ของ Log ฐาน 2 ด้วย Scilab script

2.1 ตัวอย่างที่ 1 การวาดรูปกราฟสองมิติของรูปสัญญาณไซนูซอยด์ (sinusoid waveform) ดังนี้

สูตร $y = \sin(2\pi ft)$

2.1.1 สำหรับส่วนของ Front Panel ให้เลือกตัวควบคุมสำหรับแสดงผลสัญญาณการวาดรูปกราฟ 2 มิติดังนี้ **Modern → Graph → XY Graph** ดังภาพที่ 2.8

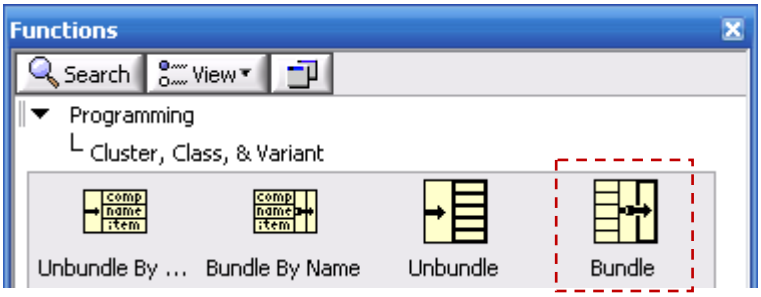


ภาพที่ 2.8 แสดงการเรียกใช้งาน Control XY Graph

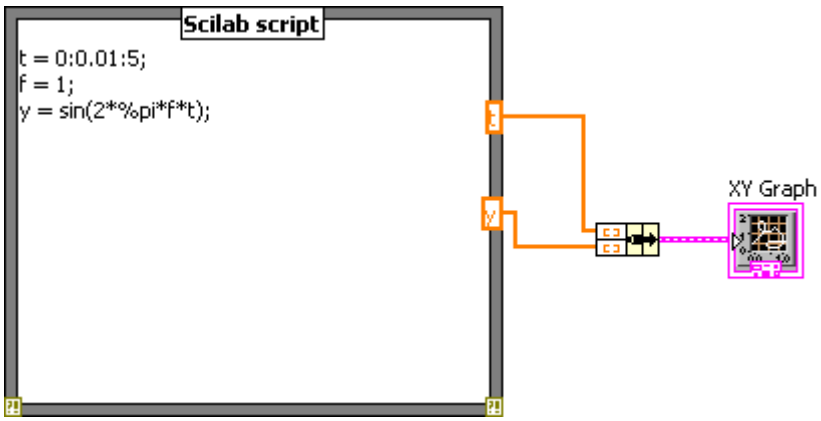
2.1.2 จากนั้นในส่วนของ Block Diagram ให้เลือกฟังก์ชัน Scilab script ขึ้นมาและเขียน code Scilab ดังนี้

```
-->t = 0:0.01:2;
-->f = 1;
-->y = sin(2*%pi*f*t);
```

2.1.3 จากนั้นให้ Add Output ที่มีชื่อว่า t และ y (สามารถดูรายละเอียดวิธีการ Add Output ได้ใน CHAPTER 1) และเรียกใช้ Function Bundle ดังนี้ **Programming → Cluster, Class, & Variant → Bundle** ตามภาพที่ 2.9 เพื่อทำการส่งสัญญาณเพียงช่องทางเดียว และเชื่อมต่อ Wires แต่ละ Node เข้าด้วยกัน ดังภาพที่ 2.10

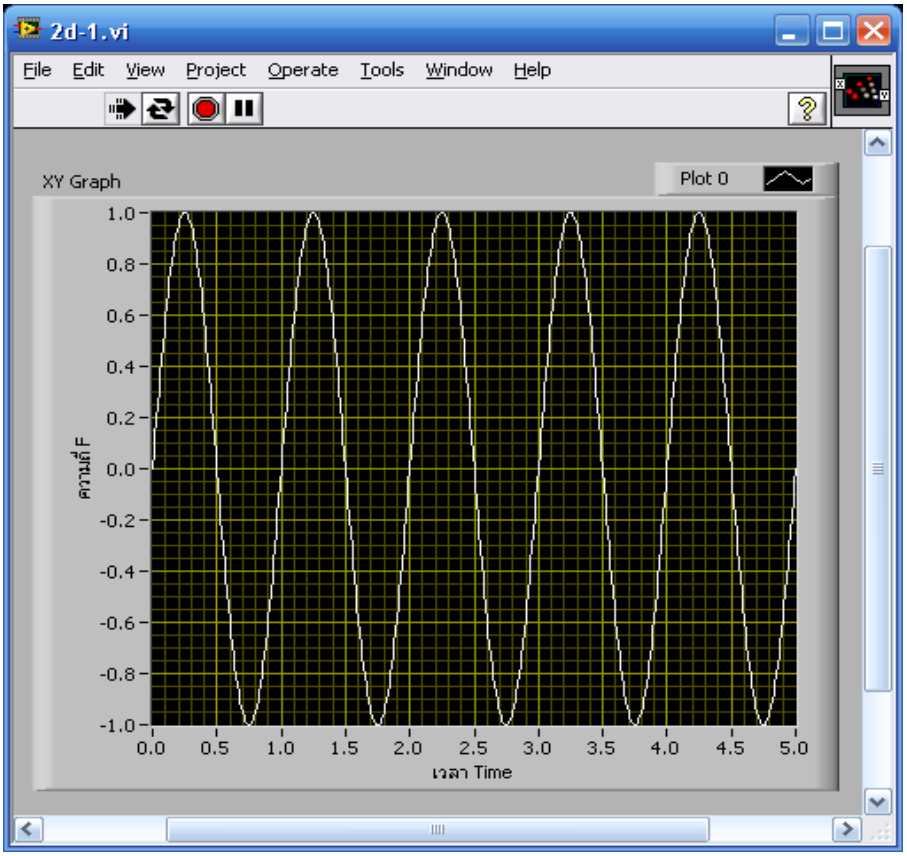


ภาพที่ 2.9 แสดงการเรียกใช้งานฟังก์ชัน Bundle



ภาพที่ 2.10 แสดงการเชื่อมต่อ Scilab script เข้ากับ XY Graph

2.1.4 จากนั้นทดสอบการวาดรูปกราฟ 2 มิติโดยการคลิกที่เมนู Run Continuously  จะแสดงรูปสัญญาณไซนูซอยด์ดังภาพที่ 2.11



ภาพที่ 2.11 แสดงผลลัพธ์จากการวาดรูปกราฟสองมิติสัญญาณไซนูซอยด์

2.2 ตัวอย่างที่ 2 การวาดรูปกราฟสองมิติของรูปสัญญาณฟังก์ชันซิงก์ (sinc function) ดังนี้

