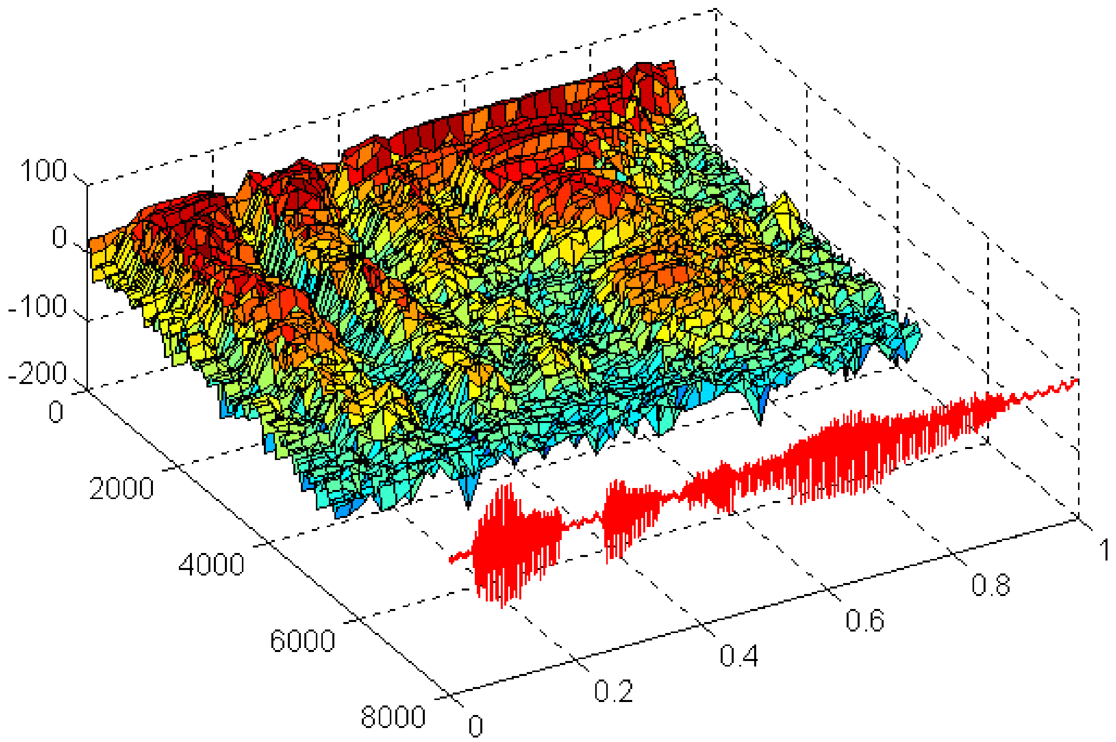
การประมวลผลสัญญาณดิจิตอลเบื้องต้น

Introduction to Digital Signal Processing

Revision 1.0

พรชัย ภววงษ์ศักดิ์



คำนำ

เทคโนโลยีทางด้านวิศวกรรมไฟฟ้า และคอมพิวเตอร์ในปัจจุบันได้ก้าวหน้าไปอย่างรวดเร็ว และบทบาทสำคัญของความก้าวหน้าเหล่านั้นก็มาจากการเปลี่ยนแปลงของเทคโนโลยีหลายอย่างที่เคยทำด้วยวงจรแอนะลอกมาเป็นวงจรดิจิตอล เรามักจะได้ยินคนพูดเสมอถึงเทคโนโลยีในโลกปัจจุบันว่าเป็น "ยุคแห่งดิจิตอล" หรือ "โลกดิจิตอล" ซึ่งผู้พูดส่วนใหญ่ก็มักมองจากสินค้าทันสมัยต่าง ๆ ที่ออกมาในท้องตลาด เช่น การมีคอมพิวเตอร์ที่มีความเร็วสูงขึ้น การมีระบบมัลติมีเดียดิจิตอลที่ดีขึ้น การเชื่อมต่อของระบบคอมพิวเตอร์ผ่านเครือข่ายอินเตอร์เน็ต การมีอุปกรณ์สื่อสาร เช่น โทรศัพท์มือถือและเพจเจอร์ที่ดีขึ้น และอุปกรณ์อื่น ๆ อีกมากมาย ซึ่งแน่นอนว่าอุปกรณ์ต่าง ๆ เหล่านี้มีพื้นฐานมาจากทฤษฎีของอิเล็กทรอนิกส์ และคอมพิวเตอร์ แต่ยังมีทฤษฎีอีกเรื่องหนึ่งที่เป็นพื้นฐาน และส่วนประกอบที่สำคัญของเทคโนโลยีเหล่านี้ นั่นก็คือ การประมวลผลสัญญาณดิจิตอล

วิศวกรไทยมักมองว่าการประมวลผลสัญญาณดิจิตอลเป็นทฤษฎีขั้นสูง และเป็นเรื่องไกลตัว ซึ่งก็มาจากการที่เรายังไม่มีการสอนเรื่องการประมวลผลสัญญาณดิจิตอลในระดับชั้นปริญญาตรีกันเท่าไรนัก บางมหาวิทยาลัยมีสอนเป็นวิชาเลือก ซึ่งก็ไม่ค่อยมีผู้เลือกเรียกมากนัก เนื่องจากมองว่าเป็นวิชาที่ยาก ซึ่งเต็มไปด้วยคณิตศาสตร์ กับทฤษฎีที่มองไม่เห็นภาพ แต่จริง ๆ แล้ว การประมวลผลสัญญาณดิจิตอลเป็นวิชาที่มีการประยุกต์ใช้งานอย่างกว้างขวางมาก ถ้าหากการสอนเน้นให้เห็นถึงการประยุกต์ใช้มากขึ้น หรือมีการทดลองประกอบ ก็จะทำให้ผู้เรียนมีความรู้สึกสนุก และเข้าใจในเนื้อหาวิชามากขึ้น ในปัจจุบันการประมวลผลสัญญาณดิจิตอลเป็นส่วนสำคัญที่เสริมกับเทคโนโลยีในสาขาวิชาต่าง ๆ ทั้งวิศวกรรมอิเล็กทรอนิกส์ วิศวกรรมโทรคมนาคม วิศวกรรมควบคุม วิศวกรรมไฟฟ้าแรงสูง และวิศวกรรมคอมพิวเตอร์ ดังนั้น วิศวกรในระดับชั้นปริญญาตรีจึงควรมีความเข้าใจในระดับพื้นฐานเกี่ยวกับการประมวลผลสัญญาณดิจิตอล

หนังสือ “การประมวลผลสัญญาณดิจิตอลเบื้องต้น” เล่มนี้ สั่งสมมาจากประสบการณ์ในการสอนวิชาการประมวลผลสัญญาณดิจิตอล ที่ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานครเป็นเวลาเกือบ 3 ปี และประสบการณ์ในการเปิดอบรมระยะสั้นให้กับบุคคลภายนอกซึ่งจัดโดยภาควิชาวิศวกรรมอิเล็กทรอนิกส์เช่นเดียวกัน ผู้เขียนมิได้มุ่งหวังให้หนังสือเล่มนี้มีความสมบูรณ์ครบถ้วนในเนื้อหา เนื่องจาก ทฤษฎีของการประมวลผลสัญญาณดิจิตอลมีมากมาย และมีรายละเอียดมากโดยเฉพาะในขั้นสูง แต่ผู้เขียนมุ่งหวังว่า หนังสือเล่มนี้จะเป็นคู่มือที่ดีสำหรับผู้ที่เริ่มต้นศึกษาวิชานี้ เนื่องจาก หนังสือทางด้านการประมวลผลสัญญาณเป็นภาษาอังกฤษแทบทั้งสิ้น และมักจะเต็มไปด้วยสมการทางคณิตศาสตร์มากมาย ซึ่งทำให้ยากลำบากสำหรับผู้ที่เริ่มต้นศึกษา ผู้เขียนได้พยายามลดความซับซ้อนเหล่านั้นลงเพื่อให้เหมาะสำหรับผู้เริ่มต้น อย่างไรก็ตาม เพื่อให้เข้าใจทฤษฎีอย่างถูกต้องผู้อ่านจำเป็นต้องมีความเข้าใจในคณิตศาสตร์พื้นฐานบ้าง ได้แก่ การแก้สมการ, โพลิโนเมียล, จำนวนเชิงซ้อน, และเวคเตอร์ เป็นต้น และควรมีประสบการณ์ด้านการเขียนโปรแกรมคอมพิวเตอร์มาบ้างไม่

ว่าจะเป็นภาษาอะไรก็ตาม นอกจากนี้ ผู้อ่านที่เคยศึกษาในบางวิชาของสาขาวิศวกรรมไฟฟ้า เช่น วงจรไฟฟ้า, สัญญาณและระบบ, ไมโครโปรเซสเซอร์ และวงจรดิจิตอล ก็จะพบว่า จะสามารถทำความเข้าใจในบทเรียนต่าง ๆ ได้ดีมากยิ่งขึ้น

การศึกษาตามหนังสือเล่มนี้จะให้ผลดีที่สุดก็ต่อเมื่อผู้อ่านได้มีการปฏิบัติตามไปด้วย ซึ่งซอฟท์แวร์ที่เหมาะสมที่สุดสำหรับทดลองประมวลผลสัญญาณดิจิตอล ก็คือ MATLAB (ของบริษัท Mathworks ประเทศสหรัฐอเมริกา) ซึ่งได้แนะนำวิธีใช้ไว้ในภาคผนวก ก ไฟล์ต่าง ๆ ที่แสดงในหนังสือเล่มนี้ รวมทั้งข้อมูลเพิ่มเติมอื่น ๆ ที่เกี่ยวข้อง สามารถดาวน์โหลดได้จากอินเตอร์เนตที่ http://www.ee.mut.ac.th/home/pornchai/ นอกจากนี้ คู่มือปฏิบัติการการประมวลผลสัญญาณดิจิตอล โดยภาควิชาวิศวกรรมอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานคร ก็เป็นแหล่งข้อมูลที่ดีสำหรับการทำการทดลองเพื่อเสริมความเข้าใจ โดยเป็นการทดลองที่ใช้ MATLAB และการทดลองแบบเวลาจริงโดยใช้ชิพ DSP (TMS320C5x)

ผู้เขียนขออุทิศหนังสือนี้ให้ผู้อ่านที่มีความสนใจทั่วไป ผู้อ่านสามารถทำสำเนาได้โดยอิสระตราบใดที่ไม่เป็นไปเพื่อการค้า ไม่มีการแก้ไขเนื้อหาในหนังสือ และมีการรวมคำนำนี้อยู่ในสำเนาด้วย ความดีทั้งหลายหากมีอยู่ ขอมอบให้ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานคร ซึ่งเป็นแหล่งที่ได้ให้ความรู้ และการสนับสนุนในการเขียนหนังสือเล่มนี้ หากมีข้อผิดพลาดประการใด ผู้เขียนขออภัยไว้ ณ ที่นี้ด้วย และยินดีรับคำแนะนำเพื่อมาปรับปรุงแก้ไขต่อไป (กรุณาส่งจดหมายอิเล็กทรอนิกส์ไปที่ pawa@ziplip.com) อย่างไรก็ตาม ผู้เขียนไม่รับผิดชอบหากมีความเสียหายใด ๆ เกิดขึ้นจากการใช้ความรู้ หรือข้อมูลที่ได้รับจากหนังสือเล่มนี้

พรชัย ภววงษ์ศักดิ์
เนื้อหาแก้ไขเมื่อ 12/1999
คำนำแก้ไขเมื่อ 10/2000

สารบัญ

บทที่ 1 บทนำ	**1**
สัญญาณต่อเนื่อง กับสัญญาณไม่ต่อเนื่อง	1
ส่วนประกอบในระบบประมวลผลสัญญาณดิจิตอล	3
การประมวลผลแบบเวลาจริง กับการเลือกใช้ตัวประมวลผลสัญญาณ	5
งานที่มีการประยุกต์ใช้การประมวลผลสัญญาณดิจิตอล	8
ข้อดีของการใช้การประมวลผลสัญญาณดิจิตอล	9
ขีดจำกัดของการประมวลผลสัญญาณดิจิตอล	10
บทที่ 2 การสุ่มสัญญาณและการสร้างสัญญาณคืน	**11**
การสุ่มสัญญาณ	11
ทฤษฎีการสุ่มสัญญาณ	14
การสร้างสัญญาณคืน	16
บทที่ 3 ระบบแบบไม่ต่อเนื่อง	**25**
ระบบแบบไม่ต่อเนื่องคืออะไร	25
ความเป็นเชิงเส้นและไม่แปรตามเวลา	26
ผลตอบสนองต่อสัญญาณอิมพัลส์	29
ตัวกรองแบบ FIR และ IIR	32
สมการผลต่าง	33
ความเป็นคอซัล	35
เสถียรภาพ	39
บทที่ 4 การแปลง z และการประยุกต์ใช้กับระบบแบบไม่ต่อเนื่อง	**41**
การแปลง z	41
การแปลง z โดยใช้สูตรสำเร็จจากตาราง (สำหรับสัญญาณคอซัล)	44
การแปลง z ผกผัน	47
การใช้การแปลง z กับระบบแบบไม่ต่อเนื่อง	51
ความเป็นคอซัล และเสถียรภาพ	55
บทที่ 5 การแปลง DTFT และผลตอบสนองเชิงความถี่	**59**
การแปลงฟูริเยร์แบบเวลาไม่ต่อเนื่อง หรือการแปลง DTFT	59
สัญญาณไม่ต่อเนื่องความถี่เดี่ยว	62
ความสัมพันธ์ของ DTFT กับการแปลง z	63

ผลตอบสนองเชิงความถี่ของระบบ 64

บทที่ 6 การแปลง DFT และ FFT **69**

ทบทวนการแปลงแบบต่าง ๆ 69
การแปลง DFT 70
ที่มา และความหมายของการแปลง DFT 72
การเติมศูนย์ 74
สเปกตรัมของพลังงาน กับสเปกตรัมของกำลัง 75
การแปลง FFT 77
การแปลง DFT ผกผัน 84
คุณสมบัติของ DFT 85
เครื่องวิเคราะห์สเปกตรัม และการปรับปรุงผลลัพธ์ที่ได้จาก FFT 85
การคอนโวลูชันอย่างเร็ว 89

บทที่ 7 ตัวกรองแบบ FIR **95**

คุณสมบัติเฟสแบบเชิงเส้น 95
คุณสมบัติความสมมาตรของตัวกรองที่มีเฟสเชิงเส้น 98
การออกแบบโดยวิธีหน้าต่าง 101
หน้าต่างแบบสี่เหลี่ยม 107
หน้าต่างแฮมมิ่ง 109
หน้าต่างไคเซอร์ 111
การออกแบบโดยวิธีสุ่มความถี่ 116
การสร้างตัวกรอง FIR 121

บทที่ 8 ตัวกรองแบบ IIR **123**

การออกแบบโดยอิงตัวกรองแอนะลอกต้นแบบ 123
ตัวกรองบัตเตอร์เวิร์ธแอนะลอกต้นแบบ 128
การออกแบบตัวกรองบัตเตอร์เวิร์ธดิจิตอลผ่านต่ำ 130
การออกแบบตัวกรองบัตเตอร์เวิร์ธแบบอื่น (นอกจากผ่านต่ำ) 130
การออกแบบโดยวิธีวางโพล และศูนย์ 138
การสร้างตัวกรอง IIR 144
เปรียบเทียบตัวกรอง FIR และ IIR 147

บทที่ 9 ระบบตัวเลขในการประมวลผล **149**

ระบบเลขจำนวนเต็ม 149
ระบบเลขจำนวนเต็มแบบมีเครื่องหมาย 149
ข้อดีของเลขแบบ 2's complement 156

ระบบเลขอิงดรรชนี	158
เปรียบเทียบเลขจำนวนเต็ม กับเลขอิงดรรชนี	164
บทที่ 10 ความคลาดเคลื่อนจากการใช้ระบบเลขจำนวนเต็ม	**165**
การแบ่งขั้นสัญญาณ	165
ความคลาดเคลื่อนจากการปัดเศษสัมประสิทธิ์	171
ความคลาดเคลื่อนจากโอเวอร์โฟล	174
ความคลาดเคลื่อนจากการปัดเศษหลังการคูณ	185
การจำลองการประมวลผลด้วยระบบเลขจำนวนเต็มใน Matlab	196
ตัวอย่างการประมวลผลด้วยระบบเลขจำนวนเต็มในภาษาซี	202
บทที่ 11 การประมวลผลแบบหลายอัตราสุ่ม	**205**
การเปลี่ยนอัตราสุ่มโดยแปลงเป็นสัญญาณแอนะลอกก่อน	205
เดซิเมเตอร์	206
อินเตอร์โพเลเตอร์	208
การเปลี่ยนอัตราสุ่มด้วยอัตราส่วนที่ไม่เป็นจำนวนเต็ม	211
การเปลี่ยนอัตราสุ่มแบบหลายขั้นตอน	212
การลดการประมวลผลของตัวเปลี่ยนอัตราสุ่ม	215
การประยุกต์ใช้งาน	217
บทที่ 12 ตัวอย่างการประยุกต์ใช้งาน	**222**
การปรับแต่งลักษณะของเสียง	222
อีควอไลเซอร์เสียง	224
เสียงสะท้อน	225
เสียงจำลองการสะท้อนของห้อง	226
เสียงคอรัส	231
เสียงสามมิติ	231
อุปกรณ์ช่วยได้ยิน	233
การกรองสัญญาณฮาร์มอนิกใน UPS	234
ตัวสร้างสัญญาณ	235
ตัวกรองปรับตัวได้	236
อัลกอริธึม LMS และเงื่อนไขการทำงานของมัน	241
การหักล้างเสียงสะท้อน	246
ภาคผนวก ก การใช้งาน Matlab	**249**
เริ่มรู้จักกับ Matlab ในฐานะเป็นเครื่องคิดเลข	249

การใช้ตัวแปรใน Matlab 250
การเรียกคำสั่งเก่ามาใช้ใหม่ 250
เมตริกซ์ และเวคเตอร์ 251
การกระทำทางเมตริกซ์ 251
การกระทำที่เข้าถึงสมาชิกทุกตัวในเมตริกซ์ 252
การอ้างถึงสมาชิกภายในเมตริกซ์ (หรือเวคเตอร์) 252
จำนวนเชิงซ้อน 253
ฟังก์ชั่นภายใน 254
โปรแกรมสคริปต์ 255
คำสั่งเกี่ยวกับการวนลูป และเปรียบเทียบ 256
โปรแกรมฟังก์ชั่น 258
การจัดการเกี่ยวกับไดเรกทอรี่ และไฟล์ 260
การวาดกราฟ 260
การวัดประสิทธิภาพของโปรแกรม 263
การเก็บตัวแปร 263
การจัดการเกี่ยวกับข้อความ 263
คำสั่งเกี่ยวกับเสียง 264
การทำกราฟที่เคลื่อนไหวได้ 265

ภาคผนวก ข ฟังก์ชั่นใน Matlab DSP Toolbox 266

ภาคผนวก ค ประมวลคำศัพท์เทคนิค 269

หนังสืออ้างอิง 272

บทที่ 1

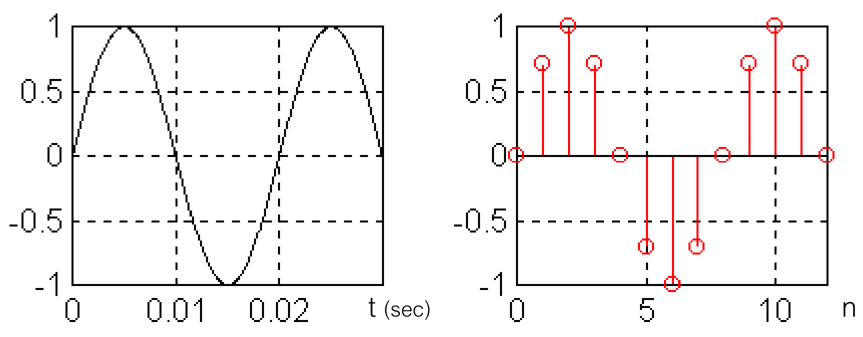
บทนำ

ในบทนี้จะได้กล่าวถึงภาพรวมของการประมวลผลสัญญาณดิจิตอล เพื่อให้ผู้อ่านได้ทราบถึงส่วนต่าง ๆ ในระบบประมวลผลสัญญาณ และเข้าใจว่าสิ่งที่จะศึกษาในบทต่อ ๆ ไปอยู่ในส่วนใดของระบบโดยรวม นอกจากนี้ยังจะกล่าวถึงข้อดี และงานที่มีการประยุกต์ใช้การประมวลผลสัญญาณดิจิตอลด้วย

สัญญาณต่อเนื่อง กับสัญญาณไม่ต่อเนื่อง

สิ่งแรกที่จะต้องทำความเข้าใจกันก่อนก็คือ คำว่า สัญญาณต่อเนื่อง (continuous-time signal) และสัญญาณไม่ต่อเนื่อง (discrete-time signal) หมายความว่าอย่างไร คำว่าต่อเนื่อง หรือไม่ต่อเนื่องนี้หมายถึงสัญญาณนั้น ๆ มีค่าต่อเนื่องในทางเวลาหรือไม่ สัญญาณต่อเนื่อง ก็คือ สัญญาณที่เราพบเห็นในชีวิตประจำวันทั่ว ๆ ไป หรือที่เห็นบนหน้าจอออสซิโลสโคป เช่น สัญญาณเสียง, สัญญาณไฟบ้าน 50 Hz, และอื่น ๆ ถ้าแทนสัญญาณด้วยสัญลักษณ์ x และแทนเวลาด้วยสัญลักษณ์ t เราจะกล่าวว่า x เป็นฟังก์ชั่นของ t หรือ x มีค่าที่เวลา t ใด ๆ เขียนแทนสัญญาณนี้ได้ว่า x(t) ซึ่งเป็นฟังก์ชั่นที่ต่อเนื่อง สัญญาณต่อเนื่องนี้เรียกอีกอย่างหนึ่งว่า สัญญาณแอนะลอก (analog signal)

สัญญาณไม่ต่อเนื่องเป็นสัญญาณที่มีค่าเพียงบางจุดของเวลา โดยทั่วไปเกิดจากการสุ่มสัญญาณต่อเนื่องด้วยคาบเวลาของการสุ่มคงที่ ดังจะได้กล่าวถึงในบทที่ 2 เราจะใช้สัญลักษณ์ n แทนเวลาแบบไม่ต่อเนื่อง โดย n เป็นตัวแปรที่มีค่าเป็นจำนวนเต็มเท่านั้น คือ n = …, -2, -1, 0, 1, 2, 3, … และสัญญาณไม่ต่อเนื่องจะเป็นฟังก์ชั่นของ n ดังนั้นจะเขียนแทนสัญญาณนี้ได้ว่า x(n)

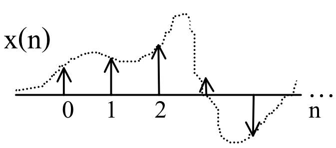


รูปที่ 1.1 สัญญาณต่อเนื่อง และสัญญาณไม่ต่อเนื่อง (โปรดสังเกตหน่วยของแกนนอนด้วย)

รูปของสัญญาณไม่ต่อเนื่องที่แสดงเปรียบเทียบกับสัญญาณต่อเนื่องดังในรูปที่ 1.1 เป็นรูปที่นิยมเขียนเพื่อแสดงให้เห็นรูปร่างของสัญญาณ แต่จริง ๆ แล้ว เราจะไม่สามารถเห็นสัญญาณนี้ได้โดยตรงเหมือนกับที่เห็นสัญญาณแอนะลอกในออสซิโลสโคป แต่เราจะมองสัญญาณไม่ต่อเนื่องในลักษณะของ "<u>ลำดับของค่า</u>" หรือ "<u>ลำดับของข้อมูล</u>" โดยข้อมูลแต่ละตัวก็แทนค่าแต่ละค่าของสัญญาณนั่นเอง ซึ่งลำดับของค่าเหล่านี้เป็นสิ่งที่วิเศษ คือ นอกจากมันเป็นตัวแทนที่ถูกต้องของสัญญาณต่อเนื่องที่ถูกสุ่มมาแล้ว มันยังสามารถถูกนำไปประมวลผลได้ด้วยคอมพิวเตอร์ หรือวงจรทางดิจิตอลได้ด้วย ในอดีต ได้เคยมีผู้คิดว่า การประมวลผลสัญญาณดิจิตอลให้ผลลัพธ์เป็นเพียงการประมาณของการประมวลผลทางแอนะลอก แต่ด้วยทฤษฎีที่ถูกต้องที่ได้มีการคิดค้นกันมา ทำให้การประมวลผลทางดิจิตอลให้ผลลัพธ์ที่ถูกต้อง และสามารถพิสูจน์ได้ว่าไม่ใช่ค่าประมาณของทางแอนะลอก

ส่วนคำว่าสัญญาณดิจิตอล กับสัญญาณไม่ต่อเนื่องนั้น โดยทั่วไปหมายถึงสัญญาณในลักษณะเดียวกัน แต่มีความหมายต่างกันเล็กน้อยตามความรู้สึก และประสบการณ์ของผู้พูด เมื่อพูดถึงคำว่าสัญญาณไม่ต่อเนื่อง เราหมายถึง ลำดับของข้อมูลดังที่ได้กล่าวมาแล้ว ซึ่งข้อมูลแต่ละตัวนั้นอาจมีขนาดเท่าไรก็ได้โดยไม่จำกัดความละเอียด ซึ่งค่าของสัญญาณนี้ เมื่อนำไปใช้งานจริงก็จะถูกแทนด้วยค่าดิจิตอลที่มีจำนวนบิตจำกัด เช่น ในรูปที่ 1.2 ได้แสดงให้เห็นว่าสัญญาณ x(n) แต่ละค่า เมื่อนำไปใช้สามารถแทนได้ด้วย 8 บิต เป็นต้น เมื่อพูดถึงคำว่าสัญญาณดิจิตอล เรามักหมายถึง สัญญาณไม่ต่อเนื่องที่แต่ละค่าของสัญญาณถูกแทนค่าด้วยเลขฐานสองที่มีจำนวนบิตจำกัด อยู่ในรูป 0 กับ 1 แล้ว

ทฤษฎีของการประมวลผลสัญญาณดิจิตอลที่เราจะได้ศึกษาต่อไป ถึงแม้ชื่อที่คนนิยมเรียกจะเรียกว่า การประมวลผล "สัญญาณดิจิตอล" (Digital Signal Processing) แต่ถ้าดูความหมายที่แท้จริงของทฤษฎีแล้ว น่าจะเรียกว่า การประมวลผลสัญญาณไม่ต่อเนื่อง (Discrete-time Signal Processing) มากกว่า หนังสือบางเล่มก็ใช้คำนี้แทนเสียเลย ทั้งนี้เพราะว่า ทฤษฎีของการประมวลผลสัญญาณเป็นการกระทำโดยมองสัญญาณขาเข้าเป็นลักษณะของลำดับของข้อมูล (ซึ่งคือสัญญาณไม่ต่อเนื่อง) โดยนำข้อมูลเหล่านี้มาประมวลผล เช่น บวก ลบ คูณ หาร เพื่อหาสัญญาณขาออกในลักษณะเป็นลำดับข้อมูลเช่นเดียวกัน



$\equiv$ x(0), x(1), x(2), x(3), x(4), …

1.21, 1.33, 1.74, 0.45, -1.56, …

ค่าของสัญญาณเหล่านี้ เมื่อนำไปใช้งานสามารถแทนได้ด้วยข้อมูลดิจิตอล เช่น 00110110, 00111000, 01100011, …

รูปที่ 1.2 *สัญญาณไม่ต่อเนื่อง ก็คือ ลำดับของข้อมูลนั่นเอง*

กล่าวอีกนัยหนึ่งก็คือ วิชานี้จะศึกษาถึง “อัลกอริธึม” ในการประมวลผลข้อมูลที่เป็นสัญญาณนั่นเอง ไม่ใช่เป็นการศึกษาการใช้ลอจิกเกต หรือฟลิปฟลอปต่าง ๆ มาประมวลผลสัญญาณดิจิตอลที่เป็น 0 กับ 1 แต่อย่างใด ทฤษฎีส่วนหลังนี้เป็นเรื่องของการออกแบบวงจรดิจิตอล หรือการออกแบบระบบดิจิตอล (Digital System Design) ซึ่งถือว่าอยู่ในระดับของการนำไปใช้งาน (implementation) แล้ว เช่น สมมติเรามีอัลกอริธึมหนึ่งที่จะใช้ในการประมวลผลสัญญาณ ทฤษฎีที่บอกว่า อัลกอริธึมนี้กระทำผลอะไรกับสัญญาณ ดีมากน้อยแค่ไหน นี่เป็นเรื่องของการประมวลผลสัญญาณดิจิตอล แต่ถ้าจะนำอัลกอริธึมนี้ไปใช้งานโดยทำเป็นวงจรดิจิตอล เมื่อนั้น จึงเป็นหน้าที่ของวิชาการออกแบบระบบดิจิตอล หรือถ้าจะนำอัลกอริธึมไปใช้งานโดยเขียนเป็นซอฟท์แวร์ก็ได้ ซึ่งเมื่อนั้น ก็จะต้องใช้ความรู้เรื่องการเขียนโปรแกรมคอมพิวเตอร์ เป็นต้น ขอให้ผู้เริ่มต้นทำความเข้าใจในภาพรวมต่าง ๆ เหล่านี้ให้ดี

ส่วนประกอบในระบบประมวลผลสัญญาณดิจิตอล

ระบบประมวลผลสัญญาณโดยส่วนใหญ่ แสดงในรูปที่ 1.3 ซึ่งประกอบด้วยส่วนต่าง ๆ ดังต่อไปนี้

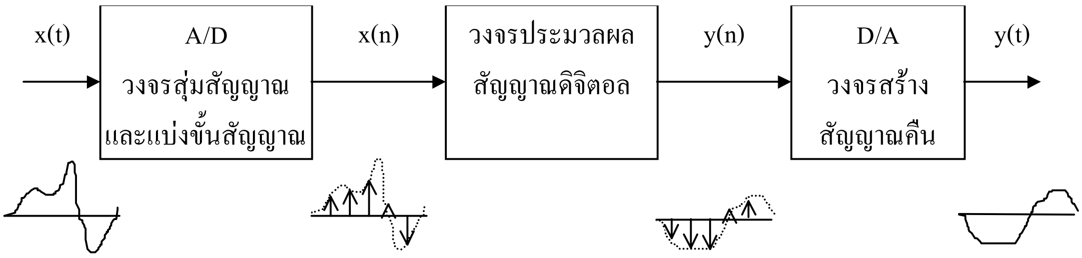
1. **<u>วงจรแปลงสัญญาณแอนะลอกเป็นดิจิตอล</u>** ซึ่งสามารถแบ่งได้เป็น 2 กระบวนการย่อย คือ

1.1 วงจรสุ่มสัญญาณ (Sampler) สัญญาณขาเข้าของวงจรนี้เป็นสัญญาณแบบแอนะลอก $x(t)$ ส่วนสัญญาณขาออกเป็นสัญญาณไม่ต่อเนื่อง $x(n)$ พารามิเตอร์วงจรสุ่มสัญญาณนี้ก็คือ ค่าอัตราการสุ่ม (sampling rate) หรือ ความถี่ในการสุ่ม ใช้สัญลักษณ์แทนว่า f_s ค่านี้เป็นตัวกำหนดว่า วงจรสุ่มจะสุ่มสัญญาณด้วยอัตรากี่ครั้งต่อวินาที หรือกี่เฮริซ์ (Hz) เราจะศึกษาหลักการของการสุ่มสัญญาณในบทที่ 2

1.2 วงจรแบ่งขั้นสัญญาณ (Quantizer) สัญญาณ $x(n)$ ที่ได้จากวงจรสุ่มสัญญาณถือว่ามีความละเอียด (นัยสำคัญ) เต็มที่ในทางขนาด ซึ่งในทางปฏิบัติเมื่อนำไปใช้งานจะต้องลดความละเอียดของ $x(n)$ ลงให้สามารถแทนได้ด้วยสัญญาณดิจิตอลที่มีจำนวนบิตจำกัด กระบวนการลดความละเอียดนี้ เรียกว่า การแบ่งขั้นของสัญญาณ (quantization) ความละเอียดที่ได้จากการแบ่งขั้นสัญญาณขึ้นอยู่กับจำนวนบิตที่จะใช้

การแบ่งขั้นสัญญาณทำให้ค่าสัญญาณที่ได้คลาดเคลื่อนไปจาก $x(n)$ จริง ซึ่งจะส่งผลเหมือนมีสัญญาณรบกวนเข้ามาในระบบ ในเนื้อหาของบทต่อ ๆ ไป เราจะละเลยผลของการแบ่งขั้นสัญญาณนี้ชั่วคราว และถือเอาว่าสัญญาณ $x(n)$ เป็นสัญญาณขาเข้าของวงจรประมวลผลสัญญาณเลย อย่างไรก็ตาม เราจะกลับมาศึกษาหลักการของการแบ่งขั้นสัญญาณ และผลของความคลาดเคลื่อนที่เกิดขึ้นนี้อีกครั้งในบทที่ 10

วงจรสุ่มสัญญาณรวมกับวงจรแบ่งขั้นสัญญาณ ในทางปฏิบัติก็คือ ตัวแปลงสัญญาณแอนะลอกเป็นดิจิตอล (A/D converter) นั่นเอง ซึ่งจะรวมสองกระบวนการนี้อยู่ในวงจรเดียวกัน และโดยทั่วไป เราจะใช้ตัวแปลงสัญญาณแอนะลอกเป็นดิจิตอลในรูปของวงจรรวมสำเร็จรูป (IC)



***รูปที่ 1.3** ส่วนประกอบในระบบประมวลผลสัญญาณดิจิตอล*

2. **<u>วงจรประมวลผลสัญญาณ</u>** ส่วนนี้เป็นหัวใจหลักที่เราจะศึกษาในวิชานี้ ซึ่งทำหน้าที่ประมวลผลสัญญาณ x(n) เพื่อกระทำผลบางอย่างกับสัญญาณ เช่น เป็นวงจรกรองความถี่บางย่านออก และให้ผลลัพธ์ของการประมวลผลเป็นสัญญาณขาออก y(n) วงจรประมวลผลสัญญาณนี้ ถ้าจะพิจารณากันอย่างง่าย ๆ แท้ที่จริงก็คือ ตัวคำนวณนั่นเอง กล่าวได้ว่า มันกระทำการคำนวณหาสัญญาณขาออกจากสัญญาณขาเข้า โดยมองเห็นสัญญาณขาเข้าในลักษณะ ”ลำดับของค่า”

ตัวอย่างการประมวลผลง่าย ๆ เช่น ถ้าเรามีสมการสำหรับการประมวลผล คือ

$$y(n) = 0.5(x(n) + x(n-1)) \qquad (1.1)$$

ถ้าพิจารณาในแง่การคำนวณ สมการนี้บอกว่า ผลตอบ ณ ตำแหน่ง n ใด ๆ สามารถหาได้ด้วยการเอาสัญญาณขาเข้าที่ตำแหน่งเวลาเดียวกัน (x(n)) บวกเข้ากับสัญญาณขาเข้าที่ตำแหน่งเวลาก่อนหน้านั้น 1 ตำแหน่ง (x(n-1)) เสร็จแล้วเอาผลบวกที่ได้คูณด้วย 0.5 ยกตัวอย่างเช่น

ที่ n=3 ตัวประมวลผลจะคำนวณหา y(3) โดย $y(3) = 0.5(x(3) + x(2))$

ที่ n=4 ตัวประมวลผลจะคำนวณหา y(4) โดย $y(4) = 0.5(x(4) + x(3))$

ที่ n=5 ตัวประมวลผลจะคำนวณหา y(5) โดย $y(5) = 0.5(x(5) + x(4))$

เป็นเช่นนี้ไปเรื่อย ๆ ดังนั้น ทุก ๆ ตำแหน่งเวลา หน้าที่ของตัวประมวลผลสัญญาณ ก็คือคำนวณหาสัญญาณขาออกตามสมการนี้เท่านั้นเอง

ผู้อ่านคงพอมองเห็นภาพพจน์ของการประมวลผลสัญญาณแล้วว่า จริง ๆ แล้วมันก็คือการคำนวณนั่นเอง แต่สิ่งที่เราจะศึกษาในบทต่อ ๆ ไป คือว่า การคำนวณเหล่านี้จะกระทำผลอะไรให้เกิดขึ้นกับสัญญาณได้บ้าง เช่น ตัวอย่างง่าย ๆ ที่ยกมาในสมการที่ 1.1 นี้ เป็นสมการของตัวกรองแบบผ่าน

ความถี่ต่ำ (low pass filter) อันดับหนึ่ง ชนิดหนึ่ง กล่าวคือ มันจะลดองค์ประกอบความถี่สูงของสัญญาณลงบางส่วน ซึ่งเราจะสามารถพิสูจน์ได้โดยใช้ความรู้ในบทต่อ ๆ ไป

ในชีวิตจริง อัลกอริธึมในการประมวลผลสัญญาณมีตั้งแต่ง่าย ๆ ดังที่แสดงในสมการที่ 1.1 จนกระทั่งถึงยากมาก หรือ ซับซ้อนมาก ๆ ซึ่งมันก็จะสามารถส่งผลที่พิสดารขึ้นกับสัญญาณได้ การคิดค้นในเรื่องของอัลกอริธึมในการประมวลผลสัญญาณนี้ ถือเป็นสาขาที่มีผู้วิจัยกันอย่างกว้างขวางและต่อเนื่องในปัจจุบัน ซึ่งถึงแม้ความรู้ในด้านนี้จะถูกพัฒนามาหลายสิบปี และเจริญก้าวหน้ามามาก แต่ก็ยังเติบโตต่อไปอย่างไม่เห็นแนวโน้มในการอิ่มตัวของมันเลย

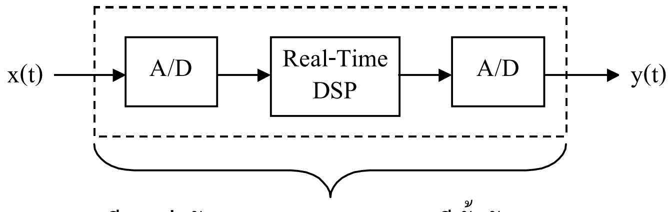
ขออธิบายเพิ่มเติมจากสมการตัวอย่างที่ได้ยกมาแล้วว่า ข้อมูลที่เราสามารถนำมาใช้ในอัลกอริธึมของการประมวลผลนี้ ได้แก่

- สัญญาณขาเข้าตัวปัจจุบัน คือ x(n)
- สัญญาณขาเข้าในอดีต คือ x(n-1), x(n-2), x(n-3), …
- สัญญาณขาเข้าในอนาคต (รับมาล่วงหน้า) คือ x(n+1), x(n+2), x(n+3), …
- สัญญาณขาออกในอดีต (ได้คำนวณไปแล้ว) คือ y(n-1), y(n-2), y(n-3), … เป็นต้น

3. <u>วงจรสร้างสัญญาณคืน (Signal Reconstruction)</u> ใช้ในระบบที่มีสัญญาณขาออกสุดท้ายเป็นสัญญาณต่อเนื่อง (การประมวลผลสัญญาณบางอย่าง ต้องการสัญญาณขาออกเป็นไม่ต่อเนื่อง ก็ไม่จำเป็นต้องมีส่วนที่ 3 นี้) โดยทำหน้าที่แปลงสัญญาณไม่ต่อเนื่อง y(n) ให้กลับเป็นสัญญาณต่อเนื่อง y(t) ซึ่งจะเป็นสัญญาณขาออกสุดท้ายของระบบ วงจรประเภทนี้ก็คือ ตัวแปลงสัญญาณดิจิตอลเป็นแอนะลอก (D/A converter) นั่นเอง ซึ่งก็มีในรูปแบบวงจรรวมสำเร็จรูปเช่นกัน

การประมวลผลแบบเวลาจริง กับการเลือกใช้ตัวประมวลผลสัญญาณ

การประมวลผลแบบเวลาจริง (Real-Time Signal Processing) หมายถึง การประมวลผลที่กระทำที่อัตราจริงของสัญญาณขาเข้า และให้สัญญาณขาออกทันกับสัญญาณขาเข้าที่เข้ามา เช่น ในระบบที่มีอัตราการสุ่มของสัญญาณขาเข้า และขาออกเท่ากัน เมื่อมีสัญญาณขาเข้าเข้ามา 1 ค่า ระบบจะต้องประมวลผลให้ได้สัญญาณขาออก 1 ค่าก่อนที่สัญญาณขาเข้าตัวถัดไปจะเข้ามา เป็นต้น การประมวลผลแบบเวลาจริงนี้มีการประยุกต์ใช้งานอย่างมาก และเป็นตัวแทนที่แท้จริงของระบบที่เคยใช้เป็นแบบแอนะลอกดังแสดงในรูปที่ 1.4 อย่างไรก็ตาม ระบบที่มีการประมวลผลแบบเวลาจริงไม่จำเป็นต้องมีสัญญาณขาเข้า และออกเป็นสัญญาณแอนะลอกทั้งคู่เสมอไป ยกตัวอย่างเช่น การถอดรหัสสัญญาณเสียงที่ถูกบีบอัดข้อมูลมา ในกรณีนี้สัญญาณขาเข้าเป็นดิจิตอล ซึ่งคือข้อมูลเสียงที่บีบอัดมาแล้ว ส่วนสัญญาณขาออก คือ สัญญาณเสียงแอนะลอกที่ต้องส่งออกที่ลำโพง ดังนั้น การประมวลผลจะต้องเกิดที่อัตราสุ่มจริงของสัญญาณเสียงขาออก อันนี้ก็ถือว่า เป็นการประมวลผลแบบเวลาจริง



รูปที่ 1.4 *การประมวลผลแบบเวลาจริงทำให้ DSP ทำหน้าที่เหมือนเป็นวงจรแอนะลอกได้*

ส่วนการประมวลผลแบบไม่เป็นเวลาจริงนั้นไม่มีข้อบังคับทางด้านเวลาในการประมวลผล ยกตัวอย่างเช่น การจำลองระบบประมวลผลด้วย MATLAB ในคอมพิวเตอร์ ในที่นี้ถือว่าคอมพิวเตอร์เป็นตัวประมวลผล ซึ่งถ้าใช้คอมพิวเตอร์ที่เร็วเราก็ได้ผลลัพธ์เร็ว แต่ถ้าใช้คอมพิวเตอร์ที่ช้าเราก็จะได้ผลลัพธ์ช้า แต่ผลลัพธ์ที่ได้ไม่แตกต่างกันเลยไม่ว่าจะเร็วหรือช้า ทั้งนี้เพราะการประมวลผลไม่ได้เกิดขึ้นที่อัตราการสุ่มจริงของสัญญาณขาเข้า หรือขาออก ตัวอย่างอีกอันหนึ่ง เช่น การใช้โปรแกรมพวกตบแต่งรูปภาพ (ภาพนิ่ง) เช่น PhotoShop ซึ่งภาพนิ่งนี้ถือเป็นสัญญาณไม่ต่อเนื่องสองมิติ และโปรแกรมพวกนี้ก็ถือเป็นโปรแกรมที่มีฟังก์ชั่นในการประมวลผลภาพ (Image Processing) เนื่องจากภาพนิ่งไม่มีอัตราการสุ่มของข้อมูลที่เทียบต่อเวลา ดังนั้น การประมวลผลภาพนิ่งจึงถือได้ว่าไม่มีข้อบังคับทางด้านเวลา (ถ้าไม่เอาอารมณ์ของผู้ใช้มาเป็นเกณฑ์ด้วย) จึงไม่เป็นการประมวลผลแบบเวลาจริง

การประมวลผลสัญญาณแบบเวลาจริงทำให้เกิดข้อกำหนดที่สำคัญขึ้นมาต่อการเลือกใช้ตัวประมวลผลสัญญาณ นั่นคือ การที่ต้องมีตัวประมวลผลที่เร็วพอที่จะประมวลผลสัญญาณให้ทันได้ โดยเฉพาะอย่างยิ่ง ถ้าสัญญาณที่ต้องการประมวลผลมีอัตราการสุ่มที่สูง หรืออัลกอริธึมที่ใช้มีความซับซ้อนในการคำนวณมาก ก็จำเป็นที่จะต้องใช้ตัวประมวลผลที่มีความเร็วสูงมากยิ่งขึ้น

มีทางเลือกใหญ่ ๆ อยู่ 3 ทางในการทำตัวประมวลผล คือ

1) การเขียนซอฟท์แวร์เพื่อใช้กับคอมพิวเตอร์ หรือใช้กับชิพไมโครโปรเซสเซอร์ทั่ว ๆ ไป ซึ่งถึงแม้ว่าคอมพิวเตอร์ หรือไมโครโปรเซสเซอร์จะไม่ได้ออกแบบมาเฉพาะสำหรับการประมวลผลสัญญาณ แต่เราก็สามารถนำมันมาใช้ได้ในงานที่ต้องการอัตราการประมวลผลไม่มากนัก หรือในการประมวผลผลแบบไม่เป็นเวลาจริง อย่างไรก็ตาม ปัจจุบันคอมพิวเตอร์ส่วนบุคคลมีความเร็วสูงมากจนสามารถนำมาใช้ทำการประมวลผลแบบเวลาจริงหลาย ๆ อย่างได้ ตัวอย่างที่เห็นได้ชัด เช่น การถอดรหัสของสัญญาณเสียง หรือวีดีโอที่ถูกบีบอัดข้อมูลมาด้วยมาตรฐาน MPEG ซึ่งแต่ก่อนเราต้องใช้ฮาร์ดแวร์พิเศษในการถอดรหัส แต่ปัจจุบันใช้เพียงซอฟท์แวร์ก็สามารถทำได้แล้ว โดยอาศัย CPU ที่มีความเร็วสูงขึ้น

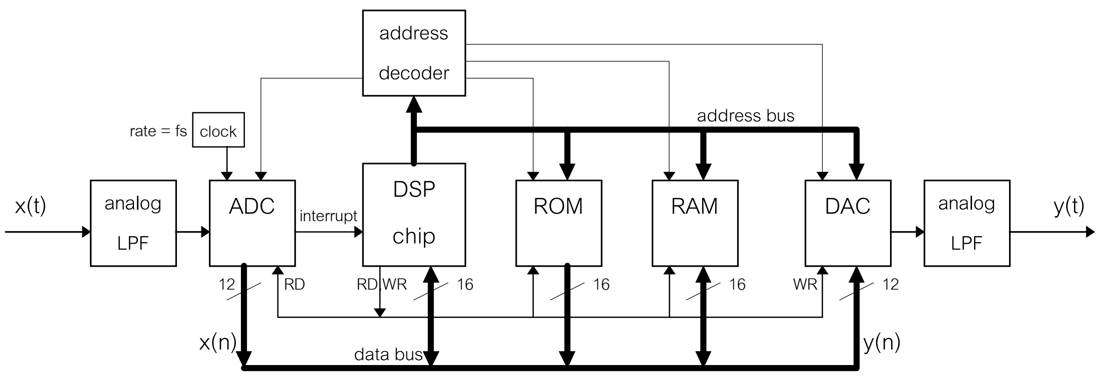
2) การใช้ซอฟท์แวร์ร่วมกับชิพ DSP ชิพ DSP เป็นชื่อเล่นของชิพประมวลผลสัญญาณ (Digital Signal Processor) ซึ่งคือ ไมโครโปรเซสเซอร์ที่ถูกออกแบบมาสำหรับงานประมวลผล

สัญญาณแบบเวลาจริงโดยเฉพาะ โดยไมโครโปรเซสเซอร์ประเภทนี้จะมีสถาปัตยกรรมที่เอื้ออำนวยต่อการคำนวณ และการโอนถ่ายข้อมูลที่มีประสิทธิภาพ และความเร็วสูง เช่น การมีคำสั่งพิเศษในการคูณ, การบวกสะสม, หรือการอ้างข้อมูลแบบ circular buffer เป็นต้น บางชนิดยังสามารถทำการประมวลผลหลาย ๆ ส่วนได้พร้อมกันในตัวเดียว (multi-processing) อีกด้วย บริษัทที่เป็นผู้นำด้านการผลิตชิพ DSP ได้แก่ Texas Instruments, Motorola, Analog Devices, และ AT&T เป็นต้น ซึ่งชิพ DSP นี้มีทั้งประเภทที่เป็นการประมวลผลข้อมูลแบบจำนวนเต็ม (fixed-point) และประเภทที่ประมวลผลข้อมูลแบบเลขอิงดรรชนี (floating-point)

การใช้งานชิพ DSP นั้น ทำได้โดยเขียนเป็นโปรแกรมภาษาแอสเซมบลี หรือภาษาซีแล้วใช้คอมไพล์เลอร์แปลเป็นแอสเซมบลี ข้อดีของการเขียนเป็นภาษาแอสเซมบลีโดยตรง คือ สามารถควบคุมการทำงานของชิพได้เต็มที่ ทำให้สามารถออกแบบโปรแกรมให้ทำงานได้เร็วกว่า และมีขนาดโปรแกรมเล็กกว่าการใช้ภาษาซี แต่ข้อเสียก็คือ ภาษาแอสเซมบลีเขียนยากกว่า และไม่สามารถโอนย้ายโปรแกรมไปทำงานได้ในชิพต่างตระกูลกัน หรือต่างผู้ผลิตกันได้

การต่อวงจรเพื่อใช้งานชิพ DSP ก็ทำเช่นเดียวกับการต่อวงจรไมโครโปรเซสเซอร์ทั่ว ๆ ไป เพียงแต่มีตัวแปลงสัญญาณแอนะลอกเป็นดิจิตอล (ADC) และดิจิตอลเป็นแอนะลอก (DAC) เพิ่มขึ้นมาเท่านั้น ในรูปที่ 1.5 เป็นแผนภาพทั่วไปของวงจร ซึ่งใช้ชิพ DSP แบบ fixed-point 16 บิต เช่น TMS320C50 ของ Texas Instruments โดยมันจะมีบัสข้อมูลขนาด 16 บิต และมีตัวคูณ และประมวลผลอื่น ๆ ขนาด 16 บิตอยู่ภายใน ในรูปนี้เราใช้ DAC และ ADC ขนาด 12 บิต ซึ่งมีขาข้อมูล 12 ขา เพื่อส่งข้อมูลแบบขนาน และต่อเข้ากับ 12 บิตล่างของบัสข้อมูล สังเกตว่ามีสัญญาณนาฬิกาซึ่งมีความถี่ f_s ป้อนให้กับ ADC เพื่อเป็นตัวกำหนดอัตราการสุ่มสัญญาณแอนะลอก ซึ่งก็คือ อัตราของข้อมูลที่จะต้องถูกอ่านเข้าชิพ DSP ไปประมวลผล

ชิพ DSP หลายยี่ห้อมีหน่วยความจำ ROM และ RAM บางส่วนอยู่ภายในตัวเอง ทำให้เพิ่มความเร็วในการทำงาน และสะดวกในการใช้งานมาก โดยงานที่ไม่ต้องการใช้ปริมาณ ROM และ RAM มากนัก ก็อาจไม่จำเป็นต้องต่อหน่วยความจำภายนอกเลย



***รูปที่ 1.5** แผนภาพแสดงตัวอย่างของวงจรที่ใช้งานชิพ DSP*

3) **การใช้ฮาร์ดแวร์ หรือ ไอซีที่ออกแบบเฉพาะงาน** ฮาร์ดแวร์ในที่นี้ก็หมายถึง วงจรดิจิตอล ซึ่งสามารถออกแบบให้ทำการประมวลผลข้อมูลได้เช่นเดียวกัน อัลกอริธึมที่เป็นที่นิยม เช่น FFT (Fast Fourier Transform) หรือ ตัวกรองดิจิตอลนั้น เราอาจสามารถหาซื้อได้ทั่วไปเป็นไอซีสำเร็จรูปที่ทำเฉพาะฟังก์ชั่นนั้น ๆ แต่ถ้าต้องการอัลกอริธึมที่เฉพาะมากขึ้น ก็อาจต้องทำการออกแบบเป็นไอซีเฉพาะงานเอง (Application Specific Integrated Circuits หรือ ASIC) ซึ่งแน่นอนว่าต้นทุนในการออกแบบสำหรับทางเลือกนี้ค่อนข้างสูง ทางเลือกอีกทางหนึ่ง คือ การใช้ไอซีดิจิตอลประเภทโปรแกรมได้ หรือ FPGA (Field Programmable Gate Array) ซึ่งปัจจุบันมีขนาดใหญ่มากพอที่จะนำมาใช้ทำการประมวลผลสัญญาณได้ การใช้ FPGA จะมีต้นทุนในการออกแบบที่ถูกกว่า ASIC

การเลือกใช้ตัวประมวลผลแต่ละแบบก็ขึ้นอยู่กับลักษณะของงาน ความเร็วที่ต้องการ และต้นทุน ถ้าต้องการทำอุปกรณ์ที่มีการประมวลผลแบบเวลาจริง โดยทั่วไปการใช้ชิพ DSP จะดีที่สุด (ซึ่งชิพ DSP ก็มีหลากหลายขนาด และความเร็วให้เลือกใช้อีก) แต่ถ้าหากการประมวลผลไม่ซับซ้อน หรืออัตราข้อมูลไม่สูงมากจนสามารถใช้ไมโครโปรเซสเซอร์ธรรมดาได้ การใช้ไมโปรโปรเซสเซอร์ก็จะทำให้ต้นทุนต่ำลงได้ ในกรณีที่ต้องการอัตราการประมวลผลสูงมาก ๆ เราก็อาจต้องใช้ฮาร์ดแวร์ในการประมวลผล ซึ่งโดยทั่วไปก็จะมีต้นทุนที่สูงขึ้น

งานที่มีการประยุกต์ใช้การประมวลผลสัญญาณดิจิตอล

ปัจจุบันมีงานหลายอย่างที่ได้นำเอาการประมวลผลสัญญาณดิจิตอลไปใช้งาน คงจะสามารถยกตัวอย่างได้เพียงแค่ส่วนหนึ่งของมันเท่านั้น ซึ่งได้แก่

1. การประมวลผลเสียง เช่น การบีบอัดเสียง หรือเข้ารหัสเสียง (speech coding), การรู้จำเสียง (speech recognition), การเติมเอฟเฟคเสียง (sound effect), การผสมเสียง, การกรองเสียงรบกวน, การสังเคราะห์เสียงดนตรี (music synthesizer) เป็นต้น
2. ในระบบสื่อสาร ได้แก่ modulation/demodulation, การชดเชยผลของช่องสัญญาณ (channel equalizer) ในอุปกรณ์โมเดม และโทรศัพท์มือถือ, การกรองเสียงสะท้อนในระบบโทรศัพท์ทางไกล และระบบการประชุมทางไกล (video conferencing), สายอากาศแบบปรับรูปแบบการรับได้เอง, ระบบเรดาร์ และโซน่าร์, ระบบนำทาง (navigation system), GPS เป็นต้น
3. ในระบบควบคุมโดยดิจิตอล (digital control system) ต่าง ๆ
4. ในวงการแพทย์ ได้แก่ การวิเคราะห์สัญญาณคลื่นสมอง (EEG) และสัญญาณคลื่นหัวใจ (ECG), เครื่องช่วยได้ยิน (hearing aid) เป็นต้น
5. ในระบบการจ่ายกำลังไฟฟ้า ซึ่งใช้ในการลดปริมาณของฮาร์มอนิกส์ที่เกิดขึ้น
6. การประมวลผลสัญญาณแบบหลายมิติ ได้แก่ การประมวลผลภาพนิ่ง (2 มิติ), วีดีโอ (3 มิติ), holography (ภาพ 3 มิติ) ตัวอย่างของการประยุกต์ใช้งาน ได้แก่ การบีบอัดสัญญาณวีดีโอ, การ

ทำภาพให้ชัดขึ้น เช่นใช้กับภาพถ่ายดาวเทียม, ภาพทางโบราณคดี, และภาพที่ถ่ายแล้วไม่ชัด, ระบบรู้จำภาพ, การมองเห็นของหุ่นยนต์, และการเคลื่อนไหวของภาพสามมิติ เป็นต้น

7. ในอุปกรณ์ และเครื่องมือทางไฟฟ้า เช่น เครื่องวิเคราะห์ความถี่ (spectrum analyzer), เครื่องสร้างสัญญาณ (function generator), และเครื่องตรวจตัวสัญญาณ (pattern matching) เป็นต้น

8. ในการวิเคราะห์ทางสถิติ และการเงิน

กล่าวได้ว่า การประมวลผลสัญญาณดิจิตอล ได้ปฏิวัติเทคโนโลยีต่าง ๆ ให้ก้าวหน้า และมีประสิทธิภาพขึ้นอย่างมากระยะเวลาที่ผ่านมา ในบทที่ 12 จะได้ทำการยกตัวอย่างการประยุกต์ใช้งานบางอย่าง รวมถึงอธิบายหลักการของการประยุกต์ใช้งานนั้น ๆ

ข้อดีของการใช้การประมวลผลสัญญาณดิจิตอล

ข้อดีของการใช้การประมวลผลสัญญาณดิจิตอล ที่เหนือกว่าการใช้วงจรในระบบแอนะลอก มีดังนี้

1. ความสามารถในการโปรแกรมได้ ทำให้ง่ายต่อการออกแบบ, เปลี่ยนแปลงแก้ไข, และทดสอบ สำหรับวงจรแอนะลอก ถ้าต้องการเปลี่ยนคุณสมบัติอะไรบางอย่าง อาจหมายถึงการต้องออกแบบวงจรใหม่เลย

2. ความถูกต้องแม่นยำที่ดีกว่า ความถูกต้องของการประมวลผลสัญญาณดิจิตอล ขึ้นอยู่กับจำนวนบิตที่ใช้แทนสัญญาณ และพารามิเตอร์ต่าง ๆ ซึ่งมีความยืดหยุ่น และควบคุมได้ง่าย คือ ในงานที่เราต้องการความแม่นยำสูง เราก็จะใช้จำนวนบิตที่มากขึ้น อีกทั้งในช่วงของการออกแบบ การจำลองระบบที่ออกแบบในคอมพิวเตอร์ จะให้ผลที่ตรงกับความเป็นจริงเมื่อนำไปสร้างเป็นวงจรจริง

3. สามารถทำฟังก์ชั่นที่พิสดารที่ไม่สามารถทำได้ด้วยวงจรแอนะลอก หรือทำได้ยากมาก เช่น ตัวกรองแบบปรับตัวได้ (adaptive filter) ตามสภาวะของสัญญาณรบกวน, สายอากาศที่ปรับทิศทางการรับเองได้, การเติมเอฟเฟคเสียงให้เป็นเสียง 3 มิติ เป็นต้น

4. มีเสถียรภาพที่ไม่ขึ้นกับเวลา และอุณหภูมิ

5. DSP เกี่ยวข้องโดยตรงกับเทคโนโลยีคอมพิวเตอร์ และ VLSI (ชิพ DSP ก็จัดเป็นชิพประเภท VLSI) ซึ่งเทคโนโลยีเหล่านี้กำลังเจริญก้าวหน้าอย่างรวดเร็ว ทั้งในด้านความเร็วที่สูงขึ้น ความจุของชิพที่มากขึ้น การกินกำลังไฟที่ต่ำลง และราคาก็ถูกลง ข้อดีนี้ถึงแม้เป็นผลพลอยได้แต่ก็มีความสำคัญมาก เพราะมันหมายถึงว่าต้นทุนของการใช้การประมวลผลสัญญาณดิจิตอลจะต่ำลง ๆ ตามความก้าวหน้าของเทคโนโลยี เราจะเห็นได้ว่า งานบางอย่างที่ในอดีตการใช้วงจรแอนะลอกให้ต้นทุนที่ต่ำกว่า แต่ในปัจจุบันกลับใช้แบบดิจิตอลแล้วคุ้มค่ากว่า หรือ อัลกอริธึมบางอย่างที่มีผู้คิดค้นได้ในอดีต เช่น Kalman Filter แต่ไม่สามารถนำมาใช้ได้ในเวลานั้น เนื่องจากมีความซับซ้อนของการ

คำนวณมากทำให้ไม่คุ้มค่ากับการนำมาใช้ ก็ปรากฏว่า อัลกอริธึมเหล่านั้นกลับนำมาใช้งานได้จริงในปัจจุบัน ทั้งนี้เป็นผลโดยตรงจาก ความก้าวหน้าของเทคโนโลยีคอมพิวเตอร์ และ VLSI

ขีดจำกัดของการประมวลผลสัญญาณดิจิตอล

ทุกอย่างที่มีข้อดี ก็มักจะมีขีดจำกัดด้วยเสมอ เทคโนโลยีของดิจิตอลที่มีข้อดีต่าง ๆ มากมายตามที่กล่าวมาก็เช่นเดียวกัน ขีดจำกัดของการใช้การประมวลผลสัญญาณดิจิตอลพอจะแจกแจงได้ ดังนี้

1. สัญญาณแอนะลอกที่มีแถบความถี่ (bandwidth) สูงมาก ๆ ไม่สามารถใช้กับการประมวลผลสัญญาณดิจิตอลได้ เนื่องจากสัญญาณพวกนี้ต้องการอัตราการสุ่มที่สูงมากเพื่อแปลงเป็นดิจิตอล ทำให้ต้องการตัวประมวลผลที่เร็วมากจนไม่คุ้มค่าต่อการใช้งาน นอกจากนี้ ขีดจำกัดที่อาจสำคัญมากกว่าตัวประมวลผล ก็คือ การที่ต้องมีตัวแปลงสัญญาณระหว่างแอนะลอกกับดิจิตอลที่มีความเร็วสูงมาก ซึ่งปัจจุบัน เทคโนโลยีของการแปลงสัญญาณแอนะลอกเป็นดิจิตอลสามารถทำได้ที่อัตราสุ่มสูงสุดประมาณ ... MHz ซึ่งหมายความว่า เราไม่สามารถประมวลผลสัญญาณมีแถบความถี่สูงกว่า ... MHz ได้ (ครึ่งหนึ่งของอัตราการสุ่ม)

2. งานที่ต้องการการกินกำลังไฟที่ต่ำมาก ๆ อาจจะต้องทำด้วยวงจรแอนะลอกอยู่ ในปัจจุบันถึงแม้ชิพ VLSI จะกินกำลังไฟต่ำลงมากเมื่อเทียบกับอดีต ประกอบกับเทคโนโลยีของแบตเตอรี่ที่ก้าวหน้าไปมาก ทำให้อุปกรณ์พกพาหลาย ๆ อย่างที่มี DSP เป็นส่วนประกอบ เช่น โทรศัพท์มือถือมีขนาดเล็กกระทัดลัดลงมาก และแถมยังใช้งานได้นานขึ้นอีก แต่อย่างไรก็ตาม การประมวลผลก็ยังจัดเป็นกระบวนการที่กินกำลังไฟพอสมควร อุปกรณ์ที่ต้องการให้มีขนาดเล็กมาก ๆ ที่ไม่ต้องการใส่แบตเตอรี่ขนาดใหญ่ลงไป เช่น อุปกรณ์ช่วยได้ยิน ก็จะมีข้อได้เปรียบของการออกแบบเป็นชิพแอนะลอก ในด้านที่จะสามารถกินกำลังไฟได้ต่ำกว่า

3. อุปกรณ์บางอย่าง ถึงแม้ทำได้ดีกว่าด้วยเทคโนโลยีดิจิตอล แต่ก็ด้วยต้นทุนที่สูงกว่า จึงมีตลาดที่จำกัดอยู่เฉพาะผู้ใช้ที่มีกำลังซื้อ อุปกรณ์เหล่านี้จึงยังคงมีใช้อยู่ทั้งแบบดิจิตอล และแอนะลอก เช่น โทรทัศน์ดิจิตอล กับโทรทัศน์แอนะลอก, เครื่องเล่น DVD กับเครื่องเล่นวีดีโอเทป, และออสซิลโลสโคปดิจิตอล กับออสซิลโลสโคปแอนะลอก, ฯลฯ เป็นต้น

4. ขีดจำกัดในข้อที่ 1 ถึง 3 จะค่อยน้อยลง ๆ ตามความเจริญของเทคโนโลยีคอมพิวเตอร์ และ VLSI ดังที่ได้กล่าวมาแล้ว แต่อย่างไรก็ตาม มีวงจรบางประเภทที่ต้องสร้างด้วยเทคโนโลยีแอนะลอกเสมอ (ถึงแม้ในอนาคตก็ตาม) และจริง ๆ แล้วระบบประมวลผลสัญญาณดิจิตอลก็ต้องพึ่งพาวงจรเหล่านี้ด้วย นั่นคือ วงจรขยายสัญญาณต่าง ๆ, ตัวแปลงสัญญาณแอนะลอกเป็นดิจิตอล, ตัวแปลงสัญญาณดิจิตอลเป็นแอนะลอก, ตัวกรองแอนะลอกในส่วน front-end (ก่อนตัวแปลงแอนะลอกเป็นดิจิตอล) และ ตัวกรองแอนะลอกในส่วน back-end (หลังตัวแปลงดิจิตอลเป็นแอนะลอก) เป็นต้น

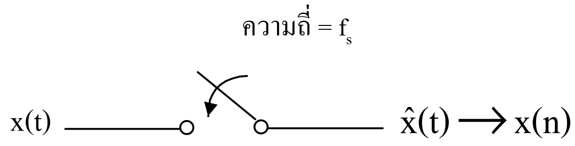
บทที่ 2

การสุ่มสัญญาณ และการสร้างสัญญาณคืน

ในบทนี้เราจะศึกษาในแง่ทฤษฎีเกี่ยวกับการสุ่มสัญญาณซึ่งคือกระบวนการแปลงสัญญาณแอนะลอกเป็นดิจิตอล และการสร้างสัญญาณคืนซึ่งคือกระบวนการแปลงสัญญาณดิจิตอลกลับเป็นแอนะลอก เพื่อให้เข้าใจถึงลักษณะของสัญญาณก่อนการสุ่ม และหลังการสุ่มทั้งในภาคเวลา และความถี่ เราจะศึกษาถึงขีดจำกัดของการสุ่มสัญญาณ และสร้างสัญญาณคืน พร้อมกับศึกษาวิธีหลีกเลี่ยงผลกระทบที่ไม่พึงปรารถนาที่จะเกิดขึ้นจากกระบวนการทั้งสองด้วย

การสุ่มสัญญาณ (Sampling)

แนวคิดอย่างง่าย ๆ ของการสุ่มก็คือ การนำเอาสัญญาณขาเข้าแบบต่อเนื่องมาผ่านสวิทซ์อุดมคติที่ต่อวงจรที่ตำแหน่งเวลาเท่ากับ ..., -2T, -T, 0, T, 2T, 3T, ... และเปิดวงจรที่เวลาอื่น ๆ นั่นคือ มีความถี่ของการตัดต่อวงจรเท่ากับ f_s ครั้งต่อวินาที หรือมีคาบของการสุ่มเท่ากับ $T=1/f_s$ ดังในรูปที่ 2.1 เรียกสัญญาณขาออกว่า $\hat{x}(t)$



รูปที่ 2.1 *การสุ่มด้วยสวิทซ์อุดมคติ*

แน่นอนว่า ถ้ามีสวิทซ์อย่างนี้จริง เราจะได้สัญญาณขาออก $\hat{x}(t)$ ซึ่งเป็นสัญญาณแอนะลอกที่มีลักษณะเป็นอุดมคติ คือ มีค่าที่เฉพาะเวลา ..., -2T, -T, 0, T, 2T, 3T, ... เท่านั้น ส่วนที่เวลาอื่น ๆ มีค่าเป็นศูนย์ สัญญาณนี้หน้าตาเหมือนสัญญาณไม่ต่อเนื่องในบทที่ 1 ทุกประการเพียงแต่มองมันเป็นสัญญาณแอนะลอกในแกนเวลา t ซึ่งถึงแม้เราจะไม่ใช้สัญญาณนี้ในลักษณะเป็นสัญญาณแอนะลอก แต่การมองนี้ก็เพื่อการวิเคราะห์ในเชิงความถี่ของสัญญาณ ซึ่งจะมีประโยชน์ในการบอกถึงขีดจำกัดของกระบวนการสุ่มได้

เราจะลองมาวิเคราะห์หาสเปกตรัมของสัญญาณ $\hat{x}(t)$ ดูว่าสอดคล้องกับสเปกตรัมของ $x(t)$ อย่างไร โดยจะพิสูจน์ในแง่คณิตศาสตร์ ถ้านิยามว่ามี สัญญาณอิมพัลส์ (impulse signal) หรือเขียนแทนด้วยสัญลักษณ์ว่า $\delta(t)$ เป็นสัญญาณที่มีค่าเท่ากับ 1 ที่เวลา $t=0$ และเป็น 0 ที่เวลาอื่น ๆ ดังนี้

$$\delta(t) = \begin{cases} 1, t = 0 \\ 0, t = \text{ค่าอื่น ๆ} \end{cases} \tag{2.1}$$

เราอาจมองว่าการสุ่ม คือ การนำเอาสัญญาณขาเข้ามาคูณเข้ากับสัญญาณอิมพัลส์หลาย ๆ ลูกที่มีคาบเท่ากับ T (ระยะห่างระหว่างอิมพัลส์แต่ละลูก) ขอนิยามสัญญาณอิมพัลส์หลาย ๆ ลูกนี้เป็นสัญญาณ $s(t)$ ซึ่งรูปร่างของมันแสดงในรูปที่ 2.2 ในทางคณิตศาสตร์ สามารถเขียน $s(t)$ ได้ว่าเป็นผลรวมของสัญญาณอิมพัลส์ที่ตำแหน่งเวลาต่าง ๆ ..., -2T, -T, 0, T, 2T, ... ดังนี้

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \tag{2.2}$$

เมื่อนำสัญญาณ $s(t)$ นี้คูณเข้ากับสัญญาณขาเข้า $x(t)$ จะได้สัญญาณที่เป็นขาออกของตัวสุ่มสัญญาณ คือ

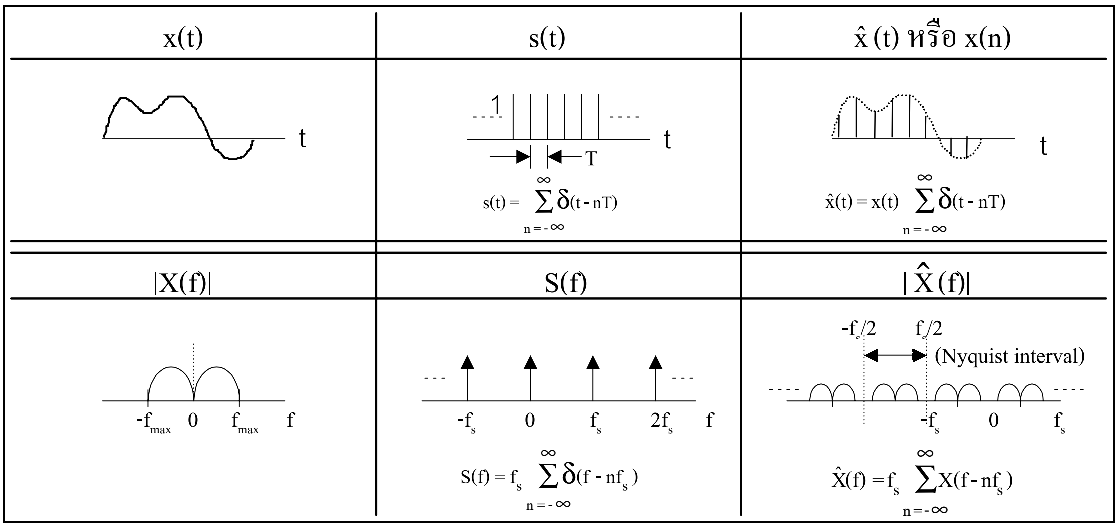
$$\hat{x}(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) \tag{2.3}$$

ตัวอย่างของสัญญาณ $\hat{x}(t)$ มีลักษณะดังแสดงในรูปที่ 2.2 สัญญาณนี้เราสามารถนิยามให้มันเป็นสัญญาณไม่ต่อเนื่อง หรือเป็น “ลำดับของค่า” ดังที่ได้กล่าวมาในบทที่ 1 ซึ่งก็จะใช้สัญญลักษณ์เป็น $x(n)$ โดยใช้ค่า n เป็นตัวชี้แทน t จะได้ว่า $x(n)$ มีความสัมพันธ์กับ $x(t)$ ดังนี้

$$x(n) = x(t)\Big|_{t = nT} \tag{2.4}$$

ขอย้ำให้เข้าใจอีกครั้งว่า สัญญาณ $\hat{x}(t)$ กับ $x(n)$ นั้น หมายถึงสัญญาณตัวเดียวกัน เพียงแต่สัญญาณ $\hat{x}(t)$ เป็นการมองสัญญาณนี้ในลักษณะเป็นสัญญาณแอนะลอกในแกนของเวลา t ซึ่งเราจะเห็นว่าสัญญาณมีค่าที่เวลา t = … -2T, -T, 0, T, 2T, … แต่สัญญาณ $x(n)$ เป็นการมองสัญญาณเป็นสัญญาณไม่ต่อเนื่อง หรือเป็นลำดับ คือ มีค่าที่ n = …, -2, -1, 0, 1, 2, … สัญญาณ $x(n)$ นี้ทิ้งความหมายของเวลาแบบแอนะลอกไปเรียบร้อยแล้ว กลายเป็นเหมือนลำดับของข้อมูลเฉย ๆ ในบทต่อ ๆ ไปเราจะทำงานกับสัญญาณประเภทนี้ และจะเห็นต่อไปว่า ตัวประมวลผลสัญญาณดิจิตอลก็ไม่จำเป็นต้องรู้

ค่าเวลาที่แท้จริงของสัญญาณเลย หรือไม่ต้องใช้ค่า f_s เลยในการประมวลผล แต่อย่างไรก็ตามเราต้องจดบันทึกไว้ว่า x(n) ที่เป็นสัญญาณขาเข้าของระบบนี้ เกิดจากการสุ่มสัญญาณแอนะลอกมาด้วยอัตรา f_s เท่าไร เพื่อที่จะใช้วิเคราะห์ในเรื่องผลตอบสนองเชิงความถี่ของระบบ และใช้เป็นพารามิเตอร์ในการออกแบบระบบดังจะได้กล่าวในบทต่อ ๆ ไป



***รูปที่ 2.2** สรุปสัญญาณต่าง ๆ ในการสุ่มทั้งภาคเวลาและความถี่*

ลองมาดูในภาคความถี่บ้างว่าเกิดอะไรขึ้น สมมติว่าสัญญาณ x(t) เมื่อใช้การแปลงฟูริเยร์ได้สเปกตรัมเป็น X(f) ซึ่งสมมติว่ามีแถบความถี่ (bandwidth) จำกัด และมีองค์ประกอบความถี่สูงสุดอยู่ที่ f_{max} ดังในรูป 2.2 ส่วนสัญญาณ s(t) จะสามารถพิสูจน์โดยใช้การแปลงฟูริเยร์ (ขอละไม่พิสูจน์ให้ดู) ได้ว่ามีสเปกตรัมเป็นสัญญาณอิมพัลส์หลายลูกที่มีคาบคงที่เช่นกัน โดยคาบในที่นี้เท่ากับ f_s และมีขนาดคงที่เท่ากับ f_s เขียนเป็นสมการได้ดังสมการที่ 2.5

$$S(f) = f_s \sum_{n=-\infty}^{\infty} \delta(f - nf_s) \tag{2.5}$$

สำหรับสเปกตรัมของสัญญาณขาออก คือ $\hat{X}(f)$ อาจหาได้โดยกฎที่ว่า การคูณในเชิงเวลาเท่ากับการคอนโวลูชัน (convolution) ในเชิงความถี่ นั่นคือ เมื่อ $\hat{x}(t)$ เป็นผลคูณระหว่างสัญญาณ s(t) กับ x(t) จะได้ว่า สเปกตรัมของ $\hat{x}(t)$ คือ $\hat{X}(f)$ จะเท่ากับผลคูณคอนโวลูชันระหว่าง X(f) กับ S(f) ดังนี้ (ขอนิยามสัญลักษณ์ * แทนการคูณคอนโวลูชัน)

$$\hat{X}(f) = X(f) * S(f)$$

เมื่อแทนค่า S(f) ลงไป แล้วจัดให้เป็นรูปอย่างง่ายจะได้

$$\hat{X}(f) = X(f) * f_s \sum_{n=-\infty}^{\infty} \delta(f - nf_s)$$

$$= f_s \sum_{n=-\infty}^{\infty} X(f) * \delta(f - nf_s)$$

$$\hat{X}(f) = f_s \sum_{n=-\infty}^{\infty} X(f - nf_s) \qquad (2.6)$$

สมการนี้ประกอบด้วยผลบวกของเทอม $X(f\text{-}nf_s)$ ซึ่งเทอมนี้ คือ สเปกตรัมของสัญญาณขาเข้าที่เลื่อนจุดศูนย์กลางไปอยู่ที่ตำแหน่งความถี่ nf_s โดย n เป็นจำนวนเต็มตั้งแต่ $-\infty$ จนถึง $+\infty$ นั่นหมายถึงว่า สเปกตรัมของสัญญาณหลังการสุ่ม คือ $\hat{X}(f)$ ประกอบด้วยสำเนาของสเปกตรัมของสัญญาณก่อนการสุ่ม คือ X(f) อยู่มากมาย โดยสำเนาเหล่านี้ จะเกิดขึ้นที่ความถี่ ..., $-2f_s$, $-f_s$, 0, f_s, $2f_s$, ... เป็นศูนย์กลาง สำเนาแต่ละตัวนี้ มีชื่อเรียกอีกอย่างหนึ่งว่า เป็นภาพฉาย (image) ของสัญญาณ รูปร่างของ $\hat{X}(f)$ ที่ได้ แสดงตัวอย่างดังในรูปที่ 2.2

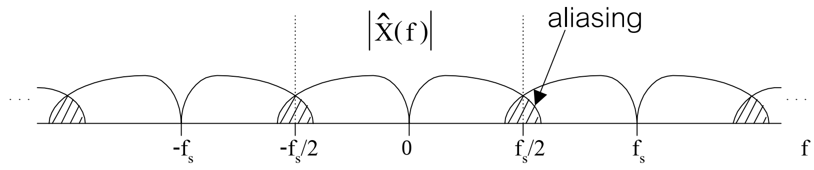
ขอให้สังเกตว่า สเปกตรัม $\hat{X}(f)$ ที่หามาได้นี้ก็มีลักษณะที่เป็นอุดมคติในเชิงสัญญาณแอนะลอก คือ มันมีองค์ประกอบของสัญญาณเรื่อย ๆ ไปจนถึงความถี่อนันต์ แสดงว่ามันมีพลังงานไม่จำกัด ซึ่งเป็นไปไม่ได้ ดังนั้น ทั้งสัญญาณ x(t) และ $\hat{X}(f)$ (ถ้ามองในเชิงแอนะลอก) เป็นสัญญาณที่เราสร้างขึ้นในทางคณิตศาสตร์ เพื่อประโยชน์ในการวิเคราะห์ขีดจำกัดของกระบวนการสุ่มต่อไป

ทฤษฎีการสุ่มสัญญาณ (Sampling Theorem)

ทฤษฎีการสุ่มสัญญาณ ระบุไว้ว่า ถ้าสัญญาณที่ต้องการสุ่มมีความถี่สูงสุดที่ f_{max} เพื่อให้ได้สัญญาณที่สุ่มแล้วเป็นตัวแทนที่ถูกต้องของสัญญาณนี้ ความถี่ในการสุ่มจะต้องมีค่ามากกว่าสองเท่าของความถี่สูงสุดในสัญญาณ หรือ

$$f_s > 2f_{max} \qquad (2.7)$$

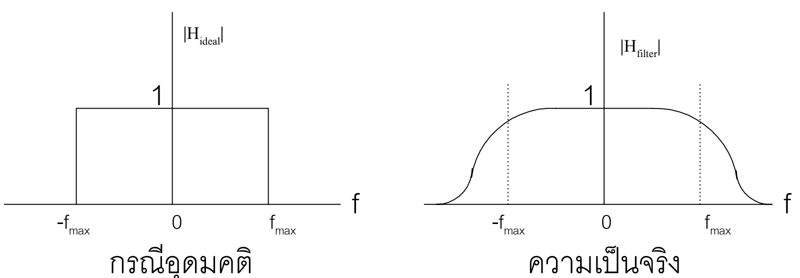
ซึ่งความถี่ที่ $2f_{max}$ นี้ มีชื่อเรียกพิเศษว่า “ความถี่ไนควิสท์” (Nyquist frequency) หรืออัตราไนควิสท์ ลองพิจารณาว่าถ้าเราใช้ f_s ต่ำกว่าค่าความถี่ไนควิสท์จะเกิดอะไรขึ้น ซึ่งผลที่เกิดขึ้นนี้จะดูได้ชัดเจนในภาคความถี่ ถ้าเรายึดหลักว่าที่ได้พิสูจน์มาในหัวข้อที่แล้วว่า หลังการสุ่มจะเกิดสเปกตรัมของสัญญาณก่อนการสุ่มอยู่ที่ความถี่ ..., $-2f_s$, $-f_s$, 0, f_s, $2f_s$, ... เป็นศูนย์กลาง แล้วลองวาดรูปคร่าว ๆ ออกมา จะได้ดังแสดงในรูป 2.3



***รูปที่ 2.3** แสดงสัญญาณในเชิงความถี่ที่เกิดขึ้น เมื่อใช้ f_s ต่ำกว่า $2f_{max}$*

เมื่อ f_s ต่ำกว่า $2f_{max}$ จะเห็นได้ว่า สำเนาของ X(f) ที่เกิดขึ้นจะมีช่วงของความถี่ส่วนปลายที่ซ้อนทับกัน เรียกองค์ประกอบความถี่ส่วนที่ซ้อนทับกันนี้ว่า aliasing (อ่านว่า เอเลียดซิ่ง) ซึ่ง aliasing นี้เป็นผลร้ายอย่างยิ่งกับระบบประมวลผลสัญญาณ เนื่องจาก ในระบบแบบไม่ต่อเนื่องเราจะสนใจความถี่ในช่วง $-f_s/2$ จนถึง $f_s/2$ ซึ่งเรียกว่า <u>ช่วงไนควิสท์</u> หรือ ย่านไนควิทซ์ (Nyquist interval) ซึ่งถ้าเกิด aliasing ซ้อนทับในช่วงไนควิสท์นี้ ก็จะถือว่า สัญญาณขาเข้าที่สุ่มมาได้มีความผิดเพี้ยนไปก่อนที่จะเข้าไปในส่วนของการประมวลผลเสียอีก หรือพูดอีกอย่างหนึ่งก็คือ สัญญาณที่สุ่มมาได้ ไม่เป็นตัวแทนที่สมบูรณ์ของสัญญาณแอนะลอกขาเข้า จึงจำเป็นอย่างยิ่งที่เราต้องพยายามไม่ให้ aliasing เกิดขึ้น หรือให้เกิดน้อยที่สุดเท่าที่จะทำได้ ซึ่งกระทำได้สองวิธี คือ

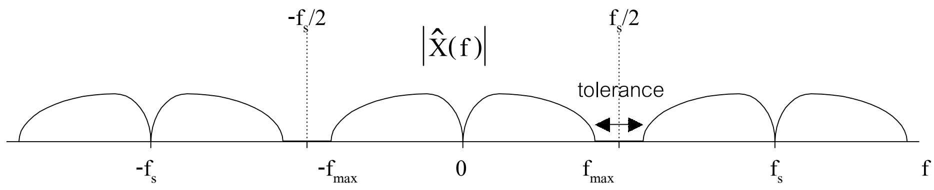
1. <u>การใช้ตัวกรองเพื่อป้องกัน aliasing</u> (anti-aliasing filter) ในกรณีที่เราไม่มั่นใจว่าสัญญาณที่จะทำการสุ่มไม่มีความถี่จำกัดอยู่ที่ f_{max} เช่น อาจมีสัญญาณรบกวนความถี่สูง หรือสัญญาณอื่น ๆ ปนอยู่ด้วย วิธีแก้ทำได้โดยการใช้ตัวกรองแอนะลอกแบบผ่านความถี่ต่ำ เพื่อจำกัดความถี่ของสัญญาณให้อยู่ในช่วงที่สนใจเท่านั้น (ต่ำกว่า f_{max}) นั่นคือ เราต้องการตัวกรองผ่านความถี่ต่ำที่มีความถี่ตัดที่ f_{max} ดังแสดงผลตอบสนองเชิงความถี่แบบอุดมคติของตัวกรองนี้ ในรูปซ้ายมือของรูปที่ 2.4



***รูปที่ 2.4** ผลตอบสนองเชิงความถี่ของตัวกรองป้องกัน aliasing*

อย่างไรก็ตาม ในชีวิตจริงตัวกรองแบบอุดมคติที่ต้องการไม่สามารถทำได้ และถ้าต้องการสร้างตัวกรองที่ใกล้เคียงกับอุดมคติก็จะทำให้ต้นทุนสูง ตัวกรองที่สามารถทำได้ในทางปฏิบัติมีรูปร่างประมาณในรูปขวามือของรูปที่ 2.4 คือ มีความชันของการตัดความถี่ไม่คมเท่ากับตัวกรองอุดมคติ จึงทำให้อาจมี aliasing บางส่วนเกิดขึ้นได้ถ้าเราใช้ความถี่ในการสุ่มพอดีเท่ากับ $2f_{max}$

2. <u>การสุ่มโดยใช้ความถี่เกินกว่า $2f_{max}$ มาก ๆ</u> หรือเรียกว่าการทำ <u>Oversampling</u> การสุ่มด้วยความถี่สูงขึ้นเป็นวิธีที่ใช้ป้องกัน aliasing เสริมจากวิธีแรก เช่นเดียวกัน ถ้าพิจารณาในเชิงความถี่จะเห็นว่า เมื่อ f_s มีค่าสูงขึ้น จะมีช่วงความถี่ที่เผื่อให้เกิด aliasing ได้กว้างขึ้น (ช่วงเผื่อนี้ คือ ย่าน tolerance ที่เขียนอยู่ในรูปที่ 2.5 เป็นย่านที่ไม่มีความถี่ของสัญญาณที่เราสนใจอยู่) ช่วงเผื่อนี้เป็นย่านความถี่ที่ยอมให้มีความถี่ของสัญญาณที่ไม่ต้องการหลุดรอดเข้ามาในระบบได้บ้าง ดังนั้น การสุ่มด้วยความถี่สูงกว่าอัตราไนควิสท์ จึงทำให้เราสามารถใช้ตัวกรองป้องกัน aliasing ที่ไม่ต้องมีคุณสมบัติคมมากนักได้



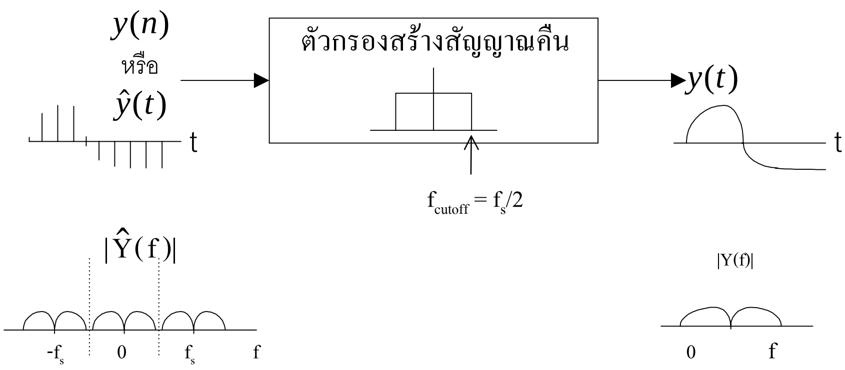
รูปที่ 2.5 *สเปกตรัมของสัญญาณที่เกิดขึ้น เมื่อสุ่มด้วยความถี่สูงกว่า $2f_{max}$ มาก ๆ*

ในทางปฏิบัติจึงมักใช้ $f_s \geq 2.5f_{max}$ เพื่อชดเชยผลของการที่ตัวกรอง anti-aliasing ไม่เป็นอุดมคติ สำหรับ f_s สูงสุดที่เราจะสุ่มได้โดยทั่วไป ก็ขึ้นอยู่กับขีดจำกัดด้านความเร็วของตัวแปลงแอนะลอกเป็นดิจิตอล และตัวประมวลผลที่เลือกใช้ ถ้าความถี่ f_s สูงขึ้นก็หมายถึงว่า ต้องใช้วงจรแปลงสัญญาณที่เร็วขึ้น และใช้ปริมาณการประมวณผลที่มากขึ้น เพราะอัตราข้อมูลสูงขึ้น โดยช่วงเวลาที่ใช้ประมวลผล เพื่อให้ได้ค่าแต่ละค่าของสัญญาณขาออก(T_{proc}) ต้องมีค่าน้อยกว่าคาบของการสุ่ม ดังนี้

$$T_{proc} < T \tag{2.8}$$

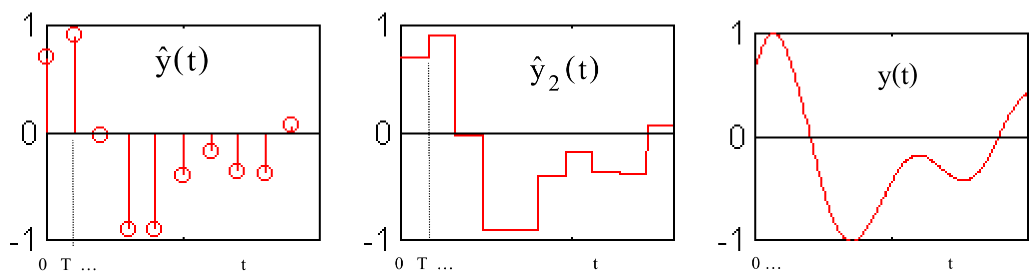
การสร้างสัญญาณคืน (Analog Reconstruction)

การสร้างสัญญาณคืน ในทางทฤษฎีทำได้โดยผ่านสัญญาณแบบไม่ต่อเนื่องเข้าไปยังตัวกรอง (แอนะลอก) แบบผ่านความถี่ต่ำที่มีความถี่ตัดที่ $f_s/2$ ตัวกรองนี้บางครั้งเรียกว่า ตัวกรองสร้างสัญญาณคืน (reconstruction filter) ตัวกรองจะผ่านเฉพาะสัญญาณในช่วงความถี่ระหว่าง $-f_s/2$ ถึง $f_s/2$ หรือช่วงไนควิสท์ออกมา ผลที่ได้ก็คือ เราจะได้สำเนาของสเปกตรัมที่อยู่รอบความถี่ 0 ออกมาเป็นสัญญาณขาออก ซึ่งมันก็ คือ สัญญาณแอนะลอกที่มีรูปร่างเป็นขอบของสัญญาณแบบไม่ต่อเนื่องก่อนสร้างกลับนั่นเอง ดังแสดงในรูปที่ 2.6



***รูปที่ 2.6** สัญญาณขาเข้าและออกของตัวกรองสร้างสัญญาณคืน*

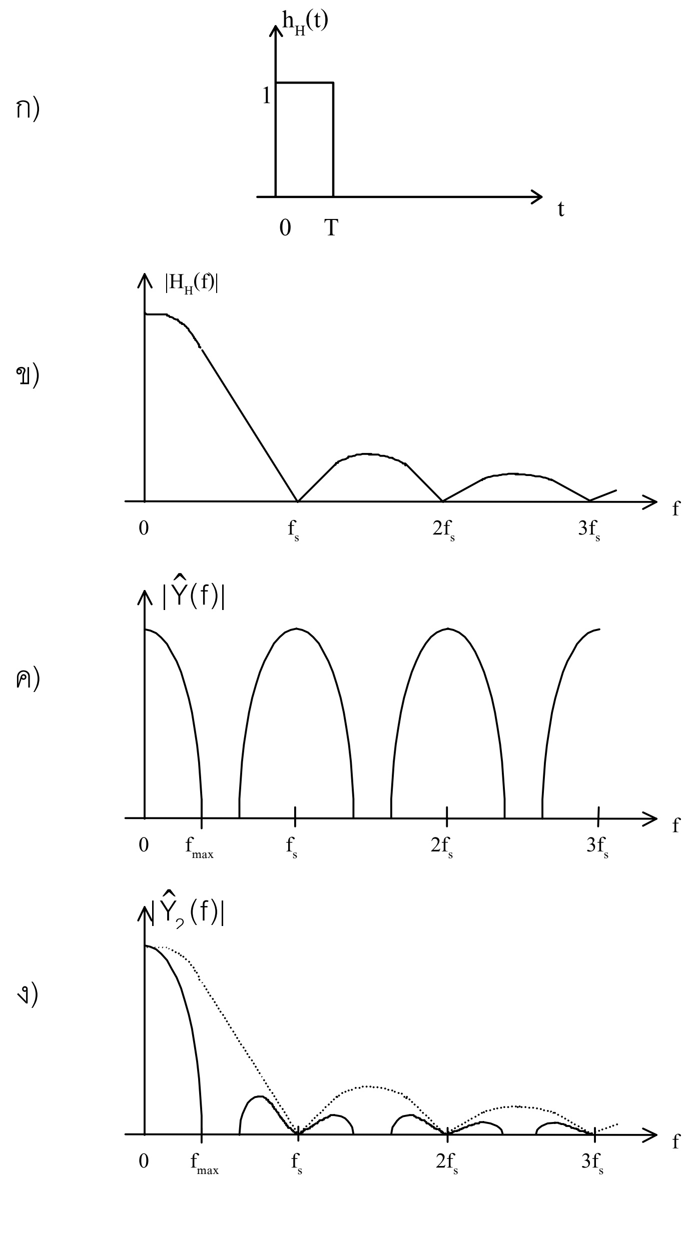
ในทางปฏิบัติ เราไม่ใช้ $\hat{y}(t)$ เป็นสัญญาณขาเข้าของตัวกรองสร้างสัญญาณคืน เนื่องจากอย่างที่กล่าวมาแล้วว่า เรามองเห็นสัญญาณไม่ต่อเนื่องในลักษณะเป็นลำดับของข้อมูล สัญญาณที่มองแบบแอนะลอก คือ $\hat{y}(t)$ เป็นสัญญาณอุดมคติที่ใช้พิสูจน์ในทางคณิตศาสตร์เท่านั้น ดังนั้น การแปลงสัญญาณดิจิตอลเป็นแอนะลอกในชีวิตจริง เราจะนำสัญญาณ $y(n)$ มาผ่านวงจรคงค่า (hold) ที่ทำงานเข้าจังหวะกับอัตราสุ่มของสัญญาณ $y(n)$ เพื่อสร้างเป็นสัญญาณลักษณะขั้นบันไดดังในรูปที่ 2.7 ก่อน จากนั้นจึงค่อยใช้สัญญาณขั้นบันไดนี้สร้างเป็นสัญญาณขาออก โดยส่งมันผ่านตัวกรองผ่านต่ำเพื่อสร้างสัญญาณคืนอีกทีหนึ่ง



***รูปที่ 2.7** สัญญาณขาเข้า และขาออกของวงจรคงค่าสัญญาณ*

บางคนอาจสงสัยว่า แล้วสัญญาณขั้นบันได $\hat{y}_2(t)$ ที่สร้างขึ้นมานี้ มีลักษณะของสเปกตรัมเหมือน หรือแตกต่างจากจากสเปกตรัมของสัญญาณ $\hat{y}(t)$ อย่างไร เราสามารถหาสเปกตรัมของสัญญาณ $\hat{y}_2(t)$ ได้ง่าย ๆ โดยมองวงจรคงค่าสัญญาณเป็นระบบแอนะลอกแบบเชิงเส้นอันหนึ่ง ซึ่งมีผลตอบสนองสัญญาณอิมพัลส์คือ $h_H(t)$ ดังแสดงในรูป 2.8ก นั่นคือ วงจรนี้เปลี่ยนอิมพัลส์ของสัญญาณขาเข้า เป็นพัลส์สี่เหลี่ยมที่มีความกว้างเท่ากับคาบของอัตราการสุ่ม เมื่อทำการแปลงฟูริเยร์ผลตอบสนองอิมพัลส์นี้ จะได้ผลตอบสนองเชิงความถี่ของระบบมีลักษณะเป็นฟังก์ชั่นซิงค์ (sinc) ดังรูป 2.8ข (สัญญาณพัลส์แปลงฟูริเยร์ได้สัญญาณซิงค์ และสัญญาณซิงค์แปลงฟูริเยร์ได้สัญญาณพัลส์)

สเปกตรัมของสัญญาณขาออกของวงจรคงค่านี้ หาได้จากผลคูณของสเปกตรัมของสัญญาณขาเข้า และผลตอบสนองเชิงความถี่ของระบบ ซึ่งได้ผลลัพธ์ดังแสดงในรูปที่ 2.8ง สังเกตได้ว่าสเปกตรัมของสัญญาณขั้นบันไดก็ยังคงมีสำเนาของสัญญาณแอนะลอกที่ความถี่ 0, f_s, $2f_s$, … อยู่ครบถ้วน เพียงแต่สำเนาแต่ละตัวถูกกดขนาดลงไปด้วยการคูณของฟังก์ชั่นซิงค์ ซึ่งสำเนาเหล่านี้จะต้องถูกกำจัดทิ้งโดยตัวกรองสร้างสัญญาณคืนที่มีความถี่ตัดที่ $f_s/2$ เพื่อให้ได้สัญญาณแอนะลอกขาออกมีรูปร่างที่เรียบสมบูรณ์ สรุปว่า ถึงแม้จะใช้วงจรคงค่าซึ่งให้ผลตอบมองดูใกล้เคียงสัญญาณแอนะลอกที่ต้องการแล้ว เราก็ยังคงต้องใช้ตัวกรองสร้างสัญญาณคืนร่วมด้วย ผู้อ่านที่ไม่คุ้นเคยกับเรื่องสัญญาณ และระบบ อาจเข้าใจการพิสูจน์ที่อธิบายมายากสักหน่อย แต่ขอให้เพียงดูในภาพรวม และข้อสรุปที่ได้นี้ก็พอ



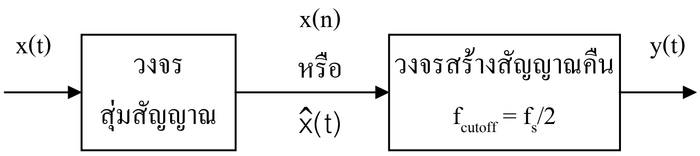
***รูปที่ 2.8** ก) ผลตอบสนองสัญญาณอิมพัลส์ของวงจรคงค่า, ข) ผลตอบสนองเชิงความถี่ของวงจรคงค่า, ค) สเปกตรัมของสัญญาณ $\hat{y}(t)$, ง) สเปกตรัมของสัญญาณ $\hat{y}_2(t)$*

เช่นเดียวกันกับตัวกรองป้องกัน aliasing คือ เราไม่สามารถทำตัวกรองอุดมคติสำหรับสัญญาณคืนได้ ซึ่งผลข้างเคียงก็คือ จะทำให้สัญญาณที่อยู่ในช่วงความถี่สูงกว่า $f_s/2$ หลุดลอดออกมาที่สัญญาณขาออกด้วย กลายเป็นความผิดเพี้ยนของสัญญาณขาออกไป อย่างไรก็ตาม ถ้าหากใช้ความถี่ในการสุ่มสูง ๆ หรือ oversampling เพื่อให้มีระยะห่างระหว่างสำเนาความถี่แต่ละตัวมากขึ้น นอกจากจะช่วยแก้ปัญหาของการใช้ตัวกรองป้องกัน aliasing ที่ไม่เป็นอุดมคติ ดังที่ได้กล่าวในหัวข้อก่อนหน้านี้แล้ว ก็ยังจะช่วยแก้ปัญหาของการใช้ตัวกรองสร้างสัญญาณคืนที่ไม่เป็นอุดมคติได้ในทำนองเดียวกันอีกด้วย

ขอสรุปเรื่องการสุ่มสัญญาณ และการสร้างสัญญาณคืน ด้วยทฤษฎีการสุ่มสัญญาณว่า “ถ้าเราสุ่มสัญญาณแอนะลอกด้วยความถี่ f_s ที่มากกว่า $2f_{max}$ เราจะสามารถสร้างสัญญาณแอนะลอกคืนมาได้โดยสมบูรณ์” นั่นก็คือ สัญญาณไม่ต่อเนื่องที่เกิดจากการสุ่มจะเป็นตัวแทนที่สมบูรณ์ของสัญญาณต้นฉบับ โดยไม่มีการผิดเพี้ยนเลย อย่าลืมหมายเหตุไว้ด้วยความเข้าใจเองด้วยว่า ในทางปฏิบัติ f_s ควรมากกว่า $2f_{max}$ พอประมาณเพื่อชดเชยการที่เราหาตัวกรองอุดมคติไม่ได้ ทฤษฎีการสุ่มสัญญาณนี้กำเนิดขึ้นตั้งแต่ปี ค.ศ. ….. โดย Shannon และเป็นการเปิดยุคของการสื่อสารด้วยสัญญาณดิจิตอล, การเก็บสัญญาณด้วยดิจิตอล และก็ตามมาด้วยการประมวลผลสัญญาณดิจิตอล

<u>**ตัวอย่างที่ 2.1**</u> แผนภาพข้างล่างแสดงการสื่อสารด้วยสัญญาณดิจิตอล หรือการเก็บสัญญาณดิจิตอล (ยังไม่มีส่วนประมวลผลสัญญาณ) โดยสัญญาณแอนะลอกถูกสุ่มเป็นสัญญาณดิจิตอลที่ต้นทาง และถูกแปลงกลับที่ปลายทาง ถ้าสัญญาณขาเข้าของระบบในภาพ คือ

$$x(t) = \sin(2\pi f_1 t) + 0.3\cos(2\pi f_2 t)$$



โดยที่ f_1 = 2 kHz และ f_2 = 4 kHz และระบบใช้ความถี่ในการสุ่ม คือ f_s = 14kHz จงหา

1.1) x(n), สมมติว่าเริ่มสุ่มจุดแรกที่ t = 0

1.2) วาดสัญญาณในเชิงความถี่ของ x(t) และ $\hat{x}(t)$ ซึ่งก็คือ $|X(f)|$ และ $|\hat{X}(f)|$

1.3) วาดสัญญาณ $\hat{x}(t)$ และ y(t)

<u>วิธีทำ</u>

1.1) ตำแหน่งเวลาที่เกิดการสุ่มสัญญาณ คือ t = nT, n=0, 1, 2, … แทนค่า t นี้ ลงในสมการ x(t) และใช้ตัวชี้ของฟังก์ชั่นใหม่เป็น n จะได้

$$x(n) = x(t)\Big|_{t = nT}$$

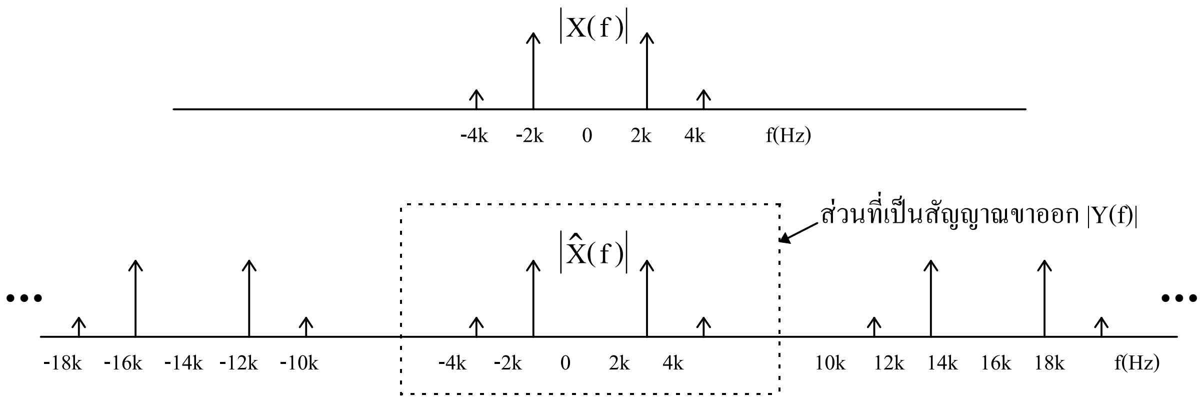
$$= \sin(2\pi n f_1 T) + 0.3\cos(2\pi n f_2 T)$$

$= \sin(2\pi n f_1/f_s) + 0.3\cos(2\pi n f_2/f_s)$

$= \sin(\frac{2\pi n}{7}) + 0.3\cos(\frac{4\pi n}{7})$, n = 0, 1, 2, …

= [0.300 0.7151 0.7046 0.6209 -0.2468 -1.2452 -0.8486 …]

1.2) จะได้ว่าสัญญาณ $|\hat{X}(f)|$ ก็คือ สำเนาของ $|X(f)|$ ที่เกิดขึ้นทุก ๆ ความถี่ …, $-2f_s$, $-f_s$, 0, f_s, $2f_s$, … เป็นศูนย์กลาง ดังแสดงในรูปที่ 2.9

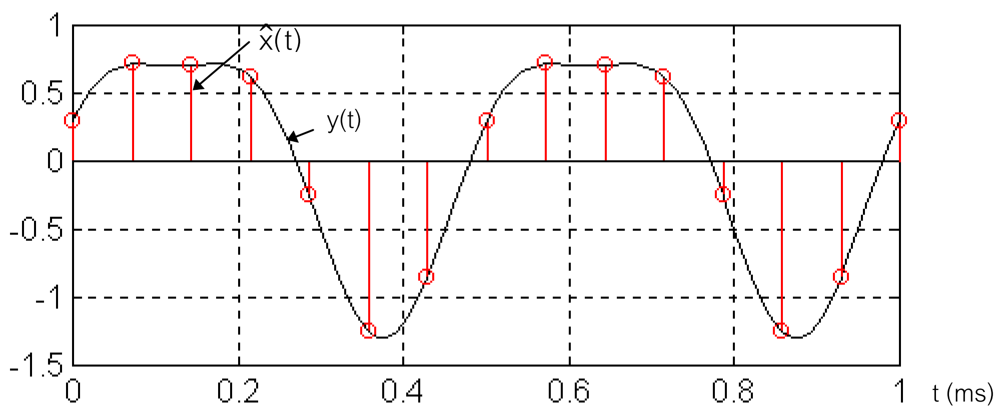


***รูปที่ 2.9** สเปกตรัมของ x(t) และ $\hat{x}(t)$ เมื่อสุ่มด้วย f_s = 14kHz*

1.3) สัญญาณ $\hat{x}(t)$ คือสัญญาณเดียวกับ x(n) แต่มองในเชิงเวลา ซึ่งจะได้ว่าแต่ละจุดอยู่ที่เวลา 0, T, 2T, 3T, …

สำหรับสัญญาณ y(t) คือสัญญาณที่ผ่านตัวกรองสร้างสัญญาณคืนแล้ว ซึ่งในที่นี้จะได้ความถี่ 2kHz และ 4kHz เป็นสัญญาณขาออก หรือ

$y(t) = \sin(2\pi f_1 t) + 0.3\cos(2\pi f_2 t)$, f_1 = 2kHz และ f_2 = 4kHz



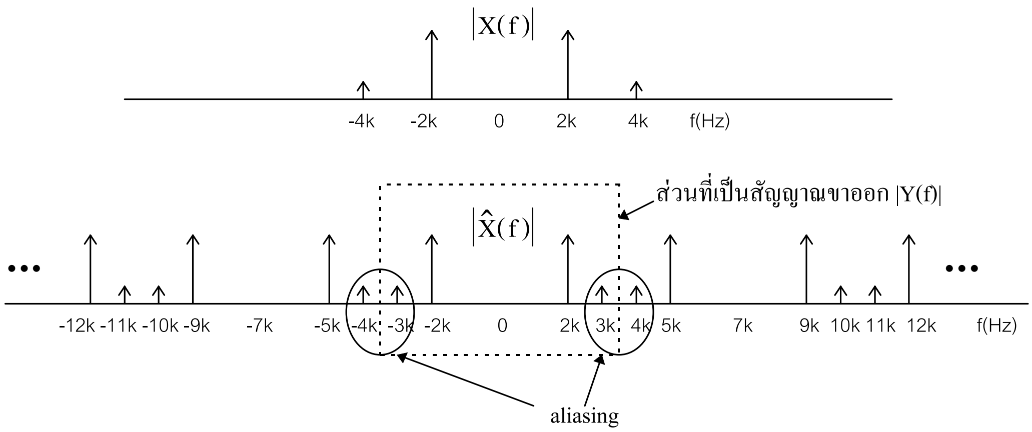
***รูปที่ 2.10** สัญญาณในเชิงเวลาของ y(t) และ $\hat{x}(t)$ เมื่อ f_s = 14kHz*

<u>ตัวอย่างที่ 2.2</u> ทำตัวอย่างที่ 2.1 ซ้ำ โดยใช้ $f_s = 7$ kHz ให้ระบุในภาพของ $|\hat{X}(f)|$ ด้วยว่า ส่วนใด คือ aliasing พร้อมทั้งตอบคำถามว่า ต้องใช้ f_s เท่ากับเท่าไรจึงไม่เกิด aliasing

จาก $x(n) = \sin(2\pi n f_1/f_s) + 0.3\cos(2\pi n f_s/f_s)$ คราวนี้แทน $f_s = 7$kHz จะได้

$$= \sin\left(\frac{4\pi n}{7}\right) + 0.3\cos\left(\frac{8\pi n}{7}\right),\ n = 0, 1, 2, \ldots$$

$$= [0.300 \quad 0.7046 \quad -0.2468 \quad -0.8486 \quad \ldots]$$



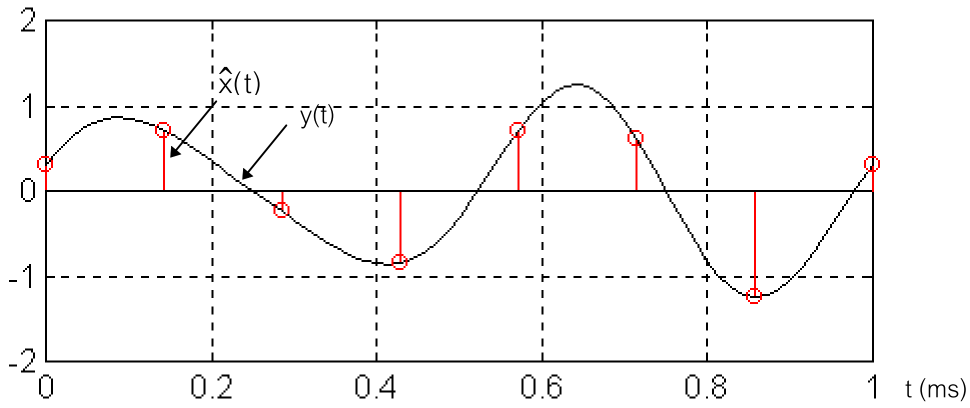
***รูปที่ 2.11** สเปกตรัมของ x(t) และ $\hat{x}(t)$ เมื่อสุ่มด้วย f_s = 7kHz*

จากรูป $|\hat{X}(f)|$ เราพบสำเนาของ X(f) ที่มีศูนย์กลางเป็น 7kHz มีส่วนปลายมาตกที่ความถี่ 3 kHz ซึ่งเป็นส่วนที่เหลื่อมกับสำเนาของ X(f) ที่มีศูนย์กลางเป็น 0 Hz ส่วนที่เหลื่อมกันนี้คือ aliasing ดังแสดงในรูปที่ 2.11 ดังนั้นเมื่อสัญญาณนี้ผ่านตัวกรองสร้างสัญญาณคืน ซึ่งมีความถี่ตัด ที่ 3.5 kHz ก็จะได้สัญญาณขาออก y(t) เป็นองค์ประกอบของความถี่ 2kHz และ 3kHz ซึ่งผิดจากสัญญาณขาเข้าซึ่งมีความถี่ 2kHz และ 4kHz

$y(t) = \sin(2\pi f_1 t) + 0.3\cos(2\pi f_2 t)$, $f_1 = 2$kHz และ $\mathbf{f_2 = 3kHz}$

นี่คือ ความผิดเพี้ยนที่เกิดขึ้นจาก aliasing เราสามารถมองเห็นความผิดเพี้ยนในเชิงเวลาได้ โดยเปรียบเทียบรูปที่ 2.12 กับรูปที่ 2.10 มีข้อสังเกต คือ x(n) ที่ได้จากตัวอย่างนี้ ในช่วงเวลาที่สุ่มเท่ากันจะได้จำนวนจุดเป็นครึ่งหนึ่งของตัวอย่างแรก หรือได้จุดที่สุ่มเป็นจุดเว้นจุดของกรณีแรกพอดี จะเห็นได้ว่ากราฟของ y(t) ที่ผิดเพี้ยนจะทับพอดีกับจุดที่สุ่มได้ทุกจุด ในเชิงเวลา เราสรุปได้คร่าว ๆ ว่า ถ้าจำนวนจุดที่สุ่มได้น้อยไป จะไม่สามารถสร้างสัญญาณจริงคืนได้อย่างถูกต้อง

ในทางทฤษฎี ความถี่ในการสุ่มที่ไม่ทำให้ไม่เกิด aliasing ขึ้นต้องมีค่ามากกว่า 2 เท่าของความถี่สูงสุดที่มีอยู่ในสัญญาณขาเข้า ในที่นี้ความถี่สูงสุด คือ 4Hz ดังนั้น f_s ต้อง<u>มากกว่า</u> 8 kHz



รูปที่ 2.12 *สัญญาณในเชิงเวลาของ y(t) และ $\hat{x}(t)$ เมื่อ f_s = 7kHz*

ผู้เรียนมักมีปัญหาว่า มองไม่เห็นว่า aliasing คืออะไร และคิดไปว่าเป็นสิ่งที่มองไม่เห็น ต้องคิดแต่ในทฤษฎีเท่านั้น จริง ๆ แล้วไม่ใช่ เช่น ในตัวอย่างที่ 2.2 นี้ aliasing อยู่ที่ความถี่ 3 kHz ซึ่งสามารถมองเห็นได้ชัดเจน เพราะมันผ่านออกมาที่สัญญาณขาออกด้วย สัญญาณขาออกมีองค์ประกอบที่ความถี่ 3 kHz อยู่ด้วย และก็เป็นส่วนที่ทำให้สัญญาณผิดเพี้ยนไปจากสัญญาณขาเข้า

<u>ตัวอย่างที่ 2.3</u> ถ้าสัญญาณขาเข้าของระบบในตัวอย่างที่ 2.1 เป็นดังต่อไปนี้ (ไม่มีตัวกรองป้องกัน aliasing) จงระบุว่าจะเกิด aliasing ขึ้นที่ความถี่ใดบ้างในช่วงระหว่าง 0 ถึง $f_s/2$ ให้ใช้ f_s=18kHz

ก) $x(t) = \sin(8000\pi t)\cos(12000\pi t)$

สัญญาณนี้ประกอบด้วยสองความถี่คูณกัน (รูปแบบสัญญาณ $\sin(2\pi ft+\phi)$ คือ รูปแบบของสัญญาณซายน์ หรือสัญญาณความถี่เดียวที่ความถี่ f) แต่ไม่ได้หมายความว่ามันมีองค์ประกอบความถี่ที่สองความถี่นี้ เราจะต้องกระจากมันออกมาให้อยู่ในรูปของผลบวกก่อนจึงจะบอกได้ โดยใช้สูตรตรีโกณมิติว่า $\sin A\cos B = 0.5\{\sin(A+B) + \sin(A-B)\}$ จะได้

$$\begin{aligned} x(t) &= 0.5\sin(20000\pi t) + 0.5\sin(-4000\pi t) \\ &= 0.5\sin(20000\pi t) - 0.5\sin(4000\pi t) \end{aligned}$$

ดังนั้น x(t) มีองค์ประกอบความถี่ 10 kHz และ 2 kHz เมื่อสุ่มด้วยความถี่ f_s = 18 kHz องค์ประกอบความถี่ที่ 10 kHz เป็นส่วนที่เกินย่านไนควิทซ์ ($f_s/2$) ซึ่งจะทำให้เกิด aliasing ขึ้น เราอาจจะใช้วิธีวาดรูปเพื่อหาความถี่ของ aliasing เหมือนที่ทำมา หรืออาจใช้วิธีลัดโดยใช้สูตรว่า

$$\text{ความถี่ aliasing} = |\, m f_s - \text{ความถี่ของสัญญาณขาเข้าที่เกินย่านไนควิทซ์} \,| \qquad (2.9)$$

โดยที่ m มีค่าเป็นจำนวนเต็มที่ทำให้ผลลัพธ์ของค่าความถี่ aliasing อยู่ในช่วงไนควิทซ์ ในข้อนี้ ความถี่ของสัญญาณขาเข้าที่ 10 kHz เลือก m=1 จะได้ว่า

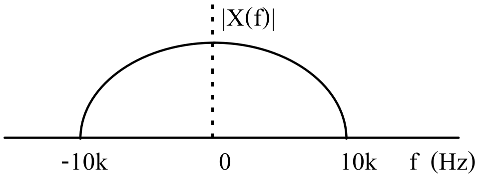
ความถี่ aliasing = | 18k - 10k | = 8 kHz

ขอยกตัวอย่างเพิ่มเติมเพื่อให้เข้าใจวิธีใช้สมการที่ 2.9 ได้ดียิ่งขึ้น เช่น ถ้าสมมติว่าในข้อนี้มีความถี่ของสัญญาณขาเข้าที่ 40 kHz คราวนี้เราจะเลือก m=2 ซึ่งจะได้ว่า

ความถี่ aliasing = | 2 (18k) - 40k | = |-4k| = 4 kHz

ซึ่งถ้า m เท่ากับค่าอื่น จะไม่ทำให้ความถี่ aliasing ตกอยู่ในช่วงไนควิทซ์ (0 ถึง 9 kHz) การใช้ m=2 นี้ หมายถึงว่า aliasing ที่เกิดขึ้นมาจากสำเนาของ X(f) ที่อยู่รอบจุดความถี่ $2f_s$

ข) x(t) ซึ่งมีสเปกตรัมดังในรูป

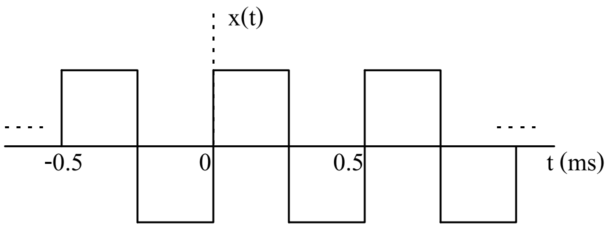


ความถี่ในช่วง 9 ถึง 10 kHz จะทำให้เกิด aliasing โดยที่

aliasing อยู่ในช่วงความถี่ = f_s - 10 kHz ถึง f_s - 9 kHz

= 8 kHz ถึง 9 kHz

ค) x(t) ซึ่งเป็นสัญญาณรายคาบสี่เหลี่ยมดังในรูป



x(t) นี้เป็นสัญญาณสี่เหลี่ยมที่มีคาบเท่ากับ 0.5 ms เพราะฉะนั้น มีความถี่พื้นฐานที่ f_0 = 1/0.5ms = 2 kHz ถ้าเปิดสูตรเกี่ยวกับอนุกรมฟูริเยร์ จะพบว่า สัญญาณสี่เหลี่ยมรายคาบจะมีความถี่ที่ประกอบด้วยความถี่พื้นฐาน และความถี่ฮาร์มอนิกคี่ (คือ $3f_0$, $5f_0$, $7f_0$, ...) ดังนั้น สัญญาณ x(t) ในที่นี้จะประกอบด้วยความถี่ 2 kHz, 6 kHz, 10 kHz, 14 kHz, 18 kHz, ...

ความถี่ที่เกิน 9 kHz คือ 10 kHz, 14 kHz, 18 kHz, ... เป็นองค์ประกอบที่จะทำให้เกิด aliasing ในทางปฏิบัติที่ฮาร์มอนิกสูง ๆ ของสัญญาณสี่เหลี่ยมจะมีพลังงานต่ำลง ๆ ซึ่งเราอาจจะไม่ต้องวิเคราะห์ที่ความถี่สูงเหล่านี้ก็ได้ แต่ในที่นี้จะลองวิเคราะห์ดูที่ทุกความถี่ก่อน ดูว่าจะเกิดความถี่ alaiasing ที่ใดบ้าง

ที่ความถี่ 10 kHz จะทำให้เกิด aliasing ที่	|18k - 10k|	= 8 kHz
ที่ความถี่ 14 kHz จะทำให้เกิด aliasing ที่	|18k - 14k|	= 4 kHz
ที่ความถี่ 18 kHz จะทำให้เกิด aliasing ที่	|18k - 18k|	= 0 kHz
ที่ความถี่ 22 kHz จะทำให้เกิด aliasing ที่	|18k - 22k|	= 4 kHz
ที่ความถี่ 26 kHz จะทำให้เกิด aliasing ที่	|18k - 26k|	= 8 kHz
ที่ความถี่ 30 kHz จะทำให้เกิด aliasing ที่	|2(18k) - 30k|	= 6 kHz
ที่ความถี่ 34 kHz จะทำให้เกิด aliasing ที่	|2(18k) - 34k|	= 2 kHz
ที่ความถี่ 38 kHz จะทำให้เกิด aliasing ที่	|2(18k) - 38k|	= 2 kHz
ที่ความถี่ 42 kHz จะทำให้เกิด aliasing ที่	|2(18k) - 42k|	= 6 kHz
ที่ความถี่ 46 kHz จะทำให้เกิด aliasing ที่	|3(18k) - 46k|	= 8 kHz

. . .

เห็นได้ว่า ในข้อนี้ aliasing เกิดซ้ำ ๆ กันที่ความถี่เดิม ๆ พอดี ถ้าวิเคราะห์ต่อไปที่ความถี่สูงขึ้นจะไม่พบความถี่ aliasing อื่น ๆ นอกเหนือจากที่ได้แสดงมานี้เลย ดังนั้น ในข้อนี้ความถี่ aliasing ที่เกิดขึ้นทั้งหมดอยู่ที่ความถี่ 0, 2 kHz, 4 kHz, 6 kHz, และ 8 kHz

บทที่ 3

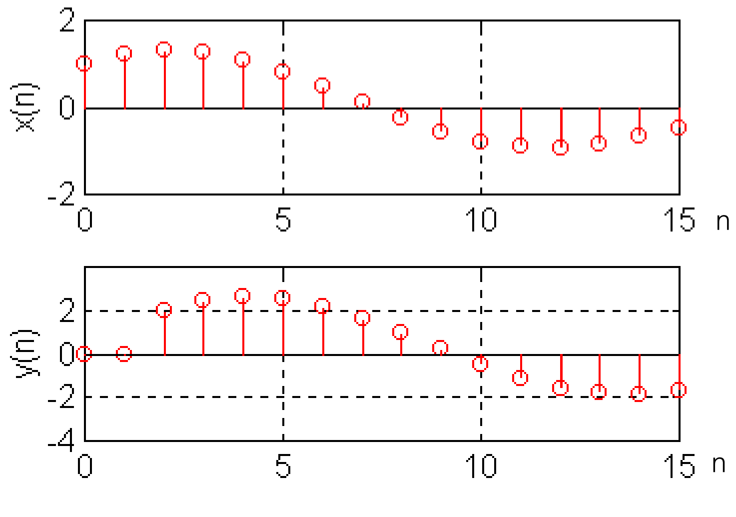
ระบบแบบไม่ต่อเนื่อง (Discrete-Time Systems)

ในบทนี้จะกล่าวถึงระบบแบบไม่ต่อเนื่อง ซึ่งระบบการประมวลผลสัญญาณดิจิตอลที่จะศึกษาต่อไปก็คือ ระบบแบบไม่ต่อเนื่องนี้เอง จึงมีความจำเป็นที่จะต้องเข้าใจเนื้อหาส่วนนี้ให้ดี ถ้าใครได้ศึกษาระบบแบบต่อเนื่อง (Continuous-Time System) มาแล้ว ก็จะพบว่า ระบบทั้งสองมีลักษณะแนวทางในการคิด และการวิเคราะห์คล้ายคลึงกัน ไม่ว่าในเรื่องของความเป็นเชิงเส้น และไม่แปรตามเวลา, การหาผลตอบ, การแปลงระหว่างสัญญาณในเชิงเวลากับความถี่, เสถียรภาพ และอื่น ๆ

ระบบแบบไม่ต่อเนื่องคืออะไร

ระบบแบบไม่ต่อเนื่องก็คือ ส่วนประมวลผลที่ได้กล่าวถึงในบทที่ 1 นั่นเอง มีสัญญาณขาเข้า x(n) และสัญญาณขาออก y(n) เป็นสัญญาณไม่ต่อเนื่อง ระบบแบบไม่ต่อเนื่องจะทำการประมวลผลหรือ ทำแปลงจากสัญญาณขาเข้าไปเป็นสัญญาณขาออก การประมวลผลนี้อาจเป็นการกระทำใด ๆ ก็ได้ เช่น

ก) $y(n) = 2x(n-2)$ ให้สัญญาณขาออกเป็นสัญญาณขาเข้าที่ถูกขยายขึ้นสองเท่า และล้าหลัง (delay) ไป 2 ลำดับเวลา ดังในรูปที่ 3.1



รูปที่ 3.1 *แสดงตัวอย่างของสัญญาณขาเข้า และขาออกของระบบไม่ต่อเนื่อง* $y(n) = 2x(n - 2)$

สังเกตว่า x(n) เริ่มเป็นค่าลบที่ n=8 แต่ y(n) เริ่มที่ n=10 (ล้าหลังไป 2 ลำดับเวลา) ขอให้จำให้แม่นว่ารูปแบบ x(n-k) คือ สัญญาณ x(n) ที่ล้าหลังลง k ลำดับ และ x(n+k) คือ สัญญาณ x(n) ที่นำหน้าขึ้น k ลำดับ (โดย k เป็นค่าจำนวนเต็มบวก)

ข) y(n) = 0.5 (x(n) + x(n-1)) สมการเดียวกับที่ใช้ในบทที่ 1 ให้สัญญาณขาออกเป็นการเฉลี่ยสองตำแหน่งของสัญญาณขาเข้าที่ติดกัน สมการนี้เป็นตัวกรองผ่านต่ำชนิดหนึ่ง

ค) y(n) = 0.5y(n-1) + x(n) เป็นสมการของตัวกรองผ่านต่ำอีกชนิดหนึ่ง ซึ่งใช้สัญญาณขาออกในอดีตมาคำนวณด้วย

ง) $y(n) = x^2(n)$ ให้สัญญาณขาออกเป็นกำลังสองของสัญญาณขาเข้า

จ) y(n) = x(2n) สมการนี้ดูยากสักหน่อย ถ้าลองแทนค่า n = 0, 1, 2, … ลงไป จะได้ y(0) = x(0), y(1) = x(2), y(2) = x(4), … นั่นคือ เราจะเห็นสัญญาณขาออกเป็นค่าเว้นค่าของสัญญาณขาเข้า หรือคือ [x(0), x(2), x(4), x(6), …] ดังนั้น สมการนี้ทำการลดอัตราสุ่มลงเท่าหนึ่งนั่นเอง

ความเป็นเชิงเส้น และไม่แปรตามเวลา (Linearity and Time Invariance)

ระบบที่เราจะสนใจเป็นพิเศษในเนื้อหาเบื้องต้นของการประมวลผลสัญญาณ คือ ระบบแบบเชิงเส้น และไม่แปรตามเวลา ก่อนที่จะดูถึงความสำคัญของระบบแบบนี้ เรามาดูวิธีคิดก่อนว่าระบบแบบใดที่มีคุณสมบัติดังกล่าว

สมมติว่าระบบแบบไม่ต่อเนื่องที่เราสนใจระบบหนึ่ง เมื่อป้อนสัญญาณขาเข้า $x_1(n)$ ทำให้เกิดสัญญาณขาออก $y_1(n)$ และเมื่อป้อนสัญญาณขาเข้า $x_2(n)$ ทำให้เกิดสัญญาณขาออก $y_2(n)$

ถ้าให้ x(n) เป็นสัญญาณขาเข้าใหม่ ที่เกิดจากการคำนวณแบบเชิงเส้นระหว่าง $x_1(n)$ และ $x_2(n)$ นั่นคือ

$$x(n) = a_1 x_1(n) + a_2 x_2(n) \tag{3.1}$$

โดย a_1 และ a_2 เป็นค่าคงที่ใด ๆ ที่ไม่เท่ากับศูนย์ เรากล่าวว่าระบบนี้เป็น<u>ระบบแบบเชิงเส้น</u> (linear system) ถ้า x(n) นี้ทำให้เกิดสัญญาณขาออก คือ

$$y(n) = a_1 y_1(n) + a_2 y_2(n) \tag{3.2}$$

นั่นคือ จะต้องได้ y(n) เป็นการคำนวณแบบเชิงเส้นระหว่าง $y_1(n)$ และ $y_2(n)$ โดยที่มีสัมประสิทธิ์ที่ใช้คูณ (a_1 และ a_2) ตัวเดียวกันกับสัญญาณขาเข้า

ระบบเชิงเส้นบอกเราว่า

- ถ้าเราคูณสัญญาณขาเข้าด้วยค่าสัมประสิทธิ์ค่าหนึ่ง สัญญาณขาออกก็เปลี่ยนไปด้วยตัวคูณเดียวกัน

- ถ้าสัญญาณขาเข้าเป็นผลรวมของสัญญาณหลาย ๆ สัญญาณ สัญญาณขาออกก็จะเป็นผลรวมของสัญญาณขาออก ที่เกิดจากสัญญาณขาเข้าแต่ละตัวมารวมกัน

คุณสมบัติของระบบที่เป็นเชิงเส้นนี้ เรียกอีกอย่างหนึ่งว่า คุณสมบัติ superposition

สำหรับคุณสมบัติ<u>ความไม่แปรตามเวลา</u>ของระบบไม่ต่อเนื่อง มีเงื่อนไขว่า

ถ้าสัญญาณขาเข้า x(n) ทำให้เกิดสัญญาณขาออก y(n) แล้ว ถ้าให้สัญญาณขาเข้าล้าหลัง หรือนำหน้าไปเท่ากับ k ลำดับ คือ กลายเป็น x(n-k) ระบบจะให้สัญญาณขาออกเป็น y(n-k) (ได้สัญญาณเหมือนเดิม และล้าหลัง หรือนำหน้าไปเท่ากับสัญญาณขาเข้า)

ระบบแบบไม่แปรตามเวลา (time-invariant system) บอกเราว่า ไม่ว่าเราจะใส่สัญญาณขาเข้าที่เวลาใด เราจะได้สัญญาณขาออกที่เหมือนเดิมเสมอ นั่นก็คือในระบบมีค่าพารามิเตอร์ หรือสัมประสิทธิ์ต่าง ๆ คงที่ ไม่แปรตามเวลา หรือแปรตามสภาวะแวดล้อมใด ๆ เลย

<u>ตัวอย่างที่ 3.1</u> จงทดสอบว่าระบบที่มีความสัมพันธ์ระหว่างสัญญาณขาออก และขาเข้าตามสมการเหล่านี้เป็นระบบแบบเชิงเส้น และไม่แปรตามเวลาหรือไม่

1) $y(n) = ax(n) + bx(n-1)$

1.1) ทดสอบว่าเป็นเชิงเส้นหรือไม่ ?

ใส่ $x_1(n)$ เป็นสัญญาณขาเข้า จะได้ $y_1(n) = ax_1(n) + bx_1(n-1)$

ใส่ $x_2(n)$ เป็นสัญญาณขาเข้า จะได้ $y_2(n) = ax_2(n) + bx_2(n-1)$

สมมติให้ $x(n) = a_1x_1(n) + a_2x_2(n)$ ใส่ x(n) เข้าไปในระบบ จะได้

$y(n) = a\,[a_1x_1(n) + a_2x_2(n)] + b\,[a_1x_1(n-1) + a_2x_2(n-1)]$

จัดลำดับเทอมทั้งสี่ใหม่จะได้

$y(n) = a_1\,[ax_1(n) + bx_1(n-1)] + a_2\,[ax_2(n) + bx_2(n-1)]$

$\quad = a_1y_1(n) + a_2y_2(n)$

เห็นได้ว่า x(n) ทำให้เกิด y(n) ที่ตรงตามคุณสมบัติ ดังนั้น ระบบนี้เป็นแบบเชิงเส้น

1.2) ทดสอบว่าแปรตามเวลาหรือไม่ ?

สมมติให้ $x_d(n) = x(n-k)$ ลองใส่ $x_d(n)$ เข้าไปในระบบ จะได้ผลตอบคือ

$y_d(n) = ax_d(n) + bx_d(n-1)$

$= ax(n-k) + bx((n-1)-k)$

$= ax(n-k) + bx((n-k)-1)$

$= y(n-k)$

เห็นได้ว่า x(n-k) ทำให้เกิด y(n-k) นั่นคือ ระบบนี้ไม่แปรตามเวลา

2) $y(n) = x(2n)$

2.1) ทดสอบว่าเป็นเชิงเส้นหรือไม่ ?

ใส่ $x_1(n)$ เป็นสัญญาณขาเข้า จะได้ $y_1(n) = x_1(2n)$

ใส่ $x_2(n)$ เป็นสัญญาณขาเข้า จะได้ $y_2(n) = x_2(2n)$

สมมติให้ $x(n) = a_1x_1(n) + a_2x_2(n)$ ใส่ x(n) เข้าไปในระบบ จะได้

$y(n) = x(2n) = a_1x_1(2n) + a_2x_2(2n)$

ซึ่งเห็นได้ชัดว่า $y(n) = a_1y_1(n) + a_2y_2(n)$

ดังนั้น ระบบนี้เป็นแบบเชิงเส้น

2.2) ทดสอบว่าแปรตามเวลาหรือไม่ ?

สมมติให้ $x_d(n)= x(n-k)$ ลองใส่ $x_d(n)$ เข้าไปในระบบ จะได้ผลตอบคือ

$y_d(n) = x_d(2n) = x(2n-k)$

แต่ $y(n-k) = x(2(n-k)) = x(2n - 2k)$

เห็นได้ชัดว่า $y_d(n) \neq y(n-k)$ กล่าวคือ สัญญาณขาเข้า x(n-k) ไม่ทำให้เกิด y(n-k) ดังนั้น ระบบนี้แปรตามเวลา (time-varying system)

ระบบในข้อสองนี้เรียกว่าระบบลดอัตราการสุ่ม (downsampler) ซึ่งจะกล่าวถึงในบทที่ 11 ขอยกตัวอย่างสัญญาณเพื่อแสดงให้เห็นความไม่แปรตามเวลาอย่างชัดเจนของระบบ ดังนี้

$x(n) = [\, x_0, x_1, x_2, x_3, x_4, x_5, \ldots \,]$ ได้ขาออก คือ $y(n) = [\, x_0, x_2, x_4, x_6, \ldots \,]$

แต่เมื่อสัญญาณขาเข้าถูกดึงให้ล้าหลังไป 1 ตำแหน่ง นั่นคือ

$x_d(n) = x(n-1) = [\, 0, x_0, x_1, x_2, x_3, x_4, x_5, \ldots \,]$ ได้ขาออก คือ $y_d(n) = [\, 0, x_1, x_3, x_5, x_7, \ldots \,]$

ถ้าระบบเป็นแบบไม่แปรตามเวลา เราควรจะได้ $y_d(n) = y(n-1)$ แต่เรากลับได้ $y_d(n)$ ที่ต่างจาก y(n) โดยสิ้นเชิงดังที่แสดง ดังนั้น ระบบนี้แปรตามค่าของเวลาที่ล้าหลัง

3) $y(n) = 2x(n) + 5$

3.1) ทดสอบว่าเป็นเชิงเส้นหรือไม่ ?

ใส่ $x_1(n)$ เป็นสัญญาณขาเข้า จะได้ $y_1(n) = 2x_1(n) + 5$

ใส่ $x_2(n)$ เป็นสัญญาณขาเข้า จะได้ $y_2(n) = 2x_2(n) + 5$

สมมติให้ $x(n) = a_1x_1(n) + a_2x_2(n)$ ใส่ $x(n)$ เข้าไปในระบบ จะได้

$$y(n) = 2\,[a_1x_1(n) + a_2x_2(n)] + 5$$

ซึ่งจะได้ $y(n) = 2a_1x_1(n) + 2a_2x_2(n) + 5$

แต่ $a_1y_1(n) + a_2y_2(n) = a_1(2x_1(n) + 5) = a_2(2x_2(n) + 5)$

$$= 2a_1x_1(n) + 2a_2x_2(n) + 5a_1 + 5a_2$$

เห็นได้ว่า $y(n) \neq a_1y_1(n) + a_2y_2(n)$ ดังนั้น ระบบนี้ไม่เป็นเชิงเส้น (nonlinear system)

3.2) ทดสอบว่าแปรตามเวลาหรือไม่ ?

สมมติให้ $x_d(n) = x(n-k)$ ลองใส่ $x_d(n)$ เข้าไปในระบบ จะได้ผลตอบคือ

$$\begin{aligned} y_d(n) &= 2x_d(n) + 5 \\ &= 2x(n-k) + 5 \\ &= y(n-k) \end{aligned}$$

เห็นได้ว่า $x(n-k)$ ทำให้เกิด $y(n-k)$ นั่นคือ ระบบนี้ไม่แปรตามเวลา

เห็นจะพอได้แล้วสำหรับตัวอย่างการทดสอบว่าระบบเป็นเชิงเส้น และไม่แปรตามเวลาหรือไม่ รูปแบบระบบที่เราจะสนใจในเบื้องต้นนี้ คือ

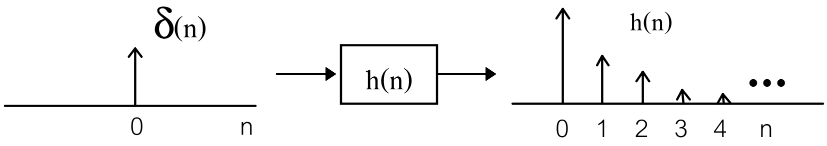
$$y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2) + \ldots - b_1y(n-1) - b_2y(n-2) - \ldots$$

$$\text{หรือ} \quad y(n) = \sum_{i=0}^{\infty} a_i x(n-i) - \sum_{i=1}^{\infty} b_i y(n-i) \qquad (3.3)$$

เมื่อ a_i และ b_i เป็นค่าคงที่ (ไม่แปรตาม n) รูปแบบสมการนี้ จะเป็นระบบแบบเชิงเส้น และไม่แปรตามเวลาเสมอ

ผลตอบสนองต่อสัญญาณอิมพัลส์ (Impulse Response)

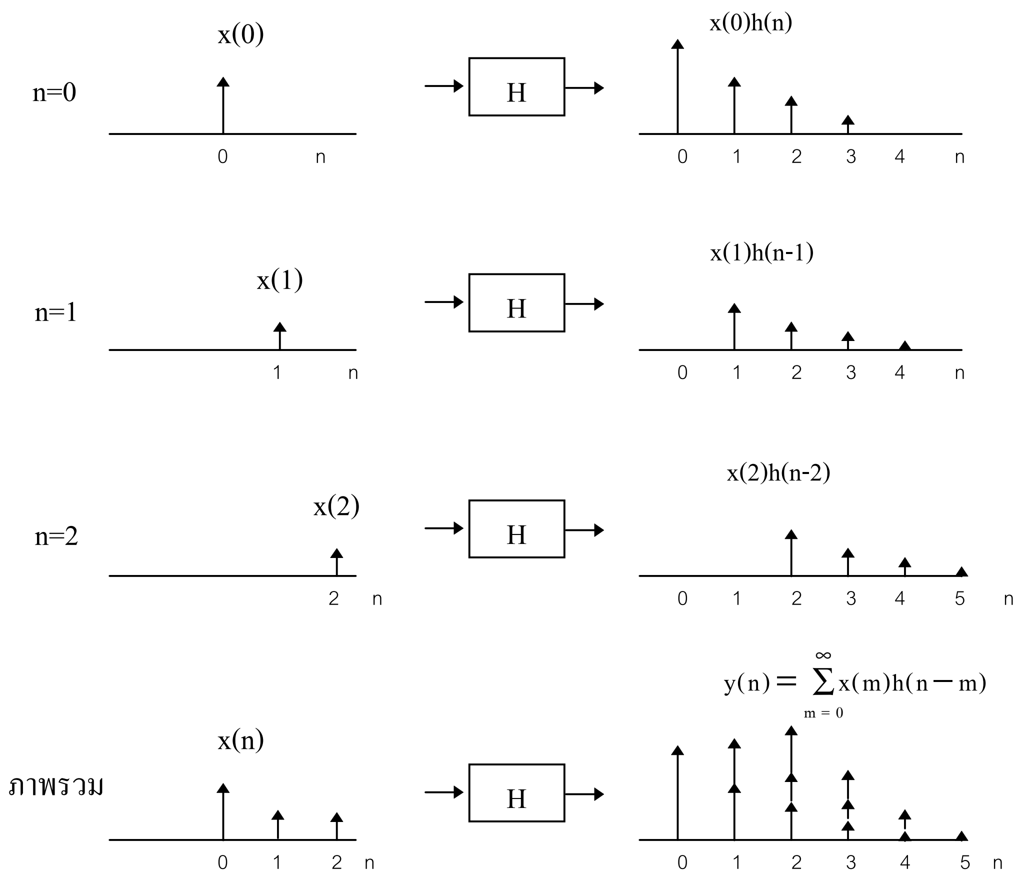
ระบบที่เป็นเชิงเส้น และไม่แปรตามเวลา มีคุณลักษณะพิเศษ คือ สามารถระบุคุณลักษณะของระบบได้โดยสมบูรณ์ด้วยผลตอบสนองต่อสัญญาณอิมพัลส์ ผลตอบสนองต่อสัญญาณอิมพัลส์ ซึ่งจะใช้สัญลักษณ์แทนว่า h(n) เสมอในหนังสือเล่มนี้ คือ สัญญาณขาออกของระบบเมื่อมีสัญญาณขาเข้าเป็นสัญญาณอิมพัลส์ ($\delta(n)$) ดังแสดงในรูปที่ 3.2



***รูปที่ 3.2** ผลตอบสนองต่อสัญญาณอิมพัลส์*

ผลตอบสนองต่อสัญญาณอิมพัลส์สามารถเป็นตัวแทนของระบบได้ เนื่องจากเมื่อรู้ผลตอบสนองต่อสัญญาณอิมพัลส์ เราจะสามารถหาผลตอบของระบบเมื่อสัญญาณขาเข้าเป็นสัญญาณ ใด ๆ ได้ ขอพิสูจน์โดยสมมติว่า x(n) เป็นสัญญาณขาเข้าที่เป็นสัญญาณไม่ต่อเนื่องใด ๆ เพื่อให้ง่ายต่อการวิเคราะห์ เราจะสมมติว่า x(n) เริ่มมีค่าที่ n=0 เราสามารถเขียน x(n) กระจายเป็นผลบวกของสัญญาณอิมพัลส์ที่ค่าเวลา n=0, 1, 2, … ได้ดังนี้

$$x(n) = x(0)\delta(n) + x(1)\delta(n-1) + x(2)\delta(n-2) + \dots \tag{3.4}$$



***รูปที่ 3.3** ที่มาของการหาผลตอบของระบบโดยวิธีคอนโวลูชัน*

แต่ละเทอมในสมการจะแทนค่าแต่ละตำแหน่งเวลาในสัญญาณ x(n) เช่น ถ้าลองแทนค่า n=1 จะมีเฉพาะเทอม $x(1)\delta(n-1)$ เท่านั้นที่มีค่าไม่เท่ากับศูนย์ และจะได้ $x(n)_{n=1} = x(1)$ เป็นต้น

ด้วยคุณสมบัติความไม่แปรตามเวลาของระบบ เมื่อ $\delta(n)$ ทำให้เกิดผลตอบ h(n) ก็จะได้ว่า $\delta(n-1)$ ทำให้เกิดผลตอบ h(n-1) และ $\delta(n-2)$ ทำให้เกิดผลตอบ h(n-2) เป็นเช่นนี้ไปเรื่อย ๆ

อาศัยคุณสมบัติความเป็นเชิงเส้นของระบบ เราก็จะได้ว่า $x(0)\delta(n)$ ทำให้เกิดผลตอบ x(0)h(n) และ $x(1)\delta(n-1)$ ทำให้เกิดผลตอบ x(1)h(n-1) เป็นเช่นนี้ไปเรื่อย ๆ ดังแสดงในรูป และผลตอบโดยรวม คือ y(n) ก็สามารถคิดได้ว่า มาจากผลรวมของผลตอบย่อยแต่ละตัว นั่นคือ

$$y(n) = x(0)h(n) + x(1)h(n-1) + x(2)h(n-2) + \ldots$$

$$y(n) = \sum_{m=0}^{\infty} x(m)h(n-m) \qquad (3.5)$$

ถ้าไม่จำกัดว่า x(n) เริ่มมีค่าที่ n=0 ก็จะได้เป็นสมการทั่วไปของผลตอบของระบบ เป็น

$$y(n) = \sum_{m=-\infty}^{\infty} x(m)h(n-m) \qquad (3.6)$$

ขอนิยามการกระทำระหว่าง x(n) และ h(n) ในสมการนี้ว่า คอนโวลูชันแบบไม่ต่อเนื่อง (discrete convolution) ผู้ที่เคยศึกษาเรื่องระบบแบบต่อเนื่องมาจะพบว่า สมการนี้คล้ายกับคอนโวลูชันแบบต่อเนื่องมาก เพียงแค่เปลี่ยนจากการบวก เป็นการอินทิเกรตเท่านั้น เราจะใช้สัญลักษณ์ * แทนการกระทำคอนโวลูชันนี้ และสามารถเขียนสมการของ y(n) ได้ใหม่เป็น

$$y(n) = x(n) * h(n) \qquad (3.7)$$

ถ้าเราแทน m ด้วย n - k ในสมการของคอนโวลูชัน จะได้ว่า

$$y(n) = \sum_{n-k=-\infty}^{\infty} x(n-k)h(n-(n-k))$$

$$y(n) = \sum_{k=n-\infty}^{n+\infty} h(k)x(n-k)$$

เนื่องจาก n เป็นปริมาณที่จำกัด ดังนั้น $n-\infty$ จึงมีค่าเทียบเท่ากับ $-\infty$ และ $n+\infty$ ก็มีค่าเทียบเท่ากับ ∞ จะได้ว่า สมการนี้กลายเป็น

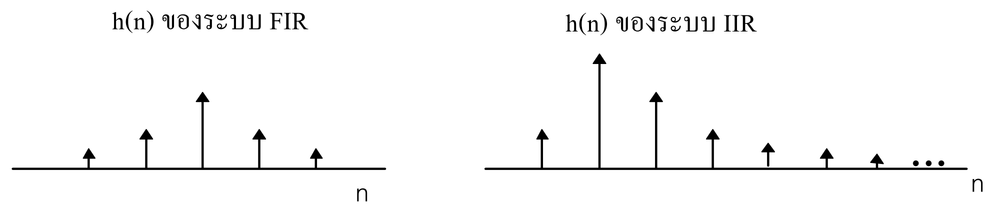
$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) \qquad (3.8)$$

สมการที่ 3.5 และ 3.8 นี้ แสดงให้เห็นว่าการคูณคอนโวลูชัน มีคุณสมบัติสลับที่ของตัวถูกคูณได้ หรือ $y(n) = x(n) * h(n) = h(n) * x(n)$ แต่สูตรในสมการที่ 3.8 นี้ (ใช้ n-k เป็นตัวชี้ของ x) จะสามารถนำมาใช้งานได้สะดวกกว่าในการประมวลผลแบบเวลาจริงดังจะได้เห็นต่อไป ขออย่าได้งงในการสลับตัวแปรระหว่าง m กับ k เพราะ m และ k ในสมการข้างต้นเป็นเพียงตัวชี้เพื่อให้เกิดการบวกกันใน Σ เท่านั้น เราจะใช้ตัวแปรชื่ออะไรก็ได้ แต่ตัวชี้ของเวลาจริง ๆ คือ n

ตัวกรองแบบ FIR และ IIR

ในที่นี้ขอแนะนำคำที่จะพบบ่อยมากในวิชานี้ คือ FIR และ IIR เป็นคำที่ใช้แยกแยะประเภทของระบบตามลักษณะของผลตอบสนองต่ออิมพัลส์ของระบบ คำว่า FIR ย่อมาจาก Finite Impulse Response ระบบแบบ FIR จะมีความยาวของ h(n) จำกัด ถ้า h(n) ของระบบมีความยาวจำกัดเท่ากับ N จุด เรากล่าวว่า ระบบนี้เป็นตัวกรองแบบ FIR ที่มีอันดับ (order) เท่ากับ N-1 ถ้าสมมติว่า h(n) มีค่าแรกที่ n=0 จะได้ว่า h(n) มีค่าเป็นศูนย์ ที่ n<0 และ n>N-1

สำหรับ IIR คือ Infinite Impulse Response ระบบแบบ IIR จะมี h(n) ไม่จำกัด ซึ่ง h(n) อาจมีความยาวเป็นไม่จำกัดในช่วงที่ n เป็นค่าบวก หรือ ลบ หรือ ทั้งสองด้านก็ได้ ดังในรูปที่ 3.4 h(n) มีค่าไปเรื่อย ๆ จนถึงที่เวลาที่ n มีค่าเป็นอนันต์ (ถึงแม้ค่าของ h(n) จะลู่เข้าสู่ศูนย์ที่เวลาอนันต์ ก็ยังถือว่าเป็นระบบแบบ IIR เพราะ เรานับจำนวนจุดของ h(n) ได้ไม่จำกัดอยู่ดี)



***รูปที่ 3.4** ตัวอย่างของผลตอบสนองต่อสัญญาณอิมพัลส์ของตัวกรองแบบ FIR และ IIR*

เราจะศึกษาถึงวิธีการออกแบบตัวกรองทั้งสองในบทที่ 7 และ 8 ในที่นี้ขอให้เข้าใจก่อนว่าระบบไม่ต่อเนื่องที่มีคุณสมบัติเชิงเส้น และไม่แปรตามเวลา สามารถถูกกำหนดลักษณะได้โดย h(n) ซึ่ง h(n) ที่แตกต่างกันก็ยังผลให้เกิดฟังก์ชั่นของระบบที่แตกต่างกันไป ฟังก์ชั่นพื้นฐานที่เราจะเรียนในวิชานี้ก็คือ การกรองความถี่ ซึ่งสามารถออกแบบได้เป็นระบบทั้งแบบ FIR หรือ IIR ก็ได้ ซึ่งแต่ละแบบก็มีข้อดีข้อเสียแตกต่างกันไป

ตัวอย่างที่ 3.2 สมมติว่าตัวกรองแบบ FIR ระบบหนึ่งมี h(n) = [4 2 1] จงหาผลตอบ y(n) เมื่อ x(n) = [1 2 3 2 1] โดยทั้ง h(n) และ x(n) มีค่าแรกที่ n=0

ในที่นี้ เราจะคำนวณคอนโวลูชั่น โดยใช้สมการที่ 3.6 โดยเขียนในรูปของตาราง ดังนี้

n	=	0	1	2	3	4	5	6	7	8	...
ผลตอบจาก x(0) = x(0)h(n)	=	4	2	1	0	0	0	0	0	0	...
ผลตอบจาก x(1) = x(1)h(n-1)	=	0	8	4	2	0	0	0	0	0	...
ผลตอบจาก x(2) = x(2)h(n-2)	=	0	0	12	6	3	0	0	0	0	...
ผลตอบจาก x(3) = x(3)h(n-3)	=	0	0	0	8	4	2	0	0	0	...
ผลตอบจาก x(4) = x(4)h(n-4)	=	0	0	0	0	4	2	1	0	0	...
ผลตอบ y(n) = ผลบวกของทุกแถว	=	4	10	17	16	11	4	1	0	0	...

สังเกตว่าถ้า h(n) มีความยาวจำกัดเท่ากับ L_h จุด และ x(n) มีความยาวจำกัดเท่ากับ L_x จุด จะได้ y(n) มีขนาดเท่ากับ $L_h + L_x - 1$ เสมอ

สมการผลต่าง (Difference Equation)

ระบบแบบไม่ต่อเนื่อง นอกจากสามารถถูกกำหนดลักษณะได้ด้วย h(n) แล้ว ยังสามารถถูกกำหนดได้ด้วยสมการผลต่าง ซึ่งสมการผลต่างในที่นี้ คือ สมการที่บ่งบอกความสัมพันธ์ในเชิงเวลาระหว่างสัญญาณขาเข้าของระบบ คือ x(n) และสัญญาณขาออกของระบบ คือ y(n) สมการผลต่าง คือสิ่งที่จะใช้สำหรับสร้างตัวประมวลผล เพราะมันบอกว่าเราจะหา y(n) ได้ด้วยการคำนวณอย่างไร

สมการผลต่างของระบบแบบเชิงเส้น และไม่แปรตามเวลา มีรูปแบบเฉพาะตามสมการ 3.3 ที่ได้กล่าวถึงไปแล้ว ขอยกมาเขียนอีกครั้งหนึ่ง ดังนี้

$$y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2) + \ldots - b_1y(n-1) - b_2y(n-2) - \ldots$$

หรือ เขียนให้กระทัดรัดลงจะได้

$$y(n) = \sum_{i=0}^{\infty} a_i x(n-i) - \sum_{i=0}^{\infty} b_i y(n-i) \qquad (3.9)$$

โดยทั่วไป สำหรับตัวกรอง FIR จะมีค่าสัมประสิทธ์ b_i เป็นศูนย์ทั้งหมด หรือไม่มีการใช้ค่าผลตอบในอดีตนั่นเอง แต่สำหรับตัวกรอง IIR จะมีค่า b_i อย่างน้อย 1 ตัวไม่เท่ากับศูนย์

ตัวอย่าง 3.3 จงหาผลตอบของโจทย์ในตัวอย่างที่ 3.2 โดยใช้วิธีคำนวณหาจากสมการผลต่าง

สมการผลต่างของ FIR สามารถหาได้จาก สมการคอนโวลูชั่นในรูปสมการที่ 3.8 คือ

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

เมื่อแจกแจงการบวกออกมาเป็นเทอมต่าง ๆ จะได้ รูปทั่วไปของสมการผลต่างของ FIR คือ

$$y(n) = h(0)x(n) + h(1)x(n-1) + h(2)x(n-2) + \ldots + h(N-1)x(n-N+1) \qquad (3.10)$$

โดย N แทนจำนวนจุดของสัญญาณ h(n) สมการนี้จริง ๆ แล้วก็คือ สมการเดียวกับ 3.9 โดยที่ $a_i = h(i)$ และ $b_i = 0$ สำหรับในข้อนี้ h(n) มีค่าที่ n = 0, 1, 2 เท่านั้น เพราะฉะนั้น จะได้ สมการผลต่างของระบบนี้ คือ

$$y(n) = h(0)x(n) + h(1)x(n-1) + h(2)x(n-2)$$

$$y(n) = 4x(n) + 2x(n-1) + x(n-2)$$

เราจะใช้สมการนี้สำหรับหาผลตอบ y(n) ได้ โดยแทนค่า n = 0, 1, 2, 3 … จะได้

เมื่อ n=0, $y(0) = 4\,x(0) + 2\,x(-1) + x(-2) = 4(1) + 2(0) + 0 = 4$

เมื่อ n=1, $y(1) = 4\,x(1) + 2\,x(0) + x(-1) = 4(2) + 2(1) + 0 = 10$

เมื่อ n=2, $y(2) = 4\,x(2) + 2\,x(1) + x(0) = 4(3) + 2(2) + (1) = 17$

. . .

ใช้วิธีนี้ หา y(n) ไปเรื่อย ๆ จะพบว่าได้ค่าเท่ากับในตัวอย่าง 3.2 ทุกประการ จะเห็นได้ว่า ถ้าเราต้องการหาผลตอบที่ละค่า การหาผลตอบโดยวิธีการใช้สมการผลต่างจะทำได้สะดวกกว่าวิธีในตัวอย่างที่ 3.2 ที่สำคัญก็คือ วิธีที่ใช้สมการผลต่างนี้มีความเหมาะสมอย่างยิ่งในการประมวลผลแบบเวลาจริง เพราะเราสามารถรับค่าสัญญาณขาเข้า 1 ค่า และคำนวณหาสัญญาณขาออก 1 ค่าได้ทันที โดยไม่ต้องรอให้สัญญาณขาเข้าเข้ามาทั้งหมด

สำหรับตัวกรองแบบ IIR จะมีความสัมพันธ์ระหว่าง h(n) กับสมการผลต่างที่ค่อนข้างดูได้ยาก เนื่องจาก IIR จะมีการป้อนกลับ (feed back) หรือมีการใช้ผลตอบในอดีตมาคำนวณด้วย ซึ่งการป้อนกลับนี้เองเป็นส่วนที่ส่งผลให้ h(n) ของ IIR มีความยาวไปจนถึงอนันต์ ขอให้ลองศึกษาดูลักษณะของ h(n) ของ IIR จากตัวอย่างถัดไป

<u>ตัวอย่าง 3.4</u> ตัวกรอง IIR ระบบหนึ่งมีสมการผลต่าง คือ $y(n) = 0.5y(n-1) + x(n)$ จงหา h(n) ของระบบ นี้

วิธีหนึ่งที่จะหา h(n) ได้ก็คือ ใส่สัญญาณขาเข้าเป็นสัญญาณอิมพัลส์ ซึ่งคือการแทนค่า $x(n) = \delta(n)$ แล้วหาผลตอบที่ได้จากสมการ

$$y(n) = 0.5y(n-1) + \delta(n)$$

โดยการแทนค่า n = 0, 1, 2, ... จะได้

$y(0) = 0.5y(-1) + \delta(0) = 0.5x0 + 1 = 1$

$y(1) = 0.5y(0) + \delta(1) = 0.5x1 + 0 = 0.5$

$y(2) = 0.5y(1) + \delta(2) = 0.5x0.5 + 0 = 0.25$

$y(3) = 0.5y(2) + \delta(3) = 0.5x0.25 + 0 = 0.125$

...

ผลตอบที่ได้นี้ก็จะเป็น ผลตอบสนองต่อสัญญาณอิมพัลส์นั่นเอง ซึ่งคือ

$h(n) = y(n) = [\ 1,\ 0.5,\ 0.25,\ 0.125,\ \ldots\ ]$ จะเห็นได้ว่า h(n) มีความยาวไปจนถึงอนันต์ และในกรณีนี้ h(n) มีรูปแบบแน่นอนซึ่งสามารถเขียนเป็นสมการที่เป็นฟังก์ชั่นของ n ได้เป็น

$$h(n) = \begin{cases} 0.5^n, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

อย่างไรก็ดี การใช้วิธีแทนค่า n เพื่อหา h(n) ของตัวกรอง IIR อย่างที่แสดงมานี้ ไม่สามารถทำได้ หรือทำได้ยากถ้าสมการผลต่างอยู่ในรูปแบบที่ซับซ้อนมากขึ้น วิธีทำที่ถูกต้องจะใช้การแปลง z เพื่อช่วยในการคำนวณ ซึ่งเราจะได้ศึกษาในบทที่ 4 ต่อไป

ความเป็นคอซัล (Causality)

คุณสมบัติความเป็นคอซัล เป็นคุณสมบัติที่กำหนดได้ทั้งกับสัญญาณ และระบบ ลองมองดูที่สัญญาณก่อน เรากล่าวว่า

x(n) เป็นสัญญาณคอซัล (causal signal) เมื่อ $x(n) = 0$ ที่ $n < 0$ (3.11)

นั่นก็คือ สัญญาณนี้มีค่าเฉพาะช่วงที่ n เป็นบวกเท่านั้น และเรากล่าวว่า

x(n) เป็นสัญญาณคอซัลแบบตรงข้าม (anticausal signal) เมื่อ $x(n) = 0$ ที่ $n \geq 0$ (3.12)

หรือ สัญญาณคอซัลแบบตรงข้ามจะมีค่าเฉพาะช่วงที่ n เป็นลบเท่านั้น

อาจตีความเป็นภาษาพูดง่าย ๆ ว่า สัญญาณคอซัล ก็คือ “สัญญาณที่เกิดหลังเวลา n=0” นั่นเอง สัญญาณที่มีค่าทั้งส่วนที่ n เป็นบวก และลบ เราเรียกว่า สัญญาณแบบสองด้าน (two-sided signal) และถ้าพูดถึงสัญญาณที่ไม่เป็นคอซัล (non-causal signal) ก็จะหมายถึง สัญญาณที่มีค่าที่เวลา n<0 ด้วย ซึ่งอาจเป็นสัญญาณแบบสองด้าน หรือสัญญาณคอซัลแบบตรงข้ามก็ได้

เมื่อพิจารณาในแง่ของ**ระบบ** ก็จะสามารถคิดได้ในทำนองเดียวกัน โดยจะพิจารณาความเป็นคอซัลจากผลตอบสนองต่อสัญญาณอิมพัลส์ของระบบ คือ h(n) ดังนี้

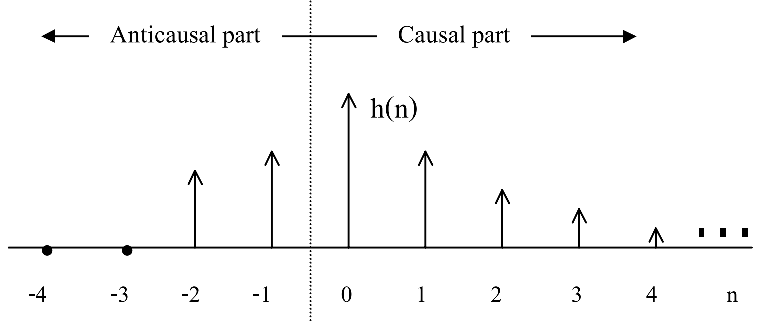
ระบบเป็นคอซัล (causal system) ก็ต่อเมื่อ $h(n) = 0$ ที่ $n < 0$ (3.13)

ระบบเป็นคอซัลแบบตรงข้าม (anticausal system) เมื่อ $h(n) = 0$ ที่ $n \geq 0$ (3.14)

ระบบที่ h(n) มีค่าทั้งฝั่ง n เป็นบวก และลบ ก็เรียกว่า ระบบแบบสองด้าน (two-sided system) และถ้าพูดถึงระบบที่ไม่เป็นคอซัล (non-causal system) ก็จะหมายถึง ระบบที่ h(n) มีค่าที่เวลา n<0 ด้วย ซึ่งอาจเป็นระบบแบบสองด้าน หรือระบบเป็นคอซัลแบบตรงข้ามก็ได้

ขออธิบายเพิ่มเติมถึงระบบที่ไม่เป็นคอซัลว่ามีลักษณะเป็นอย่างไร ในรูปที่ 3.5 เป็นตัวอย่างของผลตอบสนองต่ออิมพัลส์ของระบบที่ไม่เป็นคอซัล ซึ่งเห็นได้ว่า h(n) มีค่า h(-1) และ h(-2) ไม่เท่ากับศูนย์ ถ้าพิจารณาจากความจริงที่ว่า h(n) เป็นผลตอบ เมื่อมีสัญญาณขาเข้าเป็นสัญญาณอิมพัลส์เข้ามาที่เวลา n = 0 ก็หมายความว่า ระบบนี้ให้ผลตอบบางส่วนออกมาล่วงหน้า (ที่เวลา n = -1 และ n = -2) ก่อนที่จะมีสัญญาณขาเข้าเข้ามาในระบบเสียอีก

$$y(n) = h(-2)x(n+2) + h(-1)x(n+1) + h(0)x(n) + h(1)x(n-1) + \ldots \qquad (3.15)$$



***รูปที่ 3.5** ตัวอย่างผลตอบสนองต่อสัญญาณอิมพัลส์ ของระบบที่ไม่เป็นคอซัล*

ถ้ามองในมุมมองของสัญญาณที่เข้ามา กล่าวได้ว่า ระบบที่ไม่เป็นคอซัลสามารถสร้างผลตอบล่วงหน้าบางส่วนได้ก่อนที่สัญญาณขาเข้าจริงจะเข้ามา หรือ ถ้ามองในมุมมองของระบบ ก็กล่าวได้ว่าระบบที่ไม่เป็นคอซัลมีการนำเอาสัญญาณขาเข้าในอนาคต (สัญญาณขาเข้าล่วงหน้า) มาร่วมในการคำนวณหาผลตอบ ณ เวลาปัจจุบันด้วย คำกล่าวนี้จะเห็นได้ชัดเจนเมื่อเราเขียนสมการผลต่างของระบบออกมา ใช้รูปทั่วไปของคอนโวลูชันตามสมการที่ 3.8 เราจะสามารถเขียนสมการผลต่างของระบบนี้ได้ดังสมการที่ 3.15 ที่แสดงประกอบในรูปที่ 3.5

ขอทบทวนความหมายของเทอมต่าง ๆ สักเล็กน้อย เทอม x(n-1) คือ สัญญาณขาเข้าที่ล้าหลังไป 1 ตำแหน่ง ส่วน x(n+1) คือ สัญญาณขาเข้าที่ถูกดึงให้ล้ำหน้าไป 1 ตำแหน่ง และคิดในทำนองเดียวกันได้สำหรับ x(n-2), x(n-3), … หรือ x(n+1), x(n+2), … เมื่อพิจารณาที่ n หนึ่ง ๆ (คือ ณ ขณะเวลาหนึ่งขณะที่ระบบกำลังทำงาน) อาจกล่าวได้ว่า x(n) และ y(n) คือ สัญญาณในเวลาปัจจุบัน ส่วน x(n-1), x(n-2), … คือสัญญาณขาเข้าในอดีต และ x(n+1), x(n+2), … คือ สัญญาณขาเข้าในอนาคต ย้อนกลับไปดูสมการผลต่างของระบบนี้ จะเห็นว่า ระบบนี้ใช้สัญญาณขาเข้าในอนาคตล่วงหน้า 2 ตำแหน่ง เพื่อมาคำนวณผลตอบ ณ ขณะเวลาปัจจุบัน

ระบบที่ไม่เป็นคอซัล จากคำอธิบายข้างต้นฟังดูเหมือนเป็นไปไม่ได้ จึงมีคำถามว่า จำเป็นหรือไม่ที่เราต้องสนใจระบบเช่นนี้ คำตอบคือ จำเป็น มีงานบางอย่างที่ใช้แนวความคิดของระบบที่ไม่เป็นคอซัลในการออกแบบ และอีกทั้งระบบแบบไม่เป็นคอซัลก็สามารถปรับให้กลายเป็นระบบแบบคอซัลได้ ดังจะได้ยกตัวอย่างต่อไป

__ตัวอย่างที่ 3.5__ ตัวกรอง FIR ตัวหนึ่งมีค่า h(n) ต่อไปนี้ จงวิเคราะห์ถึงการนำระบบนี้ไปใช้

$$h(n) = \begin{cases} 1/5\ , & -2 \leq n \leq 2 \\ 0\ , & n = \text{ค่าอื่น ๆ} \end{cases} \tag{3.16}$$

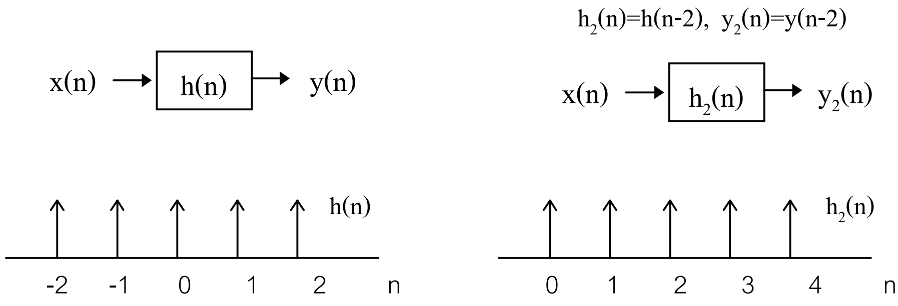
เราเขียนสมการผลต่างของระบบนี้ได้เป็น

$$y(n) = 1/5\ \{\ x(n+2)+x(n+1)+x(n)+x(n-1)+x(n-2)\ \} \tag{3.17}$$

ระบบนี้บางทีเรียกว่า ตัวกรองเกลี่ย (smoothing filter) เพราะระบบนี้จะทำการเกลี่ย (smooth) สัญญาณขาเข้า ณ จุด n ใด ๆ ให้กระจายออกรอบตัวมัน ยังผลให้ถ้ามีการเปลี่ยนแปลงที่ฉับพลันในสัญญาณขาเข้า การเปลี่ยนแปลงนั้นก็จะถูกดูดซับไปในบริเวณรอบ ๆ หรือจะบอกว่าระบบนี้เป็นตัวกรองแบบผ่านต่ำชนิดหนึ่งก็ได้

ระบบนี้สามารถนำไปใช้ได้โดยการเลื่อนสัญญาณ h(n) ให้ช้าลง 2 ตำแหน่ง เพื่อทำให้ได้ h(n) เป็นคอซัล ดังแสดงในรูปที่ 3.6 มองระบบนี้เป็นระบบใหม่ซึ่งมีผลตอบสนองต่ออิมพัลส์เป็น $h_2(n) = h(n-2)$ เราจะได้สมการผลต่างของระบบใหม่เป็น

$$y_2(n) = y(n-2) = 1/5 \{ x(n)+x(n-1)+x(n-2)+x(n-3)+x(n-4) \} \qquad (3.18)$$



ระบบเดิมเป็นแบบสองด้าน

ระบบเมื่อ h(n) ล้าหลังลง เป็นแบบคอซัล

***รูปที่ 3.6** h(n) ของตัวกรองเกลี่ย (smoothing filter)*

$y_2(n)$ เป็นสัญญาณขาออกของระบบใหม่ ซึ่งมีค่าเท่ากับ y(n) ที่ช้าลง 2 ตำแหน่ง ถ้าเรามองดูที่จุด ๆ หนึ่ง เช่น ที่ n = 3 จะได้ว่า $y_2(3)$ ในระบบใหม่นี้ ถือว่าเป็นผลตอบโดยตรงของ x(3) แต่ในมุมมองของระบบเก่า $y_2(3)$ จะเป็นผลตอบที่ออกมาล่าช้า (delay) ไป 2 ตำแหน่ง เพราะ $y_2(3)$ จริง ๆ แล้วคือ y(1) ซึ่งเป็นผลตอบโดยตรงของ x(1) และควรได้ค่าออกมาเมื่อ x(1) เข้ามาในระบบถ้าไม่มีการล่าช้า แต่ระบบกลับรอจนกระทั่ง x(3) เข้ามา จึงสามารถให้คำตอบ y(1) ออกมาได้

กล่าวโดยรวมก็คือ การใช้งานระบบที่ไม่เป็นคอซัลทำได้โดยเลื่อน h(n) ให้ล้าหลังลงจนระบบเป็นคอซัล แล้วจึงนำระบบใหม่ไปประยุกต์ใช้งานได้เหมือนปกติ ซึ่งฟังก์ชั่นการทำงานของระบบใหม่จะเหมือนระบบเก่าทุกประการ ยกเว้นเพียงแต่ ผลตอบของระบบใหม่จะถูกเลื่อนล้าหลังลงเท่ากับจำนวนตำแหน่งที่เราเลื่อน h(n) ไป

สำหรับการประมวลผลแบบเวลาจริง การที่ผลตอบของระบบล่าช้าลงไป อาจส่งผลกระทบหรือไม่ส่งผลกระทบต่อการใช้งานก็ได้ ทั้งนี้ขึ้นกับลักษณะงานว่าสามารถยอมให้มีความล่าช้าได้มากน้อยแค่ไหน ยกตัวอย่างเช่น การประมวลผลสัญญาณเสียงพูดจากโทรศัพท์ ถ้าสมมติว่าสุ่มสัญญาณเสียงด้วยอัตรา $f_s = 8$ kHz และยอมให้มีการล่าช้าของสัญญาณได้ไม่เกินครึ่งวินาที เพื่อให้ผู้สนทนาไม่สามารถรู้สึกถึงผลของมันได้ จะได้ว่าระบบนี้สามารถมีการล่าช้าของระบบได้ 4000 ขั้นเวลาโดยไม่ส่งผลกระทบใด ๆ ต่อการใช้งาน ขอให้ทำความเข้าใจให้ถูกต้องเกี่ยวกับคำว่า เวลาจริง (real-time) และคำว่าการล่าช้า (delay) สองคำนี้ไม่เกี่ยวข้องกัน การประมวลผลแบบเวลาจริงอาจมี หรือไม่มีการล่าช้าของสัญญาณก็ได้

สำหรับการประมวลผลแบบไม่เป็นเวลาจริง หรือการประมวลผลสัญญาณที่ n ไม่ได้เป็นตัวชี้สัญญาณในทางเวลา (เช่น การประมวลผลภาพนิ่ง ซึ่ง n เป็นตัวชี้ตำแหน่งของจุดในภาพ) การล่าช้าของผลตอบมักไม่ส่งผลกระทบใด ๆ และจริง ๆ แล้วในงานประเภทนี้ ถ้าเรามีสัญญาณขาเข้าทั้งหมดเก็บอยู่ในหน่วยความจำของระบบประมวลผลแล้ว ก็จะสามารถใช้งานระบบแบบไม่เป็นคอซัลได้ทันทีโดยไม่จำเป็นต้องทำให้เป็นคอซัลก่อน เพราะ เราสามารถดึงเอาสัญญาณขาเข้าล่วงหน้ามาประมวลผลได้เลยโดยไม่ต้องรอ

เสถียรภาพ (Stability)

ระบบที่มีเสถียรภาพมีเงื่อนไขว่า สัญญาณขาเข้าที่มีขอบเขตของขนาดจำกัด จะทำให้เกิดสัญญาณขาออกที่มีขอบเขตจำกัด หรือ เขียนเป็นสัญลักษณ์ว่า

ถ้า $|x(n)| < A$ โดยที่ A เป็นจำนวนจริงบวกใด ๆ ที่น้อยกว่าอนันต์ แล้ว (3.19)
จะได้ $|y(n)| < B$ โดยที่ B เป็นจำนวนจริงบวกใด ๆ ที่น้อยกว่าอนันต์

เงื่อนไขนี้ ภาษาอังกฤษ เรียกว่าเสถียรภาพแบบ bounded-input/bounded-output หรือ BIBO เงื่อนไขดังกล่าว สามารถแปลงเป็นเงื่อนไขทางคณิตศาสตร์ง่าย ๆ ที่สมมูลกัน และเป็นเงื่อนไขที่ระบุต่อสัญญาณ h(n) ได้ โดยพิจารณาจากสมการที่ 3.8 ซึ่งใช้หาผลตอบของระบบโดยการทำคอนโวลูชันคือ

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

เนื่องจาก y(n) มาจากการบวกกันของผลคูณของ x(n-k) กับ h(k) ที่ k ทุก ๆ ค่า เมื่อ x(n) เป็นสัญญาณขาเข้าที่มีขอบเขตจำกัด และไม่จำเป็นต้องเท่ากับศูนย์ เราจะได้ว่า y(n) จะมีขนาดจำกัดได้ก็ต่อเมื่อ ค่าของ h(n) ต้องลู่เข้าสู่ศูนย์ที่ n เข้าเป็นค่าอนันต์ และลบอนันต์ หรือเขียนเป็นสมการได้ว่า

$$\lim_{\substack{n \to \infty \\ n \to -\infty}} |h(n)| = 0 \qquad (3.20)$$

เงื่อนไขทั้งสองนี้สมมูลกัน กล่าวคือ จะใช้เงื่อนไขใดเป็นตัวตรวจสอบว่าระบบมีเสถียรภาพก็ได้ แต่เราจะสามารถนำเงื่อนไขที่สองไปใช้ได้ง่ายกว่า เนื่องจากเป็นเงื่อนไขที่กำหนดต่อ h(n) ซึ่งสามารถตรวจสอบได้ง่าย เช่น ในกรณีของสัญญาณคอซัล ก็ตรวจสอบเพียงว่า h(n) ลู่เข้าสู่ศูนย์หรือ

ไม่ เมื่อ n มีค่ามากขึ้น ๆ สำหรับระบบแบบ FIR จะมีเสถียรภาพเสมอ เนื่องจาก h(n) มีความยาวจำกัด ดังนั้น h(n) เป็นศูนย์แน่นอนที่ n เป็นอนันต์

ในทางปฏิบัติระบบส่วนใหญ่ที่เราใช้เป็นระบบที่เสถียร แต่ระบบที่ไม่เสถียรก็มีการนำมาใช้ประโยชน์ได้บ้าง เช่น ในวงจรกำเนิดสัญญาณ ซึ่งเป็นระบบที่ทำงานที่ภาวะคาบเส้นเสถียรภาพ ถ้าระบบเมื่อสัญญาณขาเข้าเป็นอิมพัลส์ ที่ n=0 จะให้สัญญาณขาออกเป็นสัญญาณที่ต้องการสร้าง เช่น สัญญาณซายน์ที่มีความยาวไม่จำกัดก็ได้

บทที่ 4

การแปลง z และการประยุกต์ใช้กับระบบแบบไม่ต่อเนื่อง

ในบทนี้เราจะศึกษาการแปลง z (z-transforms) และการนำมาใช้กับระบบแบบไม่ต่อเนื่อง เราจะศึกษาทั้งสัญญาณ และระบบในโดเมนของตัวแปร z ซึ่งจะพบว่าสามารถนำมาช่วยในการวิเคราะห์ระบบได้อย่างมีประสิทธิภาพ นอกจากนี้ เสถียรภาพ และความเป็นคอซัลของระบบก็สามารถนิยามได้ในโดเมน z ด้วย

การแปลง z

การแปลงแบบ z เป็นการแปลงที่กระทำกับสัญญาณไม่ต่อเนื่อง แล้วให้ผลลัพธ์เป็นฟังก์ชั่นของตัวแปรเชิงซ้อนพิเศษ คือ ตัวแปร z สมมติว่าเรามีสัญญาณ x(n) ใด ๆ ซึ่ง n มีค่าได้ตั้งแต่ ..., -3, -2, -1, 0, 1, 2, 3, ... การแปลง z ของ x(n) เขียนแทนด้วยสัญลักษณ์ X(z) มีนิยามว่า

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{4.1}$$

เมื่อแจกแจงสมการนี้อย่างตรงไปตรงมาจะได้

$$X(z) = \ldots + x(-2)z^{2} + x(-1)z + x(0) + x(1)z^{-1} + x(2)z^{-2} + \ldots \tag{4.2}$$

เขียนสัญลักษณ์ได้ว่า $X(z) = Z\{x(n)\}$ หรือ $x(n) \xrightarrow{Z} X(z)$

เห็นได้ว่า X(z) เกิดจากการคำนวณของผลบวกของสัญญาณคูณกับ z^{-n} ซึ่งบวกกันไปจนครบทุกค่าของ x(n) ดังนั้น X(z) มีโอกาสที่จะหาค่าไม่ได้ หรือมีค่าเป็นอนันต์ได้ ในที่นี้ เราจะพิจารณาตัวอย่างการหาการแปลง z ของสัญญาณที่อยู่ในรูปของฟังก์ชั่นของการยกกำลัง n ซึ่งจะพิจารณาทั้งกรณีที่สัญญาณเป็นแบบคอซัล, แบบคอซัลตรงข้าม, และแบบสองด้าน

<u>ตัวอย่างที่ 4.1</u> (กรณีสัญญาณคอซัล) หาการแปลง z ของ $x(n) = \begin{cases} a^n, & n \geq 0 \\ 0, & n < 0 \end{cases}$

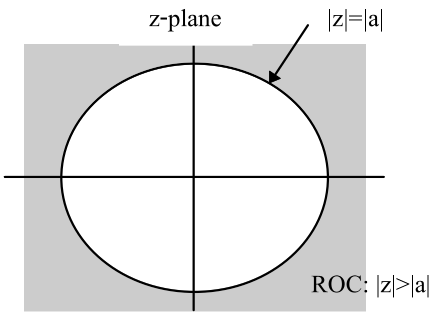
$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}$$

$$= \sum_{n=0}^{\infty} a^n z^{-n}$$

$$= \sum_{n=0}^{\infty} \left(\frac{a}{z}\right)^n$$

$$= \frac{1}{1 - a/z} \quad \text{ถ้า} \quad \left|\frac{a}{z}\right| < 1$$ (ใช้สูตรอนุกรมเลขาคณิต ในสมการที่ 4.3)

$$= \frac{z}{z - a} \quad \text{ถ้า} \; |z| > |a|$$



<u>หมายเหตุ</u> ทบทวนสูตรอนุกรมเลขาคณิตที่กล่าวว่า ถ้า $|a| < 1$ จะได้

$$1 + a + a^2 + \ldots = \frac{1}{1 - a} \tag{4.3}$$

และ
$$a + a^2 + a^3 + \ldots = \frac{a}{1 - a} \tag{4.4}$$

เราเรียกบริเวณของ z ที่หาค่า X(z) ว่า <u>บริเวณของการลู่เข้า</u> (region of convergence) หรือ <u>ROC</u> เนื่องจาก z สามารถมีค่าเป็นจำนวนเชิงซ้อนได้ ดังนั้น ROC จะแสดงได้เป็นพื้นที่ในกราฟของ z (z-plane) ซึ่งเป็นกราฟของจำนวนเชิงซ้อนมีแกนนอนเป็นส่วนจริง (real) ของ z และแกนตั้งเป็นส่วนจินตภาพ (imaginary) ของ z ดังในตัวอย่างที่ผ่านมา ROC คือ พื้นที่นอกวงกลมที่มีรัศมีเท่ากับ a หรือเขียนเป็นสมการได้ว่าคือ บริเวณที่ $|z| > |a|$

ขอนิยามรากของโพลิโนเมียลที่เป็นตัวหารในฟังก์ชั่น X(z) ว่า คือ โพล (pole) ของสัญญาณ หรือในกรณีที่ใช้ X(z) เป็นฟังก์ชั่นถ่านโอนของระบบก็เรียก รากพวกนี้ว่า คือโพลของระบบ สัญญาณหนึ่ง ๆ มีโพลได้หลายตัว และ ROC จะมีขอบเขตที่กำหนดโดยตำแหน่งของโพลเสมอ ดังเช่นในตัวอย่างที่ 4.1 นี้ X(z) มีโพลตัวเดียว คือ a และ มี ROC คือ บริเวณ $|z| > |a|$

<u>ตัวอย่างที่ 4.2</u> (สัญญาณคอซัลแบบตรงข้าม) หาการแปลง z ของ $x(n) = \begin{cases} a^n, & n < 0 \\ 0, & n \geq 0 \end{cases}$

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}$$

$$= \sum_{n=-\infty}^{-1} a^n z^{-n}$$

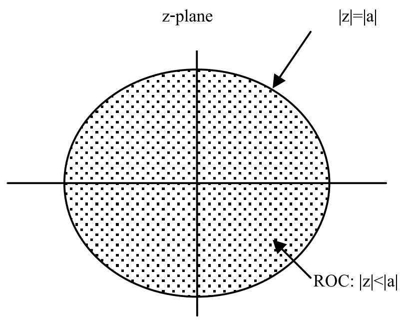
ต้องการตัวชี้ให้เป็นค่าบวก ซึ่งทำได้โดยใช้ m เป็นตัวชี้ใหม่ โดยที่ m = -n จะได้

$$X(z) = \sum_{m=1}^{\infty} a^{-m} z^m$$

$$= \sum_{m=1}^{\infty} \left(\frac{z}{a}\right)^m$$

$$= \frac{z/a}{1 - z/a} \quad \text{ถ้า } \left|\frac{z}{a}\right| < 1$$

$$= \frac{z}{a - z} \quad \text{ถ้า } |z| < |a|$$



(ใช้สูตรอนุกรมเลขาคณิต ในสมการที่ 4.4)

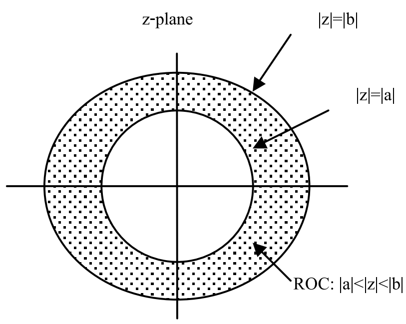
<u>ตัวอย่างที่ 4.3</u> (สัญญาณแบบสองด้าน) หาการแปลง z ของ $x(n) = \begin{cases} a^n, & n \geq 0 \\ b^n, & n < 0 \end{cases}$

$$X(z) = \sum_{n=-\infty}^{-1} b^n z^{-n} + \sum_{n=0}^{\infty} a^n z^{-n}$$

$$= \sum_{n=1}^{\infty} \left(\frac{z}{b}\right)^n + \sum_{n=0}^{\infty} \left(\frac{a}{z}\right)^n$$

ใช้ผลลัพธ์จากตัวอย่างที่ 4.1 และ 4.2 จะได้

$$X(z) = \frac{z}{b - z} + \frac{z}{z - a}$$



เนื่องจาก การแปลง z ของทั้งสองเทอม มี ROC ที่แตกต่างกัน เทอมแรกเป็นสัญญาณส่วนที่เป็นคอซัลแบบตรงข้าม มี ROC ที่ $|z| < |b|$ ส่วนเทอมที่สองเป็นสัญญาณส่วนที่เป็นคอซัล มี ROC ที่ $|z| > |a|$ ดังนั้น เมื่อรวมเงื่อนไขทั้งสอง จะได้ ROC รวมคือ ส่วนที่ทับกันของทั้งสองเงื่อนไข นั่นคือ $|a| < |z| < |b|$ ดังแสดงในรูป

เราเรียก a ว่าเป็นโพลคอซัล (causal pole) เพราะมาจากเทอมที่มาสัญญาณส่วนที่เป็นคอซัล ($n \geq 0$) และเรียก b ว่าเป็นโพลคอซัลแบบตรงข้าม (anticausal pole) ด้วยเหตุผลทำนองเดียวกัน

อาจสรุปในเรื่องความสัมพันธ์ของ ROC กับความเป็นคอซัลของสัญญาณในขั้นต้นนี้ได้ว่า สัญญาณคอซัลจะมี ROC อยู่ภายนอกวงปิดวงหนึ่ง ส่วนสัญญาณคอซัลแบบตรงข้ามจะมี ROC อยู่ภายในวงปิดวงหนึ่ง และสัญญาณที่เป็นแบบสองด้านจะมี ROC อยู่ภายในวงแหวนปิดวงหนึ่ง

จุดหนึ่งที่ต้องสังเกตก็คือ การระบุสมการ x(n) เพียงอย่างเดียวโดยที่ไม่กำหนดว่า n อยู่ในช่วงไหน (หรือไม่ระบุว่าเป็นสัญญาณคอซัลหรือไม่) จะทำให้เราหา X(z) ได้หลายคำตอบ ดังที่เราได้เห็นในตัวอย่างทั้งสามข้างต้นว่า สมการ x(n) เดียวกัน เมื่ออยู่ในรูปแบบที่เป็นคอซัลจะได้ X(z) และ ROC ที่แตกต่างจากกรณีที่เป็นคอซัลแบบตรงข้าม ดังนั้นเราจึงต้องรู้ช่วงของ n ทุกครั้งที่จะหาการแปลง z

การแปลง z โดยใช้สูตรสำเร็จจากตาราง (สำหรับสัญญาณคอซัล)

โดยทั่วไปเราไม่จำเป็นต้องทำการแปลง z โดยใช้นิยามดังที่แสดงในตัวอย่างข้างต้น สัญญาณคอซัล (มีค่าที่ $n \geq 0$) ที่อยู่ในรูปแบบที่พบเห็นบ่อย สามารถหาการแปลง z ได้โดยใช้สูตรในตารางที่ 4.1 และสัญญาณอีกหลายแบบก็สามารถใช้คุณสมบัติของการแปลง z ในตารางที่ 4.2 มาประยุกต์ใช้เพื่อจัดให้อยู่ในรูปแบบมาตรฐานเหมือนในตารางที่ 4.1

<u>ตัวอย่างที่ 4.4</u> จงหาการแปลง z ของ $x(n) = 1 - e^{-an}$ โดยที่ $n \geq 0$ และ a เป็นค่าคงที่มากกว่า 0

ใช้สูตรข้อ 2 กับ 5 ในตารางที่ 4.1 และคุณสมบัติความเป็นเชิงเส้น จะได้

$$X(z) = \frac{z}{z-1} - \frac{z}{z-e^{-a}}$$

$$= \frac{z(z-e^{-a}) - z(z-1)}{(z-1)(z-e^{-a})}$$

$$= \frac{z(1 - e^{-a})}{z^2 - z(1+e^{-a}) + e^{-a}}$$

ROC คือ $|z| > 1$ และ $|z| > e^{-a}$ แต่เนื่องจาก $a > 0$ ดังนั้น $e^{-a} < 1$ และ ROC จึงยุบรวมเหลือแต่เพียงเงื่อนไขเดียว คือ $|z| > 1$

<u>ตัวอย่างที่ 4.5</u> จงหาการแปลง z ของ $x(n) = 3a^{(n-1)},\ n \geq 1$

รูปแบบนี้ไม่ตรงกับสูตรในตารางที่ 4.1 แต่สามารถดัดแปลงได้โดยการดึงสัญญาณ x(n) ให้ขึ้นหน้ามา 1 ตำแหน่งก่อน โดย

สมมติให้ $x_1(n) = x(n+1)$ จะได้ $x_1(n) = 3a^n, n \geq 0$

เห็นได้ว่า $x_1(n)$ ตรงกับรูปแบบข้อ 5 ในตารางที่ 4.1 จะได้ว่า

$$X_1(z) = \frac{3z}{z-a} \;,\; \text{ROC: } |z| > |a|$$

จากคุณสมบัติการเลื่อนในเชิงเวลา (shift) ของข้อ 2 ในตารางที่ 4.2 จะเห็นว่า $x(n) = x_1(n-1)$ ดังนั้น เราสามารถหา $X(z)$ จาก $X_1(z)$ ได้ดังนี้

$$X(z) = z^{-1}X_1(z)$$

แทนค่า $X_1(z)$ จะได้ $$X(z) = z^{-1}\left(\frac{3z}{z-a}\right) = \frac{3}{z-a}$$

ตัวอย่างที่ 4.6 จงหาการแปลง z ของ $x(n) = \begin{cases} 2^n & , -2 \leq n \leq -1 \\ 2 & , n \geq 0 \end{cases}$

$x(n)$ เป็นสัญญาณแบบสองด้าน ถ้าเขียน $x(n)$ ใหม่ให้อยู่ในรูปทั่วไปของ n จะได้

$$x(n) = 2^{-2}\delta(n+2) + 2^{-1}\delta(n+1) + 2u(n)$$

$$x(n) = 0.25\delta(n+2) + 0.5\delta(n+1) + 2u(n)$$

โดยที่ $u(n)$ เป็นฟังก์ชั่นขั้นบันได (unit step) มีค่าเป็น 1 ที่ $n \geq 0$

เมื่อใช้ตารางที่ 4.1 ร่วมกับคุณสมบัติเชิงเส้น และคุณสมบัติการเลื่อนทางเวลาจะได้

$$X(z) = 0.25z^2 + 0.5z + \frac{2z}{z-1} \;,\; \text{ROC: } |z| > 1$$

ตัวอย่างที่ 4.7 จงหาการแปลง z ของผลตอบสนองอิมพัลส์ของ FIR ระบบหนึ่ง ซึ่งเป็นระบบแบบคอซัล โดย $h(n) = [\,-1\;\;0\;\;2\;\;0\;\;-1]$ ค่าเริ่มต้นที่เวลา n=0

สามารถเขียน $h(n)$ ในรูปแบบผลบวกของสัญญาณอิมพัลส์ได้ ดังนี้

$h(n) = -\,\delta(n) + 2\delta(n-2) - \delta(n-4)$

ดังนั้น จะได้ $H(z) = -1 + 2z^{-2} - z^{-4}$ ROC : ทุกค่าของ z

หมายเลข	สัญญาณ	การแปลง z	ROC

	$x(n),\ n \geq 0$	$X(z)$	ของ $X(z)$
1	$\delta(n)$	1	ทุกที่
2	$1 = u(n)$	$\frac{z}{z-1}$	$\|z\| > 1$
3	n	$\frac{z}{(z-1)^2}$	$\|z\| > 1$
4	n^2	$\frac{z(z+1)}{(z-1)^3}$	$\|z\| > 1$
5	α^n	$\frac{z}{z-\alpha}$	$\|z\| > \|\alpha\|$
6	$n\alpha^n$	$\frac{\alpha z}{(z-\alpha)^2}$	$\|z\| > \|\alpha\|$
7	$(-\alpha)^n$	$\frac{z}{z+\alpha}$	$\|z\| > \|\alpha\|$
8	$\cos(\alpha n)$	$\frac{z(z-\cos\alpha)}{z^2-2z\cos\alpha+1}$	$\|z\| > 1$
9	$\sin(\alpha n)$	$\frac{z\sin\alpha}{z^2-2z\cos\alpha+1}$	$\|z\| > 1$
10	$e^{-\alpha n}\sin(\alpha n)$	$\frac{ze^{-\alpha}\sin\alpha}{z^2-2e^{-\alpha}z\cos\alpha+e^{-2\alpha}}$	$\|z\| > e^{-\alpha}$
11	$e^{\alpha n}\cos(\alpha n)$	$\frac{ze^{-\alpha}(ze^{\alpha}-\cos\alpha)}{z^2-2ze^{-\alpha}\cos\alpha+e^{-2\alpha}}$	$\|z\| > e^{-\alpha}$
12	$\cosh(\alpha n)$	$\frac{z^2-z\cosh\alpha}{z^2-2z\cosh\alpha+1}$	$\|z\| > \cosh\alpha$
13	$\sinh(\alpha n)$	$\frac{z\sinh\alpha}{z^2-2z\cosh\alpha+1}$	$\|z\| > \sinh\alpha$
14	$2\|c\|\|p\|^n\cos(n\angle p+\angle c)$	$\frac{cz}{z-p}+\frac{c^*z}{z-p^*}$	

***ตารางที่ 4.1** สูตรการแปลง z ของสัญญาณมาตรฐานต่าง ๆ (เฉพาะสัญญาณคอซัล)*

คุณสมบัติ	สัญญาณ	การแปลง z
1. ความเป็นเชิงเส้น (linearity)	a {f(n)} + b {g(n)} a, b เป็นค่าคงที่	$aF(z)+bG(z)$

2. การเลื่อนทางเวลา (time shifting)	$f(n-k)$ มีเงื่อนไขเริ่มต้น = 0	$z^{-k}F(z)$
3. Differentiation	$n\,f(n)$	$-z\dfrac{dF(z)}{dz}$
4. การกลับเชิงเวลา (time reversal)	$f(-n)$	$F\left(1/z\right)$
5. การคูณด้วยค่ายกกำลัง (exponentiation)	$a^n f(n)$ a เป็นค่าคงที่	$F\left(a^{-1}z\right)$
6. การคูณทางเวลา	$f(n)\,g(n)$	$\dfrac{1}{2\pi}\int_{-\pi}^{\pi} F\left(e^{j\theta}\right)G\left(\dfrac{z}{e^{j\theta}}\right)d\theta$
7. คอนโวลูชัน (convolution)	$f(n) * g(n)$	$F(z)G(z)$
8. การแปลง z สำหรับ แก้สมการผลต่างที่มี	$f(n-m),\ m>0$	$z^{-m}\left[F(z)+\sum_{i=1}^{m} f(-i)z^{i}\right]$
ที่มีเงื่อนไขเริ่มต้น ไม่เท่ากับศูนย์	$f(n+m),\ m>0$	$z^{m}\left[F(z)-\sum_{i=0}^{m-1} f(i)z^{-i}\right]$

ตารางที่ 4.2 *คุณสมบัติที่สำคัญของการแปลง z โดยที่* $f(n) \xrightarrow{Z} F(z)$ *และ* $g(n) \xrightarrow{Z} G(z)$

การแปลง z ผกผัน (Inverse z-Transforms)

การแปลง z ผกผันใช้สำหรับแปลงสัญญาณในโดเมน z หรือ X(z) กลับเป็นสัญญาณในโดเมนเวลา หรือ x(n) เขียนสัญลักษณ์ได้ว่า $x(n) = Z^{-1}\{X(z)\}$ หรือ $X(z) \xrightarrow{Z^{-1}} x(n)$

ในเนื้อหาเบื้องต้นนี้ เราจะสนใจฟังก์ชั่นของ z ที่อยู่ในรูปแบบเศษส่วนของโพลิโนเมียลเท่านั้น และจะใช้วิธีการกระจายเป็นเศษส่วนย่อย (partial fraction expansion) และตาราง 4.1/4.2 ในการแปลง z ผกผัน สมมติว่า X(z) ที่ต้องการจะแปลงอยู่ในรูปแบบเศษส่วนของโพลิโนเมียล ดังนี้

$$X(z) = \frac{a_0 + a_1 z + a_2 z^2 \ldots + a_N z^N}{b_0 + b_1 z + b_2 z^2 \ldots + b_M z^M}$$

$$X(z) = \frac{a_0 + a_1 z + a_2 z^2 \ldots + a_N z^N}{(z-p_1)(z-p_2)\ldots(z-p_M)} \tag{4.5}$$

โดยที่ N เป็นอันดับของเศษ, M เป็นอันดับของส่วน, และ $M \geq N$ สมมติให้ $p_1, p_2, ..., p_M$ เป็นค่ารากของโพลิโนเมียลส่วน หรือเรียกว่าโพล (pole) ถ้าไม่มีโพลใดมีค่าเป็นศูนย์ และไม่มีโพลค่าซ้ำกัน ด้วยวิธีการกระจายเป็นเศษส่วนย่อย เราสามารถหา X(z) ในอยู่ในรูปของผลบวกของเศษส่วนโพลิโนเมียล ดังต่อไปนี้

$$X(z) = A_0 + \frac{A_1 z}{z - p_1} + \frac{A_2 z}{z - p_2} + \ldots + \frac{A_M z}{z - p_M} \qquad (4.6)$$

โดยที่

$$A_0 = X(z)\big|_{z=0} \qquad (4.7)$$

$$A_i = \left[\frac{z - p_i}{z} \cdot X(z)\right]_{z = p_i} \qquad (4.8)$$

ในกรณีที่มีโพลมีค่าซ้ำกัน เช่น สมมติว่า มีโพล p_k ซ้ำกันอยู่ m ค่า หรือเขียน X(z) ได้เป็น

$$X(z) = \frac{a_0 + a_1 z + a_2 z^2 \ldots + a_N z^N}{(z-p_1)(z-p_2)\ldots(z-p_k)^m \ldots (z-p_M)} \qquad (4.9)$$

เราสามารถกระจาย X(z) ให้อยู่ในรูปต่อไปนี้

$$X(z) = A_0 + \frac{A_1 z}{z - p_1} + \frac{A_2 z}{z - p_2} + \ldots + \frac{C_1 z}{z - p_k} + \frac{C_2 z}{(z - p_k)^2} + \ldots + \frac{C_m z}{(z - p_k)^m} + \ldots + \frac{A_M z}{z - p_M} \qquad (4.10)$$

โดยที่ A_0 และ A_i หาได้จากสมการข้างต้น และ C_i หาได้จาก

$$C_i = \frac{1}{(m - i)!} \cdot \frac{d^{m-i}}{dz^{m-i}}\left[\frac{(z - p_k)^m}{z} X(z)\right]_{z = p_k} \qquad (4.11)$$

<u>ตัวอย่างที่ 4.8</u> จงหาการแปลง z ผกผันของ $X(z) = \dfrac{z^2}{(z-0.5)(z-1)^2}$

ฟังก์ชั่นนี้มีโพลซ้ำกันสองตัว หรือโพลอันดับสองที่ z = 1 เราจะทำการกระจายเป็นเศษส่วนย่อย โดยสามารถจัดให้ X(z) อยู่ในรูปต่อไปนี้

$$X(z) = \frac{Az}{z-0.5} + \frac{C_1 z}{z-1} + \frac{C_2 z}{(z-1)^2} \qquad \text{———} \ (\#)$$

หา A โดย $A = \left[\frac{(z-0.5)}{z}\frac{z^2}{(z-0.5)(z-1)^2}\right]_{z=0.5} = 2$

หา C_1 โดยแทนค่า i=1, m=2, p_1=1 ในสมการที่ 4.11

$$C_1 = \frac{d}{dz}\left[\frac{(z-1)^2}{z}X(z)\right]_{z=1}$$

$$= \frac{d}{dz}\left(\frac{z}{(z-0.5)}\right)_{z=1}$$

$$= \left.\frac{(z-0.5)-z}{(z-0.5)^2}\right|_{z=1} = -2$$

หา C_2 โดยแทนค่า i=2, m=2, p_1=1 ในสมการที่ 4.11

$$C_2 = \left[\frac{(z-1)^2}{z}X(z)\right]_{z=1}$$

$$= \left[\frac{z}{z-0.5}\right]_{z=1} = 2$$

แทนค่า A, C_1, C_2 ใน (#) จะได้ $X(z) = \frac{2z}{z-0.5} - \frac{2z}{z-1} + \frac{2z}{(z-1)^2}$

เปิดตาราง 4.1 เพื่อแปลง z ผกผันของแต่ละเทอม จะได้

$$x(n) = 2(0.5)^n - 2 + 2n \quad , n \geq 0$$

การแปลง z ผกผันมีจุดที่ต้องสังเกตเช่นเดียวกับการแปลง z ก็คือ ถ้าเรากำหนด X(z) โดยที่ไม่ได้กำหนดว่าสัญญาณเป็นคอซัล หรือเป็นคอซัลแบบตรงข้าม หรือไม่ได้กำหนด ROC ของ z เราจะสามารถหาการแปลง z ผกผันได้หลายคำตอบ ดังตัวอย่างเช่น

ถ้ามี $X(z) = \frac{z}{z-a}$

กรณีที่สัญญาณเป็นคอซัล หรือ ROC คือ $|z| > a$ จะได้ว่า

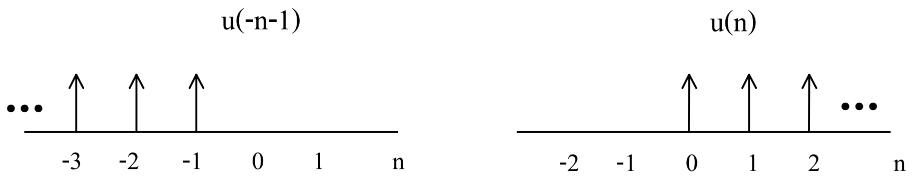
$$x(n) = a^n,\ n \geq 0 \quad \text{หรือ} \quad x(n) = a^n u(n) \qquad \text{(ตามตัวอย่างที่ 4.1)}$$

แต่ในกรณีที่สัญญาณเป็นคอซัลแบบตรงข้าม หรือ ROC คือ $z < a$ จะได้ว่า

$$x(n) = -\ a^n,\ n < 0 \quad \text{หรือ}\ x(n) = -\ a^n u(-n-1) \qquad \text{(ตามตัวอย่างที่ 4.2)}$$

ดังนั้น เราควรจะต้องรู้ ROC ทุกครั้งที่มีการระบุถึงสัญญาณ X(z) ใด ๆ อย่างไรก็ตาม เนื่องจากสัญญาณแบบคอซัลเป็นสัญญาณที่มีบทบาทในการใช้งานมากที่สุด และเราจะสนใจสัญญาณชนิดนี้แทบทั้งสิ้นในบทต่อ ๆ ไปมี ดังนั้น ถ้าหากมีการกล่าวถึง สัญญาณในโดเมน z โดยมิได้ระบุ ROC ก็ขอให้ถือว่าเป็นกรณีแบบคอซัลเสมอ

หมายเหตุ u(n) คือ ฟังก์ชั่นขั้นบันได (unit step) ทางด้าน n บวก การคูณ x(n) ด้วย u(n) เสมือนเป็นการกำหนดว่า x(n) เป็นสัญญาณคอซัล โดยไม่จำเป็นต้องระบุว่า n อยู่ในช่วงไหน และเช่นเดียวกัน u(-n-1) เป็นสัญญาณขั้นบันไดแบบกลับทางกับ u(n) ดังแสดงในรูปที่ 4.1 การคูณด้วย u (-n-1) ก็เสมือนกำหนดให้สัญญาณเป็นคอซัลแบบตรงข้าม



รูปที่ 4.1 *ฟังก์ชั่นขั้นบันไดแบบคอซัลตรงข้าม และแบบคอซัล*

การใช้การแปลง z กับระบบแบบไม่ต่อเนื่อง

เราได้ศึกษาพื้นฐานของการแปลง z ไปพอสังเขปแล้ว ในส่วนนี้จะได้นำการแปลง z ไปใช้เป็นเครื่องมือในการวิเคราะห์ระบบแบบไม่ต่อเนื่อง สมมติว่า ระบบแบบไม่ต่อเนื่องระบบหนึ่งมีผลตอบสนองต่อสัญญาณอิมพัลส์เป็น h(n) และการแปลง z ของ h(n) ได้ค่าเป็น H(z) เรากล่าวว่า H(z)

เป็นฟังก์ชั่นถ่ายโอน (transfer function) ของระบบ ซึ่งมีความสัมพันธ์กับการแปลง z ของสัญญาณขาเข้า และขาออกดังสมการ

$$H(z) = \frac{Y(z)}{X(z)} \qquad (4.12)$$

ความจริงข้อนี้สอดคล้องกับความสัมพันธ์ในเชิงเวลา ตามสมการคอนโวลูชันซึ่งใช้หาผลตอบของระบบในเชิงเวลา นั่นคือ $y(n) = h(n) * x(n)$ เพราะเมื่อใช้คุณสมบัติคอนโวลูชัน (ข้อ 7 ตาราง 4.2) ของการแปลง z กับสมการนี้จะได้ $Y(z) = H(z)X(z)$ ซึ่งคือ สมการ 4.12 นั่นเอง

เราสามารถนำสมการความสัมพันธ์ในโดเมน z นี้ ไปใช้ประโยชน์ในการคำนวณหาค่าต่าง ๆ ของระบบ ดังต่อไปนี้

1. สมการผลต่าง
2. h(n)
3. H(z)
4. y(n) เมื่อกำหนด x(n)

โดยถ้าหากทราบค่าใดค่าหนึ่งในสามข้อแรกนี้ เราจะสามารถใช้การแปลง z ในการหาค่าที่เหลืออยู่ทั้งหมดได้อย่างมีประสิทธิภาพ

ตัวอย่างที่ 4.9 ระบบหนึ่งมีสมการผลต่างดังนี้ $y(n) = 0.5y(n-1) + x(n)$ จงหาค่า h(n), H(z), และ y(n) เมื่อ $x(n) = 2u(n)$ และให้ถือว่าเงื่อนไขเริ่มต้น (initial condition) คือ y(n) มีค่าเป็น 0 ก่อนเวลา n=0

ถ้าเราทำการแปลง z กับสมการผลต่างทั้งสองข้าง จะได้

$$Y(z) = 0.5z^{-1}Y(z) + X(z)$$

$$Y(z)(1-0.5z^{-1}) = X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - 0.5z^{-1}} = \frac{z}{z - 0.5}$$

ถ้าระบบเป็นคอซัล จากตาราง 4.1 เมื่อทำการแปลง z ย้อนกลับ เราจะได้

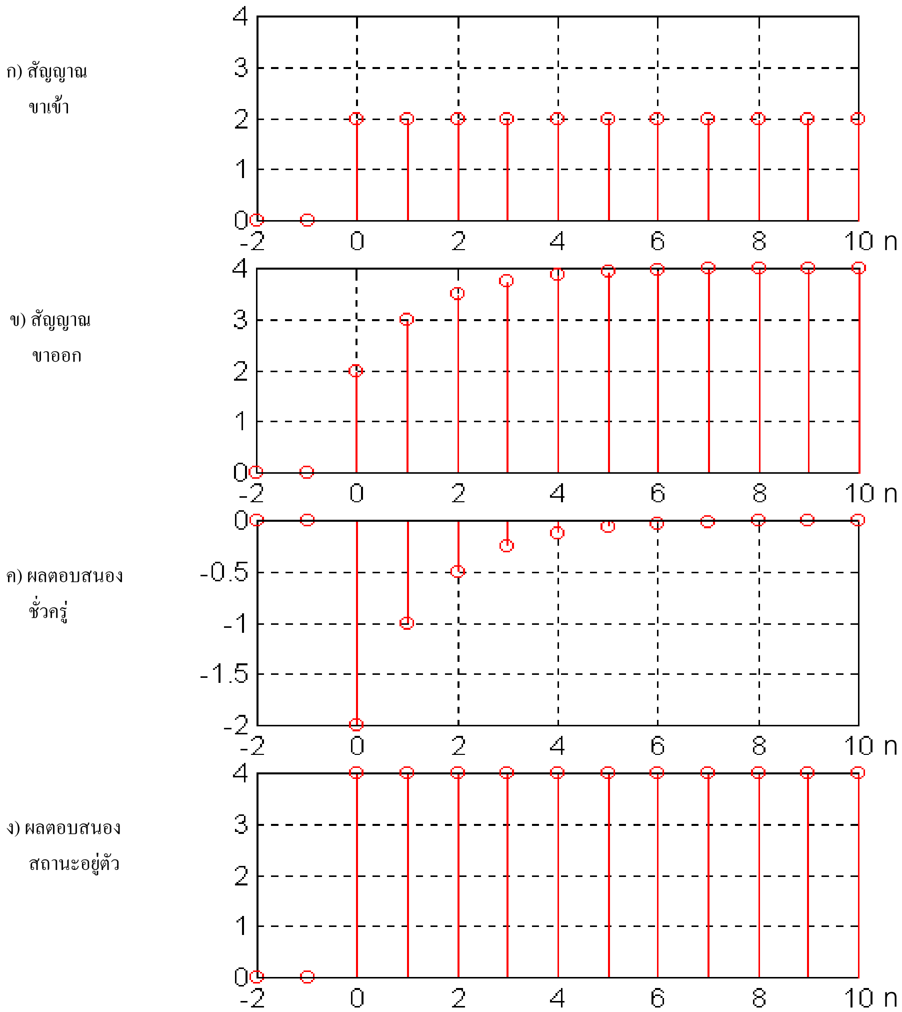
$$h(n) = 0.5^{n}u(n)$$

จะเห็นได้ว่าเราสามารถหา h(n) ได้ง่าย และสะดวกมากขึ้น ลองเปรียบเทียบกับวิธีทำโดยที่ไม่ใช้การแปลง z ในตัวอย่างที่ 3.4 ซึ่งมีสมการผลต่างที่เหมือนกัน

สำหรับการหาผลตอบของระบบ เราเริ่มจากการแปลง z ของสัญญาณขาเข้า

$$x(n) = 2u(n) \quad \text{จากตาราง 4.1 จะได้} \quad X(z) = \frac{2z}{z-1}$$

$$Y(z) = H(z)X(z) = \frac{z}{z-0.5} \cdot \frac{2z}{z-1} = \frac{2z^2}{(z-0.5)(z-1)}$$



***รูปที่ 4.2** สัญญาณขาเข้า และขาออกของตัวอย่างที่ 4.9*

ใช้วิธีกระจายเป็นเศษส่วนย่อย จะได้ผลลัพธ์ คือ

$$Y(z) = \frac{2z^2}{(z-0.5)(z-1)} = -\frac{2z}{z-0.5} + \frac{4z}{z-1}$$

เราจะสามารถหาผลตอบ y(n) ได้จากการแปลง z ผกผันของ Y(z) ดังนี้

$$y(n) = Z^{-1}\{Y(z)\} = -2\,(0.5)^n u(n) + 4u(n)$$

ผลตอบที่ได้นี้สามารถแยกแยะได้เป็นสองส่วน คือ

1. ผลตอบสนองชั่วครู่ (transient response) คือ ส่วนของผลตอบที่มีค่าเป็น 0 เมื่อ n ลู่เข้าสู่อนันต์ ในข้อนี้ ผลตอบสนองชั่วครู่ คือ $-2\,(0.5)^n u(n)$

2. ผลตอบสถานะอยู่ตัว (steady-state response) คือ ส่วนของผลตอบที่เหลืออยู่เมื่อ n ลู่เข้าสู่อนันต์ ในข้อนี้ ผลตอบสถานะอยู่ตัว คือ 4u(n)

จะเห็นได้ว่า เราสามารถหาผลตอบของระบบได้ง่ายกว่าการใช้คอนโวลูชันในเชิงเวลา การใช้การแปลง z มีประโยชน์อย่างมากในเชิงวิเคราะห์ และออกแบบระบบ อย่างไรก็ตาม ในการใช้งานเป็นตัวประมวลผลจริง ๆ ซึ่งมีสัญญาณขาเข้าเป็นสัญญาณใด ๆ ที่ไม่มีรูปแบบแน่นอน เราจะใช้สมการผลต่างในการหาสัญญาณขาออกของระบบ ดังจะได้เห็นในบทต่อ ๆ ไป

<u>**ตัวอย่าง 4.10**</u> จากตัวอย่างที่ 4.8 ซึ่งมีสมการผลต่าง คือ y(n) = 0.5y(n-1) + x(n) เราได้ทำการหาค่า H(z) ไว้แล้วคือ $H(z) = \dfrac{z}{z-0.5}$ จงหาผลตอบ y(n) เมื่อสัญญาณขาเข้าคือ x(n) = sin(πn/6)

- หาการแปลง z ของ x(n) ใช้ตารางที่ 4.1 ข้อ 9

$$X(z) = \frac{z\sin \pi/6}{z^2 - 2z\cos \pi/6 + 1}$$

<u>หมายเหตุ</u> $\sin \pi/6 = \frac{1}{2}$, $\cos \pi/6 = \frac{\sqrt{3}}{2}$

$$= \frac{z/2}{z^2 - \sqrt{3}z + 1}$$

$$= \frac{z/2}{\left(z - \frac{\sqrt{3}}{2} - \frac{j}{2}\right)\left(z - \frac{\sqrt{3}}{2} + \frac{j}{2}\right)}$$

- หา $Y(z) = H(z)X(z)$

$$= \frac{z^2/2}{(z-0.5)\left(z - \frac{\sqrt{3}}{2} - \frac{j}{2}\right)\left(z - \frac{\sqrt{3}}{2} + \frac{j}{2}\right)}$$

$$= A_0 + \frac{A_1 z}{z - 0.5} + \frac{A_2 z}{z - \frac{\sqrt{3}}{2} - \frac{j}{2}} + \frac{A_3 z}{z - \frac{\sqrt{3}}{2} + \frac{j}{2}}$$

- หา สัมประสิทธิ์ A_0, A_1, A_2, A_3 โดยสูตรของการกระจายเป็นเศษส่วนย่อย

$$A_0 = Y(0) = 0$$

$$p_1 = 0.5; \qquad A_1 = \left[\frac{z-0.5}{z} Y(z)\right]_{z=0.5}$$

$$= \left.\frac{z/2}{z^2 - \sqrt{3}z + 1}\right|_{z=0.5} \qquad = 0.65108$$

$$p_2 = \frac{\sqrt{3}}{2} + \frac{j}{2}; \quad A_2 = \left[\frac{z - \sqrt{3}/2 - j/2}{z} Y(z)\right]_{z=\sqrt{3}/2 + j/2}$$

$$= \left.\frac{z/2}{(z-0.5)\left(z - \sqrt{3}/2 + j/2\right)}\right|_{z=\sqrt{3}/2 + j/2}$$

$$= \frac{\sqrt{3} + j}{-2 + \left(2\sqrt{3} - 2\right)j} \qquad = -0.32554 - j0.73831$$

$$p_3 = \frac{\sqrt{3}}{2} - \frac{j}{2}; \quad A_3 = \left[\frac{\left(z - \sqrt{3}/2 + j/2\right)}{z} Y(z)\right]_{z=\sqrt{3}/2 - j/2}$$

$$= \left.\frac{z/2}{(z-0.5)\left(z - \sqrt{3}/2 - j/2\right)}\right|_{z=\sqrt{3}/2 - j/2}$$

$$= \frac{\sqrt{3} - j}{-2 - \left(2\sqrt{3} - 2\right)j} \qquad = -0.32554 + j0.73831$$

สังเกตว่า ถ้าค่าโพล หรือศูนย์เป็นจำนวนเชิงซ้อน จะต้องมีคู่คอนจูเกต (conjugate) ของมันอยู่ด้วยเสมอ ($p_2 = p_3*$ ในข้อนี้) และจะได้สัมประสิทธิ์ของการกระจายเศษส่วนย่อย ที่เป็นคู่คอนจูเกต

กันด้วยเสมอ ($A_2 = A_3^*$ ในข้อนี้) เช่นกัน โพล และศูนย์ที่เป็นคู่คอนจูเกตนี้จะทำให้เมื่อคูณออกมาเป็นโพลิโนเมียลแล้ว ได้ค่าสัมประสิทธิ์ของโพลิโนเมียลในฟังก์ชั่นถ่ายโอนเป็นจำนวนจริงเสมอ

- แทนค่า A_0, A_1, A_2, A_3 จะได้

$$Y(z) = \frac{0.65108z}{z - 0.5} + \frac{(-0.32554 - j0.73831)z}{z - \sqrt{3}/2 - j/2} + \frac{(-0.32554 + j0.73831)z}{z - \sqrt{3}/2 + j/2}$$