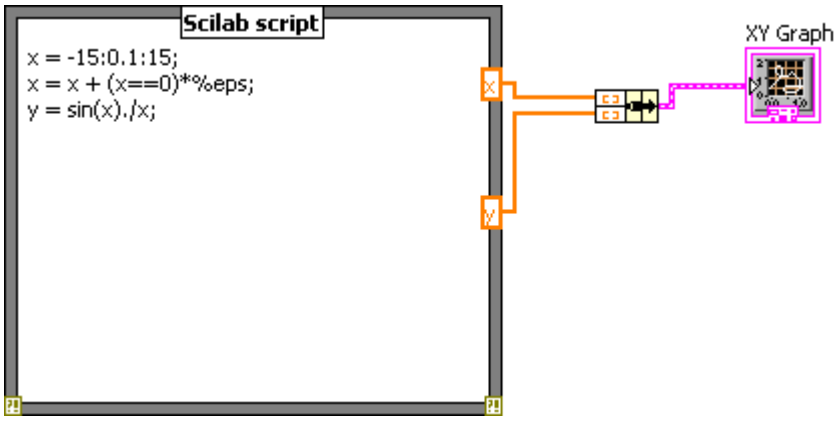
สูตร $y = \sin(x) / x$

2.2.1 สำหรับส่วนของ Front Panel ให้เลือกตัวควบคุมสำหรับแสดงผลสัญญาณการวาดรูปกราฟ 2 มิติดังนี้ **Modern → Graph → XY Graph** ดังภาพที่ 2.8 (ตามตัวอย่างที่ 1)

2.2.2 จากนั้นในส่วนของ Block Diagram ให้เลือกฟังก์ชัน Scilab script ขึ้นมาและเขียน code Scilab ดังนี้

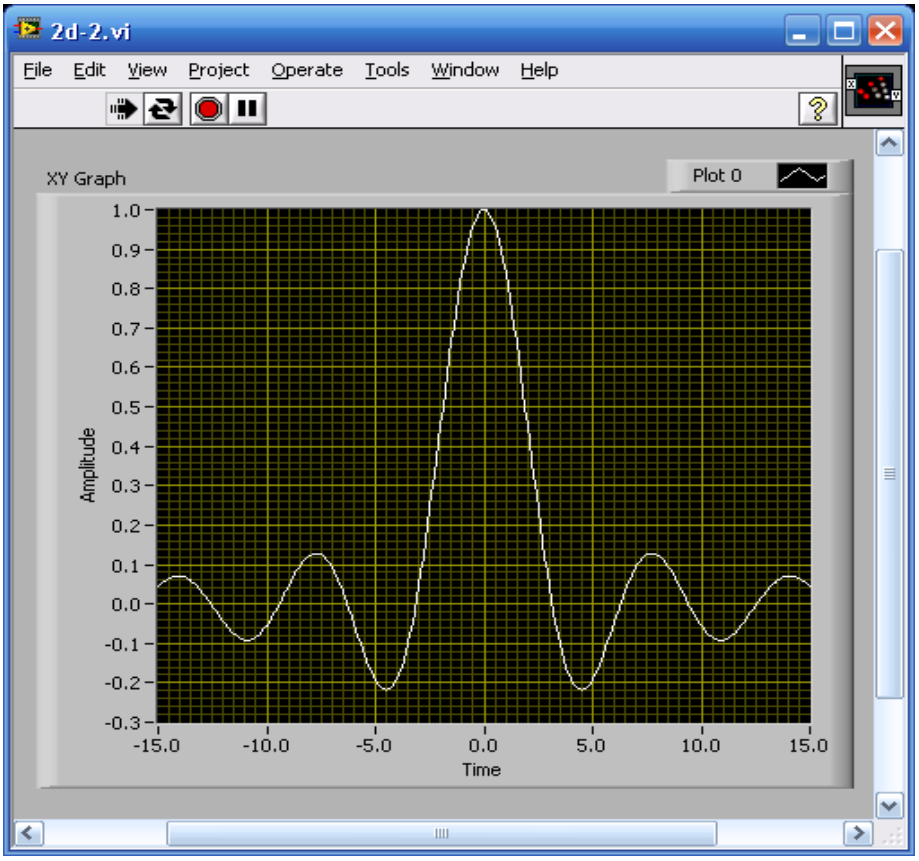
```
x = -15:0.1:15;
x = x + (x==0)*%eps;
y = sin(x)./x;
```

2.2.3 จากนั้นให้ Add Output ที่มีชื่อว่า t และ y (สามารถดูรายละเอียดวิธีการ Add Output ได้ใน CHAPTER 1) และเรียกใช้ Function Bundle ดังนี้ **Programming → Cluster, Class, & Variant → Bundle** ตามภาพที่ 2.9 (ตามตัวอย่างที่ 1) เพื่อทำการส่งสัญญาณเพียงช่องทางเดียว และเชื่อมต่อ Wires แต่ละ Node เข้าด้วยกัน ดังภาพที่ 2.12



ภาพที่ 2.12 แสดงการเชื่อมต่อ Scilab script เข้ากับ XY Graph

2.2.4 จากนั้นทดสอบการวาดรูปกราฟ 2 มิติโดยการคลิกที่เมนู Run Continuously  จะแสดงรูปกราฟสองมิติสัญญาณฟังก์ชันซิงก์ ดังภาพที่ 2.13



ภาพที่ 2.13 แสดงผลลัพธ์จากการวาดรูปกราฟสองมิติสัญญาณฟังก์ชันซิงก์

2.3 ตัวอย่างที่ 3 การวาดรูปกราฟสองมิติของรูปสัญญาณฟังก์ชันซิงก์แบบตัดข้อมูลบางส่วนของกราฟ (sinc function) ดังนี้

สูตร

$$y = \sin(x) / x$$

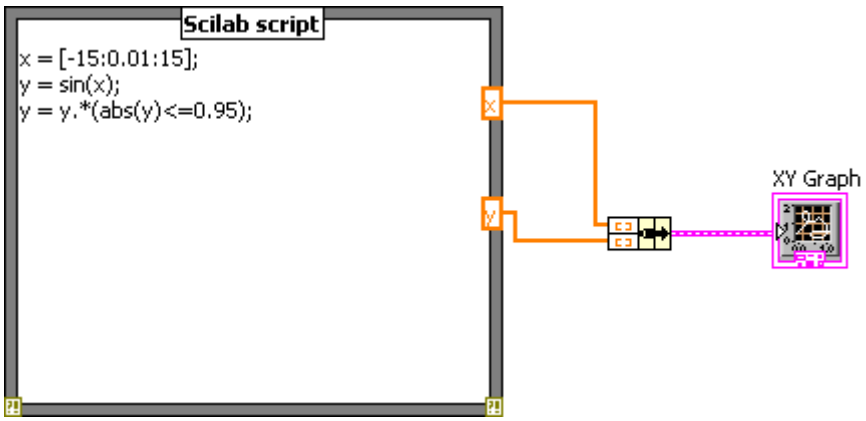
หมายเหตุ y จะเท่ากับ y คูณกับ (ค่าสมบูรณ์ของ y จะต้องน้อยกว่าหรือเท่ากับ 0.95)