- ทำการแปลง z ผกผันโดยใช้สูตรจากตาราง 4.1 ข้อ 5 สำหรับเทอมที่ 1 และข้อ 14 เทอมที่ 2 และ 3 (เราจะต้องใช้ค่า A_2 และ p_2 ในรูปโพล่า ซึ่งในที่นี้ $A_2 = 0.8069\angle{-1.986}$ และ $p_2 = 1\angle\pi/6$) จะได้ผลลัพธ์ คือ

$$\begin{aligned} y(n) &= 0.65108(0.5)^n + 1.6138\cos\left(\pi/6\, n - 1.986\right) \\ &= \underbrace{0.65108(0.5)^n}_{\text{ผลตอบสนองชั่วครู่}} + \underbrace{1.6138\sin\left(\pi/6\, n - 0.4153\right)}_{\text{ผลตอบสนองสถานะอยู่ตัว}} \end{aligned}$$

จากตัวอย่าง 4.9 และ 4.10 ถ้าเราสังเกตดูจะเห็นว่า ผลตอบสนองสถานะอยู่ตัวของระบบจะมีรูปแบบเหมือนกับสัญญาณขาเข้าเสมอ (ถ้าดูในโดเมน z ก็จะพบว่ามีโพลเหมือนกัน) เช่น ถ้าสัญญาณขาเข้าเป็นสัญญาณขั้นบันได (ความถี่เท่ากับศูนย์) ก็จะได้ผลตอบเป็นสัญญาณขั้นบันไดเช่นเดียวกัน แต่อาจมีขนาดเปลี่ยนไป หรือ ถ้าสัญญาณขาเข้าเป็นสัญญาณซายน์ที่ความถี่หนึ่ง ก็จะได้สัญญาณขาออกเป็นสัญญาณซายน์ที่ความถี่เดียวกัน โดยอาจมีขนาด และเฟสเปลี่ยนไปเท่านั้น

สำหรับผลตอบสนองชั่วครู่ก็เช่นเดียวกัน จะมีรูปแบบเหมือนผลตอบสนองต่ออิมพัลส์ของระบบ (ถ้าดูในโดเมน z ก็จะมีโพลเหมือนกับโพลในฟังก์ชั่นถ่ายโอน) ซึ่งลักษณะเช่นนี้ของผลตอบสนองสถานะอยู่ตัว และผลตอบสนองชั่วครู่ เป็นลักษณะของระบบเชิงเส้นที่เสถียร และก็เป็นจุดที่เหมือนกันกับระบบแบบต่อเนื่องอีกจุดหนึ่ง

ความเป็นคอซัล และเสถียรภาพ

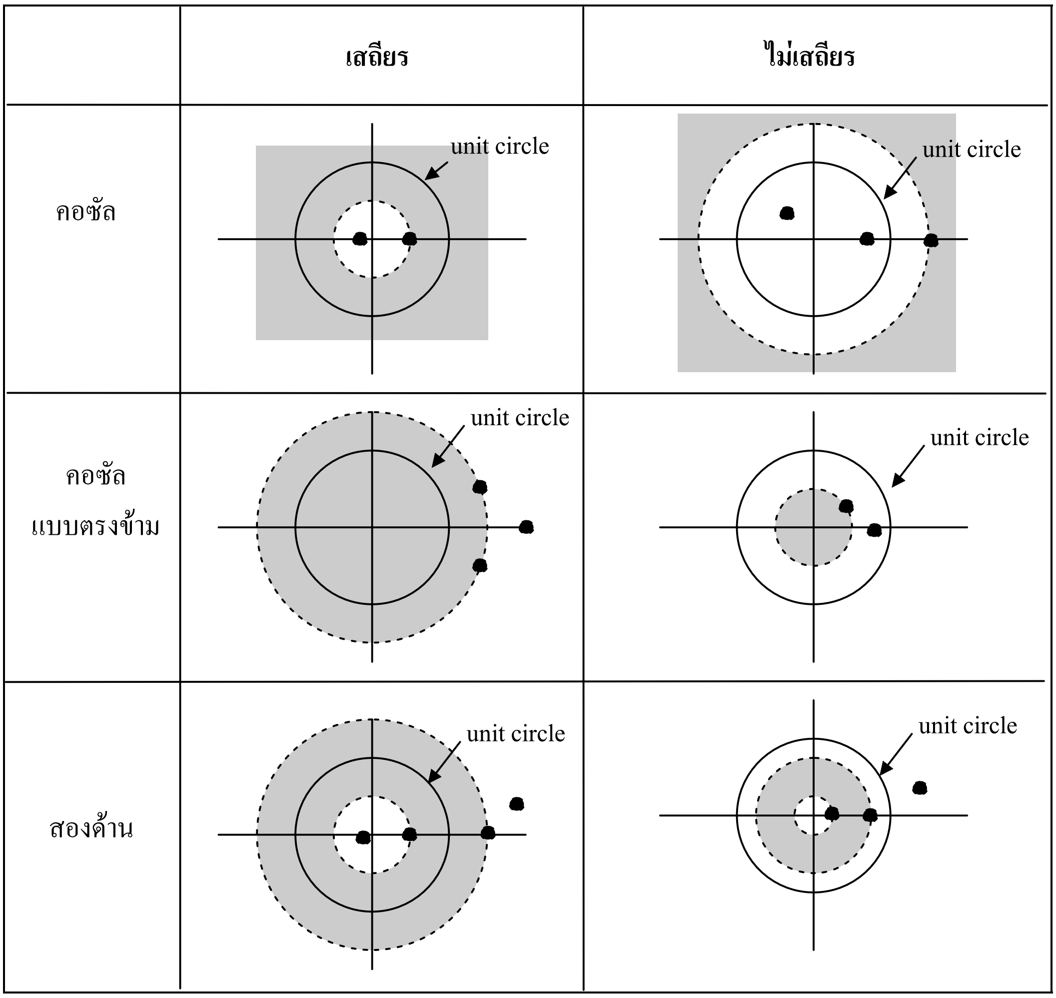
ในบทที่ 3 เราได้เรียนรู้วิธีบอกเสถียรภาพ และความเป็นคอซัลของระบบโดยดูจากผลตอบสนองต่ออิมพัลส์ ในที่นี้จะขอแนะนำวิธีที่ใช้ดูจากฟังก์ชั่นถ่ายโอน หรือ H(z) ซึ่งเราสามารถใช้ตำแหน่งของโพล และ ROC ในการบอกเสถียรภาพ และความเป็นคอซัลของระบบได้เช่นเดียวกัน

โดยอาศัยหลักบางส่วนจากตัวอย่างที่ 4.1 ถึง 4.3 เราสามารถแยกแยะกรณีที่สำคัญ ๆ สำหรับอธิบายเชิงสรุปเรื่องความเป็นคอซัล และเสถียรภาพของระบบได้ 3 กรณี ดังนี้

1. ระบบแบบ IIR ที่เป็นคอซัล จะมีลักษณะคือ
   - โพลทุกตัวของระบบเป็นโพลคอซัล
   - ROC เป็นพื้นที่ภายนอกวงกลมซึ่งมีรัศมีเท่ากับขนาดของโพลที่ใหญ่ที่สุด (เนื่องจากโพลแต่ละตัวส่งผลต่อ ROC เป็นพื้นที่ที่อยู่นอกวงกลมที่มีขนาดเท่ากับโพลนั้น ๆ ดังนั้น ผลรวมของ ROC ซึ่งเป็นส่วนที่ ROC ย่อยทั้งหมดทับกัน จึงเท่ากับ พื้นที่ที่อยู่นอกวงกลมที่มาจากโพลตัวที่ใหญ่ที่สุดนั่นเอง)
   - ระบบมีเสถียรภาพเมื่อโพลทุกตัวมีขนาดน้อยกว่าหนึ่ง (อยู่ภายใต้วงกลมหนึ่งหน่วย)
2. ระบบแบบ IIR ที่เป็นคอซัลแบบตรงข้าม จะมีลักษณะคือ
   - โพลทุกตัวของระบบเป็นโพลคอซัลตรงข้าม
   - ROC เป็นพื้นที่ภายในวงกลมซึ่งมีรัศมีเท่ากับขนาดของโพลที่เล็กที่สุด
   - ระบบมีเสถียรภาพเมื่อโพลทุกตัวมีขนาดใหญ่กว่าหนึ่ง (อยู่นอกวงกลมหนึ่งหน่วย)
3. ระบบแบบ IIR ที่เป็นแบบสองด้าน จะมีลักษณะคือ
   - โพลบางตัวของระบบเป็นโพลคอซัล และบางตัวเป็นโพลคอซัลตรงข้าม
   - ROC เป็นพื้นที่วงแหวน ที่มีขอบในเป็นวงกลมรัศมีเท่ากับขนาดของโพลคอซัลที่ใหญ่ที่สุด และมีขอบนอกเป็นวงกลมรัศมีเท่ากับขนาดของโพลคอซัลตรงข้ามที่เล็กที่สุด
   - ระบบมีเสถียรภาพเมื่อโพลคอซัลทุกตัวมีขนาดน้อยกว่าหนึ่ง และโพลคอซัลตรงข้ามทุกตัวมีขนาดใหญ่กว่าหนึ่ง
4. ระบบแบบ FIR ไม่ว่าความเป็นคอซัลจะเป็นอย่างไร จะมีลักษณะคือ
   - ไม่มีโพล
   - ROC เป็นพื้นที่ทั้งหมด (จริง ๆ แล้วถ้าเป็น FIR แบบคอซัล ROC จะไม่รวมจุด $z=0$ และถ้าเป็น FIR แบบคอซัลตรงข้ามจะไม่รวมจุด $z=\infty$ แต่พอจะอนุโลมคิดว่าเป็นพื้นที่ทั้งหมดได้โดยไม่ทำให้การวิเคราะห์ผิดไป)
   - ระบบมีเสถียรภาพเสมอ

โดยสรุป เสถียรภาพมีเงื่อนไขรวมง่าย ๆ ซึ่งสามารถใช้ได้กับทุกกรณีว่า “ระบบที่เสถียรจะต้องมี ROC ครอบคลุมวงกลมหนึ่งหน่วย (unit circle) ไว้ด้วย” ซึ่งเงื่อนไขนี้ สอดคล้องกับตำแหน่งของโพลของระบบที่เสถียรในกรณีทั้งสี่ที่ผ่านมา และดังที่สรุปในรูปที่ 4.3

เงื่อนไขของโพลที่ใช้ระบุเสถียรภาพนี้ จริง ๆ แล้วก็เป็นเงื่อนไขที่สมมูลกับเงื่อนไขของ $h(n)$ ตามสมการที่ 3.20 ยกตัวอย่างเช่น กรณีระบบคอซัลซึ่งมีเงื่อนไขเสถียรภาพว่า $\lim_{n \to \infty} |h(n)| = 0$ ถ้าพิจารณาฟังก์ชั่นถ่ายโอนซึ่งประกอบด้วยโพลหลาย ๆ ตัว จะสามารถเขียนให้อยู่ในรูปต่อไปนี้



รูปที่ 4.3 *ROC ของระบบในกรณีต่าง ๆ (สมมติว่า จุดดำในภาพคือโพลของระบบ)*
โพลที่อยู่ด้านในของวงกลม คือ โพลของส่วนเป็นคอซัล
ส่วนโพลที่อยู่ด้านนอกของวงกลม คือ โพลของส่วนที่เป็นคอซัลแบบตรงข้าม

$$H(z) = A_0 + \frac{A_1 z}{z - p_1} + \frac{A_2 z}{z - p_2} + \ . \ . \ . \ + \frac{A_M z}{z - p_M} \qquad (4.13)$$

ซึ่งเมื่อแปลง z ผกผัน จะได้ผลตอบสนองต่ออิมพัลส์เป็น

$$h(n) = A_0\delta(n) + A_1(p_1)^n + A_2(p_2)^n + \ldots + A_M(p_M)^n \quad , n \geq 0 \qquad (4.14)$$

เห็นได้ชัดว่า เมื่อ n เข้าใกล้อนันต์ ค่า h(n) จะลู่เข้าสู่ศูนย์ได้ก็ต่อเมื่อ เทอม $(p_i)^n$ ทุกตัวต้องลู่เข้าสู่ศูนย์ นั่นก็คือ โพลทุกตัวจะต้องมีขนาดน้อยกว่าหนึ่ง ข้อนี้เป็นจริงกับโพลที่เป็นจำนวนเชิงซ้อนด้วย และเป็นจริงกับโพลที่อันดับมากกว่าหนึ่งด้วย (โพลซ้ำกันมากกว่าหนึ่งตัว)

สำหรับกรณีของระบบส่วนที่เป็นคอซัลแบบตรงข้าม ก็สามารถมองได้ในทำนองเดียวกัน เพียงแต่คราวนี้ ค่า h(n) จะต้องลู่เข้าสู่ศูนย์ที่ n เข้าใกล้ลบอนันต์ ดังนั้น กรณีนี้เราจะได้ว่าโพลทุกตัวของส่วนคอซัลแบบตรงข้ามจะต้องมีขนาดมากกว่าหนึ่งแทน

ตัวอย่างที่ 4.11 $H(z) = \dfrac{z}{z-0.8} + \dfrac{z}{z-1.25}$ จงหากรณีของ ROC ที่เป็นไปได้ทั้งหมด พร้อมทั้งระบุถึงความเป็นคอซัล และเสถียรภาพของระบบ

ระบบนี้มีโพลอยู่ 2 ตัว อยู่ที่ 0.8 และ 1.25 ถ้าไม่มีการระบุ ROC ของระบบมา หรือไม่มีการระบุว่าระบบเป็นคอซัลหรือไม่ เราสามารถตีความเป็นกรณีทั่วไปได้สามกรณี คือ

1. ถ้า ROC คือ บริเวณ $|z| < 0.8$ จะได้ว่า โพลทั้งสองเป็นโพลคอซัลแบบตรงข้าม และระบบนี้เป็นคอซัลแบบตรงข้าม ระบบนี้ไม่เสถียร เพราะ ROC ไม่ทับวงกลมหนึ่งหน่วย
2. ถ้า ROC คือ บริเวณ $|z| > 1.25$ จะได้ว่า โพลทั้งสองเป็นโพลคอซัล และระบบนี้เป็นคอซัล ระบบนี้ไม่เสถียร เพราะ ROC ไม่ทับวงกลมหนึ่งหน่วย
3. ถ้า ROC คือ บริเวณ $0.8 < |z| < 1.25$ จะได้ว่าระบบนี้เป็นแบบสองด้าน จะได้ว่า 0.8 เป็นโพลของส่วนคอซัล และ 1.25 เป็นโพลของส่วนคอซัลแบบตรงข้าม ระบบนี้เสถียร เพราะ ROC ทับวงกลมหนึ่งหน่วย

ขอกล่าวย้ำอีกครั้งว่า สำหรับระบบ IIR เราจะใช้งานระบบแบบคอซัลเป็นส่วนใหญ่ ดังนั้น ในบทต่อ ๆ ไป ถ้ามีการกล่าวถึงฟังก์ชั่นถ่ายโอนโดยไม่ได้ระบุ ROC ให้ถือว่าเป็นฟังก์ชั่นของระบบคอซัล ผู้อ่านที่อาจจะยังสับสนเกี่ยวกับหลักการเรื่องความเป็นคอซัลของระบบแบบ IIR ก็ขอให้เข้าใจเฉพาะแบบที่เป็นคอซัลก็พอ ซึ่งก็จะพิจารณาเสถียรภาพได้ง่าย ๆ โดยดูเพียงว่า โพลทุกตัวของระบบมีขนาดน้อยกว่าหนึ่งหรือไม่เท่านั้น

บทที่ 5

การแปลง DTFT และผลตอบสนองเชิงความถี่

ในบทนี้เราจะศึกษาสัญญาณแบบไม่ต่อเนื่องในเชิงความถี่ ซึ่งในบทที่ 2 เราได้เห็นสัญญาณในเชิงความถี่โดยคร่าว ๆ ไปแล้วครั้งหนึ่งโดยการศึกษาจากกระบวนการสุ่มสัญญาณ ซึ่งเราได้พบว่าสัญญาณหลังการสุ่ม จะมีองค์ประกอบของความถี่ที่มีลักษณะเป็นคาบ โดยมีสำเนาสเปกตรัมของสัญญาณก่อนการสุ่ม เกิดขึ้นรอบจุดที่มีความถี่ ..., $-2f_s$, $-f_s$, 0, f_s, $2f_s$, ... เป็นศูนย์กลาง ในบทนี้เราจะศึกษาการแปลงสัญญาณเชิงความถี่นี้จากสัญญาณเชิงเวลา และนำความเข้าใจจากจุดนี้ไปเชื่อมโยงเพื่อศึกษาถึงเรื่องผลตอบสนองเชิงความถี่ของระบบ

การแปลงฟูริเยร์แบบเวลาไม่ต่อเนื่อง หรือ การแปลง DTFT (Discrete-Time Fourier Transform)

เราคงได้เคยเรียนรู้มาแล้วว่าการหาสัญญาณในเชิงความถี่จากสัญญาณในเชิงเวลาทำได้โดยการแปลงฟูริเยร์ ซึ่งถ้าสัญญาณในเชิงเวลาเราเป็น x(t) ก็จะได้สัญญาณในเชิงความถี่เป็น X(f) ดังสมการของการแปลงฟูริเยร์ คือ

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}\,dt \tag{5.1}$$

การแปลงนี้ใช้ได้กับสัญญาณแบบต่อเนื่องทั่ว ๆ ไป สำหรับสัญญาณแบบไม่ต่อเนื่อง ถ้าเรายังมองมันอยู่ในเชิงเวลา ก็จะเป็นสัญญาณที่มีค่าเป็นอิมพัลส์ ณ ตำแหน่งเวลา $t = nT$ โดยที่ $T = 1/f_s$ ก็พบว่าเราจะยังสามารถใช้รูปแบบของการแปลงฟูริเยร์ ในการกระทำกับสัญญาณนี้ได้ โดยเปลี่ยนการอินทิเกรตไปเป็นการบวกกันแทน และแทนค่า t ด้วย nT ซึ่งก็จะได้สมการของสัญญาณเชิงความถี่ในรูปนี้

$$X(f) = \sum_{n=-\infty}^{\infty} x(nT)e^{-j\omega nT} \tag{5.2}$$

x(nT) สามารถมองเป็นสัญญาณแบบไม่ต่อเนื่อง หรือลำดับข้อมูลได้ ดังนั้น สามารถใช้สัญลักษณ์แทนว่า x(n) เหมือนที่เราทำในบทที่ 2 จากนั้น ลองพิจารณาเทอม ωT ซึ่งเท่ากับ $2\pi f/f_s$ จะเห็นได้ว่าเมื่อถูกหารด้วย f_s จะทำให้ ωT เหลือหน่วยเป็นเรเดียนเท่านั้น ซึ่งเหมือนเป็นหน่วยของมุม เราจะนิยาม ตัว ωT นี้ใหม่เป็น

$$\omega' = \omega T = 2\pi f/f_s \qquad (5.3)$$

และ นิยามให้ $f' = f/f_s$ ซึ่งก็จะได้ว่า $\omega' = 2\pi f'$ (5.4)

โดยเรียก f' ว่าเป็น<u>ความถี่ดิจิตอล</u> หรือบางทีก็เรียกว่า <u>ความถี่นอร์แมลไลซ์</u> (normalized frequency) และ ω' ก็เป็นความถี่ดิจิตอลเชิงมุม โดยที่ f' ไม่มีหน่วยในทางฟิสิกส์ หรือจริง ๆ แล้วสามารถคิดได้ว่ามีหน่วยเป็น รอบต่อจุด (cycle/sample) ส่วน ω' มีหน่วยเป็นเรเดียน หรือ เรเดียนต่อจุด (radian/sample) นั่นก็เหมือนกับว่า f' ได้หมดความหมายของการเป็นความถี่จริง ๆ ในแบบแอนะลอกที่มีหน่วยเป็น Hz ไป เช่นเดียวกับในโดเมนเวลา ที่ค่า n ได้หมดความหมายของเวลาจริง ๆ ไป เมื่อเรามองเป็นสัญญาณแบบไม่ต่อเนื่อง

เราจะเขียนสมการฟูริเยร์ข้างต้นใหม่ โดยใช้ ω' ซึ่งจะได้ว่า

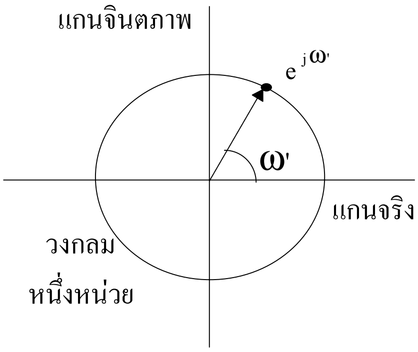
$$X(\omega') = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega' n} \qquad (5.5)$$

มักนิยมมอง X ว่าเป็นฟังก์ชั่นของ $e^{j\omega'}$ แทนที่จะเป็นฟังก์ชั่นของ ω' เฉย ๆ เพราะ ω' ในสมการนี้จะติดอยู่ในรูป $e^{j\omega'}$ เสมอ ดังนั้น เราจะเขียนเป็น $X(\omega')$ แทนว่าเป็น $X(e^{j\omega'})$ ได้ ขอให้อย่าเข้าใจผิดว่าเป็นการแทน ω' ในสมการด้วย $e^{j\omega'}$ จะได้

$$X(e^{j\omega'}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega' n} \qquad (5.6)$$

สมการนี้มีชื่อเรียกว่า การแปลงฟูริเยร์แบบเวลาไม่ต่อเนื่อง หรือ DTFT (Discrete-Time Fourier Transform) ซึ่ง $X(e^{j\omega'})$ ผลลัพธ์จากการแปลง คือ สัญญาณในเชิงความถี่ หรือสเปกตรัมของสัญญาณไม่ต่อเนื่อง

ลองพิจารณา $e^{j\omega'}$ จะพบว่า มันเป็นจำนวนเชิงซ้อนมีลักษณะเป็นคาบทุก ๆ ค่าของ ω' ที่เพิ่มขึ้นหรือลดลงเท่ากับ 2π ถ้าลองวาดภาพของ $e^{j\omega'}$ ในกราฟจำนวนเชิงซ้อน จะเห็นได้ว่าเมื่อค่า ω' เปลี่ยนไปจะได้ว่าค่าของ $e^{j\omega'}$ วิ่งอยู่บนวงกลมหนึ่งหน่วย ดังแสดงในรูปที่ 5.1

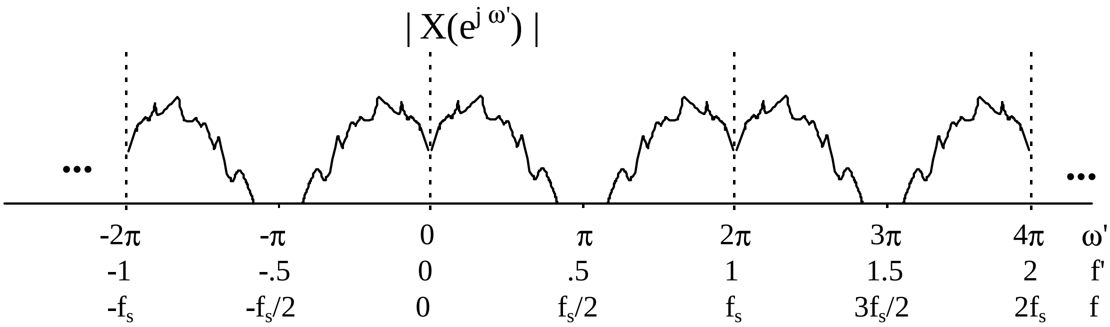


รูปที่ 5.1 *ภาพของ $e^{j\omega'}$ ในกราฟจำนวนเชิงซ้อน*

สรุปว่า $e^{j\omega'}$ มีขนาดคงที่ แต่มีมุมเปลี่ยนตาม ω' และมีลักษณะเป็นคาบทุก ๆ ช่วงของ ω' ที่เพิ่มขึ้นหรือลดลงเท่ากับ 2π ดังนั้น $X(e^{j\omega'})$ ซึ่งเป็นฟังก์ชั่นของ $e^{j\omega'}$ ก็จะต้องมีค่าเป็นคาบทุก ๆ ช่วงของ ω' ที่เพิ่มขึ้นหรือลดลงเท่ากับ 2π เช่นกัน ตัวอย่างของสัญญาณแบบไม่ต่อเนื่องในเชิงความถี่แสดงอยู่ในรูปที่ 5.2

เราจะได้ว่า สเปกตรัมของสัญญาณไม่ต่อเนื่องที่ได้จากการแปลง DTFT จะมีลักษณะดังนี้

1. เป็นฟังก์ชั่นแบบต่อเนื่องของ ω'
2. เป็นคาบ
3. มีพลังงานไม่จำกัด (เนื่องจาก รูปร่างของสเปกตรัมนี้ยาวไปจนถึงความถี่อนันต์)



รูปที่ 5.2 *ตัวอย่างสเปกตรัมของสัญญาณแบบไม่ต่อเนื่อง*
และการเทียบค่าความถี่ดิจิตอล ไปเป็นความถี่แอนะลอก

ประเด็นที่สำคัญอีกอย่างหนึ่งก็คือ การเทียบค่าจากความถี่ดิจิตอลไปเป็นความถี่แอนะลอก เนื่องจากในการวิเคราะห์สัญญาณแบบไม่ต่อเนื่องในเชิงความถี่เราจะยุ่งเกี่ยวกับ ω' และไม่จำเป็นต้องรู้เกี่ยวกับค่าความถี่แอนะลอกเลย แต่ในการมองออกไปที่สัญญาณแอนะลอก (ที่ขาเข้าก่อนการสุ่มหรือขาออกหลังจากสร้างสัญญาณคืน) เราจำเป็นต้องรู้ว่าสัญญาณที่เราจะประมวลผล ซึ่งมีความถี่ในรูปความถี่ดิจิตอล สามารถเทียบออกไปเป็นค่าความถี่แอนะลอกได้ในย่านไหน

คำตอบมีอยู่แล้วจากสมการ 5.3 ที่เราเริ่มต้นนิยาม ω' นั่นคือ $\omega' = 2\pi f/f_s$ ลองแทนค่าบางความถี่ดู เช่น

ที่ตำแหน่งความถี่ $\omega' = 2\pi$ จะตรงกับความถี่แอนะลอกที่ $f = f_s$

ที่ตำแหน่งความถี่ $\omega' = \pi$ จะตรงกับความถี่แอนะลอกที่ $f = f_s/2$

นั่นคือ ถ้าระบุ f_s มา เราสามารถเทียบค่าความถี่ดิจิตอล ไปเป็นความถี่แอนะลอกได้ และสามารถวาดรูปสัญญาณในเชิงความถี่ โดยเทียบแกนนอนเป็นความถี่ ω' หรือความถี่แอนะลอกก็ได้ ขอให้ดูการเทียบค่าความถี่ทั้งสองโดยสมบูรณ์จากรูปที่ 5.2

บางคนอาจสงสัยว่า สเปกตรัมของสัญญาณไม่ต่อเนื่องที่เห็นในรูปที่ 5.2 มีจริง หรือไม่ เพราะสเปกตรัมของสัญญาณจะมีพลังงานไปจนถึงอนันต์ได้อย่างไร คำตอบก็คล้ายกับที่ได้อธิบายไปในส่วนสัญญาณไม่ต่อเนื่องในเชิงเวลา กล่าวคือ สเปกตรัมรูปนี้ “มีจริง แต่มองไม่เห็นโดยตรง” เราไม่สามารถเห็นมันในลักษณะเดียวกับที่เราเห็นสเปกตรัมของสัญญาณแอนะลอกได้ สเปกตรัมนี้เกิดจากการแปลงฟูริเยร์มาจากสัญญาณไม่ต่อเนื่องที่ประกอบด้วยอิมพัลส์อุดมคติ ดังนั้น ตัวสเปกตรัมเองก็เป็นอุดมคติ เราสามารถหาค่ามันได้ด้วยการคำนวณการแปลง DTFT ในสมการที่ 5.6

อย่างไรก็ตาม การวิเคราะห์ทางสเปกตรัมของสัญญาณนี้มีประโยชน์ และเราสามารถแสดงให้เห็นได้ว่าสเปกตรัมนี้มีอยู่จริง เพราะในการอธิบายปรากฏการณ์ต่าง ๆ ที่เกี่ยวข้องกับความถี่ของสัญญาณไม่ต่อเนื่อง เช่น ผลตอบสนองเชิงความถี่ของระบบ, เรื่อง aliasing, และเรื่องความเพี้ยนจากสำเนาความถี่ (imaging), การเพิ่มหรือลดอัตราการสุ่ม (บทที่ 11) เป็นต้น จะต้องใช้หลักการของสเปกตรัมของสัญญาณไม่ต่อเนื่องมีลักษณะเป็นคาบ และมีพลังงานไม่จำกัดนี้มาอธิบาย

สัญญาณไม่ต่อเนื่องความถี่เดี่ยว (Discrete Sinusoidal Signal)

ตามที่ได้ทราบมาแล้วว่า สัญญาณแอนะลอกความถี่เดี่ยวมีรูปแบบสมการ คือ $x(t) = \sin(2\pi ft + \phi)$ หรือ $\sin(\omega t + \phi)$ โดยที่ f มีหน่วยเป็น รอบต่อวินาที (เฮิรตซ์) และ ω มีหน่วยเป็น เรเดียนต่อวินาที ความถี่แอนะลอกในที่นี้บ่งบอกว่า สัญญาณมีการแกว่งไปกี่รอบในหนึ่งวินาที หรือ มีการเปลี่ยนแปลงไปกี่เรเดียนในหนึ่งวินาที ส่วน ϕ คือ เฟสของสัญญาณ

สำหรับสัญญาณไม่ต่อเนื่องความถี่เดี่ยวมีรูปแบบสมการที่คล้ายคลึงกัน คือ

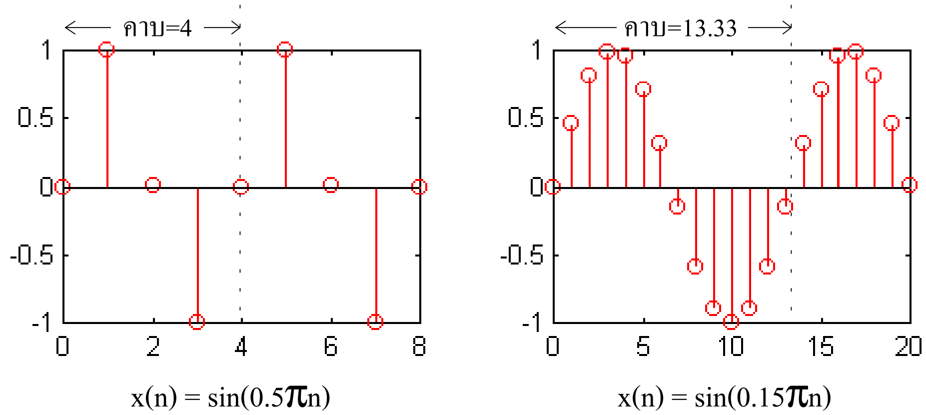
$$x(n) = \sin(2\pi f' n + \phi) \quad \text{หรือ} \quad x(n) = \sin(\omega' n + \phi) \qquad (5.7)$$

โดย f' มีหน่วยเป็นรอบต่อจุด (cycle/sample) และ ω' มีหน่วยเป็นเรเดียนต่อจุด (radian/sample) ซึ่งมีความหมายว่า ใน 1 จุด หรือ 1 ขั้นเวลาสัญญาณมีการเปลี่ยนแปลงไปกี่รอบ หรือกี่เรเดียน และคาบของสัญญาณจะนับเป็นหน่วยขั้นเวลา คือ

$$\text{คาบ} = 1/f' = 2\pi/\omega' \tag{5.8}$$

ลองดูตัวอย่างของสัญญาณความถี่เดี่ยวในรูปที่ 5.3 สัญญาณซ้ายมือมี $\omega'=0.5\pi$ เรเดียนต่อจุด ซึ่งหมายถึง แต่ละจุดของสัญญาณมีการเปลี่ยนแปลงไป 0.5π เรเดียน เราจะสังเกตได้จากรูปว่าสัญญาณ 4 จุด ทำให้ครบ 2π เรเดียน หรือ 1 รอบพอดี และสัญญาณนี้มีคาบเท่ากับ 4 ขั้นเวลา

ส่วนสัญญาณขวามือ มี $\omega'=0.15\pi$ เรเดียนต่อจุด ซึ่งหมายถึง แต่ละจุดของสัญญาณมีการเปลี่ยนแปลงไป 0.15π เรเดียน ดังนั้น 1 รอบประกอบด้วยสัญญาณเท่ากับ $2\pi/0.15\pi = 13.333$ จุด คราวนี้ปรากฏว่าคาบของสัญญาณไม่ลงตัวเป็นเลขจำนวนเต็ม ทำให้ค่าของสัญญาณในแต่ละคาบไม่ตรงกันเหมือนกับกรณีความถี่เท่ากับ 0.5π ดังแสดงในรูป อย่างไรก็ตาม สัญญาณทั้งสองนี้ถือว่าเป็นสัญญาณที่มีความถี่เดียว และความถี่คงที่



***รูปที่ 5.3** ตัวอย่างของสัญญาณไม่ต่อเนื่องความถี่เดี่ยว*

ความสัมพันธ์ของ DTFT กับการแปลง z

ลองย้อนกลับไปดูสมการของการแปลง z ของสัญญาณ x(n) ซึ่งคือ

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}$$

เมื่อพิจารณาเทียบกับสมการของการแปลง DTFT จะพบว่า การแปลง DTFT ของสัญญาณหนึ่ง ๆ ก็คือ การแปลง z ของสัญญาณนั้น เมื่อ z มีค่ารอบวงกลมหนึ่งหน่วย หรือ เขียนเป็นรูปสมการได้เป็น

$$X(e^{j\omega'}) = X(z)\Big|_{z=e^{j\omega'}} \qquad (5.9)$$

นั่นคือ ถ้าแทนค่า z ในสัญญาณที่เขียนในโดเมน z ใด ๆ ด้วย $e^{j\omega'}$ ก็จะได้ฟังก์ชันสเปกตรัมของสัญญาณนั้นทันที ความจริงนี้ทำให้เราสามารถกระทำการต่าง ๆ กับสัญญาณได้ในเชิงโดเมน z และเมื่อใดที่สนใจค่าในเชิงความถี่ของมัน (DTFT ของสัญญาณ) ก็ทำได้โดยแทนค่า $z=e^{j\omega'}$

ผลตอบสนองเชิงความถี่ของระบบ (Frequency Response)

จากสมการฟังก์ชั่นถ่ายโอนของระบบ คือ $Y(z) = H(z)X(z)$ เมื่อเราแทนค่า $z=e^{j\omega'}$ จะได้

$$Y(e^{j\omega'}) = H(e^{j\omega'})X(e^{j\omega'}) \qquad (5.10)$$

สมการนี้แสดงว่า สเปกตรัมของสัญญาณขาออกจะมีค่าเปลี่ยนไปจากสเปกตรัมของสัญญาณขาเข้าด้วยตัวคูณ $H(e^{j\omega'})$ เราจึงเรียก $H(e^{j\omega'})$ ว่าเป็น “ผลตอบสนองเชิงความถี่” (frequency response) ของระบบ ซึ่ง $H(e^{j\omega'})$ นี้แท้ที่จริงก็คือ การแปลง DTFT ของ h(n) ของระบบนั่นเอง เพราะฉะนั้น ลักษณะของ DTFT และแนวความคิดของความถี่ดิจิตอลที่เราได้ศึกษาไปแล้ว ก็จะนำมาใช้ได้กับสัญญาณ h(n) และสเปกตรัมของมัน คือ $H(e^{j\omega'})$ ได้ทุกประการ

ถ้าลองแจกแจง H ให้อยู่ในรูปผลคูณของของขนาด และเฟส ดังนี้

$$H(e^{j\omega'}) = A(e^{j\omega'})\, e^{\theta(e^{j\omega'})} \qquad (5.11)$$

รูปแบบของ $H(e^{j\omega'})$ นี้ทำให้สามารถอธิบายได้ว่า ที่ความถี่ ω' สัญญาณขาออกจะมีขนาดเปลี่ยนไปจากสัญญาณขาเข้าด้วยตัวคูณ $A(e^{j\omega'})$ และมีเฟสเปลี่ยนไปเท่ากับ $\theta(e^{j\omega'})$ จึงมีการเรียก $A(e^{j\omega'})$ ว่าเป็นผลตอบสนองทางขนาด (magnitude response) และเรียก $\theta(e^{j\omega'})$ ว่าผลตอบสนองทางเฟส (phase response)

<u>**ตัวอย่าง 5.1**</u> จากระบบตัวอย่างที่ 4.10 ซึ่งมี $H(z) = \dfrac{z}{z-0.5}$ จงใช้แนวความคิดของผลตอบสนองเชิงความถี่ในการหาค่าผลตอบสนองสถานะอยู่ตัว เมื่อสัญญาณขาเข้าคือ $x(n) = \sin(\pi n/6)$

แทน $z = e^{j\omega'}$ ลงในฟังก์ชั่นถ่ายโอน จะได้ผลตอบสนองเชิงความถี่ของระบบเป็น

$$H(e^{j\omega'}) = \frac{e^{j\omega'}}{e^{j\omega'} - 0.5}$$

สัญญาณขาเข้า คือ x(n) = sin(πn/6) ซึ่งเป็นสัญญาณความถี่เดี่ยวซึ่งมีความถี่ดิจิตอล คือ ω' = π/6 แทนค่า ω' นี้ลงไปในสมการของ H($e^{j\omega'}$) จะผลตอบสนองเชิงความถี่ที่ความถี่นี้ ดังนี้

$$\begin{aligned} H(e^{j\pi/6}) &= \frac{e^{j\pi/6}}{e^{j\pi/6} - 0.5} \\ &= \frac{\sqrt{3}/2 + j0.5}{\sqrt{3}/2 + j0.5 - 0.5} \\ &= 1.6138e^{-j\,0.4153} \end{aligned}$$

นั่นคือ ที่ความถี่ π/6 สัญญาณขาออกจะถูกขยายด้วยอัตรา 1.6138 เท่า และมีเฟสเปลี่ยนไปเท่ากับ -0.4153 หรือประมาณ -23.8 องศา ดังนั้น เราจะได้สัญญาณขาออก คือ

$$y(n) = 1.6138\sin(\pi n/6 - 0.4153)$$

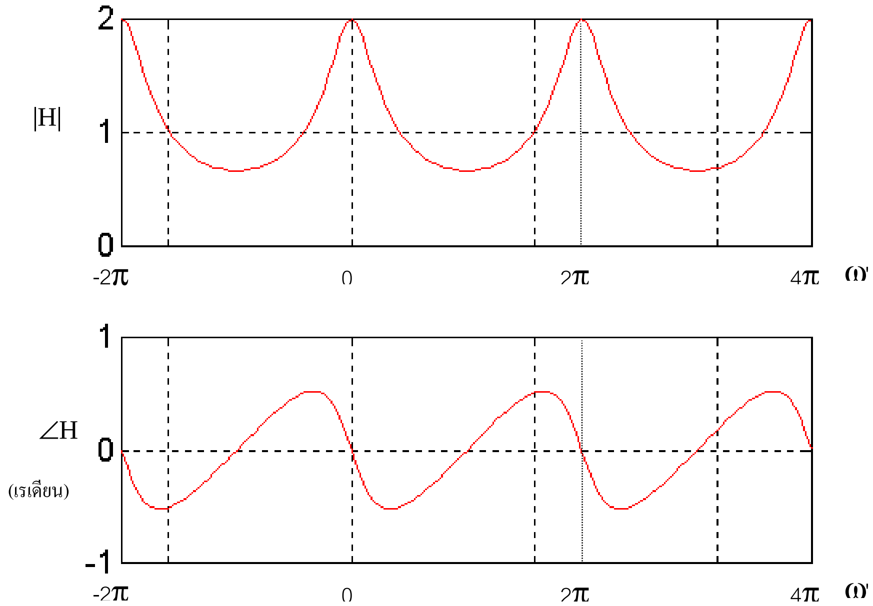
ซึ่งพบว่าคำตอบที่ได้นี้ ตรงกับคำตอบในส่วนของผลตอบสนองสถานะอยู่ตัวในตัวอย่าง 4.10 ทุกประการ ข้อนี้แสดงให้เห็นอย่างชัดเจนถึงความเหมายของผลตอบสนองเชิงความถี่ และการป้อนสัญญาณความถี่เดียวเข้าไปในระบบก็เป็นการทดสอบผลตอบสนองเชิงความถี่ที่ความถี่นั้น ๆ

ถ้าเราลองวาดรูปของผลตอบสนองเชิงความถี่ออกมาในแกน ω' จะพบว่าผลตอบสนองเชิงความถี่มีลักษณะเป็นคาบไปเรื่อย ๆ ดังในรูปที่ 5.4 ซึ่งคุณสมบัตินี้ก็ตรงกับความจริงที่เราได้ศึกษามาในเรื่อง DTFT แต่ผลตอบสนองช่วงที่เราสนใจ จะอยู่ในช่วงความถี่ตั้งแต่ 0 ถึง $f_s/2$ ซึ่งตรงกับความถี่ดิจิตอล (ω') ตั้งแต่ 0 ถึง π สาเหตุที่เราสนใจเฉพาะช่วงความถี่นี้เป็นพิเศษ เนื่องจาก

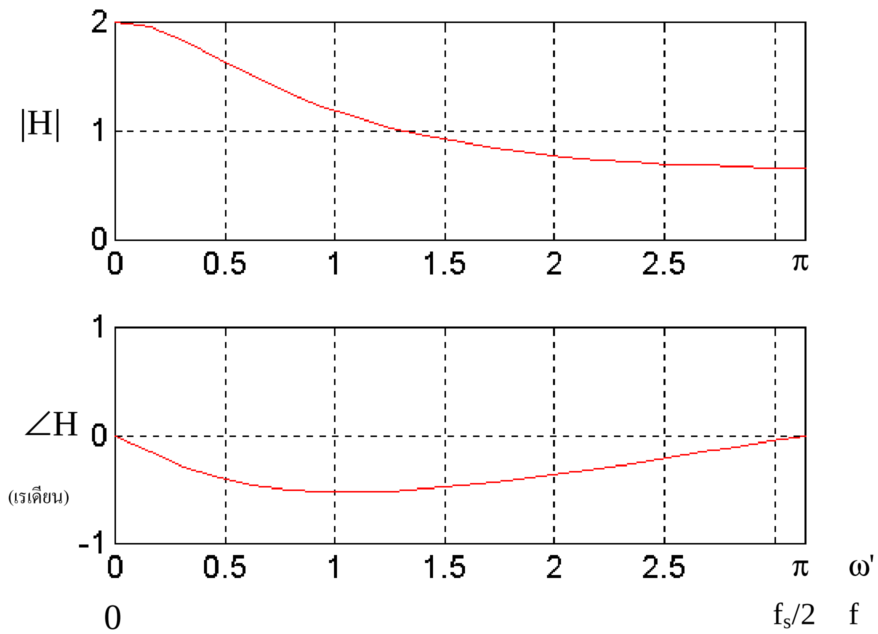
- ผลตอบสนองเชิงความถี่มีลักษณะเป็นคาบ และสมมาตรในด้านบวก/ลบ เช่นเดียวกันกับสัญญาณขาเข้าของระบบซึ่งเป็นสัญญาณไม่ต่อเนื่อง จะมีลักษณะในเชิงความถี่เป็นคาบ และสมมาตรเช่นเดียวกัน ดังนั้น ถ้ามีเหตุการณ์ใดเกิดขึ้นในช่วงความถี่ ω' = [0,π] ในช่วงความถี่อื่นก็จะเกิดเหตุการณ์เดียวกันหมด เราจึงไม่จำเป็นต้องสนใจ

- ในระบบการประมวลผลแบบไม่ต่อเนื่อง สัญญาณแอนะลอกขาเข้า (ก่อนการสุ่ม) และสัญญาณแอนะลอกขาออก จะต้องถูกจำกัดให้อยู่ในช่วง 0 ถึง $f_s/2$ เท่านั้น ดังนั้น ระบบที่เราออกแบบขึ้นก็จะเป็นตัวกรองที่ทำงานในย่านความถี่จริงตั้งแต่ 0 ถึง $f_s/2$ เท่านั้น ซึ่งก็คือ ช่วงความถี่ดิจิตอลที่ ω' = 0 ถึง π

ถ้าเราเข้าใจหลักการตรงนี้แล้ว การวาดผลตอบสนองเชิงความถี่ในช่วงที่เกิน $f_s/2$ จึงเป็นเรื่องที่ไม่จำเป็น ในบทต่อ ๆ ไป ถ้ามีการวาดผลตอบสนองเชิงความถี่ของระบบอีก จะขอแสดงเฉพาะในช่วงความถี่ตั้งแต่ 0 ถึง $f_s/2$ หรือ ความถี่ดิจิตอล ω' ตั้งแต่ 0 ถึง π เท่านั้น เราลองวาดผลตอบสนองเชิงความถี่ของระบบนี้ใหม่ ดังแสดงในรูปที่ 5.5 ซึ่งพบว่า รูปใหม่นี้ทำให้เราสามารถตีความได้ชัดเจนว่าระบบนี้ทำหน้าที่เป็นตัวกรองแบบผ่านความถี่ต่ำ



รูปที่ 5.4 *ผลตอบสนองเชิงความถี่ของระบบ* $H(z) = \dfrac{z}{z - 0.5}$



รูปที่ 5.5 *ผลตอบสนองเชิงความถี่ของ* $H(z) = \dfrac{z}{z - 0.5}$ *เฉพาะส่วนที่สนใจ*

```
function[] = freqres(a,b,fs,db)
if nargin == 3, db='n'; end
fnorm = 0:1/1000:0.5;
f = fnorm*fs; w = 2*pi*fnorm;
H = polyval(a, exp(j*w)) ./ polyval(b, exp(j*w));

if  db=='db' | db=='dB'
   plot(f,20*log10(abs(H)));
   ylabel('Magnitude Response (dB}') ;
else
   plot(f,abs(H));
   ylabel('Magnitude Response');
end
grid on
```

ถ้ามีการใส่ค่า 'db' มา ฟังก์ชั่นจะวาดค่าของ 20log(|H|) แทน

โปรแกรมที่ 5.1 *freqres.m สำหรับวาดผลตอบสนองเชิงความถี่ โดยรับค่าสัมประสิทธิ์ของฟังก์ชั่นถ่ายโอน*

```
function[] = freqres2(H,fs,db)
if nargin == 2, db='n'; end
fnorm = 0:1/1000:0.5;
f = fnorm*fs; w = 2*pi*fnorm;
z = exp(j*w);

for i=1:length(H)
    if H(i)=='*' | H(i)=='/' | H(i)=='^'
 Hnew=[Hnew,'.'];
end
Hnew=[Hnew,H(i)];
end
H=eval(Hnew);

if  db=='db' | db=='dB'
   plot(f,20*log10(abs(H)));
   ylabel('Magnitude Response (dB}');
else
   plot(f,abs(H));
   ylabel('Magnitude Response');
end
grid on
```

สร้างตัวแปร Hnew ขึ้นมาใหม่จาก H โดยเติมจุดหน้าเครื่องหมาย *, / ,และ ^

Eval ใช้คำนวณค่า Hnew ซึ่งติดอยู่ในรูปสมการ ให้กลายเป็นค่าผลลัพธ์ที่เป็นตัวเลข

โปรแกรมที่ 5.2 *freqres2.m สำหรับวาดผลตอบสนองเชิงความถี่ โดยรับค่าสมการของฟังก์ชั่นถ่ายโอน*

โปรแกรมที่ 5.1 และ 5.2 แสดงการใช้ Matlab เพื่อวาดผลตอบสนองเชิงความถี่จากฟังก์ชั่นถ่ายโอน โดยโปรแกรมที่ 5.1 รับค่าเวคเตอร์ที่เป็นสัมประสิทธิ์ของเศษ และส่วน เช่น สำหรับฟังก์ชั่นถ่ายโอนดังในตัวอย่างที่ 5.1 คือ $H(z) = \dfrac{z}{z - 0.5}$ เวคเตอร์ของเศษ คือ [1 0] หรือ 1 เฉย ๆ ก็ได้ และเวคเตอร์ของส่วน คือ [1 -0.5] ฟังก์ชั่นนี้เรียกใช้ฟังก์ชั่นภายในของ Matlab ชื่อ polyval ซึ่งใช้คำนวณค่าของโพลิโนเมียลเศษและส่วน ถ้าสมมติว่าใช้ f_s=1 เราสามารถเรียกใช้ฟังก์ชั่นนี้ได้ ดังนี้

```
>> freqres(1, [1, -0.5], 1)
```

ฟังก์ชั่น freqres นี้ ถ้าให้ a เป็นเวคเตอร์ของสัญญาณไม่ต่อเนื่องใด ๆ และ b=1 ก็จะกลายเป็นการคำนวณการแปลง DTFT ของสัญญาณนั้น ๆ แล้ววาดสเปกตรัมทางขนาดออกมาในช่วงความถี่ 0 ถึง π

สำหรับโปรแกรมที่ 5.2 จะรับค่าสมการของฟังก์ชั่นถ่ายโอนเป็นลักษณะของข้อความ (ต้องอยู่ในเครื่องหมายคำพูด) แล้วใช้คำสั่ง eval เพื่อแปลงข้อความเป็นค่า เช่น สำหรับฟังก์ชั่นถ่ายโอน $H(z) = \dfrac{z}{z - 0.5}$ เราสามารถเรียกใช้ฟังก์ชั่นนี้ได้ ดังนี้

```
>> freqres2('z/(z-0.5)', 1)
```

ฟังก์ชั่นทั้งสองสามารถวาดผลตอบสนองเชิงความถี่ในหน่วยของ dB ได้ โดยใส่พารามิเตอร์ 'db' เพิ่มให้กับฟังก์ชั่น เช่น freqres(1, [1, -0.5], 1, 'db') เป็นต้น

ตัวกรองที่ทำขึ้นโดยการประมวลผลสัญญาณดิจิตอลนี้ ถูกเรียกว่า ตัวกรองดิจิตอล ซึ่งจะเห็นได้ว่าการประมวลผลที่ทำในภาคสัญญาณแบบไม่ต่อเนื่อง จะมีผลถึงการเปลี่ยนแปลงทางความถี่ของสัญญาณแอนะลอกได้ เช่นเดียวกับการใช้ตัวกรองแอนะลอก ในบทที่ 7 และ 8 เราจะได้ศึกษาต่อไปว่า จะหาสัมประสิทธิ์ของระบบเพื่อให้ระบบทำหน้าที่เป็นตัวกรองแบบต่าง ๆ ได้อย่างไร

จะเห็นได้ว่าช่วงความถี่ที่ตัวกรองแบบดิจิตอลทำงาน จะขึ้นอยู่กับค่าความถี่ในการสุ่ม (f_s) และถ้า f_s สูงหรือต่ำไปก็จะมีผลไม่ดีต่อระบบ การเลือก f_s สูงเกินไป เช่น ถ้าความถี่ย่านที่เราสนใจอยู่ในช่วงประมาณ 0-100Hz แต่เราสุ่มสัญญาณด้วยความถี่ 10kHz นั่นคือ ความถี่ 0-100Hz จะครอบคลุมบริเวณแค่ 1 ใน 50 หรือ 2% ของช่วงความถี่ที่ตัวกรองนี้ทำงาน ($f_s/2$= 5kHz) นอกจากจะสิ้นเปลืองเพราะต้องใช้ A/D converter และ โปรเซสเซอร์ที่ทำงานได้เร็วแล้ว ยังทำให้ได้ผลตอบสนองเชิงความถี่ที่ไม่ดีเท่าที่ควรในย่านที่สนใจอีกด้วย

หรือถ้าเราสุ่มด้วยความถี่ที่ต่ำเกินไป เช่น f_s = 220 Hz นั่นคือ ความถี่ 0-100 Hz จะครอบคลุมบริเวณถึงประมาณ 90% ของความถี่ที่ตัวกรองนี้ทำงาน ($f_s/2$= 110Hz) ก็อาจทำให้มีผลของความผิดเพี้ยนจาก aliasing ในการแปลงแอนะลอกเป็นดิจิตอล และความผิดเพี้ยนจากสำเนาความถี่ในการแปลงดิจิตอลเป็นแอนะลอกได้

<u>หมายเหตุ</u> หนังสือหลายเล่มใช้สัญลักษณ์ ω แทนความถี่ดิจิตอล ขอให้ระวังด้วยในการอ่านอ้างอิงเพิ่มเติมจากหนังสือเล่มอื่น

บทที่ 6

การแปลง DFT และ FFT

ในบทนี้จะพิจารณาการแปลงที่สำคัญมากในวิชาประมวลผลสัญญาณ นั่นคือ การแปลง FFT โดยจะได้เริ่มต้นจากการทบทวนการแปลงแบบต่าง ๆ ที่เกี่ยวข้องกับสัญญาณในเชิงความถี่และในเชิงเวลา แล้วจึงเข้าไปในรายละเอียดของ DFT และ FFT รวมถึงการประยุกต์ใช้งาน FFT

ทบทวนการแปลงแบบต่าง ๆ

การแปลงระหว่างสัญญาณในเชิงความถี่ และเชิงเวลามีอยู่หลายแบบ ซึ่งทุกแบบก็มีวัตถุประสงค์เดียวกัน คือ ต้องการแปลงระหว่างสัญญาณในเชิงเวลา กับสัญญาณในเชิงความถี่ (หรือสเปกตรัม) จุดที่แตกต่างกันของการแปลงแต่ละแบบก็คือ คุณลักษณะของสัญญาณที่จะแปลงเท่านั้น เช่น การแปลงฟูริเยร์จะใช้กับสัญญาณในเชิงเวลาที่มีความต่อเนื่อง และมีพลังงานจำกัด ในขณะที่อนุกรมฟูริเยร์ใช้กับในเชิงเวลาที่มีความต่อเนื่อง เป็นคาบ และมีพลังงานไม่จำกัด ขอให้ดูสรุปจากตารางที่ 6.1 และ 6.2

การแปลงฟูริเยร์ Fourier Transform (FT) เชิงเวลา ⟵⟶ เชิงความถี่ ต่อเนื่อง ต่อเนื่อง	อนุกรมฟูริเยร์ Fourier Series (FS) เชิงเวลา ⟵⟶ เชิงความถี่ ต่อเนื่องและเป็นคาบ ไม่ต่อเนื่อง
การแปลงฟูริเยร์แบบเวลาไม่ต่อเนื่อง Discrete Time Fourier Transform (DTFT) เชิงเวลา ⟵⟶ เชิงความถี่ ไม่ต่อเนื่อง ต่อเนื่องและเป็นคาบ	การแปลงฟูริเยร์แบบไม่ต่อเนื่อง Discrete Fourier Series (DFS) หรือ Discrete Fourier Transform (DFT) เชิงเวลา ⟵⟶ เชิงความถี่ ไม่ต่อเนื่องและเป็นคาบ ไม่ต่อเนื่องและเป็นคาบ

ถ้าไม่ระบุว่าเป็นคาบหมายถึงสัญญาณที่ไม่เป็นคาบ และมีพลังงานจำกัด (ที่อนันต์มีค่าเป็นศูนย์)

ตารางที่ 6.1 *การแปลงแบบต่าง ๆ กับคุณลักษณะของสัญญาณที่เกี่ยวข้อง*

ตารางที่ 6.1 และ 6.2 ให้ไว้สำหรับอ้างอิง และให้สังเกตถึงความเหมือน และแตกต่างของกันแต่ละแบบ เราจะไม่กล่าวถึงในรายละเอียดของการแปลงแต่ละแบบ เพราะไม่ใช่ประเด็นสำคัญของวิชานี้ ยกเว้นเรื่อง DFT ซึ่งจะเป็นส่วนสำคัญที่จะต้องใช้

การแปลง	จากเชิงเวลาไปเป็นความถี่	จากเชิงความถี่ไปเป็นเวลา (การแปลงผกผัน)
การแปลงฟูริเยร์ (FT)	$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-j\omega t} dt$	$x(n) = \int_{-\infty}^{\infty} X(f)\, e^{j\omega t} df$
การแปลงฟูริเยร์แบบเวลาต่อเนื่อง (DTFT) มีคาบในเชิงความถี่ = 2π	$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n}$	$x(n) = \int_{-\pi}^{\pi} X(e^{j\omega})\, e^{j\omega t} \frac{d\omega}{2\pi}$
อนุกรมฟูริเยร์ (FS) มีคาบในเชิงเวลา = T	$c(k) = \frac{1}{T} \int_{0}^{T} x(t)\, e^{-jk\omega_0 t} dt$ โดยที่ c(k) เป็นสัมประสิทธิ์ของความถี่ $k\omega_0$	$x(t) = \sum_{k=-\infty}^{\infty} c(k)\, e^{jk\omega_0 t}$
อนุกรมฟูริเยร์แบบไม่ต่อเนื่อง (DFS) หรือ การแปลงฟูริเยร์แบบไม่ต่อเนื่อง (DFT) มีคาบในเชิงเวลา = คาบเชิงเวลา = N samples	$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N}$	$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, e^{j2\pi kn/N}$

ตารางที่ 6.2 *สรุปสมการของการแปลงแบบต่าง ๆ*

การแปลง DFT (Discrete Fourier Transform)

จากตารางที่ 6.1 จะเห็นได้ว่า มีเพียงการแปลง DFT เท่านั้นที่มีทั้งสัญญาณในเชิงเวลา และในเชิงความถี่เป็นแบบไม่ต่อเนื่อง จุดนี้เป็นจุดที่สำคัญมาก เพราะมันบ่งบอกว่า เราสามารถจะกระทำการแปลงนี้ได้โดยใช้การคำนวณ (การคูณ และบวก) ทางดิจิตอลได้ ซึ่งสามารถประยุกต์ได้สะดวกมากในคอมพิวเตอร์ หรือในฮาร์ดแวร์โดยตรงก็ได้ การแปลงแบบอื่นมีสัญญาณแบบต่อเนื่องเกี่ยวข้องด้วย ซึ่งทำให้การแปลงต้องใช้วิธีอินทิเกรตซึ่งยุ่งยากกว่ามาก

DFT มีความเหมือนกันกับ DFS มาก ทั้ง DFT กับ DFS มีสมการในการแปลงเหมือนกัน จุดที่ต่างกันก็คือที่มาและความหมายของทั้งสอง DFS คือ อนุกรมฟูริเยร์แบบไม่ต่อเนื่อง (Fourier Series) ใช้ในกรณีที่สัญญาณเชิงเวลาเป็นแบบไม่ต่อเนื่อง และเป็นคาบ ซึ่งก็จะได้ว่าสัญญาณในเชิงความถี่จะเป็นแบบไม่ต่อเนื่อง และเป็นคาบเช่นเดียวกัน ส่วน DFT เป็นการนำเอาความจริงที่เกิดขึ้นจาก DFS มาใช้ นั่นคือ

1. สัญญาณทั้งในเชิงเวลา และความถี่เป็นแบบไม่ต่อเนื่อง

2. สัญญาณเป็นรายคาบทั้งในเชิงเวลาและความถี่ สามารถทำการแปลงโดยการใช้ค่าที่เกิดขึ้นใน 1 คาบเท่านั้น (สังเกตว่าในสูตรจะเป็นการหาผลรวมของสัญญาณในตำแหน่งที่ 0 ถึง N-1 เท่านั้น) ดังนั้น จำนวนค่าที่นำมาคำนวณในการแปลงไป และแปลงผกผันจึงมีความจำกัด

DFT ก็คือ DFS ที่เราสนใจเพียงคาบเดียวเท่านั้น ก็คือ เราสนใจว่าสัญญาณในเชิงเวลาเป็นสัญญาณไม่ต่อเนื่องมีความยาวจำกัดเท่ากับ N และเป็นสัญญาณที่มีรูปร่างใด ๆ ก็ได้ เมื่อทำการแปลง DFT แล้ว ก็จะได้สัญญาณในเชิงความถี่เป็นเป็นสัญญาณไม่ต่อเนื่อง และมีความยาวจำกัดเท่ากับ N เท่ากัน

สมมติให้ x(n) เป็นสัญญาณในเชิงเวลา และ X(k) เป็นสัญญาณในเชิงความถี่ที่เกิดจาก DFT โดย k แทนตัวชี้ลำดับของสัญญาณทางด้านความถี่ ทั้งสองสัญญาณมีความยาวเท่ากัน คือ N เราจะเขียนสัญลักษณ์ได้ว่า

$$x(n) \overset{\text{DFT; N}}{\longleftrightarrow} X(k)$$

จากสูตรในตารางที่ 6.2 เราจะได้ว่า x(n) และ X(k) มีความสัมพันธ์กันดังนี้

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N} \qquad (6.1)$$

เพื่อจัดรูปสมการให้ง่ายขึ้น ขอนิยามให้ $W_N = e^{-j2\pi/N}$ เป็นค่าที่ขึ้นกับ N เท่านั้น สำหรับในการแปลงครั้งหนึ่ง ๆ N จะมีค่าคงที่ ดังนั้น W_N จึงเสมือนเป็นค่าคงที่ เราสามารถเขียนการแปลง DFT ได้ใหม่เป็น

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn} \qquad (6.2)$$

หรือเขียนในรูปเมตริกซ์ได้เป็น

$$\begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} W_N^0 & W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^{1x1} & W_N^{1x2} & \cdots & W_N^{1x(N-1)} \\ W_N^0 & W_N^{2x1} & W_N^{2x2} & \cdots & W_N^{2x(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ W_N^0 & W_N^{(N-1)x1} & W_N^{(N-1)x2} & \cdots & W_N^{(N-1)x(N-1)} \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} \qquad (6.3)$$

```
function X = dft(x)
N = length(x);
c = j*2*pi/N;
for k=0:N-1
   X(k+1) = sum(x.*exp(-c*k*[0:N-1]));
end
```

โปรแกรมที่ 6.1 *dft.m สำหรับคำนวณการแปลง DFT*

ที่มา และความหมายของการแปลง DFT

ลองดูว่า X(k) ที่ได้จากการแปลง DFT นี้มีความหมายอย่างไร ขอให้ดูจากรูปที่ 6.1 พร้อมคำอธิบายดังต่อไปนี้

ก) ให้ $x_a(t)$ เป็นสัญญาณแบบต่อเนื่องใด ๆ ที่มีความยาวจำกัด (พลังงานจำกัด) และมีสเปกตรัม (ซึ่งหามาได้จากการแปลงฟูริเยร์) คือ $X_a(f)$ สมมติว่าได้ $X_a(f)$ มีความถี่จำกัด โดยมีความถี่สูงสุดอยู่ที่ f_{max}

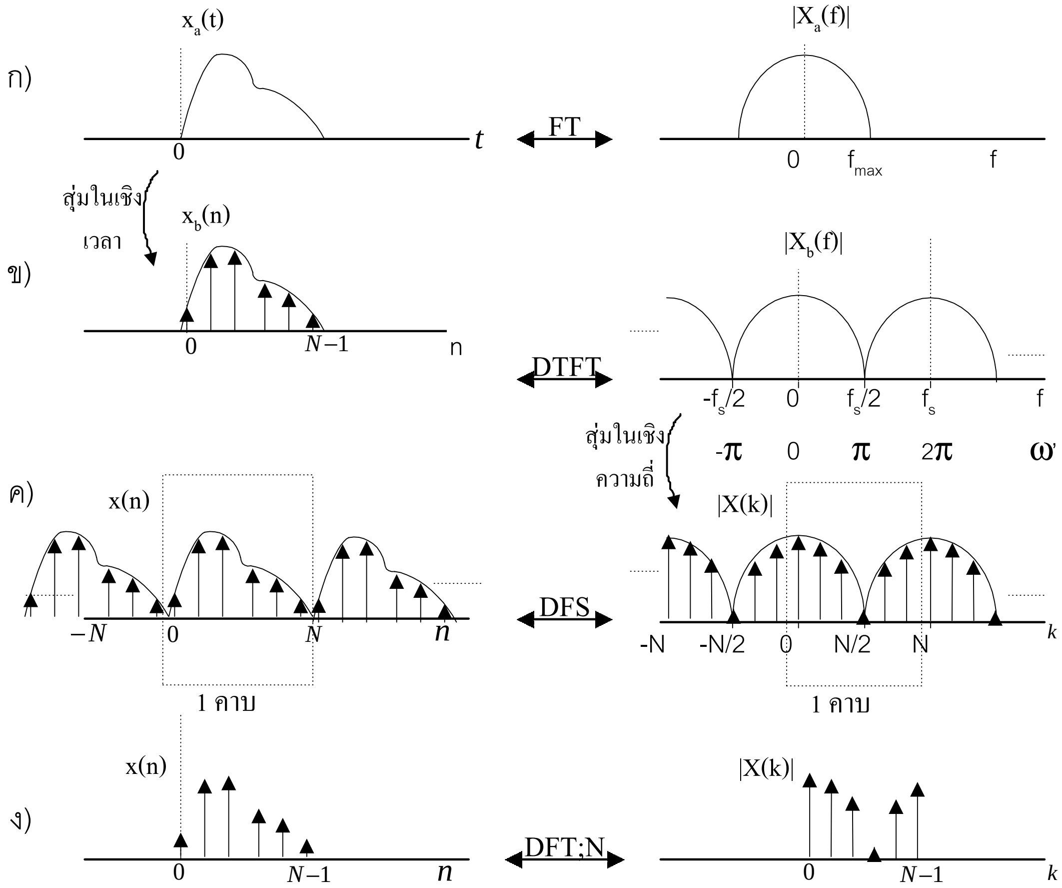
ข) ถ้าเราสุ่มสัญญาณ $x_a(t)$ ด้วยอัตรา f_s โดยที่ $f_s > 2f_{max}$ จะได้สัญญาณไม่ต่อเนื่อง $x_b(n)$ ซึ่งมีสเปกตรัม คือ $X_b(e^{j\omega'})$ มีลักษณะเป็นรายคาบดังที่เราได้ศึกษามาแล้วในเรื่อง DTFT โดยที่สัญญาณในช่วงความถี่ดิจิตอล $\omega' = -\pi$ ถึง π จะเหมือนกับสเปกตรัมของสัญญาณแอนะลอกเดิม คือ $X_a(f)$ ในช่วง $-f_s/2$ ถึง $f_s/2$ ทุกประการ เพราะไม่เกิด aliasing ในการสุ่มครั้งนี้

ค) ถ้าเราสุ่มสัญญาณในเชิงความถี่ $X_b(e^{j\omega'})$ ด้วยความถี่ N ตัวอย่างต่อ 1 คาบ หรือมีคาบในการสุ่มเท่ากับ $2\pi/N$ เราจะได้สัญญาณในเชิงความถี่เป็นแบบไม่ต่อเนื่อง และเป็นรายคาบ ให้สัญญาณใหม่นี้เป็น X(k) สิ่งที่เกิดขึ้นในเชิงเวลาก็คือ สัญญาณในเชิงเวลาจะเกิดเป็นรายคาบขึ้นดังรูป (เช่นเดียวกับที่เราสุ่มสัญญาณในเชิงเวลา แล้วเกิดสำเนาของสัญญาณขึ้นในเชิงความถี่ การสุ่มสัญญาณในเชิงความถี่ ก็จะทำให้เกิดสำเนาของสัญญาณขึ้นในเชิงเวลา)

ง) ถ้าเราดึงเอาเฉพาะสัญญาณในช่วง 1 คาบออกมาออกมาทั้งในเชิงเวลา และความถี่ คือในช่วง n และ k เท่ากับ 0 ถึง N-1 ส่วนนี้ก็คือ การแปลง DFS ในคาบเวลาเดียว หรือ ก็คือ การแปลง DFT นั่นเอง

จะสังเกตได้ว่า คาบหนึ่ง ๆ ของสัญญาณ x(n) คือ สัญญาณที่มาจากการสุ่มของสัญญาณ $x_a(t)$ และคาบหนึ่ง ๆ ของสัญญาณ X(k) ก็คือ สัญญาณมาจากการสุ่มสัญญาณ $X_a(f)$

นั่นก็คือ DFT สามารถใช้หาสัญญาณในเชิงความถี่แทนการแปลงฟูริเยร์ได้อย่างสมบูรณ์ โดยให้สเปกตรัมเป็นสัญญาณไม่ต่อเนื่องซึ่งจะมีรูปร่างเหมือนสเปกตรัมจริงของสัญญาณ โดยมีเงื่อนไขว่าสัญญาณที่ต้องการหาค่าในเชิงความถี่ต้องมีพลังงานจำกัด และความถี่จำกัด คือ ไม่เกิน f_{max} ซึ่งความถี่ที่ใช้ในการสุ่มเพื่อแปลงสัญญาณเป็นดิจิตอลต้องมีค่ามากกว่า $2f_{max}$



รูปที่ 6.1 *ความเกี่ยวโยงกันระหว่างการแปลง DFT กับการแปลงฟูริเยร์*

ถ้าเงื่อนไขดังกล่าวเป็นจริง เราสามารถสรุปเป็นคำพูดได้ว่า “การแปลง DFT ให้ผลลัพธ์เป็นสัญญาณไม่ต่อเนื่องซึ่งมีค่าเท่ากับเป็นการสุ่มสเปกตรัมที่ได้จากการแปลง DTFT” หรือ หมายถึงว่าค่าทุกค่าที่ได้จากการแปลง DFT จะอยู่บนเส้นของผลลัพธ์ที่ได้จากการแปลง DTFT เสมอ

การค้นพบนี้เป็นสิ่งที่สำคัญมาก เพราะทำให้เราสามารถแปลงสัญญาณกลับไปกลับมาระหว่างเชิงเวลา กับเชิงความถี่ได้ โดยกระทำกับสัญญาณแบบไม่ต่อเนื่องล้วน ๆ ดังนั้น การแปลง DFT ก็เป็นการประมวลผลแบบดิจิตอลอย่างหนึ่งเพื่อหาสเปกตรัมของสัญญาณ

ข้อสังเกตอีกอันหนึ่งก็คือ ผลตอบที่ได้จาก DFT ถ้าเทียบกับผลที่ได้จาก DTFT จะอยู่ในช่วง 0 ถึง 2π หรือคือความถี่จริง ๆ ที่ 0 ถึง f_s ซึ่งผลตอบนี้เมื่อพับที่จุดกึ่งกลางจะสมมาตรกัน เราสนใจผลตอบในช่วงครึ่งแรกเท่านั้น ซึ่งคือความถี่จริงที่ 0 ถึง $f_s/2$ ซึ่งก็คือช่วง

k = 0 ถึง N/2 ในกรณีที่ N เป็นเลขคู่

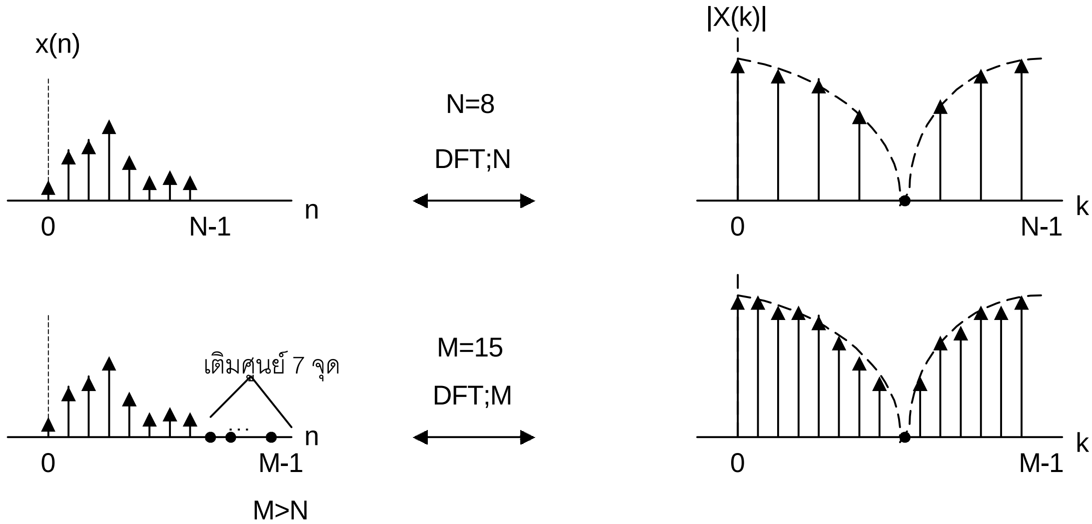
และ k = 0 ถึง (N+1)/2 ในกรณีที่ N เป็นเลขคี่

```
function  dftspec(x,fs)
if nargin < 2, fs=1; end;          %ถ้าไม่ใส่ค่า fs มา จะให้ fs = 1
N = length(x);
N_plot = (N+1)/2;                  %กำหนดจำนวนจุดที่จะวาดเป็นครึ่งหนึ่งของทั้งหมด
f = [0:N-1]*fs/N;                  %ใช้ fs เป็นตัวปรับสเกลของแกนนอนเพื่อให้แสดงใน
X = dft(x);                          หน่วยของความถี่แอนะลอก
plot(f(1:N_plot),abs(X(1:N_plot)));
grid on
```

โปรแกรมที่ 6.2 *dftspec.m สำหรับวาดสเปกตรัมทางขนาดของสัญญาณ โดยใช้ DFT*

การเติมศูนย์ (zero padding)

“การเติมศูนย์” เป็นการเติมจุดที่มีค่าเป็นศูนย์ต่อท้ายเข้าไปในสัญญาณ x(n) ก่อนที่จะทำ การแปลง DFT ซึ่งจะส่งผลให้สเปกตรัมที่ได้มีจำนวนจุดมากขึ้น ซึ่งเสมอเป็นการสุ่มสเปกตรัมด้วยจำนวนจุดที่มากขึ้น การเติมศูนย์ช่วยให้มองเห็นรูปร่างได้ละเอียด และชัดเจนขึ้น แต่ในทางทฤษฎีแล้ว ไม่ได้เป็นการเพิ่มข้อมูลใด ๆ ให้แก่สัญญาณเลย เส้นประที่แสดงในรูปที่ 6.2 คือ เส้นที่แสดงผลลัพพ์ที่เกิดจากการแปลง DFT ซึ่งไม่ว่าเราจะเติมศูนย์เข้าไปมากเท่าไรเส้นนี้ก็จะคงเดิม กล่าวคือ ผลลัพธ์ที่เราได้ดีที่สุดจากการเติมศูนย์มากขึ้น ๆ ก็คือผลที่เข้าใกล้ผลของการแปลง DTFT นั่นเอง



รูปที่ 6.2 *ผลของการเติมศูนย์กับการแปลง DFT*

สเปกตรัมของพลังงาน กับสเปกตรัมของกำลัง

ผลลัพธ์ที่เกิดจากการแปลง DTFT หรือ DFT ที่ได้กล่าวถึงมานี้ เรียกว่า สเปกตรัมทางขนาด (หรือสเปกตรัมของโวลท์เทจ) ถ้าเรามีสมมติฐานว่าสัญญาณที่นำมาหาสเปกตรัมนี้เกิดขึ้น และสิ้นสุดภายในช่วงที่นำมาคิดเท่านั้น หรือที่เวลาอื่น ๆ สัญญาณมีค่า สัญญาณประเภทนี้เรียกว่า สัญญาณที่มีพลังงานจำกัด (finite-energy signal) สัญญาณที่มีพลังงานจำกัดมีสเปกตรัมที่เราสนใจ คือ สเปกตรัมของพลังงาน (energy spectrum) ซึ่งสามารถหาได้จาก

$$S_x(k) = |X(k)|^2 \qquad (6.4)$$

โดย $S_x(k)$ คือ สเปกตรัมของพลังงานของสัญญาณ $x(n)$ และ $X(k)$ คือ ผลการแปลง DFT ของสัญญาณ $x(n)$ สำหรับการแปลง DTFT ก็มีสูตรในลักษณะเดียวกัน คือจะได้ $S_x(\omega') = |X(e^{j\omega'})|$ ผู้สนใจสามารถดูวิธีพิสูจน์ได้ในหนังสืออ้างอิง [3]

แต่ถ้าเรามีสมมติฐานว่า สัญญาณที่นำมาหาสเปกตรัมนี้เป็นเพียงส่วนย่อยหนึ่งของสัญญาณที่มีคุณลักษณะทางสถิติไม่แปรตามเวลา (stationary signal) สัญญาณหนึ่ง กล่าวคือ สัญญาณนี้ยาวไปจนถึงเวลาเป็นอนันต์ แต่เราตัดเอาเพียงส่วนหนึ่งของมันมาดูเท่านั้น ซึ่งสัญญาณนี้อาจเป็นสัญญาณที่มีรูปร่างไม่แน่นอน (random signal) แต่ขอให้มีคุณลักษณะทางสถิติที่คงที่ เช่น มีค่าเฉลี่ยคงที่ และมีกำลังคงที่ (เรื่องของสัญญาณแรนดอม มีรายละเอียดมาก และเป็นสิ่งที่สำคัญมากในการประมวลผลสัญญาณขั้นสูง ซึ่งเกินขอบเขตในขั้นพื้นฐานที่จะอธิบายในหนังสือเล่มนี้)

สัญญาณประเภทนี้เรียกว่า สัญญาณที่มีพลังงานไม่จำกัด ซึ่งไม่สามารถหาค่าพลังงานได้ แต่สามารถหาค่า “กำลังเฉลี่ย” ของสัญญาณได้ ดังนั้น สเปกตรัมที่เราสนใจของสัญญาณประเภทนี้ คือ สเปกตรัมของกำลัง (power spectrum) ซึ่งมีสูตร คือ (เช่นเดียวกัน ดูพิสูจน์ใน [3])

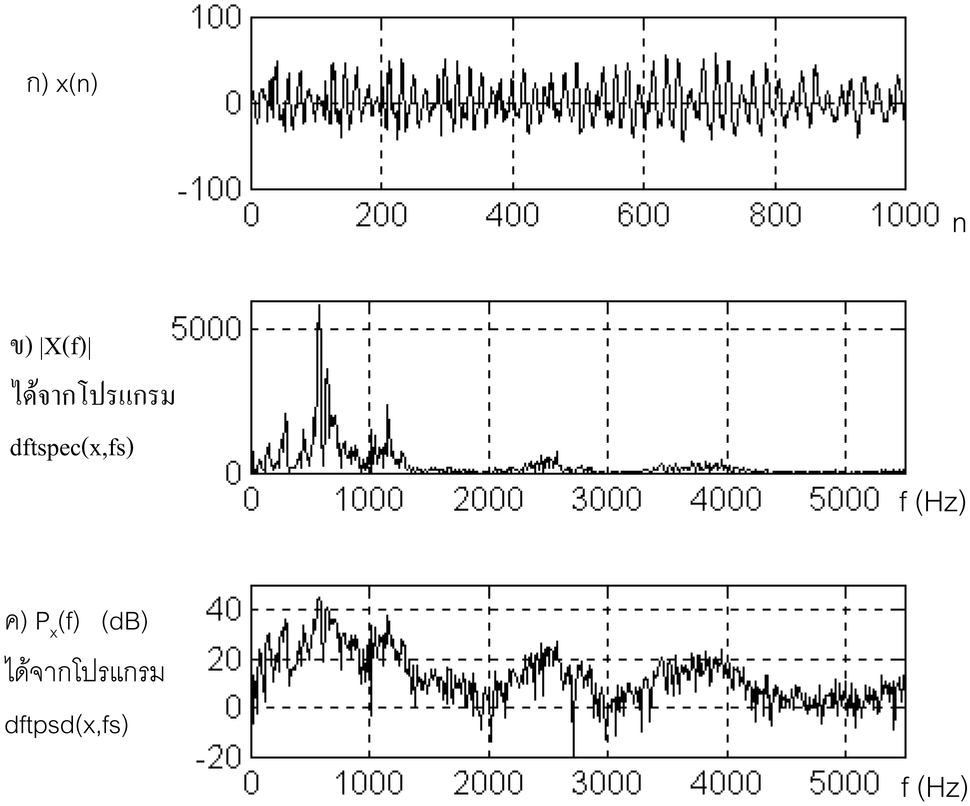
$$P_x(k) = \frac{1}{N}|X(k)|^2 \qquad (6.5)$$

สเปกตรัมของกำลังเป็นตัวบอกว่า สัญญาณมีกำลัง (หรือพลังงาน) กระจายอยู่ในความถี่ต่าง ๆ อย่างไร ถ้าเราบวกค่าทุกค่าของ $S_x(k)$ เข้าด้วยกัน ก็จะได้กำลังรวมของสัญญาณในทุก ๆ ความถี่ ซึ่งก็ควรจะเท่ากับกำลังเฉลี่ยของสัญญาณนั่นเอง ซึ่งความจริงข้อนี้มีกล่าวในทฤษฎีบทของ Parseval ที่ว่า “กำลังเฉลี่ยในทางเวลา จะเท่ากับกำลังเฉลี่ยในทางความถี่” ซึ่งเขียนเป็นสูตรสำหรับสัญญาณไม่ต่อเนื่อง ได้ว่า

$$\sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2 \qquad (6.6)$$

มักนิยมแสดง สเปกตรัมของกำลังในหน่วย dB ซึ่งมีสูตรว่า

$$P_x(k) = 10 \log\left(\frac{1}{N} |X(k)|^2 \right) \text{ (dB)} \qquad (6.7)$$



รูปที่ 6.3 *ตัวอย่างของสัญญาณในเชิงเวลา และสเปกตรัมที่ได้จากการใช้ DFT*

```
function  dftpsd(x,fs)
if nargin < 2, fs=1; end;
N = length(x);
N_plot = (N+1)/2;
f = [0:N-1]*fs/N;
X = dft(x);
plot(f(1:N_plot),10*log10(abs(X(1:N_plot)).^2/N));
grid on
```

โปรแกรมที่ 6.3 *dftpsd.m สำหรับวาดสเปกตรัมของกำลังในหน่วย dB โดยใช้ DFT*

รูปที่ 6.3 แสดงตัวอย่างของสเปกตรัมทางขนาด และสเปกตรัมกำลังที่ได้จากการใช้ DFT สัญญาณ x(n) ที่ใช้เป็นสัญญาณเสียงที่อ่านเข้ามาใน Matlab เป็นเวคเตอร์แบบแถวยาว 1000 จุด มีอัตราการสุ่มเท่ากับ 11025 Hz (ดูวิธีอ่านสัญญาณเสียงใด ๆ เข้ามาใน Matlab ได้ในภาคผนวก ก) สเปกตรัมที่ได้ในรูป ข และ ค ที่จริงแล้วเป็นสัญญาณแบบไม่ต่อเนื่อง แต่เนื่องจากมีจำนวนจุดถึง 1000 จุด เมื่อวาดออกมาแล้วแต่ละจุดจึงติด ๆ กันมองเหมือนเป็นสัญญาณที่ต่อเนื่อง แต่ละจุดในสเปกตรัมมีระยะห่างกันเท่ากับ $f_s/N = 11025/1000 = 11.025$ Hz

สังเกตว่าสเปกตรัมของกำลังที่แสดงในหน่วย dB จะสามารถแสดงให้เห็นถึงในรายละเอียดของสเปกตรัมได้ดีกว่าโดยเฉพาะในย่านที่มีกำลังของสัญญาณต่ำ ๆ และโดยทั่วไปแล้ว สเปกตรัมของกำลังเป็นค่าที่นิยมหามากกว่าสเปกตรัมของขนาด ถึงแม้สัญญาณจะไม่เป็นแบบ stationary ก็ตาม ดังจะได้กล่าวถึงตัวอย่างในส่วนหลังในเรื่องเครื่องวิเคราะห์สเปกตรัม

การแปลง FFT (Fast Fourier Transform)

เนื่องจาก การแปลง DFT มีประโยชน์ในการใช้งานมาก จึงมีได้มีความพยายามคิดค้นหาวิธีที่จะคำนวณ DFT ให้เร็วขึ้น และมีประสิทธิภาพขึ้นกว่าปกติ การแปลง FFT ก็คือชื่อที่ใช้เรียก “วิธีการคำนวณ DFT อย่างรวดเร็ว” กว่าการคิดปกตินั่นเอง เพราะฉะนั้น เมื่อกล่าวถึงการแปลง FFT โดยหลักการแล้วขอให้นึกถึงว่ามันคือ การแปลง DFT นั่นเอง และการแปลง FFT ไม่ใช่การแปลงชนิดใหม่แต่อย่างใด

การคำนวณ DFT โดยตรงจากนิยาม ถ้าสัญญาณมีความยาวเท่ากับ N จะต้องใช้การคำนวณถึงประมาณ N^2 CMACs (CMAC คือ Complex Multiplication and Accumulation, เป็นหน่วยวัดการคำนวณ ซึ่ง 1 CMAC เท่ากับการกระทำทางคณิตศาสตร์ที่ประกอบด้วยการคูณเลขเชิงซ้อน 2 จำนวนเสร็จแล้วนำเอาผลลัพธ์ที่ได้ไปบวกสมทบเข้ากับเลขเชิงซ้อนอีกจำนวนหนึ่ง) ซึ่งมีค่าที่มาก โดยเฉพาะเมื่อ N มีค่าสูง ๆ การใช้ FFT จะช่วยลงจำนวน CMAC ที่ต้องใช้ลงได้มาก

ในปัจจุบันได้มีผู้คิดค้นการคำนวณ DFT อย่างรวดเร็วได้หลายวิธี คำว่า FFT เป็นชื่อ กลาง ๆ ที่ไม่ได้บ่งบอกว่าเป็นวิธีไหน ในบทนี้เราจะศึกษาวิธีทำ FFT วิธีพื้นฐานวิธีหนึ่ง คือวิธี <u>radix-2 แบบ decimation-in-time</u> (แตกเป็นส่วนย่อยทางฝั่งเวลา)

เราลองย้อนกลับไปดูการแปลง DFT ในสมการที่ 6.2 คือ

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn} \qquad \text{โดยที่} \quad W_N = e^{-j2\pi/N} \text{ และ } k = 0, 1, \ldots N-1$$

ถ้าให้ N เป็นเลขคู่ เราสามารถกระจาย X(k) ให้อยู่ในรูปของผลบวกของเทอมที่ n เป็นคู่และเทอมที่ n เป็นคี่ได้ ดังนี้

$$X(k)=\underbrace{\sum_{n=0}^{\frac{N}{2}-1}x(2n)W_N^{2nk}}_{\underline{\text{เทอมคู่}}}+\underbrace{\sum_{n=0}^{\frac{N}{2}-1}x(2n+1)W_N^{(2N+1)k}}_{\underline{\text{เทอมคี่}}} \qquad (6.8)$$

มาจาก x(0),x(2),x(4),… มาจาก x(1),x(3),x(5),…

$$X(k)=\sum_{n=0}^{\frac{N}{2}-1}x(2n)W_N^{2nk}+\sum_{n=0}^{\frac{N}{2}-1}x(2n+1)W_N^{2nk}W_N^{k} \qquad (6.9)$$

ถ้าพิจารณาเทอม W_N^{ab} ที่มี a และ b เป็นจำนวนใด ๆ ที่ไม่เท่ากับ 0 จะพบว่า เราสามารถย้ายตัวยกกำลังของ W ไปเป็นตัวหารของ N ได้ดังนี้

$$W_N^{ab}=e^{-j\frac{2\pi}{N}ab}=e^{-j\frac{2\pi}{N/b}a}=W_{N/b}^{a} \qquad (6.10)$$

เราใช้ความจริงข้อนี้ แทนค่าเทอม W_N^{2nk} ด้วย $W_{N/2}^{nk}$ ในสมการที่ 6.9 จะได้

$$X(k)=\underbrace{\sum_{n=0}^{\frac{N}{2}-1}x(2n)W_{N/2}^{nk}}_{\text{DFT }\frac{N}{2}\text{ จุด}}+\underbrace{\sum_{n=0}^{\frac{N}{2}-1}x(2n+1)W_{N/2}^{nk}}_{\text{DFT }\frac{N}{2}\text{ จุด}}\times\underbrace{W_N^{k}}_{\substack{\text{สัมประสิทธิ์พิเศษ}\\\text{ใช้คูณเทอมคี่}}} \qquad (6.11)$$

จะเห็นได้ว่า X(k) ได้กลายเป็นผลบวกของสองเทอม แต่ละเทอมเป็นรูปแบบของการคำนวณ DFT N/2 จุด โดยเทอมแรกกระทำกับสัญญาณ x(0), x(2), …, x(N-2) และเทอมที่สองกระทำกับสัญญาณ x(1), x(3), …, x(N-1)

ถ้าเรายุติการแตกกระจายเป็นเทอมย่อยแต่เพียงเท่านี้ และคำนวณ DFT โดยใช้สมการที่ 6.11 จะได้ว่า เราต้องคำนวน DFT N/2 จุด เป็นจำนวน 2 ชุด ซึ่งแต่ละชุดจะต้องใช้จำนวน CMAC ในการคำนวณเท่ากับ $(N/2)^2$ ดังนั้น ต้องใช้จำนวน CMAC ในการคำนวณทั้งสิ้นประมาณ $2\left(\frac{N}{2}\right)^2=\frac{N^2}{2}$ (จริง ๆ แล้ว ต้องใช้การบวกอีก N จุด เพื่อนำผลลัพธ์ของแต่ละชุดมาบวกกัน แต่เนื่องจาก ถ้า N ใหญ่พอประมาณ N จะมีค่าน้อยเมื่อเทียบกับ $\frac{N^2}{2}$ จึงประมาณว่าไม่ต้องคิดการบวก N ครั้งนี้ได้)

สรุปว่า การหา W(k) ซึ่งเป็น DFT N จุด สามารถกระจายให้อยู่ในเทอมของ DFT N/2 จุด ซึ่งจะทำให้จำนวน CMAC ที่ต้องใช้ลดลงประมาณครึ่งหนึ่ง เช่นเดียวกัน ถ้าเราทำการแตกเทอม DFT

N/2 จุดที่อยู่ในสมการที่ 6.11 นี้ต่อไป แต่ละเทอมก็จะสามารถกระจายให้กลายเป็นผลบวกของ DFT N/4 จุดสองเทอม ซึ่งก็จะทำให้จำนวน CMAC ที่ต้องใช้ลดลงอีกประมาณครึ่งหนึ่ง เราสามารถกระจายเช่นนี้ไปเรื่อย ๆ จนกระทั่งทุกตัวอยู่ในรูปของ DFT 2 จุด ซึ่ง DFT 2 จุดสามารถคำนวณได้ง่าย ๆ ดังนี้

สมมติ x(n) ยาว 2 จุด จะได้ $$X(k) = \sum_{n=0}^{1} x(n)\, W_2^{kn} \qquad (6.12)$$

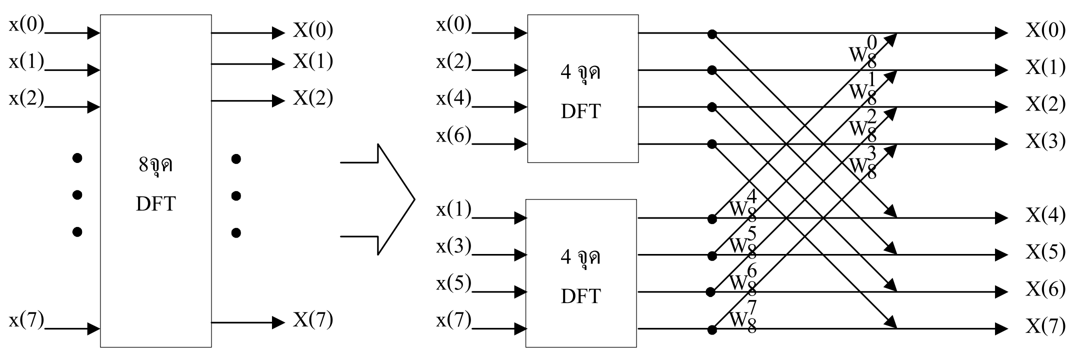
อาศัยความจริงว่า $W_2^0 = 1$ และ $W_2^1 = e^{-j\pi} = -1$ จะได้ว่า

$$\left.\begin{aligned} X(0) &= x(0) + x(1) \\ X(1) &= x(0) - x(1) \end{aligned}\right\} \qquad (6.13)$$

ขั้นตอนที่ได้อธิบายมาทั้งหมดนี้รวมเรียกว่า การแปลง FFT เรามักเขียนวิธีคำนวณ FFT โดยใช้แผนภาพเรียกว่า แผนภาพผีเสื้อ (butterfly diagram) ขอให้ศึกษาการเขียนแผนภาพผีเสื้อจากตัวอย่างที่ 6.1

<u>ตัวอย่างที่ 6.1</u> จงแสดงขั้นตอนของการคิดแผนภาพผีเสื้อสำหรับการแปลง FFT เมื่อ N=8

การกระจาย DFT 8 จุด ให้อยู่ในรูปของ DFT 4 จุด สองเทอมบวกกัน สามารถเขียนเป็นแผนภาพได้ดังนี้ ขอให้เปรียบเทียบกับสมการที่ 6.4



รูปที่ 6.4 *การกระจาย DFT 8 จุด เป็น DFT 4 จุด*

ขอนิยามสัญลักษณ์ ที่ใช้ในแผนภาพผีเสื้อ ดังนี้

ก่อนจะกระจายต่อไป เราสามารถทำสัมประสิทธิ์ที่คูณอยู่ในแผนภาพในรูปที่ 6.4 ให้ง่ายลงได้ โดยใช้คุณสมบัติความสมมาตรของ W_N ดังนี้

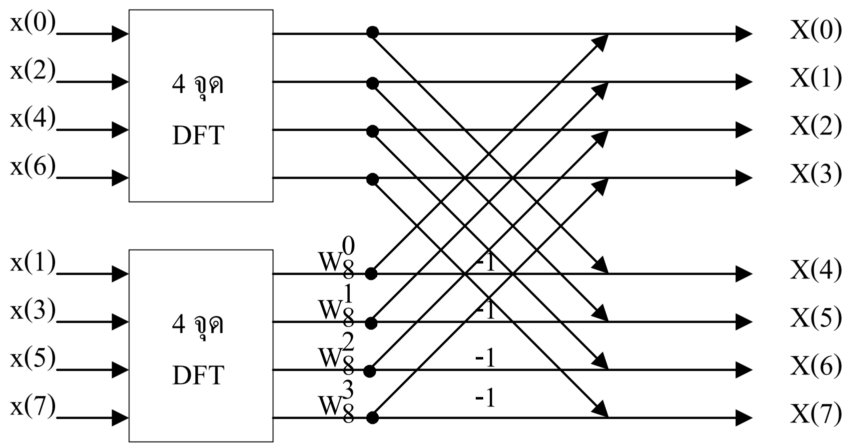
$$W_N^{k+N/2} = W_N^k W_N^{N/2} = W_N^k(-1) = -W_N^k \qquad (6.14)$$

ใช้คุณสมบัติตามสมการที่ 6.14 จะได้ว่า

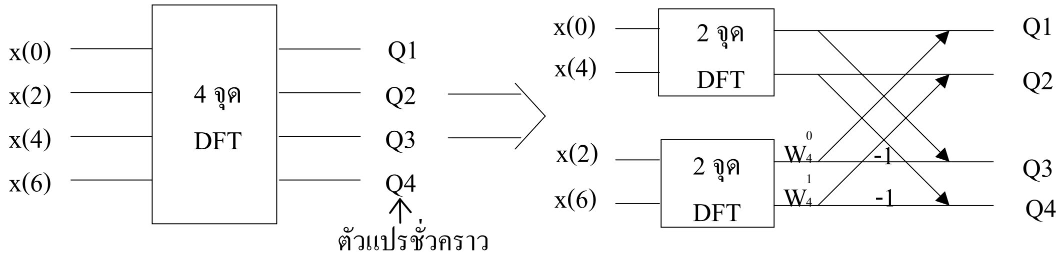
$$W_8^4 = -W_8^0,\ W_8^5 = -W_8^1,\ W_8^6 = -W_8^2,\ \text{และ}\ W_8^7 = -W_8^3 \qquad (6.15)$$

แทนค่าทั้งหมดลงในแผนภาพในรูป 6.4 จะได้แผนภาพดังรูปที่ 6.5

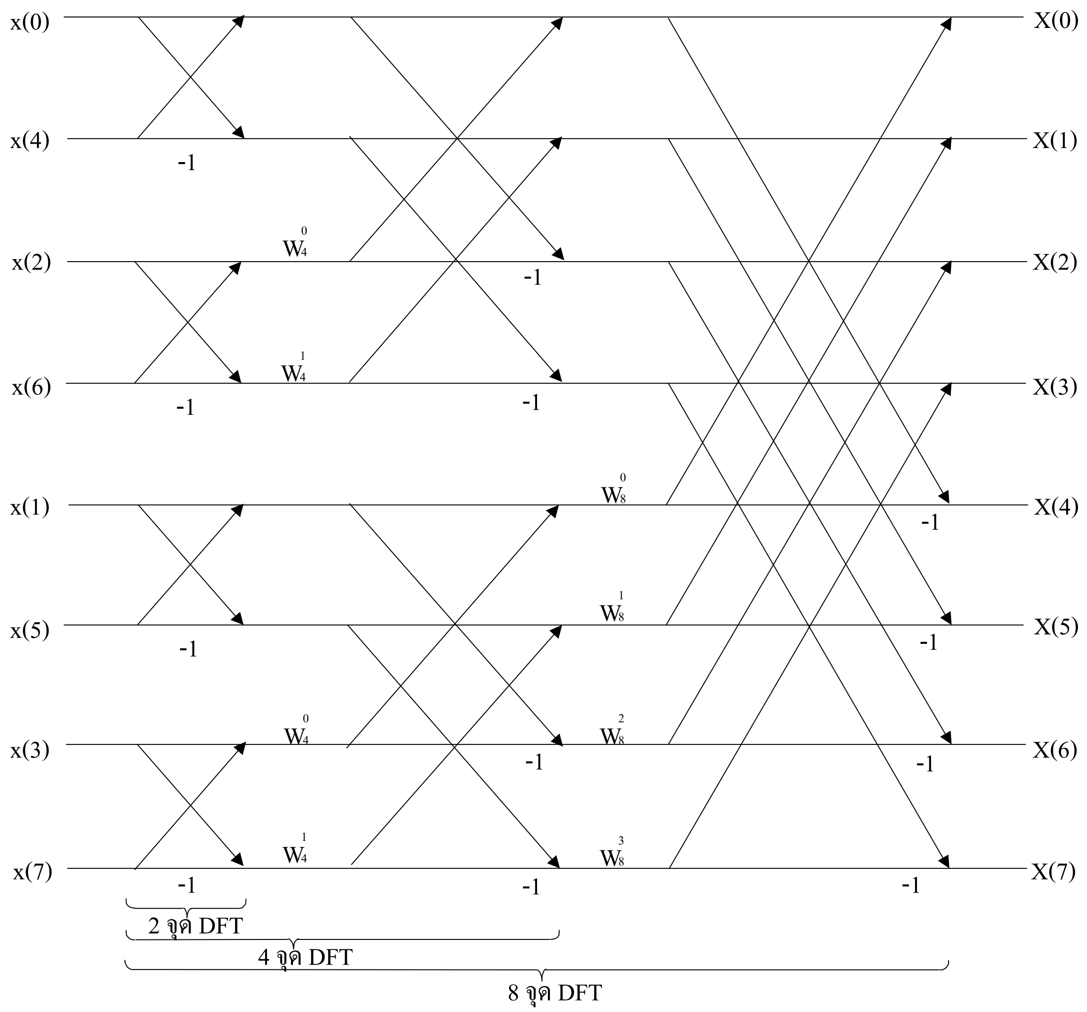
ในทำนองเดียวกัน DFT 4 จุดก็สามารถกระจายเป็น DFT 2 จุดได้ดังรูปที่ 6.6 และเมื่อรวมผลลัพธ์แต่ละส่วนเข้าเป็นแผนภาพเดียวกัน ก็จะปรากฏดังในรูปที่ 6.7 จากรูปที่ 6.7 นี้เราสามารถใช้เป็นแนวทางในการเขียนแผนภาพสำหรับ FFT N จุดใด ๆ ได้ทันที โดยไม่จำเป็นต้องเริ่มต้นจากการกระจายทีละขั้นดังที่ได้แสดงมา รวมทั้งใช้เป็นแนวทางในการเขียนโปรแกรมเพื่อคำนวณ FFT N จุดใด ๆ ได้



รูปที่ 6.5 *การกระจาย DFT 8 จุด เป็น DFT 4 จุด หลังจากใช้คุณสมบัติความสมมาตร*



รูปที่ 6.6 *การกระจาย DFT 4 จุด เป็น DFT 2 จุด*



รูปที่ 6.7 *แผนภาพรวมของการคำนวณ FFT 8 จุด*

จุดที่ควรสังเกตจากแผนภาพผีเสื้อของการคำนวณ FFT มีดังนี้คือ

1. ถ้าต้องการได้ผลตอบในเชิงความถี่เรียงตามลำดับจาก X(0), X(1), ..., X(7) เราต้องทำการเรียงลำดับสัญญาณขาเข้าใหม่ เป็นดังนี้ x(0), x(4), x(2), x(6), x(1), x(5), x(3), และ x(7) ลองเขียนลำดับเหล่านี้ในเลขฐานสองจะได้ดังตารางต่อไปนี้

ลำดับใหม่ฐานสิบ	ลำดับใหม่ฐานสอง	ลำดับปกติฐานสิบ	ลำดับปกติฐานสอง
0	000	0	000
4	100	1	001
2	010	2	010
6	110	3	011
1	001	4	100
5	101	5	101
3	011	6	110
7	111	7	111

จะเห็นได้ว่าลำดับใหม่เกิดจากการเรียงลำดับบิตจากหลังไปหน้าของลำดับปกติ (bit- reversed order) ซึ่งข้อนี้พบว่าเป็นจริงสำหรับ FFT ที่จำนวนจุดใด ๆ ด้วย

2. ค่าคงที่ W ที่ใช้คูณกับส่วนคี่ สามารถเปลี่ยนให้อยู่ในรูปของ W_8 ได้ทั้งหมด โดยคูณตัวห้อยและตัวยกกำลังด้วยค่าเดียวกัน ดังในตัวอย่างเราสามารถเปลี่ยนเทอมต่อไปนี้ได้

$$W_4^0 \rightarrow W_8^0 \text{ และ } W_4^1 \rightarrow W_8^2 \qquad (6.16)$$

ดังนั้น สามารถใช้ W_8 แทนค่าได้ทั้งหมด ซึ่งเราสามารถคำนวณ W_8 ที่ k ต่าง ๆ นี้ไว้ล่วงหน้าได้ และใช้มันเสมือนเป็นค่าคงที่สำหรับ FFT 8 จุด ข้อนี้ก็เป็นจริงเช่นกันสำหรับ FFT จำนวน N จุดใด ๆ

3. พิจารณาโดยรวมแล้ว จะได้ว่าการคำนวณ FFT N จุด ถูกแบ่งเป็น $\log_2 N$ ขั้นตอน โดยอาจประมาณได้ว่าแต่ละขั้นตอนมีการคำนวณเท่ากับ N CMAC's (มีเส้นแทยงในแผนภาพ N เส้นในทุก ๆ ขั้นตอน) ดังนั้นจะได้ว่า

$$\text{จำนวน CMAC ที่ต้องใช้ในการคำนวณ FFT N จุด} = N\log_2 N \qquad (6.17)$$

4. วิธี radix-2 นี้ใช้ได้ก็ต่อเมื่อค่า N เท่ากับ 2^b โดย b เป็นจำนวนเต็มบวกใด ๆ ซึ่งข้อนี้ไม่เป็นปัญหา เนื่องจาก ถ้าไม่สามารถแบ่งสัญญาณให้มีความยาวเท่ากับ 2^b ได้ ก็ใช้วิธีเติมศูนย์เพิ่มไปในสัญญาณให้ได้ความยาวตามที่ต้องการ

ตัวอย่างที่ 6.2 จงเปรียบเทียบจำนวน CMAC ที่ต้องใช้สำหรับการคำนวณ DFT 1024 จุด และ FFT 1024 จุด

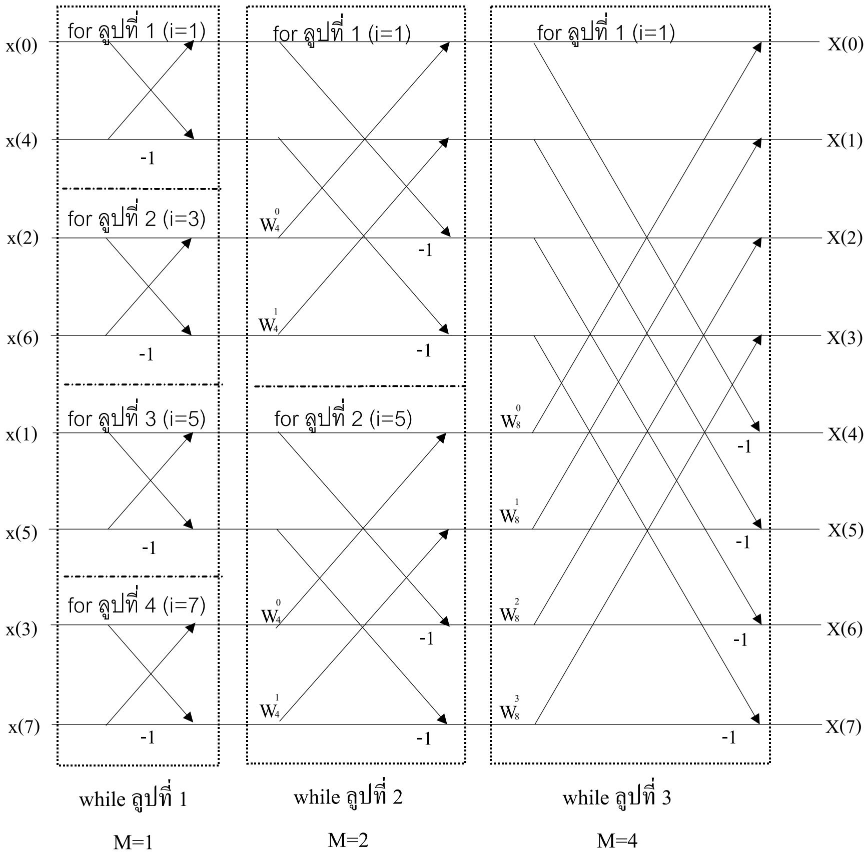
DFT ปกติใช้จำนวน CMAC $\approx N^2 = 1024\text{x}1024 \approx 10^6$ CMAC

FFT ใช้จำนวน CMAC $\approx N\log_2 N = 1024\text{x}10 \approx 10^4$ CMAC

นั่นคือ FFT ช่วยให้การคำนวณ DFT 1024 จุดนี้ เร็วขึ้นถึงประมาณ 100 เท่า

ดังได้กล่าวมาแล้วว่าการคำนวณ FFT มีหลายวิธี ซึ่งวิธีต่าง ๆ จะมีความซับซ้อนต่างกัน และได้จำนวน CMAC ต่างกันด้วย วิธี radix-2 ที่แสดงไว้นี้จัดเป็นวิธีที่เป็นพื้นฐาน และนิยมใช้วิธีหนึ่ง ผู้ที่สนใจขอให้ศึกษาวิธีคำนวณ FFT ด้วยวิธีอื่น ๆ จากหนังสืออ้างอิง [3], [8] ซึ่งหลายวิธีสามารถให้การคำนวณที่เร็วขึ้นได้ แต่ก็แลกด้วยความซับซ้อนที่มากขึ้น นอกจากนี้ ยังมีผู้คิดเทคนิคที่ทำให้การคำนวณเร็วขึ้นในกรณีที่สัญญาณขาเข้าของ FFT เป็นจำนวนจริง[8] ซึ่งเป็นกรณีส่วนใหญ่ที่เราใช้งาน

ใน Matlab มีคำฟังก์ชั่นภายในชื่อ fft ซึ่งจะใช้คำนวณการแปลง fft ได้อย่างรวดเร็วมาก ถ้าลองแทนคำสั่ง dft(x) ในโปรแกรม dftspec.m และ dftpsd.m (โปรแกรม 6.2 และ 6.3) ด้วย fft(x) แล้วสร้างเป็นฟังก์ชั่นใหม่ชื่อ fftspec.m และ fftpsd.m จะพบว่าโปรแกรมที่ใช้ fft นี้ทำงานได้เร็วขึ้นมาก



รูปที่ 6.8 *ส่วนย่อยที่ลูปต่าง ๆ คำนวณ สำหรับหา DFT 8 จุดตามโปรแกรมที่ 6.4*

โปรแกรมที่ 6.4 แสดงการเขียนโปรแกรมใน Matlab เพื่อคำนวณการแปลง FFT แบบ radix-2 สำหรับความยาว N ใด ๆ โดยในส่วนที่วนลูปเพื่อคำนวณ FFT ตามแผนภาพผีเสื้อนั้น จะประกอบด้วยการวน 2 ลูปชุดด้วยกัน คือ ลูป for ซ้อนอยู่ในลูป while ตัวอย่างเช่น การแปลง DFT 8 จุด จะมีจำนวนของลูปที่วน และค่าตัวแปรที่ใช้ในลูปดังแสดงในรูปที่ 6.8

จากตัวอย่างที่ 6.2 ที่บอกว่า FFT ควรคำนวณได้เร็วกว่า DFT ปกติประมาณ 100 เท่า แต่จากการทดสอบกับสัญญาณที่มีความยาว 1024 จุด พบว่าโปรแกรม myfft.m นี้ทำงานเร็วกว่า dft.m (โปรแกรมที่ 6.1) เพียงแค่ประมาณ 3 เท่า สาเหตุก็เนื่องมาจากปัจจัยอื่น ได้แก่ การที่ Matlab ไม่มีคำสั่งกระทำกับระดับบิต ดังนั้น การเปลี่ยนลำดับสัญญาณขาเข้าเป็นลำดับที่สลับบิตซ้ายขวาจึงกินเวลาคำนวณมาก และในส่วนของการคำนวณแผนภาพผีเสื้อซึ่งมีการชี้ค่าต่าง ๆ แบบพิสดารของตัวแปรอะเรย์ก็ทำให้การคำนวณช้า ในกรณีที่นำไปใช้งานซึ่งอาจมี N คงที่ และใช้ฮาร์ดแวร์ หรือภาษาแอสเซมบลี้เป็นตัวประมวลผล ก็จะทำให้การการคำนวณเร็วขึ้นกว่านี้มาก

อย่างไรก็ตาม โปรแกรมที่ 6.4 ได้แสดงให้เห็นว่า การคำนวณ FFT ถึงแม้จะมีขั้นตอนที่ซับซ้อนกว่าการคำนวณ DFT ปกติ แต่ก็จะทำงานได้เร็วกว่ามาก ผู้ที่สนใจสามารถดูตัวอย่างการใช้ภาษาซีเพื่อคำนวณ FFT ได้จากหนังสืออ้างอิง [1] และตัวอย่างของภาษาแอสเซมบลี้ของชิพ DSP ตระกูล TMS320 ได้จากหนังสืออ้างอิง [15]

```
function X=myfft(x)
N=length(x);
b=log10(N)/log10(2);
W=exp(-j*2*pi*[0:N/2-1]/N);
```

คำนวณค่าคงที่ W_N^k ไว้ก่อน

```
xold=x;
for i=0:N-1
   index_old=i;
   index=0;
   for j=1:b
       index=index*2;
       a=rem(index_old,2);
       index_old=(index_old-a)/2;
       index=index+a;
   end
   x(index+1)=xold(i+1);
end
```

สลับลำดับของสัญญาณขาเข้า เป็นลำดับที่สลับบิตซ้ายขวา

```
M=1;
while M < N,
   M2 = 2*M;
   for i=1:M2:N
     xtemp1 = x(i:i+M-1);
     xtemp2 = x(i+M:i+M2-1).*W(1:N/M2:N/2);
     x(i:i+M-1) = xtemp1 + xtemp2;
     x(i+M:i+M2-1) = -xtemp2 + xtemp1;
   end
   M=M2;
end
X=x;
```

คำนวณตามแผนภาพผีเสื้อ

โปรแกรมที่ 6.4 *myfft.m สำหรับคำนวณการแปลง FFT ที่จำนวนจุด N ใด ๆ*

การแปลง DFT ผกผัน (IDFT, Inverse Discrete Fourier Transform)

ลองพิจารณา สูตรของ IDFT เทียบกับ DFT จะพบว่ามีความคล้ายกันมาก ซึ่งก็พบว่าการหา IDFT สามารถหาได้โดยการใช้ DFT ดังสมการต่อไปนี้

$$x = \mathrm{IDFT}(X) = \frac{1}{N}(\mathrm{DFT}(X^*))^* \qquad (6.18)$$

เครื่องหมาย * หมายถึง conjugate ลองพิสูจน์สูตรนี้โดยใช้สมการของ DFT และแทนค่า X* ลงไปจะได้

$$DFT(X^*) = \sum_{k=0}^{N-1} X(k)^* \; e^{-j2\pi kn/N}$$

ดังนั้น จะได้ $\frac{1}{N}(DFT(X^*))^* = \frac{1}{N}\sum_{k=0}^{N-1}\left\{X(k)^* \; e^{-j2\pi kn/N}\right\}^* = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\; e^{j2\pi kn/N}$

ซึ่งมีค่าเท่ากับ IDFT(X) = x(n) จริงตามสมการที่ 6.18

ในทำนองเดียวกัน IFFT ซึ่งมีความหมายเดียวกับ IDFT และสามารถหาได้จากการคำนวณ FFT ดังสมการต่อไปนี้

$$x = IFFT(X) = \frac{1}{}(FFT(X^*))^* \qquad (6.19)$$

คุณสมบัติของ DFT

สมมติให้ X_1=DFT(x_1) และ X_2=DFT(x_2) จะได้ว่า

1. คุณสมบัติความเป็นเชิงเส้น

$$a_1x_1(n) + a_2x_2(n) \quad \overset{DFT;N}{\longrightarrow} \quad a_1X_1(k) + a_2X_2(k) \qquad (6.20)$$

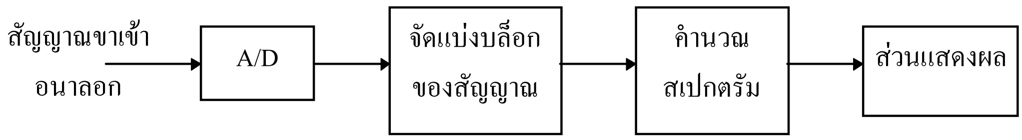
2. คุณสมบัติการเลื่อนทางเวลา

$$x_1(n\text{-}m) \quad \overset{DFT;N}{\longrightarrow} \quad e^{-j2\pi m/N}\, X_1(k) \qquad (6.21)$$

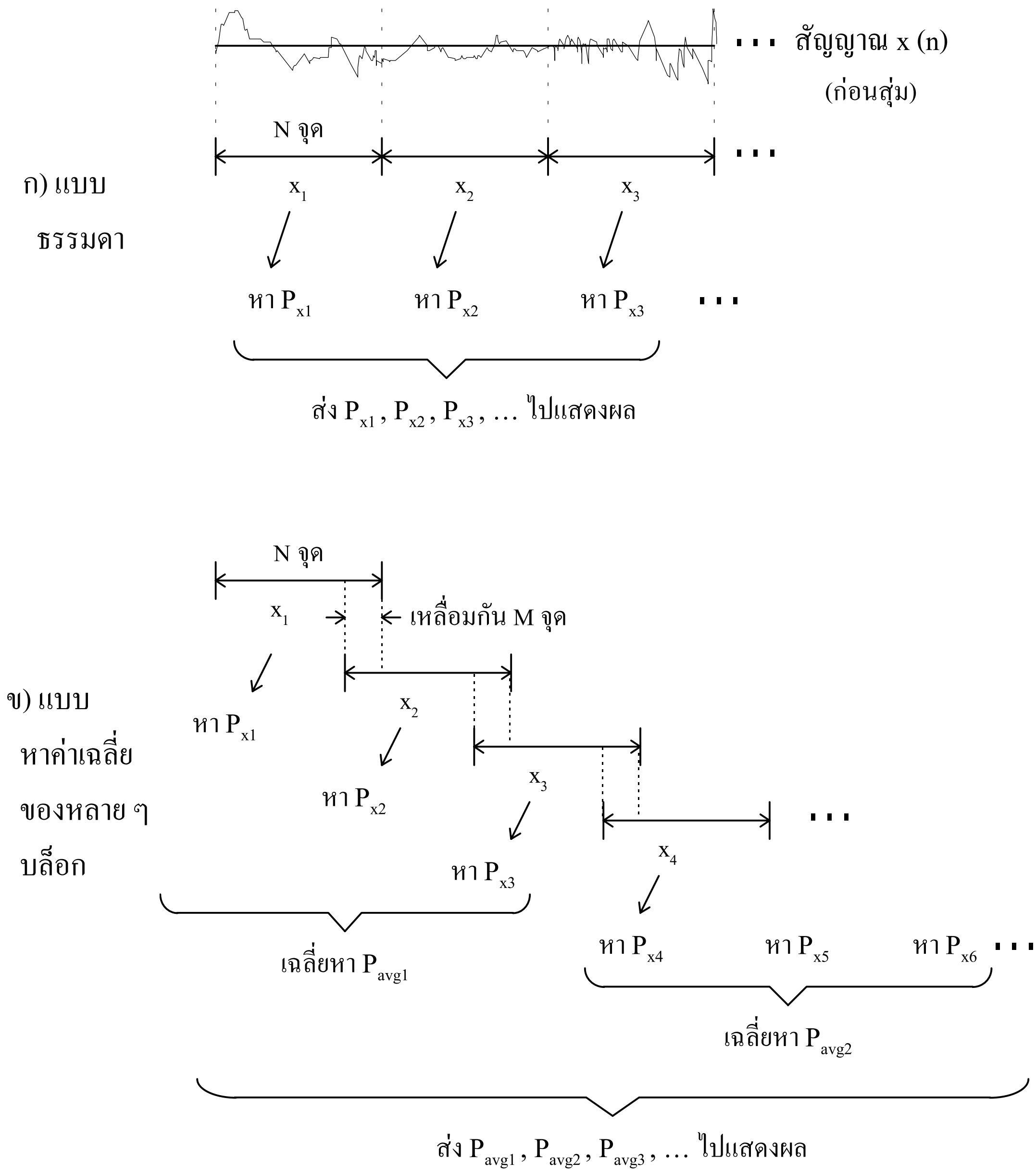
เครื่องวิเคราะห์สเปกตรัม และการปรับปรุงผลลัพธ์ที่ได้จาก FFT

เครื่องวิเคราะห์สเปกตรัม (spectrm analyzer) คือ เครื่องที่รับสัญญาณขาเข้าเป็นแอนะลอก และแสดงสเปกตรัม (ของกำลัง) ของสัญญาณออกทางหน้าจอ และสามารถติดตามการเปลี่ยนแปลงสเปกตรัมของสัญญาณได้อย่างทันทีทันใด เครื่องวิเคราะห์สเปกตรัมอย่างง่ายสามารถสร้างได้โดยใช้ FFT ตามสมการที่ 6.5 ในการคำนวณหาค่าสเปกตรัมของกำลัง

แผนภาพของเครื่องวิเคราะห์แสดงดังรูปที่ 6.9 โดยหลังจากที่สัญญาณแปลงสัญญาณเป็นแบบไม่ต่อเนื่องแล้ว ก็จะทำการจัดสัญญาณเป็นบล็อก ๆ ละ N จุด แล้วทำการหาสเปกตรัมของสัญญาณทีละบล็อกเพื่อส่งไปแสดงผลดังรูปที่ 6.10 ก)



***รูปที่ 6.9** แผนภาพของเครื่องวิเคราะห์สเปกตรัม*



***รูปที่ 6.10** การจัดบล็อกของสัญญาณเพื่อหาสเปกตรัม*

การจัดบล็อกตามรูปที่ 6.10 ก) ให้ผลลัพธ์พอใช้ได้ แต่เรายังมีวิธีปรับปรุงรูปร่างของสเปกตรัมให้ดีขึ้น (ชัดขึ้น และใกล้เคียงความเป็นจริงมากขึ้น) โดยใช้สองวิธี ดังต่อไปนี้

<u>เทคนิคที่ 1</u> ถ้าสัญญาณขาเข้าเป็นสัญญาณแรนดอม (ซึ่งคือสัญญาณที่เราต้องพบในชีวิตจริง) จะทำให้สเปกตรัมที่ได้จากสัญญาณ 1 บล็อกมีเส้นที่ไม่เรียบ คือจะมีผลของความไม่แน่นอนของสัญญาณแรนดอมปนอยู่มาก ดังเช่นในรูปที่ 6.3 ค) เทคนิคที่ใช้ปรับให้เส้นสเปกตรัมเรียบ และถูกต้องมากขึ้น ก็ทำได้โดยการนำสเปกตรัมที่ได้จากบล็อกที่อยู่ติด ๆ กันมา "เฉลี่ย" กัน สมมติว่าใช้ K บล็อกมาทำการเฉลี่ย จะได้สเปกตรัมที่เกิดจากการเฉลี่ย คือ

$$P_{avg}(f) = \frac{1}{K}\sum_{i=1}^{K} P_{xi}(f) \qquad (6.22)$$

นอกจากนี้ สัญญาณแต่ละบล็อกก็อาจแบ่งให้มีส่วนเหลื่อมกัน เพื่อเพิ่มประสิทธิภาพในการติดตามสเปกตรัมที่เปลี่ยนแปลงในสัญญาณ ในรูปที่ 6.10 ข) แสดงการแบ่งบล็อกโดยมีส่วนที่เหลื่อมกันเท่ากับ M จุด และใช้สเปกตรัมที่ได้จาก 3 บล็อกมาเฉลี่ยกัน (K = 3) เทคนิคที่ใช้นี้เป็นเทคนิคที่ใช้ประมาณค่าสเปกตรัมของกำลังที่เรียกว่า Averageing Modified Periodogram หรือ วิธีของ Welch[3]

บางท่านอาจสงสัยว่า การเลือกค่า N, M, และ K ขึ้นอยู่กับอะไร อาจพิจารณาได้ดังนี้

ก) ค่า N เล็กไปจะทำให้ได้สเปกตรัมที่ไม่ละเอียด (low resolution) แต่ถ้าค่า N ใหญ่ไปก็จะลดความสามารถในการติดตามการเปลี่ยนแปลงของสเปกตรัมลง จึงไม่เหมาะกับสัญญาณที่มีสเปกตรัมเปลี่ยนแปลงรวดเร็ว

ข) K เล็กไปทำให้รูปสเปกตรัมไม่เรียบ ยังมีผลของความไม่แน่นอนอยู่มาก แต่ถ้า K ใหญ่ไปก็จะลดความสามารถในการติดตามการเปลี่ยนแปลงของสเปกตรัมลง

ค) M เล็กไปจะลดความสามารถในการติดตามการเปลี่ยนแปลงของสเปกตรัมลง แต่ถ้า M ใหญ่ไปก็สิ้นเปลืองการประมวลผลมาก

<u>ตัวอย่างที่ 6.3</u> ในหนังสืออ้างอิง [2] ได้เสนอการทำเครื่องวิเคราะห์สเปกตรัมเอง โดยใช้ชิพ DSP ตระกูล TMS320 เป็นตัวประมวลผล และแสดงผลบนหน้าจอเครื่องไมโครคอมพิวเตอร์ เครื่องนี้ใช้อัตราสุ่มเท่ากับ 40 kHz, N=128, M=64, และ K=8 จงหาย่านความถี่ที่เครื่องวิเคราะห์สเปกตรัมนี้ทำงาน และความยาวของสัญญาณแอนะลอกที่ใช้ในการคำนวณสเปกตรัม 1 รูป

ย่านความถี่ที่เครื่องนี้ทำงาน คือ 0 ถึง 20 kHz (ครึ่งหนึ่งของอัตราสุ่ม)

สำหรับจำนวนจุดของสัญญาณไม่ต่อเนื่อง ที่นำมาใช้คำนวณสเปกตรัม 1 รูป สามารถหาได้จากผลรวมของบล็อกทั้งหมด ลบด้วยส่วนที่เหลื่อมกัน จะได้สูตร คือ

$$\text{จำนวนจุดที่ใช้คำนวณสเปกตรัม 1 รูป} = NK - M(K-1) \qquad (6.23)$$

ซึ่งในที่นี้เท่ากับ (128)(8) - (64)(7) = 576 จุด เทียบไปเป็นความยาวของสัญญาณแอนะลอกได้เท่ากับ 576 / f_s = 576/40000 = 14.4 ms

ค่านี้บอกคร่าว ๆ ว่า เครื่องนี้สามารถติดตามสเปกตรัมที่เปลี่ยนแปลงช้ากว่าประมาณ 14.4 ms (คือ สัญญาณที่ไม่มีการเปลี่ยนแปลงสเปกตรัมมากนักในช่วง 14.4 ms ใด ๆ) เช่น สัญญาณเสียงซึ่งได้มีการวิเคราะห์มาว่า เสียงคนพูดมีสเปกตรัมที่ประมาณได้ว่าคงที่ หรือเปลี่ยนแปลงน้อยมากในช่วง 10 - 20 ms ใด ๆ ดังนั้น เครื่องนี้เหมาะกับสัญญาณเสียงคนด้วย

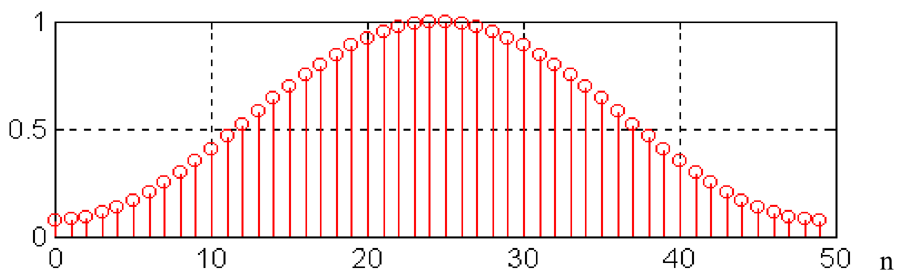
เทคนิคที่ 2 เนื่องจากเราใช้วิธีตัดสัญญาณเพื่อมาหาสเปกตรัมเป็นบล็อก ๆ ซึ่งเท่ากับเป็นการหาสเปกตรัมของ "บล็อกของสัญญาณ" ไม่ใช่ตัวสัญญาณจริง ๆ ที่เข้ามาติดต่อกันไม่มีจุดเริ่มต้น หรือสิ้นสุด การตัดสัญญาณเป็นบล็อกนี้จะทำ ให้เกิดความคลาดเคลื่อนในสเปกตรัมที่ได้ วิธีลดผลของความคลาดเคลื่อนนี้ทำได้โดยคูณสัญญาณแต่ละบล็อก หรือ $x_i(n)$ ด้วยฟังก์ชั่นหน้าต่าง (window function) ก่อนที่จะทำการหาสเปกตรัม

ฟังก์ชั่นหน้าต่างมีหลายแบบซึ่งจะได้กล่าวถึงอีกครั้งในเรื่องตัวกรอง FIR เป็นฟังก์ชั่นที่มีลักษณะสมมาตร และมีค่าสูงสุดที่จุดกึ่งกลาง มีความยาวเท่าไรก็ได้ตามต้องการ คือแล้วแต่จะแทนค่าลงในสูตร ตัวอย่างของฟังก์ชั่นหน้าต่างที่ใช้บ่อยก็คือ หน้าต่างแฮมมิ่ง (Hamming window) ซึ่งมีสมการ คือ

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \quad n = 0, 1, 2, \ldots, N-1 \qquad (6.24)$$

$$x_{i,new}(n) = x_i(n)\, w(n) \qquad (6.25)$$

เราต้องการ w(n) ที่มีความยาวเท่ากับ 1 บล็อกของสัญญาณ ดังนั้น ต้องใช้ N ในสมการ 6.22 เท่ากับความยาว 1 บล็อกของสัญญาณ รูปร่างของหน้าต่างแบบแฮมมิ่งแสดงอยู่ในรูปที่ 6.11 ซึ่งจะสังเกตเห็นได้ว่า ฟังก์ชั่นหน้าต่างมีค่าเล็กที่ส่วนต้น และปลาย ซึ่งเป็นส่วนที่ลดผลของการเปลี่ยนแปลงที่จุดเริ่มต้น และจุดสิ้นสุดของบล็อก



รูปที่ 6.11 ตัวอย่างของฟังก์ชั่นหน้าต่างแบบแฮมมิ่ง ที่ N=50

เทคนิคที่กล่าวมาทั้งสองนี้ เป็นเทคนิคที่ใช้ในวิธีของ Welch หรือเรียกอีกชื่อหนึ่งว่า วิธี averaging modified periodogram ซึ่งนิยมใช้กันมาก ผู้สนใจรายละเอียดเพิ่มเติมดูได้จาก [3] ใน Matlab DSP toolbox มีฟังก์ชั่นชื่อ psd ซึ่งใช้คำนวณหาสเปกตรัมของกำลังโดยใช้วิธีนี้

การคอนโวลูชันแบบเร็ว (Fast Convolution)

ดังที่ได้ทราบมาแล้วว่า คอนโวลูชันเป็นการกระทำที่ใช้สำหรับหาผลตอบของตัวกรอง FIR ดังสมการ

$$y(n) = h(n) * x(n) \tag{6.26}$$

การคอนโวลูชันแบบนี้มีชื่อเรียกว่า คอนโวลูชันแบบเชิงเส้น (linear convolution) ถ้าให้ N เป็นความยาวของ h(n) การคอนโวลูชันแบบนี้ต้องการการคำนวณประมาณ N CMAC's ต่อ 1 จุด ของสัญญาณขาออก ในการประยุกต์ใช้งานประเภทเวลาจริงที่มีอัตราการสุ่มสูง และในการประยุกต์ใช้งานหลายอย่าง N ก็มีค่าสูงเป็นพัน ๆ จึงมีจำเป็นที่จะต้องคำนวณคอนโวลูชันนี้อย่างรวดเร็ว ต่อมาจึงได้มีผู้คิดค้นวิธีทำคอนโวลูชันอย่างรวดเร็วขึ้น โดยอาศัยการคำนวณ FFT มาช่วย ซึ่งเราเรียกโดยรวมว่า วิธี fast convolution

จากสมการของคอนโวลูชันถ้าเราแปลงสัญญาณทั้งหมดให้อยู่ในเชิงความถี่โดย DTFT จะได้สมการในรูปของการคูณกัน ดังนี้

$$Y = \mathrm{DTFT}(y) = \mathrm{DTFT}(h)\,\mathrm{DTFT}(x)$$

$$\text{ดังนั้น}\quad y = \mathrm{IDTFT}(Y) = \mathrm{IDTFT}\{\,\mathrm{DTFT}(h)\,\mathrm{DTFT}(x)\,\} \tag{6.27}$$

นั่นคือ เราสามารถใช้การคำนวณ DTFT และ IDTFT เพื่อหาคอนโวลูชันได้ แต่การกระทำ DTFT เป็นการกระทำที่ใช้ไม่สะดวก เพราะให้สัญญาณในเชิงความถี่เป็นแบบต่อเนื่อง คำถามก็คือ เราจะสามารถแทน DTFT และ IDTFT ในสมการนี้ด้วย DFT และ IDFT ได้เลยหรือไม่ ซึ่งคำตอบคือ ไม่ได้ การแทนด้วย DFT และ IDFT จะยังผลทำให้ h และ x มีความหมายเป็นสัญญาณที่เป็นคาบ ซึ่งจะทำให้ได้สัญญาณขาออกไม่เหมือนเดิม เราจะนิยามเป็นการกระทำแบบใหม่ เรียกว่า คอนโวลูชันแบบวงกลม (circular convolution) และเขียนแทนด้วยสัญลักษณ์ ⊗ ดังนี้

$$\tilde{y} = h \otimes x \tag{6.28}$$

$$\tilde{y} = \mathrm{IDFT}\{\,\mathrm{DFT}(h)\,\mathrm{DFT}(x)\,\}$$

$$\text{หรือ} \quad \tilde{y} = \text{IFFT}\{\text{FFT(h) FFT(x)}\} \qquad (6.29)$$

ซึ่ง h และ x ต้องมีความยาว N จุดเท่ากัน หรือถ้าตัวใดตัวหนึ่งสั้นกว่าก็ให้เติมจุดที่เป็นศูนย์เข้าไปเพื่อให้ได้ความยาวเท่ากัน ลองศึกษาคอนโวลูชันแบบวงกลมจากตัวอย่างที่ 6.4 ว่ามีลักษณะแตกต่างจากคอนโวลูชันปกติอย่างไร

ตัวอย่างที่ 6.4 จากตัวอย่างในบทที่ 3 ซึ่งมี x(n) = [1 2 3 2 1] และ h(n) = [4 2 1] จงหา $\tilde{y}$ ซึ่งเกิดจากคอนโวลูชันแบบวงกลมของ x และ h

	คอนโวลูชั่นแบบเชิงเส้น							คอนโวลูชันแบบวงกลม				
n →	0	1	2	3	4	5	6	0	1	2	3	4
x(0)h(n) =	4	2	1					4	2	1		
x(1)h(n-1) =		8	4	2					8	4	2	
x(2)h(n-2) =			12	6	3					12	6	3
x(3)h(n-3) =				8	4	2		2			8	4
x(4)h(n-4) =					4	2	1	2	1			4
y(n) =	4	10	17	16	11	4	1	8	11	17	16	11

ผลตอบ $\tilde{y}$ ในที่นี้คือ [8 11 17 16 11]

ผลตอบจะมีความยาวเท่ากับความยาวของตัวตั้งเสมอ ในที่นี้มีความยาวเท่ากับ 5 จุดที่แตกต่างไปของคอนโวลูชันแบบวงกลม ก็คือ ผลตอบที่ตำแหน่ง n=5 จะถูกวนขึ้นมาบวกกับผลตอบที่ตำแหน่ง n=0 และผลตอบที่ตำแหน่ง n=6 ก็จะถูกวนขึ้นมาบวกกับผลตอบที่ตำแหน่ง n=1 เป็นเช่นนี้ไปเรื่อย ๆ เราอาจใช้วิธีหาคอนโวลูชันแบบเชิงเส้นก่อน แล้วตัดเอาผลลัพธ์ 2 จุดหลังมาบวกเข้ากับผลลัพธ์ 2 จุดแรกก็ได้ นั่นคือ

y = [4 10 17 16 11 4 1]

$\tilde{y}$ = [4 10 17 16 11] + [4 1 0 0 0] = [8 11 17 16 11]

อย่างไรก็ดี อย่าลืมว่านี่คือการแสดงวิธีหาคอนโวลูชันแบบวงกลมเท่านั้น ในการใช้งานจริงเราจะใช้ FFT ในการคำนวณดังสมการข้างต้น ซึ่งเขียนเป็นไฟล์ Matlab ได้ดังโปรแกรมที่ 6.6

```
function y=circonv(x,h)
Nx=length(x); Nh=length(h); N = max(Nx,Nh);
h = [h, zeros(1,N-Nh)];
x = [x, zeros(1,N-Nx)];          ปรับความยาวของ x กับ h ให้เท่ากัน
y = conv(x,h);
y = [y,0];
y = y(1:N) + y(N+1:2*N);  นำผลลัพธ์ส่วนที่เกิน N จุด มาบวกสมทบกับส่วนหน้า
```

***โปรแกรมที่ 6.5** circonv.m สำหรับคำนวณคอนโวลูชันแบบวงกลม*

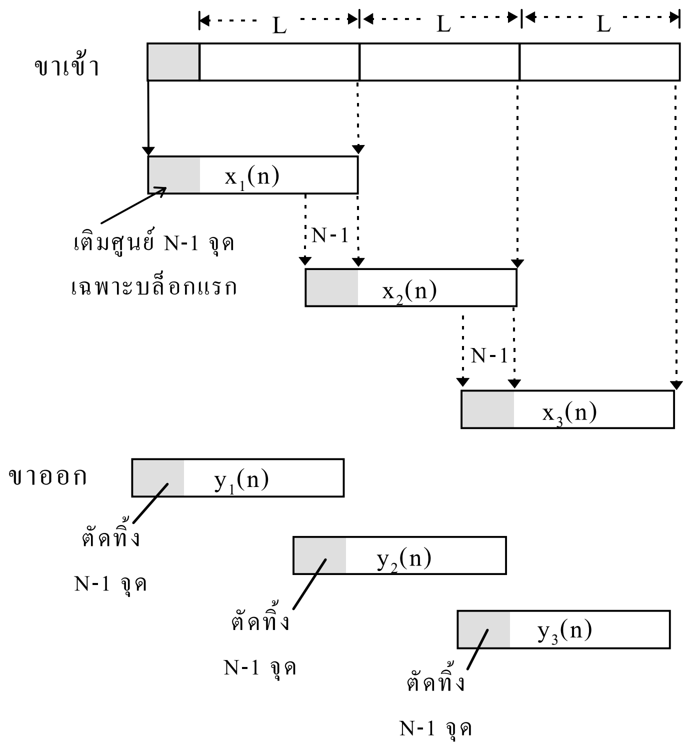
```
function y=circonv2(x,h)
Nx=length(x); Nh=length(h); N = max(Nx,Nh);
h = [h, zeros(1,N-Nh)];
x = [x, zeros(1,N-Nx)];
y = ifft( fft(x) .* fft(h) );
```

โปรแกรมที่ 6.6 *circonv2.m สำหรับคำนวณคอนโวลูชันแบบวงกลมโดยใช้ FFT*

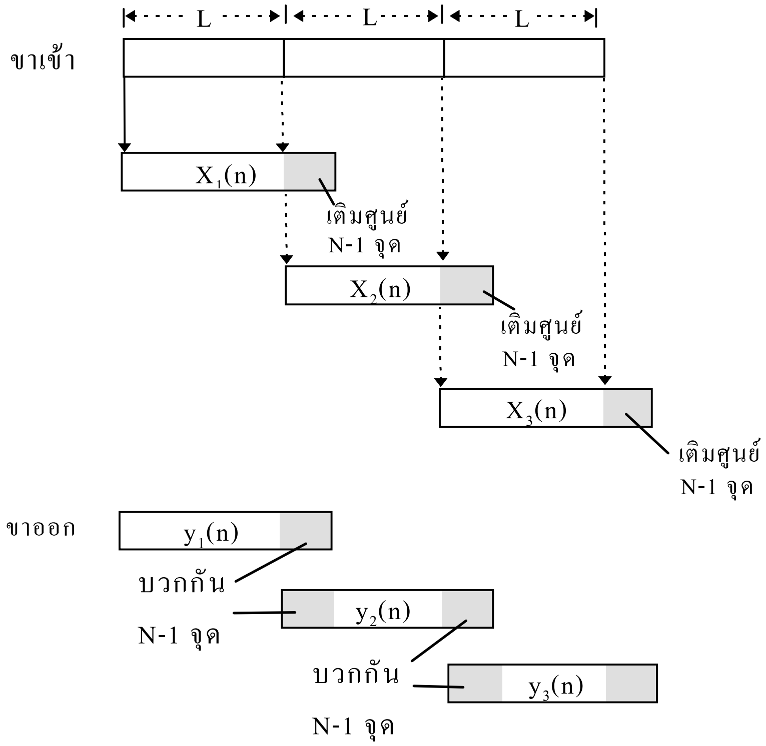
จะเห็นได้ว่าคอนโวลูชันแบบวงกลมให้ผลตอบ N-1 ค่าแรกต่างจากคอนโวลูชันแบบเชิงเส้น แต่ส่วนที่เหลือถูกต้องตามคอนโวลูชันแบบเชิงเส้น (ในตัวอย่างข้างต้น ค่า [17 16 11] เป็นส่วนที่ผลลัพธ์ทั้งสองให้ค่าตรงกัน) เราสามารถนำเอาคอนโวลูชันแบบวงกลมมาประยุกต์ใช้ในการทำคอนโวลูชันแบบเชิงเส้นได้โดยการใช้เทคนิคบางอย่างเข้าช่วย เทคนิคเหล่านี้ได้แก่ วิธี overlap-save และวิธี overlap-add ดังแสดงในรูปที่ 6.12 และ 6.13

ในการใช้งานจริง สัญญาณขาเข้าคือ x(n) จะเข้ามาอย่างต่อเนื่อง เราจะแบ่ง x(n) เป็นบล็อกเพื่อที่จะทำคอนโวลูชันแบบวงกลมทีละบล็อก สมมติว่าเราแบ่งแต่ละบล็อกมีความยาวเท่ากับ L + N -1 จุด ซึ่ง **L ต้องมากกว่า N-1** เมื่อ N เป็นความยาวของ h(n) แต่ละบล็อกมีส่วนที่เหลื่อมกับบล็อกข้างหน้าเท่ากับ N-1 จุด ดังแสดงในรูปที่ 6.12 เรียก แต่ละบล็อกว่า $x_1(n)$, $x_2(n)$, $x_3(n)$, …

นำ $x_1(n)$, $x_2(n)$, $x_3(n)$, … มาทำคอนโวลูชันแบบวงกลมกับ h(n) ทีละสัญญาณ ซึ่งจะได้ผลลัพธ์เป็น $y_1(n)$, $y_2(n)$, $y_3(n)$, … แต่ละตัวมีความยาว L+N-1 เช่นกัน จากนั้นตัดส่วนหัวของ $y_i(n)$ ทิ้ง N-1 จุด เพราะเป็นส่วนของผลลัพธ์ที่มีค่าไม่ตรงกับคอนโวลูชันแบบเชิงเส้น



รูปที่ 6.12 *แสดงการทำคอนโวลูชันแบบเชิงเส้นโดยวิธี overlap-save*



รูปที่ 6.13 *แสดงการทำคอนโวลูชันแบบเชิงเส้น โดยวิธี overlap-add*

นำส่วนที่เหลืออยู่ซึ่งมีความยาว L จุดต่อบล็อก มาต่อกันจะได้เป็นสัญญาณขาออก ซึ่งสัญญาณขาออกนี้จะเหมือนกับผลลัพธ์ของการคอนโวลูชันแบบเชิงเส้นระหว่าง x(n) และ h(n)

นั่นคือ ในแต่ละบล็อก เราคำนวณคอนโวลูชันแบบวงกลมกับสัญญาณที่มีความยาว L+N-1 จุด เพื่อให้กำเนิดผลลัพธ์ที่ต้องการจำนวน L จุด คำถามมีอยู่ว่า เบ็ดเสร็จแล้ว การใช้คอนโวลูชันแบบวงกลมบวกกับเทคนิคนี้ จะเร็วกว่าการทำคอนโวลูชันแบบเชิงเส้นหรือไม่อย่างไร

จากการคำนวณคอนโวลูชันแบบวงกลมโดยสมการ $\tilde{y}$ = IFFT{ FFT(h) FFT(x) } การคำนวณนี้ประกอบด้วย การทำ DFT 2 ครั้ง (ใช้ $2L_x\log_2 L_x$ CMAC), การคูณผลที่ได้จาก DFT ทั้งสอง (ใช้ประมาณ L_x CMAC), และการทำ IDFT 1 ครั้ง (ใช้ $L_x\log_2 L_x$ CMAC) รวมทั้งหมดจะต้องการการคำนวณประมาณ $(3L_x\log_2 L_x + L_x)$ CMAC

ในที่นี้ถ้าเราคิดการคำนวณต่อ 1 บล็อก ซึ่งมี $L_x = L+N-1$ จะได้ว่าต้องใช้การคำนวณ

$$\mathbf{3(L+N-1)\log_2(L+N-1) + (L+N-1)} \text{ CMAC ต่อ จำนวนผลตอบ L จุด} \qquad (6.30)$$

ในกรณีที่นำไปใช้กับระบบที่มีค่าสัมประสิทธิ์คงที่ (ไม่เปลี่ยนแปลงตามเวลา) เราสามารถคำนวณ FFT(h) ไว้ล่วงหน้าก่อน แล้วเก็บค่าไว้ในตาราง ทำให้ไม่จำเป็นต้องคำนวณ FFT(h) อีก ดังนั้น ในกรณีนี้จะทำให้การคำนวณลดลงเหลือ

$$\mathbf{2(L+N-1)log_2(L+N-1) + (L+N-1)} \text{ CMAC ต่อ จำนวนผลตอบ L จุด} \qquad (6.31)$$

โดยที่ L+N-1 ต้องเป็นจำนวนยกกำลังจำนวนเต็มของสอง คือเท่ากับ 2^b โดย b เป็นจำนวนเต็มบวก ในขณะที่วิธีคอนโวลูชันแบบเชิงเส้นธรรมดา ถ้าสัญญาณขาเข้าเป็นจำนวนจริงจะต้องใช้

N MAC ต่อจำนวนผลตอบ 1 จุด

หรือก็คือ **LN** MAC ต่อจำนวนผลตอบ L จุด (6.32)

สังเกตว่าในสมการที่ 6.30 และ 6.31ใช้หน่วยการคูณและบวกแบบจำนวนเชิงซ้อน (CMAC) ในขณะที่สมการที่ 6.32 ใช้หน่วยการคูณและบวกแบบจำนวนจริง (MAC) ซึ่งสามารถเปรียบเทียบกันได้ว่า

1 CMAC = การคูณจำนวนเชิงซ้อน 1 ครั้ง และ บวกจำนวนเชิงซ้อน 1 ครั้ง
= (คูณ 4 ครั้ง และบวก 2 ครั้ง) และ (บวก 2 ครั้ง)
= คูณ 4 ครั้ง และบวก 4 ครั้ง

ดังนั้น 1 CMAC = 4 MAC (6.33)

<u>ตัวอย่างที่ 6.5</u> ตัวกรองหนึ่งมีค่าสัมประสิทธิ์คงที่ และมีความยาว N=1024 จุด จงเปรียบเทียบวิธีคอนโวลูชันธรรมดา กับวิธีคอนโวลูชันแบบเร็ว (fast convolution) โดยใช้ขนาดของบล็อกในการทำคอนโวลูชันแบบวงกลมเท่ากับ 2048 จุด

คำนวณ L คือ ขนาดของบล็อกในการทำคอนโวลูชันแบบวงกลมจาก L+N-1 = 2048
จะได้ L = 2048 - N + 1 = 2048 - 1023 = 1025

สำหรับวิธีคอนโวลูชันแบบเร็วที่มีการคำนวณ FFT(h) ล่วงหน้าแล้ว ต้องใช้ปริมาณการคำนวณต่อจำนวนผลตอบ L จุด ตามสมการที่ 6.31 ซึ่งเท่ากับ

$$\begin{aligned} 2(2048)(\log_2 2048) + 2048 &= 2(2048)(11) + 2048 \text{ CMAC} \\ &= 47104 \text{ CMAC} \\ &= 4(47104) = 188{,}416 \text{ MAC} \end{aligned}$$

สำหรับวิธีคอนโวลูชันธรรมดา ต้องใช้ปริมาณการคำนวณต่อจำนวนผลตอบ L จุด ตามสมการที่ 6.32 ซึ่งเท่ากับ (1025)(1024) = 1,049,600 MAC

ดังนั้น วิธีคอนโวลูชันแบบเร็ว สามารถคำนวณได้เร็วกว่าประมาณ 1049600/188416 = **5.57 เท่า** ถ้าเราเปลี่ยนขนาดของบล็อกไป ก็จะทำให้ได้อัตราส่วนนี้เปลี่ยนไป เช่น ในข้อนี้ถ้าใช้ L+N-

1= 4096 จะได้อัตราส่วนมากถึง 30 เท่า! แต่ผลเสียของการใช้ขนาดของบล็อกใหญ่ก็คือ จะมีการหน่วงเวลา (delay) จากสัญญาณขาเข้าไปขาออกมากขึ้น และต้องการหน่วยความจำในการประมวลมากขึ้น ดังนั้นการเลือกขนาดของบล็อกให้เหมาะสมจึงเป็นสิ่งสำคัญอันหนึ่ง

การใช้งานคอนโวลูชันแบบเร็ว มีประเด็นที่น่าสนใจนอกเหนือจากที่ได้วิเคราะห์ไปแล้ว คือ

- คอนโวลูชันแบบเร็วไม่สามารถลดปริมาณการคำนวณได้ สำหรับตัวกรอง FIR ที่มีอันดับต่ำ ๆ (เราสามารถพิสูจน์ได้โดยใช้การคำนวณเหมือนในตัวอย่างที่ผ่านมา)โดยทั่วไปมันสามารถใช้งานได้ดีกับตัวกรองที่มีอันดับสูงเป็นระดับพันขึ้นไป
- คอนโวลูชันแบบเร็วจะทำให้สัญญาณมีการหน่วงอย่างน้อยเท่ากับ 1 บล็อก คือ L ขั้นเวลา การออกแบบจึงต้องคำนึงถึงว่า การหน่วงนี้จะไม่กระทบกระเทือนต่อการใช้งาน เพราะงานบางอย่างต้องการการหน่วงเวลาที่ไม่มากเกินไป
- การใช้คอนโวลูชันอย่างเร็วต้องมีการเขียนโปรแกรมที่ซับซ้อนกว่าคอนโวลูชันธรรมดามาก ซึ่งอาจส่งผลให้การทำงานช้าลงกว่าที่ได้วิเคราะห์

ตัวอย่างที่ 6.6 จากตัวอย่างที่ 6.5 ถ้านำตัวกรองนี้ไปใช้งานแบบเวลาจริง โดยมีความถี่ในการสุ่มสัญญาณเท่ากับ 10 kHz จงหาจำนวน CMAC ต่อวินาทีที่ต้องใช้

จากตัวอย่างที่ 6.4 ปริมาณการคำนวณสำหรับวิธีคอนโวลูชันแบบเร็วเท่ากับ 47,104 CMAC ต่อจำนวนผลตอบ 1025 จุด ซึ่งเท่ากับ $\dfrac{47104}{1025} = 45.955$ CMAC ต่อจุด

f_s = 10kHz หมายถึง มีอัตราของสัญญาณขาเข้า 10,000 จุดต่อวินาที ดังนั้นจะต้องใช้ปริมาณการคำนวณเท่ากับ (45.955)(10000) = 459,550 CMAC ต่อวินาที

ผลลัพธ์ที่ได้นี้สามารถนำไปประมาณว่า จำเป็นจะต้องใช้ตัวประมวลผลที่มีความเร็วแค่ไหนจึงจะสามารถทำงานได้

บทที่ 7

ตัวกรองแบบ FIR

เราได้ศึกษาเกี่ยวกับหลักการ และวิธีการวิเคราะห์ระบบแบบไม่ต่อเนื่องมาแล้วในบทก่อน ๆ รวมถึง ได้เห็นถึงความแตกต่างบางส่วนของตัวกรองแบบ FIR และ IIR แล้ว ตัวกรองแบบ FIR และ IIR นี้ถือเป็นระบบพื้นฐานที่สำคัญมากในการประมวลผลสัญญาณดิจิตอล จึงเป็นเรื่องที่จำเป็นต้องศึกษาให้เข้าใจอย่างถ่องแท้ ในบทที่ 7 และ 8 นี้เราจะได้ศึกษาเพิ่มเติมเกี่ยวกับตัวกรองทั้งสองแบบ โดยเฉพาะอย่างยิ่งในเรื่องการหาค่าสัมประสิทธิ์ของตัวกรอง

สำหรับตัวกรองแบบ FIR นั้น ในการออกแบบสิ่งที่เราต้องการหา คือ ค่าของผลตอบสนองต่ออิมพัลส์ หรือ h(n) ของระบบ สำหรับตัวกรอง FIR ที่มี h(n) ยาว N จุด เรากล่าวว่า ตัวกรองนี้มีอันดับเท่ากับ N-1 เหตุผลก็คือ มีการใช้สัญญาณขาเข้าในอดีตย้อนหลังไป N-1 ตำแหน่ง หรือ ตัวกำลังสูงสุดที่อยู่ในฟังก์ชั่น H(z) ก็คือ $z^{-(N-1)}$

คุณสมบัติเฟสแบบเชิงเส้น

คุณสมบัติเฟสแบบเชิงเส้น (linear phase) คือ คุณสมบัติของระบบที่มีผลตอบสนองทางเฟสมีลักษณะเป็นเชิงเส้น คุณสมบัตินี้เป็นคุณสมบัติที่สำคัญมากของระบบ และเฉพาะตัวกรองแบบ FIR เท่านั้นที่สามารถมีคุณสมบัตินี้ได้อย่างสมบูรณ์ เราลองมาดูก่อนว่า การมีเฟสแบบเชิงเส้นของระบบหมายความว่าอย่างไร

จากที่เราได้ศึกษามาว่า ระบบสามารถมีผลตอบสนองเชิงความถี่ที่จัดให้อยู่ในรูปได้ ดังนี้

$$H(e^{j\omega'}) = A(e^{j\omega'})\, e^{j\theta(e^{j\omega'})} \qquad (7.1)$$

โดยที่ A คือ อัตราขยายหรือลดทอนของระบบ และ θ คือเฟสของสัญญาณขาออกที่เปลี่ยนไปจากสัญญาณขาเข้า โดยทั้งคู่มีค่าแปรตามความถี่ เรากล่าวว่าระบบมีเฟสเป็นแบบเชิงเส้นโดยสมบูรณ์ เมื่อ θ เป็นฟังก์ชั่นแบบเชิงเส้นของ ω' หรือเขียนได้เป็น

$$\theta = -a\omega' \qquad (7.2)$$

โดยที่ a เป็นค่าคงที่ที่ไม่แปรตามความถี่ นั่นหมายถึงว่า เฟสของสัญญาณขาออกมีการเปลี่ยนแปลงที่เป็นเชิงเส้นกับความถี่ของสัญญาณขาเข้า ปรากฏการณ์นี้จะทำให้สัญญาณขาออกมีความล้าหลังทางเฟส (phase delay) ที่คงที่ตลอดทุก ๆ ความถี่ ซึ่งความล้าหลังทางเฟสมีสมการ คือ

$$T_p = -\theta / \omega \qquad (7.3)$$

ในที่นี้จะได้ T_p คงที่เท่ากับ a การที่ระบบมีเฟสเชิงเส้นมีผลดี คือ ทำให้ไม่เกิดความผิดเพี้ยนทางเฟส หรือเรียกว่า phase distortion ความผิดเพี้ยนนี้มีผลเสียมากในงานหลาย ๆ อย่าง เช่น การสื่อสารข้อมูล, เสียงดนตรี, วีดีโอ, และ ชีวภาพการแพทย์เป็นต้น จึงมีความจำเป็นอย่างยิ่งที่ต้องพยายามทำให้ส่วนต่าง ๆ ในระบบ ไม่ว่าจะเป็นตัวกรอง, เครื่องขยาย/ลดทอน, และสายส่งสัญญาณ มีผลตอบสนอง เฟสที่เป็นเชิงเส้นที่สุดเท่าที่จะทำได้

ลองศึกษาดูว่า ระบบที่มีเฟสเชิงเส้นให้ผลอย่างไรกับสัญญาณจากโปรแกรมที่ 7.1 และรูปที่ 7.1 ซึ่งเป็นผลลัพธ์ของการทำงานของโปรแกรม สัญญาณขาเข้าของระบบนี้ คือสัญญาณที่ประกอบด้วยความถี่ดิจิตอล 3 ความถี่ คือ $\omega' = 0.1\pi, 0.2\pi$, และ 0.3π โดยมีรูปสัญญาณ คือ

$$x(n) = \sin(0.1\pi n) + \sin(0.2\pi n) + \sin(0.3\pi n)$$

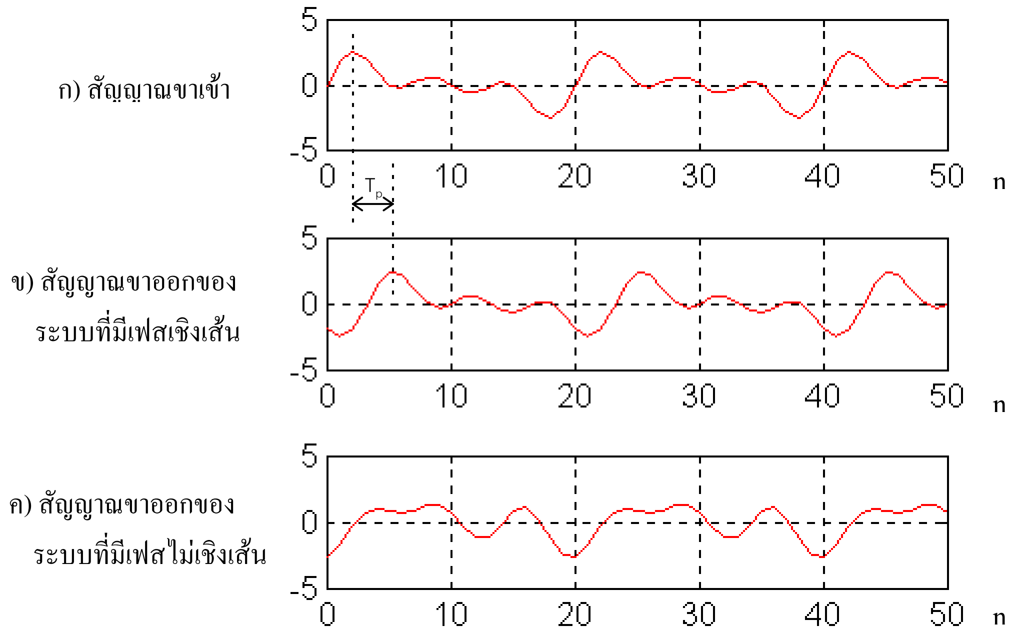
รูปที่ 7.1 ก) แสดงสัญญาณขาเข้านี้ เมื่อนำสัญญาณนี้ผ่านระบบที่ผ่านทุกความถี่ที่มีเฟสเป็นเชิงเส้น โดยมีผลตอบสนองต่อความถี่ คือ

$$H(e^{j\omega'}) = e^{-j10n/\pi} \quad \text{หรือมี} \quad \theta = -10\omega'/\pi$$

จะได้ว่า ที่สัญญาณขาออก องค์ประกอบความถี่ที่ $\omega' = 0.1\pi, 0.2\pi$, และ 0.3π จะมีเฟสล้าหลังเท่ากับ 1, 2, และ 3 เรเดียนตามลำดับ โดยรูปที่ 7.1 ข) แสดงรูปของสัญญาณขาออกนี้ และรูปที่ 7.1 ค) แสดงรูปของสัญญาณขาออกของอีกระบบหนึ่งซึ่งผ่านทุกความถี่เช่นเดียวกัน แต่มีเฟสที่ไม่เป็นเชิงเส้น ผลของความผิดเพี้ยนทางเฟสที่เกิดขึ้นในกรณีนี้สังเกตได้ง่ายมา คือ ระบบที่มี T_p คงที่ จะได้สัญญาณขาออกมีรูปร่างเหมือนสัญญาณขาเข้า แต่มีความล้าหลังไปเท่ากับ T_p จุด (T_p ไม่จำเป็นต้องเป็นจำนวนเต็ม) แต่สำหรับระบบที่มีเฟสไม่เชิงเส้นจะให้รูปร่างของสัญญาณขาออกผิดเพี้ยนไป

```
w1=0.1*pi; w2=0.2*pi; w3=0.3*pi;
n=0:50;
x=sin(w1*n)+sin(w2*n)+sin(w3*n);
d1=-1;   d2=-2;   d3=-3;
y=sin(w1*n+d1)+sin(w2*n+d2)+sin(w3*n+d3);
d1=-1;   d2=-2;   d3=-1;
y2=sin(w1*n+d1)+sin(w2*n+d2)+sin(w3*n+d3);
subplot(311); plot(n,x);  grid on
subplot(312); plot(n,y);  grid on
subplot(313); plot(n,y2); grid on
```

โปรแกรมที่ 7.1 *linearph.m สำหรับแสดงตัวอย่างของสัญญาณขาออกของระบบที่มีเฟสเชิงเส้น และไม่เชิงเส้น*



รูปที่ 7.1 *ผลลัพธ์ของโปรแกรมที่ 7.1*

ในงานบางอย่าง ไม่จำเป็นที่จะต้องมีการล้าหลังทางเฟส (T_p) คงที่ แต่ต้องการเพียงแค่มีการล้าหลังของกลุ่ม (group delay) คงที่ ซึ่งความล้าหลังของกลุ่มมีสูตร คือ

$$T_{group} = \frac{d\theta}{d\omega'} \qquad (7.4)$$

เงื่อนไขนี้เป็นเงื่อนไขที่เบากว่า และระบบใดที่มีการล้าหลังทางเฟสคงที่ก็จะมีการล้าหลังของกลุ่มคงที่ด้วย เรากล่าวว่า ระบบที่มีความล้าหลังของกลุ่มคงที่เป็นระบบที่มีเฟสเชิงเส้น ซึ่งจะได้สมการทั่วไปของผลตอบสนองทางเฟสของระบบที่มีเฟสเชิงเส้น คือ

$$\theta = -a\omega' + b \qquad (7.5)$$

โดยที่ a และ b เป็นค่าคงที่ที่ไม่แปรตามความถี่

คุณสมบัติความสมมาตรของตัวกรองที่มีเฟสเชิงเส้น

ตัวกรอง FIR ที่จะให้ผลตอบสนองเฟสที่เป็นเชิงเส้นตามสมการที่ 7.5 จะต้องมีเงื่อนไขสมมาตรสำหรับ h(n) หนึ่งในสี่ชนิด ดังต่อไปนี้

<u>ชนิดที่ 1</u> h(n) มีความสมมาตรปกติ (symmetric) และ N เป็นเลขคี่

ความสมมาตรนี้ เขียนเป็นเงื่อนไขของ h(n) ได้ดังนี้

$$h(n) = h(N-1-n), \quad n=0, 1, ..., N-1 \tag{7.6}$$

ตัวอย่างเช่น ถ้า N=7 เราจะได้ว่า h(0)=h(6), h(1)=h(5), h(2)=h(4), ส่วน h(3) ไม่มีคู่สมมาตร ถ้านิยามให้ M = (N-1)/2 ด้วยเงื่อนไขนี้จะสามารถพิสูจน์ได้ว่า ผลตอบสนองเชิงความถี่ของระบบจะอยู่ในรูปของ

$$H(e^{j\omega'}) = H_r(\omega')e^{-j\omega' M} \tag{7.7}$$

โดยที่ $H_r(\omega') = h(M) + 2\sum_{n=0}^{M-1} h(n)\cos[\omega(M-n)]$

สังเกตว่า $H_r(\omega')$ คือส่วนของขนาดซึ่งเป็นค่าจริงเสมอ ส่วน $-\omega'(N-1)/2$ คือส่วนของเฟสซึ่งเป็นเชิงเส้นตามรูปแบบสมการที่ 7.2 ตัวกรอง FIR ชนิดที่ 1 นี้ไม่มีข้อจำกัดของผลตอบสนองทางขนาดเหมือนอีก 3 ชนิด จึงใช้ออกแบบตัวกรองได้ทุกรูปแบบ และเป็นชนิดที่เราจะใช้มากที่สุดในส่วนต่อ ๆ ไป

<u>ชนิดที่ 2</u> h(n) มีความสมมาตรปกติ (symmetric) และ N เป็นเลขคู่

มีสมการเงื่อนไขเช่นเดียวกับชนิดที่ 1 คือ h(n) = h(N-1-n) ตัวอย่างเช่น ถ้า N=8 เราจะได้ว่า h(0)=h(7), h(1)=h(6), h(2)=h(5), และ h(3)=h(4)

ด้วยเงื่อนไขนี้ จะสามารถพิสูจน์ได้ว่า ผลตอบสนองเชิงความถี่จะอยู่ในรูปของ

$$H(e^{j\omega'}) = H_r(\omega')e^{-j\omega'(M+0.5)} \tag{7.8}$$

โดยที่ $H_r(\omega') = 2\sum_{n=0}^{M-1} h(n)\cos[\omega(M+0.5-n)]$

พบว่าผลตอบสนองทางขนาดของชนิดนี้มีข้อจำกัดว่า ที่ $\omega'=\pi$ จะได้ $H_r(\omega')=0$ เสมอ ดังนั้นจึงไม่เหมาะสำหรับออกแบบตัวกรองผ่านสูง และตัวกรองผ่านตัดความถี่

<u>ชนิดที่ 3</u> h(n) มีความสมมาตรแบบตรงข้าม (antisymmetric) และ N เป็นเลขคี่

ความสมมาตรนี้ เขียนเป็นเงื่อนไขของ h(n) ได้ดังนี้

$$h(n) = -h(N-1-n),\ \ n=0, 1, ..., N-1 \tag{7.9}$$

ด้วยเงื่อนไขนี้ จะสามารถพิสูจน์ได้ว่า ผลตอบสนองเชิงความถี่จะอยู่ในรูปของ

$$H(e^{j\omega'}) = H_r(\omega')e^{j(-\omega' M+\pi/2)} \tag{7.10}$$

โดยที่ $H_r(\omega') = 2\sum_{n=0}^{M-1} h(n)\sin[\omega(M-n)]$

พบว่าผลตอบสนองทางขนาดของชนิดนี้มีข้อจำกัดว่า ที่ $\omega'=0$ หรือ π จะได้ $H_r(\omega')=0$ เสมอ ดังนั้นจึงไม่เหมาะสำหรับออกแบบตัวกรองผ่านต่ำ, ผ่านสูง, และตัวกรองตัดแถบความถี่

<u>ชนิดที่ 4</u> h(n) มีความสมมาตรแบบตรงข้าม (antisymmetric) และ N เป็นเลขคู่

มีสมการเงื่อนไขของความสมมาตรเช่นเดียวกับชนิดที่ 3 คือ h(n) = - h(N-1-n)

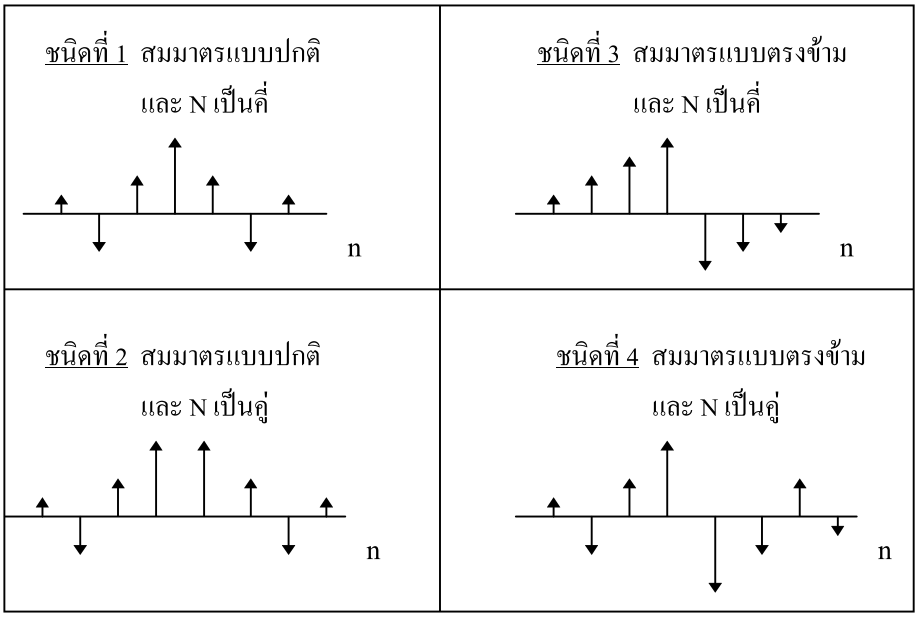
ด้วยเงื่อนไขนี้ จะสามารถพิสูจน์ได้ว่า ผลตอบสนองเชิงความถี่จะอยู่ในรูปของ

$$H(e^{j\omega'}) = H_r(\omega')e^{j(-\omega'(M+0.5)+\pi/2)} \tag{7.11}$$

โดยที่ $H_r(\omega') = 2\sum_{n=0}^{M-1} h(n)\sin[\omega(M+0.5-n)]$

พบว่าผลตอบสนองทางขนาดของชนิดนี้มีข้อจำกัดว่า ที่ $\omega'=0$ จะได้ $H_r(\omega')=0$ เสมอ ดังนั้นจึงไม่เหมาะสำหรับออกแบบตัวกรองผ่านต่ำ, และตัวกรองตัดแถบความถี่

ตัวกรอง FIR ที่มีความสมมาตรชนิดที่ 3 และ 4 เหมาะสำหรับออกแบบตัวกรองอนุพันธ์ (differentiator) และตัวกรอง Hilbert เนื่องจากมีการกลับเฟส $\pi/2$ หรือ 90 องศาอยู่ในผลตอบสนองทางเฟสด้วย



รูปที่ 7.2 *ผลตอบสนองต่ออิมพัลส์ของตัวกรอง FIR ที่มีเฟสเชิงเส้นทั้ง 4 แบบ*

<u>ตัวอย่างที่ 7.1</u> จงใช้สมการเงื่อนไขของความสมมาตร พิสูจน์สูตรของผลตอบสนองเชิงความถี่ของตัวกรอง FIR ชนิดที่ 1 (สมการที่ 7.7)

สมมติว่าตัวกรองมีความยาวของ h(n) เท่ากับ N จุด โดย N = 2M+1 (ทั้ง N,M เป็นจำนวนเต็ม) จะได้ผลตอบสนองเชิงความถี่ของตัวกรอง คือ

$$
\begin{aligned}
H(e^{j\omega'}) &= \sum_{n=0}^{N-1} h(n)\, e^{-j\omega' n} \\
&= \sum_{n=0}^{2M} h(n)\, e^{-j\omega' n} \\
&= e^{-j\omega' M} \sum_{n=0}^{2M} h(n)\, e^{j\omega'(M-n)}
\end{aligned}
$$

ทำการแตกให้เป็นสามเทอมย่อยบวกกัน คือ

$$
H(e^{j\omega'}) = e^{-j\omega' M} \left\{ h(M) + \sum_{n=0}^{M-1} h(n)\, e^{j\omega'(M-n)} + \sum_{n=M+1}^{2M} h(n)\, e^{j\omega'(M-n)} \right\}
$$

เปลี่ยนตัวชี้ของเทอมที่ 3 โดยให้มีตัวชี้ใหม่ คือ k ซึ่ง n = N-1-k จะได้

$$H(e^{j\omega'}) = e^{-j\omega' M}\left\{h(M) + \sum_{n=0}^{M-1} h(n)\, e^{j\omega'(M-n)} + \sum_{k=M-1}^{0} h(N-k-1)\, e^{-j\omega'(M-k)}\right\}$$

เปลี่ยนตัวชี้ k ของเทอมที่สามกลับเป็น n (โดยคราวนี้ให้ n=k) และใช้คุณสมบัติความสมมาตรของ h(n) คือ h(n) = h(N-n-1) แทนลงไป จะได้

$$\begin{aligned} H(e^{j\omega'}) &= e^{-j\omega' M}\left\{h(M) + \sum_{n=0}^{M-1} h(n)\, e^{j\omega'(M-n)} + \sum_{n=0}^{M-1} h(N-n-1)\, e^{-j\omega'(M-n)}\right\} \\ &= e^{-j\omega' M}\left\{h(M) + \sum_{n=0}^{M-1} h(n)\, e^{j\omega'(M-n)} + \sum_{n=0}^{M-1} h(n)\, e^{-j\omega'(M-n)}\right\} \end{aligned}$$

รวมสองเทอมหลังเข้าด้วยกันจะได้

$$H(e^{j\omega'}) = e^{-j\omega' M}\left\{h(M) + 2\sum_{n=0}^{M-1} h(n) \cos\left[\omega(M-n)\right]\right\}$$

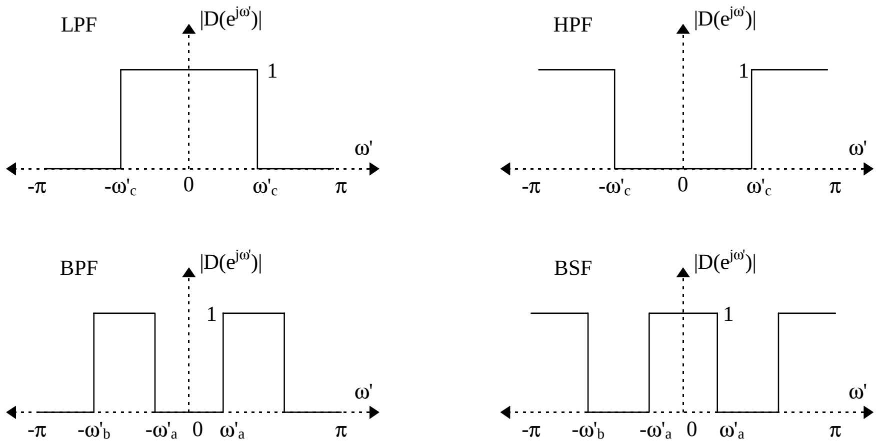
เราได้ผลลัพธ์สุดท้ายนี้เป็นไปดังสมการที่ 7.7 ตามที่ต้องการพิสูจน์

การออกแบบโดยวิธีหน้าต่าง (Window Method)

ในส่วนนี้จะได้กล่าวถึง การหาค่าสัมประสิทธิ์ของตัวกรอง FIR เมื่อกำหนดคุณลักษณะเฉพาะของตัวกรองมา ซึ่งคุณลักษณะเฉพาะนี้ ส่วนใหญ่จะเป็นการกำหนดลักษณะของผลตอบสนองเชิงความถี่ที่ต้องการ ได้แก่ ความถี่ตัด, ความคมของตัวกรอง, การลดทอนในแถบหยุด และอื่น ๆ เราจะทำการออกแบบโดยคำนึงถึงความถี่ดิจิตอล ω' ที่มีย่านความถี่ที่สนใจในช่วง $-\pi$ ถึง π หรือ f' ในช่วง -1 ถึง 1 ดังที่เราได้ศึกษามาในบทที่ 5

วิธีหน้าต่างเป็นวิธีพื้นฐานที่สุดวิธีหนึ่งที่จะใช้หาสัมประสิทธ์ของตัวกรอง เป็นวิธีที่ง่ายต่อการออกแบบ และสามารถใช้ออกแบบตัวกรองแบบต่าง ๆ ได้ ไม่ว่าจะเป็นแบบผ่านต่ำ (LPF), ผ่านสูง (HPF), ผ่านแถบความถี่ (BPF), หรือตัดแถบความถี่ (BSF)

เราจะเริ่มออกแบบโดยใช้ต้นแบบจากตัวกรองอุดมคติ ผลตอบสนองเชิงความถี่ของตัวกรองอุดมคติทั้งสี่แบบแสดงอยู่ในรูปที่ 7.3 โดยมีความถี่ตัด (cutoff frequency) เท่ากับ ω'_c สำหรับแบบผ่านต่ำ และผ่านสูง ส่วนแบบผ่านแถบความถี่ และตัดแถบความถี่มีความถี่ตัดของแถบความถี่ที่ ω'_a และ ω'_b ดังแสดงในรูป



***รูปที่ 7.3** ผลตอบสนองเชิงความถี่ของตัวกรองอุดมคติ*

สมมติให้ d(n) แทนผลตอบสนองต่ออิมพัลส์ และ $D(e^{j\omega'})$ แทนผลตอบสนองเชิงความถี่สำหรับตัวกรองแบบผ่านต่ำอุดมคติ ดังในรูปที่ 7.3 เราจะหาผลตอบสนองต่ออิมพัลส์ของตัวกรองอุดมคติเหล่านี้ได้โดยการแปลง IDTFT ดังนี้

$$d(n) = \mathrm{IDTFT}\{\,D\,\}$$

$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} D(e^{j\omega'})\,e^{j\omega' n}\,d\omega'$$

$$= \frac{1}{2\pi}\int_{-\omega'_C}^{\omega'_C} (1)e^{j\omega' n}\,d\omega'$$

$$= \left[\frac{e^{j\omega' n}}{2\pi jn}\right]_{-\omega'_C}^{\omega'_C} = \frac{e^{j\omega'_C n} - e^{-j\omega'_C n}}{2\pi jn}$$

$$d(n) = \frac{\sin(\omega'_C n)}{\pi n}\,,\ -\infty < n < \infty \qquad (7.12)$$

สมการนี้มีปัญหาที่ n=0 เพราะจะได้ว่า d(n) มีค่าเป็นเศษศูนย์ส่วนศูนย์ ซึ่งเราสามารถหาค่า d(0) ได้โดยใช้ทฤษฎีบทของโลปิตัล จะได้ว่า

$$d(0) = \frac{\lim_{n\to 0} \dfrac{d(\sin(\omega'_C n))}{dn}}{\lim_{n\to 0} \dfrac{d(\pi n)}{dn}} = \frac{\omega'_C}{\pi} \tag{7.13}$$

ในทำนองเดียวกันเราสามารถหา d(n) สำหรับตัวกรองแบบอื่น ๆ ได้โดยใช้ IDTFT กระทำกับผลตอบสนองเชิงความถี่ของตัวกรองนั้น ๆ ตารางที่ 7.1 ได้สรุปค่าของ d(n) สำหรับตัวกรองแบบต่าง ๆ ไว้ เพื่อใช้อ้างอิงในการออกแบบต่อไป

ชนิดของตัวกรอง	$d(n), -\infty < n < \infty$	$d(0)$
ผ่านต่ำ (LPF)	$\dfrac{\sin(\omega'_C n)}{\pi n}$	$\dfrac{\omega'_C}{\pi}$
ผ่านสูง (HPF)	$\delta(n) - \dfrac{\sin(\omega'_C n)}{\pi n}$	$1 - \dfrac{\omega'_C}{\pi}$
ผ่านแถบความถี่ (BPF)	$\dfrac{\sin(\omega'_b n) - \sin(\omega'_a n)}{\pi n}$	$\dfrac{\omega'_b}{\pi} - \dfrac{\omega'_a}{\pi}$
ตัดแถบความถี่ (BSF)	$\delta(n) - \dfrac{\sin(\omega'_b n) - \sin(\omega'_a n)}{\pi n}$	$1 - \dfrac{\omega'_b}{\pi} + \dfrac{\omega'_a}{\pi}$

ตารางที่ 7.1 *ผลตอบสนองต่ออิมพัลส์ของตัวกรองอุดมคติต่าง ๆ*

เราอาจหา d(n) ของตัวกรองแบบอื่น ๆ ได้จากตัวกรอง LPF และตัวกรองแบบผ่านทุกความถี่ ตัวกรองแบบผ่านทุกความถี่ในที่นี้ คือ $D(e^{j\omega'})=1$ หรือ $d(n)=\delta(n)$ ตัวอย่างเช่น ถ้าต้องการหา d(n) ของ HPF จาก d(n) ของ LPF เราสามารถใช้ความจริงที่ว่า ผลตอบสนองเชิงความถี่ของ LPF บวกกับผลตอบสนองเชิงความถี่ของ HPF ที่มีความถี่ตัดตรงกัน จะได้เป็นตัวกรองผ่านทุกความถี่ ดังนี้

$$D_{LP}(e^{j\omega'}) + D_{HP}(e^{j\omega'}) = 1 \tag{7.14}$$

เมื่อแปลง z ทั้งสมการจะได้ความสัมพันธ์ของ d(n) เป็น

$$d_{LP}(n) + d_{HP}(n) = \delta(n) \tag{7.15}$$

ยังมีความสัมพันธ์อื่นในลักษณะเดียวกันนี้อีก คือ ผลตอบสนองเชิงความถี่ของ BPF บวกกับผลตอบสนองเชิงความถี่ของ BSF ที่มีความถี่ตัดตรงกัน จะได้เป็นตัวกรองผ่านทุกความถี่ ดังนี้

$$D_{BP}(e^{j\omega'}) + D_{BS}(e^{j\omega'}) = 1 \qquad (7.16)$$

แปลง z ผกผันได้

$$d_{BP}(n) + d_{BS}(n) = \delta(n) \qquad (7.17)$$

และผลตอบสนองเชิงความถี่ของ BPF ก็สามารถหาได้จากการนำเอาผลตอบสนองเชิงความถี่ของ LPF ที่มีความถี่ตัดเท่ากับความถี่ตัดด้านต่ำของ BPF ลบออกจากผลตอบสนองเชิงความถี่ของ LPF ที่มีความถี่ตัดเท่ากับความถี่ตัดด้านสูงของ BPF ดังนี้

$$D_{BP}(e^{j\omega'}) = D_{LP}(e^{j\omega'})\Big|_{\omega'_c=\omega'_b} - D_{LP}(e^{j\omega'})\Big|_{\omega'_c=\omega'_a} \qquad (7.18)$$

แปลง z ผกผันได้

$$d_{BP}(n) = d_{LP}(n)\Big|_{\omega'_c=\omega'_b} - d_{LP}(n)\Big|_{\omega'_c=\omega'_a} \qquad (7.19)$$

เราอยากจะใช้ค่า d(n) ในตารางที่ 7.1 เพื่อเป็นผลตอบสนองต่ออิมพัลส์ของตัวกรองที่ต้องการ แต่อย่างไรก็ตาม d(n) ที่ได้นี้ไม่สามารถสร้างได้จริงในทางปฏิบัติ เนื่องจาก มีความยาวไม่จำกัด และเป็นแบบสองด้าน กล่าวคือ d(n) มีค่าตั้งแต่ n เป็น $-\infty$ จนถึง ∞ เราจะต้องใช้เทคนิคบางอย่างเพื่อให้นำ d(n) เหล่านี้มาใช้ได้ ซึ่งเทคนิคนี้ก็คือ วิธีหน้าต่างนั่นเอง

ก่อนอื่นขอให้ทำความรู้จักกับผลตอบสนองเชิงความถี่ที่เราจะออกแบบได้ก่อน ซึ่งมีรูปร่างดังในรูปที่ 7.3 ค่าที่จะใช้กำหนดเป็นคุณลักษณะเฉพาะของวิธีหน้าต่าง ได้แก่

- **ความพลิ้วของแถบผ่าน** (pass-band ripple, δ_{pass}) คือ ค่าสูงสุดที่ขนาดของแถบผ่านแกว่งออกห่างจากค่า 1 บางครั้งวัดเป็น dB โดยใช้

$$A_{pass} = 20\log\frac{1+\delta_{pass}}{1-\delta_{pass}} \ \text{(dB)} \qquad (7.20)$$

- **การลดทอนของแถบหยุด** (stop-band attenuation, A_{stop}) คือ จำนวนเท่าที่แถบหยุดลดทอนลงจากค่า 1 วัดเป็น dB โดยมีความสัมพันธ์กับความพลิ้วของแถบหยุด คือ

$$A_{stop} = -20\log\delta_{stop} \ \text{(dB)} \qquad (7.21)$$

- **ความกว้างของแถบเปลี่ยน** (transition band width, $\Delta f'$)

- **ความถี่ตัด** (cutoff frequency, f_c) คือ ค่าความถี่ที่ขนาดลดลงประมาณ 0.5 หรืออยู่ที่ประมาณครึ่งหนึ่งของแถบเปลี่ยน นิยามนี้ต่างจากความถี่ตัดของตัวกรองแอนะลอก และตัวกรอง IIR ซึ่งความถี่ตัดหมายถึง ความถี่ที่ลดทอนลงเท่ากับ 3 dB

การออกแบบโดยวิธีหน้าต่าง สามารถแบ่งเป็นขั้นตอนย่อยได้ดังนี้

1. ใช้ค่าความพลิ้วของแถบผ่าน หรือการลดทอนของแถบหยุดอย่างใดอย่างหนึ่งเพื่อเลือกชนิดของหน้าต่างที่สามารถใช้ได้จากตารางที่ 7.2 ถ้ามีข้อกำหนดทั้งสองอย่าง ให้เปลี่ยน A_{stop} เป็น δ_{stop} ก่อน แล้วเปรียบเทียบ δ_{pass} กับ δ_{stop} ว่าค่าใดน้อยกว่ากัน ถ้า δ_{pass} น้อยกว่าให้ใช้ δ_{pass} เป็นตัวเลือกหน้าต่าง แต่ถ้า δ_{stop} น้อยกว่าให้ใช้ A_{stop} เป็นตัวเลือกหน้าต่าง โดยมีหลักการว่า δ_{pass} ในตารางต้องน้อยกว่าที่ต้องการ และ A_{stop} ในตารางต้องมากกว่าที่ต้องการ

2. ใช้ค่าความกว้างของแถบเปลี่ยนหาค่าอันดับของตัวกรองที่ต้องใช้ โดยใช้ความสัมพันธ์ระหว่าง $\Delta f'$ กับ N ที่แสดงไว้ในตารางที่ 7.2 จากนั้น คำนวณฟังก์ชั่นหน้าต่างที่ต้องใช้ จะได้

$$\mathbf{w(n)},\ \text{ที่ } n = 0, 1, ..., N-1$$

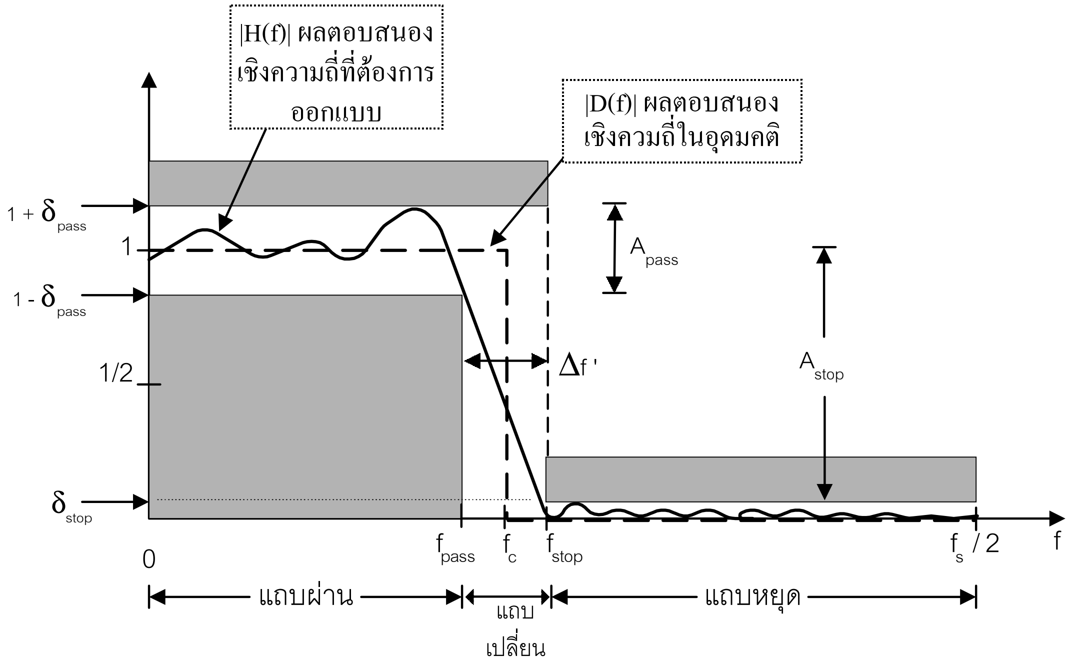
จากตารางจะสังเกตได้ว่า อันดับของตัวกรองจะเป็นสัดส่วนผกผันกับ $\Delta f'$ นั่นคือ เราสามารถปรับปรุง $\Delta f'$ ให้แคบลงได้ (ตัวกรองคมขึ้น) โดยการเพิ่มค่า N แต่เราไม่สามารถปรับปรุงค่า δ_{stop} และ A_{pass} ให้ดีขึ้นได้ เพราะค่า δ_{stop} และ A_{pass} จะมีขนาดคงที่สำหรับหน้าต่างแบบหนึ่ง ๆ ยกเว้นหน้าต่างแบบ Kaiser

3. ใช้ชนิดของตัวกรอง (LPF, HPF, …) และความถี่ตัดที่ต้องการ เลือกผลตอบสนองต่ออิมพัลส์ d(n) ที่ถูกต้องจากตารางที่ 7.1

4. เลื่อน d(n) ให้ล้าหลังลง M ตำแหน่ง โดย M=(N-1)/2 จะได้สัญญาณเป็น d(n-M) จากนั้นคูณเข้ากับฟังก์ชันหน้าต่าง w(n) ที่ได้จากข้อ 2 ซึ่งจะได้เป็นผลตอบสนองต่ออิมพัลส์ที่มีความยาว N จุด และเป็นแบบคอซัล ดังนี้

$$h(n) = d(n-M)w(n)\ ,\ n = 0, 1, ..., N-1 \qquad (7.22)$$

การออกแบบนี้จะต้องใช้ N เป็นจำนวนคี่ ซึ่งจะได้เป็นตัวกรอง FIR ที่มีเฟสเชิงเส้น และมีสมมาตรชนิดที่ 1 เพราะทั้ง d(n) และ w(n) มีสมมาตรรอบจุดกึ่งกลางทั้งคู่ สำหรับการออกแบบโดยที่ N เป็นจำนวนคู่ก็ทำได้เช่นกัน แต่ขอละไว้ไม่กล่าวถึงในที่นี้



รูปที่ 7.4 *ค่าต่าง ๆ ในการระบุคุณลักษณะเฉพาะของผลตอบสนองเชิงความถี่ของตัวกรอง FIR*

หน้าต่าง	δ_{pass}	A_{stop} $-20\log\delta_{stop}$ (dB)	$\Delta f'$ (normalized)	w(n), n=0,1, .. N-1 $\left(M=\dfrac{N-1}{2}\right)$
สี่เหลี่ยม (rectangular)	8.9%	21	2/N	1
ฮานนิ่ง (Hanning)	0.63%	44	4/N	$0.5-0.5\cos(\dfrac{2\pi n}{N-1})$
แฮมมิ่ง (Hamming)	0.22%	53	4/N	$0.54-0.46\cos(\dfrac{2\pi n}{N-1})$
แบล็กแมน (Blackman)	0.02%	74	6/N	$0.42-0.5\cos(\dfrac{2\pi n}{N-1})+0.08\cos(\dfrac{4\pi n}{N-1})$
ไคเซอร์ (Kaiser)	ปรับได้	ปรับได้	$\dfrac{A-7.95}{14.36(N-1)}$	$\dfrac{I_0(\alpha\sqrt{1-(n-M)^2/M^2})}{I_0(\alpha)}$

ตารางที่ 7.2 *หน้าต่างแบบต่าง ๆ และค่าพารามิเตอร์ที่สำคัญ* [1], [3]

หน้าต่างแบบสี่เหลี่ยม (Rectangular Window)

<u>ตัวอย่างที่ 7.2</u> จงออกแบบตัวกรองแบบ FIR LPF ที่มีความถี่ตัดที่ 500Hz โดยใช้หน้าต่างแบบสี่เหลี่ยม และให้มีความกว้างของแถบเปลี่ยน (Δf) น้อยกว่า 90 Hz ระบบนี้ใช้ความถี่ในการสุ่ม f_s = 2kHz

หาค่าความถี่ที่ต้องใช้ในหน่วยของความถี่ดิจิตอลก่อน โดยหารด้วย f_s จะได้

$\omega'_c = 2\pi f_c/f_s = 2\pi(500)/2000 = \pi/2$ เรเดียน

$\Delta f' = \Delta f/f_s = 90/2000 = 0.045$

จากตาราง 7.2 $\Delta f' = 2/N$ จะได้ $N = 2/0.045 = 44.4$

นั่นคือ ต้องใช้ N **มากกว่า 44.4** จึงจะได้ Δf ตามที่กำหนด เราเลือก N=45 (ให้ใช้เลขคี่)

และ จะได้ $M = (N-1)/2 = 22$

จากตารางที่ 7.1 ใช้ d(n) สำหรับ LPF คือ $d(n) = \dfrac{\sin(\omega'_C n)}{\pi n}$ เลื่อน d(n) ให้ล้าหลังไป M ตำแหน่ง จะได้ d(n-M) คือ

$$d(n-M) = \frac{\sin(\omega'_C(n-M))}{\pi(n-M)}$$

สำหรับหน้าต่างแบบสี่เหลี่ยม เราจะได้ w(n) = 1, n=0, 1, .. N-1 ดังนั้น หา h(n) ได้โดยใช้สมการที่ 7.22 คือ

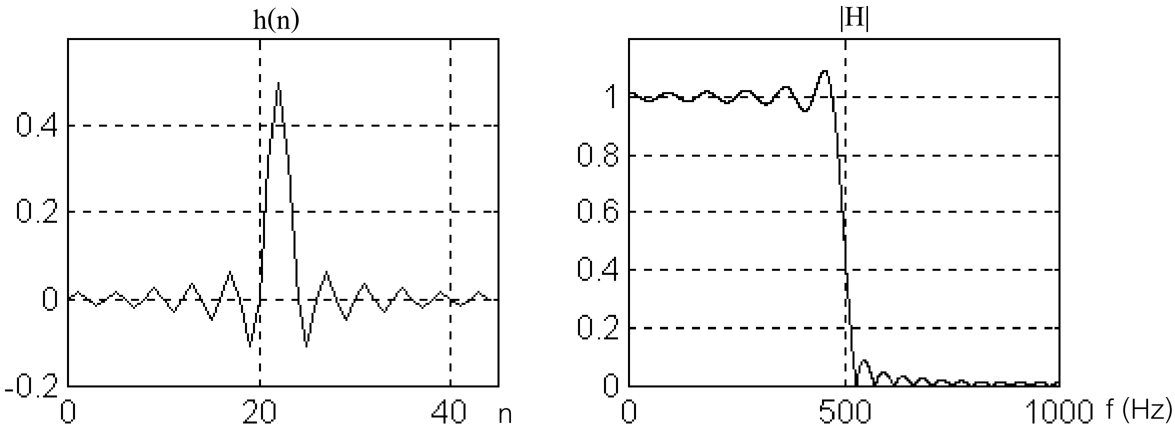
$$\begin{aligned} h(n) &= d(n-M)\,w(n) \\ &= \frac{\sin(\omega'_C(n-M))}{\pi(n-M)}, \quad n=0, 1, \ldots, N-1 \\ &= \frac{\sin(\pi/2\,(n-22))}{\pi(n-22)}, \quad n=0, 1, \ldots, 44 \end{aligned}$$

สำหรับผลตอบสนองเชิงความถี่ของระบบ หาได้โดยการหาสเปกตรัมของสัญญาณ h(n) ซึ่งก็คือ $H(e^{j\omega'})$ = DTFT(h(n)) ซึ่งสามารถประมาณได้โดยการใช้ FFT ซึ่งจะได้ H(k) = FFT(h(n)) ก็ได้ ในรูปที่ 7.5 เราใช้โปรแกรม freqres ที่ได้เขียนไว้ในบทที่ 5 หา |H| และวาดออกมาในช่วงความถี่ที่สนใจ คือ $f = (0, f_s/2)$ ดังรูปที่ 7.5 ก)

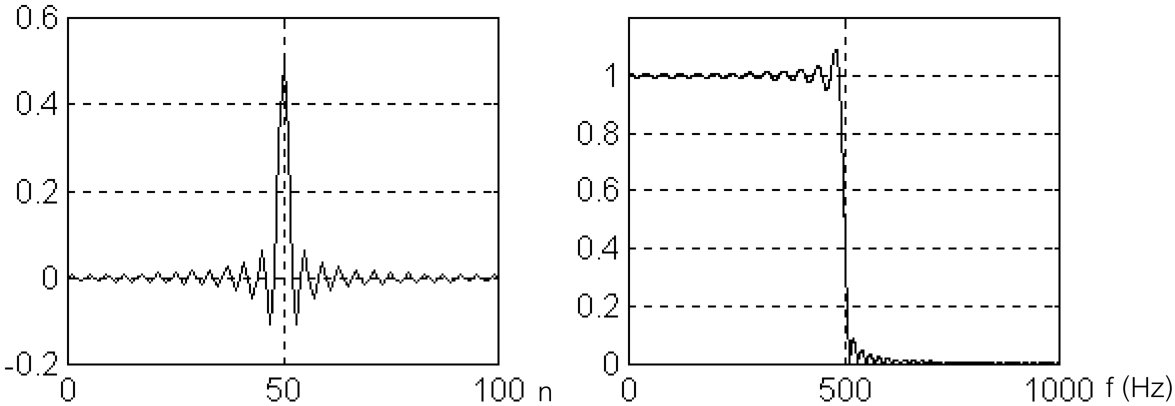
จะเห็นได้ว่าผลตอบสนองเชิงความถี่มีลักษณะเปลี่ยนไปจากอุดมคติ โดยมีย่านแถบเปลี่ยนเกิดขึ้น และมีการแกว่งขึ้นลง หรือ ความพลิ้วในช่วงแถบผ่าน และแถบหยุด ปรากฏการณ์ที่เกิดความพลิ้วขึ้นนี้ เรียกว่า ปรากฏการณ์ของ Gibbs

คราวนี้ลองศึกษาผลของการเพิ่มอันดับดูหน่อย ซึ่งเมื่อเราเพิ่ม N มากขึ้น ๆ ก็ควรจะได้ผลตอบสนองเชิงความถี่เข้าใกล้อุดมคติมากขึ้น ๆ เช่นกัน ลองใช้ค่า N เป็น 101 และวาด h(n) และ |H| ที่ได้ในรูปที่ 7.5 เมื่อเปรียบเทียบกับผลที่ได้ของกรณี N=45 จะเห็นได้ว่า แถบเปลี่ยนมีการบีบตัวแคบลง และความพลิ้วของแถบผ่านและแถบหยุดก็มีการบีบตัวแคบลง แต่ปรากฏว่าความสูงของความพลิ้วมีขนาดค่อนข้างคงที่ ซึ่งจากตารางที่ 7.2 เราจะได้ว่า $\delta_{pass} \approx \delta_{stop} \approx 0.089$ สำหรับหน้าต่างแฮมมิ่ง ซึ่งพบว่า หน้าต่างทุกชนิดจะให้ δ มีค่าค่อนข้างคงที่โดยไม่ขึ้นกับ N นี้ ถ้าเราต้องการ δ ที่เล็กลง ก็จำเป็นต้องเปลี่ยนชนิดของหน้าต่างที่ใช้

หลังจากเพิ่มค่า N จะเห็นได้ว่า Δf มีขนาดแคบลง ซึ่งเมื่อคำนวณหาค่า Δf จากสูตรในตาราง จะได้ $\Delta f = \Delta f' f_s = 2f_s/N = 2(2000)/101 = 40$ Hz ซึ่งใกล้เคียงกับค่าที่ได้ในรูป



ก) N=45



ข) N=101

***รูปที่ 7.5** ผลลัพธ์ที่ได้จากตัวอย่างที่ 7.2 (ใช้หน้าต่างแบบสี่เหลี่ยม)*

โปรแกรมที่ 7.2 เป็นโปรแกรมที่ใช้คำนวณ h(n) และวาดผลตอบสนองเชิงความถี่โดยใช้ Matlab เมื่อต้องการใช้กับตัวกรองอื่นที่ไม่ใช่ LPF หรือใช้กับหน้าต่างอื่น ก็สามารถแก้ตัวแปร d และ w ได้โดยง่าย สำหรับผู้ที่มี DSP Toolbox ก็อาจใช้ฟังก์ชันสำเร็จรูป ชื่อ fir1 สำหรับคำนวณ h(n) ได้

```
fc=500; fs=2000;                 กำหนด N = ความยาวของ h(n),
N=45;                                    M = ตำแหน่งกึ่งกลางของ h(n)
M=(N-1)/2;                       ωc = 2πfc/fs = ความถี่ตัดเชิงมุม
wc=2*pi*fc/fs;

n=0:N-1;                                 กำหนด n เป็นเวคเตอร์ที่มีค่าจาก 0 ถึง N-1
d= sin(wc*(n-M))./(n-M)/pi;              คำนวณค่า d(n-M) โดยใช้สูตรจากตารางที่ 7.1
d((N+1)/2) = wc/pi;                      (ในที่นี้คือ LPF)

w=1;                         คำนวณค่า w(n) โดยใช้สูตรจากตารางที่ 7.2
h=d.*w;                          (ในที่นี้คือหน้าต่างแบบสี่เหลี่ยม) จากนั้นหา h(n) = d(n-M) w(n)
freqres(h, 1, fs)            วาดผลตอบสนองเชิงความถี่ที่ได้
```

***โปรแกรมที่ 7.2** window.m สำหรับออกแบบตัวกรอง FIR โดยวิธีหน้าต่าง*

หน้าต่างแฮมมิ่ง (Hamming Window)

<u>ตัวอย่างที่ 7.3</u> ออกแบบตัวกรอง FIR โดยใช้หน้าต่างแฮมมิ่ง ให้ใช้ N=101 เปรียบเทียบรูป และลักษณะของผลตอบสนองเชิงความถี่ที่ได้กับผลในตัวอย่างที่ 7.2

เช่นเดียวกัน เราใช้ d(n) สำหรับ LPF จากตารางที่ 7.1 โดยเลื่อนให้ล้าหลังไป M จุด จะได้

$$d(n\text{-}M) = \frac{\sin(\omega_c'(n-M))}{\pi(n-M)}$$

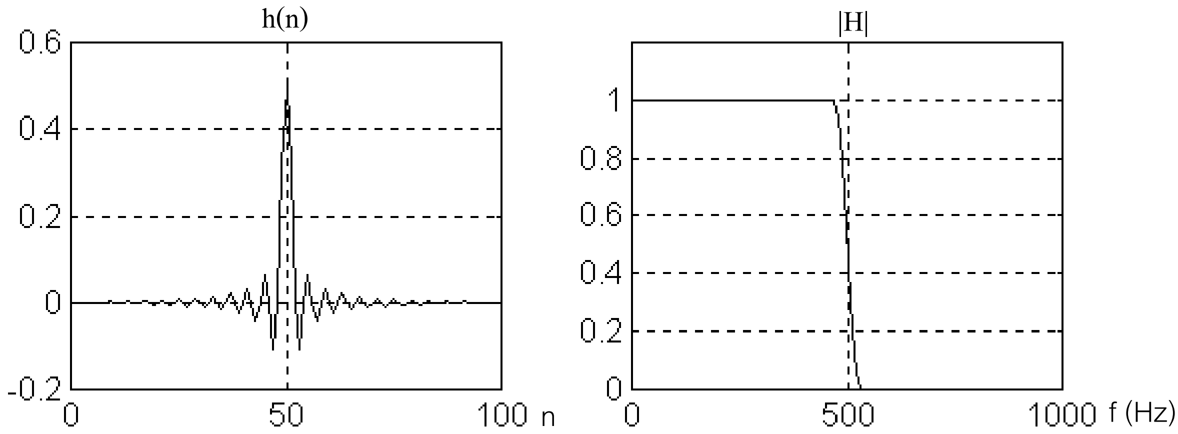
โดยที่ M = (N-1)/2 = 50

จากตารางที่ 7.2 ใช้ w(n) สำหรับหน้าต่างแฮมมิ่งจะได้

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right),\ n=0, 1, .. 100$$

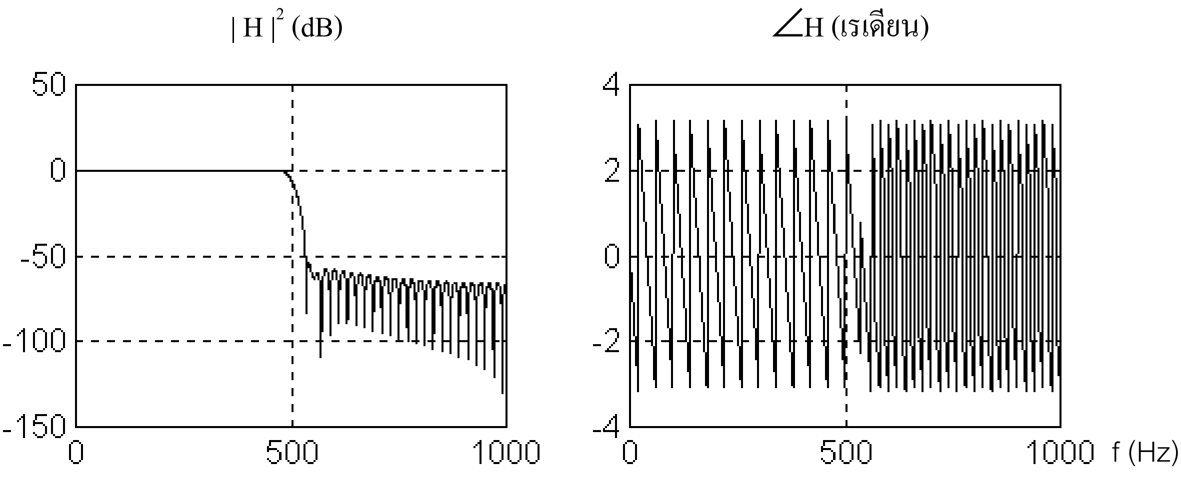
เราจะได้ h(n) = d(n-M)w(n), n=0, 1, .. 100 แทนค่า N และ w(n) ลงในโปรแกรมที่ 7.2 จะได้ $H(e^{j\omega'})$ ดังในรูปที่ 7.6

ลองคำนวณค่าความกว้างของแถบเปลี่ยนจากตารางที่ 7.2 เปรียบเทียบกับในรูปดู จะได้ $\Delta f' = 4/N = 4/101 = 0.0396$ ดังนั้น $\Delta f = \Delta f' f_s = 0.0396(2000) = 79$ Hz ซึ่งประมาณใกล้เคียงกับรูปที่ได้ และ**กว้างกว่า**กรณีหน้าต่างแบบสี่เหลี่ยมที่ N เท่ากัน ประมาณ 2 เท่า



รูปที่ 7.6 *ผลลัพธ์ที่ได้จากการตัวอย่างที่ 7.3 (ใช้หน้าต่างแฮมมิ่ง N=101)*

จะเห็นได้ว่าหน้าต่างแฮมมิ่งให้ Δf ที่กว้างกว่าในกรณีหน้าต่างแบบสี่เหลี่ยม แต่ขณะเดียวกันก็ให้ความพลิ้ว δ ที่เล็กลงมาก จากตารางที่ 7.2 เราพบว่า $\delta_{pass} \approx \delta_{stop} \approx 0.002$ ซึ่งก็แทบมองไม่เห็นจากรูปที่ 7.6 โดยมากมันนิยมวาดรูปของผลตอบสนองเชิงความถี่ในหน่วย dB (20 คูณ log ของ |H|) เพื่อให้มองเห็นอัตราการลดทอนได้ชัดเจนขึ้น รูปที่ 7.7 แสดงผลตอบสนองทางขนาดในหน่วย dB และผลตอบสนองทางเฟสที่ได้จากตัวอย่างที่ 7.3 นี้



รูปที่ 7.7 *ขนาด(dB) และเฟสของผลตอบสนองเชิงความถี่ที่ได้จากตัวอย่างที่ 7.3*

ถ้าพิจารณาถึงความเป็นเชิงเส้นของระบบ จากรูปของเฟสในรูปที่ 7.7 มองดูไม่เหมือนสภาพที่เป็นเชิงเส้น (ซึ่งควรจะได้กราฟเป็นเส้นตรง) แต่จริง ๆ แล้วภาพที่เห็นเป็นเชิงเส้น โดยมีข้อสังเกตอยู่สองข้อ คือ

1. เฟสแบ่งออกเป็นสองช่วง คือ ช่วงแถบผ่าน และช่วงแถบหยุด ซึ่งแต่ละช่วงมีความเป็นเชิงเส้นอิสระต่อกัน ซึ่งเรียกลักษณะเช่นนี้ว่า เป็นเชิงเส้นแบบเป็นช่วง ๆ (piece-wise linear) ลักษณะเช่นนี้ไม่กระทบกระเทือนต่อการทำงานของตัวกรอง เนื่องจากสัญญาณที่เราสนใจ และจะผ่านออกมาที่ขาออกอยู่ในช่วงแถบผ่านเท่านั้น สัญญาณในช่วงแถบหยุดที่ออกจะมาที่ขาออกมีน้อยมาก

2. การที่เฟสในช่วงแถบผ่านที่มีลักษณะเป็นฟันเลื่อยเกิดจากการที่ Matlab หรือเครื่องคำนวณอื่น ๆ ให้ค่าเฟสออกมาอยู่ในช่วง $-\pi$ ถึง π เท่านั้น (ใน Matlab ใช้คำสั่ง angle เพื่อหาเฟส) ลองพิจารณาจากในรูป จะเห็นว่าตอนเริ่มต้นที่ความถี่ 0 กราฟมีลักษณะลาดลงเป็นเส้นตรงจนกระทั่งถึงค่า $-\pi$ หรือประมาณ -3.14 เฟสก็จะพลิกกลับไปที่ค่า π จากนั้นก็ลาดต่อลงมาเป็นเส้นตรงด้วยความชันที่เท่าเดิม ถ้าเราเอาเส้นที่สองนี้ลากต่อกับเส้นแรกที่ตำแหน่ง $-\pi$ ไปเรื่อย ๆ (ทำได้เพราะ เมื่อบวกลบเฟสด้วยจำนวนเท่าของ 2π จะไม่ทำให้ค่าเปลี่ยน) รูปฟันเลื่อนที่เห็นนี้จริง ๆ เป็นเส้นตรงที่ลาดลงด้วยความชันคงที่ ดังนั้น ผลตอบสนองเชิงความถี่มีเฟสเชิงเส้นตรงตามทฤษฎี

หน้าต่างไคเซอร์ (Kaiser Window)

หน้าต่างไคเซอร์มีลักษณะพิเศษกว่าหน้าต่างแบบอื่น ๆ ตรงที่สามารถปรับค่าความพลิ้วของแถบผ่าน และการลดทอนของแถบหยุดได้ ซึ่งทำให้มีความยืดหยุ่นในการออกแบบกว่าหน้าต่างแบบอื่น ๆ ทำให้มักจะได้ค่า N ต่ำกว่าหน้าต่างแบบอื่นที่คุณลักษณะเฉพาะเดียวกัน

กำหนด $\delta = \min(\delta_{pass},\delta_{stop})$ = ค่าความพลิ้วที่เล็กที่สุดระหว่างแถบผ่าน กับแถบหยุด และให้ A = ค่าลดทอนของแถบหยุดซึ่งคิดจากค่าความพลิ้วที่เล็กที่สุดนี้ นั่นคือ

$$A_{pass} = A = -20\log\delta \text{ (dB)} \qquad (7.23)$$

กำหนดให้ α = shape parameter ซึ่งเป็นฟังก์ชั่นของ A ดังนี้

$$\alpha = \begin{cases} 0 & A \leq 21 \\ 0.\ 842(A-21)^{0.4} + 0.0\ 88\ (A-21) & 21\quad A\quad\ 0 \\ 0.1102(A-8.\ ) & A \geq\ \ 0 \end{cases} \qquad (7.24)$$

ค่า N ถูกกำหนดโดย $\Delta f'$ และ A และเช่นเดียวกันกับหน้าต่างแบบอื่น คือ $\Delta f'$ จะเป็นสัดส่วนกลับกับ N ดังสมการต่อไปนี้

$$N = \begin{cases} \dfrac{A - 7.95}{14.36\Delta f'} + 1 \ , & A > 21 \\ \dfrac{0.9}{\Delta f'} \ , & A \leq 21 \end{cases} \tag{7.25}$$

เมื่อได้ค่า N และ α เราสามารถหา w(n) ได้จากตารางที่ 7.2 โดย I_0 ในตาราง คือ ฟังก์ชั่นเบสเสลชนิดแรกที่มีอันดับ=0 (modified Bessel function of the first kind) มีสมการว่า

$$I_0(x) = \sum_{k=0}^{\infty} \left[\frac{(x/2)^k}{k!} \right]^2 \tag{7.26}$$

ฟังก์ชั่นเบสเสลนี้คงเป็นการยากที่จะคำนวณด้วยมือ หรือเครื่องคิดเลข โดยปกติเราต้องใช้เครื่องคอมพิวเตอร์ หรือเปิดตารางสำเร็จรูป Matlab มีฟังก์ชั่น besseli(0,x) ซึ่งจะคำนวณค่า $I_0(x)$ ให้เราได้ทันทีเมื่อแทนค่า x ลงไป ดังนั้น ถ้าต้องการใช้โปรแกรมที่ 7.2 คำนวณ w(n) ที่เป็นหน้าต่างไคเซอร์ก็สามารถเขียนได้ว่า

```
alpha = "ค่า α"
w = besseli(0,alpha*sqrt(1-(n-M).^2/M.^2))/besseli(0,alpha);
```

จะสังเกตจากสมการของ w(n) ได้ว่า เมื่อ α=0 หรือ A $\leq$ 21 จะได้ w(n)=1 นั่นคือ หน้าต่างไคเซอร์จะกลายเป็นหน้าต่างแบบสี่เหลี่ยม เมื่อ A $\leq$ 21 แต่โดยปกติเรามักใช้หน้าต่างไคเซอร์ที่มีค่า A > 21

<u>ตัวอย่างที่ 7.4</u> จงใช้หน้าต่างไคเซอร์ออกแบบตัวกรอง LPF แบบ FIR ที่มีคุณลักษณะเฉพาะดังต่อไปนี้ f_{pass} = 4kHz, f_{stop} = 5 kHz, f_s = 20kHz, δ_{pass} = 0.0058, และ A_{stop} = 80 dB

ก่อนอื่นหาค่า δ_{stop} ก่อน โดย $\delta_{stop} = 10^{-80/20} = 0.0001$

ดังนั้น $\delta = \min(\delta_{pass},\delta_{stop}) = \delta_{stop} = 0.0001$ และ $A = -20\log\delta = 80$ dB

ใช้สูตรของกรณีที่ A > 50 จะได้

α = 0.1102(A-8.7) = 0.1102(80-8.7) = 7.857

Δf = 5kHz - 4kHz =1kHz, $\Delta f' = \Delta f / f_s$ = 1k/20k = 0.05

$\omega'_c = 2\pi f_c/fs = 2\pi(4.5k)/20k = 0.45\pi$

$$N = \frac{A - 7.95}{14.36\Delta f'} + 1 = \frac{80 - 7.95}{14.36(0.05)} + 1 = 101.3$$ เลือก N = 101, M=(N-1)/2=50

เรายังคงใช้ d(n) แบบ LPF เหมือนตัวอย่างที่ 7.3 โดยใช้ ω'_c=0.45π

ส่วนฟังก์ชันหน้าต่าง คือ

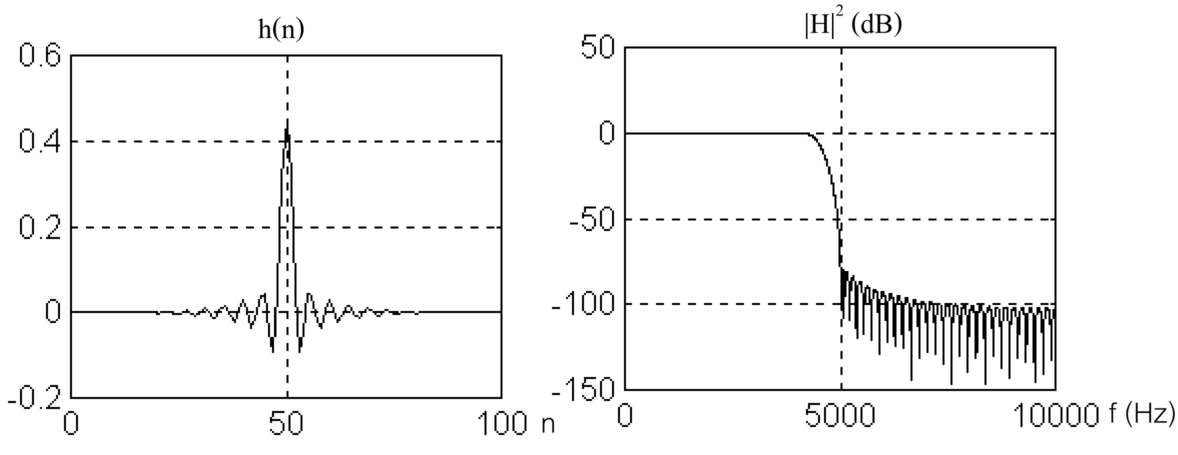
$$w(n) = \frac{I_0\left(\alpha\sqrt{1-(n-M)^2/M^2}\right)}{I_0(\alpha)}$$

$$= \frac{I_0\left(7.857\sqrt{1-(n-50)^2/50^2}\right)}{I_0(7.857)}, \quad n=0, 1, \ldots, 100$$

ดังนั้น จะได้ h(n) = d(n-M) w(n)

$$h(n) = \frac{\sin(0.45\pi(n-50))}{\pi(n-50)} \frac{I_0\left(7.857\sqrt{1-(n-50)^2/50^2}\right)}{I_0(7.857)}, \quad n=0, 1, \ldots, 100$$

วาดผลตอบสนองเชิงความถี่ที่ได้ในหน่วย dB ในรูปที่ 7.8



***รูปที่ 7.8** ตัวอย่างผลลัพธ์ที่ได้จากการใช้หน้าต่างไคเซอร์*

ตัวอย่างที่ 7.5 ต้องการตัวกรองแบบตัดแถบความถี่ (BSF) โดยมีความถี่ตัดของแถบ คือ f_a=2kHz และ f_b=3kHz ต้องการ δ_{pass} < 0.003, A_{stop} > 45 dB, และ Δf < 200 Hz หาด้วยว่าหน้าต่างชนิดใดที่ใช้ได้และใช้ได้ที่อันดับเท่าไร กำหนดให้ f_s = 10 kHz