2.3.1 สำหรับส่วนของ Front Panel ให้เลือกตัวควบคุมสำหรับแสดงผลสัญญาณการวาดรูปกราฟ 2 มิติดังนี้ **Modern → Graph → XY Graph** ดังภาพที่ 2.8 (ตามตัวอย่างที่ 1)

2.3.2 จากนั้นในส่วนของ Block Diagram ให้เลือกฟังก์ชัน Scilab script ขึ้นมาและเขียน code Scilab ดังนี้

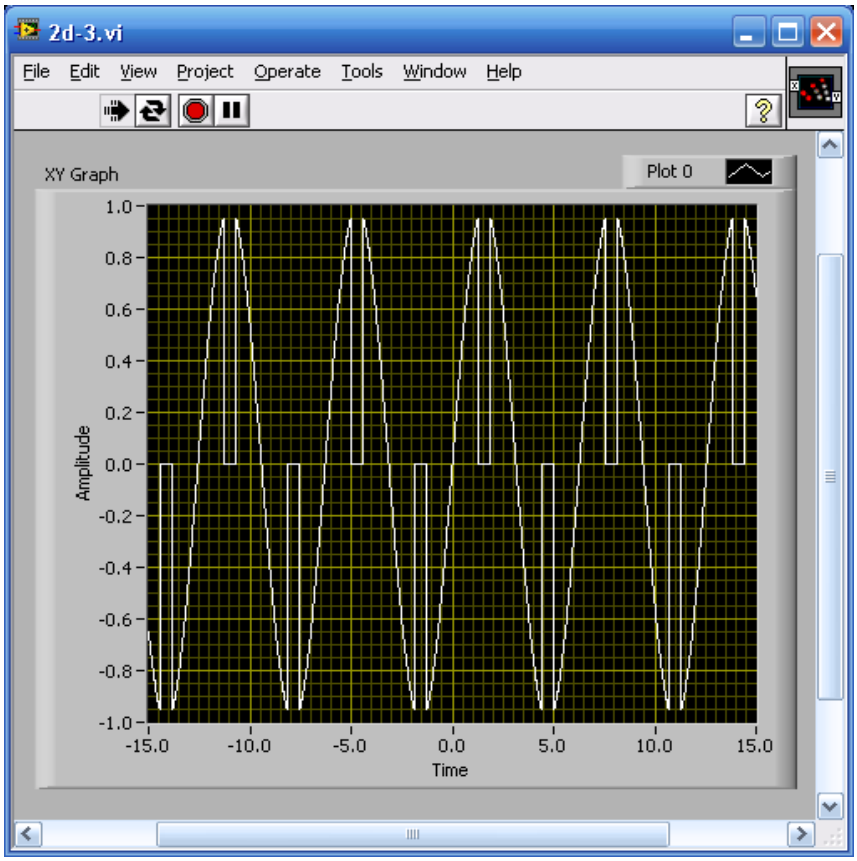
```
x = [-15:0.01:15];
y = sin(x);
y = y.*(abs(y)<=0.95);
```

2.3.3 จากนั้นให้ Add Output ที่มีชื่อว่า t และ y (สามารถดูรายละเอียดวิธีการ Add Output ได้ใน CHAPTER 1) และเรียกใช้ Function Bundle ดังนี้ **Programming → Cluster, Class, & Variant → Bundle** ตามภาพที่ 2.9 (ตามตัวอย่างที่ 1) เพื่อทำการส่งสัญญาณเพียงช่องทางเดียว และเชื่อมต่อ Wires แต่ละ Node เข้าด้วยกัน ดังภาพที่ 2.14



ภาพที่ 2.14 แสดงการเชื่อมต่อ Scilab script เข้ากับ XY Graph

2.3.4 จากนั้นทดสอบการวาดรูปกราฟ 2 มิติโดยการคลิกที่เมนู Run Continuously  จะแสดงรูปกราฟสองมิติสัญญาณฟังก์ชันซิงก์ (ตัดข้อมูลบางส่วนของกราฟ)ดังภาพที่ 2.15



ภาพที่ 2.15 แสดงผลลัพธ์จากการวาดรูปกราฟสองมิติสัญญาณฟังก์ชันซิงก์ (ตัดข้อมูลบางส่วนของกราฟ)

2.4 ตัวอย่างที่ 4 การวาดรูปกราฟสองมิติของรูปสัญญาณฟังก์ชันซิงก์แบบตัดข้อมูลบางส่วนของกราฟแบบที่ 2 (sinc function) ดังนี้

สูตร

$$y = \sin(x) / x$$

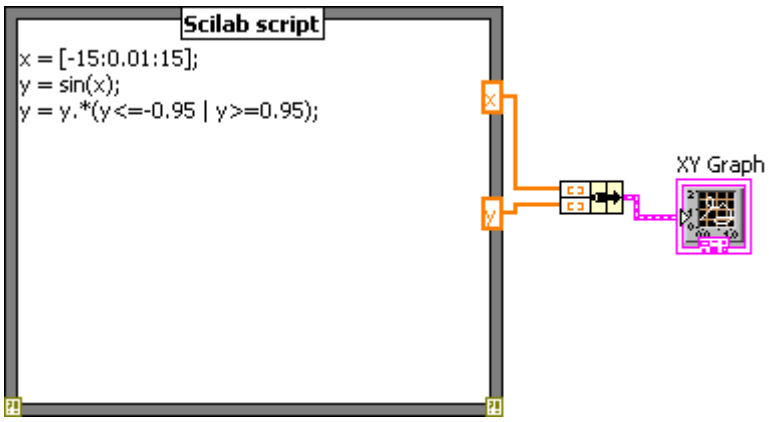
หมายเหตุ y จะเท่ากับ y คูณกับ (y น้อยกว่าหรือเท่ากับ 0.95 หรือ y มากกว่าหรือเท่ากับ 0.95)

2.4.1 สำหรับส่วนของ Front Panel ให้เลือกตัวควบคุมสำหรับแสดงผลสัญญาณการวาดรูปกราฟ 2 มิติดังนี้ **Modern → Graph → XY Graph** ดังภาพที่ 2.8 (ตามตัวอย่างที่ 1)

2.4.2 จากนั้นในส่วนของ Block Diagram ให้เลือกฟังก์ชัน Scilab script ขึ้นมาและเขียน code Scilab ดังนี้