ก่อนอื่นเปรียบเทียบก่อนว่า δ_{pass} กับ A_{stop} ใครเป็นข้อกำหนดที่แรงกว่า โดยเปลี่ยน A_{stop} เป็น δ_{stop} ได้ $\delta_{stop} = 10^{-\frac{Astop}{20}} = 0.00562$ เพราะฉะนั้น ข้อกำหนดของ δ_{pass} แรงกว่า ($\delta_{pass} < \delta_{stop}$) เราจะใช้ $\delta_{pass} = 0.003 = 0.3\%$ เป็นตัวเลือกชนิดของหน้าต่างที่ใช้ได้จากตารางที่ 7.2

พบว่า หน้าต่างแบบสี่เหลี่ยม และหน้าต่างฮานนิ่งมี $\delta_{pass} > 0.3\%$ ดังนั้นใช้ไม่ได้ และหน้าต่างที่ใช้ได้ คือ

ก) หน้าต่างแฮมมิ่ง ซึ่งมี $\delta_{pass} = 0.22\%$

$\Delta f'$ ที่ต้องการ คือ $\Delta f/fs = 200/10000 = 0.02$

หา N ที่ต้องใช้จาก $\Delta f' = 4/N$ แทนค่า $\Delta f'$ ลงไปจะได้ N = 200

เพราะฉะนั้น เลือก N = 201

ข) หน้าต่างแบล็กแมน ซึ่งมี $\delta_{pass} = 0.02\%$

หา N ที่ต้องใช้จาก $\Delta f' = 6/N$ แทนค่า $\Delta f'$ ลงไปจะได้ N = 300

เพราะฉะนั้น เลือก N = 301

ค) หน้าต่างไคเซอร์ ซึ่งมี δ_{pass} ปรับได้ตามต้องการ

ใช้ $\delta = \delta_{pass} = 0.003$ จะได้ $A = -20\log(\delta) = 50.5$ dB

ใช้สูตรกรณีที่ A > 21 หาค่า N คือ

$$N = \frac{A - 7.95}{14.36\Delta f'} + 1 = \frac{50.5 - 7.95}{14.36(0.02)} + 1 = 148.2$$

เพราะฉะนั้น เลือก N = 149

เลือกใช้หน้าต่างไคเซอร์ที่ N=149 และ M = (N-1)/2 = 74

หา α โดยใช้สูตรกรณี A > 50 ได้

$\alpha = 0.1102(A-8.7) = 0.1102(50.5-8.7) = 4.606$

ผลตอบสนองต่ออิมพัลส์สำหรับ BSF อุดมคติ จากตารางที่ 7.1 คือ

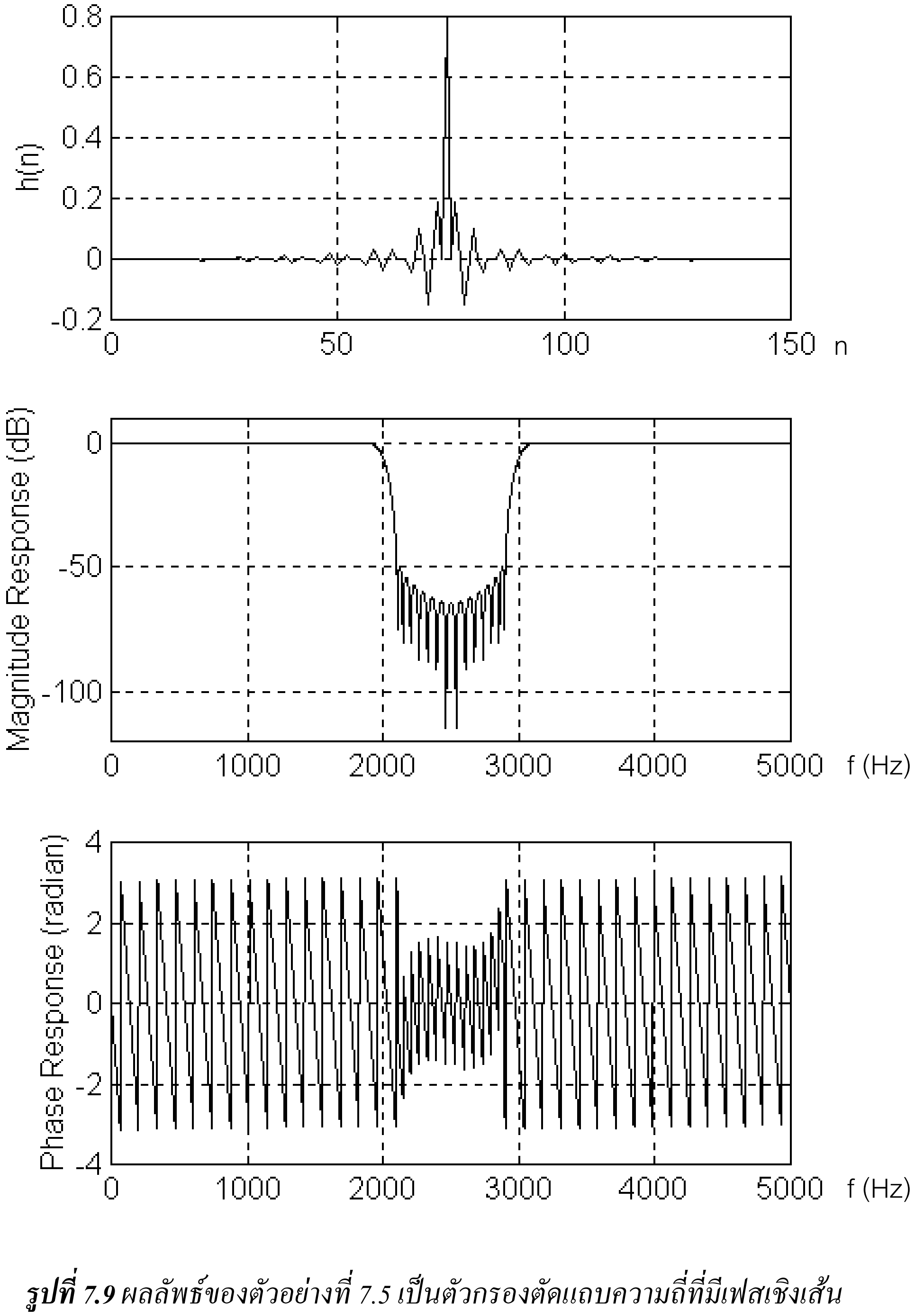
$$d(n) = \delta(n) - \frac{\sin(\omega'_b n) - \sin(\omega'_a n)}{\pi n}$$

ผลตอบสนองต่ออิมพัลส์ที่ได้ คือ h(n) = d(n-M) w(n) ดังนี้

$$h(n) = \left[\delta(n-M) - \frac{\sin(\omega'_b(n-M)) - \sin(\omega'_a(n-M))}{\pi(n-M)}\right]\frac{I_0\left(\alpha\sqrt{1-(n-M)^2/M^2}\right)}{I_0(\alpha)}$$

โดยที่ M=74, α=4.606, $\omega'_a=2\pi f_a/f_s = 0.4\pi$, $\omega'_b=2\pi f_b/f_s = 0.6\pi$, และ n=0, 1, …, 148 เราแก้ไขโปรแกรมที่ 7.2 ให้ใช้ d(n) เป็น BSF ได้ดังนี้

```
d=(-sin(wb*(n-M))+sin(wa*(n-M))./(n-M)/pi;
d((N+1)/2) = 1-wb/pi+wa/pi;
```



***รูปที่ 7.9** ผลลัพธ์ของตัวอย่างที่ 7.5 เป็นตัวกรองตัดแถบความถี่ที่มีเฟสเชิงเส้น*

การออกแบบโดยวิธีสุ่มความถี่ (Frequency Sampling Method)

วิธีสุ่มความถี่เป็นวิธีที่เหมาะสำหรับการออกแบบตัวกรองที่มีรูปร่างของผลตอบสนองเชิงความถี่แปลกไปจากปกติ โดยวิธีนี้ให้เราระบุจุดตัวอย่างของผลตอบสนองเชิงความถี่ที่ต้องการเป็นข้อกำหนดเริ่มต้นของการออกแบบ หลักการของการหาสัมประสิทธิ์สำหรับตัวกรองที่มีเฟสเชิงเส้นที่มีสมมาตรชนิดที่ 1 (N เป็นคี่) มีดังนี้

1. สมมติว่า $|D(e^{j\omega'})|$ คือ ผลตอบสนองทางขนาดที่ต้องการซึ่งอาจมีรูปร่างใด ๆ ก็ได้ และเป็นฟังก์ชั่นของ ω' การที่จะได้ตัวกรองสุดท้ายมีเฟสเชิงเส้น จะได้ว่าเฟสของระบบจะต้องเท่ากับ $-\omega'M$ โดย M=(N-1)/2 ตามที่ได้พิสูจน์มาในตัวอย่างที่ 7.1 เพราะฉะนั้นจะได้ผลตอบสนองเชิงความถี่ที่ต้องการ คือ

$$D(e^{j\omega'}) = e^{-j\omega' M}|D(e^{j\omega'})| \qquad (7.27)$$

2. ทำการสุ่มตัวอย่าง $D(e^{j\omega'})$ เป็นจำนวนทั้งสิ้น N จุด ด้วยระยะห่างเท่า ๆ กัน ในช่วงความถี่ $\omega' = 0$ ถึง 2π ดังนั้น คาบของการสุ่มเท่ากับ $2\pi/N$ นั่นคือ จะได้ผลตอบสนองเชิงความถี่เป็นแบบไม่ต่อเนื่องยาว N จุด เรียกผลตอบสนองเชิงความถี่นี้ว่า H(k) จะได้ว่า

$$H(k) = D(e^{j\omega'})\Big|_{\omega'=\frac{2\pi k}{N}} \qquad k = 0, 1, \ldots, N-1$$

$$H(k) = e^{-j\frac{2\pi k}{N}M}|H(k)|$$

$$H(k) = e^{-j\frac{N-1}{N}\pi k}|H(k)| \qquad (7.28)$$

โดยที่ $|H(k)|$ คือ ผลของการสุ่ม $|D(e^{j\omega'})|$ ในช่วง 0 ถึง 2π ด้วยระยะห่างคงที่เท่ากับ $2\pi/N$ หรือ จะคิดว่าเป็นการแทน ω' ด้วย $2\pi N/k$ ก็ได้ จากนั้นคูณ $|H(k)|$ ด้วยเฟส คือ $e^{-j\frac{N-1}{N}\pi k}$ ก็เป็นอันเสร็จ ได้ H(k) ที่เป็นผลตอบสนองที่ต้องการมีจำนวนทั้งสิ้น N จุด

3. หาผลตอบสนองต่อสัญญาณอิมพัลส์อันเนื่องมาจาก H(k) โดยใช้การแปลง IDFT หรือการแปลง IFFT คือ

$$h(n) = \mathrm{IDFT}\{H(k)\} \qquad (7.29)$$

ถ้า H(k) สมมาตรเราจะได้ h(n) เป็นค่าจริง อย่างไรก็ตาม ในบางกรณีที่ต้องใช้ H(k) ไม่สมมาตรพอดีก็จะให้ h(n) เป็นจำนวนเชิงซ้อน หรือ ถึงแม้ H(k) จะสมมาตรพอดี การคำนวณ IDFT โดยคอมพิวเตอร์ส่วนใหญ่ก็ไม่สามารถให้ผลที่แม่นยำ ซึ่งก็จะทำให้ผลลัพธ์ที่มีค่าจิตภาพหลุดออกมาบ้าง h(n) ที่เป็นจำนวนเชิงซ้อนไม่สามารถนำไปสร้างได้ การแก้ไขสามารถทำได้โดยปัดส่วนจินตภาพทิ้งทั้งหมด คือใช้เฉพาะส่วนจริงของผลลัพธ์เป็น h(n) ซึ่งก็จะทำให้ผลลัพธ์คลาดเคลื่อนไปบ้างแต่ก็ไม่มากนัก

h(n) ที่ได้จะมีความยาว N จุด และเราจะใช้ h(n) นี้เป็นผลตอบสนองต่ออิมพัลส์ของระบบเลย คำถามคือ แล้วผลตอบสนองเชิงความถี่ที่จะได้จาก h(n) นี้จะเหมือน หรือแตกต่างจากที่ตั้งไว้อย่างไร

ผลตอบสนองเชิงความถี่ที่เราจะได้ คือ $H(e^{j\omega'}) = \text{DTFT}\{\ h(n)\ \}$ ส่วนผลตอบสนองเชิงความถี่ที่ตั้งไว้ คือ $D(e^{j\omega'})$ บางคนอาจคิดว่า H กับ D น่าจะเหมือนกัน จริง ๆ แล้วมันจะไม่ทับกันสนิททีเดียว ถ้าใช้ความรู้จากเรื่องการแปลง DFT และ DTFT จะบอกได้ว่าผลตอบสนองเชิงความถี่ทั้งสองจะรับประกันว่ามีค่าตรงกันเฉพาะ ณ ตำแหน่งที่เราสุ่มตัวอย่างมาเท่านั้น ส่วนจุดอื่น ๆ ไม่รับประกัน อาจมีความใกล้เคียงกันมากหรือน้อย ก็ขึ้นกับรูปร่างของ $D(e^{j\omega'})$, จำนวน, และตำแหน่งของจุดที่สุ่มมา ขอให้ศึกษาเพิ่มเติมจากตัวอย่างต่อไปนี้

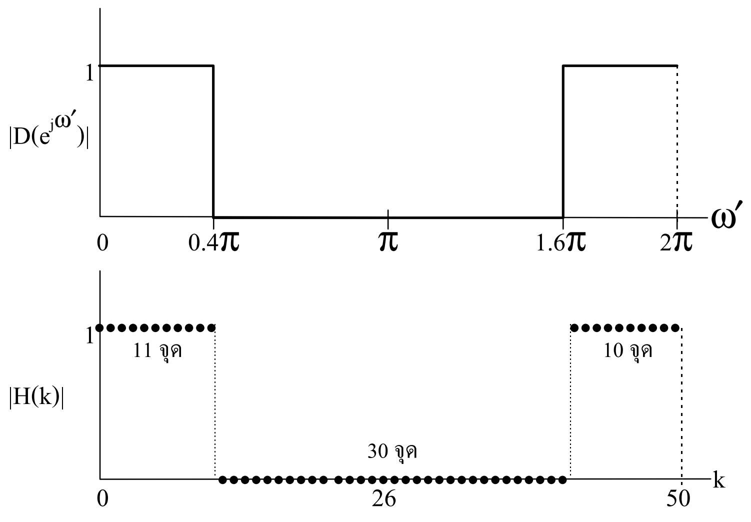
<u>**ตัวอย่างที่ 7.6**</u> หาสัมประสิทธิ์ของตัวกรอง FIR แบบผ่านต่ำที่มีความถี่ตัดที่ 2 kHz โดยวิธีการสุ่มความถี่ ใช้อัตราการสุ่มเท่ากับ 10 kHz

ความถี่ตัดทางดิจิตอล คือ $\omega_c' = 2\pi f_c/f_s = 0.4\pi$ เรเดียน

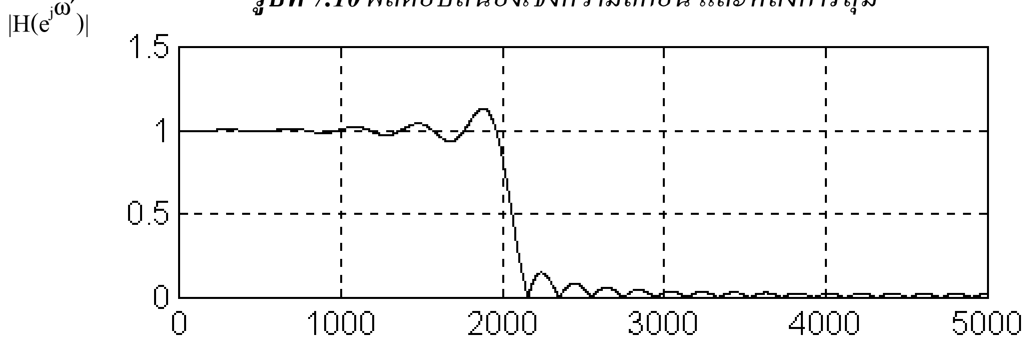
รูปของผลตอบสนองความถี่ที่ต้องการก่อน และหลังการสุ่มความถี่ แสดงดังรูปที่ 7.10 ซึ่งจะได้ |H(k)| = [หนึ่ง 11 จุด, ตามด้วยศูนย์ 30 จุด, และตามด้วยหนึ่ง 10 จุด]

จากนั้น คูณเฟสเพื่อให้ได้ผลตอบสนองเชิงความถี่เป็นเชิงเส้น จะได้

$$H(k) = |H(k)|\, e^{-j\frac{40\pi k}{41}}$$



***รูปที่ 7.10** ผลตอบสนองเชิงความถี่ก่อน และหลังการสุ่ม*



***รูปที่ 7.11** ผลตอบสนองเชิงความถี่ที่ได้จากวิธีสุ่มความถี่*

```
fs=10000;
H=[ones(1,11) zeros(1,30) ones(1,10)];   กำหนดขนาดของผลตอบสนองเชิงความถี่ที่ต้องการ
N=length(H);
k=0:N-1;
H=H.*exp(-j*pi*(N-1)*k/N);          คูณเฟสเพื่อให้ได้เฟสที่เป็นเชิงเส้น
h=real(ifft(H));
freqres(h,1,fs)
```

***โปรแกรมที่** 7.3 freqsam.m สำหรับออกแบบตัวกรอง FIR โดยวิธีสุ่มความถี่*

และผลตอบสนองต่ออิมพัลส์ของตัวกรองที่ต้องการ คือ h(n) = Real(IFFT{H(k)}) ซึ่งโปรแกรม Matlab ที่ใช้คำนวณผลลัพธ์ในข้อนี้ได้แสดงไว้ดังโปรแกรมที่ 7.3 และผลตอบสนองความถี่ที่ได้แสดงในรูปที่ 7.11

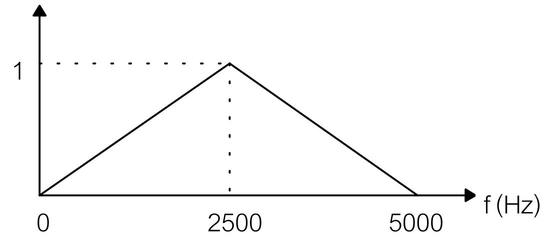
จากผลที่ได้ วัดความพลิ้วสูงสุดในช่วงแถบหยุดได้ประมาณ 0.15 หรือเท่ากับการลดทอนประมาณ 16.5 dB ซึ่งถือเป็นค่าที่ไม่ดีนักสำหรับตัวกรองที่มีอันดับเท่ากับ 50 ได้มีผู้เสนอวิธีในการปรับปรุงให้ผลตอบสนองเชิงความถี่ที่ได้ดีขึ้น เช่น

1) ใช้ฟังก์ชั่นหน้าต่างคูณเข้าไปกับ h(n) ที่ได้ ซึ่งก็จะให้ผลในทำนองเดียวกับวิธีหน้าต่างที่กล่าวไปแล้วในหัวข้อก่อน
2) เปลี่ยนจุดตัวอย่างในบริเวณแถบเปลี่ยนให้มีค่าอยู่ระหว่าง 0 ถึง 1 แทนที่จะเป็นค่า 0 หรือ 1 เลย เช่น ในตัวอย่างที่ 7.6 นี้ถ้าเราแทรกจุดหนึ่งจุดมีค่าเท่ากับ 0.3 ระหว่างช่วงแถบผ่าน และแถบหยุด จะทำให้ได้การลดทอนในแถบหยุดดีขึ้นมากในขณะที่ความกว้างของแถบเปลี่ยนกว้างขึ้นเล็กน้อย รายละเอียดว่าค่าที่ควรแทรกควรมีกี่ค่า และมีค่าเท่าใดสามารถหาได้จาก [3]

ตัวอย่างที่ 7.8 นี้แสดงให้เห็นถึงการใช้วิธีสุ่มความถี่สำหรับหาสัมประสิทธิ์ของตัวกรองพื้นฐาน ซึ่งแสดงให้เห็นถึงหลักการง่าย ๆ ของวิธีนี้ อย่างไรก็ตาม วิธีสุ่มความถี่นี้มีความเหมาะสมมาก

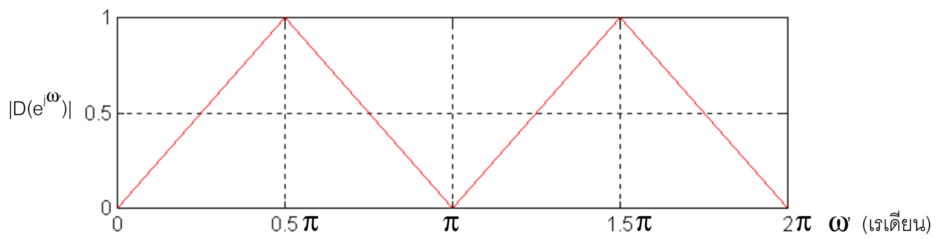
กว่า ที่จะนำไปใช้ออกแบบตัวกรองที่มีลักษณะผลตอบสนองเชิงความถี่แปลก ๆ ที่ไม่สามารถออกแบบโดยวิธีอื่นได้ หรือได้ไม่สะดวกนัก

ตัวอย่างที่ 7.7 หาสัมประสิทธิ์ของตัวกรองซึ่งมีผลตอบสนองอุดมคติดังรูปข้างล่างนี้



กำหนดให้ใช้ N=41 และ f_s=10 kHz วาดรูปของ h(n) และ $|H(e^{j\omega'})|$ ที่ได้

จาก f_s=10kHz จะได้ว่าความถี่ 5 kHz ตรงกับความถี่ดิจิตอลที่ $\omega' = \pi$ ดังนั้นตัวกรองดิจิตอลที่ต้องการมีผลตอบสนองเชิงความถี่ในช่วง 0 - 2π ดังแสดงในรูป



รูปที่ 7.12 ผลตอบสนองเชิงความถี่ของตัวกรองที่ต้องการ

ซึ่งจะได้ $|H(\omega')|$ มีสมการเป็น

$$\left|D(e^{j\omega'})\right| = \begin{cases} \dfrac{2\omega}{\pi} & , 0 \le \omega < \pi/2 \\ -\dfrac{2\omega}{\pi} + 2 & , \pi/2 \le \omega < \pi \\ \dfrac{2\omega}{\pi} - 2 & , \pi \le \omega \le 3\pi/2 \\ -\dfrac{2\omega}{\pi} + 4 & , 3\pi/2 \le \omega < 2\pi \end{cases}$$

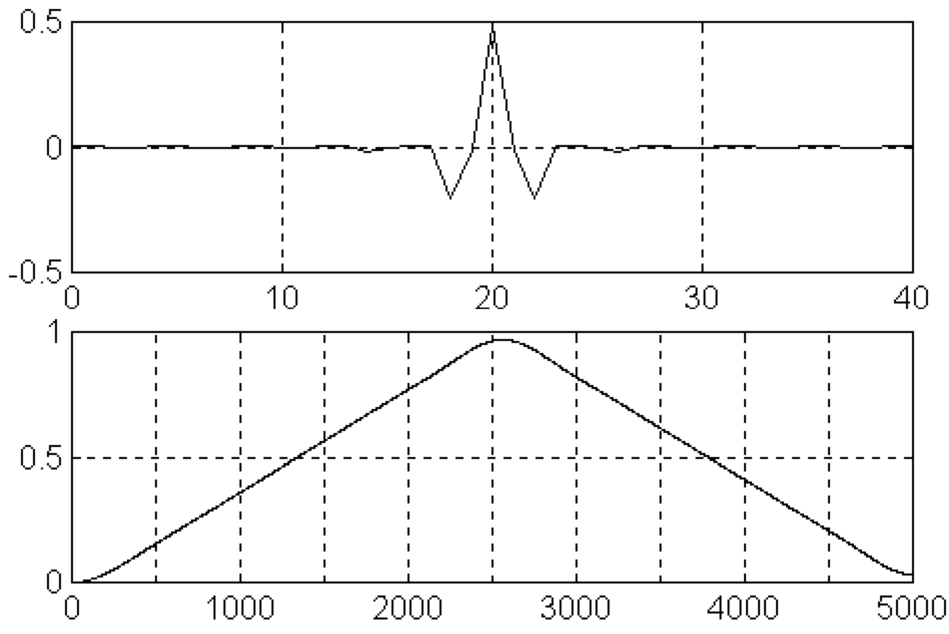
ทำการสุ่มความถี่ 41 จุดในช่วง 0-2π โดยแทน $\omega' = \dfrac{2\pi k}{41}$ จะได้

$$|H(k)| = \begin{cases} \frac{4k}{41} & , k = 0, 1, 2, \ldots, 10 \\ -\frac{4k}{41} + 2 & , k = 11, 12, 13, \ldots, 20 \\ \frac{4k}{41} - 2 & , k = 21, 22, 23, \ldots, 30 \\ -\frac{4k}{41} + 4 & , k = 31, 32, 33, \ldots, 40 \end{cases}$$

ค่า |H| สามารถกำหนดได้ใน Matlab ดังนี้

```
k1=0:10; k2=11:20; k3=21:30; k4=31:40;
H=[4*k1/41, -4*k2/41+2, 4*k3/41-2, -4*k4/41+4];
```

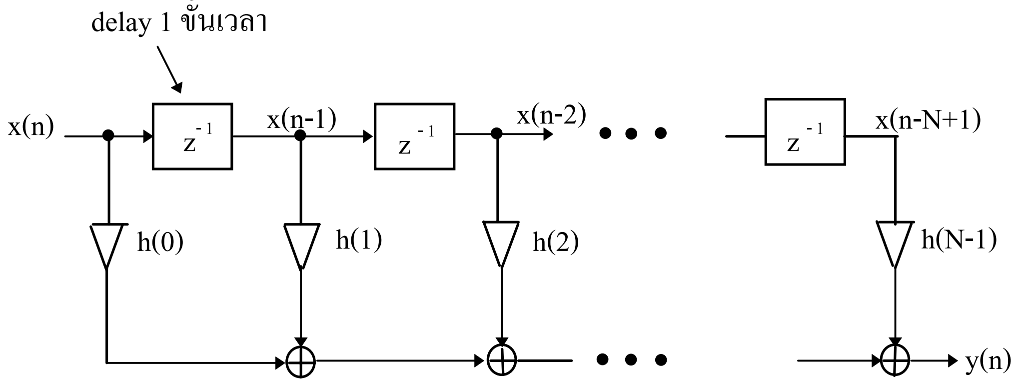
เมื่อแทนค่า H ลงในโปรแกรมที่ 7.3 และให้โปรแกรมคำนวณหาผลลัพธ์ จะได้ผลตอบสนองเชิงความถี่ของระบบดังในรูปที่ 7.13



รูปที่ 7.13 *ผลตอบสนองต่ออิมพัลส์ และผลตอบสนองเชิงความถี่ที่ได้จากวิธีสุ่มความถี่*

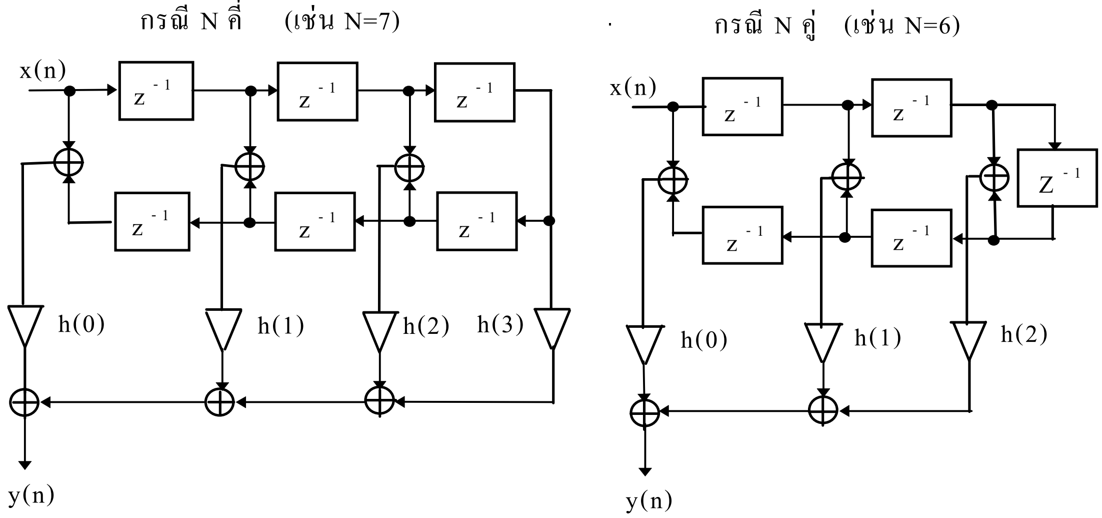
การสร้างตัวกรอง FIR (FIR Filter Realization)

คำว่า realization หมายถึง การนำเอาสิ่งที่ได้ออกแบบแล้ว หรือปรากฏการณ์ทางทฤษฎีไปประยุกต์เป็นอุปกรณ์ที่ใช้งานได้จริง ๆ ขึ้นมา สำหรับตัวกรองแบบ FIR เราจะใช้ผลตอบสนองต่ออิมพัลส์ หรือ h(n) เพื่อสร้างตัวกรอง ซึ่งกระบวนการของตัวกรองในที่นี้ก็ คือ การทำคอนโวลูชันระหว่าง h(n) และสัญญาณขาเข้า x(n) นั่นเอง ซึ่งสามารถเขียนเป็นแผนภาพได้ดัง รูปที่ 7.14



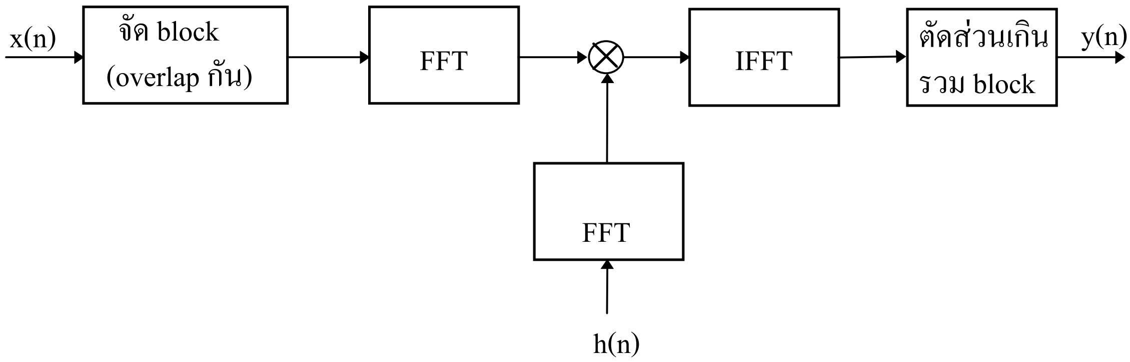
***รูปที่ 7.14** แผนภาพการสร้างตัวกรองแบบ FIR โดยคอนโวลูชันปกติ*

จากแผนภาพนี้เราสามารถนำไปประยุกต์เขียนเป็นโปรแกรมคอมพิวเตอร์ หรือทำเป็นฮาร์ดแวร์พิเศษเพื่อทำหน้าที่ตัวกรองแบบ FIR โดยตรง ในกรณีที่ตัวกรองเป็นแบบเฟสเชิงเส้น ซึ่งหมายถึง h(n) จะมีสมมาตร ณ จุดกึ่งกลาง เราอาจใช้คุณสมบัตินี้ลดโครงสร้างของ FIR ให้เล็กลงได้ดังแสดงในรูปที่ 7.15



***รูปที่ 7.15** แผนภาพการสร้างตัวกรองแบบ FIR เมื่ออาศัยคุณสมบัติการสมมาตร*

และโดยอาศัยวิธีคอนโวลูชันแบบเร็วที่เราได้ศึกษามาในบทที่ 6 ก็สามารถใช้แทนคอนโวลูชันปกติเพื่อใช้สร้างตัวกรองแบบ FIR ได้ ดังในรูปที่ 7.16



รูปที่ 7.16 *แผนภาพการสร้างตัวกรองแบบ FIR โดยวิธีคอนโวลูชันแบบเร็ว*

บทที่ 8

ตัวกรองแบบ IIR

ในบทนี้เราจะกล่าวถึงการออกแบบขั้นพื้นฐานของตัวกรองแบบ IIR, การสร้างตัวกรองแบบ IIR, และรวมถึงเปรียบเทียบข้อดีข้อเสียระหว่างตัวกรองแบบ FIR และ IIR

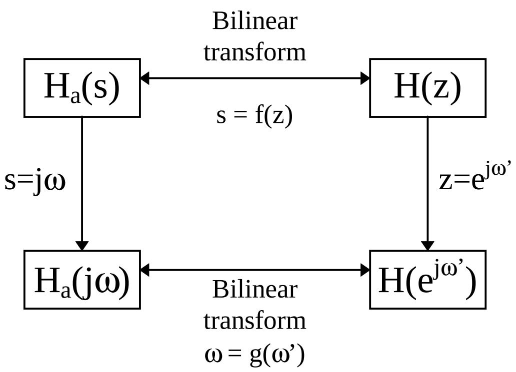
ตัวกรองแบบ IIR เป็นระบบที่มีโพล และมีความยาวของผลตอบสนองต่ออิมพัลส์ไม่จำกัด ในการออกแบบตัวกรอง IIR เราจะไม่มุ่งเป้าที่การหา h(n) เหมือนอย่างการออกแบบตัวกรอง FIR แต่เราจะมุ่งไปที่การหาฟังก์ชั่นถ่ายโอน H(z) ของระบบ และเราจะได้เห็นอีกว่า การสร้างตัวกรอง IIR สามารถกระทำได้โดยตรงจากพารามิเตอร์ใน H(z) ทันที โดยไม่ต้องสนใจ h(n) เลย

การออกแบบโดยอิงตัวกรองแอนะลอกต้นแบบ

วิธีออกแบบตัวกรองดิจิตอลที่นิยมมากวิธีหนึ่งก็คือ การออกแบบโดยอิงตัวกรองแอนะลอกต้นแบบ ซึ่งได้แก่ ตัวกรองแบบ Butterworth, Chebychev, Elliptic, Bessel, และอื่น ๆ ตัวกรองแอนะลอกเหล่านี้เป็นพื้นฐานที่ถูกศึกษา และพัฒนามาถึงจุดที่ค่อนข้างสมบูรณ์แล้ว ถ้าเราสามารถหาฟังก์ชั่น หรือการแปลงอย่างใดอย่างหนึ่งที่สามารถแปลงฟังก์ชั่นถ่ายโอนของระบบแบบแอนะลอก มาเป็นระบบแบบไม่ต่อเนื่องได้ เราก็อาจจะสามารถนำตัวกรองในระบบแอนะลอกมาใช้ในระบบไม่ต่อเนื่องได้ทันที

การแปลงดังกล่าวไม่ยากอย่างที่คิด เนื่องจากหลักการของระบบต่อเนื่อง และระบบไม่ต่อเนื่องมีลักษณะคล้ายคลึงกัน ดังที่เราได้เห็นการแปลง z ที่มีลักษณะการใช้งานเช่นเดียวกับการแปลงลาปลาสในระบบแบบต่อเนื่องมาแล้ว ในรูปที่ 8.1 แสดงความสัมพันธ์กันของระบบทั้งสอง จะเห็นได้ว่า จากฟังก์ชั่นถ่ายโอนของระบบต่อเนื่อง ($H_a(s)$ ซึ่งเป็นฟังก์ชั่นของ s) เราสามารถหาผลตอบสนองเชิงความถี่ได้ โดยแทน $s = j\omega$ และสำหรับฟังก์ชั่นถ่ายโอนของระบบไม่ต่อเนื่อง (H(z) ซึ่งเป็นฟังก์ชั่นของ z) ก็สามารถหาผลตอบสนองความถี่ได้โดยแทน $z = e^{j\omega'}$

สำหรับการแปลงระหว่างฟังก์ชั่นโอนย้ายของสองระบบ คือ จาก $H_a(s)$ ไปเป็น H(z) หรือ จาก $H_a(\omega)$ ไปเป็น $H(e^{j\omega'})$ เราต้องการฟังก์ชั่นพิเศษในการแปลงจากโดเมน s เป็นโดเมน z ซึ่งการแปลงนี้เรียกว่า <u>การแปลงไบลิเนียร์</u> (Bilinear Transform)



รูปที่ 8.1 *ความสัมพันธ์กันของระบบแบบต่อเนื่อง และไม่ต่อเนื่อง*

เราจะลองศึกษาการแปลงไบลิเนียร์สำหรับแปลงระบบผ่านต่ำแอนะลอก เป็นระบบผ่านต่ำไม่ต่อเนื่องก่อน ซึ่งพบว่าฟังก์ชั่นสำหรับแปลงจากโดเมน s เป็นโดเมน z อยู่ในรูปดังนี้

$$s \;=\; K\frac{z-1}{z+1} \qquad (8.1)$$

โดยที่ K เป็นค่าคงที่สำหรับการแปลง ขอละไม่กล่าวถึงวิธีพิสูจน์ แต่จะศึกษาถึงเฉพาะผลที่เกิดขึ้นจากการแปลงดังกล่าว (ผู้ที่สนใจวิธีพิสูจน์ สามารถดูได้จากหนังสืออ้างอิง [3]) ผลที่เกิดจากการแปลงไบลิเนียร์สามารถมองได้เป็น 2 จุดใหญ่ ๆ คือ

1. เกิดการแปลงโพล และศูนย์บน s-plane ของระบบต่อเนื่อง ไปเป็นโพล และศูนย์บน z-plane ของระบบไม่ต่อเนื่อง

ซึ่งจุดที่เราให้ความสนใจเป็นพิเศษ คือ เกิดการดึงพื้นที่ในซีกซ้ายของ s-plane ไปยังพื้นที่ภายใต้วงกลมขนาด 1 หน่วยของ z-plane ถ้าระบบแบบแอนะลอกมีโพลอยู่ในซีกซ้ายของ s-plane เมื่อแปลงเป็นระบบดิจิตอลโพลนั้นก็จะอยู่ภายใต้วงกลมขนาด 1 หน่วยของ z-plane ดังในรูปที่ 8.2 นั่นหมายถึงว่า ถ้าระบบแอนะลอกที่เป็นต้นแบบมีเสถียรภาพ และคอซัล เมื่อแปลงเป็นระบบแบบไม่ต่อเนื่องก็จะได้ระบบที่มีเสถียรภาพ และเป็นคอซัลด้วย

เราสามารถพิสูจน์ผลข้อนี้ได้พิสูจน์ว่า ถ้าส่วนจริงของ s มีค่าน้อยกว่าศูนย์ (โพลของระบบแอนะลอกอยู่ซีกซ้าย) เมื่อแปลงจะได้ขนาดของ z มีค่าน้อยกว่า 1 (โพลของระบบดิจิตอลอยู่ภายในวงกลมหนึ่งหน่วย) เริ่มต้นจากส่วนจริงของ s ซึ่งสามารถเขียนได้เป็นผลบวกของ s และ s* ดังนี้

$$\text{Real}\{s\} \;=\; \frac{(s \;+\; s^*)}{2} \qquad (8.2)$$

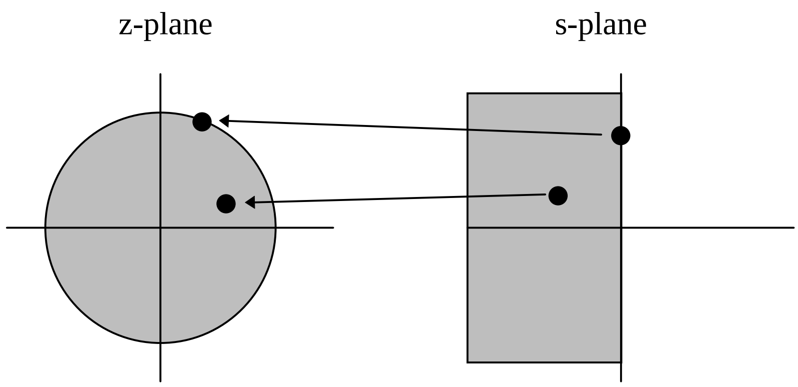
แทนค่า s ด้วยความสัมพันธ์ตามสมการที่ 8.1 จะได้

$$
\begin{aligned}
\text{Real}\{s\} &= \frac{K}{2}\left[\frac{z-1}{z+1} + \frac{z^*-1}{z^*+1}\right] \\
&= \frac{K}{2}\left[\frac{(z-1)(z^*+1) + (z+1)(z^*-1)}{(z+1)(z^*+1)}\right] \\
&= \frac{K}{2}\left[\frac{2zz^*-2}{(z+1)(z^*+1)}\right] \\
&= \frac{K(zz^* - 1)}{(z+1)(z+1)^*}
\end{aligned}
$$

ใช้ความสัมพันธ์ว่า $z\,z^*$ มีค่าเท่ากับ $|z|^2$ เราจะสามารถเขียนสมการนี้เป็นฟังก์ชั่นของขนาดของ z ได้ดังนี้

$$\text{Real}\{s\} = \frac{K(|z|^2-1)}{|z+1|^2} \qquad (8.3)$$

จากสมการนี้ จะเห็นได้ว่า $\text{Real}\{s\} = 0$ เกิดขึ้นเมื่อ $|z| = 1$ นั่นคือเส้นแบ่งระหว่างซีกซ้ายและขวาของ s-plane ถูกดึงมาที่เส้นวงกลมหนึ่งหน่วยของ z-plane ซึ่งเส้นนี้คือเส้นแบ่งเงื่อนไขความมีเสถียรภาพของระบบทั้งสอง และจะเห็นได้ว่าเมื่อ $\text{Real}\{s\} < 0$ จะได้ $|z| < 1$ ตามที่ต้องการพิสูจน์



รูปที่ 8.2 *การแปลงระหว่างโพลใน s-plane ไปยังโพลใน z-plane*

2. เกิดการแปลงระหว่างความถี่แอนะลอกไปเป็นความถี่ดิจิตอล

ถ้าเราแทนค่า $s = j\Omega$ และแทน $z = e^{jw'}$ ลงในสมการที่ 8.1 จะได้ฟังก์ชั่นที่เป็นการแปลงระหว่างความถี่แอนะลอกไปเป็นความถี่ดิจิตอล ดังนี้

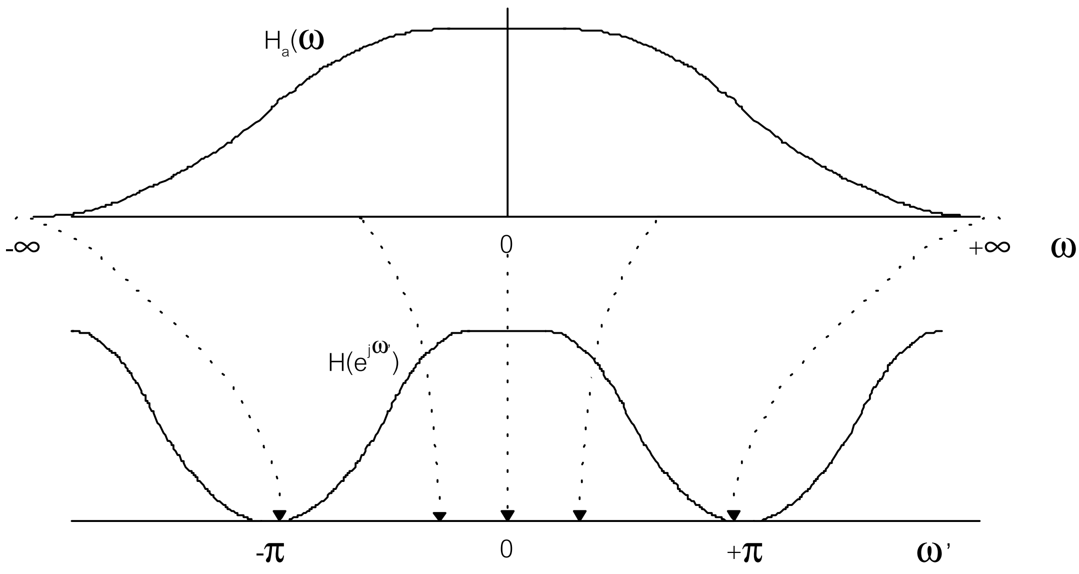
$$j\omega = K\frac{e^{j\omega'} - 1}{e^{j\omega'} + 1}$$

$$= K\frac{e^{j\omega'/2}(e^{j\omega'/2} - e^{-j\omega'/2})}{e^{j\omega'/2}(e^{j\omega'/2} + e^{-j\omega'/2})}$$

$$j\omega = K\frac{j\sin(\omega')/2}{\cos(\omega')/2}$$

$$\omega = K\tan(\frac{\omega'}{2}) \qquad (8.4)$$

เมื่อพิจารณาสมการนี้ จะได้ว่า ที่ $\omega = \infty$ จะได้ $\omega' = \pi$ และที่ ω=0 จะได้ ω'=0 นั่นคือ ความถี่ทั้งหมดของแอนะลอกในช่วง 0 ถึง ∞ ถูกดึงเข้ามาอยู่ในช่วง 0 ถึง π ของความถี่ดิจิตอล ซึ่งก็คือช่วงทำงานของความถี่ดิจิตอลนั่นเอง ผลของการแปลงความถี่โดยรวมเกิดขึ้นดังในรูปที่ 8.3 ซึ่งเห็นได้ชัดว่าเป็นการแปลงจากตัวกรองผ่านต่ำแอนะลอก เป็นตัวกรองผ่านต่ำดิจิตอล ถ้าลองสมมติให้ K=1 และวาดกราฟระหว่าง ω และ ω' จะได้ดังรูปที่ 8.4



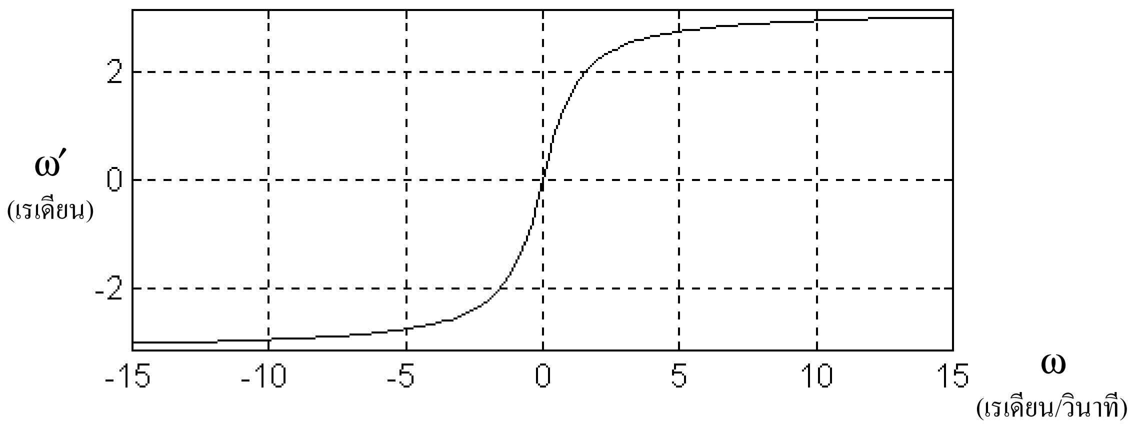
รูปที่ 8.3 *การแปลงระหว่างความถี่แอนะลอก และดิจิตอล กับผลที่เกิดขึ้นกับ ผลตอบสนองเชิงความถี่ในการแปลงไบลิเนียร์*

จากรูปที่ 8.4 จะสังเกตได้ว่า การแปลงความถี่ที่เกิดขึ้นไม่ได้มีลักษณะเป็นเชิงเส้น ซึ่งฟังก์ชั่นเชิงเส้นไม่สามารถนำมาใช้ในการแปลงความถี่ได้ เพราะความถี่แอนะลอกมีช่วงตั้งแต่ 0 ถึงอนันต์ แต่ช่วงทำงานของความถี่ดิจิตอลมีช่วงแค่ 0 ถึง π เท่านั้น ดังนั้น ถ้าใช้ฟังก์ชั่นเชิงเส้นก็คงจะได้ช่วงแถบผ่านหดลดเหลือเล็กนิดเดียว

การใช้การแปลงไบลิเนียร์ ทำให้ฟังก์ชั่นการแปลงความถี่มีลักษณะคล้ายเป็นเชิงเส้นแบบสองช่วง คือในช่วงความถี่ต้น ๆ (ω ประมาณ 0 ถึง 2 เรเดียน/วินาที) กราฟจะมีความชันมาก และช่วงความถี่หลัง ๆ กราฟจะมีความชันน้อยลงมาก ซึ่งโดยปกติเราสามารถเลือกค่า K ให้กราฟมีจุดแบ่งของสองช่วงนี้ที่ประมาณ ณ ความถี่ตัดของตัวกรอง ซึ่งก็จะทำให้ในช่วงแถบผ่านมีการแปลงความถี่อย่างช้า ๆ และในช่วงแถบหยุดมีการแปลงความถี่อย่างเร็ว ซึ่งก็จะดึงให้ความถี่แอนะลอกทั้งหมด (ยาวจนถึงอนันต์) ในแถบหยุด ถูกดึงมาจำกัดอยู่ในความถี่ π ของตัวกรองดิจิตอลได้ ปรากฏการณ์ที่ความถี่ถูกดึงหดเข้ามานี้ เรียกว่า frequency warping

ผลของ frequency warping ทำให้รูปร่างของผลตอบสนองเชิงความถี่ของตัวกรองดิจิตอลที่ได้มีความแตกต่างจากตัวกรองแอนะลอกต้นแบบบ้าง แต่ก็ไม่เป็นผลสำคัญอะไร เพราะลักษณะสำคัญของตัวกรองแอนะลอกต้นแบบได้ถูกถ่ายทอดมายังตัวกรองดิจิตอลแล้ว เช่น ลักษณะของความคมและความพลิ้วของตัวกรอง และที่สำคัญคือ ได้ความถี่ตัดของตัวกรองดิจิตอลตามที่ต้องการด้วย

เนื่องจากฟังก์ชั่น tan มีลักษณะเป็นคาบ กราฟที่แสดงในรูปที่ 8.4 นี้แสดงเฉพาะช่วงความถี่ ω' เท่ากับ $-\pi$ ถึง π เท่านั้น ในช่วงอื่น ๆ ก็จะมีลักษณะเป็นคาบทุก ๆ 2π เช่นเดียวกับกราฟนี้ ซึ่งก็จะมีผลทำให้ได้ผลตอบสนองเชิงความถี่ของระบบดิจิตอล มีลักษณะเป็นคาบทุก ๆ 2π ตรงตามทฤษฎีที่เราได้ศึกษามาในบทก่อน ๆ



รูปที่ 8.4 *กราฟแสดงความสัมพันธ์ระหว่างความถี่แอนะลอก และดิจิตอล ในการทำการแปลงไบลิเนียร์*

สรุปความว่า ถ้ามีฟังก์ชั่นถ่ายโอน $H_a(s)$ ของตัวกรองแอนะลอกผ่านต่ำ เราจะสามารถแปลงฟังก์ชั่นนี้เป็นฟังก์ชั่นถ่ายโอน $H(z)$ ของตัวกรองดิจิตอลได้ทันที โดยใช้การแปลงไบลิเนียร์ คือแทนค่า s ใน $H_a(s)$ ตามสมการที่ 8.1 คือ $s = K\dfrac{z-1}{z+1}$ หรือ เขียนได้ว่า

$$H(z) = \left[H_a(s)\right]_{s=K\frac{z-1}{z+1}} \tag{8.5}$$

ซึ่งผลที่เกิดขึ้น คือ เราได้ตัวกรองดิจิตอลแบบ IIR ซึ่งทำหน้าที่เป็นตัวกรองผ่านต่ำในลักษณะเดียวกับตัวกรองแอนะลอกที่เป็นต้นแบบ และมีเสถียรภาพเหมือนตัวกรองต้นแบบเช่นกัน ในส่วนต่อไป จะได้ยกตัวอย่างตัวกรองดิจิตอลบัตเตอร์เวิร์ธ ซึ่งเกิดจากตัวกรองบัตเตอร์เวิร์ธแอนะลอกต้นแบบ

ตัวกรองบัตเตอร์เวิร์ธแอนะลอกต้นแบบ

ตัวกรองบัตเตอร์เวิร์ธ (Butterworth) ต้นแบบที่เราจะใช้เป็นแบบผ่านต่ำ ลักษณะผลตอบสนองเชิงความถี่ของตัวกรองบัตเตอร์เวิร์ธเป็นแบบ monothonic คือ ลาดลงตลอดจากความถี่ศูนย์จนถึงความถี่อนันต์ ไม่มีลักษณะเป็นลูกคลื่น

ตัวกรองบัตเตอร์เวิร์ธอันดับ N มี ฟังก์ชั่นถ่ายโอนในรูปแบบ ดังนี้

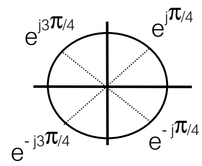
$$H_a(s) = \frac{1}{\prod_{i=1}^{N}(s-p_i)} = \frac{1}{(s-p_1)(s-p_2)\ldots(s-p_N)} \tag{8.6}$$

อันดับ (n)	มุมของโพล ($\theta_1, \theta_2, \ldots, \theta_N$)
1	π
2	$\pm\frac{3\pi}{4}$
3	$\pi,\ \pm\frac{4\pi}{6}$
4	$\pm\frac{5\pi}{8},\ \pm\frac{7\pi}{8}$
5	$\pi,\ \pm\frac{6\pi}{10},\ \pm\frac{8\pi}{10}$
6	$\pm\frac{7\pi}{12},\ \pm\frac{9\pi}{12},\ \pm\frac{11\pi}{12}$
7	$\pi,\ \pm\frac{8\pi}{14},\ \pm\frac{10\pi}{14},\ \pm\frac{12\pi}{14}$

ตารางที่ 8.1 *มุมของโพลของตัวกรองบัตเตอร์เวิร์ธ (โพลมีค่าเท่ากับ $e^{j\theta}$)*

โดยที่ p_i เป็นโพลของระบบ ซึ่ง p_i มาจากผลตอบ หรือรากที่อยู่ด้านซ้ายของ S-plane ของสมการ

$$1+\left(-s^2\right)^N = 0 \tag{8.7}$$



ตัวอย่างเช่น ถ้า N=2 จะได้สมการสำหรับหาโพลเป็น

$1+\left(-s^2\right)^2 = 0$ ซึ่งคือ $s^4 = -1$

จะได้ผลตอบของสมการนี้มี 4 ค่า ซึ่งผลตอบทั้งสี่จะอยู่บนวงกลมรัศมี 1 หน่วยดังรูป

ดังนั้นรากที่อยู่ด้านซ้าย หรือรากที่มีค่าจริงเป็นลบ คือ $e^{j3\pi/4}$ และ $e^{-j3\pi/4}$ จะเป็นโพลของตัวกรองบัตเตอร์เวิร์ธ, N=2 ค่าที่ได้นี้ตรงกับที่แสดงไว้ในตารางที่ 8.1

เมื่อหากำลังสองของขนาดของฟังก์ชั่นถ่ายโอนของตัวกรองบัตเตอร์เวิร์ธ จะได้

$$\left|H_a(s)\right|^2 = H_a(s)H_a(-s) = \frac{1}{1+(-s)^{2N}} \tag{8.8}$$

เพื่อมิให้สับสนกับสัญลักษณ์ของความถี่แอนะลอก และดิจิตอลที่ได้ใช้มาซึ่งหมายถึง ความถี่ของระบบเดียวกันที่มีสัดส่วนกันเท่ากับ fs จะขอสมมติสัญลักษณ์ใหม่ คือ Ω เพื่อใช้แทนความถี่แอนะลอกของวงจรต้นแบบ โดยที่ $\Omega = [0, \infty]$ ดังนั้น เมื่อแทน $s=j\Omega$ จะได้ผลตอบสนองเชิงความถี่ของตัวกรองบัตเตอร์เวิร์ธ ต้นแบบเป็น

$$\left|H_a(j\Omega)\right|^2 = \frac{1}{1+\Omega^{2N}} \tag{8.9}$$

ตัวกรองบัตเตอร์เวิร์ธต้นแบบนี้ มีลักษณะพิเศษ คือ ได้ถูกทำให้มีความถี่ตัด (Ω_c) เท่ากับ 1 เรเดียนต่อวินาที ซึ่งถ้าลองแทนค่า $\Omega = \Omega_c = 1$ ลงในสมการ จะได้ $\left|H_a(j\Omega)\right|^2 = \frac{1}{2}$ หรือคือจุดที่มีการลดทอนเท่ากับ 3dB

<u>ข้อสังเกต</u> 1) โพลจะมีคู่คอนจูเกตกันเสมอ

2) ถ้า n เป็นค่าคี่ จะมีโพลหนึ่งที่เป็นค่าจริง คือ มีค่าเท่ากับ $e^{j\pi}$ หรือ -1 เสมอ

การออกแบบตัวกรองบัตเตอร์เวิร์ธดิจิตอลผ่านต่ำ

ขั้นตอนต่อไปนี้เป็นวิธีการหาฟังก์ชั่นถ่ายโอนของตัวกรองบัตเตอร์เวิร์ธดิจิตอลผ่านต่ำ

1) จากค่ากำหนดของความถี่ตัดที่ต้องการ หาค่า K ของการแปลงไบลิเนียร์ โดยจับให้ความถี่ตัดของตัวกรองทั้งสองเป็นจุดเดียวกัน นั้นคือ ที่ $\Omega = \Omega_c$ ตรงกับ $\omega' = \omega'_c$ ซึ่งทำได้โดยแทนค่าความถี่ทั้งสองลงในสมการที่เป็นการแปลงความถี่ คือสมการที่ 8.4 สมการนี้เขียนใหม่โดยแทนสัญลักษณ์ของความถี่แอนะลอกด้วย Ω ได้ดังนี้

$$\Omega = K\tan\left(\frac{\omega'}{2}\right) \qquad (8.10)$$

แทนค่า $\Omega = \Omega_c$ และ $\omega' = \omega'_c$ แล้วจัดรูปสมการเพื่อหาค่า K จะได้

$$\Omega_c = K\tan\left(\frac{\omega'_c}{2}\right)$$

$$K = \Omega_c \cot\left(\frac{\omega'_c}{2}\right) \qquad (8.11)$$

ในกรณีนี้ตัวกรองแอนะลอกต้นแบบมี $\Omega_c = 1$ เมื่อแทนค่าลงไปจะทำให้สมการลดรูปเหลือ

$$K = \cot\left(\frac{\omega'_c}{2}\right) \qquad (8.12)$$

2) จากข้อกำหนดด้านความคมของผลตอบสนองเชิงความถี่ หาค่าอันดับของตัวกรองที่จำเป็นต้องใช้ (ถ้ายังไม่ได้กำหนด)

จากสมการที่ 8.9 ซึ่งคือ ผลตอบสนองเชิงความถี่ของตัวกรองแอนะลอกต้นแบบ ดังนี้

$$\left|H_a(j\Omega)\right|^2 = \frac{1}{1+\Omega^{2N}} \qquad (8.9)$$

จากสมการนี้ จะสามารถหาสมการของผลตอบสนองเชิงความถี่ของตัวกรองดิจิตอลได้ โดยแทน $\Omega = K\tan\left(\frac{\omega'}{2}\right)$ ซึ่งจะได้

$$\left|H\left(e^{j\omega'}\right)\right|^2 = \frac{1}{1+\left[K\tan\left(\frac{\omega'}{2}\right)\right]^{2N}} \tag{8.13}$$

แทนค่าข้อกำหนดของผลตอบสนองเชิงความถี่ลงไป (คือ ค่า |H| ที่ต้องการที่ความถี่ ω' ใด ๆ) ก็จะสามารถแก้สมการหาค่า N ต่ำสุดที่จำเป็นต้องใช้ ได้

3) เมื่อได้ค่าอันดับที่ต้องการ หา $H_a(s)$ ของตัวกรองต้นแบบ

สมมติให้ N=5 จะได้ $H_a(s)$ ดังนี้

$$H_a(s) = \frac{1}{(s-p_1)(s-p_1^*)}\;\frac{1}{(s-p_2)(s-p_2^*)}\;\frac{1}{(s-p_3)} \tag{8.14}$$

4) หา H(z) โดยการแปลงไบลิเนียร์ตามสมการที่ 8.5 คือ

$$H(z) = \left[H_a(s)\right]_{s=K\frac{z-1}{z+1}} \tag{8.5}$$

สูตรนี้ตรงไปตรงมา โดยแทน s ทุกตัวในสมการด้วย $K\dfrac{z-1}{z+1}$ แล้วจัดให้อยู่ในรูปเศษส่วนของโพลิโนเมียลที่จะนำไปใช้ได้ อย่างไรก็ตาม ค่อนข้างจะยุ่งในการจัดบ้าง เราพบว่า สำหรับ H(s) ที่อยู่ในรูปดังสมการที่ 8.14 เราสามารถจัด H(z) ให้อยู่ในรูปทั่วไปได้ดังนี้

$$H(z) = \frac{G_1(z+1)^2}{(z-z_1)(z-z_1^*)}\;\frac{G_2(z+1)^2}{(z-z_2)(z-z_2^*)}\;\frac{G_3(z+1)}{z-z_3} \tag{8.15}$$

$$= \frac{G_1(z+1)^2}{z^2-2\,\mathrm{Re}\{z_1\}z+|z_1|^2}\;\frac{G_2(z+1)^2}{z^2-2\,\mathrm{Re}\{z_2\}z+|z_2|^2}\;\frac{G_3(z+1)}{z-z_3} \tag{8.16}$$

โดยที่ $z_i = \dfrac{K+p_i}{K-p_i}$ และ $G_i = \dfrac{1}{|K-p_i|^2}$

สำหรับ i = 1, 2 ซึ่งสูตรนี้จะใช้ได้กับเทอมโพลที่มีคู่คอนจูเกตกัน ซึ่งในกรณีนี้ คือ เทอมที่หนึ่งและสอง สำหรับในกรณีที่อันดับเป็นคี่จะมีเทอมที่ไม่มีคู่คอนจูเกตเหลืออยู่ 1 เทอม ซึ่งในกรณีนี้เกิดจากโพล p_3 เราจะได้

$$z_3 = \frac{K+p_3}{K-p_3} \quad \text{และ} \quad G_3 = \frac{1}{K-p_3}$$

<u>ตัวอย่างที่ 8.1</u> ออกแบบตัวกรองบัตเตอร์เวิร์ธแบบผ่านต่ำที่มีความถี่ตัดที่ 2 kHz และใช้ความถี่ในการสุ่มเท่ากับ 8 kHz ให้ระบบมีอัตราการลดทอนไม่ต่ำกว่า 20 dB ที่ 3kHz

1) หาค่า K

จาก $K = \cot\left(\frac{\omega'_c}{2}\right)$ แทนค่า $\omega'_c = \frac{2\pi f_c}{f_s} = \frac{2\pi(2000)}{8000} = \frac{\pi}{2}$

$$= \cot\left(\frac{\pi}{2}\times\frac{1}{2}\right)$$

$$= \cot\left(\frac{\pi}{4}\right) = 1$$

2) หาค่า N ที่ต้องใช้จากข้อกำหนดของผลตอบสนองเชิงความถี่

ที่ 3 kHz ต้องการค่าลดทอน = 20 dB นั่นคือ ต้องกำลังขยาย = - 20 dB

จากสูตรว่า $dB = 10\log|H|^2$ จะได้ $|H|^2 = 10^{-20/10} = 0.01$

สำหรับ f = 3 kHz ตรงกับความถี่ดิจิตอลที่ $\omega' = \frac{2\pi f}{f_s} = \frac{2\pi(3k)}{8k} = \frac{3\pi}{4}$ เรเดียน

นั่นคือต้องการ $|H|^2$ ที่ $\omega' = 3\pi/4 < 0.01$ แทนค่า $|H|^2$ จากสมการที่ 8.13 จะได้

$$\frac{1}{1+\left[K\tan\left(\frac{\omega'}{2}\right)\right]^{2N}_{\omega'=3\pi/4}} < 0.01$$

$$100 < 1+\left[\tan\left(\frac{3\pi}{8}\right)\right]^{2N}$$

$$\log(99) < 2N\log\left[\tan\left(\frac{3\pi}{8}\right)\right]$$

$$5.21 < 2N$$

$$N > 2.6 \Rightarrow \text{เลือก } N = 3$$

3) หาตัวกรองแอนะลอกต้นแบบ

จาก N = 3 ใช้ตาราง 8.1 จะได้โพล คือ $p_1 = e^{j\frac{4\pi}{6}}$, $p_1^* = e^{-j\frac{4\pi}{6}}$, และ $p_2 = -1$

ดังนั้น จะได้ $$H_a(s) = \frac{1}{\left(s - e^{j4\pi/6}\right)\left(s - e^{-j4\pi/6}\right)(s+1)}$$

4) ใช้สูตรตามสมการที่ 8.16 จะหาฟังก์ชั่นถ่ายโอนของตัวกรองดิจิตอล ได้คือ

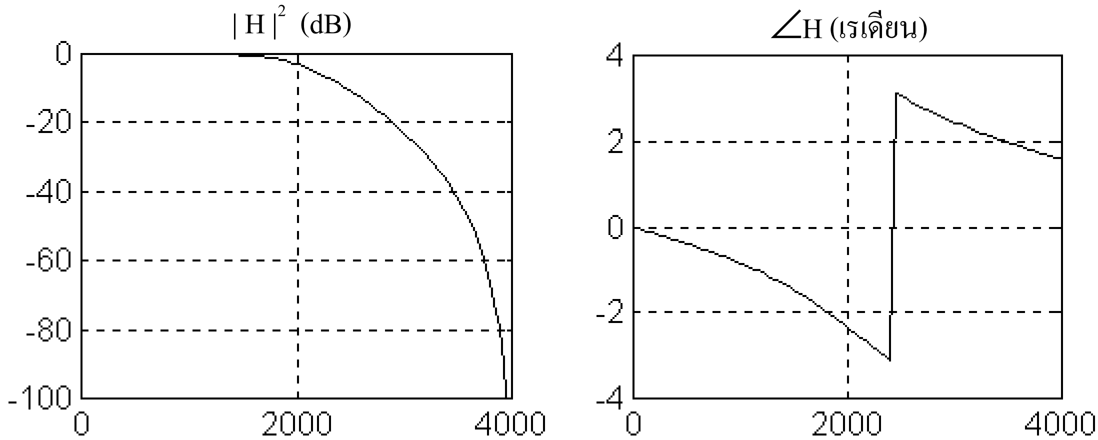
$$H(z) = \frac{G_1(z+1)^2}{z^2 - 2\,\mathrm{Re}\{z_1\}z + |z_1|^2}\;\frac{G_2(z+1)}{z - z_2}$$

โดยที่ $z_1 = \dfrac{K+p_1}{K-p_1} = \dfrac{1+e^{j4\pi/6}}{1-e^{j4\pi/6}} = j0.57735$; $\mathrm{Re}\{z_1\} = 0$, $|z_1| = 0.5773$

$$G_1 = \frac{1}{|K-p_1|^2} = \frac{1}{\left|1-e^{j4\pi/6}\right|^2} = \frac{1}{3}$$

$$z_2 = \frac{K+p_2}{K-p_2} = \frac{1+(-1)}{1-(-1)} = 0$$

$$G_2 = \frac{1}{K-p_2} = \frac{1}{1-(-1)} = \frac{1}{2}$$



รูปที่ 8.5 *ผลตอบสนองเชิงความถี่ของตัวกรองบัตเตอร์เวิร์ธที่ได้จากตัวอย่าง*

แทนค่าทั้งหมดลงในสมการข้างต้น จะได้

$$H(z) = \frac{\frac{1}{3}(z+1)^2}{z^2 + (0.57735)^2} \; \frac{\frac{1}{2}(z+1)}{z} = \frac{1}{6}\frac{(z+1)^3}{z\left(z^2 + \frac{1}{3}\right)}$$

ซึ่งวาดผลตอบสนองเชิงความถี่ของตัวกรองที่ได้ดังในรูปที่ 8.5

การออกแบบตัวกรองบัตเตอร์เวิร์ธแบบอื่น (นอกจากผ่านต่ำ)

การออกแบบโดยการแปลงจากตัวกรองแอนะลอกต้นแบบ ยังสามารถใช้ออกแบบตัวกรองแบบอื่น ๆ ได้แก่ ตัวกรองผ่านสูง, ผ่านแบนด์, และตัดแบนด์ได้ โดยการออกแบบตัวกรองเหล่านี้สามารถทำได้หลายแนวทาง ดังนี้

<u>แนวทางที่ 1</u> แปลงตัวกรองต้นแบบที่เป็นแบบผ่านต่ำให้เป็นแบบอื่นก่อน (ใช้วิธีการแปลงความถี่ในโดเมนของ s) แล้วจึงใช้การแปลงไบลิเนียร์แบบปกติ (LPF $\longrightarrow$ LPF)

<u>แนวทางที่ 2</u> หาตัวกรองดิจิตอลแบบผ่านต่ำ จากตัวกรองแอนะลอกต้นแบบดังที่เราได้ศึกษามา จากนั้นแปลงตัวกรองดิจิตอลผ่านต่ำที่ได้เป็นแบบอื่น (ใช้วิธีการแปลงความถี่ในโดเมนของ z)

<u>แนวทางที่ 3</u> ใช้การแปลงไบลิเนียร์แบบพิเศษ ซึ่งจะแปลงตัวกรองแอนะลอกผ่านต่ำ เป็นตัวกรองดิจิตอลแบบที่ต้องการได้เลย

แนวทางทั้งสามนี้ให้ผลลัพธ์เดียวกัน ในขั้นเบื้องต้นนี้เราจะศึกษาเฉพาะแนวทางที่ 3 เท่านั้น ซึ่งเป็นแนวทางที่รวมเอาการแปลงความถี่จากตัวกรองผ่านต่ำไปเป็นแบบอื่น ไว้ในสูตรเดียวกันกับการแปลงไบลิเนียร์ ดังนั้น จะได้ สูตรสำหรับการแปลงไบลิเนียร์ทั้งสิ้นสี่แบบ ทุกแบบเริ่มต้นโดยใช้ตัวกรอง แอนะลอกต้นแบบเหมือนกัน แต่มีสูตรเพื่อแปลง s เป็น z ต่างกัน ดังที่สรุปไว้ในตารางที่ 8.2 ซึ่งก็จะได้ผลลัพธ์เป็นตัวกรองดิจิตอลตามชนิดที่ต้องการ

การแปลงทั้งสี่แบบให้ผลของเสถียรภาพเหมือนกับการแปลง LPF $\longrightarrow$ LPF ที่ได้กล่าวมา คือ โพลที่อยู่ด้านซ้ายมือ s-plane ของระบบแอนะลอก จะถูกแปลงมาเป็นโพลที่มีขนาดน้อยกว่า 1 ของระบบดิจิตอลเสมอ สังเกตว่าถ้าเป็นการแปลงเป็น BPF หรือ BSF เราจะได้จำนวนโพล และศูนย์เพิ่มขึ้นเป็นสองเท่าด้วย เช่น ถ้าต้องการตัวกรอง BPF อันดับ 4 ก็จะต้องแปลงมาจากตัวกรองแอนะลอกต้นแบบที่มีอันดับ 2 เป็นต้น

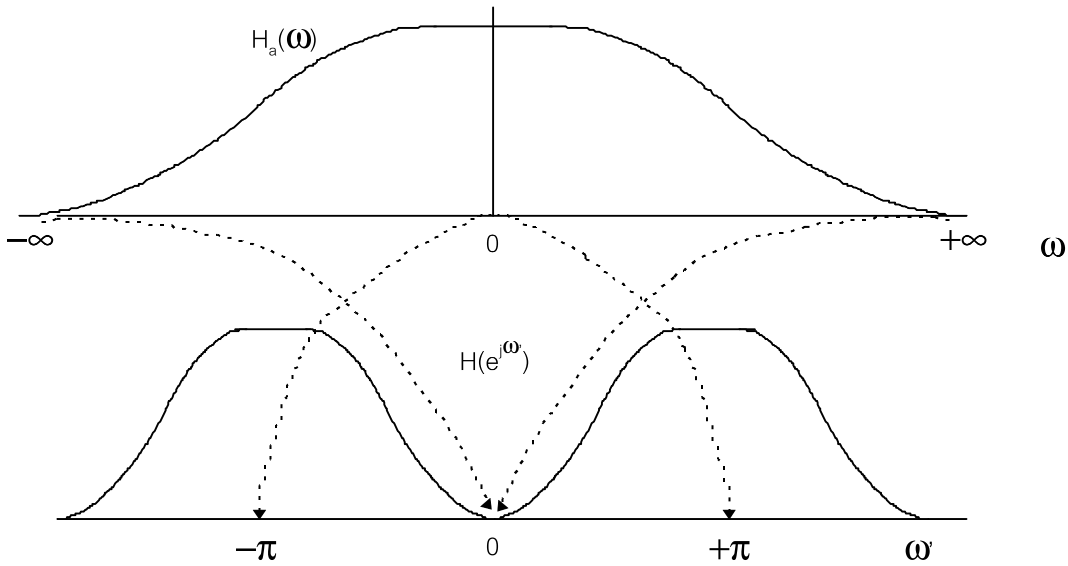
สำหรับผลของการแปลงความถี่ที่เกิดขึ้นจะเป็นไปตามสมการความสัมพันธ์ระหว่าง Ω กับ ω' ดังสรุปไว้ในคอลัมน์ที่สามของตารางที่ 8.2 ซึ่งสมการเหล่านี้ก็หามาจากการแทนค่า $s=j\Omega$ และ $z=e^{j\omega'}$ ลงในสมการของการแปลงไบลิเนียร์ในคอลัมน์ที่สองนั่นเอง ในกรณีที่เรารู้ผลตอบสนองเชิง

ความถี่ของตัวกรองต้นแบบในรูป H(jΩ) เราสามารถหาผลตอบสนองเชิงความถี่ของตัวกรองดิจิตอลได้โดยแทน Ω ด้วยฟังก์ชั่นของ ω' นี้

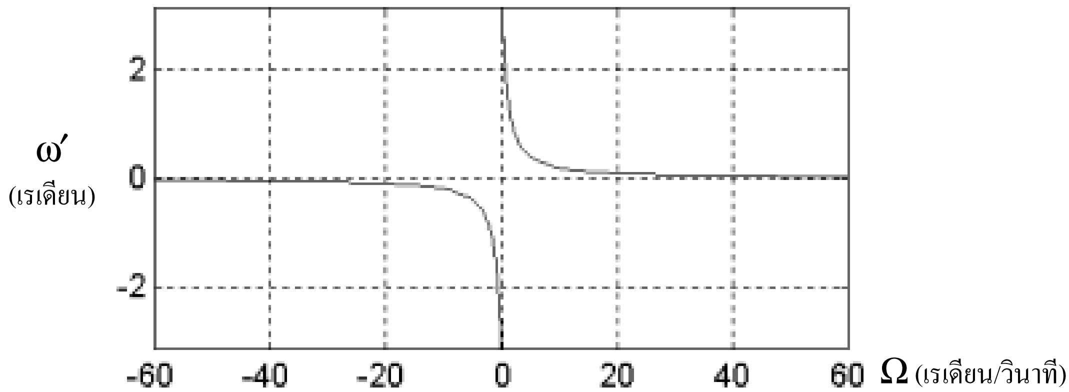
ขอยกตัวอย่าง การพิจารณาการแปลงไบลิเนียร์สำหรับตัวกรองแบบผ่านสูง (LPF $\longrightarrow$ HPF) ซึ่งคือสูตรที่สองในตารางที่ 8.2 จะได้ว่าการแปลงความถี่เป็นไปตามสมการ $\Omega = K\cot(\frac{\omega'}{2})$ พิจารณาสมการนี้จะเห็นว่า เมื่อ $\Omega=\infty$ จะได้ $\omega'= 0$ และเมื่อ $\Omega=0$ จะได้ $\omega'=\pi$ นั่นคือ การแปลงความถี่เกิดขึ้นในลักษณะตรงข้ามกับการแปลงไบลิเนียร์ในกรณีผ่านต่ำ ในกรณีผ่านสูงนี้ ส่วนที่เป็นแถบหยุดจะถูกจับมาอยู่ในช่วงความถี่ต่ำของตัวกรองดิจิตอล และส่วนที่เป็นแถบผ่านจะถูกจับมาอยู่ในช่วงความถี่สูงแทน ซึ่งก็จะได้ผลลัพธ์เป็นตัวกรองดิจิตอลแบบผ่านสูง ถ้าเราสมมติให้ค่า K=1 แล้ววาดรูประหว่าง Ω กับ ω' จะได้ดังรูปที่ 8.7

ชนิดของตัวกรองที่ได้	การแปลง s เป็น z	การแปลงความถี่	พารามิเตอร์ที่ใช้
LPF	$s = K\frac{z-1}{z+1}$	$\Omega = K\tan(\frac{\omega'}{2})$	$K = \cot(\frac{\omega'_c}{2})$
HPF	$s = K\frac{z+1}{z-1}$	$\Omega = K\cot(\frac{\omega'}{2})$	$K = \tan(\frac{\omega'_c}{2})$
BPF	$s = K\frac{z^2+2cz+1}{z^2-1}$	$\Omega = -K\frac{\cos(\omega') - c}{\sin(\omega')}$	$K = \cot\left[(\omega'_2-\omega'_1)/2\right]$ $c = \frac{\cos\left[(\omega'_2+\omega'_1)/2\right]}{\cos\left[(\omega'_2-\omega'_1)/2\right]}$
BSF	$s = K\frac{z^2-1}{z^2+2cz+1}$	$\Omega = K\frac{\sin(\omega')}{\cos(\omega') - c}$	$K = \tan\left[(\omega'_2-\omega'_1)/2\right]$ $c = \frac{\cos\left[(\omega'_2+\omega'_1)/2\right]}{\cos\left[(\omega'_2-\omega'_1)/2\right]}$

ตารางที่ 8.2 *สรุปสูตรของการแปลงไบลิเนียร์ ทั้ง 4 แบบ*
(ω'_1 และ ω'_2 หมายถึง ความถี่ตัดดิจิตอลด้านต่ำและด้านสูงของแบนด์ตามลำดับ)



รูปที่ 8.6 การแปลงระหว่างความถี่แอนะลอก และดิจิตอล ในการแปลงไบลิเนียร์ แบบ LPF $\longrightarrow$ HPF



รูปที่ 8.7 กราฟแสดงความสัมพันธ์ระหว่างความถี่แอนะลอก และดิจิตอล ในการแปลงไบลิเนียร์ แบบ LPF $\longrightarrow$ HPF

<u>ตัวอย่างที่ 8.2</u> ออกแบบตัวกรองบัตเตอร์เวิอร์ธแบบผ่านสูงที่มีความถี่ตัดที่ 2 kHz และใช้ความถี่ในการสุ่มเท่ากับ 8 kHz ให้ระบบมีอัตราการลดทอนไม่ต่ำกว่า 20 dB ที่ 1.5 kHz

หาค่า K , $K = \tan\left(\dfrac{\omega'_c}{2}\right)$, แทนค่า $\omega'_c = \dfrac{2\pi f_c}{f_s} = \dfrac{\pi}{2}$

$$K = \tan\left(\frac{\pi}{2}.\frac{1}{2}\right) = \tan\left(\frac{\pi}{4}\right) = 1$$

จากข้อกำหนดว่า ลดทอน 20dB หรือ คือ กำลังขยาย = -20 dB = 0.01 ที่ความถี่

$$\omega' = \frac{2\pi f}{f_s} = \frac{2\pi(1.5k)}{8k} = \frac{3\pi}{8}$$

จะได้ว่า $|H|^2$ ที่ $\omega' = \frac{3\pi}{8} < 0.01$

(ใช้ $|H|^2$ สำหรับ HFP) $$\frac{1}{1+\left[k\cot(\frac{\omega'}{2})\right]^{2N}_{\omega'=\frac{3\pi}{8}}} < 0.01$$

$$\frac{1}{1+\left[\cot(\frac{3\pi}{16})\right]^{2N}} < 0.01$$

$$100 < 1+\left[\cot(\frac{3\pi}{16})\right]^{2N}$$

$$\log(99) < 2N\log\left[\cot(\frac{3\pi}{16})\right]$$

$$N > \frac{\log(99)}{2\log(\cot\frac{3\pi}{16})} = 5.7$$ เลือก N = 6

ตัวกรองต้นแบบ บัตเตอร์เวิร์ธ n=6 มีฟังก์ชั่นถ่ายโอน คือ $H(s) = \frac{1}{\prod_{i=1}^{N}(s-p_i)}$

โดยที่โพลมี 6 ค่า คือ $\{e^{j\frac{7\pi}{12}} \quad e^{-j\frac{7\pi}{12}} \quad e^{j\frac{9\pi}{12}} \quad e^{-j\frac{9\pi}{12}} \quad e^{j\frac{11\pi}{12}} \quad e^{-j\frac{11\pi}{12}}\}$ (จากตารางที่ 8.1)

ใช้การแปลงไบลิเนียร์ สำหรับ LPF $\longrightarrow$ HPF จะได้

$$H(z) = H(s)\Big|_{s=k\frac{z+1}{z-1}} = H(s)\Big|_{s=\frac{z+1}{z-1}}$$

จากนั้นก็เป็นการจัดการทางคณิตศาสตร์ เพื่อให้ได้ H(z) ในรูปที่จะนำไปใช้งานได้ ซึ่งสามารถคิดด้วยมือ หรือใช้โปรแกรมคอมพิวเตอร์ช่วยได้ ขอละไม่แสดงวิธีทำในที่นี้ โดยผลตอบสุดท้ายที่ได้ คือ

$$H(z) = \frac{0.0296z^6 - 0.1775z^5 + 0.4438z^4 - 0.5918z^3 + 0.4438z^2 - 0.1775z + 0.0296}{z^6 + 0.7777z^4 + 0.1142z^2 + 0.0018}$$

การออกแบบโดยวิธีวางโพล และศูนย์

การออกแบบโดยวิธีวางโพลและศูนย์ เป็นวิธีอย่างง่าย ๆ โดยอาศัยความเข้าใจในเรื่องผลของโพล และศูนย์ที่มีต่อผลตอบสนองเชิงความถี่ของระบบ สามารถนำไปใช้ออกแบบตัวกรองที่มีอันดับต่ำ ๆ และตัวกรองประเภทผ่านความถี่เดียว (peaking filter) หรือตัดความถี่เดียว (notching filter) ได้

ก่อนอื่นต้องมีความเข้าใจเรื่องผลของโพล และศูนย์ที่มีต่อผลตอบสนองเชิงความถี่ก่อน โดยลองพิจารณาระบบที่มีโพลอยู่ที่ p_1 และมีศูนย์อยู่ที่ q_1 ดังนี้

$$H(z) = A\frac{z - q_1}{z - p_1}, \quad \text{โดยที่ } |p_1|, |q_1| \leq 1 \tag{8.17}$$

ถ้าพิจารณาผลตอบสนองความถี่ทางขนาดจะได้ว่า $|H(z)|$ ประกอบด้วยเทอม $|z-q_i|$ เป็นตัวคูณ และเทอม $|z-p_i|$ เป็นตัวหาร ซึ่งเป็นจริงสำหรับจำนวนโพล และศูนย์ใด ๆ ในที่นี้มีโพล และศูนย์อย่างละหนึ่งตัว ซึ่งจะได้

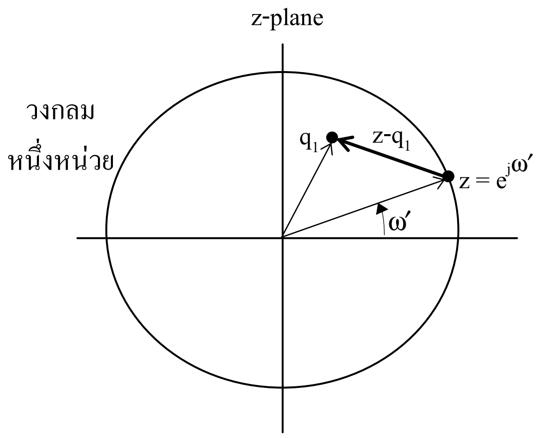
$$|H(z)| = |A|\frac{|z - q_1|}{|z - p_1|} \tag{8.18}$$

ถ้าเขียน p_1, q_1 ให้อยู่ในรูปโพลา ดังนี้ $p_1=|p_1|e^{j\theta_p}$ และ $q_1=|q_1|e^{j\theta_q}$ จะได้ว่า

1. ลองพิจารณาจากเวคเตอร์ $z-q_1$ ซึ่งเป็นเวคเตอร์ที่วาดสีเข้มในรูปที่ 8.8 เมื่อความถี่เปลี่ยนไป z ก็จะหมุนไปรอบ ๆ วงกลมหนึ่งหน่วย และ $z-q_1$ ก็จะเปลี่ยนไปด้วย และจะเห็นได้ชัดว่าขนาดของเวคเตอร์ $z-q_1$ จะมีค่าน้อยที่สุดเมื่อ z หมุนมาทับกับ q_1 พอดี หรือคือจุดที่ความถี่ ω' = มุมของ q_1

สรุปได้ว่า ที่ ω' = มุมของศูนย์ = θ_q หรือ $z=e^{j\theta_q}$ จะได้ $|z-q_1|$ มีค่าน้อยที่สุด ซึ่งเทอมนี้เป็นตัวคูณอยู่ในผลตอบสนองเชิงความถี่โดยรวม ดังนั้น ถ้าพิจารณาผลของศูนย์ตัวเดียว (ในกรณีที่ไม่มีผลของโพล และศูนย์อื่นรบกวน) จะพบว่า ขนาดของผลตอบสนองเชิงความถี่จะถูกดึงให้ต่ำสุดที่ความถี่ตรงกับมุมของศูนย์นั้น ๆ ยิ่งขนาดของศูนย์ $|q_1|$ มีค่าใหญ่ (ใกล้ 1) มากขึ้นเท่าใด ผลของศูนย์ก็จะแรงมากขึ้นเท่านั้น โดยผลแรงที่สุดเกิดขึ้นเมื่อ ถ้า $|q_1| = 1$ ซึ่งจะได้ $|H| = 0$ ที่ความถี่ θ_q นี้

2. ที่ ω' = มุมของโพล = θ_p หรือ $z=e^{j\theta_p}$ จะได้ $|z-p_1|$ มีค่าเป็นจุดต่ำสุด แต่เนื่องจากเทอมนี้เป็นตัวหารอยู่ในผลตอบสนองเชิงความถี่ ดังนั้น มันจะส่งผลให้ $|H|$ มีค่าเป็นจุดสูงสุด (ในกรณีที่ไม่มีผลของโพล และศูนย์อื่นรบกวน) และในทำนองเดียวกัน คือ ถ้าขนาดของโพล $|p_1|$ มีค่าใหญ่ (ใกล้ 1) มากขึ้น ก็จะทำให้ผลของโพลแรงขึ้นที่ความถี่นั้น ถ้า $|p_1| = 1$ จะได้ $|H| = \infty$ ซึ่งเป็นสภาวะที่เกือบเสถียร ในระบบปกติเราจะไม่ใช้ขนาดของโพลเท่ากับ 1 ยกเว้น ระบบที่ต้องการออกแบบเป็นตัวกำเนิดสัญญาณซายน์



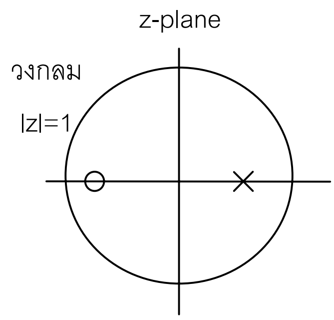
รูปที่ 8.8 *เวคเตอร์ $z\text{-}q_1$ ใน z-plane*

ตัวอย่างของการพิจารณาผลของโพล และศูนย์ เช่น

$$H(z) = A\frac{z + 0.8}{z - 0.6}$$

มีโพล = 0.6 (ω'=0),

มีศูนย์ = -0.8 = $0.8e^{j\pi}$ ($\omega'=\pi$)

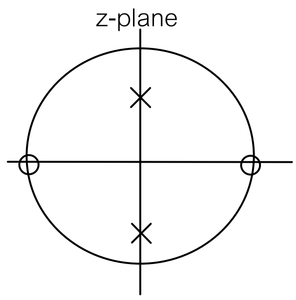


เมื่อพิจารณาในช่วงความถี่ 0 ถึง π ระบบนี้ควรมีขนาดของผลตอบสนองความถี่สูงสุดที่ความถี่ 0 และต่ำสุดที่ความถี่ π ดังนั้น ระบบนี้เป็นวงจรผ่านต่ำ แผนภาพที่แสดงในด้านขวานี้ เรียกว่า แผนภาพโพล-ศูนย์ ซึ่ง O แทนตำแหน่งของศูนย์ และ × แทนตำแหน่งของโพล

$$H(z) = A\frac{(z - 1)(z + 1)}{(z - re^{j\pi/2})(z - re^{-j\pi/2})}$$

มีโพลที่ $\omega'=\pi/2$ และ $-\pi/2$,

มีศูนย์ที่ ω'=0 และ π ดังนั้น ควรเป็นวงจรผ่านแบนด์ ที่มีความถี่ศูนย์กลางของแบนด์อยู่ที่ $\omega'=\pi/2$



<u>ตัวอย่างที่ 8.3</u> จงออกแบบวงจรผ่านต่ำ ที่มีผลตอบสนองเชิงความถี่เป็นศูนย์ที่ 1 kHz และเป็น 1 ที่ 0 kHz ให้มี $|H| = \frac{1}{4}$ ที่ 500Hz, $f_s = 2$ kHz

วงจรผ่านความถี่ต่ำนี้ต้องมีโพลอยู่ที่มุมศูนย์ และมีศูนย์อยู่ที่มุม π

f=1kHz $\rightarrow \omega' = \pi$ ต้องการ $|H| = 0$ เพราะฉะนั้น ศูนย์ = $1e^{j\pi} = -1$

f=0kHz $\rightarrow$ $\omega' = 0$ ต้องการ $|H| = 1$ สมมุติ โพล $= \gamma e^{j0} = \gamma$

f=500Hz $\rightarrow$ $\omega' = \pi/2$ ต้องการ $|H|=1/4$

จะได้รูปทั่วไปของ H(z) เป็น $H(z) = A\dfrac{z+1}{z-\gamma}$ ขั้นตอนต่อไป จะต้องหาค่า A และ γ จากเงื่อนไขของผลตอบสนองเชิงความถี่ข้างต้น

ผลตอบสนองเชิงความถี่ที่ได้ คือ $H(e^{j\omega}) = A\dfrac{e^{j\omega'}+1}{e^{j\omega'}-\gamma}$

เงื่อนไข 1; $|H|_{\omega'=0} = 1 = \left|A\dfrac{(1+1)}{1-g}\right|$ จะได้ $1-\gamma=2A$ -------- (a)

เงื่อนไข 2; $|H|_{\omega'=\frac{\pi}{2}} = \dfrac{1}{4} = \left|\dfrac{A(e^{j\frac{\pi}{2}}+1)}{e^{j\frac{\pi}{2}}-\gamma}\right|$ ------- (b)

แก้สมการ (a), (b) จะได้ A=0.2052, และ γ=0.5896

ดังนั้น $H(z) = \dfrac{0.2052(z+1)}{z-0.5896}$

การออกแบบโดยวางโพล และศูนย์ยังสามารถใช้ออกแบบ ตัวกรองประเภทกรองความถี่เดี่ยว โดยการวางโพล และศูนย์ที่มีขนาดใกล้เคียงกันไว้ความถี่เดียวกัน ถ้าให้ขนาดของโพลมีขนาดใหญ่กว่าศูนย์จะได้เป็นวงจรขยายความถี่เดียว เนื่องจากผลของโพล และศูนย์จะชดเชยกันที่ความถี่อื่น ๆ แต่ที่ความถี่ที่ตรงกับมุมของมัน โพลจะมีผลแรงกว่าศูนย์จึงทำให้เกิดยอดขึ้นมาที่ความถี่นั้น ในทางตรงกันข้าม ถ้าขนาดขนาดโพลมีขนาดเล็กกว่าศูนย์ ก็จะได้เป็นวงจรตัดความถี่เดี่ยว

ตัวกรองขยาย หรือตัดความถี่เดี่ยวจึงมีโพล และศูนย์อย่างละหนึ่งคู่ และได้เป็นฟังก์ชั่นถ่ายโอนอันดับสอง ดังนี้

$$H(z) = K\frac{(z - r_z e^{j\omega_0'})(z - r_z e^{-j\omega_0'})}{(z - r_p e^{j\omega_0'})(z - r_p e^{-j\omega_0'})} \qquad (8.19)$$

ฟังก์ชั่นถ่ายโอนนี้สามารถคำนวณโดยใช้ Matlab ตามโปรแกรมที่ 8.1 ซึ่งจะรับค่าความถี่ที่นอร์แมลไลซ์ด้วย fs/2 เหมือนฟังก์ชั่นที่คำนวณสัมประสิทธิ์อื่น ๆ ใน DSP Toolbox และรับค่า r_z, r_p, และ K ตามสมการที่ 8.19 จากนั้นคำนวณหาค่าสัมประสิทธิ์ของโพลิโนเมียลเศษ และส่วน โดยใช้ฟังก์ชั่น zp2tf ซึ่งเป็นฟังก์ชั่นภายใน DSP Toolbox

```
function [a,b] = singfreq(fn,rz,rp,k)
w=fn*pi;
z=[rz*exp(j*w); rz*exp(-j*w)];
p=[rp*exp(j*w); rp*exp(-j*w)];
[a,b]=zp2tf(z,p,k);
```

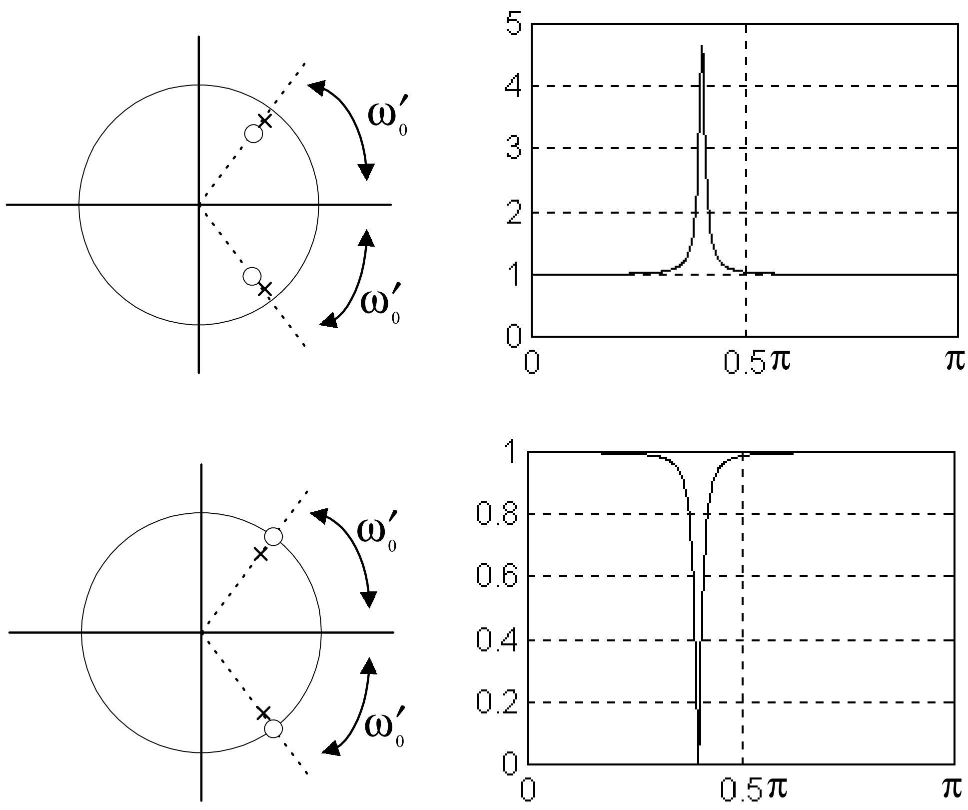
โปรแกรมที่ 8.1 ฟังก์ชั่น singfreq.m สำหรับคำนวณฟังก์ชั่นถ่ายโอนของตัวกรองความถี่เดี่ยว

ตัวอย่างผลตอบสนองเชิงความถี่ที่ได้จากตัวกรองแบบนี้แสดงอยู่ในรูปที่ 8.9 โดยทั้งสองรูปใช้ $\omega_0' = 0.4\pi$

รูปที่ 8.9 ก) ใช้ $r_z = 0.91$, $r_p = 0.98$, และ $K = 1.07$

รูปที่ 8.9 ข) ใช้ $r_z = 1.0$, $r_p = 0.95$, และ $K = 0.95$

ตัวกรองความถี่เดี่ยวนี้ยังสามารถประยุกต์ทำเป็นตัวกรองที่ขยาย หรือตัดหลาย ๆ ค่าความถี่ได้ โดยการนำเอาฟังก์ชั่นถ่ายโอนอันดับสองที่ตัดความถี่เดี่ยวมาอนุกรมกันเป็นอันดับที่สูงขึ้น ตัวอย่างของตัวกรองที่ได้ เช่น ตัวกรองตัด หรือขยายสัญญาณฮาร์มอนิกส์ ซึ่งสัญญาณฮาร์มอนิกส์ประกอบด้วยความถี่เดี่ยวหลาย ๆ ความถี่ (f_0, $2f_0$, $3f_0$, $4f_0$, …) รวมกันอยู่



รูปที่ 8.9 *แผนภาพโพลและศูนย์ และผลตอบสนองเชิงความถี่ของวงจรกรองความถี่เดี่ยว*

การสร้างตัวกรอง IIR (IIR Filter Realization)

ตัวกรองแบบ IIR สามารถนำไปสร้างใช้งานได้โดยมองจาก H(z) ได้โดยตรง เราจะดูวิธีเขียนแผนภาพการสร้างตัวกรองจาก H(z) โดยเริ่มจากการจัด H(z) ให้อยู่ในรูปดังต่อไปนี้ (เพื่อให้ง่ายต่อความเข้าใจ ขอแสดงโดยการใช้อันดับ = 2 โดยอันดับที่สูงขึ้นสามารถทำได้โดยง่ายเมื่อเข้าใจวิธีทำแล้ว)

$$H(z) = \frac{a_0 z^2 + a_1 z + a_2}{z^2 + b_1 z + b_2} \quad \text{หรือ} \quad H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} \qquad (8.20)$$

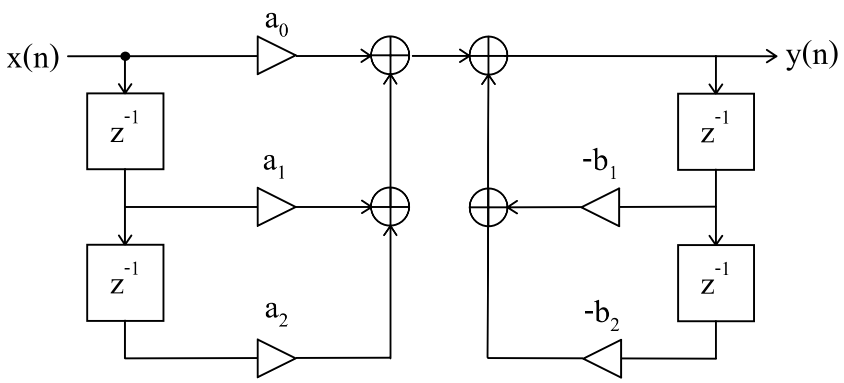
แผนภาพรูปแบบแรก คือ แบบ direct form 1 ดังแสดงในรูปที่ 8.10 รูปแบบนี้เกิดโดยตรงจากสมการผลต่างของระบบ ถ้าเราแทน H(z)=Y(z)/X(z) และแก้สมการเพื่อหาสมการผลต่างของระบบ จะได้ดังนี้

$$Y(z)\,\{1 + b_1 z^{-1} + b_2 z^{-2}\} = X(z)\,\{a_0 + a_1 z^{-1} + a_2 z^{-2}\}$$

$$Y(z) = X(z)\,\{a_0 + a_1 z^{-1} + a_2 z^{-2}\} - Y(z)\,\{b_1 z^{-1} + b_2 z^{-2}\}$$

$$y(n) = a_0 x(n) + a_1 x(n-1) + a_2 x(n-2) - b_1 y(n-1) - b_2 y(n-2) \qquad (8.21)$$

ซึ่งก็พบว่า y(n) เกิดจากส่วนที่มาจากสัญญาณขาเข้าที่เวลาปัจจุบันและอดีต คือ $a_0x(n) + a_1x(n-1) + a_2x(n-2)$ ลบด้วยส่วนที่มาจากสัญญาณขาออกในอดีต คือ $b_1y(n-1) - b_2y(n-2)$ ซึ่งก็ตรงกับที่แสดงในแผนภาพ



รูปที่ 8.10 *แผนภาพการสร้างตัวกรอง IIR (direct form 1)*

อีกรูปแบบหนึ่งของการสร้างตัวกรอง IIR คือ แบบ direct form 2 หรือ canonical form ดังแสดงในรูปที่ 8.11 การพิสูจน์ที่มาของแผนภาพนี้ทำได้โดย สมมติสัญญาณขึ้นใหม่สัญญาณหนึ่งเรียก

ว่า w(n) ซึ่งเกิดการเอา x(n) ผ่านตัวส่วนของฟังก์ชั่นถ่ายโอน หรือเขียนเป็นรูปแบบฟังก์ชั่นโอนย้ายได้ว่า

$$\frac{W(z)}{X(z)} = \frac{1}{1+b_1z^{-1}+b_2z^{-2}} \tag{8.22}$$

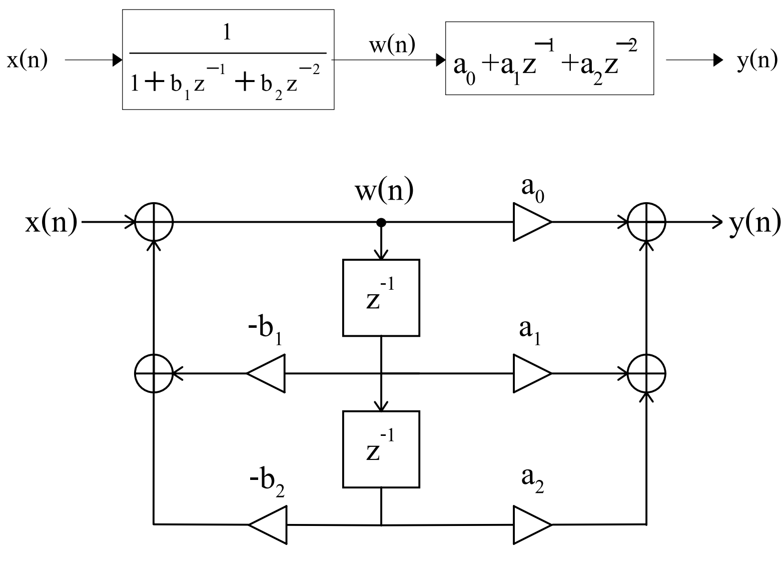
ซึ่งเมื่อทำการแปลง z ผกผันก็จะได้ว่า $w(n) = x(n) - b_1w(n-1) - b_2w(n-2)$ ซึ่งเมื่อแสดง w(n) ในแผนภาพ และมองดูเฉพาะส่วนครึ่งซ้ายมือของแผนภาพ ซึ่งเหมือนเป็นส่วนที่สร้างสัญญาณ w(n) ขึ้น ก็จะพบว่าตรงกับสมการที่เราเขียนขึ้น

จากนั้น y(n) จะหาได้จากการเอาสัญญาณ w(n) ผ่านตัวเศษของฟังก์ชั่นถ่ายโอน ซึ่งเขียนในรูปแบบของฟังก์ชั่นถ่ายโอนได้ว่า

$$\frac{Y(z)}{W(z)} = a_0+a_1z^{-1}+a_2z^{-2} \tag{8.23}$$

ซึ่งเมื่อทำการแปลง z ผกผันก็จะได้ว่า $y(n) = a_0w(n) + a_1w(n-1) + a_2w(n-2)$ ซึ่งสมการนี้ก็คือส่วนขวามือของแผนภาพนั่นเอง ฉะนั้น โดยรวมแล้วก็จะได้

$$\frac{Y(z)}{X(z)} = \frac{W(z)}{X(z)}\,\frac{Y(z)}{W(z)} = \frac{a_0+a_1z^{-1}+a_2z^{-2}}{1+b_1z^{-1}+b_2z^{-2}}$$ เป็นฟังก์ชั่นโดยรวมตามที่ต้องการ



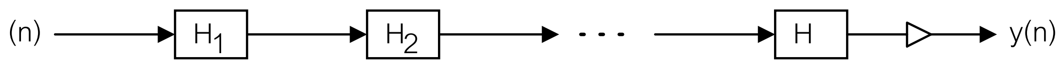
รูปที่ 8.11 *แผนภาพการสร้างตัวกรอง IIR (direct form 2)*

ในกรณีที่ตัวกรองมีอันดับสูง ๆ การสร้างตัวกรองโดยใช้ direct form 1 หรือ 2 ก็สามารถทำได้ แต่อาจส่งผลกระทบเรื่องความคลาดเคลื่อนของระบบ และเสถียรภาพ โดยทั่วไปมักนิยมแตกให้

H(z) อยู่ในรูปของฟังก์ชั่นที่อันดับต่ำ ๆ แล้วใช้รูปแบบการสร้างตัวกรองแบบอนุกรม (cascade form) หรือขนาน (parallel form) เข้ามาช่วย

รูปแบบอนุกรมแสดงอยู่ในรูปที่ 8.12 ทำได้โดยกระจาย H(z) ให้อยู่ในรูปของผลคูณของ $H_i(z)$ ก่อน ซึ่งนิยมใช้ $H_i(z)$ เป็นแบบ อันดับ=2 ยกเว้นถ้าอันดับรวมเป็นเลขคี่ก็จะมี $H_i(z)$ ตัวหนึ่งที่เป็นอันดับ=1 ส่วนที่เป็น $H_i(z)$ ในแผนภาพก็สามารถใช้ direct form 1 หรือ 2 ในการสร้างได้

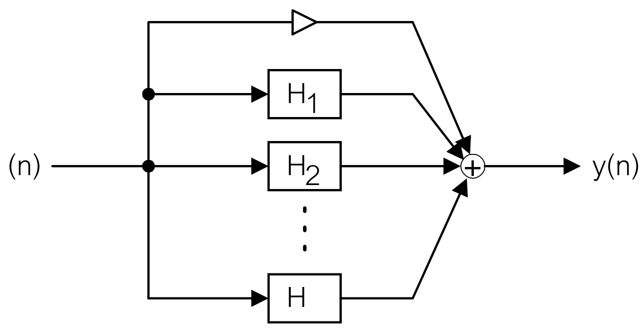
$$H(z) = c\, H_1(z)\, H_2(z) \ldots H_m(z) \tag{8.24}$$



รูปที่ 8.12 *แผนภาพการสร้างตัวกรอง IIR โดยใช้โครงสร้างแบบอนุกรม*

เช่นเดียวกัน รูปแบบขนานแสดงอยู่ในรูปที่ 8.13 ทำได้โดยกระจาย H(z) ให้อยู่ในรูปของผลบวกของ $H_i(z)$ ก่อน ซึ่งนิยมใช้ $H_i(z)$ เป็นแบบอันดับ 2 ยกเว้นถ้าอันดับรวมเป็นเลขคี่ก็จะมี $H_i(z)$ ตัวหนึ่งที่มีอันดับ 1

$$H(z) = c + H_1(z) + H_2(z) + \ldots + H_m(z) \tag{8.25}$$



รูปที่ 8.13 *แผนภาพการสร้างตัวกรอง IIR โดยใช้โครงสร้างแบบขนาน*

ตัวอย่างที่ 8.5 จากฟังก์ชั่นถ่ายโอนที่ได้ในตัวอย่างที่ 8.1 จงเขียนแผนภาพการสร้างตัวกรอง IIR ทั้งแบบ direct form 2, แบบอนุกรม, และขนาน (ใช้ canonical form ในแต่ละส่วนของแบบอนุกรมและขนาน)

จากฟังก์ชั่นถ่ายโอนในตัวอย่างที่ 8.1 กระจายตัวเศษ และส่วนออกเป็นโพลิโนเมียล จะได้

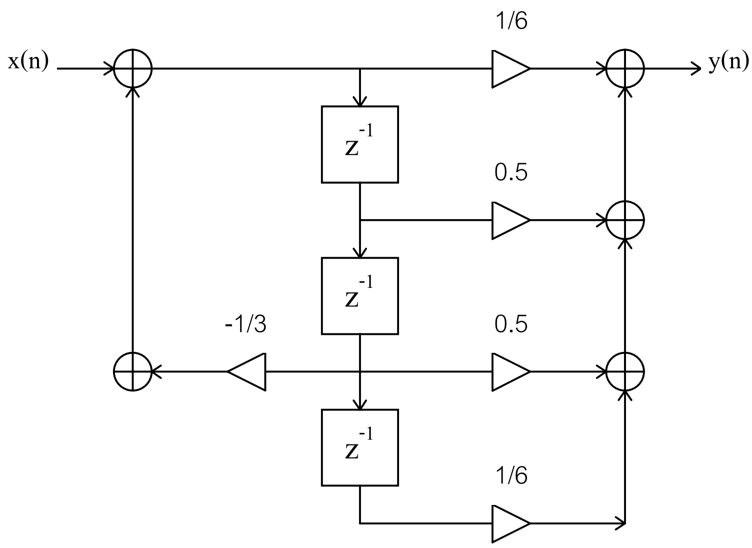
$$H(z) = \frac{1}{6}\frac{(z+1)^3}{z\left(z^2+\frac{1}{3}\right)}$$

$$= \frac{1}{6}\frac{(z^3+3z^2+3z+1)}{\left(z^3+\frac{z}{3}\right)}$$

$$= \frac{z^3/6+0.5z^2+0.5z+1/6}{z^3+z/3}$$

$$= \frac{1/6+0.5z^{-1}+0.5z^{-2}+1/6z^{-3}}{1+1/3z^{-2}}$$

เราสามารถเขียนแผนภาพการสร้าง IIR ได้โดยมองจากสัมประสิทธิ์ของโพลิโนเมียลเศษ และส่วน ดังนี้ $\{a_0 \ \ a_1 \ \ a_2 \ \ a_3\} = \{1/6 \ \ 0.5 \ \ 0.5 \ \ 1/6\}$ และ $\{b_1 \ \ b_2 \ \ b_3\} = \{0 \ \ 1/3 \ \ 0\}$



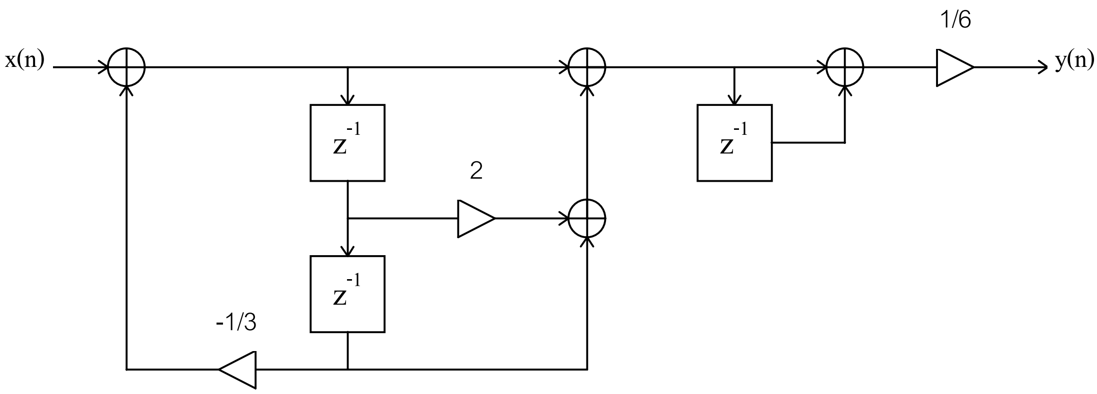
***รูปที่ 8.14** แผนภาพแบบ direct form 2*

สำหรับโครงสร้างแบบอนุกรม ทำได้โดยการกระจาย H(z) ให้อยู่ในรูปของผลคูณของเทอมที่มีอันดับไม่เกิน 2 ดังนี้

$$H(z) = \frac{1}{6}\frac{(z+1)^3}{z\left(z^2+\frac{1}{3}\right)}$$

$$= \frac{1}{6}\ \frac{(z+1)}{z}\ \frac{(z^2+2z+1)}{\left(z^2+\frac{1}{3}\right)}$$

$$= \frac{1}{6}\ (1+z^{-1})\ \frac{(1+2z^{-1}+z^{-2})}{\left(1+\frac{1}{3}z^{-2}\right)}$$



รูปที่ 8.15 *แผนภาพแบบอนุกรม*

สำหรับแบบขนาน ทำได้โดยการกระจาย H(z) ให้อยู่ในรูปของผลบวกของเทอมที่มีอันดับไม่เกิน 2 ดังนี้

$$H(z) = \frac{1}{6}\frac{(z+1)^3}{z\left(z^2+\frac{1}{3}\right)}$$

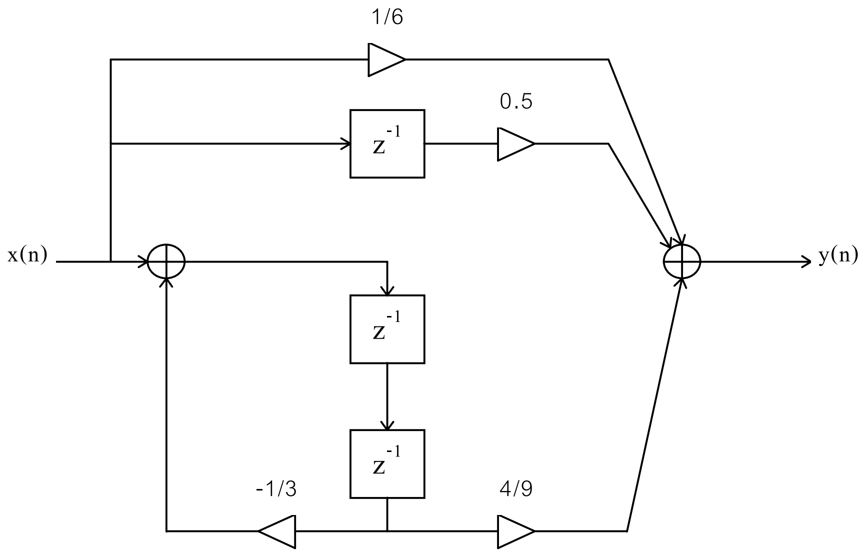
$$= \frac{1}{6z}\frac{(z^3+3z^2+3z+1)}{\left(z^2+\frac{1}{3}\right)}$$

ใช้วิธีหารยาวหาค่าของ $\frac{(z^3+3z^2+3z+1)}{\left(z^2+\frac{1}{3}\right)}$ ดังนี้

$$\begin{array}{r|l}
 & z + 3 \\
\hline
z^2 + 1/3 & z^3 + 3z^2 + 3z + 1 \\
 & \underline{z^3 \qquad + 1/3z} \\
 & \qquad 3z^2 + 8/3z \\
 & \qquad \underline{3z^2 \qquad + 1} \\
 & \qquad\qquad 8/3z
\end{array}$$

ดังนั้น $$\frac{(z^3+3z^2+3z+1)}{\left(z^2+\frac{1}{3}\right)} = z + 3 + \frac{8/3z}{z^2 + 1/3}$$

$$H(z) = \frac{1}{6z}\left(z + 3 + \frac{8/3z}{z^2 + 1/3}\right) = 1/6 + 0.5z^{-1} + \frac{4/9z^{-2}}{1 + 1/3z^{-2}}$$



รูปที่ 8.16 *แผนภาพแบบขนาน*

เปรียบเทียบตัวกรอง FIR กับ IIR

การเลือกใช้ตัวกรอง FIR หรือ IIR สำหรับงานหนึ่ง ๆ เราต้องพิจารณาถึงข้อดี และข้อเสียของตัวกรองทั้งสองแบบ ซึ่งสามารถกล่าวโดยสรุปได้ดังนี้

ข้อดีของตัวกรอง FIR เมื่อเทียบกับตัวกรอง IIR

1. ให้ผลตอบสนองเชิงความถี่ที่มีเฟสแบบเชิงเส้นโดยสมบูรณ์ ตลอดช่วงแถบผ่าน ซึ่งข้อนี้ไม่สามารถทำได้โดยตัวกรอง IIR หรือแม้แต่ตัวกรองแอนะลอกใด ๆ (ทำได้แต่ไม่เชิงเส้นสมบูรณ์)

2. ตัวกรอง FIR ไม่มีการป้อนกลับ หรือไม่มีโพล ทำให้มีเสถียรภาพเสมอ

3. ตัวกรอง FIR มีความทนทานดีกว่าต่อความคลาดเคลื่อนจากการประมาณค่าสัมประสิทธิ์และความคลาดเคลื่อนจากการคำนวณ ความคลาดเคลื่อนนี้เกิดมาจากการใช้เลขฐานสองที่มีความยาวจำกัด (finite word length) ในการแทนค่าเลขจริง ๆ ซึ่งจะกล่าวถึงในบทที่ 10 ความคลาดเคลื่อนนี้ส่งผลถึงลักษณะต่าง ๆ ของระบบจากที่ออกแบบไว้ เช่น ความถี่ตัด, รูปร่างของผลตอบสนองความถี่, และรวมถึงเสถียรภาพของระบบด้วย

ข้อดีของตัวกรอง IIR เมื่อเทียบกับตัวกรอง FIR

1. ให้ผลตอบสนองเชิงความถี่ที่ดีกว่ามากในด้านความคม (ขนาดของแถบเปลี่ยนเล็กกว่า และความพลิ้วต่ำกว่า) ที่ขนาดอันดับเท่า ๆ กับตัวกรอง FIR นั่นหมายถึงว่า โดยทั่วไปเราไม่จำเป็นต้องใช้ตัวกรอง IIR ที่มีอันดับสูง ๆ เหมือน FIR ดังนั้น ตัวกรอง IIR จึงมีความต้องการด้านความเร็วของตัวประมวลผลที่น้อยกว่า

2. สามารถออกแบบโดยอิงจากตัวกรองต้นแบบแอนะลอกได้ ถ้ามีตัวกรองแบบแอนะลอกที่เคยใช้งานอยู่แล้ว

บทที่ 9

ระบบตัวเลขในการประมวลผล

ที่ผ่านมาเราได้สมมติว่า ค่าต่าง ๆ ที่นำมาใช้ในการประมวลผล หรือที่เกิดขึ้นจากการประมวลผลเองมีความละเอียดไม่จำกัด และมีย่านที่สามารถแทนค่าได้ไม่จำกัด ซึ่งการคำนวณโดย Matlab ก็ให้ผลใกล้เคียงกับการสมมตินี้ เพราะ Matlab แทนค่าเลขแต่ละตัวด้วยระบบเลขอิงดรรชนี (floating-point number) ขนาด 64 บิต ทำให้มีผลจากความคลาดเคลื่อนต่าง ๆ น้อยมาก แต่ในความเป็นจริง โดยเฉพาะอย่างยิ่งสำหรับการประมวลผลแบบเวลาจริง เราอาจต้องนำการประมวลผลไปใช้งานโดยมีการแทนตัวเลขด้วยจำนวนบิตต่ำ ๆ โดยเฉพาะอย่างยิ่งการนำไปใช้กับระบบเลขจำนวนเต็ม (fixed-point number) ซึ่งเป็นที่นิยมมาก เพราะสร้างได้ง่าย และราคาถูกกว่า แต่ก็มีผลของความคลาดเคลื่อนมากกว่าแบบระบบเลขอิงดรรชนี

ในบทนี้จะได้กล่าวถึงรายละเอียดของระบบเลขจำนวนเต็ม เพื่อเป็นพื้นฐานสำหรับการศึกษาในบทที่ 10 ซึ่งจะกล่าวถึงความคลาดเคลื่อนที่เกิดขึ้น เมื่อเราใช้ระบบเลขจำนวนเต็มในการประมวลผล ส่วนท้ายของบทนี้จะได้แนะนำระบบเลขอิงดรรชนี และดูว่าทำไมมันจึงดีกว่าระบบเลขจำนวนเต็ม

ระบบเลขจำนวนเต็ม (Fixed-Point Number System)

คำว่า fixed-point ถ้าแปลตามตรงควรแปลว่า ระบบตัวเลขที่มีตำแหน่งจุดทศนิยมคงที่ แต่เนื่องจากลักษณะหน้าตาของมัน ที่เมื่อแปลงแล้วมองดูเหมือนเลขจำนวนเต็ม (ไม่มีทศนิยม) ประกอบกับตัวแปรชนิดเลขจำนวนเต็ม หรือ interger ในภาษาต่าง ๆ ก็ใช้การแทนค่าด้วยระบบเลขชนิดนี้ จึงขออนุโลมเรียกว่า ระบบเลขจำนวนเต็ม

ขออธิบายด้วยการยกตัวอย่างว่า สมมติเราจะใช้เลขฐานสองจำนวน 8 บิตแทนค่าของตัวเลข ซึ่งเลข 8 บิตนี้ เริ่มต้นที่ค่า 00000000_2 และสิ้นสุดที่ค่า 11111111_2 มีจำนวนรวมทั้งสิ้น 256 ค่า หรือ 2^8 ค่า ในการนำเลขฐานสองนี้ไปแทนค่าที่จะใช้ ทำได้โดยการสมมติตำแหน่งจุดทศนิยมขึ้นมา ซึ่งตำแหน่งของจุดทศนิยมจะมีผลต่อช่วงของค่าที่จะแทนได้ ดังนี้

กำหนดให้ N คือ จำนวนบิตที่อยู่ก่อนหน้าจุดทศนิยม และ M คือ จำนวนบิตที่อยู่หลังจุดทศนิยม หรือเขียนเป็นสัญลักษณ์ว่า

$$\overset{\longleftarrow\text{ N บิต }\longrightarrow}{X \;\ldots\; X \;\; X}\,.\,\overset{\longleftarrow\text{ M บิต }\longrightarrow}{X \;\; X \;\;\ldots\;\; X}$$ โดย X แทนตำแหน่งของบิต

ค่าที่แต่ละบิตแทน คือ $2^{N-1} \ldots 2^{1} \;\; 2^{0} \quad 2^{-1} \;\; 2^{-2} \;\ldots\; 2^{-M}$

และค่าที่แทนได้ คือ 0 และ นับขึ้นไปทีละ 2^{-M} จนกระทั่งถึง 2^{N}-2^{-M} ซึ่งเป็นค่าตัวสุดท้าย

ตัวอย่างเช่น สำหรับเลข 8 บิต ถ้า N=8, M=0 หรือ จุดทศนิยมอยู่ขวามือสุด จะได้

เขียนเป็นสัญลักษณ์ คือ X X X X X X X X.