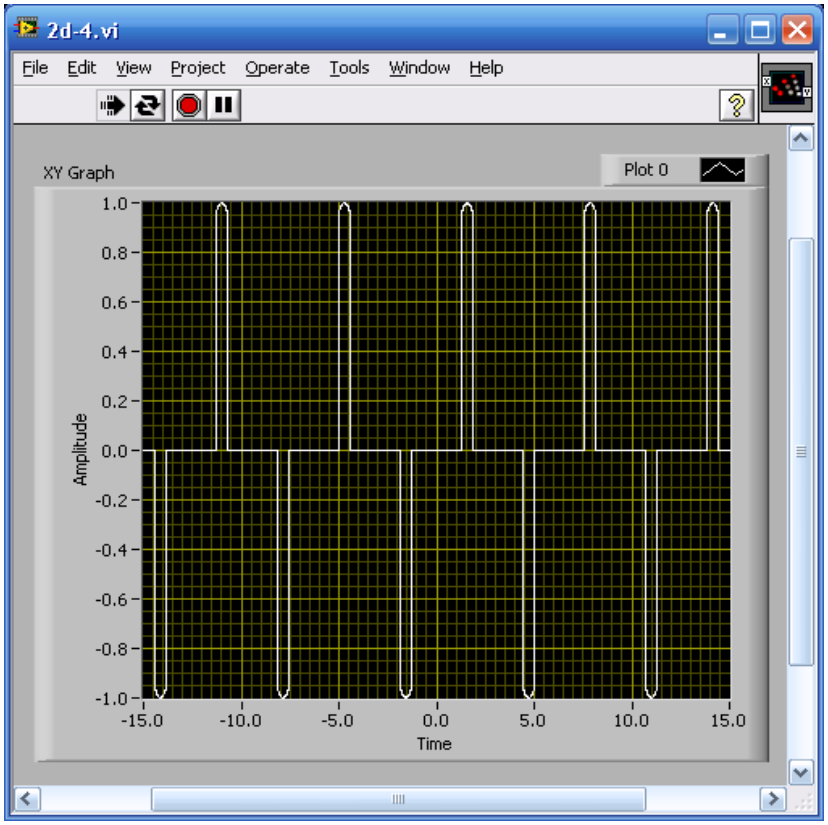
```
x = [-15:0.01:15];
y = sin(x);
y = y.*(y<=-0.95 | y>=0.95);
```

2.4.3 จากนั้นให้ Add Output ที่มีชื่อว่า t และ y (สามารถดูรายละเอียดวิธีการ Add Output ได้ใน CHAPTER 1) และเรียกใช้ Function Bundle ดังนี้ **Programming → Cluster, Class, & Variant → Bundle** ตามภาพที่ 2.9 (ตามตัวอย่างที่ 1) เพื่อทำการส่งสัญญาณเพียงช่องทางเดียว และเชื่อมต่อ Wires แต่ละ Node เข้าด้วยกัน ดังภาพที่ 2.16



ภาพที่ 2.16 แสดงการเชื่อมต่อ Scilab script เข้ากับ XY Graph

2.4.4 จากนั้นทดสอบการวาดรูปกราฟ 2 มิติโดยการคลิกที่เมนู Run Continuously  จะแสดงรูปกราฟสองมิติสัญญาณฟังก์ชันซิงก์ (ตัดข้อมูลบางส่วนของกราฟ)ดังภาพที่ 2.17

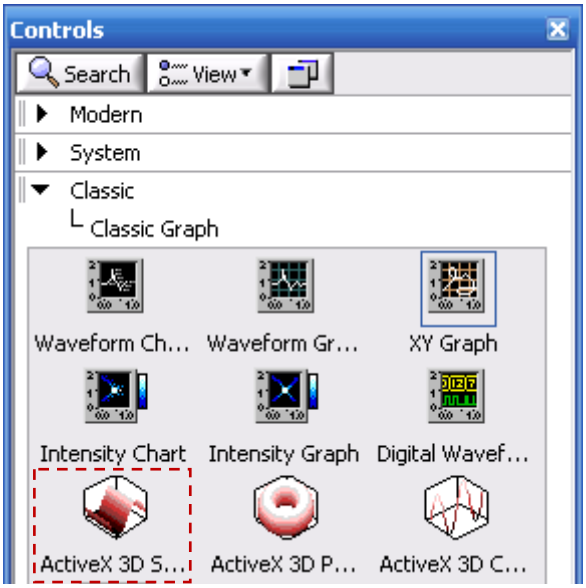


ภาพที่ 2.17 แสดงผลลัพธ์จากการวาดรูปกราฟสองมิติสัญญาณฟังก์ชันซิงก์ (ตัดข้อมูลบางส่วนของกราฟ แบบที่ 2)

3.1 ตัวอย่างที่ 1 การวาดรูปกราฟสามมิติจากสมการ z = | 0.5*cos(2πx) * cos(2πy) | ดังนี้

สูตร $z = | 0.5*\cos(2\pi x) * \cos(2\pi y) |$

3.1.1 สำหรับส่วนของ Front Panel ให้เลือกตัวควบคุมสำหรับแสดงผลสัญญาณการวาดรูปกราฟ 3 มิติดังนี้ **Classic → Classic Graph → ActiveX 3D Surface** ดังภาพที่ 2.18

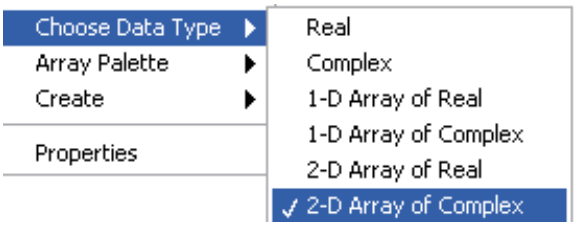


ภาพที่ 2.18 แสดงการเรียกใช้งาน Control ActiveX 3D Surface

3.1.2 จากนั้นในส่วนของ Block Diagram ให้เลือกฟังก์ชัน Scilab script ขึ้นมาและเขียน code Scilab ดังนี้

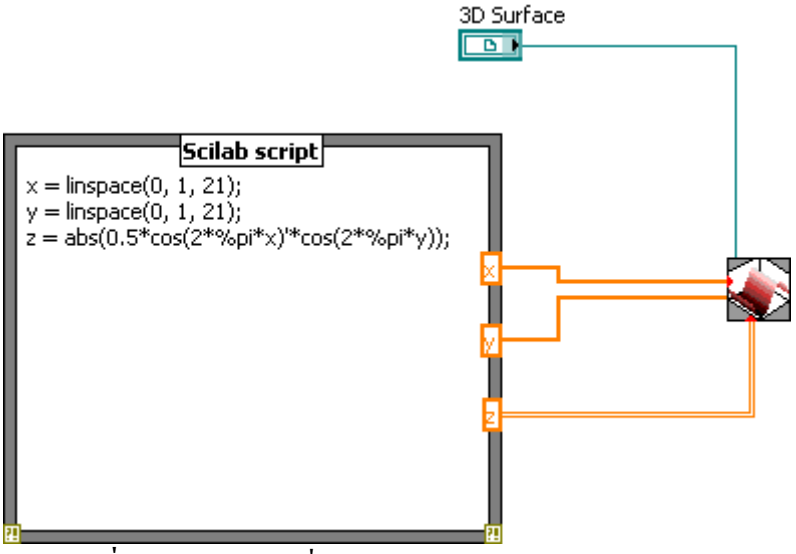
```
x = linspace(0, 1, 21);
y = linspace(0, 1, 21);
z = abs(0.5*cos(2*%pi*x)'*cos(2*%pi*y));
```

3.1.3 จากนั้นให้ Add Output ที่มีชื่อว่า x , y และ z โดยตัวแปร z จะต้องเปลี่ยนชนิดของตัวแปรเป็น Array 2 มิติ โดยการคลิกขวาที่ตัวแปรเอาต์พุต z เลือก **Choose Data Type → 2-D Array of Complex** ดังภาพที่ 2.19



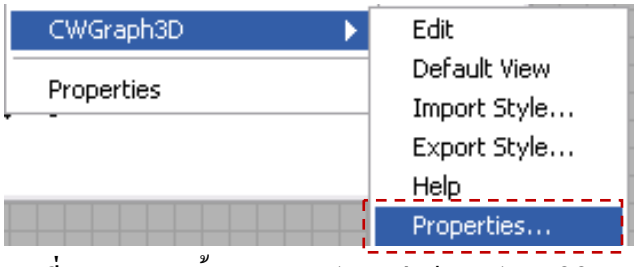
ภาพที่ 2.19 แสดงการเปลี่ยนชนิดของตัวแปรเป็น 2-D Array of Complex

3.1.4 หลังจากนั้นทำการเชื่อมต่อ Wires แต่ละ Node เข้าด้วยกัน ดังภาพที่ 2.20

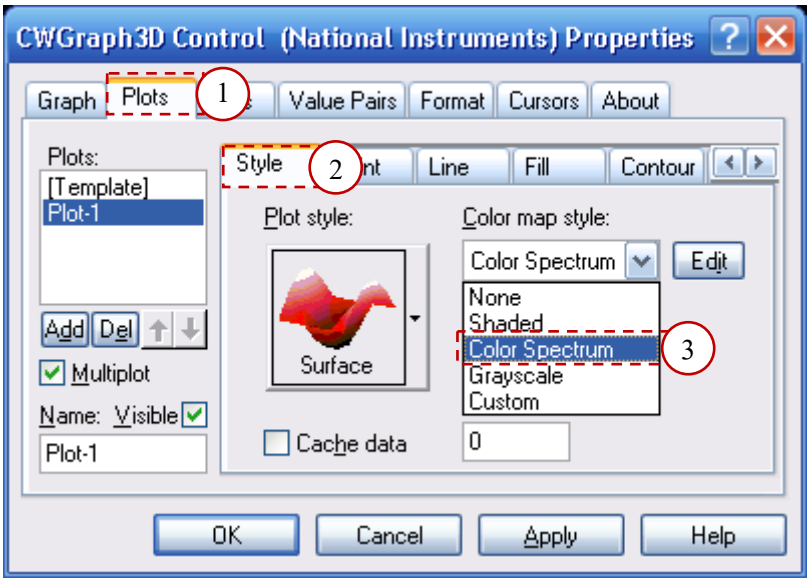


ภาพที่ 2.20 แสดงการเชื่อมต่อ Scilab script เข้ากับ 3D Surface

3.1.5 สำหรับการปรับเปลี่ยนสีกราฟ 3 มิติทำได้โดยการคลิกขวาที่ Control 3D Surface เลือก **CWGraph3D → Properties...** ดังภาพที่ 2.21 หลังจากนั้น เลือกแท็บเมนู **Plots → Style** และในส่วนของ Color map style เลือกเป็น Color Spectrum ดังภาพที่ 2.22

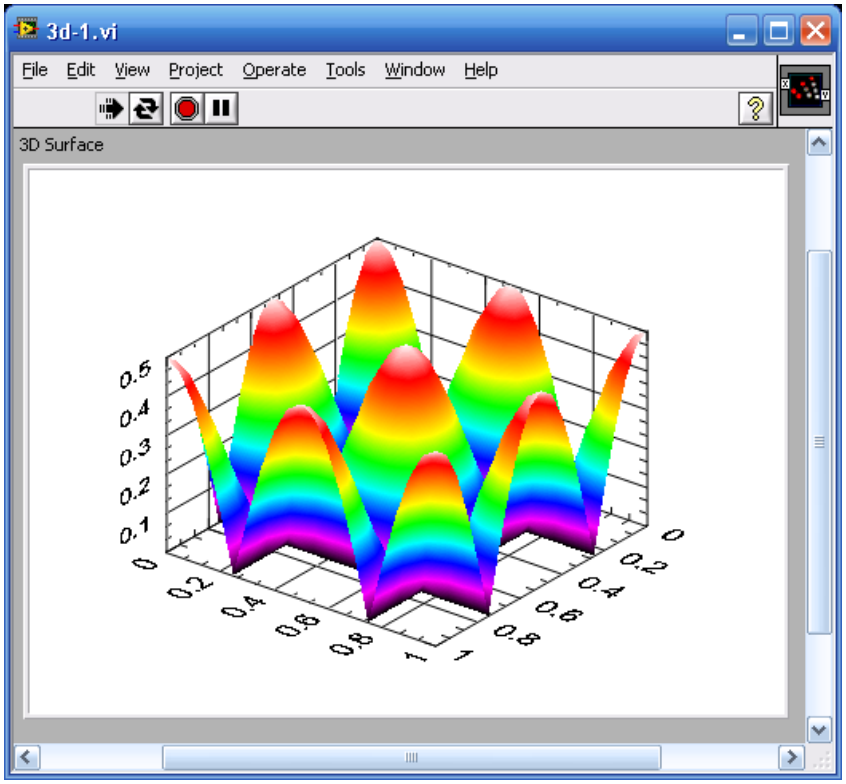


ภาพที่ 2.21 แสดงขั้นตอนการปรับแต่งสีกราฟสามมิติ



ภาพที่ 2.22 แสดงขั้นตอนการปรับแต่งสีกราฟสามมิติ (ต่อ)

3.1.6 จากนั้นทดสอบการวาดรูปกราฟ 3 มิติโดยการคลิกที่เมนู Run Continuously  จะแสดงรูปกราฟสามมิติจากสมการ z = | 0.5*cos(2πx) * cos(2πy) | ดังภาพที่ 2.23



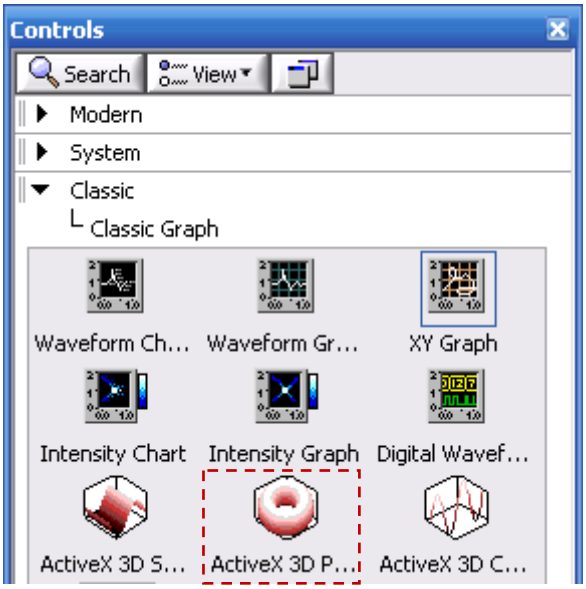
ภาพที่ 2.23 แสดงผลลัพธ์การวาดรูปกราฟสามมิติจากสมการ z = | 0.5*cos(2πx) * cos(2πy) |