ค่าของแต่ละบิต คือ $2^{7}\; 2^{6}\; 2^{5}\; 2^{4}\; 2^{3}\; 2^{2}\; 2^{1}\; 2^{0}$

ค่าที่สามารถแทนได้ คือ 0, 1, 2, …, 255

ถ้า N=0, M=8 หรือจุดทศนิยมอยู่ซ้ายมือสุด จะได้

เขียนเป็นสัญลักษณ์ คือ .X X X X X X X X

ค่าของแต่ละบิต คือ $2^{-1}\; 2^{-2}\; 2^{-3}\; 2^{-4}\; 2^{-5}\; 2^{-6}\; 2^{-7}\; 2^{-8}$

ค่าที่สามารถแทนได้ คือ $0, 2^{-8}, 2(2^{-8}), 3(2^{-8}), \ldots, 1\text{-}2^{-8}$

ซึ่งเท่ากับ 0.00390625, 0.0078125, 0.01171875, …, 0.99609375

ถ้า N=3, M=5 จะได้

เขียนเป็นสัญลักษณ์ คือ X X X . X X X X X

ค่าของแต่ละบิต คือ $2^{2}\;\; 2^{1}\;\; 2^{0}\quad 2^{-1}\; 2^{-2}\; 2^{-3}\; 2^{-4}\; 2^{-5}$

ค่าที่สามารถแทนได้ คือ $0, 2^{-5}, 2(2^{-5}), 3(2^{-5}), \ldots, 2^{3}\text{-}2^{-5}$

ซึ่งเท่ากับ 0.03125, 0.0625, 0.09375, …, 3.96875

เห็นได้ว่า ตำแหน่งของจุดทศนิยมมีผลต่อค่าที่แทนได้ ดังนั้น ในการใช้ระบบเลขจำนวนเต็มเพื่อแทนค่าตัวเลขทศนิยม นอกจากระบุจำนวนบิตที่จะใช้แล้ว ยังต้องบอกว่าใช้จุดทศนิยมอยู่ที่ตำแหน่งไหน หรือบอกว่า N กับ M มีค่าเท่ากับเท่าไร (เมื่อกล่าวถึง N และ M ในบทที่ 9 และ 10 นี้ จะหมายถึงว่า N คือจำนวนบิตที่อยู่หน้าจุดทศนิยม และ M คือจำนวนบิตหลังจุดทศนิยมเสมอ) เมื่อเลือกตำแหน่งจุดทศนิยมแล้ว ให้คงตำแหน่งไว้ที่ตำแหน่งนั้นตลอดการคำนวณหนึ่ง ๆ

ในกรณีที่ค่าที่ต้องการแทนไม่ลงตัวพอดีเป็นค่าที่สามารถแทนได้ เราจะต้องทำการปัดเศษให้เป็นค่าที่ใกล้เคียงกับค่าที่ต้องการแทนที่สุด ซึ่งวิธีทำมีดังนี้ คือ

ถ้าให้ a เป็นค่าที่ต้องการแทน และ $\mathbf{0 \le a < 2^{N}}$ และสมมติให้ a_{fix} คือ ค่าหลังจากเปลี่ยนเป็นรูปแบบจำนวนเต็มแล้ว เราสามารถหา a_{fix} ซึ่งเป็นตัวแทนที่ใกล้ค่าของ a มากที่สุด ดังนี้

$$a_{fix} = \mathrm{round}(\, a \times 2^{M}\,) \qquad (9.1)$$

โดยนิยามให้ round( ) คือ ฟังก์ชั่นในการปัดเศษเป็นจำนวนเต็ม ถ้าค่าที่ใส่ให้มีส่วนทศนิยมที่น้อยกว่า 0.5 ก็จะตัดทิ้ง แต่ถ้ามากกว่าหรือเท่ากับ 0.5 ก็จะปัดเพิ่มเป็น 1

การปัดเศษนี้จะทำให้เกิดความคลาดเคลื่อนเกิดขึ้น โดยค่าที่ a_{fix} แทน จะเป็น $\frac{a_{fix}}{2^M}$ ซึ่งไม่เท่ากับค่า a เริ่มต้นที่ต้องการ (จะเท่ากันในกรณีที่ $a \times 2^M$ ได้ค่าเป็นจำนวนเต็มเท่านั้น) ดังนั้น ในการแปลงเป็นเลขจำนวนเต็ม ค่าความคลาดเคลื่อนที่เกิดขึ้น คือ

$$err = \frac{a_{fix}}{2^M} - a \qquad (9.2)$$

ตัวอย่างที่ 9.1 จงแปลงเลข 0.66 เป็นจำนวนเต็ม 8 บิต โดยทำ 2 กรณี คือ กรณีที่ M=8 และ M=7

กรณี M=8 ในข้อนี้ a = 0.66 จะได้

$$\begin{aligned} a_{fix} &= round(0.66 \times 2^8) \\ &= round(168.96) \\ &= 169 \end{aligned}$$

ค่า a = 169 นี้จะสามารถแทนด้วยเลขฐานสองที่มีจำนวน 8 บิตได้แน่นอน ซึ่งในที่นี้ได้ $a = 10101001_2$ ในข้อต่อ ๆ ไปจะขอย่นย่อโดยตอบเป็นเลขฐานสิบเท่านั้น (โดยปกติเราก็ไม่จำเป็นต้องแปลงเป็นเลขฐานสอง เพราะซอฟท์แวร์ส่วนใหญ่ที่จะต้องนำค่าไปใช้ต่อไป สามารถรับค่าเป็นเลขจำนวนเต็มฐานสิบได้)

ค่า a_{fix} = 169 นี้ เทียบเท่ากับค่าทศนิยมที่แทนอยู่ คือ $a_{fix}/2^8 = 0.66015625$ ซึ่งหมายถึงว่าค่า a=0.66 ทีแรกนั้นไม่สามารถแทนได้พอดีด้วยรูปแบบจำนวนเต็ม 8 บิต แต่ค่าที่ใกล้เคียงที่สุดที่สามารถแทนได้ คือ 0.66015625 ซึ่งตรงกับค่า a_{fix} = 169 ดังนั้นความคลาดเคลื่อนที่เกิดขึ้นในการแทนค่าตัวเลขนี้ ก็คือ

$$err = \frac{a_{fix}}{2^8} - a = 0.00015625$$

กรณี M=7 ในข้อนี้ a = 0.66 จะได้

$$\begin{aligned} a_{fix} &= round(0.66 \times 2^7) \\ &= 84 \end{aligned}$$

และค่าความคลาดเคลื่อนที่เกิดขึ้น คือ

$$err = \frac{a_{fix}}{2^7} - a = -0.00375$$

เห็นได้ว่า กรณี M=7 สามารถแทนตัวเลขได้ในช่วงที่กว้างกว่า คือแทนค่าได้ตั้งแต่ 0 ถึง 2 แต่กรณี M=8 แทนได้เพียง 0 ถึง 1 แต่ทั้งสองกรณีถือว่ามีความละเอียดในการแทนค่าเท่ากัน เพราะใช้จำนวนบิตรวมเท่ากัน กล่าวคือ ถึงแม้กรณี M=7 จะแทนค่าได้ในช่วงใหญ่กว่าเท่าตัว แต่มันก็มีการแบ่งขั้นที่หยาบกว่า คือ มีขนาดของขั้นที่ใหญ่กว่าเท่าตัว การเลือกว่าจะใช้ M เท่ากับเท่าไร จึงเหมาะข้อมูลชุดหนึ่ง ๆ ก็ขึ้นอยู่กับว่าค่าสูงสุด และต่ำสุดในข้อมูลเป็นเท่าไร

ขนาดของขั้น จะมีค่าเท่ากับค่าของบิตขวาสุด (Least Significant Bit) ซึ่งคือ 2^{-M} และความคลาดเคลื่อนที่เกิดขึ้นในกรณีของการปัดเศษโดยวิธี rounding ก็จะมีขอบเขตไม่เกินครึ่งหนึ่งของขั้น นั่นคือ

$$- \text{ขนาดขั้น}/2 < \text{err} < \text{ขนาดขั้น}/2$$

$$\text{หรือ} \quad -2^{M+1} < \text{err} < 2^{M+1} \qquad (9.3)$$

ดังนั้น กรณี M=8 จะแทนค่าได้ในช่วง $0 \le a < 1$ และ err ไม่เกิน $\pm 2^{-9}$

และ กรณี M=7 จะแทนค่าได้ในช่วง $0 \le a < 2$ และ err ไม่เกิน $\pm 2^{-8}$

ระบบเลขจำนวนเต็มแบบมีเครื่องหมาย

ที่ผ่านมาเป็นการใช้ระบบเลขจำนวนเต็นแทนเฉพาะจำนวนบวกเท่านั้น สำหรับการแทนทั้งจำนวนบวก และลบก็สามารถทำได้ โดยมีรูปแบบใหญ่ ๆ สองแบบ คือ

1) <u>แบบ</u> sign-magnitude มีรูปแบบเหมือนเลขบวก แต่เพิ่มบิตพิเศษขึ้นมาอีก 1 บิต (โดยทั่วไปคือบิตซ้ายสุด) เป็นบิตบอกเครื่องหมาย (sign bit) โดยที่เมื่อบิตเครื่องหมายเป็น 0 หมายถึงเลขบวก และถ้าเป็น 1 หมายถึงเลขลบ ขอยกตัวอย่างเลขแบบ sign-magnitude ในกรณีที่ N=0, และ M=7 เมื่อรวมบิตเครื่องหมายอีก 1 บิต จะต้องใช้จำนวนบิตทั้งสิ้น 8 บิต ดังนี้

	sign	M บิต
เขียนเป็นสัญลักษณ์ คือ	X .	X X X X X X X
ค่าของแต่ละบิต คือ	**sign**	2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} 2^{-6} 2^{-7}

ตัวอย่างเช่น

$01010100_2 = 84_{10}$ มีค่าเท่ากับ 0.65625 (จากคำตอบของตัวอย่าง 9.1)

และ $11010100_2 = -84_{10}$ ก็จะมีค่าเท่ากับ -0.65625 เป็นต้น

จะเห็นได้ว่ารูปแบบ sign-magnitude สามารถเข้าใจได้ง่าย โดยเพียงแค่เติมบิตเครื่องหมายลงไปข้างหน้ารูปแบบปกติเท่านั้น อย่างไรก็ตาม รูปแบบนี้ไม่เป็นที่นิยมในการใช้ เนื่องจากการบวกลบเลขทำได้ไม่สะดวก และไม่ดีเท่ารูปแบบ 2’s complement ที่จะได้กล่าวถึงต่อไป

2) <u>แบบ 2's complement</u> เป็นแบบที่นิยมใช้กันมากที่สุด โดยจะกำหนดใช้<u>บิตซ้ายมือสุดมีค่าเป็น -2^{N-1}</u> ซึ่งเป็นบิตเดียวที่มีค่าติดลบ บิตอื่น ๆ มีความหมายเหมือนเดิม ตัวอย่างช่น เลข 2's complement ขนาด 8 บิต ที่มี N=2 และ M=6 จะมีรูปแบบ และค่าของบิตต่าง ๆ ดังนี้

$$\overset{\longleftrightarrow}{\text{N บิต}} \qquad \overset{\longleftrightarrow}{\text{M บิต}}$$

เขียนเป็นสัญลักษณ์ คือ X X . X X X X X X

ค่าของแต่ละบิต คือ $\mathbf{-2^1}\ 2^0\ \ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ 2^{-5}\ 2^{-6}$

ตารางที่ 9.1 แสดงตัวอย่างของค่าที่แทนได้สำหรับกรณีเลข 2's complement ขนาด 8 บิตที่มี N=1, 2, และ 8 ไล่ตั้งแต่ค่าลบเป็นค่าบวก ขอให้สังเกตว่า ขนาดของขั้นในแต่ละกรณียังมีค่าเท่ากับบิตซ้ายมือสุด คือ 2^{-M} เหมือนกับกรณีการแทนเลขบวกอย่างเดียว

ฐานสอง	ฐานสิบ (บวก)	ค่าที่แทน (กรณี N=8)	ค่าที่แทน (กรณี N=1)	ค่าที่แทน (กรณี N=2)
1000 0000	128	-128	-1 (ค่าลบมากสุด)	-2
1000 0001	129	-127	$-1+2^{-7}$ = -0.9921875	$-2+2^{-6}$ = -1.984375
1000 0010	130	-126	$-1+2(2^{-7})$ = -0.984375	$-2+2(2^{-6})$ = -1.96875
…	…	…	…	…
1111 1111	255	-1	-2^{-7} = -0.0078125	-2^{-6} = -0.015625
0000 0000	0	0	0	0
0000 0001	1	1	2^{-7} = 0.0078125	2^{-6} = 0.015625
0000 0010	2	2	$2(2^{-7})$ = 0.015625	$2(2^{-6})$ = 0.03125
…	…	…	…	…
0111 1111	127	127	$1-2^{-7}$ = 0.9921875 (ค่าบวกมากสุด)	$2-2^{-6}$ = 1.984375

ตารางที่ 9.1 *รูปแบบเลข 2's complement (กรณี 8 บิต) กับค่าที่แทนได้*

จะเห็นได้ว่า บิตซ้ายมือสุดยังสามารถมองเป็นบิตเครื่องหมายได้ เพราะถ้าบิตซ้ายสุดเป็น 1 จะได้เลขจำนวนนั้นเป็นค่าลบแน่นอน และถ้าเป็น 0 ก็จะได้เป็นค่าบวกแน่นอน

ในระบบประมวลผลสัญญาณ นิยมใช้รูปแบบ 2's complement ที่มีจุดทศนิยมหลังบิตซ้ายมือสุด 1 บิต (N=1) ซึ่งจะแทนเลขได้จาก -1 ถึงประมาณ 1 และผลดีก็คือ การคูณเลขสองจำนวนเข้าด้วยกันจะไม่เกิดโอเวอร์โฟล เพราะ ผลลัพธ์ที่ได้จะไม่มีทางเกินช่วง -1 ถึง 1

<u>ตัวอย่างที่ 9.2</u> จงแปลงเลข -0.595 เป็นจำนวนเต็มแบบ 2's complement 8 บิต โดยใช้ M=7

ในกรณีของ 2's complement ถ้าให้ a เป็นค่าที่ต้องการแทน และ $\mathbf{-2^{N-1} \le a < 2^{N-1}}$ สมมติให้ a_{fix} คือ ค่าหลังจากเปลี่ยนเป็นรูปแบบจำนวนเต็มแล้ว เราสามารถหา a_{fix} ซึ่งแทนค่า a ได้ใกล้เคียงที่สุด ด้วยสมการเดียวกับ 9.1 คือ

$$a_{fix} = \text{round}(\, a \times 2^{M}) \qquad (9.1)$$

ในข้อนี้ M=7 และ N=1 ซึ่งจะสามารถแทนค่า a ได้ระหว่าง -1 ถึง 1 ซึ่ง a=-0.595 ดังนั้นจะได้

$$\begin{aligned} a_{fix} &= \text{round}(\, -0.595 \times 2^{7}) \\ &= \text{round}(-76.16) \\ &= -76 \end{aligned}$$

ค่า -76 สามารถแปลงกลับเป็นทศนิยมได้เป็น $-76 \,/\, 2^7 = -0.59375$ ซึ่งมีความคลาดเคลื่อนจากค่า a เท่ากับ

$$\frac{a_{fix}}{2^7} - a = -0.59375 - (-0.595) = 0.00125$$

ซอฟท์แวร์ส่วนใหญ่สามารถรับค่าจำนวนเต็มฐานสิบ (ไม่ว่าจะบวกหรือลบ) นี้ไปใช้งานต่อไปได้ ซึ่งเราก็ไม่จำเป็นต้องแปลงเป็นฐานสองให้ยุ่งยาก แต่ในกรณีที่ต้องการแปลงให้เป็นเลขฐานสองก็สามารถทำได้หลายวิธี ดังนี้ (ในที่นี้คือรูปแบบ 8 บิต)

<u>วิธีที่ 1</u> แยกเลขเป็นส่วนลบและบวก จะได้ส่วนลบจะเป็นบิตซ้ายบิตเดียวมีค่า -2^{-B+1} เสมอ โดยที่ B = จำนวนบิตทั้งหมด และส่วนบวกก็แปลงตามปกติ เช่น $a_{fix} = -76$ แตกเป็นส่วนลบกับบวกได้เป็น $a_{fix} = -128 + 52 = 10000000_2 + 00110100_2 = 10110100_2$

<u>วิธีที่ 2</u> ถ้าเป็นเลขบวกให้แปลงตามปกติ แต่ถ้าเป็นเลขลบให้แปลงเป็นเลขบวกโดยการบวก 2^B เข้าไป <u>ซึ่งมันจะทำหน้าที่เปลี่ยนความหมายของบิตซ้ายสุด จาก -2^{-B+1} เป็น 2^{-B+1}</u> เช่น ในที่นี้ $a_{fix} = -76$ แปลงเป็นเลขบวกโดยบวกด้วย 2^8 เข้าไป จะได้ $a_{fix} = -76 + 2^8 = 180 = 10110100_2$

<u>ตัวอย่างที่ 9.3</u> 11001010_2 มีค่าเท่ากับเท่าไร ถ้าเลขนี้เป็นรูปแบบ 2's complement หาในกรณีที่ให้ M=0, M=6, และ M=7 บอกด้วยว่าในแต่ละกรณีสามารถแทนค่าเลขได้ในช่วงใด

<u>กรณี M=0</u> หรือ ตำแหน่งจุดทศนิยมอยู่ขวามือสุด (XXXXXXXX.)

ใช้แทนค่าเลขได้ในช่วง -128 ถึง 127 (คิดมาจาก -2^{N-1} ถึง $2^{N-1}-2^{-M}$)

ถ้าจะหาว่า 11001010_2 มีค่าเท่ากับเท่าไร อาจคิดแยกเป็นสองส่วน คือส่วนลบ กับบวก จะได้

$$a_{fix} = 10000000_2 + 01001010_2$$

ส่วนลบคือบิตซ้ายสุด จะมีค่าเท่ากับ -2^{N-1} เสมอ ในที่นี้ N=8 ดังนั้น ได้ส่วนลบ คือ -128 และส่วนบวกแปลงเป็นฐานสิบปกติได้เท่ากับ 74 ดังนั้นจะได้

$$a = -128 + 74 = -54$$

<u>กรณี M=6</u> หรือ ตำแหน่งจุดทศนิยมอยู่หลังสองบิตซ้ายมือ

เขียนเป็นสัญลักษณ์ คือ X X . X X X X X X

ค่าของแต่ละบิต คือ $\mathbf{-2}^1\ 2^0\ \ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ 2^{-5}\ 2^{-6}$

ใช้แทนค่าเลขได้ในช่วง -2 ถึงเกือบ 2 (คิดมาจาก -2^{N-1} ถึง $2^{N-1}-2^{-M}$)

จะได้ $a_{fix} = 10000000_2 + 01001010_2$

$$a = (-2) + 2^0 + 2^{-3} + 2^{-5} = -0.84375$$

คำตอบของกรณี M=6 นี้ สามารถคิดมาจากกรณี M=0 ก็ได้ โดยเอาค่าที่ได้จากกรณี M=0 มาหารด้วย 2^M ดังนี้

$$a = -54 / 2^6 = -0.84375$$

<u>กรณี M=7</u> หรือ ตำแหน่งจุดทศนิยมอยู่หลังบิตซ้ายมือสุด (X.XXXXXXX)

ใช้แทนค่าเลขได้ในช่วง -1 ถึงเกือบ 1 (คิดมาจาก -2^{N-1} ถึง $2^{N-1}-2^{-M}$)

จะได้ $a_{fix} = 10000000_2 + 01001010_2$

$$a = (-1) + 2^{-1} + 2^{-4} + 2^{-6} = -0.421875$$

เช่นเดียวกัน สามารถคิดได้อย่างเร็ว โดยเอาค่าที่ได้จากกรณี M=0 มาใช้ ดังนี้

$$a = -54 / 2^7 = -0.421875$$

<u>**ตัวอย่างที่ 9.4**</u> จะแปลงเลขต่อไปนี้ ซึ่งเป็นรูปแบบ sign-magnitude ให้เป็นรูปแบบ 2's complement

ก) 01101001_2

สำหรับเลขบวก ทั้งสองรูปแบบมีค่าเท่ากัน เพราะฉะนั้น รูปแบบ 2’s complement ก็ยังเป็น 01101001_2

ข) 10010110_2

สำหรับเลขลบ สามารถแปลงเป็น 2’s complement ได้โดยนำเลข sign-magnitude มากลับบิตทุกบิตจาก 1 เป็น 0 และจาก 0 เป็น 1 จากนั้นบวกผลลัพธ์ที่ได้ด้วย 1 ดังนี้

10010110_2 กลับบิตทุกบิตจะได้ 01101001_2

จากนั้นบวก 1 ได้ $10010110_2+1 = 11101010_2$ เป็นรูปแบบ 2’s complement ที่ต้องการ

ข้อดีของ 2's complement

รูปแบบ 2's complement เป็นรูปแบบที่ใช้กันทั่วไปในการคำนวณของคอมพิวเตอร์ และในระบบประมวลผลสัญญาณ เมื่อต้องการเลขจำนวนเต็มที่แทนได้ทั้งจำนวนบวกและลบ สาเหตุของการใช้ 2's complement มาจากข้อดีในการคำนวณ 2 ประการ คือ

1) การบวกเลขทำได้เหมือนการบวกปกติโดยไม่ต้องคำนึงถึงจำนวนบวก หรือลบ

ขอยกตัวอย่างง่าย ๆ เพื่อให้เห็นภาพชัด โดยใช้เลข 2's complement ขนาด 4 บิต และ M=0 (ไม่มีส่วนของทศนิยม) จะสามารถแทนค่าได้ในช่วง -8 ถึง 7

<u>ตัวอย่างที่ 1</u> 3 + (-5) = ?

0 0 1 1 → 3
\+ +
1 0 1 1 → -5

1 1 1 0 → -2

จะเห็นได้ว่าการบวกเลขฐานสองที่เกิดขึ้น ทำเหมือนปกติ โดยไม่ต้องสนใจว่าจะเป็นเลขลบหรือบวก ซึ่งจะได้ผลลัพธ์สุดท้ายถูกต้องเอง

<u>ตัวอย่างที่ 2</u> -2 + (-5) = ?

1 1 1 0 → -2
\+ +
1 0 1 1 → -5

~~1~~ 1 0 0 1 → -7

ในกรณีที่เกิดผลลัพธ์บิตแรกขึ้นมา ให้ตัดทิ้งและขอบเขตเฉพาะบิตที่เหลือ ซึ่งจะได้ผลลัพธ์ที่ถูกต้อง กรณีนี้ไม่ถือเป็นโอเวอร์โฟล

<u>ตัวอย่างที่ 3</u> -4 + (-5) = ?

1 1 0 0 → -4
\+ +
1 0 1 1 → -5

~~1~~ 0 1 1 1 → 7 ได้ผลลัพธ์ผิด!

ในกรณีนี้ ผลลัพธ์ที่ถูกต้องจริง ๆ คือ -9 ซึ่งเกินช่วงที่จะแทนค่าได้ และถือเป็นการเกิดโอเวอร์โฟล การตรวจสอบว่าเกิดโอเวอร์โฟลหรือไม่ อาจทำได้โดยตรวจสอบที่บิตเครื่องหมาย ถ้าจำนวนลบบวกกับจำนวนลบแล้วได้เป็นจำนวนบวก เหมือนในตัวอย่างข้างต้น หรือจำนวนบวกบวกกับจำนวนบวกแล้วได้จำนวนลบ ถือว่าเกิดโอเวอร์โฟลขึ้น

2) การบวกเลขหลาย ๆ จำนวน ไม่ต้องคำนึงถึงการเกิดโอเวอร์โฟลในผลลัพธ์กึ่งกลาง ถ้าผลลัพธ์สุดท้ายไม่เกิดโอเวอร์โฟล

ตัวอย่างเช่น 4 + 5 - 3 = ? (กำหนดให้ใช้เลข 2's complement ขนาด 4 บิต และ M=0 เหมือนข้อ 1) ในการคำนวณถ้ากระทำ 4+5 ก่อน จะได้ 9 ซึ่งเกิดโอเวอร์โฟลแล้วเพราะเกินช่วงที่จะแทนค่าได้ แต่ถ้าพิจารณา -3 เข้าไปด้วย จะได้ผลลัพธ์สุดท้ายเป็น 6 ซึ่งอยู่ในช่วงที่แทนได้ การเกิดโอเวอร์โฟลที่ผลลัพธ์กึ่งกลางในลักษณะนี้ สำหรับเลข 2's complement จะไม่มีผลใด ๆ ถ้าผลลัพธ์สุดท้ายไม่เกิดโอเวอร์โฟล โดยให้ทำการบวกต่อไปเรื่อย ๆ เหมือนไม่มีอะไรผิดปกติเกิดขึ้น จนกระทั่งได้ผลลัพธ์สุดท้าย ซึ่งจะได้ผลลัพธ์สุดท้ายถูกต้องเอง ดังแสดงให้เห็นดังต่อไปนี้

$$
\begin{array}{rcl}
0\,1\,0\,0 \; + & \rightarrow & 4 \; + \\
\underline{0\,1\,0\,1} & \rightarrow & \underline{5} \\
1\,0\,0\,1 \; + & \rightarrow & -7 \;\; \text{ได้ผลลัพธ์กึ่งกลางผิด!} \; + \\
\underline{1\,1\,0\,1} & \rightarrow & \underline{-3} \\
\underline{\underline{\cancel{1}\;0\,1\,1\,0}} & \rightarrow & \underline{\underline{6}} \;\; \text{ได้ผลลัพธ์สุดท้ายถูก!}
\end{array}
$$

ถ้าจะพิจารณาว่า ทำไมจึงได้ค่าสุดท้ายถูกต้อง ก็อาจทำได้โดยการแตกค่าลบออกมาเป็นส่วนลบ และส่วนบวก ดังนี้

$$
\begin{array}{rcrl}
4 \; + & \rightarrow & 4 \; + & \multirow{2}{*}{\Big\}\; \text{เกิดโอเวอร์โฟลทางบวก}} \\
\underline{5} & \rightarrow & \underline{5} & \\
-7 \; + & \rightarrow & -8+1 \; + & \Big\}\; \text{เกิดโอเวอร์โฟลทางลบ -8 สองตัวหักล้างกันหายไป เพราะ -8 คือบิตซ้ายสุด ดังนั้นเหลือแค่ 1+ 5 = 6} \\
\underline{-3} & \rightarrow & \underline{-8+5} & \\
\underline{\underline{6}} & \rightarrow & \underline{\underline{6}} &
\end{array}
$$

อาจกล่าวได้ว่า โอเวอร์โฟลทางบวก และโอเวอร์โฟลทางลบชดเชยกันหายไป ทำให้ได้ผลลัพธ์สุดท้ายถูกต้อง คุณสมบัติข้อนี้ของตัวเลข 2's complement มีผลดีมากต่อการประมวลผลสัญญาณ เพราะ มีการประมวลผลหลาย ๆ อย่างที่ใช้การบวกสะสมค่าเพื่อหาค่าผลลัพธ์สุดท้าย ตัวอย่างที่เราได้เห็นมาแล้วก็คือ การหาผลตอบของตัวกรอง FI R เป็นต้น

ระบบเลขอิงดรรชนี (Floating-Point Number System)

ระบบเลขอิงดรรชนี หรือถ้าแปลตามตรงจากคำว่า floating-point หมายถึง ระบบเลขที่มีตำแหน่งจุดทศนิยมลอยตัว คือ ตำแหน่งของจุดทศนิยมไม่ติดอยู่ที่ตำแหน่งเดียวเหมือนอย่างกรณีระบบเลขจำนวนเต็ม โดยมีกลไกที่ใช้ในการชี้ตำแหน่งของจุดทศนิยม คือ การใช้เลขชี้กำลัง ระบบเลขอิงดรรชนีจะแบ่งจำนวนบิตที่จะใช้แทนค่าตัวเลขเป็นสองส่วน คือ แมนทิสซา (mantissa) และเลขชี้กำลัง (exponent)

ผู้อ่านส่วนใหญ่คงเคยเห็น หรือเคยใช้ตัวเลขที่อยู่ในรูปของเลขอิงดรรชนีมาบ้างแล้วไม่มากก็น้อย ตัวอย่างเช่น ตัวเลข 0.003257 เราสามารถเขียนในรูปของเลขอิงดรรชนี (ฐานสิบ) ได้ว่า คือ 3.257×10^{-3} ซึ่งจะได้ว่าเลขนี้มีแมนทิสซา คือ 3.257 และตัวชี้กำลัง คือ -3 ซึ่งรูปแบบนี้ถ้าหากมีคนเขียนว่า ค่านี้เท่ากับ 0.3257×10^{-2} ด้วย ก็ไม่ถือว่าผิด โดยมีหลักง่าย ๆ ว่า ถ้าเลื่อนจุดทศนิยมของแมนทิสซาไปทางซ้าย 1 ตำแหน่ง ต้องเพิ่มค่าตัวชี้กำลังขึ้นหนึ่ง และถ้าเลื่อนจุดทศนิยมของแมนทิสซาไปทางขวา ก็จะต้องลดค่าตัวชี้กำลังลง อย่างไรก็ตาม เรานิยมปรับ หรือนอร์แมลไลซ์ค่าแมนทิสซานี้ให้มีค่าอยู่ระหว่าง 1 ถึง 10 หรือ -1 ถึง -10 เสมอ

ในทำนองเดียวกัน สำหรับเลขอิงดรรชนิของเลขฐานสอง หลักการต่าง ๆ ยังคงเหมือนฐาน 10 เพียงแต่จะใช้ 2 เป็นตัวฐานของเลขยกกำลังแทน ดังนี้

$$\text{ค่าที่แทน} = (\text{แมนทิสซา}) \times 2^{(\text{ตัวชี้กำลัง})} \qquad (9.4)$$

ทั้งค่าแมนทิสซา และตัวชี้กำลังจะถูกแปลงให้เป็นระบบเลขจำนวนเต็ม (ฐานสอง) แบบบวกลบที่มีจำนวนบิตตามรูปแบบที่ต้องการ โดยอาจใช้รูปแบบ 2's complement หรือ sign-magnitude ในการแทนค่าก็ได้ และสำหรับแมนทิสซานิยมนอร์แมลไลซ์ให้มีค่าอยู่ระหว่าง 1 ถึง 2 หรือ -1 ถึง -2 เสมอ ซึ่งก็หมายถึงว่า จุดทศนิยมของแมนทิสซาจะมีตำแหน่งคงที่ คืออยู่หลังบิตที่สองจากซ้ายมือ ส่วนตัวชี้กำลังต้องมีค่าเป็นจำนวนเต็มเสมอ ดังนั้น มีตำแหน่งจุดทศนิยมอยู่ซ้ายมือสุด เช่น ถ้าใช้แมนทิสซาขนาด 13 บิต และตัวชี้กำลังขนาด 4 บิต จะเขียนรูปแบบตัวเลข ได้ดังนี้

$$\text{ค่าที่แทน} = \mathrm{XX.XXXXXXXXXXX} \times 2^{\mathrm{XXXX.}} \qquad (9.5)$$

การนอร์แมลไลซ์แมนทิสซานี้ ทำให้เกิดเหตุการณ์ที่น่าสนใจขึ้น คือ ในกรณีของเลขบวกซึ่งแมนทิสซาจะมีค่ามากกว่า 1 เสมอ ดังนั้นจะได้ว่าบิตที่สองจะต้องมีค่าเป็น 1 เสมอ และในทำนองเดียวกัน กรณีของเลขลบ เราสามารถเขียนรูปแบบของเลขที่จะแทนค่าเป็นเลขอิงดรรชนี โดยแบ่งเป็นกรณี

เลขบวก และลบซึ่งแมนทิสซาจะมีค่าน้อยกว่า 1 เสมอ ดังนั้นจะได้ว่าบิตที่สองจะต้องมีค่าเป็น 0 เสมอ ดังนี้

$$\text{ค่าบวก} = 01.\,XXXXXXXXXXXX \times 2^{XXXX.} \tag{9.5}$$

$$\text{ค่าลบ} = 10.\,XXXXXXXXXXXX \times 2^{XXXX.} \tag{9.6}$$

จะสังเกตเห็นได้ว่า ถ้าเรารู้บิตเครื่องหมาย (คือ บิตที่หนึ่งของแมนทิสซา) บิตที่สองก็จะมีค่าที่แน่นอนโดยปริยาย ดังนั้น จึงนิยมละโดยไม่ต้องเก็บค่าของบิตที่สองในรูปแบบของเลขอิงดรรชนี แต่ให้เป็นที่เข้าใจกันว่ามันเสมือนมีอยู่ ดังนั้น รูปแบบตัวเลขในสมการที่ 9.5 และ 9.6 นี้ สามารถใช้เพียงแค่ 12 บิตเพื่อแทนค่าแมนทิสซาได้ ทั้ง ๆ ที่จริง ๆ แล้วมันมีทั้งสิ้น 13 บิต ทำให้ผลรวมของจำนวนบิตที่ต้องใช้ทั้งหมดในที่นี้ คือ 16 บิต

<u>ตัวอย่างที่ 9.5</u> จงหาค่าของแมนทิสซา และตัวชี้กำลัง เมื่อแปลงค่าเลขต่อไปนี้ให้อยู่ในรูปแบบเลขอิงดรรชนีตามสมการที่ 9.5 และ 9.6 ให้ใช้รูปแบบ 2’s complement สำหรับทั้งแมนทิสซา และตัวชี้กำลัง

ก) -7.687 ข) 0.005628

ก) ถ้าให้ x เป็นค่าที่ต้องการแทน ในที่นี้ คือ x = -7.687 เราจะต้องพยายามเขียน x ให้อยู่ในรูปของแมนทิสซา คูณกับกำลังของสองดังสมการที่ 9.4 สมมติให้ m = แมนทิสซา และ e = ตัวชี้กำลัง นั่นคือ

$$x = m \times 2^{e}$$

เราสามารถหาได้ว่าตัวชี้กำลังจะเป็นเท่าไรล่วงหน้าได้ โดยหา $\log_2$ ของสมการข้างต้น ดังนี้

$$\log_2(|x|) = \log_2(|m \times 2^{e}|)$$
$$\log_2(|x|) = \log_2(|m| \times 2^{e})$$
$$\log_2(|x|) = e + \log_2(|m|)$$
$$e = \log_2(|x|) - \log_2(|m|) \tag{9.7}$$

เนื่องจาก |m| จะต้องมีค่าระหว่าง 1 ถึง 2 ซึ่งจะได้ $\log_2(|m|)$ มีค่าเป็นบวกระหว่าง 0 ถึง 1 ส่วน e จะต้องมีค่าเป็นจำนวนเต็มเสมอ ดังนั้น ค่า e หาได้จากการ<u>ลบ</u>ค่าทศนิยม (ที่น้อยกว่า 1) ทิ้ง

จาก $\log_2(|x|)$ เพื่อให้ได้เป็นค่าจำนวนเต็ม หรือกล่าวได้ว่า e หาได้จากการปัดเศษค่า $\log_2(|x|)$ ไปทางด้านลบ (จะทำให้ได้ e เป็นจำนวนเต็มที่น้อยกว่าหรือเท่ากับ $\log_2(|x|)$) เขียนเป็นสมการได้ว่า

$$e = \text{floor}(\log_2(|x|)) \tag{9.8}$$

โดยที่ floor เป็นการปัดเศษให้เป็นจำนวนเต็มไปทางด้านลบ (เป็นชื่อเดียวกับฟังก์ชั่นใน Matlab ที่ทำหน้าเดียวกันนี้) ในที่นี้จะได้

$$e = \text{floor}(\log_2(7.687)) = \text{floor}(2.94242)$$
$$= 2 = 0010_2$$

จากนั้นค่าแมนทิสซาหาได้จาก

$$m = \frac{x}{2^e} \tag{9.9}$$

ซึ่งในที่นี้จะได้ $m = -7.687 / 2^2 = -1.92175$

จากนั้นแปลงค่า m ให้อยู่ในรูปแบบ 2's complement โดยใช้วิธีที่ได้อธิบายมาแล้วในหัวข้อเรื่องระบบเลขจำนวนเต็ม จะได้ (ในที่นี้ N = 2 และ M=11)

$$m_{fix} = \text{round}(-1.92175 \times 2^{11}) = -3936 = 1000010100000_2$$

จะเห็นได้ว่า บิตที่ 2 (จากซ้าย) เท่ากับ 0 ซึ่งตรงตามรูปแบบของสมการที่ 9.6 ถ้าตัดบิตนี้ทิ้งเราจะได้ส่วนของแมนทิสซาแทนได้ด้วย 12 บิต เป็น 100010100000_2

ข) x = 0.005628 ใช้วิธีเดียวกันกับข้อ ก) ในการหาค่าตัวชี้กำลังได้ ดังนี้

$$e = \text{floor}(\log_2(0.005628)) = \text{floor}(-7.47316)$$
$$= -8 = 1001_2$$

และจะได้ค่าแมนทิสซา คือ

$$m = 0.005628 / 2^{-8} = 1.440768$$
$$m_{fix} = \text{round}(1.440768 \times 2^{11}) = 2951 = 0101110000111_2$$

จะเห็นได้ว่า บิตที่ 2 (จากซ้าย) เท่ากับ 1 ซึ่งตรงตามรูปแบบของสมการที่ 9.5 ถ้าตัดบิตนี้ทิ้ง เราจะได้ส่วนของแมนทิสซาแทนได้ด้วย 12 บิต เป็น 001110000111_2

<u>ตัวอย่างที่ 9.6</u> จงหาช่วงของค่าที่สามารถแทนได้ เมื่อใช้ระบบเลขอิงดรรชนีที่มีรูปแบบดังสมการที่ 9.4 และ 9.5

ก่อนอื่นเราลองพิจารณาแต่ละส่วนแยกกันก่อน โดยสำหรับตัวชี้กำลัง ซึ่งใช้ 4 บิต จะสามารถแทนค่าได้ตั้งแต่ -8 ถึง 7 สำหรับตัวแมนทิสซา ซึ่งใช้ 12 บิต แต่เสมือนเป็น 13 บิตที่มีค่า M=11 และ N=2 การพิจารณาขอบเขตที่มันแทนได้จะไม่เหมือนเลข 2's complement ปกติ เนื่องจาก แมนทิสซาจะถูกนอร์แมลไลซ์ ดังนั้น จะแทนค่าได้ตั้งแต่ -2 ถึง -1, 0, และ 1 ถึง 2 อาจคิดโดยละเอียดได้ว่า

ก) ค่าบวกมากสุดของแมนทิสซา คือ 01.11111111111_2 ซึ่งมีค่าเท่ากับ $2 - 2^{-11}$

ข) ค่าบวกน้อยสุดของแมนทิสซา คือ 01.00000000000_2 ซึ่งมีค่าเท่ากับ 1

ค) ค่าลบน้อยสุดของแมนทิสซา คือ 10.11111111111_2 ซึ่งมีค่าเท่ากับ $-1 + 2^{-11}$

ง) ค่าลบมากสุดของแมนทิสซา คือ 10.00000000000_2 ซึ่งมีค่าเท่ากับ -2

จากนั้น เราจะพิจารณาขอบเขตของระบบเลขอิงดรรชนีนี้ซึ่งมีอยู่ 4 ขอบเขต ดังนี้

ก) <u>ค่าบวกที่มากที่สุด</u> เกิดเมื่อแมนทิสซามีค่าบวกมากที่สุด และตัวชี้กำลังมีค่าบวกมากที่สุด ซึ่งจะได้ค่าเท่ากับ $(2 - 2^{-11}) \times 2^7 = 255.9375$

ข) <u>ค่าบวกที่น้อยที่สุด</u> เกิดเมื่อแมนทิสซามีค่าบวกน้อยที่สุด และตัวชี้กำลังมีค่าลบมากที่สุด ซึ่งจะได้ค่าเท่ากับ $1 \times 2^{-7} = 0.0078125$

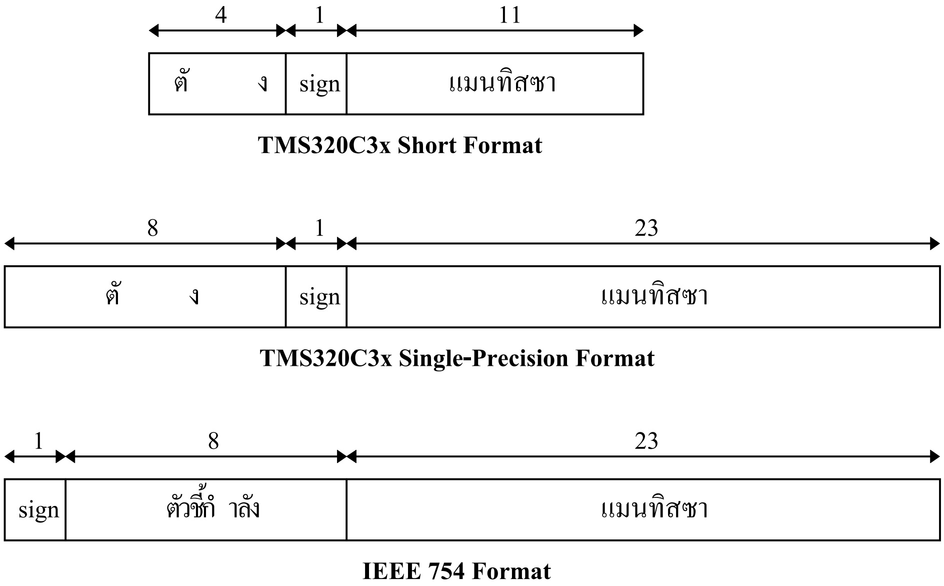
ค) <u>ค่าลบที่น้อยที่สุด</u> เกิดเมื่อแมนทิสซามีค่าลบน้อยที่สุด และตัวชี้กำลังมีค่าลบมากที่สุด ซึ่งจะได้ค่าเท่ากับ $(-1-2^{-11}) \times 2^{-7} = -0.0078163\ldots$

ง) <u>ค่าลบที่มากที่สุด</u> เกิดเมื่อแมนทิสซามีค่าลบมากที่สุด และตัวชี้กำลังมีค่าบวกมากที่สุด ซึ่งจะได้ค่าเท่ากับ $-2 \times 2^7 = -256$

ตัวอย่างของเลขอิงดรรชนีขนาด 16 บิตที่ได้ยกตัวอย่างไปนี้ มีการใช้งานในชิพ TMS320C3x ซึ่งเป็นตัวประมวลผลในระบบเลขอิงดรรชนีของบริษัท เท็กซัสอินสตรูเมนส์ โดยมีโครงสร้างของบิตที่เก็บในเฟรม 16 บิตดังแสดงในรูปที่ 9.1 ระบบเลขอิงดรรชนีที่นิยมใช้กันในการประมวลผล ใช้จำนวนบิตทั้งสิ้น 32 บิต โดยแบ่งเป็นแมนทิสซา 24 บิต และตัวชี้กำลัง 8 บิต ซึ่งจะสามารถแทนค่าได้ประมาณตั้งแต่ $\pm 5.8\times10^{-39}$ จนถึง $\pm 3.4\times10^{-3.8}$ โครงสร้างของเฟรมที่แสดงในรูปที่ 9.1 เป็นรูปแบบเลขอิงดรรชนีขนาด 32 บิตที่ใช้ในชิพ TMS320C3x และที่ถูกกำหนดเป็นมาตรฐาน IEEE 754

เลขอิงดรรชนีที่ใช้ในชิพ TMS320C3x จะใช้รูปแบบ 2's complement แทนทั้งแมนทิสซาและตัวชี้กำลัง แต่สำหรับมาตรฐาน IEEE 745 จะใช้รูปแบบ sign-magnitude แทนแมนทิสซา และรูป

แบบเลขจำนวนเต็มบวกอย่างเดียวแทนตัวชี้กำลัง โดยการคิดค่าชี้กำลังจริง ๆ จะนำค่าเลขจำนวนเต็มบวกนี้มาลบด้วย 127



***รูปที่ 9.1** ตัวอย่างของโครงสร้างเฟรมของระบบเลขอิงดรรชนี*[14]

เปรียบเทียบเลขจำนวนเต็ม กับเลขอิงดรรชนี

มีอยู่หลายจุดที่เราสามารถเปรียบเทียบระบบเลขจำนวนเต็ม กับระบบเลขอิงดรรชนีได้ จุดแรกก็คือ ลักษณะการแบ่งขั้นของมัน ระบบเลขจำนวนเต็มจะแบ่งขั้นที่มีขนาดคงที่ตลอดช่วงที่มันแทนค่าได้ ซึ่งก็เนื่องมาจากการที่มันมีตำแหน่งจุดทศนิยมที่คงที่ ส่วนระบบเลขอิงดรรชนีจะแบ่งขั้นที่มีขนาดไม่เท่ากัน โดยเมื่อค่ามีขนาดน้อย ๆ (ใกล้ศูนย์) การแบ่งขั้นก็จะละเอียด หรือขนาดของขั้นจะเล็ก แต่เมื่อขั้นมีขนาดใหญ่ขึ้น ขนาดของขั้นก็จะใหญ่ขึ้น ทั้งนี้ การแบ่งช่วงว่าจะมีขนาดของขั้นทั้งหมดกี่ขนาด และแต่ละขนาดมีการใช้งานในช่วงกว้างแค่ไหนก็ขึ้นอยู่กับการเลือกจำนวนบิตให้กับแมนทิสซา และตัวชี้กำลัง

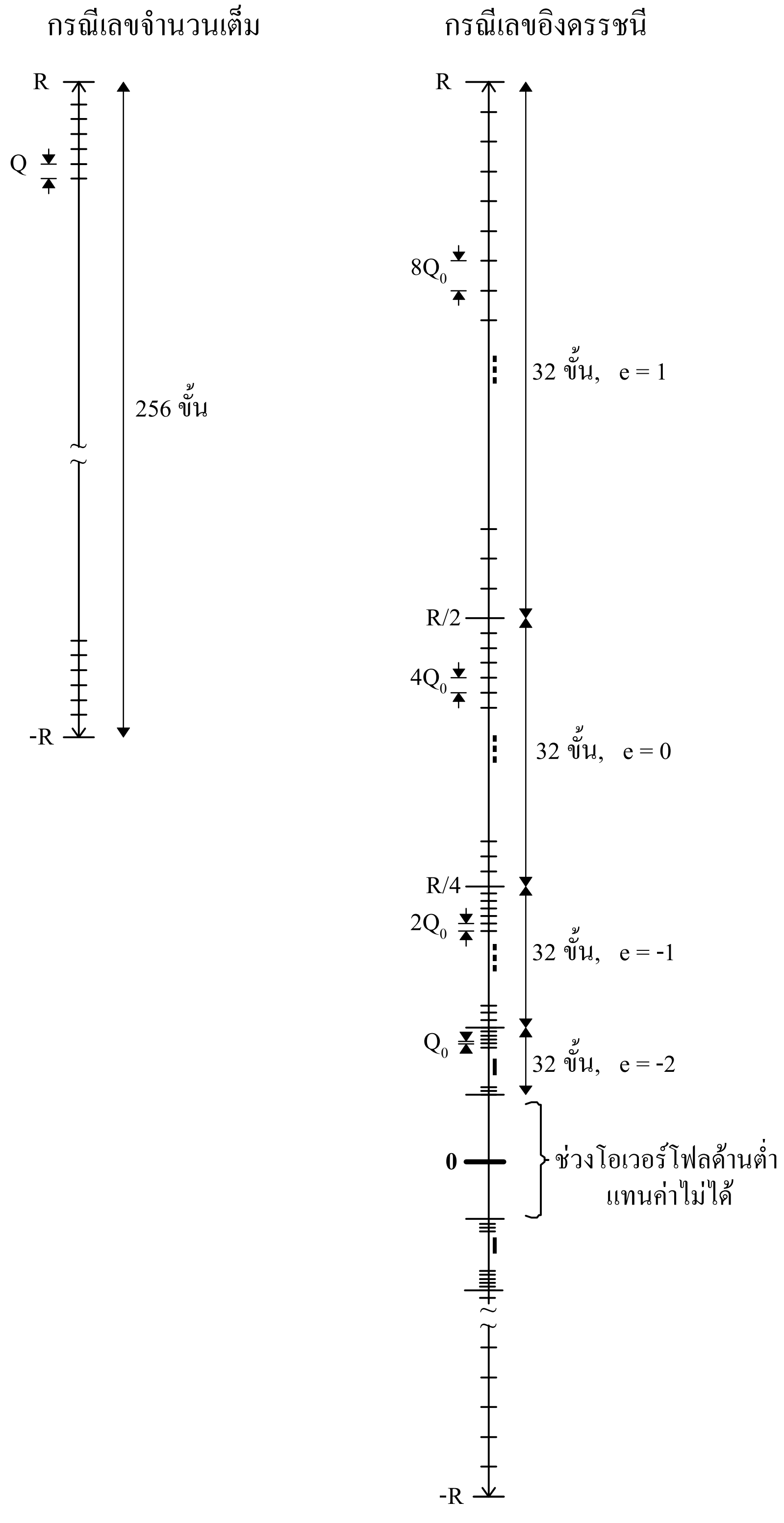
ขอยกตัวอย่างง่าย ๆ เพื่อเปรียบเทียบกัน ดังนี้ ถ้ามีระบบเลขจำนวนเต็มที่มีขนาด 8 บิต และช่วงของค่าที่แทนได้คือ ระหว่าง -R ถึง R จะได้ว่าตลอดช่วงนี้มีการแบ่งขั้นทั้งสิ้นเท่ากับ 256 ขั้น โดยแต่ละขั้นมีระยะห่างเท่ากัน ดังแสดงในรูปที่ 9.2

ในขณะเดียวกัน ถ้าเราใช้เลข 8 บิตนี้เป็นเลขอิงดรรชนี โดยแบ่ง 2 บิตใช้เป็นตัวชี้กำลัง (e) และเหลือ 6 บิตเป็นแมนทิสซา จะได้ว่าเราแบ่งการแทนค่าออกเป็นสี่ช่วง และในแต่ละช่วงมีจำนวนขั้นเท่ากันเท่ากับ 64 ขั้น ช่วงแรก คือ ช่วงที่ e = -2 สมมติว่าได้ขนาดขั้น คือ Q_0 จะมี 32 ขั้นในช่วงบวก และ32 ขั้นในช่วงลบ ช่วงที่สอง คือ ช่วงที่ e = -1 จะได้ขนาดของขั้นเป็นสองเท่าของ Q_0 ดัง

แสดงในรูปที่ 9.2 มี 32 ขั้นในช่วงบวก และ32 ขั้นในช่วงลบเช่นกัน เมื่อ e มากขึ้น ๆ ขนาดของขั้นก็จะใหญ่ขึ้นทีละสองเท่าตัวเรื่อยไปจนถึงค่า e ตัวสุดท้าย

จะเห็นได้ว่า การแบ่งขั้นของระบบเลขอิงดรรชนีมีลักษณะที่ฉลาดกว่า เนื่องจาก มีการปรับนัยสำคัญของเลขที่ใช้แทนให้เหมาะสมตามค่าขนาดของตัวเลข กล่าวคือ ที่ค่าของตัวเลขมาก ๆ ขนาดของขั้นก็จะกว้าง ส่วนที่ค่าตัวเลขน้อย ๆ ขนาดของขั้นก็มีค่าเล็กลง การทำเช่นนี้ ทำให้ระบบเลขอิงดรรชนีที่จำนวนบิตมากพอสมควร จะมีความสามารถในการแทนตัวเลขได้ในช่วงที่กว้างกว่าระบบแบบจำนวนเต็มมาก และแทนได้ดีแม่นยำกว่า เพราะมีนัยสำคัญที่เหมาะสมกับขนาดของตัวเลข โดยทั่วไป ระบบเลขอิงดรรชนีก็จะให้ความคลาดเคลื่อนในการประมวลผลที่ต่ำกว่าด้วย

การใช้ระบบเลขอิงดรรชนี มีข้อเสียในด้านต้นทุนที่แพงกว่า เพราะ วงจรที่ใช้ในการประมวลผลของเลขอิงดรรชนีทำได้ยากกว่าเลขจำนวนเต็ม นอกจากนี้ วงจรของระบบเลขอิงดรรชนียังใหญ่กว่า, ทำงานได้ช้า, และกินไฟมากกว่าด้วย โดยทั่วไป เราจะทดสอบการทำงานของระบบกับระบบเลขจำนวนเต็มก่อนว่าสามารถใช้ได้หรือไม่ ถ้าไม่ได้จึงค่อยลองระบบเลขอิงดรรชนี ซึ่งระบบเลขอิงดรรชนีจะทำงานได้ดีกว่ามากในระบบที่มีสัญญาณทั้งค่าสูง ๆ และต่ำ ๆ และเราต้องการรักษานัยสำคัญของสัญญาณที่มีค่าต่ำ ๆ ไว้



รูปที่ 9.2 *เปรียบเทียบการแบ่งขั้นของระบบเลขจำนวนเต็ม และเลขอิงดรรชนี*

บทที่ 10

ความคลาดเคลื่อนจากการใช้ระบบเลขจำนวนเต็ม

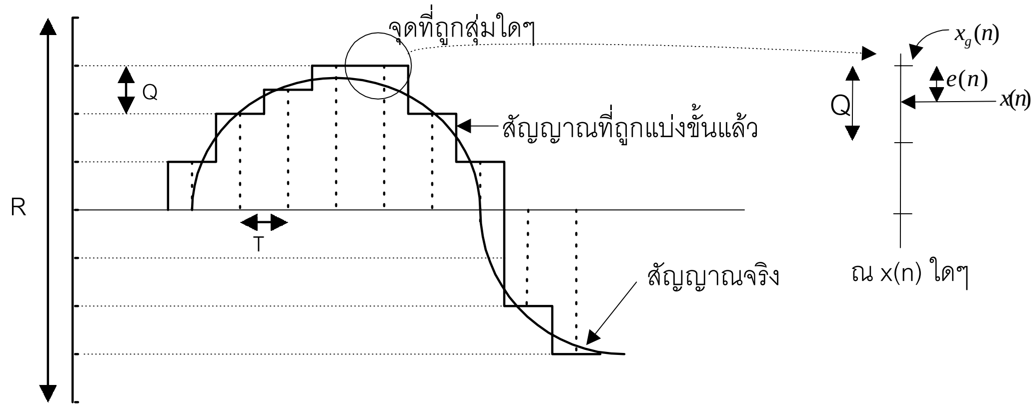
ในบทที่ 9 เราได้รู้จักลักษณะของระบบเลขจำนวนเต็มแบบ 2's complement มาพอสมควรแล้ว ในบทนี้จะได้กล่าวถึงความคลาดเคลื่อนที่เกิดขึ้นในระบบ เมื่อนำเอาระบบเลขนี้มาใช้แทนค่าสัญญาณ และมาใช้ในการประมวลผล ความคลาดเคลื่อนเหล่านี้ เกิดจากการแบ่งขั้นสัญญาณ, การปัดเศษสัมประสิทธิ์, โอเวอร์โฟล, และการปัดเศษหลังการคูณ การวิเคราะห์ในบางส่วนจะมีการใช้ทฤษฎีของตัวแปรแรนดอม (random variable) บ้าง ซึ่งผู้ที่ไม่เข้าใจก็อาจข้ามไปในส่วนของที่มาของการวิเคราะห์นั้น ๆ และไปสนใจในส่วนของผลการวิเคราะห์ได้เลย ในส่วนท้ายของบทจะได้แนะนำการจำลองการประมวลผลแบบระบบเลขจำนวนเต็มโดยใช้ Matlab และตัวอย่างของโปรแกรมภาษาซีที่ใช้งานระบบเลขจำนวนเต็ม

การแบ่งขั้นสัญญาณ (Signal Quantization)

กระบวนการแรกที่เกิดความคลาดเคลื่อนขึ้นในระบบก็คือ การแบ่งขั้นสัญญาณ ซึ่งกระทำหลังจากการสุ่มสัญญาณ การแบ่งขั้นสัญญาณ หมายถึง การแทนค่าสัญญาณที่ถูกสุ่มมาซึ่งมีความละเอียดไม่จำกัด (เพราะมาจากระดับสัญญาณแอนะลอก) ด้วยระบบเลขฐานสองที่มีจำนวนบิตจำกัด ซึ่งก็จะเกิดความคลาดเคลื่อนจากการแทนค่าขึ้น การแบ่งขั้นสัญญาณนี้ในทางปฏิบัติรวมอยู่ในตัวแปลง แอนะลอกเป็นดิจิตอล (A/D) แต่ในทางทฤษฎีแยกการแปลงสัญญาณแอนะลอกเป็นดิจิตอลเป็นสองกระบวนการ คือ การสุ่มซึ่งทำให้ได้สัญญาณไม่ต่อเนื่อง และการแบ่งขั้นสัญญาณซึ่งทำให้ได้สัญญาณที่มีความละเอียดทางขนาดจำกัดลง

ในที่นี้จะอธิบายเฉพาะกรณีที่ใช้ระบบเลขจำนวนเต็ม (fixed-point) ซึ่งจะให้ขนาดของขั้นคงที่ตลอดช่วงที่แทนค่าสัญญาณ การใช้จำนวนบิตยิ่งมากเท่าไรก็จะได้การแบ่งขั้นที่ละเอียด และแทนสัญญาณจริงได้ถูกต้องมากขึ้นเท่านั้น ถ้าเราให้ B เป็นจำนวนบิตที่ใช้แทน 1 ค่าของสัญญาณ, R เป็นช่วงของค่าสัญญาณที่สามารถแทนได้, และ Q แทนเป็นความกว้างของขั้น จะได้ว่า

$$\text{จำนวนขั้นทั้งหมด} = \frac{R}{Q} = 2^{B} \qquad (10.1)$$



***รูปที่ 10.1** แสดงการแบ่งขั้นสัญญาณ (เส้นประในแกนตั้งเป็นตำแหน่งที่มีการสุ่ม)*

สัญญาณก่อนที่จะแบ่งขั้น จะต้องมีค่าอยู่ในช่วง -R/2 ถึง R/2 ถ้าเราพิจารณาตัวอย่างสัญญาณที่จุดหนึ่ง ๆ คือ x(n) ซึ่งหลังจากถูกแบ่งขั้นแล้วจะมีค่าเปลี่ยนไปเพื่อให้ตรงกับขั้นที่แทนค่าได้ สมมติว่าได้ค่าใหม่เป็น $x_q(n)$ ถ้าเราใช้กฎเกณฑ์การปัดเศษหาขั้นที่ใกล้ที่สุด (rounding) จะได้ว่า x(n) ที่มีค่าอยู่ระหว่าง $x_q(n)$ - Q/2 ถึง $x_q(n)$ + Q/2 จะถูกปัดเป็นค่า $x_q(n)$ ซึ่งก็จะทำให้ผลต่างระหว่าง $x_q(n)$ และ x(n) มีค่าไม่เกิน Q/2 หรือครึ่งหนึ่งของขั้น

เรานิยามผลต่างนี้เป็น e(n) ซึ่งเป็นฟังก์ชั่นของความคลาดเคลื่อนที่เกิดขึ้นจากการแบ่งขั้นสัญญาณ จะได้

$$e(n) = x_q(n) - x(n) \tag{10.2}$$

และ
$$-Q/2 < e(n) < Q/2 \tag{10.3}$$

สมการนี้ คล้ายกับสมการที่ 9.3 ซึ่งเป็นค่าคลาดเคลื่อนของการแทนค่าเป็นเลขจำนวนเต็ม เพียงแต่คราวนี้ค่าที่จะแทนเป็นจุดในสัญญาณ เพราะฉะนั้นค่าคลาดเคลื่อนก็จะเกิดขึ้นที่ทุก ๆ จุดของสัญญาณ และค่า Q หาได้จากสมการที่ 10.1 โดยจะมีหน่วยเป็นแรงดันของสัญญาณแอนะลอก

ถ้าสมมติให้ $x_q(n)$ ซึ่งเป็นสัญญาณหลังจากที่ถูกแทนค่าด้วยจำนวนเต็ม เราอาจมองว่าความคลาดเคลื่อนที่เกิดขึ้นกับ $x_q(n)$ เป็นเหมือนสัญญาณรบกวนที่ถูกเติมเข้าไปในสัญญาณเริ่มแรก นั่นคือ $x_q(n) = x(n) + e(n)$ ดังนั้น e(n) จึงมีอีกชื่อหนึ่งว่า เป็นสัญญาณรบกวนที่เกิดจากแบ่งขั้นสัญญาณ (quantization noise)

ฟังก์ชั่น e(n) นี้เป็นฟังก์ชั่นที่มีค่าไม่แน่นอน (random) ชนิดหนึ่ง เพราะเราไม่สามารถรู้ล่วงหน้าได้ว่าแต่ละค่าของ e(n) จะเป็นเท่าไรแน่ (เพราะไม่รู้ว่าสัญญาณที่จะแทนจะมีรูปร่างอย่างไร) แต่รู้แน่ว่า สัญญาณความคลาดเคลื่อนที่จะเกิดจะมีค่าอยู่ระหว่าง -Q/2 ถึง Q/2 เท่านั้น เพื่อให้ง่ายต่อการวิเคราะห์ เราจะสมมติให้ค่าของ e(n) ที่มีการกระจายตัวสม่ำเสมอ (uniform distribution) ในช่วงของค่าระหว่าง -Q/2 ถึง Q/2 หรือมีฟังก์ชั่นการกระจายตัวของโอกาส (probability density function) เป็น

$$p(e) = \begin{cases} 1/Q, & -Q/2 < e < Q/2 \\ 0, & e = \text{ค่าอื่น ๆ} \end{cases} \tag{10.4}$$

จากทฤษฎีของความน่าจะเป็น เราจะได้ว่าค่าเฉลี่ยของ e มีค่าเท่ากับ 0 และค่าเฉลี่ยของกำลังสองของ e มีค่าเท่ากับ $Q^2/12$ ซึ่งหาได้จาก

$$P_q = \int_{-Q/2}^{Q/2} e^2 p(e) \cdot de = \frac{Q^2}{12} \tag{10.5}$$

ค่าเฉลี่ยของกำลังสองของสัญญาณนี้ ก็คือ ค่ากำลังเฉลี่ยนั่นเอง ดังนั้น ขอใช้สัญลักษณ์แทนค่านี้ว่า P_q ซึ่งในที่นี้เราหาค่ากำลังเฉลี่ยของสัญญาณรบกวนจากการแบ่งขั้นสัญญาณได้เท่ากับ $Q^2/12$ สังเกตว่ากำลังของสัญญาณรบกวนจะแปรตาม Q^2 ซึ่งแปรผกผันกับจำนวนบิต

ถ้าเราทราบว่ากำลังเฉลี่ยของสัญญาณมีค่าเท่ากับ P_s เราสามารถหาค่าอัตราส่วนระหว่างกำลังของสัญญาณต่ำกำลังของสัญญาณรบกวน หรือ SNR ที่เนื่องมาจากสัญญาณรบกวนของการแบ่งขั้นสัญญาณได้ คือ

$$SNR_{A/D} = \frac{P_s}{P_q}$$

$$\text{หรือ } SNR_{A/D}(dB) = P_s(dB) - P_e(dB) \tag{10.6}$$

P_s อาจหาได้จาก v_{rms}^2 ซึ่งค่า v_{rms} ของสัญญาณหนึ่ง ๆ นั้นขึ้นกับลักษณะรูปร่างของสัญญาณ ในการวิเคราะห์ต่อไป เราอาจนิยามค่าคุณลักษณะของสัญญาณค่าหนึ่ง เพื่อบ่งบอกถึงค่า v_{rms} ของสัญญาณ ซึ่งค่านั้นก็คือ loading factor หรือ LF ซึ่งเป็นค่าอัตราส่วนระหว่าง v_{rms} ต่อ v_p (Peak Voltage, ค่าสูงสุดของสัญญาณ) ของสัญญาณหนึ่ง ๆ นั่นคือ

$$LF = \frac{v_{rms}}{v_p} \tag{10.7}$$

สัญญาณที่มีระดับสัญญาณสูงสุดอยู่ห่างจากระดับสัญญาณ ณ เวลาอื่น ๆ มาก จะมีค่า LF ที่ต่ำกว่าสัญญาณที่มีระดับสัญญาณสูงสุดอยู่พอ ๆ กับระดับสัญญาณ ณ เวลาอื่น ๆ เช่น สัญญาณซายน์มี $v_{rms} = \frac{v_p}{\sqrt{2}}$ ดังนั้น จะมี $LF = \frac{1}{\sqrt{2}} = 0.707$ ส่วนสัญญาณสี่เหลี่ยมมี $v_{rms} = v_p$ ดังนั้น จะมี LF = 1 ซึ่ง

เป็นค่าที่มากที่สุดเท่าที่จะเป็นไปได้สำหรับสัญญาณใด ๆ สำหรับสัญญาณเสียงคนพูด มีค่า LF ประมาณ 0.15 - 0.2 (จากการทดลอง) เป็นต้น

จากนิยามของ LF เราสามารถหาค่ากำลังของสัญญาณได้เป็น

$$P_s = v_{rms}^2 = (LF)^2 v_p^2 \tag{10.8}$$

สมมติว่าเราสามารถรู้ค่าสูงสุดของสัญญาณ คือ รู้ว่าสัญญาณมีค่าอยู่ระหว่าง $-v_p$ และ $+v_p$ และสามารถเลือกค่าในช่วงนี้เป็นช่วงที่ตัวแปลงแอนะลอกเป็นดิจิตอลสามารถแทนค่าได้พอดี นั่นคือ ค่า R ในรูปที่ 10.1 มีค่าเท่ากับ $2v_p$ เมื่อแทนค่า R ลงในสมการที่ 10.1 จะได้ว่า ขนาดของ 1 ขั้นของสัญญาณมีค่าเป็น

$$Q = \frac{2v_p}{2^B} \tag{10.9}$$

ดังนั้น ได้ SNR อันเนื่องมาจากการแบ่งขั้นสัญญาณ มีค่าเป็น

$$SNR_{A/D} = \frac{P_s}{P_q} = \frac{(LF)^2 v_p^2}{\left(\frac{2v_p}{2^B}\right)^2 \Big/ 12}$$

$$= 3\,(LF)^2\,(2^{2B}) \quad \text{แปลงเป็น dB โดยใส่ 10log( ) จะได้}$$

$$SNR_{A/D} = 10\log\left\{ 3\,(LF)^2\,(2^{2B}) \right\} \ \text{dB}$$

$$= (2B)10\log(2) + 10\log(3) + 20\log(LF) \ \text{dB}$$

$$= 6.02B + 4.77 + 20\log(LF) \ \text{dB} \tag{10.10}$$

ข้อสังเกต และพิจารณาสำหรับ ค่า $SNR_{A/D}$ ที่ได้จากสมการที่ 10.10 นี้ คือ

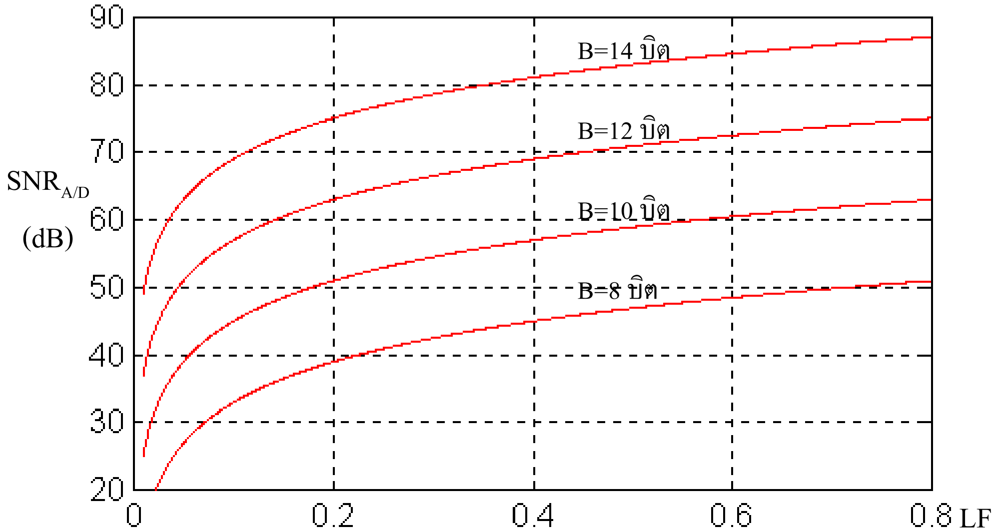
1) $SNR_{A/D}$ มีค่าแปรตามจำนวนบิต (B) โดย SNR จะมีค่ามากขึ้น <u>6 dB ต่อ 1 บิต</u> ที่เพิ่มขึ้น

2) $SNR_{A/D}$ ยังแปรตามค่า LF ดังแสดงในรูปที่ 10.2 ซึ่งวาดกราฟระหว่าง $SNR_{A/D}$ กับ LF ที่จำนวนบิตต่าง ๆ

3) ค่า $SNR_{A/D}$ ที่ได้นี้เป็นค่าที่มากเกินกว่าที่เราจะได้จริงในทางปฏิบัติ เนื่องจากในการหาค่า $SNR_{A/D}$ นี้ เราได้สมมติให้ค่าสูงสุดของสัญญาณ เป็นค่าสูงสุดที่ A/D สามารถแปลงค่าได้พอดี แต่ในทางปฏิบัติเราอาจไม่รู้ค่าสูงสุดของสัญญาณ จึงมักเผื่อให้ค่าสูงสุดที่ A/D สามารถแปลงได้ มีค่ามากกว่าช่วงที่จะใช้งานสักเล็กน้อย นอกจากนี้ ภายในวงจร A/D เองในทางปฏิบัติก็จะมีความคลาด

เคลื่อนในการแปลงค่าด้วย ซึ่งก็จะส่งผลให้มีสัญญาณรบกวนอันเนื่องมาจากความคลาดเคลื่อนของวงจร A/D เองปนเข้ามาอีก

4) การเลือกว่า จะใช้ข้อมูลดิจิตอลกี่บิตแทน 1 ค่าสัญญาณ นอกจากขึ้นกับลักษณะของงานว่าสามารถยอมรับสัญญาณรบกวนได้ที่เท่าไรแล้ว ก็ยังขึ้นกับ SNR ของสัญญาณแอนะลอกขาเข้าด้วย สัญญาณขาเข้าโดยทั่วไป มักจะมีสัญญาณรบกวนปนมาจากแหล่งอื่นอยู่แล้ว ดังนั้น จะมีค่า SNR อยู่ค่าหนึ่ง ถ้าเราใช้จำนวนบิตที่ให้ $SNR_{A/D}$ สูงกว่า SNR ของสัญญาณขาเข้ามาก ๆ ก็อาจไม่มีประโยชน์มากนักเทียบกับต้นทุนที่เพิ่มขึ้น เพราะไม่ว่าจำนวนบิตจะเป็นเท่าไรก็ตาม SNR ของสัญญาณขาออกก็จะยังถูกจำกัดด้วย SNR เริ่มต้นของสัญญาณขาเข้า (ไม่มีทางได้ SNR ดีกว่า SNR ขาเข้า)



***รูปที่ 10.2** แสดงค่า $SNR_{A/D}$ ในฟังก์ชั่นของจำนวนบิต (B) และ loading factor (LF)*

<u>ตัวอย่างที่ 10.1</u> ตัวแปลงแอนะลอกเป็นดิจิตอลขนาด 13 บิต (รวมบิตเครื่องหมาย) ตัวหนึ่ง ให้สัญญาณดิจิตอลขาออกเป็นแบบ 2's complement มีการปรับให้รับขนาดของสัญญาณขาเข้าได้เต็มที่ในช่วง -5 โวล์ท ถึง 5 โวล์ท จงหาว่า

1) ขนาดของขั้นของการแปลงมีค่าเท่ากับเท่าไร
2) ถ้าสัญญาณขาเข้ามีค่า 2V ควรได้สัญญาณขาออกเป็นเท่าไร และมีค่าคลาดเคลื่อนเท่าไร
3) ถ้าใช้กับสัญญาณขาเข้าที่มีค่า vrms = 1.5 V จะได้ $SNR_{A/D}$ เท่ากับเท่าไร

1) ขนาดของขั้น มีค่าเท่ากับ $$Q = \frac{R}{2^B}$$

โดยที่ R = ช่วงของค่าที่แทนได้ = 5 - (-5) = 10 V

B = จำนวนบิตต่อ 1 ค่า = 13 บิต และ 2^B = จำนวนขั้นทั้งหมด = 2^{13} = 8192

แทนค่า R และ B ลงไปจะได้ Q = 10 / 8192 = 1.2207 mV

สังเกตว่า ในกรณีนี้การแบ่งขั้นไม่ลงตัวเหมือนที่เราพบเห็นในเรื่องระบบเลข 2's complement ในบทที่ 9 (R <u>ไม่</u>เป็นค่าเท่ากับ 2^N โดย N เป็นจำนวนเต็ม) ดังนั้นจะไม่สามารถบอกตำแหน่งจุดทศนิยมได้แน่นอนว่าอยู่หลังบิตใด และค่าของแต่ละบิตก็ไม่ลงตัวเป็นกำลังของสอง อย่างไรก็ตาม การแบ่งขั้นที่ไม่ลงตัวของสัญญาณขาเข้านี้ไม่มีผลสำคัญใด ๆ เลย ตรงกันข้าม หากพยายามทำให้ขั้นลงตัว เช่น ถ้าสัญญาณขาเข้าอยู่ระหว่าง $\pm$ 5V แต่ปรับให้ A/D รับค่าได้ในช่วง $\pm$ 8V ก็จะทำให้สูญเสียช่วงที่แปลงค่าได้ไปโดยเปล่าประโยชน์ และทำให้ $SNR_{A/D}$ แย่ลง

การแบ่งขั้นให้ลงตัว จำเป็นต้องใช้สำหรับการแปลงค่าสัมประสิทธิ์ของระบบให้เป็นระบบเลขจำนวนเต็ม เพราะจะทำให้การปัดเศษหลังการคูณทำได้ง่าย ดังจะได้กล่าวต่อไป

2) ถ้าสัญญาณขาเข้า = 2V

ขั้นที่ใกล้เคียง 2V มากที่สุด หาได้จาก round(2V / Q) = round(1638.4) = 1638

ดังนั้น ควรได้ 1638 เป็นค่าผลลัพธ์จาก A/D และค่านี้จริง ๆ แล้วมีค่าเทียบเท่ากับสัญญาณแอนะลอกเท่ากับ 1638 *Q $\approx$ 1.999512 V หรือมีค่าความคลาดเคลื่อนในการแปลงประมาณ

$$1.999512 \text{ V} - 2 \text{ V} = -0.488 \text{ mV}$$

3) ถ้าสัญญาณขาเข้ามี v_{rms} = 1.5 V

จะได้

$$SNR_{A/D} = \frac{P_s}{P_q} = \frac{v_{rms}^2}{Q^2/12} = \frac{1.5^2}{(1.2207\text{mV})^2/12} = 1.8119 \times 10^7$$

$$SNR_{A/D} = 10\log(1.8119 \times 10^7) = 72.6 \text{ dB}$$

<u>**ตัวอย่างที่ 10.2**</u> สัญญาณเสียงสัญญาณหนึ่งมีค่า LF ประมาณ 0.18 ต้องการ $SNR_{A/D}$ = 80 dB จะต้องใช้จำนวนบิตของ A/D อย่างน้อยเท่าไร

แทนค่า $SNR_{A/D}$ ลงในสมการ 10.10 และหาค่า B ดังนี้

$$\begin{aligned} SNR_{A/D} &= 6.02B + 4.77 + 20\log(LF) \\ 80 &= 6.02B + 4.77 + 20\log(0.18) \\ B &= 14.97 \end{aligned}$$

ดังนั้น ต้องการ A/D ที่มีจำนวนบิตเท่ากับ 15 บิต

<u>**หมายเหตุ**</u> การเก็บเสียงในแผ่นซีดี ใช้จำนวนบิตเท่ากับ 16 บิต ซึ่งถ้านำมาใช้กับสัญญาณเสียงในข้อนี้ก็จะให้ SNR ดีกว่า 80 dB ซึ่งถือว่าอยู่ในระดับที่ดีมาก

ความคลาดเคลื่อนจากการปัดเศษสัมประสิทธิ์ (Coefficient Rounding)

ความคลาดเคลื่อนจากการปัดเศษสัมประสิทธิ์ เกิดจากการที่เรานำค่าสัมประสิทธิ์ของระบบที่ได้ออกแบบไว้ไปใช้กับระบบเลขจำนวนเต็ม ซึ่งต้องแทนค่าสัมประสิทธิ์ทุกตัวด้วยจำนวนเต็มที่มีจำนวนบิตจำกัด (โดยทั่วไปใช้ 2's complement) ซึ่งก็แน่นอนว่าจะต้องเกิดความคลาดเคลื่อนกับค่าสัมประสิทธิ์เหล่านั้น การแทนค่าสัมประสิทธิ์ด้วยจำนวนเต็มนี้ เรียกอีกอย่างหนึ่งว่า การปัดเศษสัมประสิทธิ์ เพราะเป็นการปัดให้ค่าสัมประสิทธิ์ไปลงที่ค่าที่สามารถแทนค่าได้ด้วยจำนวนเต็ม

ความคลาดเคลื่อนที่เกิดขึ้นกับสัมประสิทธิ์จะส่งผลถึงความคลาดเคลื่อนของลักษณะของระบบจากที่ได้ออกแบบไว้ ได้แก่ ผลตอบสนองเชิงความถี่มีรูปร่างเปลี่ยนไป, ความถี่ตัดมีค่าเปลี่ยนไป, ตำแหน่งของโพล และศูนย์เปลี่ยนไป จนถึงกระทั่งระบบเสียความมีเสถียรภาพไป ผลดังกล่าวจะมีมากน้อยขึ้นอยู่กับ

1) จำนวนบิตที่ใช้ จำนวนบิตน้อยลงย่อมส่งผลกระทบมากขึ้น

2) ชนิดของตัวกรอง ผลของความคลาดเคลื่อนต่อตัวกรอง IIR จะมากกว่าตัวกรอง FIR มาก และนอกจากนี้ตัวกรองประเภท BPF และ BSF ที่มีแถบความถี่แคบ ๆ มีแนวโน้มที่จะได้รับผลกระทบมากกว่าตัวกรองแบบอื่น ๆ

3) อันดับของตัวกรอง ผลกระทำต่อตัวกรองที่มีอันดับสูงจะมากกว่าตัวกรองที่มีอันดับต่ำ ข้อนี้สามารถแก้ไขได้โดยการแตกตัวกรองที่มีอันดับสูง ให้อยู่ในโครงสร้างอนุกรม หรือขนานของตัวกรองที่มีอันดับสอง ดังที่ได้แสดงตัวอย่างในบทที่ 8

เราบอกได้เพียงว่าความคลาดเคลื่อนของสัมประสิทธิ์ทำให้คุณลักษณะของระบบเปลี่ยนไป แต่จะเปลี่ยนไปอย่างไร มากน้อยแค่ไหน และอยู่ในภาวะที่ยอมรับได้หรือไม่นั้น ยังไม่สามารถหาสูตรทั่วไปที่จะบอกได้ แนวทางที่ดีที่สุดก็คือ ลองปัดเศษสัมประสิทธิ์ดูเป็นเฉพาะกรณี ๆ ไป แล้วนำสัมประสิทธิ์ชุดใหม่ที่ได้หลังจากการปัดเศษไปตรวจสอบดู เช่น ถ้าเป็นตัวกรองดิจิตอลก็ตรวจดูว่าผลตอบสนองเชิงความถี่เปลี่ยนไปอย่างไร

สมมติ ในระบบมีสัมประสิทธิ์ค่าหนึ่งเท่ากับ a และเราต้องการนำไปใช้กับตัวเลข 16 บิต (2's complement) ก่อนอื่นต้องเลือกตำแหน่งจุดทศนิยมของเลขจำนวนเต็มก่อน ในกรณีที่ค่าสัมประสิทธิ์ทุกค่ามีค่าอยู่ระหว่าง -1 ถึง 1 เราสามารถเลือกให้ N=1 และ M=15 ได้ (ขอทบทวนว่า N คือ จำนวนบิตหน้าจุดทศนิยม และ M คือจำนวนบิตหลังจุดทศนิยม) แต่ถ้ามีสัมประสิทธิ์บางตัวที่มีค่าเกินช่วง -1 ถึง 1 ก็อาจต้องเลือกให้ N มากกว่า 1

หลังจากเลือกจำนวนบิต และตำแหน่งของจุดทศนิยมแล้ว ก็สามารถหาค่าของสัมประสิทธิ์ที่จะมีค่าเปลี่ยนไปได้ สมมติให้ $\hat{a}$ คือค่าของสัมประสิทธิ์หลังการปัดเศษ จะได้ว่า

$$\hat{a} = \mathrm{round}(a \times 2^M) / 2^M \qquad (10.11)$$

เราจะใช้สมการนี้กระทำกับสัมประสิทธิ์ทุกค่า และจะได้สัมประสิทธิ์ชุดใหม่ คือชุดหลังการปัดเศษ ซึ่งจะนำไปวิเคราะห์หาผลตอบสนองเชิงความถี่ หรือลักษณะอื่น ๆ ต่อไป อย่าลืมว่า ค่า $\hat{a}$ ที่ได้นี้เป็นค่าที่คำนวณขึ้นมาเพื่อไปใช้วิเคราะห์ผลของความคลาดเคลื่อนเท่านั้น แต่ค่าที่จะนำไปใช้จริง ๆ คือ $a_{fix} = \text{round}(a \times 2^M)$ ซึ่งมีค่าเป็นจำนวนเต็ม

<u>ตัวอย่างที่ 10.3</u> ตัวกรองบัตเตอร์เวอร์ธผ่านต่ำอันดับสอง มี $\omega'_c = 0.4\pi$ มีฟังก์ชั่นถ่ายโอน ดังนี้

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

ต้องการนำไปใช้กับระบบซึ่งแทนตัวเลขด้วย 2’s complement 6 บิต จงหาค่าสัมประสิทธิ์ที่จะนำไปใช้ (ตอบเป็นจำนวนเต็มฐานสิบก็ได้) และหากราฟของผลตอบสนองเชิงความถี่ที่คลาดเคลื่อนไป

ตารางที่ 10.1 แสดงค่าผลลัพธ์ของค่าสัมประสิทธิ์ที่แปลงเป็นจำนวนเต็ม ซึ่งเป็นค่าที่จะนำไปใช้งาน และค่าสัมประสิทธิ์หลังการปัดเศษ ซึ่งจะเป็นค่าที่จะนำไปวิเคราะห์ความคลาดเคลื่อนในข้อนี้พบว่า สัมประสิทธิ์ทุกตัวมีค่าอยู่ระหว่าง -1 ถึง 1 เพราะฉะนั้นสามารถใช้รูปแบบ N=1 และ M=5 ได้

ในที่นี้จะขอแนะนำการใช้ Matlab ซึ่งแสดงในโปรแกรมที่ 10.1 ในการคำนวณค่าสัมประสิทธิ์ และวาดผลตอบสนองเชิงความถี่ที่คลาดเคลื่อนไปเทียบกับตัวต้นฉบับ โปรแกรมนี้เรียกใช้ฟังก์ชั่น butter ใน DSP Toolbox ใน Matlab ซึ่งจะคำนวณค่าสัมประสิทธิ์เริ่มต้นของตัวกรองบัตเตอร์เวิร์ธที่ต้องการ แล้วเก็บไว้ในเวคเตอร์ a และ b โดยที่ $a = [a_0\ \ a_1\ \ a_2]$ และ $b = [b_0\ \ b_1\ \ b_2]$ ค่าที่ได้แสดงไว้ในคอลัมน์ที่สองของตาราง

จากนั้นโปรแกรมจะคำนวณค่าสัมประสิทธิ์ในรูปจำนวนเต็ม แล้วเก็บผลลัพธ์ไว้ในเวคเตอร์ afix และ bfix ในการคำนวณจะเรียกใช้ฟังก์ชั่น fixpnt.m โดยผ่านค่า M และ N ให้กับฟังก์ชั่น ฟังก์ชั่นนี้จะทำการตรวจสอบว่า ค่าสัมประสิทธิ์ที่รับมาสามารถแปลงเป็นจำนวนเต็มในรูปแบบที่กำหนดได้หรือไม่ ถ้าอยู่ในช่วงที่แปลงได้ (ระหว่าง $2^{-(N-1)}$ ถึง 2^{N-1}) ก็จะแปลงให้ ถ้าอยู่นอกช่วงก็จะแจ้งว่ามีข้อผิดพลาดขึ้น ค่าที่แปลงได้แสดงไว้ในคอลัมน์ที่สามของตาราง

สำหรับค่าสัมประสิทธิ์หลังการปัดเศษ ทำโดยการนำ afix และ bfix มาหารด้วย 2^M ซึ่งได้ผลลัพธ์ดังแสดงในคอลัมน์ที่สี่ และค่าสัมประสิทธิ์หลังการปัดเศษนี้จะถูกนำไปใช้หาผลตอบสนองเชิงความถี่ที่เปลี่ยนไปซึ่งได้แสดงไว้ในรูปที่ 10.3

จุดที่ต้องระวัง คือ ค่า b_0 ซึ่งจะต้องมีค่าเป็น 1 เสมอสำหรับตัวกรอง IIR ทั่ว ๆ ไป ค่า b_0 นี้จะไม่ถูกนำไปใช้ไม่ว่าเราจะใช้โครงสร้าง direct form 1 หรือ 2 ก็ตาม ดังนั้น ค่า b_0 เป็นค่าที่จะไม่เกิดความคลาดเคลื่อนขึ้น ในโปรแกรมที่ 10.1 ได้บังคับให้ bfix(1) = 2^M ซึ่งจะทำให้ได้ b_0 หลังการปัดเศษเป็น 1 เสมอ

สัมประสิทธิ์	ค่าสัมประสิทธิ์จริง = a (คำนวณจาก butter ใน Matlab)	ค่าจำนวนเต็มที่นำไปใช้ a_{fix} = round($a \times 2^M$)	ค่าสัมประสิทธิ์หลังการปัดเศษ $\hat{a} = a_{fix} / 2^M$
a_0	0.20657208382615	7	0.21875
a_1	0.41314416765230	13	0.40625
a_2	0.20657208382615	7	0.21875
b_0	1	ไม่ใช้	1
b_1	-0.36952737735124	-12	-0.375
b_2	0.19581571265583	6	0.1875

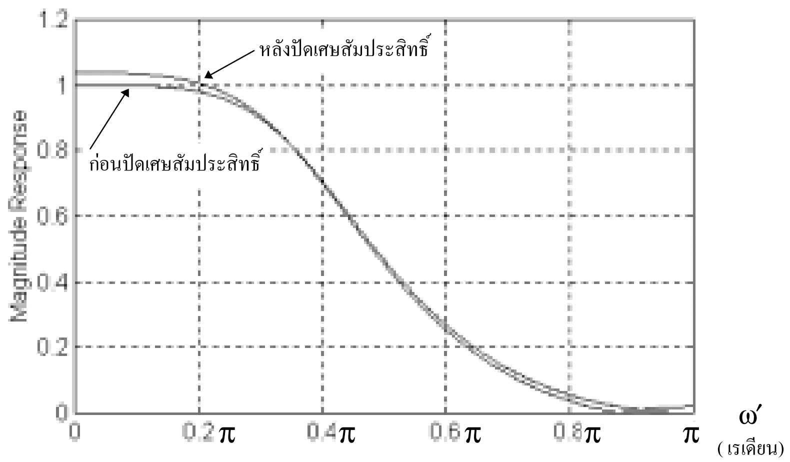
ตารางที่ 10.1 *สัมประสิทธิ์ที่เป็นคำตอบของตัวอย่างที่ 10.3*

```
M=5; N=1;
[a,b]=butter(2,0.4);          %หาสัมประสิทธิ์ของ LPF มีอันดับ=2 และความถี่ตัด=0.4π
afix=fixpnt(a,M,N);
bfix=fixpnt(b,M,N);  bfix(1)=2^M;
freqres(a,b,2)
hold on
freqres(afix/2^M, bfix/2^M, 2)
hold off
```

โปรแกรมที่ 10.1 *ex10_7.m สำหรับหาสัมประสิทธิ์ที่แสดงในตารางที่ 10.1*

```
function y=fixpnt(x,M,N);
max_y = 2^(M+N-1);              %หาค่าสูงสุดที่แทนได้
y = round(x*2^M);               %ทำเป็นจำนวนเต็ม
if max(abs(x)) > max_y          %ตรวจสอบว่าอยู่ในช่วงที่สามารถแทนค่าได้หรือไม่
    error('Overflow Error when converting to fixed-point!');
end
y = y - (y==max_y);             %ในกรณีที่ x=2^(N-1) พอดี จะได้ y = 2^(M+N-1) เป็นจุดที่เกิดโอเวอร์โฟล
                                 ทางบวกพอดี แต่จะอนุโลมให้ใช้ได้โดยใช้ y = 2^(M+N-1) - 1
```

โปรแกรมที่ 10.2 *ฟังก์ชั่น fixpnt.m สำหรับแปลงเป็นค่าเลขจำนวนเต็ม จำนวนบิต = M+N*



รูปที่ 10.3 *ผลตอบสนองเชิงความถี่ก่อน และหลังการปัดเศษสัมประสิทธิ์ของตัวอย่างที่ 10.3*

โปรแกรมที่ 10.1 นี้ สามารถนำไปใช้กับการตรวจสอบความคลาดเคลื่อนของตัวกรองใด ๆ ได้โดยการแก้บรรทัดที่คำนวณสัมประสิทธิ์เริ่มต้น [a, b] ให้หาค่าสัมประสิทธิ์ของตัวกรองที่ต้องการเท่านั้น ในกรณีที่มีค่าสัมประสิทธิ์บางตัวมีค่าเกินช่วง -1 ถึง 1 โปรแกรมก็จะหยุดการทำงาน และแจ้งให้ทราบ ซึ่งผู้ใช้ก็จำเป็นจะต้องเปลี่ยนค่า N ให้มากขึ้นจนกว่าช่วงที่แทนค่าได้จะครอบคลุมค่าของสัมประสิทธิ์ได้ โดยปกติสัมประสิทธิ์ทุกตัวจะปัดเศษด้วยค่า N (และ M) เหมือนกันหมด แต่ในบางโครงสร้าง เช่น โครงสร้าง IIR แบบ direct form 2 อนุญาตให้สัมประสิทธิ์ของโพลิโนเมียลเศษ และโพลิโนเมียลส่วนใช้ค่า N (และ M) ต่างกันได้ ซึ่งรายละเอียด และตัวอย่างจะได้อธิบายในหัวข้อเรื่องการจำลองระบบที่ใช้จำนวนเต็มใน Matlab ตอนท้ายของบทนี้

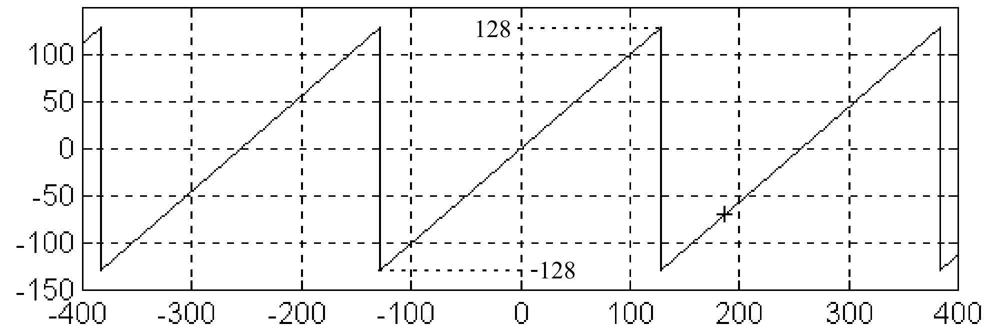
การวิเคราะห์ผลของความคลาดเคลื่อนจากการปัดเศษสัมประสิทธิ์เพียงอย่างเดียว มิได้เป็นหลักประกันร้อยเปอร์เซนต์ว่า เมื่อนำไปใช้จะได้ระบบที่มีผลตอบสนองเชิงความถี่เปลี่ยนไปเหมือนที่ได้วิเคราะห์ไว้ โดยเฉพาะอย่างยิ่งถ้านำไปใช้ที่จำนวนบิตต่ำ ๆ ทั้งนี้เนื่องจาก ยังจะมีผลจากความคลาดเคลื่อนอื่นในการประมวลผลรวมเข้ามาอีกด้วย ซึ่งเราจะได้กล่าวถึงในหัวข้อถัดไป การวิเคราะห์ความคลาดเคลื่อนจากการปัดเศษเป็นการวิเคราะห์เบื้องต้น ซึ่งทำได้ง่าย และรวดเร็ว การวิเคราะห์โดยละเอียดอาจต้องทำโดยการจำลองระบบให้ประมวลผลด้วยระบบเลขจำนวนเต็มตามที่จะใช้จริง ๆ แล้วดูผลลัพธ์ว่าเป็นอย่างไร

ความคลาดเคลื่อนจากโอเวอร์โฟล (Overflow)

โอเวอร์โฟล คือ เหตุการณ์ที่ผลลัพธ์ของการประมวลผลมีค่าเกินช่วงที่จะสามารถแทนค่าได้ โอเวอร์โฟลที่จะกล่าวถึงในหัวข้อนี้ คือ โอเวอร์โฟลที่เกิดจากผลลัพธ์ของการบวก ก่อนอื่นลองพิจารณาดูก่อนว่า เมื่อเกิดโอเวอร์โฟลกับผลลัพธ์ของการบวกเลข 2’s complement จะส่งผลอย่างไรกับตัวเลขนั้น ลองพิจารณาตัวเลข 2’s complement แบบ 8 บิต ที่มี M=0 (แทนค่าตัวเลขได้ในช่วง -128 ถึง 127) สมมติเราต้องการหาผลลัพธ์ของ 100 + 87 ซึ่งผลลัพธ์ที่ได้ คือ 187 ซึ่งเกินช่วงที่จะแทนค่าได้ ลองดูว่าการบวกเลข 2’s complement จะให้ผลลัพธ์อะไรออกมา ดังนี้