3.2 ตัวอย่างที่ 2 การวาดรูปกราฟสามมิติพื้นผิวแบบมีสีของรูปทรงห่วงยาง (torus) ดังนี้

สูตร

$$x = (a+c\cos(\phi))\cos(\theta)$$
$$y = (a+c\cos(\phi))\sin(\theta)$$
$$z = c\sin(\theta)$$

3.2.1 สำหรับส่วนของ Front Panel ให้เลือกตัวควบคุมสำหรับแสดงผลสัญญาณการวาดรูปกราฟ 3 มิติดังนี้ **Classic → Classic Graph → ActiveX 3D Parametric Graph** ดังภาพที่ 2.24



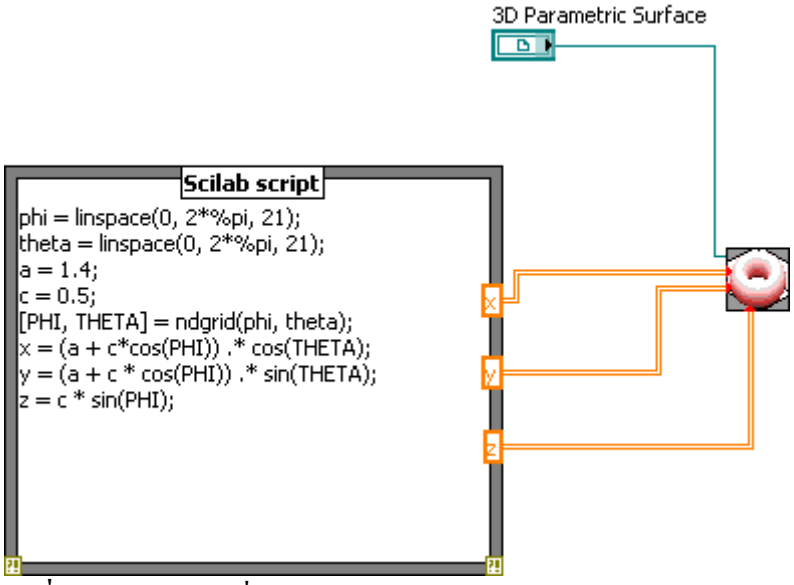
ภาพที่ 2.24 แสดงการเรียกใช้งาน Control ActiveX 3D Parametric Graph

3.2.2 จากนั้นในส่วนของ Block Diagram ให้เลือกฟังก์ชัน Scilab script ขึ้นมาและเขียน code Scilab ดังนี้

```
phi = linspace(0, 2*%pi, 21);
theta = linspace(0, 2*%pi, 21);
a = 1.4;
c = 0.5;
[PHI, THETA] = ndgrid(phi, theta);
x = (a + c*cos(PHI)) .* cos(THETA);
y = (a + c * cos(PHI)) .* sin(THETA);
z = c * sin(PHI);
```

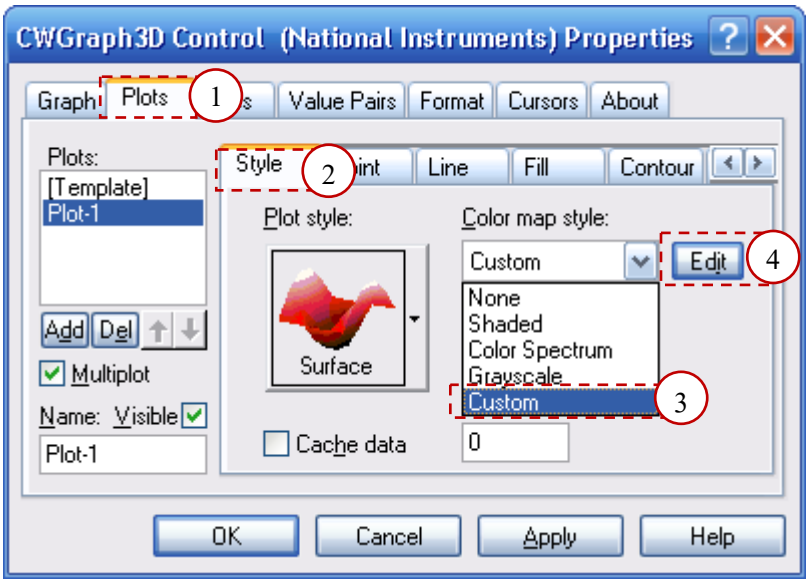
3.2.3 จากนั้นให้ Add Output ที่มีชื่อว่า x , y และ z โดย**ทุกตัวแปร**จะต้องเปลี่ยนชนิดของตัวแปรเป็น Array 2 มิติ โดยการคลิกขวาที่ตัวแปรเอาต์พุต z เลือก **Choose Data Type → 2-D Array of Complex** ดังภาพที่ 2.19 (ตัวอย่างที่ 1)