0 1 1 0 0 1 0 0	→	100
+ 0 1 0 1 0 1 1 1	→	+ 87
1 0 1 1 1 0 1 1	→	-69

ปรากฏว่าได้ผลลัพธ์ คือ -69 หรือมาจากผลลัพธ์ที่ถูกต้องลบด้วย 2^8 นั่นคือ 187 - 256 = -69 เราสามารถพิสูจน์หาสูตรทั่วไปได้ไม่ยากว่า ผลลัพธ์ที่เกิดโอเวอร์โฟลจะมีค่าเปลี่ยนไปดังแสดงในกราฟในรูปที่ 10.4 โดยแกนนอนแทนค่าของผลลัพธ์ที่ถูกต้อง และแกนตั้งแทนค่าที่จะได้ จุด + ที่แสดงไว้ในกราฟ คือ ผลลัพธ์ของการคำนวณข้างต้น (ค่าแกนนอนเป็น 187 และแกนตั้งเป็น -69)



รูปที่ 10.4 *กราฟระหว่างผลลัพธ์ที่ถูกต้องกับผลลัพธ์ที่เกิดโอเวอร์โฟล (กรณี 8 บิต, M=0)*

จากกราฟจะเห็นได้ว่า โอเวอร์โฟลทำให้ผลลัพธ์ที่ได้เปลี่ยนจากหน้ามือเป็นหลังมือทีเดียว ไม่ได้เป็นเหมือนสัญญาณรบกวน หรือความเพี้ยนที่เกิดขึ้นเหมือนกับความคลาดเคลื่อนที่มาจากสาเหตุอื่น ๆ ดังนั้น โอเวอร์โฟลอาจทำให้สัญญาณขาออกเสียรูปร่างจนดูไม่รู้เรื่องไปเลยก็ได้ ถ้าหากปล่อยให้เกิดขึ้น

อย่างไรก็ตาม โอเวอร์โฟลเป็นความคลาดเคลื่อนที่สามารถป้องกันได้ โดย

1) <u>ลดขนาดของสัญญาณขาเข้า</u> หรือ ใช้ตัวคูณลดทอนที่สัญญาณขาเข้า หรือ ใช้วิธีเลื่อนบิตไปทางซ้าย (left shift) ซึ่งการเลื่อนบิตไปทางซ้าย 1 บิตเท่ากับการหารด้วย 2

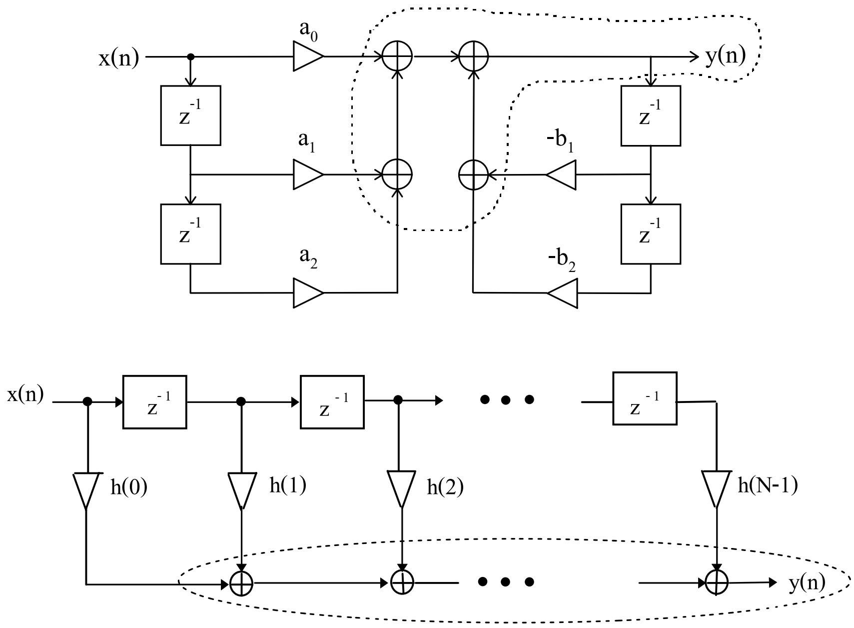
2) <u>สเกลค่าของสัมประสิทธิ์</u>ที่เป็นตัวคูณสัญญาณไปข้างหน้าลง (สัมประสิทธิ์ที่ไม่ใช่ตัวคูณของสัญญาณป้อนกลับ) การดูว่าต้องลดค่าสัมประสิทธิ์ตัวใดบ้างนั้น ขึ้นอยู่กับโครงสร้างของระบบ

การปรับลดขนาดของสัญญาณ หรือสัมประสิทธิ์ข้างต้น ถ้าลดมากเกินไป จะทำให้สัญญาณมีขนาดเล็กเกินความจำเป็น ซึ่งส่งผลให้ SNR ต่ำลง การพิจารณาว่าต้องลดค่าลงเท่าไร อาจทำโดยการทดลอง หรือจำลองใส่สัญญาณขาเข้าที่จะใช้จริงดู หรืออาจประมาณค่าที่ต้องลดล่วงหน้าโดยทำการวิเคราะห์ก่อนก็ได้ ในที่นี้จะแสดงวิธีวิเคราะห์สำหรับตัวกรอง FIR และ IIR เป็นตัวอย่าง

<u>โอเวอร์โฟลของกรณีตัวกรอง FIR และ IIR แบบโครงสร้าง Direct Form 1</u>

สาเหตุที่จัดตัวกรองที่มีโครงสร้างทั้งสองแบบนี้อยู่ในกรณีเดียวกัน เนื่องจาก ทั้งสองมีพฤติกรรมการหาผลตอบเหมือนกัน ในแง่ที่ว่า ในระบบมีการบวกอยู่จุดเดียวซึ่งใช้หาผลตอบสุดท้าย ไม่มีการบวกที่อื่นอีกเลย ซึ่งก็คือจะไม่มีผลลัพธ์กึ่งกลางที่มาจากการบวกเกิดขึ้นในระบบ ดังนั้น ผลลัพธ์ของการบวกที่จะเกิดโอเวอร์โฟลได้ ก็มีอยู่ที่เดียว คือ ที่สัญญาณขาออก ดังแสดงในรูปที่ 10.5

จากที่เราได้ศึกษามาในเรื่องข้อดีของ 2's complement ที่ว่า การบวกตัวเลขหลาย ๆ ตัว ถ้าผลลัพธ์สุดท้ายไม่โอเวอร์โฟล การเกิดโอเวอร์โฟลในระหว่างการบวก (ถ้ามี) จะไม่ส่งผลใด ๆ ดังนั้น ในกรณีนี้ เนื่องจากโอเวอร์โฟลมีโอกาสเกิดที่ค่า y(n) เพียงค่าเดียว ทำให้ความสนใจเราเหลือเพียงว่าจะต้องลดทอนสัญญาณขาเข้าลงด้วยอัตราส่วนเท่าไร จึงจะไม่เกิดโอเวอร์โฟล์ที่สัญญาณ y(n)



***รูปที่ 10.5** ตำแหน่งที่มีโอกาสเกิดโอเวอร์โฟลในโครงสร้าง FIR และ IIR direct form 1*

สมมติว่า เราต้องลดทอนสัญญาณขาเข้าลง s เท่า (s > 1) จะเรียก s ว่าเป็นตัวประกอบการสเกล (scaling factor) นั่นคือ จะมีตัวคูณที่มีสัมประสิทธิ์เท่ากับ 1/s คูณที่สัญญาณขาเข้า

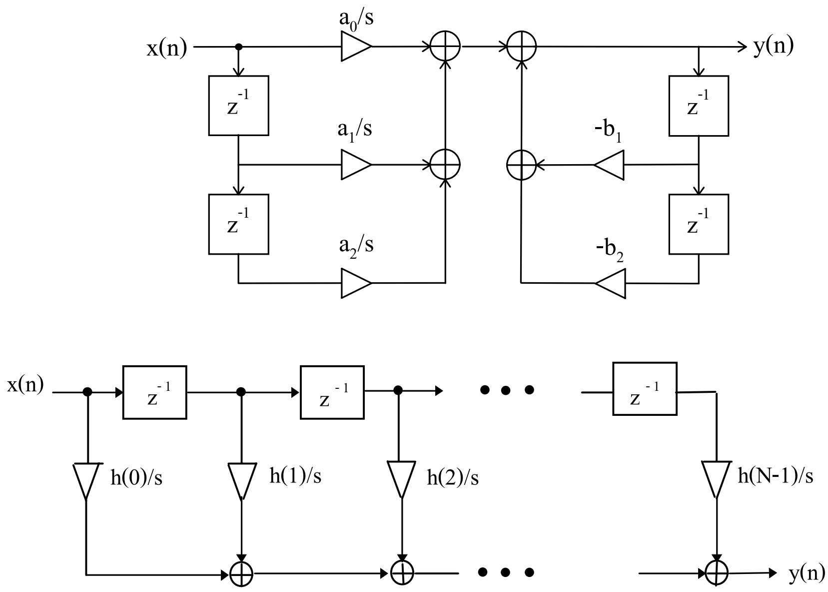
วิธีที่ฉลาดกว่า โดยไม่ต้องใช้ตัวคูณเพิ่มอีก 1 ตัวที่สัญญาณขาเข้า ก็คือ ย้ายอัตราส่วน 1/s ไปปรับลดค่าของสัมประสิทธิ์ของตัวกรองแทน โดยที่สำหรับตัวกรอง FIR เห็นได้ชัดว่า เราสามารถนำ 1/s ไปคูณเข้ากับสัมประสิทธิ์ทุก ๆ ตัวเกิดเป็นสัมประสิทธิ์ชุดใหม่แทนได้ ดังนี้

$$h_{new}(n) = \frac{h(n)}{s} \tag{10.12}$$

ในทำนองเดียวกัน สำหรับตัวกรอง IIR แบบ direct form 1 เราจะย้าย 1/s ไปคูณที่สัมประสิทธิ์ที่เป็นตัวคูณไปข้างหน้าแทนได้ ซึ่งจะได้สัมประสิทธิ์ใหม่ คือ

$$a_{i,new} = \frac{a_i}{s} \tag{10.13}$$

ได้โครงสร้างของทั้งสองแบบสรุปอยู่ในรูปที่ 10.6 และคำถามต่อไปก็ คือ จะใช้ค่า s เท่ากับเท่าไรจึงเหมาะสม ค่า s นี้ขึ้นอยู่กับหลายอย่าง ได้แก่ ฟังก์ชั่นถ่ายโอน, โครงสร้างที่ใช้, และลักษณะของสัญญาณขาเข้า เราสามารถประมาณค่า s ได้ 3 แนวทาง คือ



***รูปที่ 10.6** การลดค่าสัมประสิทธิ์เพื่อป้องกันโอเวอร์โฟลในโครงสร้าง FIR และ IIR direct form 1*

1) ใช้ขอบเขต L_1 แนวทางนี้มีเงื่อนไขว่า ค่าตัวประกอบการสเกล (s) ที่ได้ต้องทำให้สัญญาณขาออกมีขอบเขตของขนาดไม่เกินขอบเขตของขนาดของสัญญาณขาเข้า เราสามารถพิสูจน์โดยสมมติว่า ถ้าสัญญาณขาเข้ามีขนาดอยู่ในช่วง ±1 จะต้องหาค่า s ที่ทำให้สัญญาณขาออกมีขนาดไม่เกิน ±1

สัญญาณขาออก คือ y(n) สามารถพิจารณาได้ว่า เกิดจากการคอนโวลูชั่นระหว่าง x(n) และ y (n) ซึ่งสำหรับกรณีระบบแบบคอซัล จะได้

$$y(n) = x(n) * h(n) = \sum_{m=0}^{\infty} h(m)x(n-m)$$

ถ้า x(n) มีขนาดอยู่ในช่วง ±1 y(n) จะมีโอกาสที่จะมีค่าสูงที่สุด เมื่อ x(n-m) มีขนาดเท่ากับ 1 และมีเครื่องหมายตรงกับ h(m) ทุก ๆ ค่าของ m ซึ่งถ้าเกิดกรณีนี้ขึ้น จะได้ค่าสูงสุด คือ

$$y_{max} = \sum_{n=0}^{\infty} |h(n)|$$

เราจะต้องทำให้ค่าของ y_{max} ไม่เกิน 1 ซึ่งก็สามารถทำได้โดยการลดขนาดของ y(n) ลด y_{max} เท่า เนื่องจากระบบเป็นเชิงเส้น การต้องการให้สัญญาณขาออกลดขนาดลง ก็ทำได้โดยลดขนาดของสัญญาณขาเข้าด้วยอัตราส่วนเดียวกัน ดังนั้น ค่าตัวประกอบการสเกลที่จะต้องนำไปหารสัญญาณขาเข้าในกรณีนี้ คือ

$$s = L_1 = \sum_{n=0}^{\infty} |h(n)| \qquad (10.14)$$

```
function s=sum_h(a,b);
if length(b)==1,                          %ใช้ฟังก์ชั่น filter ซึ่งเป็นฟังก์ชั่นใน DSP Toolbox
   s = sum(abs(a))/abs(b);                %กรณี FIR  h(n) คือ a
else
   [out,zi] = filter(a,b,1);              %กรณี IIR
   outold=100 ;                           %ใส่สัญญาณขาเข้าเป็น [1 0 0 0 ….] แล้วคำนวณผล
   s=0;
   in=0;                                  บวกจากสัญญาณขาออกจนกระทั่งขาออกลู่เข้าสู่ศูนย์
   while (abs(out) > 1e-10) | (abs(outold-out) > 1e-6)
      s = s + abs(out);
      [out,zi] = filter(a,b,in,zi);
      outold = out;
   end
end
```

โปรแกรมที่ 10.3 *ฟังก์ชั่น sum_h.m สำหรับคำนวนหาขอบเขต L_1*

สูตรนี้ใช้ได้กับทั้งตัวกรอง FIR และ IIR ซึ่งสำหรับตัวกรอง IIR ที่เสถียร ค่า h(n) จะต้องลู่เข้าสู่ศูนย์ ดังนั้น ก็สามารถบวกค่าของ h(n) ไปจนกระทั่ง มันมีค่าน้อยมาก ๆ โปรแกรมที่ 10.3 เป็นโปรแกรม Matlab เพื่อใช้คำนวณค่า L_1 โดยให้ผู้ใช้ป้อนค่าสัมประสิทธิ์ของโพลิโนเมียลเศษ และส่วนลงไปในรูปของเวคเตอร์ a และ b (ลักษณะเดียวกับการใช้งานโปรแกรม freqres.m ในบทที่ 5)

เงื่อนไขที่ใช้หาค่า s โดยวิธีนี้ เป็นเงื่อนไขที่ตรงกับเงื่อนไขของการเกิดโอเวอร์โฟลจริง ดังนั้นวิธีนี้รับประกันว่า จะไม่มีการเกิดโอเวอร์โฟลขึ้นอย่างแน่นอน อย่างไรก็ตาม ค่าที่ได้จากกรณีนี้โดยทั่วไปเป็นค่าที่สูงเกินความจำเป็น เนื่องจาก กรณีที่จะทำให้เกิดค่าสูงสุดของสัญญาณขาออก (y_{max}) ที่ได้กล่าวถึงไปนั้นเป็นกรณีสุดโต่งมาก ในการใช้งานจริง อาจไม่มีสัญญาณขาเข้าที่เหมือนหรือใกล้เคียงกับกรณีที่ทำให้เกิดสัญญาณขาออกดังกล่าวได้

2) ใช้ขอบเขต L_2 แนวทางนี้มีเงื่อนไขว่า ค่าตัวประกอบการสเกลที่ได้ต้องทำให้สัญญาณขาออกมีขอบเขตของกำลัง ไม่เกินขอบเขตของกำลังของสัญญาณขาเข้า (กำลังในที่นี้ คือ ค่าเฉลี่ยกำลังสองของสัญญาณ หรือเท่ากับ v_{rms}^2) ซึ่งจะได้ค่าตัวประกอบการสเกล คือ

$$s = L_2 = \sqrt{\sum_{n=0}^{\infty}\left[h(n)\right]^2} \qquad (10.15)$$

ขอละการพิสูจน์ค่า L_2 เนื่องจาก ค่อนข้างเกินขอบเขตของความรู้ในเบื้องต้น สำหรับการคำนวณค่า L_2 โดย Matlab แสดงในโปรแกรมที่ 10.4 ซึ่งก็สามารถกระทำได้คล้ายคลึงกับการคำนวณค่า L_1

การใช้ค่าตัวประกอบการสเกลเท่ากับ L_2 นี้ ไม่รับประกันว่าจะไม่เกิดโอเวอร์โฟลขึ้น ทั้งนี้เนื่องจาก เงื่อนไขของการที่สัญญาณมีกำลังจำกัด ไม่ได้รับประกันว่าสัญญาณจะมีขนาดจำกัดที่ขอบเขตใดแน่นอน

```
function s=sum_h2(a,b);
if length(b)==1,
   s = sum(a.^2)/b^2;
else
   [out,zi] = filter(a,b,1);
   outold=100;
   s=0; in=0;
   while (abs(out) > 1e-10) | (abs(outold-out) > 1e-6)
      s = s + out^2;
      [out,zi] = filter(a,b,in,zi);
      outold = out;
   end
end
s = sqrt(s);
```

โปรแกรมที่ 10.4 *ฟังก์ชั่น sum_h2.m สำหรับคำนวนหาขอบเขต* L_2

3) ใช้ขอบเขต L_∞ แนวทางนี้มีเงื่อนไขว่า เมื่อสัญญาณขาเข้าเป็นสัญญาณความถี่เดี่ยว จะไม่เกิดโอเวอร์โฟลขึ้น ซึ่งทำได้โดยการนอร์แมลไลซ์ให้ผลตอบสนองเชิงความถี่ทางขนาดมีค่าไม่เกินหนึ่งตลอดช่วงของความถี่ที่ใช้งานทั้งหมด นั่นคือ จะทำให้ ที่แต่ละความถี่ (เมื่อมีสัญญาณความถี่เดี่ยวเป็นขาเข้า) สัญญาณจะไม่ถูกขยายใหญ่กว่าสัญญาณขาเข้า ดังนั้น ค่าตัวประกอบการสเกลในกรณีนี้จะเท่ากับค่าสูงสุดของผลตอบสนองเชิงความถี่ทางขนาด คือ

$$s = L_\infty = \max\left|H(e^{j\omega'})\right| \tag{10.16}$$

โดยปกติเราจะได้ $\mathbf{L_2 < L_\infty < L_1}$[2] ซึ่งก็อาจจะเลือกใช้ค่า s เป็นค่าใด ๆ ในช่วงนี้ก็ได้ การเลือกค่า s มากก็จะมีความปลอดภัยจากโอเวอร์โฟลมากขึ้น แต่ขณะเดียวกันค่า SNR ของสัญญาณก็ลดลง ค่า s ที่เลือกใช้อาจเป็นเพียงค่าเริ่มต้นที่จะนำไปใช้ทดลองกับสัญญาณจริง แล้วค่อยปรับแต่งทีหลังอีกครั้งหนึ่ง

<u>ตัวอย่างที่ 10.4</u> ตัวกรอง IIR อันดับสองตัวหนึ่งมีฟังก์ชั่นถ่ายโอนดังนี้

$$H(z) = \frac{0.1299z^2 + 0.2597z + 0.1299}{z^2 - 1.4836z + 0.8309}$$

ถ้านำตัวกรองนี้ไปใช้ด้วยโครงสร้าง direct form 1 จะหาค่าตัวประกอบการสเกลที่ต้องใช้ เปรียบเทียบกันระหว่างวิธี L_1, L_2, และ L_∞

<u>หาขอบเขต L1</u> โดยใช้โปรแกรมที่ 10.3 ใน Matlab ดังนี้

a = [0.1299 0.2597 0.1299];

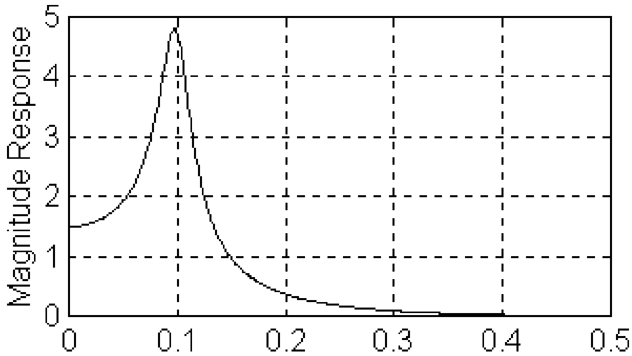
b = [1 -1.4836 0.8309];

sum_h(a,b)

ซึ่งได้ค่าออกมา คือ $L_1 = 6.159$

หาขอบเขต L2 โดยใช้โปรแกรมที่ 10.4 ดังนี้ sum_h2(a,b) ซึ่งได้ค่าเป็น $L_2 = 1.4485$

หาขอบเขต L∞ โดยวาดรูปผลตอบสนองเชิงความถี่ของระบบ และตรวจดูจุดสูงสุด จากรูปที่ 10.7 เห็นได้ว่าจุดสูงสุดอยู่ที่ความถี่ประมาณ 0.1π เรเดียน และได้ค่า $L_\infty = 4.80$



***รูปที่ 10.7** ผลตอบสนองเชิงความถี่ทางขนาดของระบบในตัวอย่างที่ 10.4*

โอเวอร์โฟลของกรณีตัวกรอง IIR แบบโครงสร้าง Direct Form 2

โครงสร้างแบบ direct form 2 มีลักษณะแตกต่างจาก direct form 1 คือ มีผลลัพธ์ที่เกิดจากการบวกสองตัว คือ y(n) และ w(n) ดังแสดงในรูปที่ 10.8 ดังนั้น คราวนี้จะมีโอกาสเกิดโอเวอร์โฟลที่ผลลัพธ์ภายใน คือ w(n) ด้วย ไม่ใช่เพียงแค่ผลลัพธ์สุดท้ายเท่านั้น การเกิดโอเวอร์โฟลที่ผลลัพธ์กึ่งกลางเช่นนี้ เรียกว่า โอเวอร์โฟลภายใน (internal overflow) ซึ่งเป็นจุดที่เสียเปรียบของโครงสร้างนี้เมื่อเทียบกับโครงสร้าง direct form 1

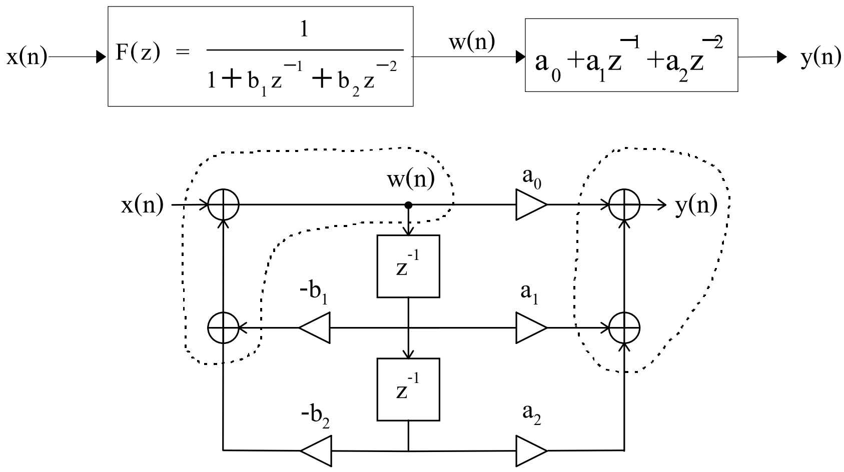
การป้องกันโอเวอร์โฟลภายในก็สามารถทำได้ในทำนองเดียวกับการป้องกันโอเวอร์โฟลที่ผลลัพธ์สุดท้าย โดยถ้าพิจารณาฟังก์ชั่นถ่ายโอนเฉพาะส่วนของโพล คือ F(z) ดังแสดงในรูปที่ 10.8 จะเห็นได้ว่า w(n) เกิดจากการการนำสัญญาณ x(n) ผ่าน F(z) เพียงส่วนเดียว เราจะพิจารณาหาตัวประกอบการสเกลเพื่อลดขนาดของ x(n) <u>เพื่อไม่ให้เกิดโอเวอร์โฟลที่ w(n)</u> ก่อน สมมติให้ตัวประกอบการสเกลนี้ คือ s_1 ซึ่งสามารถหาได้สามแนวทางดังที่ได้กล่าวมาแล้ว แต่คราวนี้การคำนวณหา s_1 จะใช้ฟังก์ชั่น F(z) กับ f(n) ซึ่งคือ ผลตอบสนองต่ออิมพัลส์ของ F(z) ดังนี้

แนวทางที่ 1 $$s_1 = L_1 = \sum_{n=0}^{\infty}\left|f(n)\right| \qquad (10.17)$$

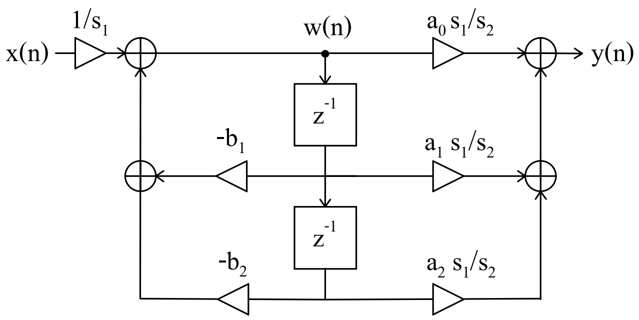
แนวทางที่ 2 $$s_1 = L_2 = \sqrt{\sum_{n=0}^{\infty}\left[f(n)\right]^2} \qquad (10.18)$$

แนวทางที่ 3 $$s_1 = L_\infty = \max\left|F(e^{j\omega'})\right| \qquad (10.19)$$

s_1 นี้ต้องนำไปใช้ลดขนาดที่สัญญาณขาเข้าเท่านั้น ดังแสดงในรูปที่ 10.9 ไม่สามารถนำไปปรับลดค่าสัมประสิทธิ์ได้ เพราะ F(z) ไม่มีสัมประสิทธิ์ที่เป็นตัวคูณไม่ข้างหน้า



รูปที่ 10.8 *ตำแหน่งที่มีโอกาสเกิดโอเวอร์โฟลในโครงสร้าง IIR direct form 2*



รูปที่ 10.9 *การลดค่าสัมประสิทธิ์เพื่อป้องกันโอเวอร์โฟลในโครงสร้าง IIR direct form 2*

หลังจากนั้น เราจะทำการหาตัวประกอบการสเกลเพื่อป้องกันไม่ให้เกิดโอเวอร์โฟลที่สัญญาณ y(n) ซึ่ง y(n) คือ สัญญาณขาออกที่ได้ผ่านฟังก์ชั่นถ่ายโอนมาทั้งหมด เพราะฉะนั้นเราจะใช้ฟังก์ชั่น H (z) เพื่อหาตัวประกอบการสเกลนี้ สมมติว่าค่าที่ได้ คือ s_2 ซึ่งจะหาได้สามแนวทางเช่นเดียวกันโดยสมการที่ 10.17 - 10.19 ด้วยการเปลี่ยน s_1 เป็น s_2, เปลี่ยน f(n) เป็น h(n), และเปลี่ยน F(z) เป็น H(z)

ค่า s_2 ที่ได้นี้ จะนำไปใช้โดยหลักการว่า ต้องทำให้สัญญาณขาเข้ามีการลดทอนด้วยอัตราส่วน $1/s_2$ ก่อนที่จะถึงสัญญาณขาออก แต่ว่าในการป้องกันโอเวอร์โฟลที่สัญญาณ w(n) เราได้ทำการลดทอนสัญญาณขาเข้าไปแล้วด้วยอัตราส่วน $1/s_1$ ดังนั้น เพื่อให้อัตราการลดทอนโดยรวมจากขาเข้าถึงขาออกเท่ากับ $1/s_2$ ก็จะต้องทำการคูณเพิ่มเข้าไปด้วยอัตราส่วนเท่ากับ s_1/s_2

ในกรณีที่ s_2 มากกว่า s_1 อัตราส่วน s_1/s_2 นี้อาจจะคูณเข้าไปที่สัญญาณขาเข้าเลยก็ได้ ซึ่งก็จะทำให้ได้ตัวคูณที่สัญญาณขาเข้าเปลี่ยนเป็น $1/s_2$ ซึ่งเล็กกว่า $1/s_1$ แต่โดยทั่วไปแล้ว เราจะได้ s_1 มากกว่า s_2 เนื่องจาก s_1 คิดมาจากฟังก์ชั่นถ่ายโอนที่เป็นส่วนของโพลซึ่งจะมีการขยายมากกว่าส่วนของศูนย์ ดังนั้น โดยทั่วไปอัตราส่วน s_1/s_2 จะได้ค่ามากกว่า 1 ซึ่งค่านี้จะนำมาใช้คูณที่สัมประสิทธิ์ a_i ดังแสดงในรูปที่ 10.9

ในกรณีที่ได้ s_1/s_2 มากกว่า 1 นี้ เห็นได้ชัดว่าถึงแม้เราไม่คูณค่านี้ที่สัมประสิทธิ์ a_i สัญญาณ y (n) ก็ไม่เกิดโอเวอร์โฟลแน่นอนอยู่แล้ว แต่ในทางตรงกันข้าม y(n) ที่ได้จะมีขนาดเล็กเกินความจำเป็น เราจึงใช้อัตราส่วน s_1/s_2 คูณเข้าไปที่ a_i เพื่อขยายสัญญาณ y(n) ให้ใหญ่ขึ้น (เพราะคิดโดยรวมแล้วจะได้อัตราส่วนจากต้นถึงปลายเท่ากับ $1/s_2$) ซึ่งก็จะส่งผลให้ SNR ของสัญญาณขาออกดีขึ้น

<u>ตัวอย่างที่ 10.5</u> ตัวกรอง IIR อันดับสองตัวหนึ่งมีฟังก์ชั่นถ่ายโอนดังนี้ (เหมือนกับตัวอย่างที่ 10.4)

$$H(z) = \frac{0.1299z^2 + 0.2597z + 0.1299}{z^2 - 1.4836z + 0.8309}$$

ถ้านำตัวกรองนี้ไปใช้ด้วยโครงสร้าง direct form 2 จะวาดโครงสร้างของตัวกรองนี้พร้อมทั้งระบุค่าของสัมประสิทธิ์ที่ต้องใช้ กำหนดให้ใช้ขอบเขต L_1 สำหรับคำนวณตัวประกอบการสเกล

คำนวณหา s_1 โดยคิดเฉพาะส่วนของตัวหารของฟังก์ชั่นถ่ายโอน ใช้โปรแกรมที่ 10.3 ใน Matlab ช่วยคำนวณได้ดังนี้

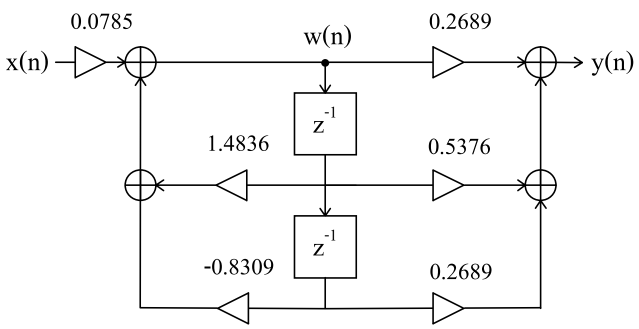
```
a = [ 0.1299   0.2597   0.1299];
b = [1   -1.4836   0.8309];
sum_h(1, b)
```

จะได้ค่า s_1 = 12.741 และ $1/s_1$ = 0.0785

สำหรับค่า s_2 หาโดยใช้ H(z) ด้วยคำสั่ง sum_h(a, b) ซึ่งจะได้ s_2 = 6.159

ดังนั้น จะได้อัตราส่วน s_1/s_2 = 2.07 ซึ่งเป็นอัตราส่วนที่จะนำไปคูณกับ a_i

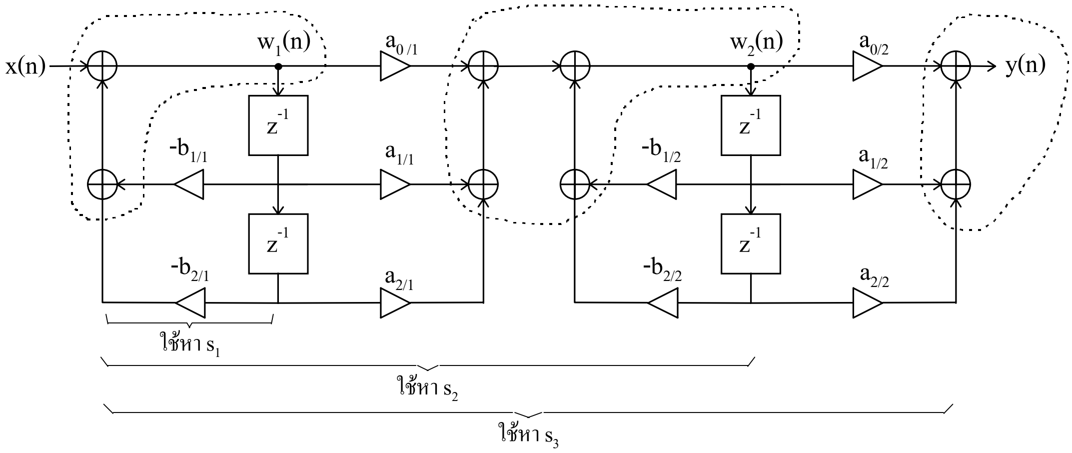
แทนค่าทั้งหมดลงในแผนภาพในรูปที่ 10.9 จะได้คำตอบดังในรูปที่ 10.10



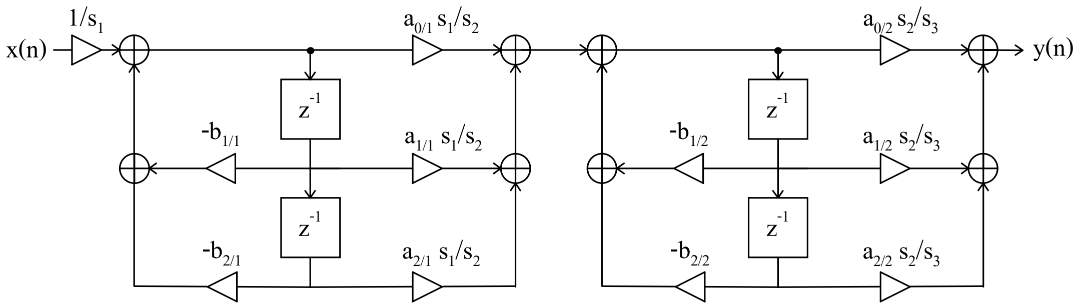
***รูปที่ 10.10** แผนภาพซึ่งเป็นคำตอบของตัวอย่างที่ 10.5*

<u>โอเวอร์โฟลของกรณีตัวกรอง IIR โครงสร้างอนุกรม</u>

ในที่นี้จะแสดงวิธีคิดสำหรับกรณีที่ใช้โครงสร้างย่อยเป็นแบบ direct form 2 ซึ่งจากรูปที่ 10.10 จะเห็นได้ว่า มีจุดที่สามารถเกิดโอเวอร์โฟลได้ 3 ที่ คือ $w_1(n)$, $w_2(n)$, และ y(n) ซึ่งหลักการในการป้องกันโอเวอร์โฟลก็ทำในทำนองเดียวกับในกรณีโครงสร้าง direct form 2 ที่ได้อธิบายไปแล้ว



รูปที่ 10.10 *ตำแหน่งที่มีโอกาสเกิดโอเวอร์โฟลในโครงสร้างอนุกรม*



รูปที่ 10.11 *การลดค่าสัมประสิทธิ์เพื่อป้องกันโอเวอร์โฟลในโครงสร้างอนุกรม*

สมมติให้ $H_1(z)$ คือ ฟังก์ชั่นถ่ายโอนย่อยตัวแรก ซึ่งมี $F_1(z)$ เป็นส่วนของฟังก์ชั่นเฉพาะส่วนของโพล และให้ $H_2(z)$ คือ ฟังก์ชั่นถ่ายโอนย่อยที่สอง ซึ่งมี $F_2(z)$ เป็นส่วนของฟังก์ชั่นเฉพาะส่วนของโพล เราจะหาตัวประกอบการสเกล 3 ค่า คือ s_1, s_2, และ s_3 โดยที่

s_1 หาโดยใช้ฟังก์ชั่น $F_1(z)$

s_2 หาโดยใช้ฟังก์ชั่น $H_1(z)\,F_2(z)$

s_3 หาโดยใช้ฟังก์ชั่น $H_1(z)\,H_2(z)$

สังเกตว่า เราไม่ต้องใช้ตัวประกอบการสเกลที่คิดจากฟังก์ชั่น $H_1(z)$

จากนั้น หาอัตราส่วน $1/s_1$, s_1/s_2, และ s_2/s_3 ซึ่งแต่ละค่าจะใช้เป็นตัวคูณเพื่อป้องกันโอเวอร์โฟลที่ $w_1(n)$, $w_2(n)$, และ $y(n)$ ตามลำดับ ตำแหน่งที่ต้องทำการคูณค่าอัตราส่วนนี้เข้าไปแสดงอยู่ในรูปที่ 10.11

<u>ตัวอย่างที่ 10.6</u> ตัวกรองตัดความถี่ตัวหนึ่งออกแบบโดยใช้โปรแกรม singfreq.m ในบทที่ 8 ให้ตัดความถี่ 0.2π และ 0.4π โดยสั่ง Matlab ดังนี้

```
[a1,b1] = singfreq(0.2,1,0.95,0.95);
[a2,b2] = singfreq(0.4,1,0.95,0.95);
```

a = conv(a1,a2); (สามารถใช้ conv ในการคูณโพลิโนเมียลได้!)

b = conv(b1,b2);

จะได้เป็นตัวกรองอันดับ 4 โดยมี a และ b เป็นเวคเตอร์ที่เก็บสัมประสิทธิ์ของเศษ และส่วน

a = [0.9025 -2.0181 2.7075 -2.0181 0.9025]

b = [1.0000 -2.1243 2.7075 -1.9171 0.8145]

a1 และ b1 เป็นเวคเตอร์ที่เก็บสัมประสิทธิ์ของเศษ และส่วนของฟังก์ชั่นถ่ายโอนย่อย $H_1(z)$

a1 = [0.9500 -1.5371 0.9500] b1 = [1.0000 -1.5371 0.9025]

a2 และ b2 เป็นเวคเตอร์ที่เก็บสัมประสิทธิ์ของเศษ และส่วนของฟังก์ชั่นถ่ายโอนย่อย $H_2(z)$

a2 = [0.9500 -0.5871 0.9500] b2 = [1.0000 -0.5871 0.9025]

ถ้านำไปใช้ด้วยโครงสร้างอนุกรม แบบ direct form 2 จะหาค่าสัมประสิทธิ์ที่ต้องใช้ (กำหนดให้ใช้ขอบเขต L_1 ในการคิดตัวประกอบการสเกล)

ตัวกรองนี้จะมีโครงสร้างตามรูปที่ 10.11 ก่อนอื่นหาค่าตัวประกอบการสเกลก่อน

หา s_1 โดยใช้ s1 = sum_h(1,b1) ได้ $s_1 = 4.855$

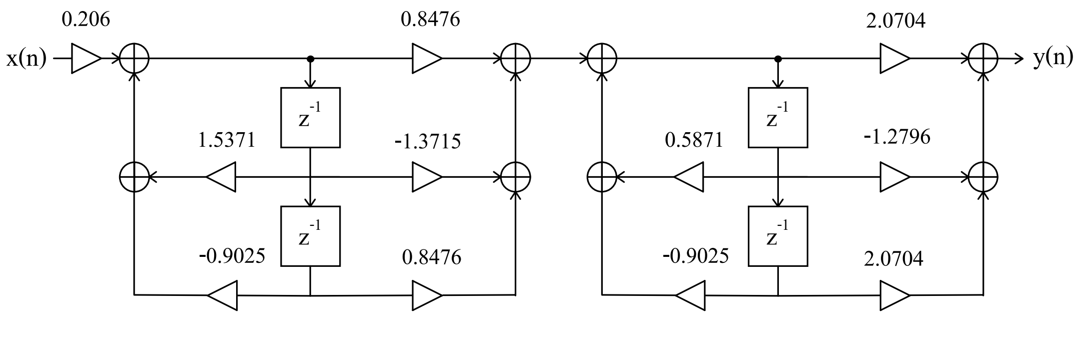
หา s_2 โดยใช้ s2 = sum_h(a1,b) ได้ $s_2 = 5.441$

หา s_3 โดยใช้ s3 = sum_h(a,b) ได้ $s_3 = 2.497$

จากนั้นหาอัตราส่วนที่ต้องใช้ในการคูณกับสัมประสิทธิ์ ได้

$1/s_1 = 0.206$, $s_1/s_2 = 0.839$, และ $s_2/s_3 = 1.192$

นำค่าอัตราส่วนเหล่านี้ไปคูณกับสัมประสิทธิ์ของตัวกรองตามที่แสดงในรูปที่ 10.11 จะได้ผลลัพธ์ของสัมประสิทธิ์สุดท้ายดังแสดงในรูปที่ 10.12



***รูปที่ 10.12** โครงสร้างของตัวกรองที่เป็นคำตอบของตัวอย่างที่ 10.6*

<u>ตัวอย่างที่ 10.7</u> ถ้านำระบบในตัวอย่างที่ 10.6 ไปใช้กับตัวประมวลผลที่มีความละเอียด 16 บิต โดยสัญญาณ x(n) รับมาจาก A/D ขนาด 14 บิต และสัญญาณขาออกส่งให้กับ D/A 14 บิต จะต้องเปลี่ยนแปลงค่าสัมประสิทธิ์ที่ใช้ในตัวอย่างที่ 10.6 อย่างไรบ้าง จึงจะได้ SNR สูงสุด

กรณีของการใช้ A/D ที่มีจำนวนบิตต่ำกว่าจำนวนบิตของตัวประมวลผลเช่นนี้นิยมทำกันในทางปฏิบัติ ตัวอย่างเช่น บอร์ดทดลอง TMS320C5x (TMS320C5x Evaluation Board ของบริษัท TI) ใช้ชิพ DSP เบอร์ TMS320C50 ซึ่งเป็นตัวประมวลผลแบบ fixed-point 16 บิต และใช้ A/D รวมกับ D/A เป็นชิพเดียวกันเบอร์ TLC32046 ซึ่งมีความละเอียด 14 บิต

การทำเช่นนี้ก็เหมือนกับการใช้สัญญาณขาเข้าที่มีการหารสี่ไปแล้ว แต่ขณะเดียวกันสัญญาณขาออกก็รับได้เพียงแค่ 14 บิต ซึ่งก็คือ ลดทอนจากค่าสูงสุดไปสี่เท่าเช่นกัน แต่ส่วนที่พิเศษก็คือ ค่าอื่น ๆ ที่เกิดขึ้นกึ่งกลางในระบบจะสามารถถูกแทนได้ด้วย 16 บิต หรือคิดง่าย ๆ ได้ว่า ค่าอื่น ๆ สามารถมีค่าได้มากกว่าค่าที่ขาเข้า และขาออก 4 เท่าโดยไม่เกิดโอเวอร์โฟล ดังนั้น การใช้ A/D และ D/A ที่มีจำนวนบิตต่ำลงนี้ มีผลดีกับการประมวลผลที่มีโอกาสเกิดโอเวอร์โฟลภายใน และแน่นอนว่าเป็นการเพิ่มความละเอียดของการคำนวณภายในด้วย

จุดที่ต้องเปลี่ยนแปลงจากรูปที่ 10.12 คือ

1) เนื่องจาก สัญญาณขาเข้าเหมือนมีการลดทอนไปแล้วเท่ากับ 1/4 หรือ 0.25 แต่การลดทอนเราต้องการการเท่ากับ 0.206 ดังนั้น ตัวคูณที่สัญญาณขาเข้านี้ต้องเปลี่ยนค่าเป็น 0.206 / 0.25 = 0.824 (ค่านี้ใกล้เคียง 1 ในทางปฏิบัติอาจลองตัดทิ้งไม่ต้องใช้ไปเลยก็ได้) ซึ่งจะทำให้การลดทอนโดยรวมที่สัญญาณขาเข้าเท่ากับ 0.206

2) สัมประสิทธิ์อื่น ๆ มีค่าเหมือนเดิมยกเว้นตัวสุดท้ายก่อนออก y(n) คือ $a_{0/2}$, $a_{1/2}$, และ $a_{2/2}$ ซึ่งจะต้องทำหน้าที่ลดทอนค่า y(n) ลง 4 เท่า ดังนั้น ค่าเหล่านี้ต้องถูกหารด้วย 4 และได้สัมประสิทธิ์ใหม่ คือ 0.5176, -0.3199, และ 0.1576

ความคลาดเคลื่อนจากการปัดเศษหลังการคูณ (Product Rounding)

ก่อนอื่นขอให้ทำความเข้าใจเรื่องการคูณในระบบเลขจำนวนเต็มก่อน และเหตุผลที่ว่าทำไมจึงต้องมีการปัดเศษหลังการคูณ สมมติว่าเรามีเลขบวกขนาด 5 บิต สองจำนวน ซึ่งใช้ M=0 ดังนั้น แต่ละจำนวนสามารถแทนตัวเลขได้ระหว่าง 0 ถึง 32 ถ้านำเลขสองจำนวนนี้มาบวกกัน จะได้ผลลัพธ์ที่มีค่าได้ระหว่าง 0 ถึง 1024 ซึ่งต้องใช้จำนวนบิตในการแทนถึง 10 บิต (M=0) ดังนี้

$$
\begin{array}{r}
\text{X X X X X .} \quad \times \\
\underline{\text{X X X X X .}} \\
\underline{\text{X X X X X X X X X X .}}
\end{array}
$$

ถ้าผลคูณจะต้องถูกนำเอาไปใช้คำนวณต่อไป เราต้องการให้ผลคูณมีขนาด 5 บิตเหมือนตัวตั้ง (มิฉะนั้นก็จะต้องใช้ตัวประมวลผลที่มีจำนวนบิตเพิ่มขึ้นเป็นสองเท่า) แต่ในกรณีนี้ทำไม่ได้ เพราะผลคูณจำเป็นต้องใช้จำนวนบิตในการแทนถึง 10 บิต การคูณในกรณีนี้ถือว่าการเกิดโอเวอร์โฟลจากการคูณขึ้น ซึ่งทำให้ใช้งานไม่ได้ในระบบประมวลผล

ถ้าพิจารณากรณีที่ตัวคูณเป็นรูปแบบเลข 5 บิต ที่มี M=5 และ N=0 จะเกิดอะไรขึ้น ยังแน่นอนว่าเราจะได้ผลคูณ 10 บิต แต่ทว่าเป็น 10 บิตที่แทนค่าแค่ 0 ถึง 32 เท่านั้น เนื่องจากตัวตั้งมีค่าได้ตั้งแต่ 0 ถึง 32 และตัวคูณมีค่าได้ตั้งแต่ 0 ถึง 1 ดังนั้น ผลคูณไม่มีทางได้ค่าเกิน 32 ซึ่งจะเป็นรูปแบบเลข 10 บิต ที่มี M=5 และ N=5 ดังนี้

$$\begin{array}{r} \mathrm{X\,X\,X\,X\,X.}\;\times \\ \underline{\mathrm{.X\,X\,X\,X\,X}} \\ \underline{\underline{\mathrm{X\,X\,X\,X\,X.X\,X\,X\,X\,X}}} \end{array}$$

ดังนั้น ถ้าต้องการนำผลคูณนี้ไปใช้คำนวณต่อ ก็สามารถทำได้โดยการ<u>ตัดเศษ (truncate)</u> 5 บิตหลังทิ้ง หรือ <u>ปัดเศษ (round)</u> 5 บิตหลัง การตัดเศษเทียบเท่ากับการตัด 5 บิตหลังทิ้ง ซึ่งทำได้ง่ายกว่าการปัดเศษในการใช้งานจริง เราจะได้ผลลัพธ์สุดท้ายหลังการตัดเศษ หรือปัดเศษเป็นรูปแบบ 5 บิตที่มี M=5 เหมือนตัวตั้งโดยที่ไม่มีทางเกิดโอเวอร์โฟล และจะสามารถนำไปคำนวณต่อไปได้โดยใช้ตัวประมวลผลที่มีจำนวนบิตเท่าเดิม สิ่งที่เราสูญเสียไปก็ คือ นัยสำคัญหรือความละเอียดของผลลัพธ์ 5 บิตล่าง ซึ่งก็ทำให้เกิดความคลาดเคลื่อนขึ้น

ในกรณีที่ตัวคูณมีค่ามากกว่า 1 ด้วย เช่น M=3 และ N=2 ผลลัพธ์ที่ได้จะเป็นรูปแบบ 10 บิตที่มี M=3 โดยเราต้องทำการปัดเศษ 3 บิตหลังทิ้ง จุดที่พิเศษขึ้นมา คือ กรณีนี้ผลลัพธ์มีโอกาสเกิดโอเวอร์โฟลได้ ถ้า 2 บิตหน้าของผลลัพธ์ไม่เท่ากับศูนย์ ดังนั้น ตัวตั้งและตัวคูณต้องมีขนาดไม่ใหญ่จนทำให้ผลลัพธ์มากเกินกว่าที่จะแทนค่าได้

$$\begin{array}{r} \mathrm{X\,X\;\;X\,X\,X.}\;\times \\ \underline{\mathrm{X\,X.X\,X\,X}} \\ \underline{\underline{\mathrm{X\,X\,X\,X\,X\,X\,X.X\,X\,X}}} \end{array}$$

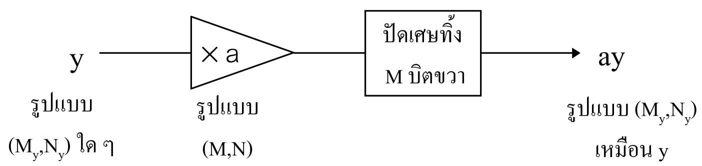
$\longleftrightarrow$ ต้องเป็น 0 มิฉะนั้น เป็น โอเวอร์โฟล
$\longleftrightarrow$ ผลลัพธ์ หลังปัดเศษ
$\longleftrightarrow$ ปัดทิ้ง

<u>สรุปว่า</u> ในการคูณเลขจำนวนเต็ม ตัวตั้งจะมีจุดทศนิยมอยู่ตรงไหนก็ได้ไม่ต้องสนใจ ส่วนตัวคูณ ถ้ามีจำนวนบิตอยู่หลังจุดทศนิยมเท่ากับ M บิต หลังจากคูณแล้วต้องปัดเศษผลลัพธ์ทิ้ง M บิต วิธีทำนี้จะทำให้ได้ผลลัพธ์หลังการปัดเศษถูกต้อง และเป็นรูปแบบเดียวกับตัวตั้ง

สำหรับรูปแบบเลข 2’s complement ที่แทนได้ทั้งบวกและลบ ก็มีกฎเกณฑ์นี้ตรงกัน โดยถ้าเราแยกเครื่องหมายออกมา (แปลงเป็นรูปแบบ sign-magnitude) ส่วนของขนาดก็จะมีวิธีคูณเหมือนดังที่กล่าวมาข้างต้น และส่วนเครื่องหมายก็สามารถคิดต่างหากโดยถ้าเครื่องหมายเหมือนกันก็ได้ผลลัพธ์เป็นบวก ถ้าต่างกันก็ได้ผลลัพธ์เป็นลบ ดังนั้น เลข 2’s complement ขนาด B บิต สองจำนวนคูณกันจะได้ผลลัพธ์ที่สามารถแทนได้ด้วย 2B - 1 บิต

รูปที่ 10.13 แสดงการคูณค่าสัมประสิทธิ์ a เข้ากับสัญญาณขาเข้า การคูณที่ถูกต้องจะต้องมีการปัดเศษทิ้ง M บิตหลังการคูณ เพื่อให้ได้ผลลัพธ์มีรูปแบบเหมือนกับสัญญาณขาเข้า โดยไม่ต้องสนใจเลยว่าสัญญาณขาเข้าจะมีจุดทศนิยมอยู่ตรงไหน นี่คือสาเหตุที่เราสามารถแทนค่าสัญญาณโดยใช้รูปแบบที่มีจุดทศนิยมไม่ลงตัวได้ ดังที่เห็นในตัวอย่างที่ 10.1 กระบวนการในรูปนี้ใช้ได้กับทั้งเลขบวกอย่างเดียว และเลข 2's complement

สำหรับ 2's complement ถ้า N=1 คือ $-1 \le a < 1$ จะได้ผลลัพธ์ที่ไม่มีทางเกิดโอเวอร์โฟล แต่ถ้า N>1 ผลลัพธ์จะมีโอกาสเกิดโอเวอร์โฟลได้ ในบางครั้งเราจำเป็นต้องใช้รูปแบบที่ N>1 ถ้าสัมประสิทธิ์มีค่าเกินช่วง -1 ถึง 1 กล่าวคือ ถ้าสัมประสิทธิ์มีค่าไม่เกิน -2 ถึง 2 ก็ใช้ N=2 และถ้าไม่เกิน -4 ถึง 4 ก็ใช้ N=3 เป็นต้น โอเวอร์โฟลที่ผลลัพธ์ของการคูณนี้ไม่ร้ายแรงอย่างที่คิด ถ้ามีการป้องกันโอเวอร์โฟลที่ผลลัพธ์ของการบวกดังที่กล่าวมาในหัวข้อก่อนแล้ว ดังที่จะอธิบายเหตุผลในส่วนต่อไปเมื่อกล่าวถึงการใช้ตัวเก็บผลบวกที่มีจำนวนบิตเป็น 2 เท่า



***รูปที่ 10.13** การคูณโดยใช้ระบบเลขจำนวนเต็ม ให้ได้ผลลัพธ์มีรูปแบบเหมือนตัวตั้ง*

<u>ตัวอย่างที่ 10.8</u> เลข 2's complement สองจำนวนมีรูปแบบ 8 บิต และ M=7 มีค่า 114 และ -87 ถ้านำเลขทั้งสองมาคูณกันแล้ว<u>ปัดเศษ</u>ให้ได้ผลลัพธ์มีรูปแบบ 8 บิตเหมือนเดิม จะได้ผลลัพธ์เท่ากับเท่าไรและมีความคลาดเคลื่อนไปเท่าไร

114	×	01110010	×
-87		10101001	
-9918		101100101000010	

ขอให้หมายเหตุไว้ว่า การคูณเลขฐานสองแบบ 2's complement มีวิธีคิดไม่เหมือนการคูณเลขปกติ ซึ่งการอธิบายวิธีคูณอยู่นอกเหนือเนื้อหาของหนังสือเล่มนี้ ผู้ที่สนใจรายละเอียดสามารถศึกษาได้จาก [4]

ผลลัพธ์ที่ได้จะอยู่ในรูปแบบ 15 บิต และ M=14 ซึ่งเราจะปัดเศษทิ้ง 7 บิต ได้เป็น 10110011_2 ซึ่งเป็นรูปแบบ 8 บิต และ M=7 เหมือนตัวตั้ง มีค่าเป็นฐานสิบเท่ากับ -77

ถ้าลองแปลงค่าจำนวนเต็มทั้งตัวตั้ง, ตัวคูณ, และผลลัพธ์เป็นค่าทศนิยมที่มันแทนอยู่ ด้วยการหารด้วย 2^M จะได้ค่าดังนี้

ตัวตั้ง คือ 114 เท่ากับค่า $114/2^7$ = 0.890625

ตัวคูณ คือ -87 เท่ากับค่า $-87/2^7$ = -0.6796875

ผลลัพธ์ก่อนการปัดเศษ คือ -9918 เท่ากับค่า $-9918/2^{14}$ = - 0.605346679688 เป็นค่าที่ถูกต้องร้อยเปอร์เซนต์ แต่ต้องแทนด้วย 15 บิต

ผลลัพธ์หลังการปัดเศษ คือ -77 เท่ากับค่า $-77/2^7$ = - 0.6015625

ดังนั้น การปัดเศษนี้ทำให้เกิดความคลาดเคลื่อนเท่ากับ

- 0.6015625 - (-0.605346679688) = 0.003784179688

การปัดเศษหลังการคูณนี้สามารถคำนวณโดยใช้ Matlab ได้ดังโปรแกรมที่ 10.5 โดยใส่ค่า x ซึ่งคือผลคูณที่ต้องการปัดเศษ และกำหนดค่า M และ N ของตัวคูณ นอกจากโปรแกรมจะทำการปัดเศษ M บิตทิ้งให้แล้ว ยังทำการตรวจสอบโอเวอร์โฟลให้ด้วย โดยถ้ามีโอเวอร์โฟลเกิดขึ้น โปรแกรมจะหยุดทำงาน และรายงานให้ผู้ใช้ทราบ

ตัวอย่างการคำนวณคำตอบของตัวอย่างที่ 10.8 สามารถทำได้โดยใช้โปรแกรมที่ 10.5 ดังนี้

rndfix(114*(-87), 7, 1)

ซึ่งจะได้ค่าผลคูณออกมาเท่ากับ -77 ตรงตามที่เราต้องการ โปรแกรมนี้จะมีประโยชน์ในการจำลองการประมวลผลโดยใช้รูปแบบจำนวนเต็ม ดังจะได้กล่าวในหัวข้อต่อไป

```
function y = rndfix(x,M,N);
max_y = 2^(M+N-1);                %หาค่าสูงสุดที่แทนได้ หลังปัดเศษ
y = round(x/2^M);                 %ปัดเศษ M บิต ถ้าต้องการเป็นการตัดเศษให้ใช้ floor แทน round
if max(abs(y)) > max_y            %ตรวจสอบว่าอยู่ในช่วงที่สามารถแทนค่าได้หรือไม่
   error('Overflow Error after rounding!');
end
y = y - (y==max_y);               %ในกรณีที่ y = 2^(M+N-1) พอดี เป็นจุดที่เกิดโอเวอร์โฟลทางบวก
                                  พอดี แต่จะอนุโลมให้ใช้ได้โดยใช้ y = 2^(M+N-1) - 1
```

***โปรแกรมที่ 10.5** ฟังก์ชั่น rndfix.m สำหรับปัดเศษทิ้ง M บิต (หลังการคูณ)*

ดังที่ได้เห็นแล้วว่า ทุกครั้งที่มีการคูณทำให้เกิดความคลาดเคลื่อนขึ้น โดยลักษณะความคลาดเคลื่อนของการปัดเศษหลังการคูณนี้ คล้ายคลึงกับความคลาดเคลื่อนจากการแบ่งขั้นสัญญาณ อาจมองได้ว่า การปัดเศษหลังการคูณ คือ การที่เราต้องการแทนค่าผลลัพธ์จากการคูณซึ่งมีความละเอียดมากด้วยจำนวนบิตที่จำกัดลง (เท่ากับจำนวนบิตเดิมของสัญญาณ) สมมติว่าขนาดของ 1 ขั้นของสัญญาณมีค่าเท่ากับ Q มีหน่วยเป็นแรงดัน และหาค่าได้จากสมการที่ 10.1 คือ

$$Q = \frac{R}{2^{B}} \tag{10.1}$$

ถ้าให้ y(n) เป็นสัญญาณที่เป็นผลลัพธ์ของการคูณก่อนปัดเศษ และ $y_e(n)$ เป็นผลลัพธ์หลังปัดเศษ เราสามารถมองได้ว่า ความคลาดเคลื่อนที่เกิดขึ้นเป็นเหมือนสัญญาณรบกวนที่ถูกเติมเข้ามาในระบบ และเรียกมันว่าสัญญาณรบกวนจากการคูณ (multiplication noise) ขอใช้สัญลักษณ์แทนว่า $e_e(n)$ ซึ่งจะได้

$$e(n) = y_e(n) - y(n) \tag{10.20}$$

ในกรณีของการปัดเศษ (rounding) สัญญาณรบกวนนี้จะมีค่าไม่เกินครึ่งหนึ่งของขั้น หรือ

$$- Q/2 < e_{round}(n) < Q/2 \tag{10.21}$$

เช่นเดียวกันกับการวิเคราะห์สัญญาณรบกวนจากการแบ่งขั้นสัญญาณ เราจะได้ว่าค่ากำลังเฉลี่ย หรือค่าเฉลี่ยของกำลังสองของสัญญาณ คือ (เหมือนสมการที่ 10.5)

$$P_{e,round} = \frac{Q^2}{12} \tag{10.22}$$

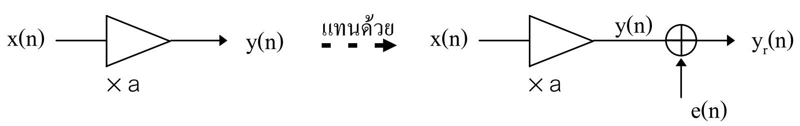
ในกรณีของการตัดเศษ (truncation) สัญญาณรบกวนนี้จะมีค่าเป็นลบเสมอ (ค่าจะถูกปัดลงเป็นขั้นที่ต่ำลงเสมอ) ดังนั้น

$$- Q < e_{trunc}(n) < 0 \tag{10.23}$$

เราสามารถพิสูจน์หากำลังเฉลี่ยของสัญญาณรบกวนสำหรับกรณีนี้ได้เป็น

$$P_{e,trunc} = \frac{Q^2}{3} \tag{10.24}$$

ในการวิเคราะห์สัญญาณรบกวนจากการคูณ เราจะต้องแทนตัวคูณทุกตัวในระบบด้วย ตัวคูณที่มีสัญญาณรบกวนปนที่ขาออก ดังแสดงในรูปที่ 10.14 จากนั้น จึงหาผลรวมของสัญญาณรบกวนทั้งหมดที่จะปรากฏที่สัญญาณขาออกของระบบ



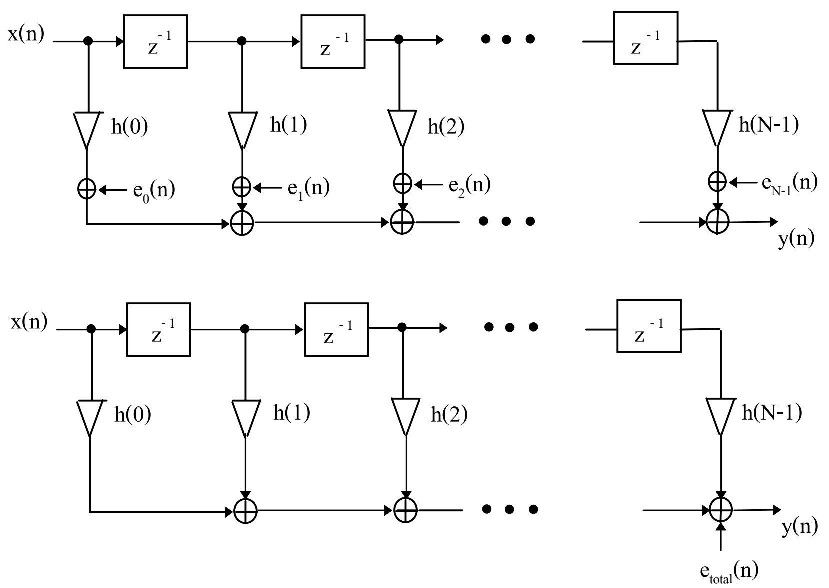
ย มี P_e

รูปที่ 10.14 *การแทนตัวคูณด้วยตัวคูณที่มีสัญญาณรบกวนปน เพื่อใช้ในการวิเคราะห์*

<u>การวิเคราะห์สัญญาณรบกวนจากการคูณในตัวกรอง FIR</u>

ตัวกรอง FIR อันดับ N-1 มีตัวคูณทั้งสิ้น N ตัว ดังนั้น จะมีสัญญาณรบกวนจากการคูณเกิดขึ้นทั้งสิ้น N สัญญาณ เนื่องจาก ตัวคูณทุกตัวมีผลลัพธ์ที่มาบวกกัน ณ จุดเดียวกัน ดังนั้นสัญญาณรบกวนที่เกิดขึ้นทุกตัวก็จะมาบวกกันที่สัญญาณขาออก ดังแสดงในรูปที่ 10.15 เราจะได้ว่า

$$e_{total}(n) = e_0(n) + e_1(n) + \ldots + e_{N-1}(n) \tag{10.25}$$



รูปที่ 10.15 *สัญญาณรบกวนจากการคูณที่เกิดขึ้นในตัวกรอง FIR*

เราจะใช้สมมติฐานว่า สัญญาณรบกวนแต่ละตัวไม่ขึ้นแก่กัน ซึ่งถึงแม้เป็นสมมติฐานที่ไม่ถูกต้อง แต่ก็พอใช้ได้ และก็ทำให้ง่ายต่อการวิเคราะห์ เนื่องจาก จะทำให้การหากำลังรวมของสัญญาณรบกวนทำได้จากการบวกกันของกำลังของสัญญาณรบกวนแต่ละตัว ดังนี้

$$P_{e,total} = P_{e1} + P_{e2} + \ldots + P_{e,N-1} \tag{10.26}$$

$$P_{e,total} = NP_e \tag{10.27}$$

ค่า $P_{e,total}$ ที่ได้นี้ ถือว่าสัมประสิทธิ์ทุกตัวของตัวกรองมีค่าไม่เท่ากับศูนย์ เพราะถ้ามีค่าใดที่เป็นศูนย์ ที่จุดนั้นก็ไม่มีการคูณ และก็จะไม่เกิดสัญญาณรบกวนขึ้น ซึ่งก็จะได้ $P_{e,total}$ ลดลง

ในทางปฏิบัตินิยมใช้ตัวบวกที่มีขนาดจำนวนบิตเป็นสองเท่าของตัวคูณ เช่น ในชิพ DSP ทั่ว ๆ ไป จะมีรีจีสเตอร์ที่ใช้บวก หรือ accumulator ที่มีจำนวนบิตเป็นสองเท่าของปกติ และสามารถทำการบวกที่จำนวนบิตเป็นสองเท่านี้ได้ ดังนั้น ในกรณีนี้จะทำให้ไม่จำเป็นต้องปัดเศษทันทีหลังการคูณ แต่เอาผลลัพธ์ที่ได้จากตัวคูณแต่ละตัวมาบวกกันก่อน เมื่อได้ผลลัพธ์สุดท้ายของการบวก จึงค่อยปัดเศษให้ผลลัพธ์สุดท้ายมีจำนวนบิตตามต้องการ เทคนิคนี้ทำให้สัญญาณรบกวนจากการคูณลดลงเหลือเพียงตัวเดียว (เพราะเหลือการปัดเศษแค่ครั้งเดียว) จะได้

$$P_{e,total} = P_e \tag{10.28}$$

นอกจากเป็นการลดสัญญาณรบกวนจากการคูณลงอย่างมากมายแล้ว เทคนิคนี้ยังช่วยป้องกันโอเวอร์โฟลที่อาจเกิดขึ้นจากการคูณอีกด้วย (ในกรณีที่ ตัวคูณใช้ $N > 1$) เพราะโอเวอร์โฟลจะเกิดขึ้นก็ต่อเมื่อมีการปัดเศษ ถ้าเราไม่ปัดเศษทันทีทันใด แต่บวกผลลัพธ์ทุกตัวก่อน ก็อาจมีโอกาสที่ผลลัพธ์สุดท้ายจะไม่เกิดโอเวอร์โฟล (ซึ่งสามารถป้องกันไม่ให้ผลลัพธ์สุดท้ายเกิดโอเวอร์โฟลได้ตามวิธีการใช้ตัวประกอบการสเกลที่ได้กล่าวมาแล้ว) และก็จะทำให้การเกิดโอเวอร์โฟลที่ผลลัพธ์จากตัวคูณบางตัวไม่ส่งผลใด ๆ ต่อการคำนวณ

<u>การวิเคราะห์สัญญาณรบกวนจากการคูณในตัวกรอง IIR โครงสร้าง direct form 1</u>

โครงสร้าง direct form 1 มีลักษณะคล้ายกับตัวกรอง FIR โดยสำหรับตัวกรองอันดับสองที่แสดงในรูปที่ 10.16 จะมีตัวคูณอยู่ 5 ตัว ซึ่งสัญญาณรบกวนที่เกิดจากทุกตัวสามารถมารวมกันได้เป็น $e_{total}(n)$ ดังแสดงในรูป

$e_{total}(n)$ ที่เกิดขึ้นในโครงสร้าง direct form 1 นี้ ถึงแม้มองดูเหมือนเป็นสัญญาณรบกวนที่เกิดขึ้นที่ขาออกเหมือนกรณีตัวกรอง FIR แต่จริง ๆ แล้วไม่ใช่ เนื่องจาก สัญญาณรบกวนนี้จะถูกป้อนกลับเข้ามาในระบบอีก แต่สัญญาณรบกวนของกรณีตัวกรอง FIR ไม่ถูกป้อนกลับมาในระบบ และนี่ก็เป็นเหตุผลที่ทำให้สัญญาณรบกวนจากการคูณมีผลต่อตัวกรอง IIR มากกว่าต่อตัวกรอง FIR

เราจะทำการหาว่าสัญญาณรบกวนจากการคูณที่ปรากฏออกมาที่สัญญาณขาออกจริง ๆ รวมแล้วมีค่าเท่าไร สมมติให้ $y(n)$ คือ สัญญาณขาออกขณะไม่มีสัญญาณรบกวนเกิดขึ้น และ $y_e(n)$ คือ สัญญาณขาออกในกรณีมีสัญญาณรบกวน โดยที่

$$y_e(n) = y(n) + e_{out}(n) \tag{10.29}$$

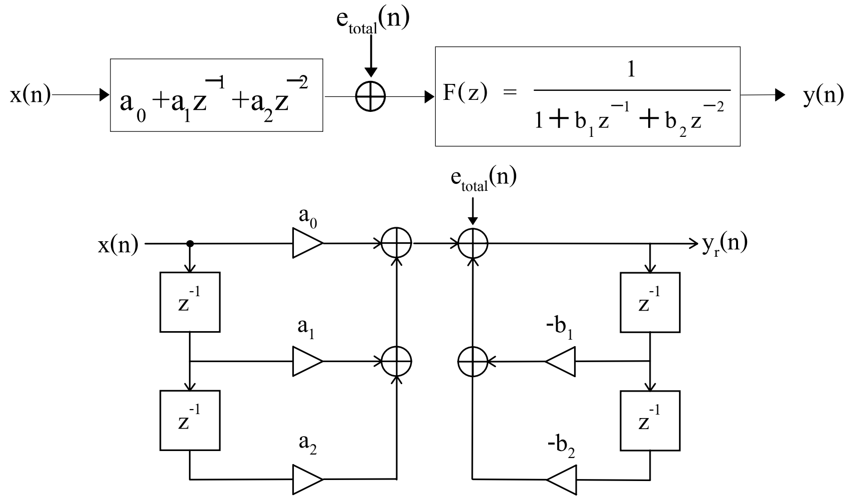
สมการผลต่างของ $y_e(n)$ เขียนได้เป็น

$$y_e(n) = a_0x(n) + a_1x(n\text{-}1) + a_2x(n\text{-}2) - b_1\, y_e(n\text{-}1) - b_2\, y_e(n\text{-}2) + e_{total}(n) \qquad (10.30)$$

เมื่อแปลง z ทั้งสมการจะได้

$$Y_e(z)\left\{\, 1 + b_1\, z^{-1} - b_2\, z^{-2}\right\} = X(z)\left\{a_0 + a_1\, z^{-1} + a_2\, z^{-2}\right\} + E_{total}(z)$$

$$Y_e(z) = \frac{a_0 + a_1z^{-1} + a_2z^{-2}}{1+ b_1z^{-1} + b_2z^{-2}}X(z) + \frac{E_{total}(z)}{1+ b_1z^{-1} + b_2z^{-2}} \qquad (10.31)$$



***รูปที่ 10.16** สัญญาณรบกวนจากการคูณที่เกิดขึ้นในตัวกรอง IIR โครงสร้าง direct form 1*

จะได้
$$E_{out}(z) = \frac{E_{total}(z)}{1+ b_1z^{-1} + b_2z^{-2}} = F(z)E_{total}(z) \qquad (10.32)$$

นั่นคือ สัญญาณรบกวนที่เกิดขึ้น จะผ่านฟังก์ชั่น F(z) ออกมาที่สัญญาณขาออก ดังแสดงในรูปที่ 10.16 สิ่งที่เราสนใจก็คือ กำลังของสัญญาณ $e_{out}(n)$ เป็นเท่าไร มีกฎว่า ถ้าสัญญาณขาเข้าของระบบหนึ่งมีลักษณะเป็นสัญญาณรบกวนขาว (white noise) กำลังของสัญญาณขาออกจะมีค่าเท่ากับกำลังของสัญญาณขาเข้าคูณด้วยผลบวกกำลังสองของผลตอบสนองต่ออิมพัลส์ของระบบ การพิสูจน์กฎนี้จะต้องใช้ความรู้เรื่องสัญญาณแรนดัม จึงไม่ขอกล่าวถึงในที่นี้ แต่จะยกผลมาใช้งาน โดยเราสามารถพิจารณาว่าสัญญาณ $e_{total}(n)$ มีลักษณะเป็นสัญญาณรบกวนขาว ซึ่งจะได้กำลังของสัญญาณรบกวนที่ขาออกเป็น

$$P_{e,out} = P_{e,total} \sum_{n=0}^{\infty} f^2(n) \qquad (10.33)$$

โดย f(n) คือ ผลตอบสนองต่ออิมพัลส์ของ F(z) ถ้าสัมประสิทธิ์ทุกตัวไม่เท่ากับศูนย์ จะได้กำลังรวมของสัญญาณรบกวนเป็น $P_{e,total} = 5P_e$ ซึ่งจะได้กำลังของสัญญาณรบกวนที่ขาออกเป็น

$$P_{e,out} = 5P_e \sum_{n=0}^{\infty} f^2(n) \tag{10.34}$$

ทำนองเดียวกัน ถ้าเราใช้ตัวบวกที่มีจำนวนบิตเป็นสองเท่าของตัวคูณ จะได้

$$P_{e,out} = 5P_e \sum_{n=0}^{\infty} f^2(n) \tag{10.35}$$

ค่าของ $\sum_{n=0}^{\infty} f^2(n)$ สามารถคำนวณได้จากโปรแกรมที่ 10.4 (sum_h2.m) ที่เราได้เขียนไว้สำหรับคำนวณ L_2 ซึ่ง คือ ค่ารากที่สองของผลบวกของกำลังสองของผลตอบสนองต่ออิมพัลส์

<u>การวิเคราะห์สัญญาณรบกวนจากการคูณในตัวกรอง IIR โครงสร้าง direct form 2</u>

โครงสร้างแบบ direct form 2 จะมีตัวคูณสองตัวอยู่ที่ตัวบวกด้านซ้าย และสามตัวอยู่ที่ตัวบวกด้วนขวา สมมติว่า สัญญาณรบกวนรวมของด้านซ้าย คือ $e_{t1}(n)$ และสัญญาณรบกวนรวมของด้านขวา คือ $e_{t2}(n)$ ดังแสดงในรูปที่ 10.17 เราสามารถหากำลังของสัญญาณรบกวนที่ขาออกได้ โดยมองว่า et1(n) จะต้องผ่านฟังก์ชั่น H(z) ส่วน et2(n) ไม่ต้องผ่านอะไร เพราะเกิดที่ขาออกโดยตรง ดังนั้น จะได้ว่า

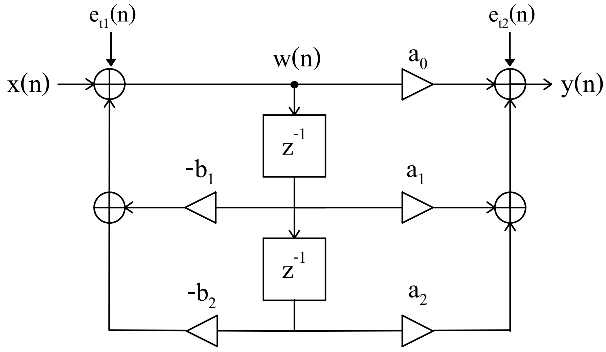
$$P_{e,out} = P_{t1} \sum_{n=0}^{\infty} h^2(n) + P_{t2} \tag{10.36}$$

ถ้าสัมประสิทธิ์ทุกตัวไม่เป็นศูนย์ จะได้

$$P_{e,out} = 2P_e \sum_{n=0}^{\infty} h^2(n) + 3P_e \tag{10.37}$$

ถ้าสามารถใช้ตัวบวกที่มีจำนวนบิตเป็นสองเท่าได้ ทั้งตัวบวกด้านซ้าย และขวา จะได้

$$P_{e,out} = P_e \sum_{n=0}^{\infty} h^2(n) + P_e \tag{10.38}$$



***รูปที่ 10.17** สัญญาณรบกวนจากการคูณที่เกิดขึ้นในตัวกรอง IIR โครงสร้าง direct form 2*

ตัวอย่างที่ 10.9 ตัวกรอง IIR อันดับสองดังในตัวอย่างที่ 10.4 และ 10.5 มีฟังก์ชั่นถ่ายโอนดังนี้

$$H(z) = \frac{0.1299z^2 + 0.2597z + 0.1299}{z^2 - 1.4836z + 0.8309}$$

จงหากำลังของสัญญาณรบกวนจากการคูณที่ปรากฏที่ขาออก เปรียบเทียบกันระหว่างกรณี direct form 1 และ direct form 2 กำหนดให้มีการสเกลเพื่อป้องกันโอเวอร์โฟลโดยใช้ขอบเขต L_1, ใช้การปัดเศษ (rounding) และการประมวลผลใช้ตัวบวกที่มีจำนวนบิตเป็นสองเท่าของปกติ

การวิเคราะห์สัญญาณรบกวนจากการคูณ ต้องทำหลังจากการปรับค่าสัมประสิทธิ์เพื่อป้องกันโอเวอร์โฟลเสมอ ดังนั้น เราจะใช้สัมประสิทธิ์ที่เป็นคำตอบของตัวอย่างที่ 10.4 และ 10.5 มาคำนวณในข้อนี้

สำหรับการปัดเศษ กำลังเฉลี่ยของสัญญาณรบกวนที่เกิดขึ้นแต่ละตัว คือ $P_e = P_{e,round} = \frac{Q^2}{12}$

<u>กรณี direct form 1</u> หาสัญญาณรบกวนที่ขาออกได้จากสมการที่ 10.35 คือ

$$P_{e,out} = P_e \sum_{n=0}^{\infty} f^2(n) = \frac{Q^2}{12} \sum_{n=0}^{\infty} f^2(n)$$

ค่า $\sum_{n=0}^{\infty} f^2(n)$ สามารถคำนวณได้จากโปรแกรม sum_h2 ใน Matlab ดังนี้

```
b = [1   -1.4836   0.8309];
sum_h(1,b)^2
```

ซึ่งได้ค่าออกมาเท่ากับ 162.32 เมื่อแทนค่าลงในสมการข้างต้น จะได้

$$P_{e,out} = \frac{Q^2}{12}(162.32) = 13.53Q^2$$

<u>กรณี direct form 2</u> หาสัญญาณรบกวนที่ขาออกได้จากสมการที่ 10.37 คือ

$$P_{e,out} = P_e \sum_{n=0}^{\infty} h^2(n) + P_e = \frac{Q^2}{12} \sum_{n=0}^{\infty} h^2(n) + \frac{Q^2}{12}$$

ค่า $\sum_{n=0}^{\infty} h^2(n)$ สามารถคำนวณได้จากโปรแกรม sum_h2 ใน Matlab ดังนี้

```
a = [0.2689  0.5376  0.2689];
b = [1   -1.4836   0.8309];
sum_h(a,b)^2
```

ซึ่งได้ค่าออกมาเท่ากับ 162.53 เมื่อแทนค่าลงในสมการข้างต้น จะได้

$$P_{e,out} = \frac{Q^2}{12}(162.53) + \frac{Q^2}{12} = 13.63Q^2$$

ปรากฏว่า ค่าที่ได้ออกมาของทั้งสองกรณีใกล้เคียงกันมาก อย่างไรก็ตาม สำหรับฟังก์ชั่นถ่ายโอนใด ๆ เราไม่สามารถบอกได้ล่วงหน้าว่า โครงสร้างแบบใดจะให้สัญญาณรบกวนต่ำกว่า เพราะค่าจะขึ้นกับฟังก์ชั่นถ่ายโอนนั้น ๆ ซึ่งต้องพิจารณาเป็นกรณี ๆ ไป

<u>การวิเคราะห์สัญญาณรบกวนจากการคูณในโครงสร้างอนุกรม</u>

ในที่นี้จะยกตัวอย่างโครงสร้างอนุกรมอันดับ 4 ที่มีโครงสร้างย่อยอันดับ 2 เป็นแบบ direct form 2 ดังแสดงในรูปที่ 10.18 จะเห็นได้ว่า มีสัญญาณรบกวนจากการคูณเกิดขึ้นที่ 3 ตำแหน่ง คือ

ก) ที่ขาเข้าของระบบ มาจากตัวคูณ 2 ตัว สัญญาณรบกวนนี้ต้องผ่านฟังก์ชั่น H(z)

ข) ที่กึ่งกลางระบบ มาจากตัวคูณ 5 ตัว สัญญาณรบกวนนี้ต้องผ่านฟังก์ชั่น $H_2(z)$

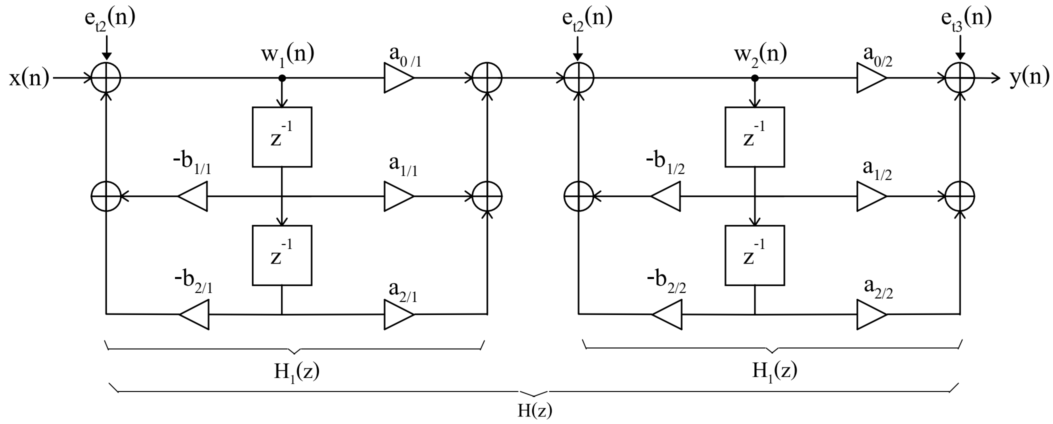
ค) ที่ขาออกของระบบ มาจากตัวคูณ 3 ตัว

ดังนั้น จะได้ผลรวมของสัญญาณรบกวนที่เกิดขึ้นที่สัญญาณขาออก คือ

$$P_{e,out} = 2P_e \sum_{n=0}^{\infty} h^2(n) + 5P_e \sum_{n=0}^{\infty} h_2^2(n) + 3P_e \qquad (10.39)$$

และในกรณีที่ใช้ตัวบวกที่มีจำนวนบิตเป็นสองเท่า จะได้ผลรวมของสัญญาณรบกวนเป็น

$$P_{e,out} = P_e \sum_{n=0}^{\infty} h^2(n) + P_e \sum_{n=0}^{\infty} h_2^2(n) + P_e \qquad (10.40)$$



***รูปที่ 10.18** สัญญาณรบกวนจากการคูณที่เกิดขึ้นในโครงสร้างอนุกรม*

การจำลองการประมวลผลด้วยระบบเลขจำนวนเต็มใน Matlab

เนื่องจาก Matlab เป็นโปรแกรมที่เราทำงานด้วยเป็นส่วนใหญ่ ในการทดสอบ หรือออกแบบอัลกอริธึมในการประมวลผลสัญญาณ ดังนั้น ถ้าสามารถจำลองการประมวลผลด้วยระบบเลขจำนวนเต็มได้ใน Matlab ก็จะทำให้สะดวกต่อการทำงานมาก อย่างไรก็ตาม ก็มีข้อเสียในการใช้ Matlab ทำหน้าที่ดังกล่าว เนื่องจาก Matlab แทนตัวเลขต่าง ๆ ด้วยระบบเลขอิงดรรชนีขนาด 64 บิต และยังไม่มีทางเลือกให้ผู้ใช้เลือกเป็นระบบเลขจำนวนเต็ม ดังนั้น หมายถึงว่า เราต้องจำลองการคำนวณเลขจำนวนเต็มด้วยเลขอิงดรรชนี ซึ่งทำให้การคำนวณช้าลงกว่าการคำนวณปกติมาก

ทางเลือกอื่นสำหรับการคำนวณระบบเลขจำนวนเต็ม ได้แก่ การใช้ Fixed-Point Toolbox ของ Matlab ซึ่งเป็นชุดฟังก์ชั่นพิเศษสำหรับคำนวณระบบเลขจำนวนเต็มได้ แต่ข้อเสีย คือ ปัจจุบันชุดฟังก์ชั่นนี้เป็นฟังก์ชั่นที่เรียกใช้ได้ใน Simulink เท่านั้น (Simulink เป็นสิ่งแวดล้อมพิเศษใน Matlab ซึ่งใช้จำลองระบบต่าง ๆ ได้ทั้งแบบต่อเนื่อง และไม่ต่อเนื่อง มีการติดต่อกับผู้ใช้แบบกราฟฟิก) หรืออีกทางเลือกหนึ่ง คือ การเขียนอัลกอริธึมด้วยภาษาซี และทำให้ติดต่อกับ Matlab ด้วยการเก็บสัญญาณขาเข้าและขาออกเป็นไฟล์ หรือคอมไพล์ให้เป็นไฟล์ที่สามารถเรียกใช้ได้จาก Matlab เลย (เรียกว่า ไฟล์ MEX) ก็สามารถทำได้

ในที่นี้ผู้เขียนขอแนะนำวิธีแรกสุด คือ การใช้การคำนวณปกติของ Matlab มาจำลองเป็นการคำนวณเลขจำนวนเต็ม ซึ่งเป็นวิธีที่ง่าย สะดวก และเข้าถึงกลไกของการคำนวณได้ดีด้วย และถ้าพิจารณาถึงความเร็วอันมหาศาลของไมโครคอมพิวเตอร์ในปัจจุบัน ทางเลือกนี้ก็ถือว่าไม่เลวนัก

การจำลองการคำนวณเลขจำนวนเต็มนี้ เท่ากับเป็นการวิเคราะห์ผลโดยรวมของความคลาดเคลื่อนทุกชนิดที่กล่าวมา ไม่ว่าจะเป็นความคลาดเคลื่อนจากการปัดเศษสัมประสิทธิ์, โอเวอร์โฟล, หรือสัญญาณรบกวนจากการคูณ เป็นการทดสอบที่เหมือนกับการที่จะนำไปใช้งานจริง นอกจากนี้ ในกรณีที่นำไปใช้สร้างเป็นวงจรรวม หรือไอซี ยังสามารถใช้การจำลองนี้สร้างชุดสัญญาณเพื่อ

ทดสอบวงจร (test vector) ง่าย ๆ ได้อีกด้วย ซึ่งประกอบด้วยสัญญาณขาเข้าชุดหนึ่ง และสัญญาณขาออกที่จะต้องได้เมื่อสัญญาณขาเข้านี้เข้าไป

เราได้เตรียมพร้อมสำหรับการใช้ Matlab มาจำลองการคำนวณเลขจำนวนเต็มมาบ้างแล้ว ด้วยการเขียนฟังก์ชั่นรองรับไว้สองฟังก์ชั่นในหัวข้อก่อนหน้านี้ ฟังก์ชั่นแรก คือ fixpnt.m (โปรแกรมที่ 10.2) ซึ่งเป็นฟังก์ชั่นสำหรับแปลงค่าทั่วไปให้เป็นค่าจำนวนเต็ม ฟังก์ชั่นที่สอง คือ rndfix.m (โปรแกรมที่ 10.5) ซึ่งเป็นฟังก์ชั่นที่ทำการปัดเศษผลลัพธ์หลังการคูณ ทั้งสองฟังก์ชั่นให้ผลลัพธ์เป็นเลขจำนวนเต็มที่มีเครื่องหมาย เทียบเท่ากับรูปแบบ 2's complement โดยเรากำหนดค่า M และ N ให้กับฟังก์ชั่น

หลักการง่าย ๆ ของการใช้ Matlab เพื่อจำลองการคำนวณเลขจำนวนเต็ม คือ

1) กำหนดค่า M และ N ให้เหมาะสม แล้วแปลงสัมประสิทธิ์ของระบบให้เป็นจำนวนเต็ม โดยใช้ฟังก์ชั่น fixpnt.m

2) สัญญาณขาเข้าต้องเป็นจำนวนเต็ม มีขนาดตั้งแต่ 2^{-B+1} จนถึง 2^{B-1} (B เป็นจำนวนบิตที่จะแทนค่าสัญญาณ และ B=M+N) ถ้ามีสัญญาณขาเข้าที่เป็นทศนิยม ต้องปรับขนาดให้เหมาะสมก่อน โดยใช้ฟังก์ชั่น fixpnt.m

3) ในส่วนประมวลผล ถ้ามีการคูณ ต้องใช้ฟังก์ชั่น rndfix.m กระทำกับผลลัพธ์ของการคูณ

4) ในส่วนประมวลผล ถ้ามีการบวก ต้องใช้ฟังก์ชั่นตรวจสอบผลลัพธ์ว่าเกิดโอเวอร์โฟลหรือไม่ ในกรณีที่เป็นการบวกสะสม ให้ตรวจเฉพาะผลลัพธ์สุดท้ายเท่านั้น ฟังก์ชั่นนี้เรายังไม่ได้เขียนไว้ ซึ่งสามารถเขียนได้ง่าย ๆ ดังแสดงในฟังก์ชั่น chkovr.m (โปรแกรมที่ 10.6)

```
function  chkovr(x,B);
if max(abs(x)) > 2^(B-1)
   error('Overflow Error after Adding!');
end
```

โปรแกรมที่ 10.6 *ฟังก์ชั่น chkovr.m สำหรับตรวจจับโอเวอร์โฟลจากการบวก*

ในการประมวลผลเพื่อจำลองระบบจำนวนเต็ม ถ้าหากมีโอเวอร์โฟล โปรแกรมจะหยุดการทำงาน และแจ้งให้ทราบว่าโอเวอร์โฟลมาจากฟังก์ชั่นใด ซึ่งก็จะทำให้รู้ว่าเป็นโอเวอร์โฟลจากสาเหตุใด

ในที่นี้จะได้ยกตัวอย่าง การจำลองการทำงานของตัวกรอง IIR แบบโครงสร้าง direct form 1 และ direct form 2

การจำลองตัวกรอง IIR โครงสร้าง direct form 1

โปรแกรมที่ 10.7 เป็นฟังก์ชั่นที่คำนวณตัวกรอง IIR แบบ direct form 1 โดยฟังก์ชั่นจะรับค่า

ก) a และ b ซึ่งเป็นเวคเตอร์ของสัมประสิทธิ์ของเศษ และส่วนตามลำดับ

ข) M และ N กำหนดรูปแบบของจำนวนเต็มที่จะใช้กับ a และ b

ค) x เป็นเวคเตอร์ของสัญญาณขาเข้า ซึ่งต้องอยู่ในรูปแบบจำนวนเต็มแล้ว

และจะให้สัญญาณขาออกเป็นเวคเตอร์ y ซึ่งก็จะอยู่ในรูปแบบจำนวนเต็มเช่นเดียวกับขาเข้า ฟังก์ชั่นนี้เขียนให้สามารถคำนวณตัวกรองอันดับใด ๆ ก็ได้

ให้สังเกตตัวอักษรที่เน้นเป็นตัวเข้ม ซึ่งเป็นส่วนที่เติมเข้ามาเพื่อให้ทำให้ Matlab จำลองการคำนวณแบบจำนวนเต็มได้ ถ้าตัดส่วนที่เป็นตัวเข้มออก ก็จะได้โปรแกรมที่ทำการคำนวณตามปกติโดยปราศจากผลของความคลาดเคลื่อนที่มาจากการใช้เลขจำนวนเต็ม

```
function y = iir1(a,b,x,M,N)
B = M+N;
a = fixpnt(a,M,N);
b = fixpnt(b,M,N);
Nx=length(x); Na=length(a); Nb=length(b);
xold=zeros(1,Na); yold=zeros(1,Nb);
y=zeros(1,Nx);

for i=1:Nx
   xold(1)=x(i);
   for j=1:Na
      y(i) = y(i) + rndfix(a(j)*xold(j),M,N);
   end
   for j=2:Nb
      y(i) = y(i) - rndfix(b(j)*yold(j),M,N);
   end
   chkovr(y(i), B);
   yold(1)=y(i);
   xold(2:Na) = xold(1:Na-1);
   yold(2:Nb) = yold(1:Nb-1);
end
```

โปรแกรมที่ 10.7 *ฟังก์ชั่น iir1.m สำหรับคำนวณตัวกรอง IIR โครงสร้าง direct form 1*

```
function y = iir1a(a,b,x,M,N)
a = fixpnt(a,M,N);
b = fixpnt(b,M,N);
Nx=length(x); Na=length(a); Nb=length(b);
xold=zeros(1,Na); yold=zeros(1,Nb);
y=zeros(1,Nx);

for i=1:Nx
   xold(1)=x(i);
   for j=1:Na
      y(i) = y(i) + a(j)*xold(j);
   end
   for j=2:Nb
      y(i) = y(i) - b(j)*yold(j);
   end
   y(i) = rndfix(y(i),M,N);
   yold(1)=y(i);
   xold(2:Na) = xold(1:Na-1);
   yold(2:Nb) = yold(1:Nb-1);
end
```

โปรแกรมที่ 10.8 *ฟังก์ชั่น iir1a.m เหมือน 10.7 แต่จำลองว่าใช้ตัวบวกที่มีจำนวนบิตเป็นสองเท่า*

โปรแกรมที่ 10.8 เป็นระบบที่จำลองว่ามีตัวบวกที่มีจำนวนบิตเป็นสองเท่าของปกติ เพราะฉะนั้น ผลลัพธ์จากการคูณจะไม่ถูกปัดเศษทันที แต่จะมาบวกกันก่อน แล้วค่อยปัดเศษที่ผลลัพธ์สุดท้ายแค่ครั้งเดียว ให้สังเกตตำแหน่งของการใช้ฟังก์ชั่น rndfix ที่เปลี่ยนไป

ตัวอย่างที่ 10.10 ศึกษาผลของการใช้ระบบเลขจำนวนเลขจำนวนเต็มขนาด 8 บิต กับตัวกรอง LPF บัตเตอร์เวิร์ธ ที่มีความถี่ตัดที่ 0.6π โดยเทียบผลตอบสนองต่ออิมพัลส์ และสเปกตรัมของมัน

เราจะใช้ฟังก์ชั่นสำเร็จรูปใน DSP Toolbox ชื่อ butter เพื่อหาสัมประสิทธิ์ของตัวกรองนี้ ดังนี้

[a,b]=butter(4,0.6) หาสัมประสิทธิ์ของ LPF มีอันดับ=4 และความถี่ตัด=0.4π

ได้ a = [0.1672 0.6687 1.0031 0.6687 0.1672]

b = [1.0000 0.7821 0.6800 0.1827 0.0301]

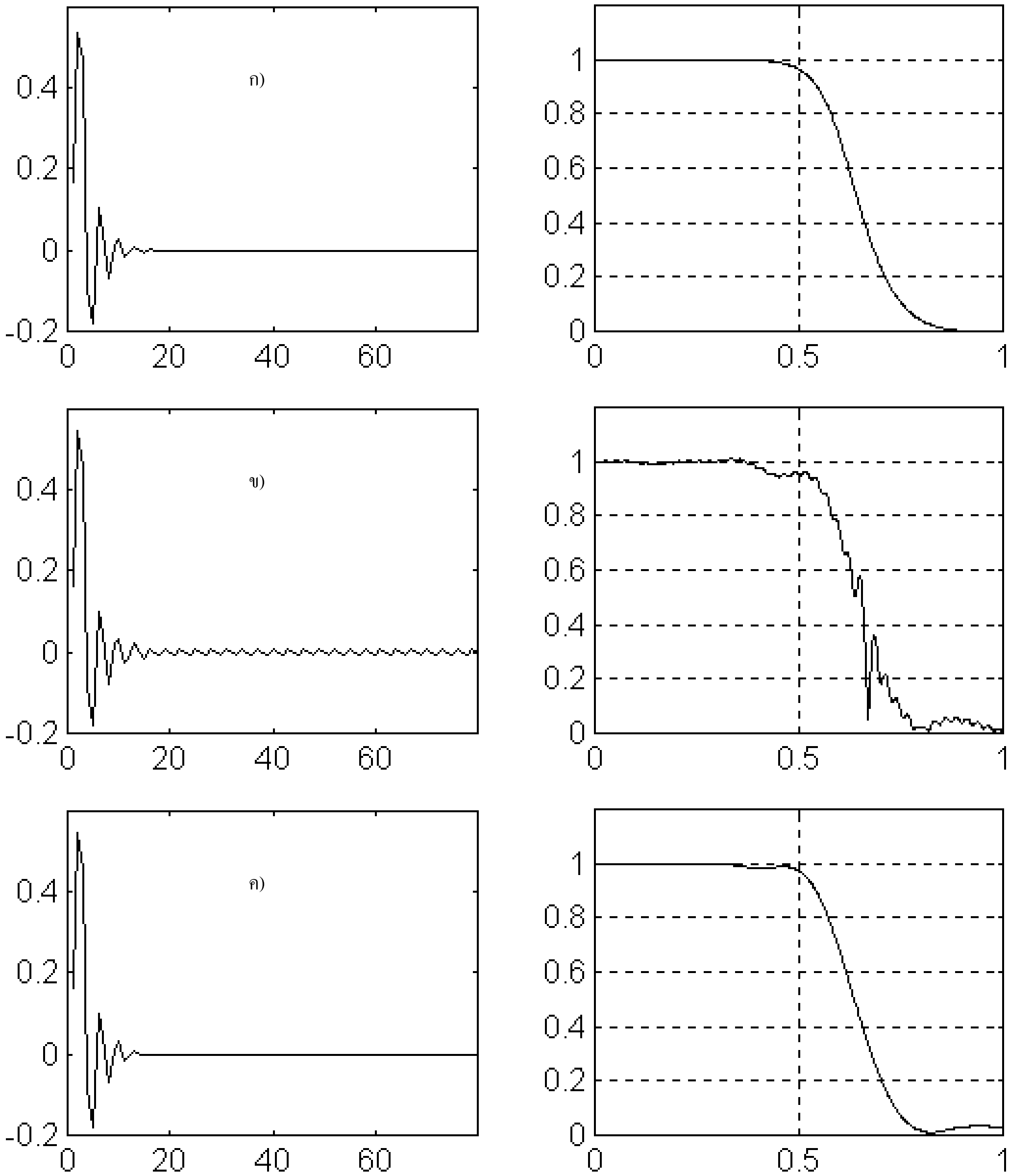
กรณีที่ไม่มีความคลาดเคลื่อน สามารถคำนวณหาผลตอบสนองต่ออิมพัลส์ได้โดยใช้โปรแกรมที่ 10.7 (iir1.m) ที่ตัดส่วนที่เป็นตัวเข้มที่ได้แสดงไว้ออก หรือจะใช้ฟังก์ชั่น filter ใน DSP Toolbox ก็ได้ ซึ่งจะได้ผลตอบสนองต่ออิมพัลส์ และสเปกตรัมของมันดังในรูปที่ 10.19 ก) สเปกตรัมของผลตอบสนองต่ออิมพัลส์ในที่นี้ก็คือ ผลตอบสนองเชิงความถี่ของระบบนั่นเอง

ในกรณีที่ใช้ระบบเลข 8 บิต เราจะใช้ฟังก์ชั่น iir1.m และ iir1a.m ในการคำนวณ จากสัมประสิทธิ์ในข้อนี้พบว่า ทุกตัวมีขนาดไม่เกิน 1 ยกเว้น a(3)=1.0031 ซึ่งเราจะอนุโลมปัดให้ a(3)=1 ดังนั้น สามารถเลือกให้ M=7 และ N=1 ได้ เราจะใช้คำสั่งต่อไปนี้ ในการหาผลตอบสนองต่ออิมพัลส์ และสเปกตรัมของมัน (กรณีที่ใช้ iir1.m)

```
M=7; N=1;
x = [128 zeros(1,79)];        ให้ขาเข้าเป็นอิมพัลส์ และเทียบขนาดว่า 128 มีค่าเท่ากับ 1
y = iir1(a,b,x,M,N);
y = y/128;                    หารผลลัพธ์ที่ได้กลับเป็นค่าปกติ เพื่อนำไปวิเคราะห์ต่อ
figure(1);  plot(y)
figure(2);  dtft(y)
```

ผลที่ได้แสดงในรูปที่ 10.19 ข) และ ค) จะสังเกตได้ถึงความผิดเพี้ยนที่เกิดขึ้นกับผลตอบสนองต่ออิมพัลส์ และสเปกตรัมของมัน จุดที่ต้องระวัง ก็คือ สเปกตรัมของผลตอบสนองต่ออิมพัลส์ในกรณีนี้ จะไม่ใช่ของผลตอบสนองเชิงความถี่ของระบบที่เราจะได้ ทั้งนี้เนื่องจาก ผลของสัญญาณรบกวนจากการคูณเป็นผลที่ไม่เป็นเชิงเส้น การเกิดผลทางความถี่ต่อผลตอบสนองต่ออิมพัลส์ในลักษณะหนึ่ง ไม่ได้หมายความว่า ผลนั้นจะเหมือนกันเมื่อสัญญาณขาเข้าเป็นสัญญาณอื่น ดังนั้น รูปของสเปกตรัมที่ได้ในรูปที่ 10.19 ข) และ ค) จะใช้เพื่อเปรียบเทียบได้กับรูปที่ 10.19 ก) เท่านั้น <u>ไม่</u>สามารถถือได้ว่าเป็นผลตอบสนองเชิงความถี่ของระบบที่เปลี่ยนไป

h(n) |H|



รูปที่ 10.19 *ผลตอบต่ออิมพัลส์ และสเปกตรัมของมันจากตัวอย่างที่ 10.10*
ก) เมื่อไม่มีผลของความคลาดเคลื่อนใด ๆ ข) เมื่อใช้กับระบบเลข 8 บิต
ค) เมื่อใช้กับระบบเลข 8 บิต แต่ใช้ตัวบวก 16 บิต

ผลในรูปที่ 10.19 ข) มีลักษณะที่ประหลาด คือ ผลตอบไม่ลู่เข้าสู่ศูนย์ ทั้งที่สัญญาณขาเข้าเป็นศูนย์ไปนานแล้ว โดยผลตอบจะแกว่งเป็นคาบ มีค่า (ก่อนหารกลับเป็นทศนิยม) เป็น … ,0, -1, 1, 0, -1, 1, 0, -1, 1, 0, … ปรากฏการณ์นี้ เรียกว่า limit cycle oscillation ซึ่งเป็นผลจากความคลาดเคลื่อนจากการคูณที่เกิดขึ้นกับระบบที่มีการป้อนกลับในบางกรณีเท่านั้น ในหนังสืออ้างอิง [4] ได้แสดงการวิเคราะห์สำหรับกรณีตัวกรองอันดับสอง

ในตัวอย่างที่ 10.10 นี้ ไม่ได้ทำการลดค่าสัมประสิทธิ์เพื่อป้องกันโอเวอร์โฟล ซึ่งบังเอิญเราใช้สัญญาณขาเข้าที่มีค่าน้อย ทำให้ไม่พบปัญหาเกี่ยวกับโอเวอร์โฟล ในการทดสอบจริง ๆ เราอาจนำ

ตัวอย่างของสัญญาณขาเข้าที่ใกล้เคียงกับการใช้งานจริงมาใส่ให้กับการจำลอง และก็สามารถทดสอบเปลี่ยนค่าตัวประกอบการสเกลให้เหมาะสม เพื่อให้ป้องกันโอเวอร์โฟลได้

การจำลองตัวกรอง IIR โครงสร้าง direct form 2

โปรแกรมที่ 10.9 และ 10.10 (iir2 และ iir2a) เป็นฟังก์ชั่นที่คำนวณตัวกรอง IIR แบบ direct form 2 ซึ่งหลักการ และการใช้งานโปรแกรมทั้งสองก็เหมือนกับของกรณีโครงสร้าง direct form 1

จุดพิเศษที่เพิ่มเข้ามา ก็คือ โปรแกรม iir2 และ iir2a อนุญาตให้เราเลือกใช้ค่า M ที่แตกต่างกันได้ สำหรับสัมประสิทธิ์ของโพลิโนเมียลเศษ (a_i) และโพลิโนเมียลส่วน (b_i) โดยที่โปรแกรมจะรับค่า M, N, M2, และ N2 เข้าไป โดยที่ M และ N จะถูกใช้กับสัมประสิทธิ์ของโพลิโนเมียลเศษ ส่วน M2 และ N2 จะถูกใช้กับสัมประสิทธิ์ของโพลิโนเมียลเศษ ถ้าผู้ใช้ไม่ใส่ค่า M2 และ N2 โปรแกรมจะตั้งให้เท่ากับ M และ N โดยอัตโนมัติ คำถาม คือ ทำไมเราจึงสามารถใช้ค่า M ที่แตกต่างกันได้ภายในระบบเดียวกัน

เนื่องจาก สัมประสิทธิ์ทุกตัวในตัวกรองจะถูกใช้สำหรับเป็นตัวคูณ ดังนั้น จริง ๆ แล้วเราสามารถเลือกค่า M และ N สำหรับสัมประสิทธิ์แต่ละตัว และทุกตัวได้โดยอิสระต่อกัน ด้วยเงื่อนไขเพียงอย่างเดียวว่า หลังการคูณด้วยสัมประสิทธิ์ค่าหนึ่ง ๆ จะต้องทำการปัดเศษผลลัพธ์ทิ้ง M บิต โดยใช้ค่า M ที่สัมประสิทธิ์นั้นใช้อยู่ ถ้าเราปัดเศษถูกต้องในลักษณะนี้จะได้รูปแบบของสัญญาณขาออก

```
function y = iir2(a,b,x,M,N,M2,N2)
if nargin==5
   M2=M; N2=N;
end
B = M+N;
a = fixpnt(a,M,N);
b = fixpnt(b,M2,N2);
Nx=length(x); Na=length(a); Nb=length(b);
Nab=max(length(a),length(b));
wold=zeros(1,Nab);
y=zeros(1,Nx);

for i=1:Nx
   wold(1) = x(i);
   for j=2:Nb
      wold(1) = wold(1) - rndfix(b(j)*wold(j),M2,N2);
   end
   chkovr( wold(1),B );

   for j=1:Na
      y(i) = y(i) + rndfix(a(j)*wold(j),M,N);
   end
   chkovr( y(i),B );
   wold(2:Nab) = wold(1:Nab-1);
end
```

โปรแกรมที่ 10.9 *ฟังก์ชั่น iir2.m สำหรับคำนวณตัวกรอง IIR โครงสร้าง direct form 2*

```
function y = iir2a(a,b,x,M,N,M2,N2)
if nargin==5
   M2=M; N2=N;
end
```

```
    B = M+N;
    a = fixpnt(a,M,N);
    b = fixpnt(b,M2,N2);
    Nx=length(x); Na=length(a); Nb=length(b);
    Nab=max(length(a),length(b));
    wold=zeros(1,Nab);
    y=zeros(1,Nx);

    for i=1:Nx
       wold(1) = 0;
       for j=2:Nb
          wold(1) = wold(1) - b(j)*wold(j);
       end
       wold(1) = x(i) + rndfix(wold(1),M2,N2);
       chkovr( wold(1),B );
       for j=1:Na
          y(i) = y(i) + a(j)*wold(j);
       end
       y(i) = rndfix(y(i),M,N);
       wold(2:Nab) = wold(1:Nab-1);
    end
```

โปรแกรมที่ 10.10 *ฟังก์ชั่น iir2a.m เหมือน 10.7 แต่จำลองว่าใช้ตัวบวกที่มีจำนวนบิตเป็นสองเท่า*

ของตัวคูณไม่เปลี่ยนแปลงจากสัญญาณขาเข้าดังที่ได้ศึกษาไปในเรื่องการคูณระบบเลขจำนวนเต็ม ดังนั้น สำหรับโปรแกรม iir1 และ iir2 จริง ๆ แล้ว สามารถเลือกสัมประสิทธิ์ที่แตกต่างกันสำหรับสัมประสิทธิ์แต่ละตัวได้ เช่น ถ้ามีสัมประสิทธิ์เพียงตัวเดียวในระบบมีขนาดเกิน 1 แต่ไม่เกิน 2 เราสามารถใช้ค่า N=2 สำหรับสัมประสิทธิ์ตัวนั้น ในขณะที่ใช้ค่า N=1 สำหรับสัมประสิทธิ์ตัวอื่นได้