3.2.4 หลังจากนั้นทำการเชื่อมต่อ Wires แต่ละ Node เข้าด้วยกัน ดังภาพที่ 2.25

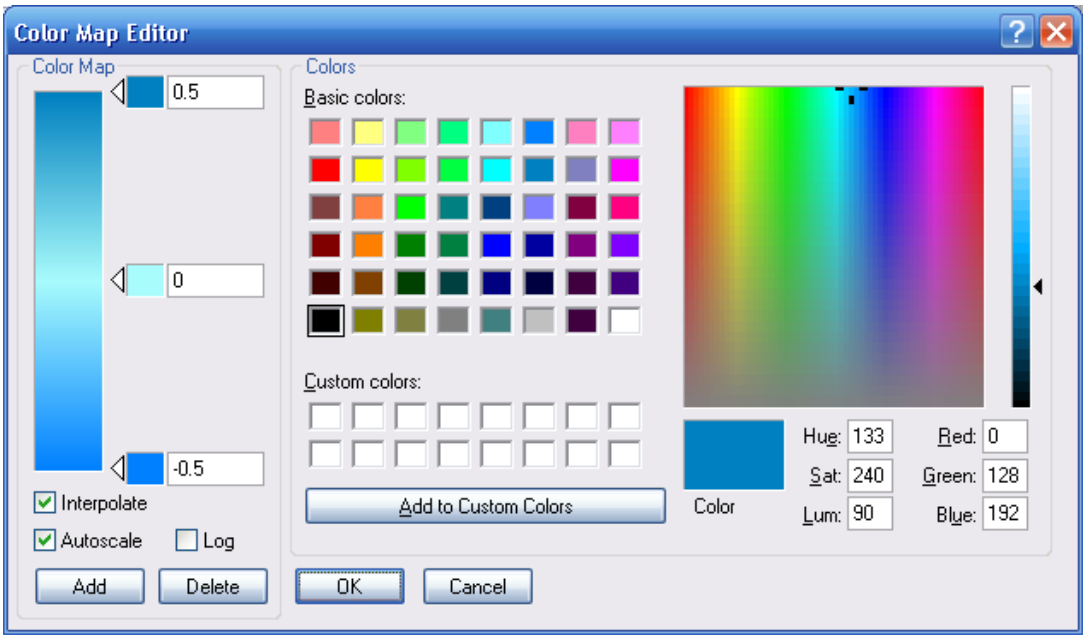


ภาพที่ 2.25 แสดงการเชื่อมต่อ Scilab script เข้ากับ 3D Parametric Surface

3.2.5 สำหรับการปรับเปลี่ยนสีกราฟ 3 มิติทำได้โดยการคลิกขวาที่ Control 3D Surface เลือก **CWGraph3D → Properties...** ดังภาพที่ 2.21 (ตัวอย่างที่ 1) หลังจากนั้น เลือกแท็บเมนู **Plots → Style** และในส่วนของ Color map style เลือกเป็น Custom และคลิกที่ปุ่ม Edit เพื่อเลือกสี ดังภาพที่ 2.26 และเลือกสีที่ต้องการดังภาพที่ 2.27

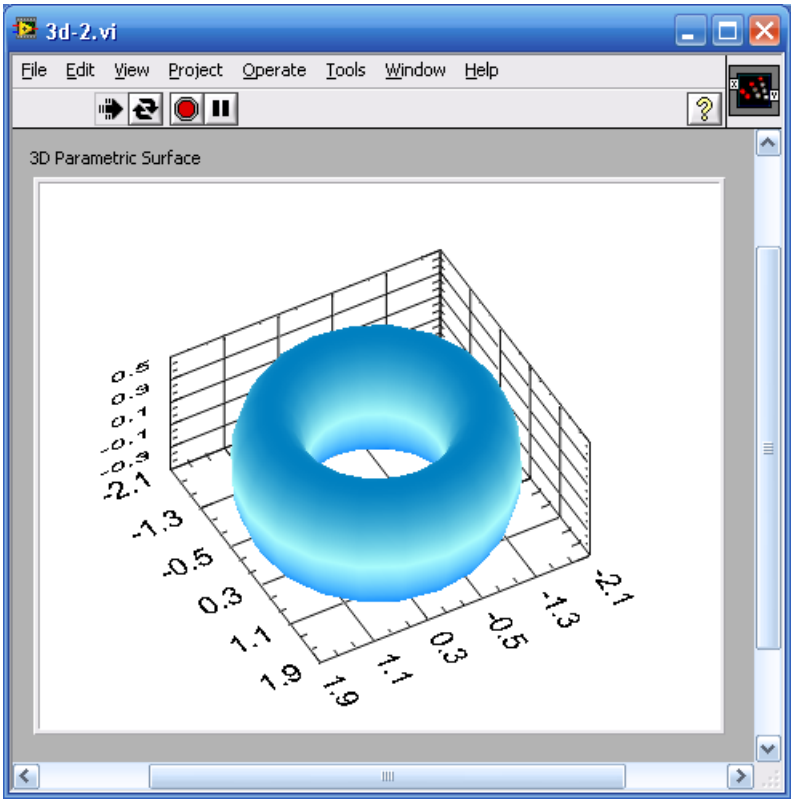


ภาพที่ 2.26 แสดงขั้นตอนการปรับแต่งสีกราฟสามมิติแบบ Custom



ภาพที่ 2.27 แสดงหน้าต่างสี Color Map Editor

3.2.6 จากนั้นทดสอบการวาดรูปกราฟ 3 มิติโดยการคลิกที่เมนู Run Continuously  จะแสดงรูปกราฟสามมิติพื้นผิวแบบมีสีของรูปทรงห่วงยาง (torus) ดังภาพที่ 2.28



ภาพที่ 2.28 แสดงผลลัพธ์การวาดรูปกราฟสามมิติพื้นผิวแบบมีสีของรูปทรงห่วงยาง (torus)

3.3 ตัวอย่างที่ 3 การวาดรูปกราฟสามมิติรูปสัญญาณฟังก์ชันซิงก์แบบ 3 มิติ (3D sinc function) ดังนี้

สูตร $y = \frac{\sin(d)}{d}$ และ $d = \sqrt{x^2 + y^2}$

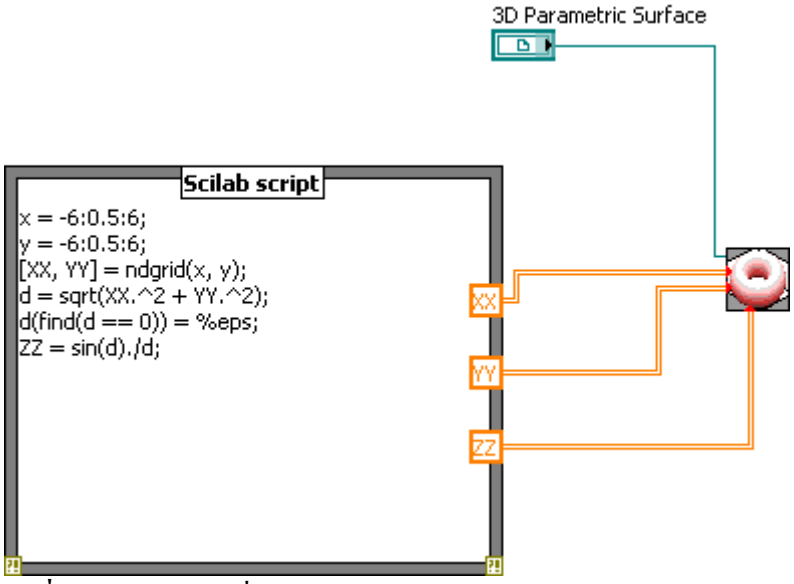
3.3.1 สำหรับส่วนของ Front Panel ให้เลือกตัวควบคุมสำหรับแสดงผลสัญญาณการวาดรูปกราฟ 3 มิติดังนี้ **Classic → Classic Graph → ActiveX 3D Parametric Graph** ดังภาพที่ 2.24 (ตัวอย่างที่ 2)

3.3.2 จากนั้นในส่วนของ Block Diagram ให้เลือกฟังก์ชัน Scilab script ขึ้นมาและเขียน code Scilab ดังนี้

```
x = -6:0.5:6;
y = -6:0.5:6;
[XX, YY] = ndgrid(x, y);
d = sqrt(XX.^2 + YY.^2);
d(find(d == 0)) = %eps;
ZZ = sin(d)./d;
```

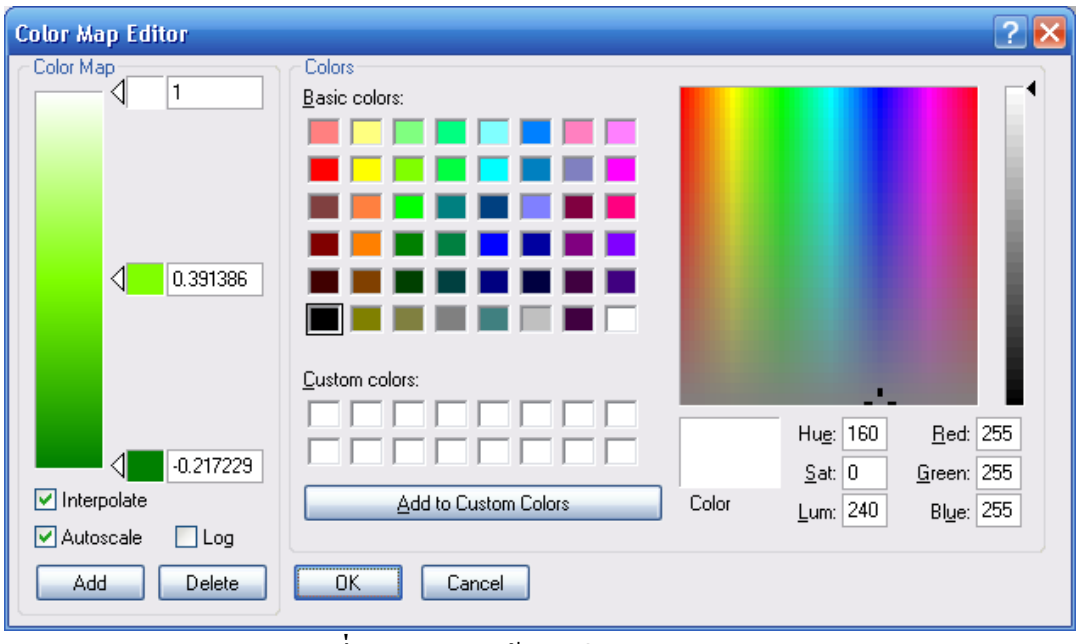
3.3.3 จากนั้นให้ Add Output ที่มีชื่อว่า x , y และ z โดย**ทุกตัวแปร**จะต้องเปลี่ยนชนิดของตัวแปรเป็น Array 2 มิติ โดยการคลิกขวาที่ตัวแปรเอาต์พุต z เลือก **Choose Data Type → 2-D Array of Complex** ดังภาพที่ 2.19 (ตัวอย่างที่ 1)

3.3.4 หลังจากนั้นทำการเชื่อมต่อ Wires แต่ละ Node เข้าด้วยกัน ดังภาพที่ 2.29



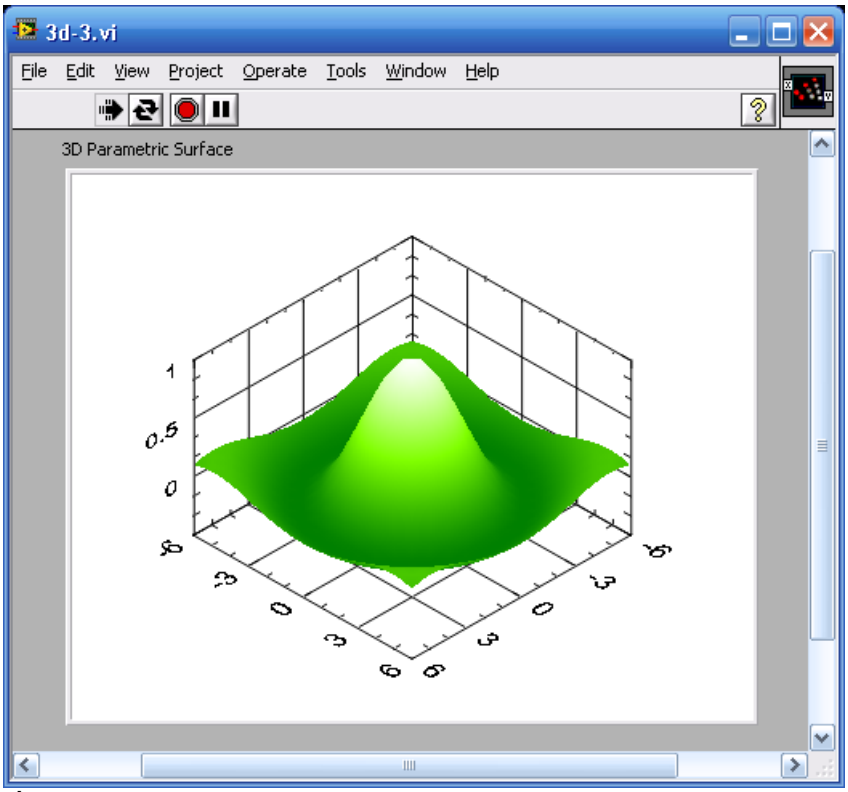
ภาพที่ 2.29 แสดงการเชื่อมต่อ Scilab script เข้ากับ 3D Parametric Surface

3.3.5 สำหรับการปรับเปลี่ยนสีกราฟ 3 มิติทำได้โดยการคลิกขวาที่ Control 3D Surface เลือก **CWGraph3D → Properties...** ดังภาพที่ 2.21 (ตัวอย่างที่ 1) หลังจากนั้น เลือกแท็บเมนู **Plots → Style** และในส่วนของ Color map style เลือกเป็น Custom และคลิกที่ปุ่ม Edit เพื่อเลือกสี ดังภาพที่ 2.26 (ตัวอย่างที่ 2) และเลือกสีที่ต้องการดังภาพที่ 2.30



ภาพที่ 2.30 แสดงหน้าต่างสี Color Map Editor

3.3.6 จากนั้นทดสอบการวาดรูปกราฟ 3 มิติโดยการคลิกที่เมนู Run Continuously  จะแสดงรูปกราฟสามมิติพื้นผิวแบบมีสีของรูปทรงห่วงยาง (torus) ดังภาพที่ 2.31



ภาพที่ 2.31 แสดงผลลัพธ์การวาดรูปกราฟสามมิติรูปสัญญาณพังก์ชันซิงก์แบบ 3 มิติ

REFRENCE

[1] รศ.ร.อ.ดร.กนต์ธร ชำนิประศาสน์, การวัดเชิงกลด้วย LabVIEW, มหาวิทยาลัยเทคโนโลยีสุรนารี.

[2] ผศ.ดร.ปิยะ โควินท์ทวีวัฒน์, คู่มือโปรแกรมภาษา SCILAB สำหรับผู้เริ่มต้น (พิมพ์ครั้งที่ 2), ศูนย์ผลิตตำราเรียนสถาบันเทคโนโลยีพระจอมเกล้าเจ้าพระนครเหนือ, 2549.

[3] www.scilab.org , LabVIEW to Scilab Gateway Version 1.1 for Windows, December 1, 2008