สำหรับโปรแกรม iir1a ซึ่งทำการบวกผลลัพธ์จากการคูณก่อนจึงค่อยปัดเศษ ในกรณีนี้ จำเป็นต้องใช้ค่า M เท่ากันสำหรับสัมประสิทธิ์ทุกตัว เพราะ ต้องทำให้ผลลัพธ์จากการคูณทุกตัวมีรูปแบบเหมือนกัน ส่วนโปรแกรม iir2a ซึ่งทำการปัดเศษหลังการบวกเช่นกัน แต่โปรแกรมนี้มีการบวก 2 แยกกันอยู่สองส่วน คือ บวกผลลัพธ์จากการคูณของสัมประสิทธิ์เศษ (a) ส่วนหนึ่ง และบวกผลลัพธ์จากการคูณของสัมประสิทธิ์ส่วน (b) อีกส่วนหนึ่ง ดังนั้น กรณีนี้ ต้องใช้ค่า M ของสัมประสิทธิ์เศษทุกตัวเท่ากัน แต่ไม่จำเป็นต้องเท่ากับค่า M ของสัมประสิทธิ์ของส่วน ดังที่ได้แสดงไว้แล้วในโปรแกรมดังกล่าว

ตัวอย่างการประมวลผลด้วยระบบเลขจำนวนเต็มในภาษาซี

การนำระบบประมวลผลสัญญาณดิจิตอลไปใช้จริง สามารถทำได้โดย การใช้ฮาร์ดแวร์ (ทำเป็นไอซีเฉพาะงาน) หรือ การใช้ซอฟท์แวร์ร่วมกับชิพ DSP หรือ CPU ทั่ว ๆ ไป ดังที่ได้กล่าวไว้ในบทนำ ในที่นี้จะขอยกตัวอย่าง การทำตัวกรอง IIR แบบโครงสร้าง direct form 2 โดยใช้ภาษาซี ซึ่งเขียนเป็นฟังก์ชั่น iir2a ดังแสดงในโปรแกรมที่ 10.11 ฟังก์ชั่นนี้มีการทำงานเทียบเท่ากับโปรแกรม iir2a.m ใน Matlab ในหัวข้อที่แล้ว

ฟังก์ชั่นนี้จะรับตัวแปรเข้า คือ x และคืนตัวแปรออก คือ y โดยที่ x และ y เป็นตัวแปรแบบจำนวนเต็มในภาษาซี ซึ่งเป็นรูปแบบ 2's complement ขนาด 16 บิต ตัวแปรโกลบอลที่ฟังก์ชั่นนี้ต้องใช้ (ต้องมีการประกาศค่าในโปรแกรมหลัก) ได้แก่

ก) Order คือ ค่าอันดับของตัวกรอง

ข) a[ ] และ b[ ] คือ ค่าสัมประสิทธิ์ของตัวกรอง

ค) wold[ ] คือ ตัวแปรสำหรับเก็บค่าสัญญาณในอดีตเพื่อใช้ในการประมวลผล

ฟังก์ชั่นนี้จะทำการบวกผลลัพธ์จากการคูณด้วยตัวบวกขนาด 32 บิต ทั้งนี้ ได้ตั้งตัวแปร ชื่อ ACC ซึ่งเป็น 2's complement ขนาด 32 บิต สำหรับสะสมค่าจากการบวก การปัดเศษให้กลับเป็น 16 บิต จะทำหลังจากการบวกครบทุกเทอมแล้ว โดยการปัดเศษทำได้โดยการเลื่อนบิตไปทางขวา 15 บิต (right shift)

การเลื่อนบิตนี้เป็นการตัดเศษทิ้งเลย (ไม่มีการปัดขึ้น) ผลก็คือ จะทำให้สัญญาณรบกวนจากการคูณมากขึ้น แต่ข้อดีก็คือ จะทำให้เขียนโปรแกรมได้ง่าย และทำงานได้เร็วขึ้น ดังนั้น ในที่นี้จะขออนุโลมในการใช้วิธีนี้ก่อน การตัดเศษทิ้งนี้ใน Matlab ใช้คำสั่งว่า floor ดังนั้น ถ้าต้องการเปรียบเทียบกับผลที่ได้กับ Matlab ก็ทำได้โดยเปลี่ยนการเรียกใช้ฟังก์ชั่น round ในโปรแกรม rndfix.m ให้มาใช้เป็นฟังก์ชั่น floor แทน

โปรแกรมที่ 10.12 เป็นตัวอย่างการเรียกใช้ฟังก์ชั่น iir2a ในเทอร์โบซี (ของบริษัท บอร์แลนด์) ซึ่งได้ใส่สัญญาณขาเข้าเป็นตัวแปรแบบอะเรย์ขนาด 10 ค่า และแสดงค่าของสัญญาณขาออกที่คำนวณได้ที่หน้าจอ โปรแกรมนี้ใช้ค่าสัมประสิทธิ์ที่คำนวณจาก Matlab เป็นตัวกรองบัตเตอร์เวิร์ธ LPF ที่มีความถี่ตัดที่ 0.6π และใช้ M=15, N=1 ในการแปลงเป็นเลขจำนวนเต็ม ซึ่งโปรแกรมนี้ได้ทดสอบแล้วว่าให้ผลตรงกับการจำลองด้วยโปรแกรม iira2.m ใน Matlab ทุกประการ ในการใช้งานจริง เราก็เพียงเปลี่ยนให้ x ไปรับค่าจากสัญญาณขาเข้าจริง ๆ ซึ่งอาจมาจากไฟล์ หรือจากพอร์ตอินพุตเอาต์พุตของคอมพิวเตอร์ก็ได้

```
int iir2a(int x)
{
   long int ACC = 0;
   int i,y;

   for (i=1; i<=Order; i++)
      ACC -= b[i] * wold[i];
   wold[0] = x + (ACC >> 15);
   ACC = 0;
   for (i=0; i<=Order; i++)
      ACC += a[i] * wold[i];
   y = ACC >> 15;

   for (i=Order; i>0; i--)
      wold[i] = wold[i-1];

   return(y);
}
```

โปรแกรมที่ 10.11 *ฟังก์ชั่น iir2a.c สำหรับคำนวณตัวกรอง IIR โครงสร้าง direct form 2*

โปรแกรมที่ 10.13 เป็นตัวอย่างของโปรแกรมภาษาซีที่เรียกใช้ฟังก์ชั่น iir2a เพื่อทำการประมวลผลแบบเวลาจริงบนบอร์ดทดลอง TMS320 DSK ของบริษัท เทกซัส อินสตรูเมนส์ ในโปรแกรมนี้ การประมวลผลจะกระทำในโปรแกรมย่อยตอบสนองอินเทอร์รัพต์ ทุกครั้งที่มีสัญญาณขาเข้าใหม่เข้ามาที่ A/D ก็จะมีสัญญาณอินเทอร์รัพต์เข้าชิพ DSP ดังนั้น การประมวลผลจะเข้าจังหวะกับค่าของสัญญาณขาเข้าที่เข้ามา ด้วยการประมวลผลแบบเวลาจริงนี้ ทำให้มันทำหน้าที่เสมือนเป็นตัวกรองแอนะลอกตัวหนึ่ง

```
#include "stdio.h"
#define  Order  4
const long int a[Order+1] = { 5478,  21913, 32767, 21913, 5478 };
const long int b[Order+1] = { 32767, 25628, 22282, 5986,  987  };
long int wold[Order+1] = { 0, 0, 0, 0, 0 };

void main()
{
   int i;                                              % x ในที่นี้เป็นสัญญาณทดสอบ
   int x[10] = { 5500,150,-2800,-20000,-25000,         ซึ่งเก็บอยู่เป็นแบบตัวแปรอะเรย์
                 -10000,2000,18000,21000,10000 };

   for (i=0; i<10; i++)                                % วนประมวลผลไปจนหมดทุกค่าของ x
      printf("  %d\n", iir2a(x[i]) );                    และแสดงผลลัพธ์ออกทางหน้าจอ

   getch();
}
```

โปรแกรมที่ 10.12 *การเรียกใช้ฟังก์ชั่น iir2a.c โดยใช้เทอร์โบซี*

```
#define  Order  4
extern void aicinit();
int *DRR = (int *) 0x20;
int *DXR = (int *) 0x21;
const int a[Order+1] = { 5478,  21913, 32767, 21913, 5478 };
const int b[Order+1] = { 32767, 25628, 22282, 5986,  987  };
int wold[Order+1] = { 0, 0, 0, 0, 0 };

main()
{
  aicinit();          %เรียกโปรแกรมย่อยสำหรับตั้งค่าควบคุมให้กับไอซี A/D และ D/A

  for(;;);            %วนรออินเทอร์รัพต์จาก A/D
}

void c_int5(void)            %โปรแกรมย่อยเพื่อตอบสนองอินเทอร์รัพต์จาก A/D
{
   int x;
   x = *DRR >> 2;            %รับค่าจากรีจิสเตอร์ที่ติดต่อกับ A/D

   *DXR = iir2a(x) << 2;     %ประมวลผล และส่งค่าออกทางรีจิสเตอร์ที่ติดต่อกับ D/A
}
```

โปรแกรมที่ 10.13 *การเรียกใช้ฟังก์ชั่น iir2a.c สำหรับประมวลผลแบบเวลาจริงด้วย TMS320C50*

บทที่ 11

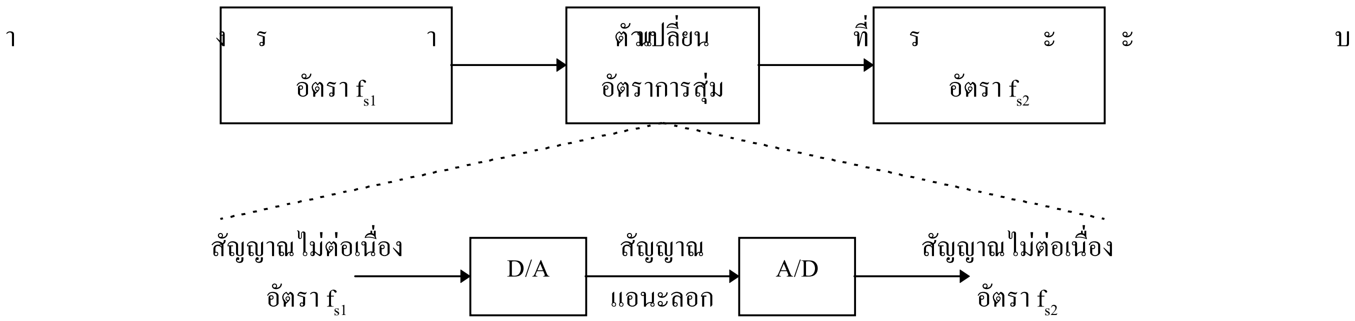
การประมวลผลแบบหลายอัตราสุ่ม

ในบทนี้เราจะได้ศึกษาถึงระบบที่มีอัตราการสุ่ม (f_s) หรืออัตราของข้อมูลมากกว่าหนึ่งอัตราในระบบ ซึ่งสิ่งที่จำเป็นสำหรับระบบประเภทนี้ก็คือ การเปลี่ยนอัตราการสุ่มได้ เราจะได้ศึกษาวิธีการในการลด และเพิ่มอัตราการสุ่มที่ถูกต้อง จากนั้น ในท้ายบทก็จะเป็นการยกตัวอย่างการใช้งานของระบบที่ใช้หลายอัตราการสุ่ม

การเปลี่ยนอัตราการสุ่มโดยแปลงเป็นสัญญาณแอนะลอกก่อน

วิธีที่จะเปลี่ยนอัตราการสุ่มของสัญญาณอย่างง่าย ๆ ก็คือ การแปลงสัญญาณนั้นกลับเป็นสัญญาณแอนะลอกก่อน แล้วสุ่มสัญญาณแอนะลอกใหม่ด้วยอัตราการสุ่มใหม่ที่ต้องการ ดังแสดงในรูปที่ 11.1 วิธีนี้ให้อิสระในการเปลี่ยนอัตราการสุ่มพอสมควร โดยเราสามารถใช้อัตราการสุ่มใหม่เป็นเท่าไรก็ได้ ทั้งนี้โดยไม่ขัดแย้งกับหลักการของการสุ่มที่ถูกต้อง นั่นก็คือ อัตราการสุ่มใหม่จะต้องมากกว่าสองเท่าของความถี่สูงสุดที่มีอยู่ในสัญญาณ

กรณีที่เป็นการเพิ่มอัตราการสุ่ม จะไม่มีปัญหาเกี่ยวกับ aliasing เนื่องจาก อัตราการสุ่มใหม่มากกว่าอัตราเก่า ซึ่งหมายถึงว่า ย่านไนควิทซ์กว้างขึ้น ดังนั้น ถ้าสัญญาณไม่ต่อเนื่องเดิมไม่มี aliasing สัญญาณใหม่ก็จะไม่มีทางเกิด aliasing เช่นกัน แต่ในกรณีของการลดอัตราการสุ่ม ซึ่งย่านไนควิทซ์จะแคบลง จะต้องระวังไม่ให้ความถี่ของสัญญาณเดิมมากเกินกว่าครึ่งหนึ่งของอัตราการสุ่มใหม่ ถ้าหากมากกว่า เราจำเป็นจะต้องกรองความถี่ในช่วงที่มากกว่านั้นทิ้งไป โดยอาจใช้ตัวกรองดิจิตอล



รูปที่ 11.1 *การเปลี่ยนอัตราการสุ่มโดยการแปลงกลับเป็นสัญญาณแอนะลอกแล้วสุ่มใหม่*

กรองทิ้งก่อนแปลงเป็นแอนะลอก หรือใช้ตัวกรองอนาลอกกรองก่อนที่จะทำการสุ่มใหม่ก็ได้ เรื่องการป้องกัน aliasing จากการลดอัตราการสุ่มนี้ จะได้กล่าวถึงอีกครั้ง ในเรื่องการลดอัตราการสุ่มในโหมดสัญญาณไม่ต่อเนื่องล้วน ๆ ซึ่งมีหลักการป้องกันคล้ายกัน

วิธีเปลี่ยนอัตราการสุ่มโดยแปลงเป็นสัญญาณแอนะลอกก่อนเช่นนี้ เป็นวิธีที่ไม่นิยมทำ เนื่องจากมีข้อเสีย คือ

ก) เพิ่มสัญญาณรบกวนจากการแบ่งขั้นสัญญาณ (quantization noise) เข้ามาในระบบ ในจุดที่ทำการสุ่มใหม่

ข) เพิ่มความเพี้ยนจาก aliasing ถ้าใช้ตัวกรองแอนะลอกที่ D/A และ A/D ไม่คมพอ

ค) ไม่สะดวกที่จะมีวงจรแอนะลอกเพิ่มเข้ามาในกึ่งกลางของการประมวลผล

ดังนั้น เราต้องการตัวเปลี่ยนอัตราการสุ่ม ที่ทำงานได้โดยไม่ต้องแปลงสัญญาณกลับเป็นแอนะลอก ซึ่งก็เป็นไปได้ ดังจะได้กล่าวในหัวข้อถัดไป

เดซซิเมเตอร์ (Decimator)

การลดอัตราการสุ่มในโหมดสัญญาณไม่ต่อเนื่องล้วน ๆ มองดูแล้วเหมือนเป็นสิ่งง่าย ๆ กล่าวคือ ถ้าเราต้องการลดอัตราลงด้วยอัตราส่วน 1/D หรือ ลดอัตราลง D เท่า โดยที่ D เป็นจำนวนเต็ม ก็สามารถทำได้โดยสร้างสัญญาณใหม่ขึ้นมาโดยดึงเอาสัญญาณเก่ามา 1 ค่า แล้วเว้น D-1 ค่า ตัวอย่างเช่น ถ้า D=3 เราจะต้องสร้างสัญญาณใหม่ y(n) โดยดึงเอาค่าจากสัญญาณเก่า x(n) 1 ค่า แล้วเว้น 2 ค่า ดังแสดงในรูปที่ 11.2 เราจะได้<u>อัตราข้อมูล</u>ของ y(n) น้อยลง (หรือช้าลง) กว่า x(n) 3 เท่า หรือถ้าคิดในแง่ปริมาณของข้อมูลแล้ว เราก็จะได้ว่า<u>จำนวนข้อมูล</u>ของ y(n) น้อยลงกว่า x(n) 3 เท่า

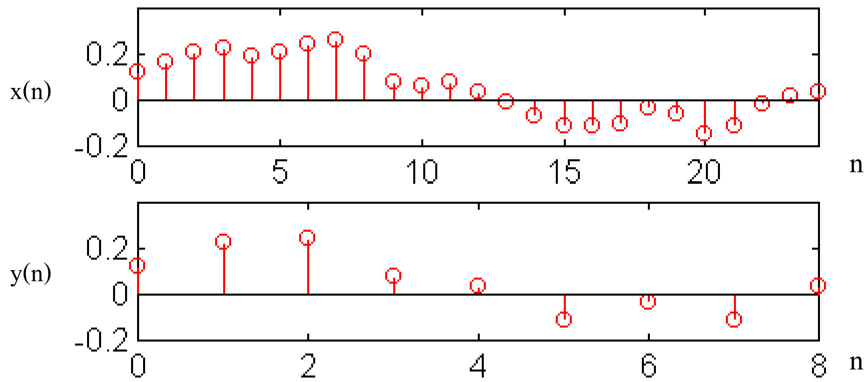
ในโดเมนความถี่ การลดอัตราการสุ่มลง 3 เท่า จะทำให้ย่านไนควิทซ์แคบลงเป็น 1/3 ของเดิม นั่นคือ อาจคิดได้ว่าสัญญาณใหม่จะมีความถี่ π ย้ายมาอยู่ที่ความถี่ $\pi/3$ ของเดิม ดังแสดงในรูปที่ 11.3 ปัญหาจะเกิดขึ้นถ้าหากว่าสัญญาณเดิมมีองค์ประกอบความถี่ที่สูงกว่า $\pi/3$ อยู่ด้วย เพราะเมื่อลดอัตราการสุ่มแล้ว จะทำให้เกิดการซ้อนทับของสเปกตรัมขึ้น หรือเกิด aliasing นั่นเอง ซึ่งเป็นหลักการเดียวกับที่ได้ศึกษามาแล้วในบทที่ 2 ดังนั้น ถ้าสัญญาณองค์ประกอบความถี่ที่สูงกว่า $\pi/3$ อยู่ด้วย การลดอัตราการสุ่มที่ถูกต้อง จะต้องกรองสัญญาณความถี่สูงนี้ทิ้งไป โดยใช้ตัวกรองดิจิตอลผ่านต่ำที่มีความถี่ตัดที่ $\pi/3$

ขอให้สังเกตว่า ในรูปที่ 11.2 และ11.3 พยายามวาดเพื่อแสดงให้เห็นความสัมพันธ์ระหว่างสัญญาณก่อน และหลังการลดอัตราการสุ่ม ในเชิงเวลา และความถี่ ดังนั้น จึงวาดโดยใช้สเกลในแกนนอนที่มีไม่เท่ากัน

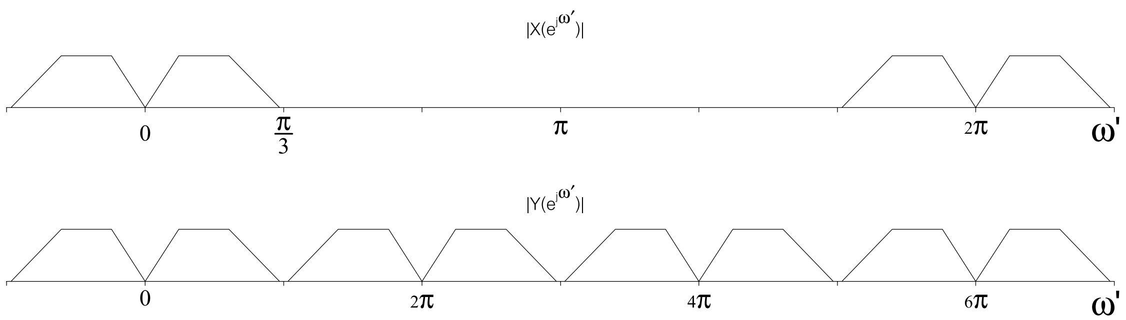
โดยสรุป กระบวนการลดอัตราการสุ่มลง D เท่าโดยไม่ทำให้สัญญาณใหม่เกิด aliasing ขึ้น ซึ่งเราจะเรียกว่า "เดซซิเมเตอร์" จะประกอบด้วย (แสดงในรูปที่ 11.4)

1. ตัวกรองดิจิตอลแบบ LPF อยู่ก่อนหน้าการลดอัตราการสุ่ม โดยตัวกรองดิจิตอลที่ใช้มีความถี่ตัดที่ π/D ดังแสดงในรูปที่ 11.4 ตัวกรองนี้บางทีก็เรียกว่า ตัวกรองเดคซิเมชัน (Decimation Filter) ซึ่งนิยมใช้เป็นตัวกรองแบบ FIR

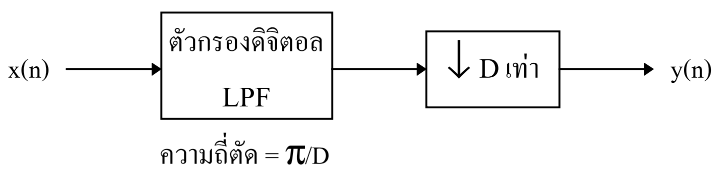
2. ตัวลดอัตราการสุ่ม (down-sampler) ขอนิยามสัญลักษณ์ของตัวนี้เป็น $\downarrow$ D ซึ่งตัวลดอัตราการสุ่มทำหน้าที่ดึงเอาสัญญาณขาเข้ามา 1 ค่า เว้น D-1 ค่า สลับกันไป ตามที่ได้กล่าวมาแล้ว



รูปที่ 11.2 *สัญญาณทางเวลาก่อน และหลังการลดอัตราการสุ่มลง 3 เท่า*



รูปที่ 11.3 *สเปกตรัมของสัญญาณก่อน และหลังการลดอัตราการสุ่มลง 3 เท่า (กรณีที่ไม่เกิด aliasing)*



รูปที่ 11.4 *ส่วนประกอบของเดซซิเมเตอร์*

โปรแกรมที่ 11.1 ทำหน้าที่เป็นตัวลดอัตราการสุ่มลง D เท่า โปรแกรมนี้เมื่อใช้ร่วมกับตัวกรองดิจิตอล ก็จะทำหน้าที่เป็นตัวเดซซิเมเตอร์ที่ต้องการได้ ใน Matlab DSP Toolbox มีฟังก์ชั่นชื่อ decimate ซึ่งได้รวมหน้าที่ของโปรแกรมนี้ กับตัวกรองดิจิตอลเข้าไว้ด้วยกัน แต่เพื่อให้เกิดความยืด

หยุ่นในการใช้งาน (เพราะในบางครั้งเราไม่ต้องการใช้ร่วมกับตัวกรองดิจิตอลแบบ LPF) จึงได้แยกเขียนเป็นฟังก์ชั่น downsam.m ต่างหาก

```
function y = downsam(x,D);
n=1:D:length(x);
y=x(n);
```

โปรแกรมที่ 11.1 *ฟังก์ชั่น downsam..m สำหรับลดอัตราการสุ่มของสัญญาณลง D เท่า*

อินเตอร์โพเลเตอร์ (Interpolator)

เช่นเดียวกันกับการลดอัตราการสุ่ม คือ เราสามารถเพิ่มอัตราการสุ่มของสัญญาณขึ้นได้ I เท่า โดยที่ I เป็นจำนวนเต็ม สมมติว่า ใช้ I=3 หมายถึงว่า สัญญาณใหม่ y(n) จะมีอัตราข้อมูลเพิ่มขึ้นจากสัญญาณเก่า x(n) 3 เท่า หรือถ้าคิดในแง่ของการประมวลผลแบบไม่เป็นเวลาจริง ก็จะได้ว่า y(n) จะมีจำนวนข้อมูลมากขึ้น 3 เท่า

สำหรับการเพิ่มอัตราการสุ่มกรณี I=3 เราจะได้ย่านในควิทซ์เพิ่มขึ้นจากเดิมสามเท่า ดังนั้น aliasing จะไม่เป็นปัญหาที่ต้องคำนึงถึงเหมือนในกรณีของการลดอัตราการสุ่ม แต่ปัญหาใหม่ก็คือ การสร้างสัญญาณ y(n) ขึ้นมานั้น จะต้องนำสัญญาณ x(n) มา แล้วแทรกข้อมูลใหม่เพิ่มลงไป 2 จุดในระหว่างทุก ๆ จุดของ x(n) เพื่อให้มีอัตราข้อมูลเพิ่มขึ้นสามเท่า ปัญหาคือ เราจะหาข้อมูลใหม่เหล่านี้มาได้อย่างไร โดยเงื่อนไขของข้อมูลใหม่ที่จะเติมลงไป คือ มันต้องเป็นตัวเฉลี่ยที่ถูกต้องระหว่างข้อมูลที่มันเข้าไปแทรก โดยไม่ทำให้สเปกตรัมเดิมของสัญญาณผิดเพี้ยนไป

ปรากฏว่า ถ้าเราลองทำการแทรกศูนย์ลงไป 2 จุดในระหว่างค่าของสัญญาณเก่าทุก ๆ จุด เรียกสัญญาณใหม่นี้ว่า w(n) ดังแสดงในรูปที่ 11.5 จะได้ว่า อัตราข้อมูลของ w(n) มากขึ้นเป็นสามเท่าจริง แต่เห็นได้ชัดว่า ศูนย์ที่แทรกเข้าไปไม่ใช่ค่าเฉลี่ยที่ดีแน่นอน เพราะทำให้รูปร่างสัญญาณผิดเพี้ยนไปเลย แต่ลองมาพิจารณาดูก่อนว่า ในภาคความถี่เกิดอะไรขึ้นกับสัญญาณ w(n) ซึ่งขอเริ่มต้นที่การพิจารณาการแปลง z ของ w(n) ดังนี้

$$W(z) = \sum_{n=0}^{\infty} w(n)z^{-n}$$

ถ้าสัญญาณ w(n) เริ่มต้นที่ w(0) = x(0) จะได้ว่าค่า w(n) ที่ไม่เท่ากับศูนย์ คือ w(3), w(6), w(9), … โดยที่ w(3) = x(1), w(6) = x(2), … เป็นเช่นนี้ไปเรื่อย ๆ หรือเขียนเป็นสูตรได้ว่า w(n) = x(n/3) จะได้

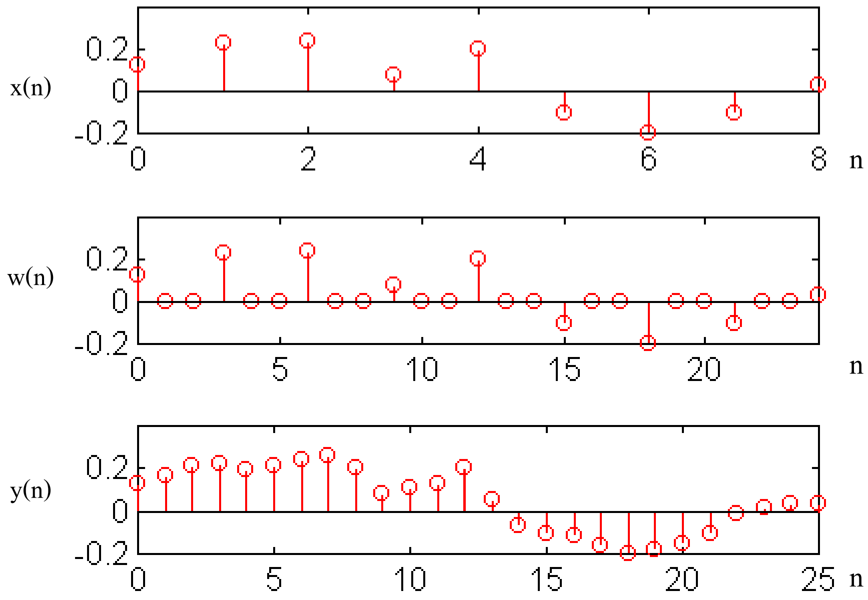
$$\begin{aligned} W(z) &= \sum_{n=0,3,6,9,...}^{\infty} w(n)z^{-n} \\ &= \sum_{n=0,3,6,9,...}^{\infty} x(n/3)z^{-n} \end{aligned}$$

แทน n ด้วย 3n จะได้ตัวชี้ที่เรียงลำดับเป็นปกติ คือ n=0,1,2,3, … ดังนี้

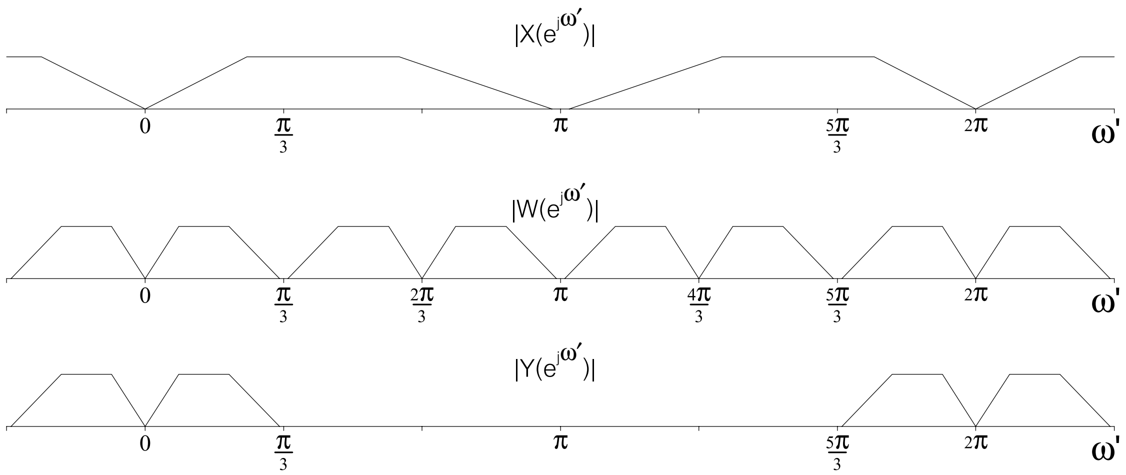
$$W(z) = \sum_{n=0}^{\infty} x(n) z^{-3n}$$
$$= X(z^3)$$

แทน $z=e^{j\omega'}$ จะได้ความสัมพันธ์ระหว่างสเปกตรัมของ x(n) กับ w(n) ดังนี้

$$W(e^{j\omega'}) = X(e^{j3\omega'}) \tag{11.1}$$



รูปที่ 11.5 สัญญาณทางเวลาก่อน และหลังการเพิ่มอัตราการสุ่มขึ้น 3 เท่า



รูปที่ 11.6 สเปกตรัมของสัญญาณก่อน และหลังการเพิ่มอัตราการสุ่มขึ้น 3 เท่า

จะได้ว่า สเปกตรัมของ w(n) คือ สเปกตรัมของ x(n) ที่ถูกบีบให้แคบลง 3 เท่า ดังแสดงในรูปที่ 11.6 ย่านความถี่ในช่วง 0 ถึง π เดิมของ x(n) จะถูกบีบให้อยู่ในช่วงความถี่ 0 ถึง $\pi/3$ ของ w(n) ซึ่งช่วงความถี่นี้ คือสิ่งที่เราต้องการ ส่วนความถี่ในช่วง $\pi/3$ ถึง π จะมีสำเนาของความถี่เดิมของ x(n) ปนเข้ามา ซึ่งคือสิ่งที่เราไม่ต้องการ และต้องตัดทิ้งโดยใชตัวกรองดิจิตอล

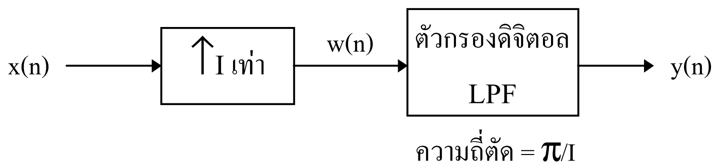
ฉะนั้น ตัวเพิ่มอัตราการสุ่มที่ถูกต้อง ซึ่งจะเรียกว่า อินเตอร์โพเลเตอร์ (แสดงในรูปที่ 11.7) จะต้องประกอบด้วย

1.) ตัวเพิ่มอัตราข้อมูลขึ้น I เท่า (up-sampler) ซึ่งจะขอแทนด้วยสัญลักษณ์ $\uparrow$I ตัวเพิ่มอัตราข้อมูลนี้ทำหน้าที่แทรกศูนย์ I-1 ตัวลงไปในระหว่างข้อมูลของสัญญาณขาเข้า ดังได้กล่าวมาแล้ว

2.) เราจะใช้ตัวกรองดิจิตอลแบบ LPF ที่มีความถี่ตัดที่ $\pi/3$ เพื่อกรองสำเนาความถี่ของ x(n) ที่เกิดขึ้นจากการแทรกศูนย์ทิ้งไป ตัวกรองนี้มีชื่อเรียกว่า ตัวกรองอินเตอร์โพเลชัน (Interpolation Filter)

สัญญาณขาออกของอินเตอร์โพเลเตอร์ คือ y(n) เป็นสัญญาณที่เทียบเท่ากับสัญญาณ x(n) และมีอัตราการสุ่มเพิ่มขึ้นสามเท่า ขอให้ดูการเปรียบเทียบรูปในทางเวลา และความถี่ของ y(n) กับ w(n) และ x(n) ในรูปที่ 11.5 และ 11.6

โปรแกรมที่ 11.2 ใช้สำหรับแทรกศูนย์เพื่อเพิ่มอัตราการสุ่มขึ้น I เท่า โปรแกรมนี้เมื่อใช้ร่วมกับตัวกรองดิจิตอล ก็จะทำหน้าที่เป็นตัวอินเตอร์โพเลเตอร์ที่ต้องการได้ ใน Matlab DSP Toolbox มีฟังก์ชั่นชื่อ interp ซึ่งได้รวมหน้าที่ของโปรแกรมนี้ กับตัวกรองดิจิตอลเข้าไว้ด้วยกัน



รูปที่ 11.7 *ส่วนประกอบของอินเตอร์โพเลเตอร์*

```
function y = upsam(x,I);
Ny = length(x)*I;
y=zeros(1,Ny);
n=1:I:Ny;
y(n)=x;
```

โปรแกรมที่ 11.2 *ฟังก์ชั่น upsam.m สำหรับเพิ่มอัตราการสุ่มของสัญญาณขึ้น I เท่า (โดยแทรกศูนย์)*

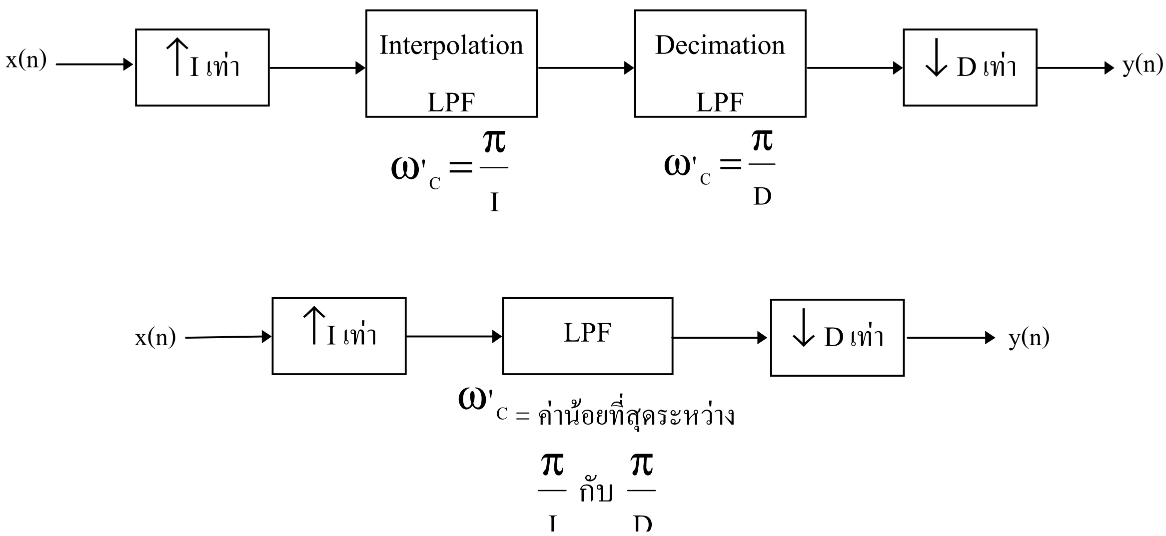
การเปลี่ยนอัตราการสุ่มด้วยอัตราส่วนที่ไม่เป็นจำนวนเต็ม

ที่ผ่านมาเราได้เห็นแล้วว่า การเปลี่ยนอัตราการสุ่มโดยไม่ต้องแปลงสัญญาณกลับเป็นแอนะลอกก่อนสามารถทำได้ แต่ด้วยข้อจำกัดว่า สามารถเพิ่ม หรือลดอัตราการสุ่มด้วยจำนวนเท่า ที่เป็นจำนวนเต็มเท่านั้น ในงานบางอย่าง ต้องใช้การเปลี่ยนอัตราการสุ่มด้วยอัตราส่วนที่ไม่เป็นจำนวนเต็ม เช่น สัญญาณเสียงที่เก็บในแผ่น CD ใช้อัตราข้อมูลเท่ากับ 44.1 kHz ถ้าต้องการจัดเก็บในเทป DAT (Digital Audio Tape) ซึ่งใช้อัตราข้อมูลเท่ากับเท่ากับ 48 kHz จำเป็นจะต้องเปลี่ยนอัตราสุ่มให้มากขึ้นเท่ากับ 48/44.1 = 1.088.. เท่า ซึ่งเป็นอัตราส่วนที่ไม่เป็นจำนวนเต็ม

เราสามารถเปลี่ยนอัตราสุ่มด้วยค่าที่ไม่เป็นจำนวนเต็มนี้ได้ โดยใช้เทคนิคแปลงให้ค่า 48/44.1 เป็นเศษส่วนของเลขจำนวนเต็ม ซึ่งในที่นี้จะได้

$$\frac{48}{44.1} \xrightarrow{\text{คูณ 10 ทั้งเศษและส่วน}} \frac{480}{441} \xrightarrow{\text{ตัดเศษ และส่วนให้เป็นจำนวนเต็มที่ต่ำที่สุด}} \frac{160}{147}$$

ดังนั้น หมายถึงว่า ถ้าเพิ่มอัตราการสุ่มขึ้นไปก่อน 160 เท่า จากนั้นลดอัตราสุ่มลงมา 147 เท่า เราจะได้ค่าของอัตราส่วนที่เปลี่ยนไปโดยรวมเท่ากับ 48/44.1 ตามที่ต้องการ ดังแสดงในแผนภาพทั่วไปในรูปที่ 11.8 จะเห็นได้ว่าตัวกรองสำหรับเพิ่มอัตราการสุ่ม และลดอัตราการสุ่มจะต่ออนุกรมกันพอดี ดังนั้น เราสามารถรวมตัวกรองทั้งสองเป็นตัวเดียวกันได้ดังแสดงในรูป และใช้ความถี่ตัดที่ต่ำที่สุดระหว่างตัวกรองทั้งสอง ในที่นี้ I=160 และ D=147 ดังนั้น ใช้ $\pi/160$ ซึ่งเป็นค่าที่ต่ำกว่าเป็นความถี่ตัดของตัวกรอง



รูปที่ 11.8 *การเปลี่ยนอัตราสุ่มที่มีอัตราส่วนไม่เป็นจำนวนเต็ม อัตราส่วนที่เปลี่ยน=I/D*

วิธีเปลี่ยนอัตราการสุ่มแบบนี้ มองดูค่อนข้างเป็นวิธีที่ขวานผ่าซาก แต่ก็ทำงานได้ดี ข้อเสียก็คือ ถ้าอัตราส่วนที่ได้เป็นจำนวนเต็มที่มีค่ามาก เราจะต้องใช้การประมวลผลที่มาก เนื่องจาก ตัวกรองดิจิตอลที่ใช้จะต้องทำงานที่ I เท่าของอัตราการสุ่มของสัญญาณขาเข้า ยิ่ง I มากเท่าไรมันก็จำเป็นต้องทำงานเร็วขึ้นเท่านั้น ในที่นี้ I=160 และ f_s ขาเข้า = 44.1 kHz ดังนั้น ตัวกรองดิจิตอลนี้ต้องทำงานที่อัตราสุ่มของข้อมูลเท่ากับ 7056 kHz ซึ่งถือว่า ค่อนข้างมาก และไม่เหมาะกับระบบที่ต้องการต้นทุนที่ต่ำ ทางแก้ก็คือ อาจเปลี่ยนโครงสร้างของระบบแทนที่จะเพิ่มอัตราสุ่มทีเดียว 160 เท่า ก็ค่อย ๆ เพิ่มและค่อย ๆ ลดสลับกันไป ดังที่จะได้อธิบายในหัวข้อถัดไป และทางเลือกอีกทางก็คือ แปลงให้เป็นสัญญาณแอนะลอกแล้วสุ่มใหม่

การเปลี่ยนอัตราการสุ่มแบบหลายขั้นตอน

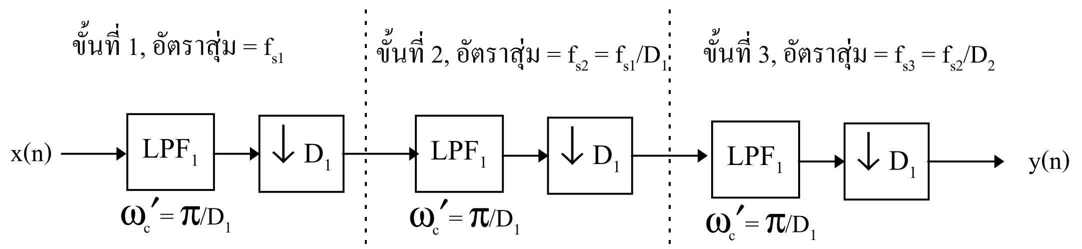
การเปลี่ยนอัตราสุ่มด้วยอัตราส่วนที่มีค่ามาก จะทำให้ต้องใช้ตัวกรองที่มีอันดับสูง และทำงานที่อัตราข้อมูลที่สูงด้วย ซึ่งก็จะทำให้มันต้องการการประมวลผลที่มาก โดยที่อัตราการประมวลผลของตัวกรอง FIR จะแปรผันตรงกับจำนวนสัมประสิทธิ์ของตัวกรอง (N) และอัตราการสุ่ม (f_s) ดังนี้

$$\text{อัตราการประมวลผล} = Nf_s \ \text{MAC / วินาที} \qquad (11.2)$$

อัตราการประมวลผลนี้สามารถลดได้ โดยการแบ่งการปลี่ยนอัตราสุ่มเป็นหลาย ๆ ขั้น แต่ละขั้นมีการเปลี่ยนทีละน้อย ๆ เช่น ในการลดอัตราการสุ่มลง D เท่า ถ้าเราสามารถแยกตัวประกอบของ D ได้เป็น

$$D = D_1 D_2 D_3 \dots D_M \qquad (11.3)$$

จะได้ว่า เราสามารถแบ่งการเปลี่ยนอัตราการสุ่มได้เป็น M ขั้น โดยขั้นที่หนึ่งมีการลดอัตราการสุ่มลง D_1 เท่า ขั้นที่สอง D_2 เท่า เช่นนี้ไปเรื่อย ๆ จนถึงขั้นที่ M แต่ละขั้นนั้นจะมีตัวกรองดิจิตอลเป็นส่วนประกอบด้วยหนึ่งตัว สมมติว่า ขั้นที่ i ใช้ตัวกรองที่มีจำนวนสัมประสิทธิ์เท่ากับ N_i และทำงานที่อัตราข้อมูล $f_{s,i}$ เราจะได้อัตราการประมวลผลโดยรวมเป็น



รูปที่ 11.9 *การลดอัตราการสุ่มแบบสามขั้น*

$$\text{อัตราการประมวลผลโดยรวม} = \sum_{i=1}^{M} N_i f_{s,i} \ \text{MAC / วินาที} \qquad (11.4)$$

โดยที่ $N_i f_{si}$ เป็นอัตราการประมวลผลของตัวกรองในขั้นที่ i จะเห็นได้ว่าที่ขั้นที่สูงขึ้น อัตราข้อมูลก็จะลดลง ดังแสดงในรูปที่ 11.9 ซึ่งอัตราข้อมูลนี้แปรผันโดยตรงกับอัตราการประมวลผล ดังนั้น จึงเป็นไปได้ว่าอัตราการประมวลผลในกรณีแตกเป็นขั้นย่อย ๆ นี้จะน้อยกว่ากรณีทำทีเดียว นอกจากนี้ ลำดับของการจัดเรียก D_i ก็มีผลต่ออัตราการประมวลผลเช่นกัน โดยพบว่า อัตราการประมวลผลจะน้อยที่สุด ถ้าจัดให้ $D_1 > D_2 > D_3 > \ldots > D_M$ ขอให้ศึกษาจากตัวอย่างที่ 11.1 ซึ่งจะแสดงถึงอัตราการประมวลผลที่น้อยลงเมื่อแตกเป็นขั้นย่อย ๆ และแสดงถึงวิธีการออกแบบตัวกรองที่จะใช้ในเดซซิเมเตอร์ สำหรับอินเตอร์โพเลเตอร์ก็มีหลักการแตกระบบเป็นขั้นย่อย ๆ และหลักการออกแบบตัวกรองที่ต้องใช้ในทำนองเดียวกัน

ตัวอย่างที่ 11.1 สัญญาณ x(n) มีอัตราการสุ่ม 96 kHz ต้องการลดอัตราการสุ่มลงเหลือ 1 kHz จงออกแบบตัวกรองดิจิตอลเดคซิเมชันที่ใช้โดยใช้วิธีหน้าต่างไคเซอร์ กำหนดให้ความพลิ้วของตัวกรองในช่วงแถบไม่เกิน 0.01 และแถบหยุดไม่เกิน 0.001 ($\delta_p \le 0.01$ และ $\delta_s \le 0.001$) สมมติให้สัญญาณที่ต้องการอยู่ในช่วงความถี่ 0 ถึง 450Hz

ให้หาอันดับของตัวกรองที่ต้องใช้, หน่วยความจำที่ต้องใช้เก็บสัมประสิทธิ์, และประมาณ MAC ที่ต้องใช้คำนวณ สำหรับกรณีลดอัตราสุ่มทีเดียว 96 เท่า และกรณีแยกเป็น 2 ขั้น

กรณีที่ 1 ลดทีเดียว 96 เท่า จาก f_{s1} = 96 kHz เป็น f_{s2} = 1 kHz (M=96)

จากทฤษฎี (รูปที่ 11.4) จะได้ว่า เราต้องการตัวกรอง LPF ที่มีความถี่ตัดที่ $\pi/96$ หรือตรงกับความถี่แอนะลอกที่ 500 Hz แต่เมื่อพิจารณาว่า อัตราการสุ่มสุดท้ายคือ 1 kHz ซึ่งมีช่วงไนควิทซ์คือ 0 ถึง 500 Hz และโจทย์กำหนดว่าย่านความถี่ที่ต้องการคือ 0 ถึง 450 Hz ดังนั้น ช่วงความถี่ 450 ถึง 500 Hz เป็นช่วงว่างที่สามารถใช้เป็นย่านแถบเปลี่ยนของตัวกรอง decimation ได้

จริง ๆ แล้วเราสามารถเลือกใช้ตัวกรองที่มีแถบเปลี่ยนในช่วง 450 ถึง 550 Hz ก็ยังได้ ถ้าไม่สนใจว่าช่วงความถี่ 450 ถึง 500 Hz ที่ขาออกจะมี aliasing หรือไม่ (สัญญาณช่วง 500 ถึง 550 Hz ที่หลุดผ่านตัวกรองมา จะกลายเป็น aliasing ที่ขาออกในช่วงความถี่ 450 ถึง 500 Hz) แต่ในที่นี้ขอยึดหลักอนุรักษ์นิยม โดยเลือกใช้แถบเปลี่ยนของตัวกรองเป็นช่วง 450 ถึง 500 Hz ซึ่งจะไม่มี aliasing ที่สัญญาณขาออก

จะได้ $\Delta f = 500 - 450 = 50$ Hz

$$\Delta f' = 50/f_{s1} = 50/96\text{k}$$

จากโจทย์ว่า $\delta_s \leq 0.001$ ซึ่งเป็นข้อกำหนดที่แรกกว่า δ_p

ดังนั้น $A = -20\log\delta_p = 60$ dB

จากสูตรการหาค่าอันดับของวิธีหน้าต่างไคเซอร์ จะได้

$$N = \frac{A - 7.95}{14.36\Delta f'} + 1 = 6960.3 \longrightarrow \text{เลือก } N = 6961$$

ดังนั้น ต้องใช้หน่วยความจำเพื่อเก็บค่าสัมประสิทธิ์ เท่ากับ 6961 ตำแหน่ง

จำนวน MAC ที่ต้องคำนวณ เท่ากับ $Nf_{s1} = (6961)(96k) \approx 668 \times 10^6$ MAC/วินาที

<u>กรณีที่ 2</u> แบ่งการลดอัตราสุ่มเป็นสองขั้น ขอเลือกใช้ว่า ขั้นที่ 1 ลดจาก f_{s1} = 96 kHz เป็น f_{s2} = 3 kHz (M_1=32) และขั้นที่สองจาก f_{s2} = 3 kHz เป็น f_{s3} = 1 kHz (M_2=3)

<u>ขั้นที่ 1</u> จาก f_{s1} = 96 kHz เป็น f_{s2} = 3 kHz (M_1=32)

ตามทฤษฎีเราต้องการความถี่ตัดที่ $\pi/32$ ซึ่งตรงกับความถี่แอนะลอกที่ 1500 Hz เช่นเดียวกันคือสามารถวิเคราะห์ได้ว่า สัญญาณที่ต้องการอยู่ในช่วง 0 ถึง 450 Hz เท่านั้น ดังนั้น ช่วงความถี่ 450 ถึง 1500 Hz เป็นช่วงที่สามารถใช้เป็นย่านความถี่ตัดของตัวกรองได้ เนื่องจาก สัญญาณขาออกของขั้นนี้เป็นสัญญาณกึ่งกลาง ซึ่งจริง ๆ แล้วย่านความถี่ 500 ถึง 1500 Hz จะไม่ผ่านไปจนถึงสัญญาณออกสุดท้าย (จะถูกกรองทิ้งไป ในขั้นการลดอัตราสุ่มในขั้นที่สอง) ดังนั้น เราสามารถยอมให้ย่านความถี่ 500 ถึง 1500 Hz นี้มี aliasing ได้โดยไม่เป็นปัญหาอะไร แถบเปลี่ยนของตัวกรองในขั้นที่หนึ่งที่เหมาะสมจึงควรเป็นช่วง 500 ถึง 2500 Hz (สัญญาณในช่วง 1500 ถึง 2500 Hz จะทำให้เกิด aliasing ในช่วง 500 ถึง 1500 Hz ที่ขาออก)

จะได้ $(\Delta f)_1 = 2500 - 500 = 2000$ Hz

$(\Delta f')_1 = 2000/f_{s1} = 2k/96k = 1/48$

A ยังคงเท่ากับ 60 dB เหมือนในกรณีแรก และเมื่อใช้สูตรการหาค่าอันดับของวิธีหน้าต่างไคเซอร์ จะได้

$$N_1 = \frac{A - 7.95}{14.36(\Delta f')_1} + 1 = 175.0 \longrightarrow \text{เลือก } N_1 = 175$$

<u>ขั้นที่ 2</u> จาก f_{s2} = 3 kHz เป็น f_{s3} = 1 kHz (M_2=3)

ตามทฤษฎีตัวกรองต้องการความถี่ตัดที่ $\pi/3$ ซึ่งตรงกับความถี่แอนะลอกที่ 500 Hz คราวนี้การวิเคราะห์จะเหมือนกับกรณีที่ 1 คือ ตัวกรองในขั้นนี้ควรมีความถี่ตัดในช่วง 450 ถึง 500 Hz

จะได้ $(\Delta f)_2 = 500 - 450 = 50$ Hz

$$(\Delta f')_2 = 50/f_{s2} = 50/3k$$

A ยังคงเท่ากับ 60 dB และเมื่อใช้สูตรการหาค่าอันดับของวิธีหน้าต่างไคเซอร์ จะได้

$$N_2 = \frac{A - 7.95}{14.36(\Delta f')_2} + 1 = 218.5 \longrightarrow \text{เลือก } N_2 = 219$$

ดังนั้น ต้องใช้หน่วยความจำเพื่อเก็บค่าสัมประสิทธิ์ทั้งหมด เท่ากับ $N_1 + N_2 = 394$ ตำแหน่ง

จำนวน MAC ที่ต้องคำนวณ เท่ากับ การประมวลผลในตัวกรองทั้งสองขั้น

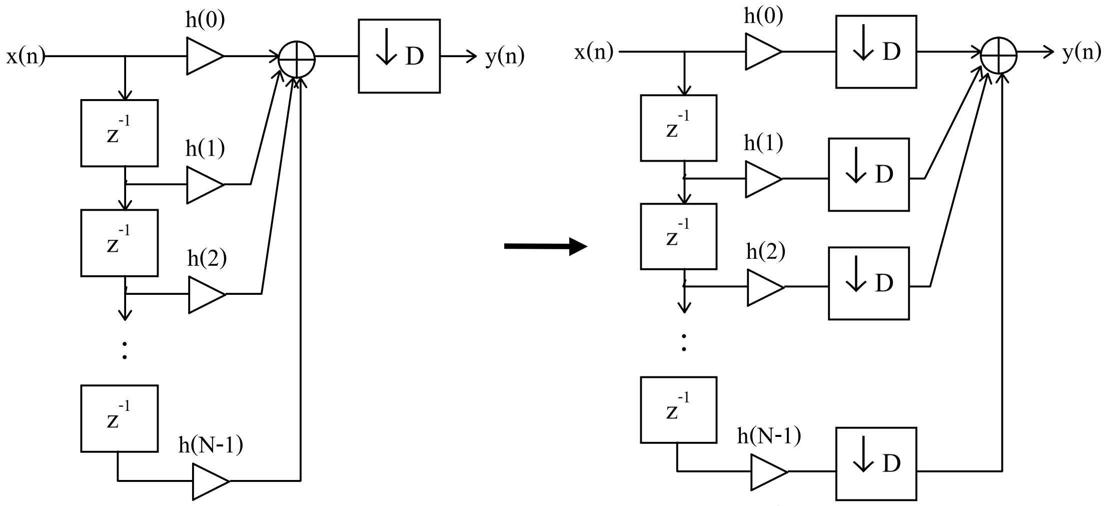
$$= N_1 f_{s1} + N_2 f_{s2} = (175)(96k) + (219)(3k) \approx 17.46 \times 10^6 \text{ MAC/วินาที}$$

เห็นได้ชัดว่า เมื่อแบ่งการประมวลผลเป็นสองขั้น เราสามารถลดอันดับโดยรวม และการประมวลผลโดยรวมลงไปได้มาก เราซึ่งอาจทำให้ดีขึ้นกว่านี้ได้อีก โดยแบ่งการประมวลผลให้มากขั้นกว่านี้ แต่การแบ่งมากเกินไปบางครั้งก็ไม่ทำให้ดีขึ้น หรืออาจแย่ลง (ดังแสดงใน [2]) ดังนั้น ในทางปฏิบัติจึงต้องคำนวณ และวิเคราะห์ให้ดีก่อนนำไปใช้ ขอให้ผู้อ่านลองนำตัวอย่างนี้ไปวิเคราะห์ดูต่อว่า การแบ่งเป็น 3 ขั้น, 4 ขั้น, หรือ 2 ขั้น แต่ใช้อัตราส่วนที่แตกต่างจากที่ได้ทำมา จะให้ผลดี หรือแย่ลงอย่างไร

การลดการประมวลผลของตัวเปลี่ยนอัตราสุ่ม

ถ้าลองสังเกตดูจากแผนภาพที่ผ่านมาจะพบว่า ตัวกรองดิจิตอลที่ใช้อยู่ในเดซซิเมเตอร์ และอินเตอร์โพเลเตอร์ ล้วนแล้วแต่ทำงานอยู่ในตำแหน่งที่มีอัตราข้อมูลสูงทั้งสิ้น ยกตัวอย่างเช่น เดซซิเมเตอร์จาก 96 kHz เป็น 3kHz ใช้ตัวกรองดิจิตอลทำงานอยู่ที่อัตรา 96 kHz และเช่นเดียวกัน อินเตอร์โพเลเตอร์จาก 3 kHz เป็น 96 kHz ก็ใช้ตัวกรองดิจิตอลทำงานที่อัตรา 96 kHz ถ้าเราสามารถทำให้ตัวกรองดิจิตอลทำงานที่ 3 kHz ได้ ก็จะทำให้อัตราการประมวลในตัวอย่างนี้ลดลงได้ถึง 32 เท่า ซึ่งจริง ๆ แล้วก็มีเทคนิคที่ทำได้ง่ายมากในการทำให้ตัวกรองดิจิตอลมาทำงานอยู่ที่อัตราข้อมูลต่ำได้

<u>สำหรับเดซซิเมเตอร์</u> ซึ่งมีแผนภาพที่ใช้โครงสร้างของตัวกรอง FIR ปกติดังแสดงในรูปที่ 11.10 ก) เราสามารถลดอัตราการประมวลผลได้ง่ายมาก โดยการดึงเอาตัวลดอัตราสุ่มรวมเข้าไปกับโครงสร้างของตัวกรอง FIR ดังแสดงในรูปที่ 11.10 ข) หรือ อีกนัยหนึ่งก็คือ การคำนวณสัญญาณขาออกเท่าที่จำเป็นสำหรับอัตราขาออกเท่านั้น ยกตัวอย่างเช่น การลดอัตราข้อมูลจาก 96 kHz เป็น 3 kHz หรือ ลดลง 32 เท่า ก็ทำได้โดยคำนวณขาออกของตัวกรอง FIR 1 ค่า แล้วก็ข้ามไป 31 ค่า ไปคำนวณค่าถัดไปเลย (แต่เก็บค่าสัญญาณขาเข้าไว้เหมือนปกติ)



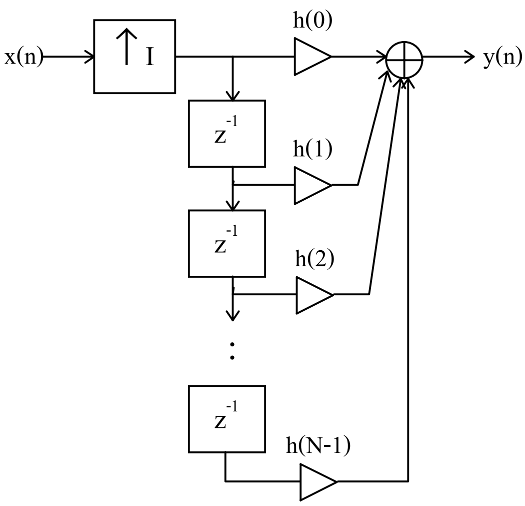
ก) โครงสร้างปกติ ข) โครงสร้างที่ลดอัตราการประมวลผลลง

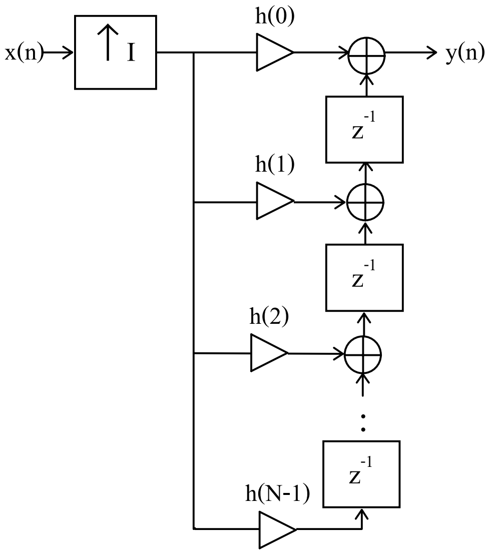
***รูปที่ 11.10** การลดอัตราการประมวลผลสำหรับเดซิเมเตอร์*

<u>สำหรับอินเตอร์โพเลเตอร์</u> ซึ่งมีแผนภาพที่ใช้โครงสร้างของตัวกรอง FIR ปกติดังแสดงในรูปที่ 11.11 ก) ใช้เทคนิคที่ซับซ้อนกว่าเดซิเมเตอร์ เนื่องจาก เราไม่สามารถย้ายตัวเพิ่มอัตราการสุ่ม (ตัวแทรกศูนย์นั่นเอง) ไปไว้หลังตัวคูณในตัวกรองได้ทันที จำเป็นที่จะต้องปรับโครงสร้างของตัวกรอง FIR ก่อน ซึ่งพบว่า โครงสร้างแบบทรานโพสด์ (transposed structure) สามารถทำให้บรรลุวัตถุประสงค์ได้

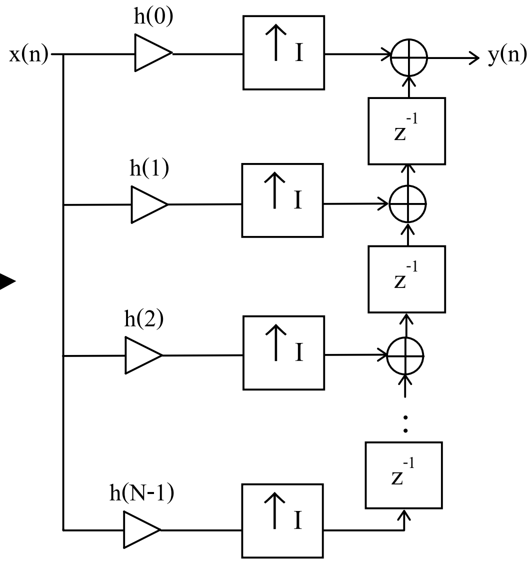
โครงสร้างแบบทรานโพสด์แสดงดังในรูปที่ 11.11 ข) ซึ่งสามารถพิสูจน์ได้ไม่ยากว่า ให้ผลตอบเหมือนกับโครงสร้างแบบปกติ โครงสร้างมีข้อเสียกว่าโครงสร้างปกติตรงที่ต้องใช้ตัวบวกสะสมหลายตัว ดังนั้น โดยปกติจึงไม่มีการนำมาใช้งาน แต่การใช้โครงสร้างทรานสโพสด์ในที่นี้ทำให้เราสามารถย้ายตัวเพิ่มอัตราการสุ่มไปไว้หลังตัวคูณได้ ดังแสดงในรูปที่ 11.11 ค) และทำให้ลดอัตราการประมวลผลลงได้ I เท่า เหมือนอย่างกรณีของเดซิเมเตอร์

รูปที่ 11.11 ก)
โครงสร้างปกติ





ข) ใช้ FIR เป็นโครงสร้างแบบทรานสโพสด์



ค) โครงสร้างที่ลดอัตราการประมวลผลลง

รูปที่ 11.11 *การลดอัตราการประมวลผลสำหรับอินเตอร์โพเลเตอร์*

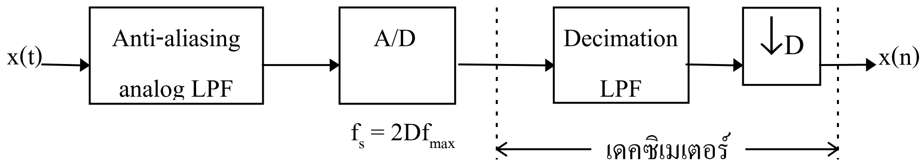
การประยุกต์ใช้งาน

การประยุกต์ใช้งานโดยตรงของตัวแปลงอัตราการสุ่ม ก็คือ การเชื่อมต่อระบบประมวลผลสองระบบที่มีอัตราสุ่มของข้อมูลไม่เท่ากัน ดังที่ได้ยกตัวอย่างไปในเรื่องการแปลงอัตราสุ่มของสัญญาณเสียง นอกจากการประยุกต์ใช้ในเรื่องนี้แล้ว การเปลี่ยนแปลงอัตราการสุ่มยังมีที่ใช้งานอีกอย่างกว้างขวาง ในที่นี้จะขอยกตัวอย่างที่สำคัญ 2 เรื่อง ก็คือ การใช้ในตัวแปลงระหว่างดิจิตอลกับแอนะลอก และการใช้ในการประมวลผลที่มีการแยกย่านความถี่ย่อย

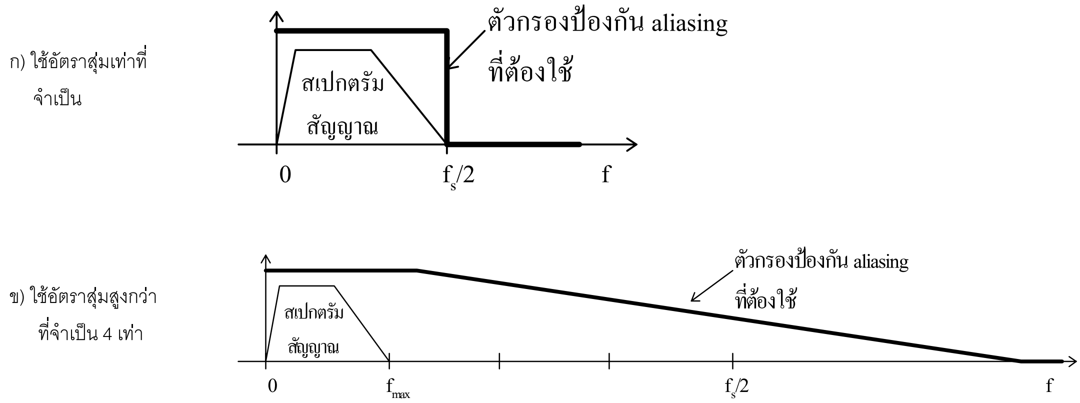
<u>ตัวแปลงแอนะลอกเป็นดิจิตอลแบบอัตราสุ่มสูง</u> (Oversampling A/D) [1],[11]

อุปกรณ์แปลงสัญญาณแอนะลอกเป็นดิจิตอลนอกจากจะมีความคลาดเคลื่อนจากการแบ่งขั้นสัญญาณตามที่ได้กล่าวไปแล้วในบทที่ 10 ยังมีความคลาดเคลื่อนของวงจรที่เกิดมาจากกระบวนการผลิตชิพ เพราะวงจรเหล่านี้เป็นวงจรแอนะลอกซึ่งต้องขึ้นกับค่าความต้านทาน และค่าการเก็บประจุที่ไม่สามารถทำให้แม่นยำ 100% ได้ ทำให้วงจรมีความคลาดเคลื่อนจากที่ตั้งไว้ และตัวสุดท้ายก็คือความคลาดเคลื่อนที่เกิดจาก aliasing ในกรณีที่เราใช้ตัวกรองป้องกัน aliasing ที่ไม่คมพอ

ในงานบางอย่างต้องการตัวแปลงแอนะลอกเป็นดิจิตอลที่มีคุณภาพสูง จึงต้องลดความคลาดเคลื่อนต่าง ๆ เหล่านี้ลง ความคลาดเคลื่อนจากการแบ่งขั้นสัญญาณลดลงได้ด้วยการเพิ่มจำนวนบิต ส่วนความคลาดเคลื่อนจากวงจร และ aliasing สามารถชดเชยได้ด้วยเทคนิคการใช้อัตราสุ่มที่สูงเกินจำเป็น (oversampling)



***รูปที่ 11.12** แผนภาพของตัวแปลงแอนะลอกเป็นดิจิตอลแบบอัตราสุ่มสูง*



***รูปที่ 11.13** ตัวกรองป้องกัน aliasing ที่ต้องใช้ในตัวแปลงแอนะลอกเป็นดิจิตอล*

จากที่ทราบมาแล้วว่าตัวแปลงสัญญาณแอนะลอกเป็นดิจิตอลโดยทั่วไปจะมี ตัวกรองป้องกัน aliasing ที่ขาเข้า ซึ่งมีความถี่ตัดที่ $f_s/2$ ถ้าหากว่าเราใช้อัตรา fs เท่าที่จำเป็น คือ ประมาณสองเท่าของความถี่สูงสุงในสัญญาณ (f_{max}) หรือมากกว่าเล็กน้อย จะได้ว่า จะต้องใช้ตัวกรองป้องกัน aliasing ที่มีความคมมากดังแสดงในรูปที่ 11.13 ก)

แผนภาพในรูปที่ 11.12 แสดงตัวแปลงสัญญาณแอนะลอกเป็นดิจิตอล ซึ่งสุ่มสัญญาณด้วยอัตราที่สูงกว่าจำเป็น D เท่า จากนั้นใช้ตัวเดซซิเมเตอร์ลดอัตราสุ่มลงมา D เท่า จะได้สัญญาณขาออกมีอัตราสุ่มเท่ากับในกรณีแรก และจะมีคุณภาพของสัญญาณขาออกดีขึ้น ด้วยเหตุผลคือ

1. เมื่ออัตราสุ่มสัญญาณสูงขึ้นกว่าที่จำเป็น เราจะสามารถใช้ตัวกรองป้องกัน aliasing ที่มีความคมต่ำได้ ดังแสดงในรูปที่ 11.13 ข) ซึ่งมีค่า D=4 ซึ่งตัวกรองนี้จะออกแบบง่าย และราคาถูกกว่ากรณีแรกมาก

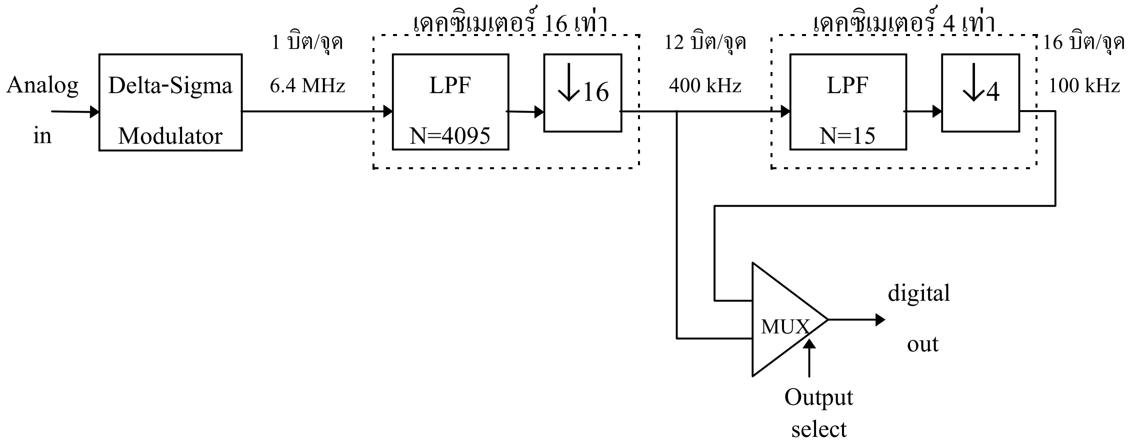
2. เมื่ออัตราสุ่มสัญญาณสูงขึ้นกว่าที่จำเป็น เราจะสามารถได้ SNR ที่ดีขึ้นได้ เนื่องจากสัญญาณรบกวนจากการแบ่งขั้นสัญญาณจะกระจายอยู่ในย่านความถี่ทั้งหมด ดังนั้น ถ้าพิจารณาเฉพาะย่านความถี่ที่เราสนใจ คือ 0 ถึง f_{max} ในกรณีนี้ จะมีกำลังของสัญญาณรบกวนน้อยกว่ากรณีแรก D เท่า

3. ตัวกรองดิจิตอลในเดซซิเมเตอร์ ทำหน้าที่สำคัญ คือ กรองสัญญาณที่สูงกว่าความถี่ที่สนใจทิ้ง ซึ่งอาจประกอบด้วย aliasing และสัญญาณขาเข้าที่สูงเกิน f_{max} นอกจากนี้ ยังทำหน้าที่เฉลี่ยสัญญาณให้ได้นัยที่สำคัญมากขึ้น (จำนวนบิตมากขึ้น) ด้วย ผลตอบของ FIR ถ้าไม่ปัดเศษทิ้ง จะมีจำนวนบิตมากเกินจากสัญญาณขาเข้าอยู่แล้ว (จากบทที่ 10) แต่โดยปกติเราจะปัดเศษทิ้งให้มีจำนวน

บิตเท่ากับสัญญาณขาเข้า ในที่นี้เนื่องจาก เรามั่นใจว่าสัญญาณขาเข้ามี SNR ดีกว่าปกติ ดังนั้น สามารถปัดเศษให้สัญญาณขาออกที่มีจำนวนบิตที่มากกว่าปกติได้ โดยสามารถพิสูจน์ได้ว่า การสุ่มด้วยอัตราสูงขึ้น 4 เท่าจะสามารถให้ SNR ที่ดีขึ้นเทียบเท่ากับการใช้จำนวนบิตมากขึ้น 1 บิต[11]

ถ้าเราใช้อัตราสุ่มของ A/D สูงมากขึ้น ๆ เราจะสามารถลดจำนวนบิตที่ต้องใช้ที่ A/D ให้น้อยลงได้ แล้วใช้เดซซิเมเตอร์เป็นตัวลดอัตราสุ่มลง พร้อมกับเพิ่มจำนวนบิตขึ้นไปด้วย ในกรณีสุดโต่งที่สุด ซึ่งปรากฏว่าเป็นกรณีที่ให้ผลดีที่สุด คือ การใช้ A/D ขนาด 1 บิต ที่อัตราสุ่มสูงมาก ๆ โดยใช้ A/D 1 บิต ที่สร้างโดยวิธีเดลต้า-ซิกม่ามอดูเลเตอร์ (Delta-Sigma Modulator) ซึ่งมีคุณสมบัติพิเศษ คือ จะทำให้สัญญาณรบกวนจากการแบ่งขั้นถูกเลื่อนขึ้นไปอยู่ในย่านความถี่สูงได้ (ทำให้ SNR ในย่านความถี่ของสัญญาณที่ต้องการดีขึ้นไปอีก) นอกจากนี้ การใช้ A/D 1 บิตยังมีข้อดี คือ สร้างได้ง่าย และความคลาดเคลื่อนของวงจรแทบมีผลต่อการทำงานน้อยมาก เพราะสัญญาณขาออกเป็นเพียงแค่หนึ่งบิตต่อหนึ่งจุด สำหรับตัวกรองป้องกัน aliasing ก็สามารถใช้เป็นวงจร RC อันดับหนึ่งง่าย ๆ ก็พอ

ตัวแปลง A/D ที่ใช้เดลต้า-ซิกม่ามอดูเลเตอร์ รวมกับ เดซซิเมเตอร์นี้รวมเรียกว่า ตัวแปลง A/D แบบเดลต้า-ซิกม่า (Delta-Sigma A/D Converter) ซึ่งปัจจุบันมีในท้องตลาดมากมายในรูปแบบของไอซีสำเร็จรูป รูปที่ 11.14 แสดงตัวอย่างของ A/D ชนิดนี้ของบริษัทโมโตโรลลา ซึ่งสุ่มสัญญาณด้วยความถี่สูงถึง 6.4 MHz และสามารถให้สัญญาณดิจิตอลขาออกได้สองแบบ คือ 12 บิตที่อัตรา 400 kHz หรือ 16 บิต ที่อัตรา 100 kHz



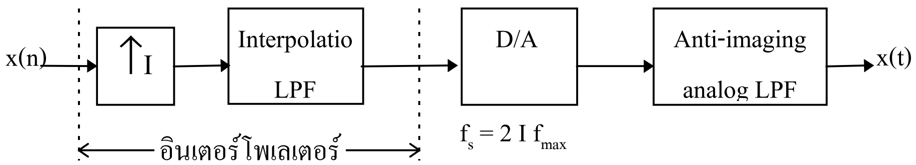
รูปที่ 11.14 *ตัวอย่างของ A/D แบบเดลต้า-ซิกม่า (Motorola DSP56ADC16)*

<u>ตัวแปลงดิจิตอลเป็นแอนะลอกแบบอัตราสุ่มสูง</u> (Oversampling D/A) [1],[11]

กระบวนการแปลงดิจิตอลเป็นแอนะลอกแบบใช้อัตราสุ่มสูง ก็เป็นกระบวนการย้อนกลับของการแปลงแอนะลอกเป็นดิจิตอล โดยมีหลักการในส่วนต่าง ๆ เหมือนกัน การใช้อัตราสุ่มสูงเกินกว่าที่จำเป็นใน D/A จะช่วงลดผลของความคลาดเคลื่อนของวงจร และความผิดเพี้ยนของสำเนาความถี่ของสัญญาณที่เกิดขึ้นอันเนื่องมาจากความไม่เป็นอุดมคติของตัวกรองป้องกันสำเนา (anti-imaging filter) หรือ ตัวกรองสร้างสัญญาณคืน (reconstruction filter)

รูปที่ 11.15 แสดงตัวแปลง D/A ที่ใช้อัตราสุ่มสูง ซึ่งประกอบด้วยอินเตอร์โพเลเตอร์เพื่อเพิ่มอัตราการสุ่มของสัญญาณให้สูงขึ้น พร้อม ๆ กับอาจจะลดจำนวนบิตลงด้วย จากนั้นใช้ตัวแปลง D/A แปลงเป็นสัญญาณแอนะลอกที่มีลักษณะเป็นขั้นบันได แล้วใช้ตัวกรองแอนะลอกกรองสำเนาความถี่ที่เหลืออยู่ออกไปเสีย ก็จะได้สัญญาณแอนะลอกที่สมบูรณ์

การใช้อินเตอร์โพเลเตอร์จะทำให้สำเนาความถี่แยกออกจากกัน (มีช่วงว่างระหว่างความถี่ที่กว้างออก) ทำให้สามารถใช้ D/A ที่มีจำนวนบิตต่ำลงได้ และใช้ตัวกรองแอนะลอกที่มีอันดับต่ำลงได้เช่นกัน จะเห็นได้ว่ามีเหตุผลที่คล้ายคลึงกับในกรณีของ A/D และเช่นเดียวกัน ในกรณีสุดโต่งจะเพิ่มอัตราสุ่มให้สูงมากจนสามารถใช้แค่ 1 บิตต่อจุด แล้วใช้วงจรเดลต้า-ซิกม่ามอดูเลเตอร์ทำหน้าที่เป็น D/A 1 บิต แปลงสัญญาณเป็นแอนะลอก กรณีนี้ก็จะเรียกว่า ตัวแปลง D/A แบบเดลต้า-ซิกม่า

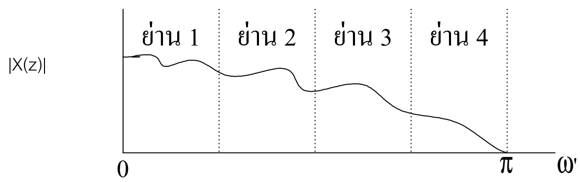


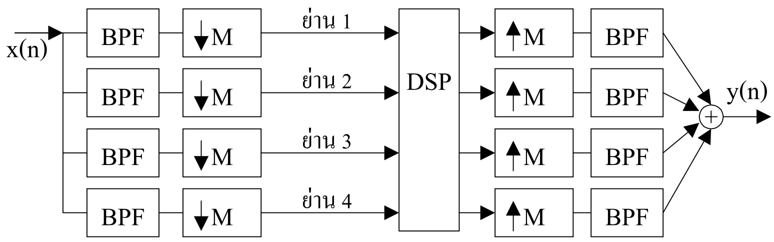
***รูปที่ 11.15** แผนภาพของตัวแปลงดิจิตอลเป็นแอนะลอกแบบอัตราสุ่มสูง*

<u>การประมวลผลโดยแยกย่านความถี่ย่อย</u>[3]

การประมวลผลสัญญาณในงานบางอย่าง สามารถให้ผลที่ดีขึ้นได้ โดยแยกสัญญาณที่จะประมวลผลเป็นสัญญาณในย่านความถี่ย่อย ๆ แล้วประมวลผลในแต่ละย่านความถี่แยกจากกัน (หรืออาจใช้ผลในแต่ละย่านความถี่มาประมวลผลร่วมกันทีหลังก็ได้) ดังแสดงในรูปที่ 11.6 เป็นการแยกสัญญาณออกเป็น 4 ย่านความถี่ ซึ่งหลังจากประมวลผลแล้วก็จะนำสัญญาณทั้งสี่มารวมกันเป็นสัญญาณขาออก

เครื่องมือที่ใช้สำหรับแยกย่านความถี่ ก็คือ ตัวกรองดิจิตอล 4 ตัวที่เป็นแบบผ่านย่านความถี่ตัวละย่าน บางทีเรียกตัวกรองเหล่านี้รวมว่า แผงตัวกรอง (filter bank) เมื่อสัญญาณถูกแยกออกเป็น 4 ย่านย่อยแล้ว เนื่องจากแต่ละย่านจะมีแถบความถี่เป็น 1/4 ของแถบความถี่เดิม ดังนั้น เราสามารถลดอัตราข้อมูลของแต่ละย่านลงให้เหลือเพียง 1/4 ของอัตราเดิมได้ด้วยโดยที่ไม่เกิด aliasing ซึ่งจะทำให้อัตราการประมวลผลที่ต้องการลดลงมาก





***รูปที่ 11.16** แผนภาพของการประมวลผลแบบแยกย่านความถี่ย่อย (4 ย่านความถี่)*

ตัวกรองที่เราใช้กรองย่านความถี่ทั้งสี่ออกมา จึงทำหน้าที่เป็นตัวกรองเดคซิเมชันไปด้วยในตัว ซึ่งสำหรับการลดอัตราการสุ่มสำหรับย่านที่ 1 จะทำได้เหมือนโดยตรง แต่การลดอัตราสุ่มสำหรับย่านอื่น ๆ ซึ่งไม่ได้มีความถี่เริ่มต้นที่ศูนย์ก็สามารถทำได้ โดยมอดูเลต (หรือคูณด้วยสัญญาณความถี่เดี่ยวค่าหนึ่ง) ให้สัญญาณทุกย่านมาอยู่เริ่มต้นที่ความถี่ศูนย์ก่อน

หลักจากประมวลผลเสร็จเรียบร้อยแล้ว สำหรับการประมวลผลที่ต้องการสัญญาณขาออกจะก็แปลงให้แต่ละย่านมีอัตราสุ่มเท่าเดิม โดยใช้อินเตอร์โพเลเตอร์ 4 ตัว ดังแสดงในรูปที่ 11.16 จากนั้นจึงมอดูเลตสัญญาณแต่ละย่านให้กลับสู่ย่านความถี่เดิม แล้วนำมาบวกกัน ก็จะได้สัญญาณขาออก

การประมวลผลแบบแบ่งเป็นย่านความถี่ย่อย ได้มีผู้ประยุกต์ใช้ในงานหลายอย่าง เช่น

- การเข้ารหัสเสียง (speech coding)
- การรู้จำเสียง (speech recognition)
- การวิเคราะห์สเปกตรัม
- ตัวกรองแบบปรับตัวได้ (adaptive filter)
- การแปลงเวฟเล็ต (wavelet transform)

บทที่ 12

ตัวอย่างการประยุกต์ใช้งาน

เราได้เห็นการประยุกต์ใช้งานไปบ้างแล้วในบทที่ผ่านมา ได้แก่ การทำตัวกรองแอนะลอกจากตัวกรองดิจิตอล การเปลี่ยนแปลงอัตราการสุ่ม และการทำเครื่องวิเคราะห์สเปกตรัมโดยใช้การแปลง FFT ในบทนี้จะได้กล่าวถึง การประยุกต์ใช้ตัวกรองดิจิตอลในงานต่าง ๆ รวมทั้ง แนะนำการประยุกต์ใช้งานในขั้นสูงขึ้น ได้แก่ ตัวกรองแบบปรับตัวได้ เพื่อให้ผู้อ่านได้มองเห็นภาพของการประมวลผลสัญญาณดิจิตอลที่กว้างขึ้น และเพื่อเป็นแนวทางในการศึกษาในขั้นสูงต่อไป

การปรับแต่งลักษณะของเสียง

เสียงเป็นสัญญาณที่เหมาะสำหรับนำมาประมวลผลอย่างยิ่ง เนื่องจาก ย่านความถี่ของเสียงจัดว่าอยู่ในย่างความถี่ที่ต่ำ กล่าวคือ เสียงคนพูดสามารถสุ่มได้คุณภาพดีด้วยอัตรา 8 kHz และเสียงดนตรีสามารถสุ่มได้ด้วยอัตรา 44 kHz ซึ่งเป็นอัตราที่ตัวแปลง A/D และ D/A มีราคาถูก และการประมวลผลก็สามารถกระทำได้โดยชิพ DSP ทั่วไป ในห้องสตูดิโอสมัยใหม่ ปัจจุบันจะทำการประมวลผลเสียงในโหมดดิจิตอลอย่างครบวงจร ทั้งการผสมเสียง การเติมเสียงพิเศษ และการปรับแต่งลักษณะของเสียง จนกระทั่งถึง การจัดเก็บในรูปของแผ่นซีดี และดีวีดี เพื่อส่งขายไปยังผู้บริโภค เสียงที่เก็บก็อยู่ในรูปของดิจิตอลแทบทั้งสิ้น

นอกจากนี้ เสียงยังเป็นสิ่งที่ละเอียดอ่อนมาก เพราะมนุษย์สามารถสัมผัสได้ด้วยการฟัง และแยกแยะลักษณะบางอย่างจากเสียงที่ได้ยินได้ เช่น เป็นเสียงทุ้มหรือแหลม, มีทิศทางของเสียงมาจากทิศใด, ใครคือผู้พูด, และคำพูดที่อยู่ในเสียงคือคำว่าอะไร เป็นต้น ซึ่งลักษณะเหล่านี้ ได้ถูกนำมาจำลอง และวิเคราะห์ด้วยการประมวลผลสัญญาณดิจิตอล ซึ่งความเข้าใจในลักษณะเหล่านี้ บางอย่างก็อยู่ในขั้นที่ค่อนข้างสมบูรณ์แล้ว แต่บางอย่างก็ยังอยู่ในขั้นที่ต้องวิจัยเพิ่มเติมต่อไป

<u>อีควอไลเซอร์เสียง</u> (Audio Equalizer)[1]

อีควอไลเซอร์เป็นอุปกรณ์สำหรับปรับลักษณะทางความถี่ของเสียง การทำอีควอไลเซอร์ด้วยการประมวลผลสัญญาณดิจิตอล เราจะใช้ตัวกรองดิจิตอลแบ่งเสียงออกเป็นย่านความถี่หลาย ๆ ย่าน และให้ผู้ใช้ปรับค่าอัตราขยายในแต่ละย่านได้ ขอยกตัวอย่างของอีควอไลเซอร์ที่มี 5 ย่านความถี่ โดยใช้อัตราสุ่มเท่ากับ 44.1 kHz และตัวกรองที่ใช้ในแต่ละย่านความถี่มีความถี่ตัด คือ

1) $H_1(z)$ ผ่านย่านความถี่ 0 - 3 kHz

2) $H_2(z)$ ผ่านย่านความถี่ 3 - 7 kHz

3) $H_3(z)$ ผ่านย่านความถี่ 7 - 11 kHz

4) $H_4(z)$ ผ่านย่านความถี่ 11 - 15 kHz

5) $H_5(z)$ ผ่านย่านความถี่ 15 - 22.05 kHz

ตัวกรองทั้ง 5 สามารถสร้างโดยใช้ตัวกรอง FIR หรือ IIR ก็ได้ โดยต้องออกแบบให้ผลรวมของฟังก์ชั่นถ่ายโอนทั้งหมดรวมแล้วเป็นตัวกรองแบบผ่านตลอด (all pass filter) ที่มีอัตราขยายคงที่ตลอดทุกความถี่ การออกแบบนี้ทำได้ง่ายมากโดยใช้ตัวกรองแบบ FIR โดยให้ความถี่ตัดของตัวกรองมีค่าเท่ากับจุดความถี่ปลายของความถี่ ซึ่งจะได้ผลตอบสนองเชิงความถี่ของแต่ละย่านความถี่ดังแสดงในรูปที่ 12.1 สังเกตว่า แถบเปลี่ยนของย่านความถี่ที่อยู่ติดกันจะมีลักษณะที่สมมาตรกัน คือ ขณะที่ผลตอบสนองทางความถี่ของย่านหนึ่งลาดลง ผลตอบสนองในย่านที่ติดกันก็จะลาดขึ้นในลักษณะที่สมมาตรกัน ซึ่งก็จะทำให้ผลตอบสนองความถี่รวมเป็นหนึ่ง

ตัวกรองที่ใช้มีอันดับเท่ากับ 50 ออกแบบโดยใช้วิธีหน้าต่างแฮมมิ่ง ตัวที่หนึ่งเป็น LPF, ตัวที่สองถึงสี่เป็น BPF, และตัวที่ห้าเป็น HPF ซึ่งมีผลตอบสนองต่ออิมพัลส์ คือ

$$\left.\begin{aligned}
h_1(n) &= w(n)\left[\frac{\sin(\omega_1'(n-M))}{\pi(n-M)}\right] \\
h_2(n) &= w(n)\left[\frac{\sin(\omega_2'(n-M)) - \sin(\omega_1'(n-M))}{\pi(n-M)}\right] \\
h_3(n) &= w(n)\left[\frac{\sin(\omega_3'(n-M)) - \sin(\omega_2'(n-M))}{\pi(n-M)}\right] \\
h_4(n) &= w(n)\left[\frac{\sin(\omega_4'(n-M)) - \sin(\omega_3'(n-M))}{\pi(n-M)}\right] \\
h_5(n) &= w(n)\left[\delta(n-M) - \frac{\sin(\omega_4'(n-M))}{\pi(n-M)}\right]
\end{aligned}\right\} \quad (12.1)$$

โดยที่ M=25 และ $\omega_i' = 2\pi f_i/f_s$ เมื่อ f1, f2, f3, และ f4 เท่ากับ 3, 7, 11, 15 kHz ตามลำดับ

สังเกตได้ว่า ผลบวกของ $h_i(n)$ ทุกตัว จะได้เป็นตัวกรองแบบผ่านตลอด ที่สัญญาณขาออกล้าหลังสัญญาณขาเข้า M จุด เขียนเป็นสมการได้ว่า

$$h_1(n) + h_2(n) + h_3(n) + h_4(n) + h_5(n) = \delta(n-M) \qquad (12.2)$$

แปลง z ได้เป็น $$H_1(z) + H_2(z) + H_3(z) + H_4(z) + H_5(z) = z^{-M} \qquad (12.3)$$

ซึ่งทำให้เราสามารถเขียนฟังก์ชั่นถ่ายโอนของตัวกรองตัวที่ห้าได้เป็น

$$H_5(z) = z^{-M} - H_1(z) - H_2(z) - H_3(z) - H_4(z) \tag{12.4}$$

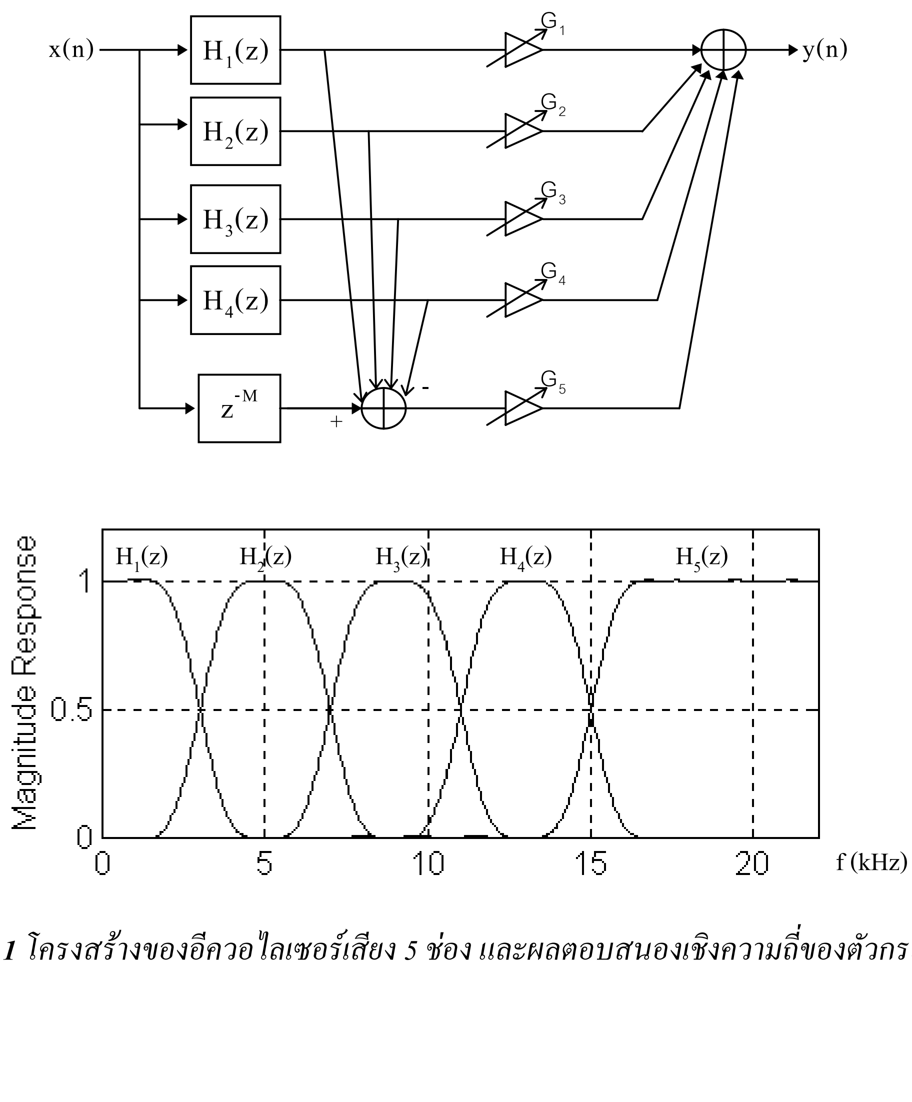
จะได้ว่า $Y_5(z) = z^{-M}X(z) - Y_1(z) - Y_2(z) - Y_3(z) - Y_4(z)$

และสมการทางเวลา คือ $y_5(n) = x(n-M) - y_1(n) - y_2(n) - y_3(n) - y_4(n)$

ดังนั้น เราสามารถประหยัดโดยไม่จำเป็นต้องประมวลผลในส่วนของ $H_5(z)$ จริง ๆ แต่ว่าสามารถหา $y_5(n)$ ได้โดยใช้สัญญาณขาเข้าดึงให้ล้าหลังลง M จุด แล้วลบออกด้วยสัญญาณขาออกจากตัวกรองอื่นอีกสี่ตัว โครงสร้างของอีควอไลเซอร์นี้แสดงดังในรูปที่ 12.1

ผลตอบจากตัวกรองทั้งหมด จะนำมาผ่านตัวคูณที่ปรับค่าได้โดยผู้ใช้ และนำผลคูณทั้งหมดมาบวกกันเป็นผลลัพธ์สุดท้าย จะได้ว่าฟังก์ชั่นถ่ายโอนรวมของระบบนี้ คือ

$$H_{total}(z) = G_1H_1(z) + G_2H_2(z) + G_3H_3(z) + G_4H_4(z) + G_5H_5(z) \tag{12.5}$$



รูปที่ 12.1 *โครงสร้างของอีควอไลเซอร์เสียง 5 ช่อง และผลตอบสนองเชิงความถี่ของตัวกรองที่ใช้*

การใช้ตัวกรอง FIR ในงานนี้มีข้อดี คือ ออกแบบสัมประสิทธิ์ของตัวกรองได้ง่าย และให้เฟสที่เป็นเชิงเส้นที่สมบูรณ์ในทุกย่านความถี่ อย่างไรก็ตาม มีข้อเสีย คือ มันจะต้องการการประมวลผลที่ค่อนข้างมาก เพราะต้องคำนวณตัวกรอง FIR อันดับ 40 ถึง 4 ตัว ดังนั้น ถ้าหากมีข้อจำกัดในเรื่องทรัพยากรในการประมวลผลอาจต้องพยายามออกแบบให้เป็นตัวกรองแบบ IIR ซึ่งจะทำได้ยากกว่า เพราะ ผลตอบสนองเชิงความถี่ของตัวกรอง IIR ไม่สมมาตรกันในช่วงแถบเปลี่ยนเหมือนตัวกรอง FIR ใน [4] ได้เสนอวิธีการออกแบบอีควอไลเซอร์ที่ใช้ตัวกรอง IIR หลาย ๆ ตัวมาต่ออนุกรมกัน แทนที่จะเป็นการต่อขนานกัน ส่วนใน [17] ได้เสนอการใช้ตัวกรอง IIR หลาย ๆ ตัวมาขนานกันเพื่อทำอีควอไลเซอร์ในอุปกรณ์ช่วยได้ยิน

<u>เสียงสะท้อน</u> (Echo)

การสร้างเอ็ฟเฟ็คเสียงสะท้อน เป็นการเปลี่ยนเสียงขาเข้าที่เป็นเสียงโมโนปกติ ให้มีลักษณะคล้ายเป็นเสียงที่มีการสะท้อนเหมือนจากผนังของห้องโถง หรือผนังในถ้ำ การทำเสียงสะท้อนทำได้ง่าย ๆ โดยใช้ตัวกรอง FIR ที่มีค่าผลตอบสนองต่ออิมพัลส์ดังในรูปที่ 12.2 ข) ค่า h(n) ที่ n=0 เปรียบเสมือนเป็นเสียงที่ได้ยินโดยตรงจากต้นกำเนิดเสียง และค่า h(n) ที่ n=D เป็นส่วนของเสียงที่มาจากการสะท้อนของวัตถุแล้วกลับมาได้ยินอีกครั้งหนึ่ง ซึ่งความดังของเสียงก็จะลดลงกว่าเสียงที่เข้ามาโดยตรงจากแหล่งกำเนิด ดังนั้น จะใช้ค่าสัมประสิทธิ์ $h(D) = \alpha$ โดยที่ $\alpha < 1$

ค่า D จะแปรตามระยะทางจากผู้ฟังถึงวัตถุที่สะท้อนเสียง ถ้าใช้ค่า D มากขึ้นก็เป็นการจำลองว่า วัตถุที่สะท้อนอยู่ไกลขึ้น โดยเราอาจคำนวณประมาณค่า D ได้ดังนี้

สมมติว่าต้องการสร้างเสียงสะท้อนที่เกิดจากวัตถุที่อยู่ไกลออกไป 30 เมตร จะได้ว่า เสียงต้องเดินทางไปกลับจากวัตถุที่สะท้อนเป็นระยะทาง = 60 เมตร

เนื่องจาก เสียงมีความเร็วประมาณ 335 เมตรต่อวินาที ฉะนั้น ระยะเวลาที่เสียงต้องใช้ในการเดินทาง = 60/335 = 0.1791 วินาที นั่นคือ D จะต้องล้าหลังจากจุดเริ่มต้นเท่ากับ 0.1791 วินาที

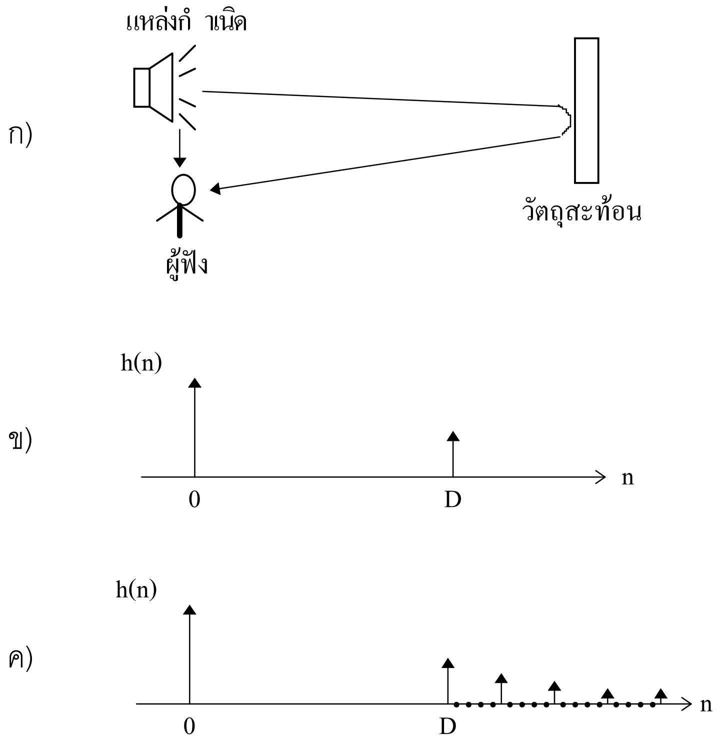
เทียบ D ให้เป็นจำนวนจุดสัญญาณ (sample) โดยใช้ค่าอัตราการสุ่มมาคำนวณ เช่น สมมติว่าใช้อัตราการสุ่ม = 8 kHz จะได้ว่าเวลา 0.1791 วินาทีนี้ มีค่าเทียบเท่ากับจำนวนจุดของสัญญาณ คือ $0.1791 \times 8000 = 1430$ จุด ดังนั้น ต้องใช้ D = 1430

ถึงแม้ D มีค่ามากถึง 1430 ซึ่งหมายความว่า อันดับของตัวกรองนี้เท่ากับ 1430 แต่ตัวกรองนี้ไม่ได้มีการคำนวณที่มากมายถึงขนาดนั้น เนื่องจากสัมประสิทธิ์ที่จุดอื่น ๆ ของ h(n) มีค่าเป็นศูนย์ เพราะฉะนั้น สมการผลต่างของระบบนี้ จะเหลือเพียงแค่

$$y(n) = x(n) + \alpha\, x(n - D) \qquad (12.6)$$

เพียงใช้ระบบง่าย ๆ แค่นี้เราก็จะได้ตัวกำเนิดเสียงสะท้อนอย่างง่ายแล้ว แต่เสียงที่ได้อาจฟังดูไม่เหมือนจริงนัก เนื่องจาก โดยปกติเสียงที่สะท้อนจากวัตถุ ถ้าวัตถุไม่ได้เรียบสนิท หรือมีวัตถุใกล้เคียงในละแวกนั้นอีก เสียงมักไม่ได้สะท้อนกลับมาที่เวลาเดียวดังที่ได้วิเคราะห์ ดังนั้น เพื่อให้เหมือนจริงมากขึ้น เราจะให้เสียงสะท้อนกลับมาหลาย ๆ ครั้ง และมาถึงผู้ฟังด้วยเวลาต่าง ๆ กัน และด้วยขนาดที่เบาลง ๆ ตัวอย่างของผลตอบสนองต่ออิมพัลส์ที่ต้องการใช้ในกรณีนี้ แสดงดังในรูปที่ 12.2 ค)

แต่การทำเสียงสะท้อนนี้ เป็นตัวอย่างของการใช้ผลทางภาวะชั่วครู่ (transient) ของระบบให้เกิดปรากฏการณ์ที่พิสดารขึ้นมา ซึ่งต่างจากการใช้งานระบบเป็นตัวกรองความถี่ที่เราได้ศึกษามา เพราะการใช้งานแบบนั้นเราสนใจผลทางสภาวะอยู่ตัว (steady state) ของระบบ นั่นก็คือ ผลตอบสนองเชิงความถี่ของระบบนั่นเอง



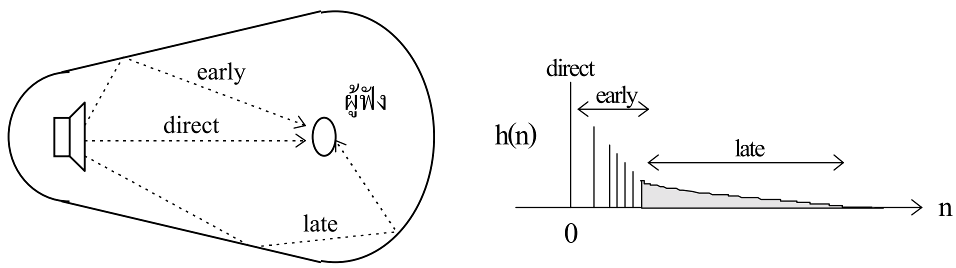
***รูปที่ 12.2** การเกิดเสียงสะท้อน และผลตอบสนองต่ออิมพัลส์ของระบบที่สร้างเสียงสะท้อน*

<u>เสียงจำลองการสะท้อนของห้อง</u> (Reverberation)[1],[4]

การจำลองเสียงสะท้อนของห้อง บางทีก็เรียกว่า เสียงเซอร์ราวด์ (surround) จะแปลงเสียงปกติให้เป็นเสียงที่มีการสะท้อนก็กก้องเสมือนเสียงนั้นเกิดขึ้นในห้องขนาดใหญ่ ประโยชน์ของมันได้แก่ การใช้ในการเล่นเสียงดนตรีในบ้าน แล้วจำลองให้เหมือนกับวงดนตรีกำลังเล่นอยู่ในโรงแสดงดนตรีขนาดใหญ่ โดยใช้เพียงแค่ลำโพงหน้าคู่เดียวเท่านั้น หรือการฉายภาพยนต์ในบ้านแล้วให้ความรู้สึกเหมือนชนภาพยนตร์อยู่ในโรงขนาดใหญ่ ในปัจจุบันได้มีการประยุกต์ไปใช้ในสนามกีฬากลางแจ้ง

ขนาดใหญ่ เพื่อใช้แสดงคอนเสิร์ต โดยทำให้เสียงที่ออกมาเสมือนเป็นการเล่นอยู่ในโรงแสดงคอนเสิร์ต หรืออาจใช้ในขณะมีการแข่งขันกีฬา โดยทำให้เสียงเชียร์กีฬาของผู้ดู กึกก้องเสมือนเชียร์อยู่ในสนามกีฬาในร่มก็ได้

หลักการของการทำเสียงสะท้อนของห้อง ก็เหมือนกับการทำเสียงสะท้อนจากวัตถุดังที่ได้กล่าวมา แต่คราวนี้เสียงสะท้อนจะเกิดขึ้นด้วยจำนวน และความหนาแน่นที่มากกว่ามาก เพราะเกิดจากการสะท้อนจากผนัง และเพดานรอบตัวผู้ฟัง ดังแสดงในรูปที่ 12.3 องค์ประกอบหลักของเสียงที่ผู้ฟังได้ยิน คือ เสียงที่มาตรงจากแหล่งกำเนิด จากนั้น ก็จะตามด้วยเสียงสะท้อนในระยะเริ่มแรก (early reflection) ซึ่งเกิดจากการสะท้อนของผนังที่อยู่ใกล้แหล่งกำเนิด การสะท้อนในระยะเริ่มแรกมีความหนาแน่นที่น้อย แต่มีขนาดใหญ่ หลักจากนั้นก็เป็นการสะท้อนในระยะท้าย (late reflection) ซึ่งมาจากการสะท้อนของผนังที่อยู่ไกลออกไป และการสะท้อนมากกว่า 1 ทอดก่อนมาถึงผู้ฟัง ซึ่งการสะท้อนระยะท้ายนี้มีความหนาแน่นมาก แต่มีขนาดเล็ก ระยะเวลาของการเกิดเสียงสะท้อนทั้งหมดจนผู้พูดไม่ได้ยินอาจกินเวลามากถึง 2 ถึง 4 วินาที เราสามารถจำลองขนาดของเสียงที่สะท้อนให้น้อยลง ๆ ที่เวลาผ่านไปได้ดีด้วยฟังก์ชั่นเอกซ์โปเนนเชียล



รูปที่ 12.3 *การเกิดเสียงสะท้อนในห้อง และผลตอบสนองต่ออิมพัลส์ของระบบ*

ฟังก์ชั่นถ่ายโอนที่จำลองการเกิดเสียงสะท้อนในห้องมีหลายแบบ ในที่นี้จะยกตัวอย่างโมเดลที่เสนอโดย Schroeder ในปี 1985 โดยใช้ฟังก์ชั่นถ่ายโอนแบบ IIR 2 ชนิดมาต่อกัน เพื่อทำให้เกิดผลตอบสนองต่ออิมพัลส์ที่ยาวไปจนถึงเวลาเป็นอนันต์ ฟังก์ชั่นถ่ายโอน 2 ชนิดนี้ ได้แก่

1) <u>ตัวกำเนิดเสียงสะท้อนแบบพื้นฐาน</u> (echo generator หรือ plane reverberator) เป็นตัวสร้างเสียงสะท้อนที่ยาวไปจนถึงเวลาเป็นอนันต์ โดยแต่ละเสียงห่างกัน D จุด และมีขนาดลดลงด้วยอัตราส่วน a^n โดยที่ a น้อยกว่า 1 จะได้ผลตอบสนองต่ออิมพัลส์ดังแสดงในรูปที่ 12.4 ซึ่งมีสมการคือ

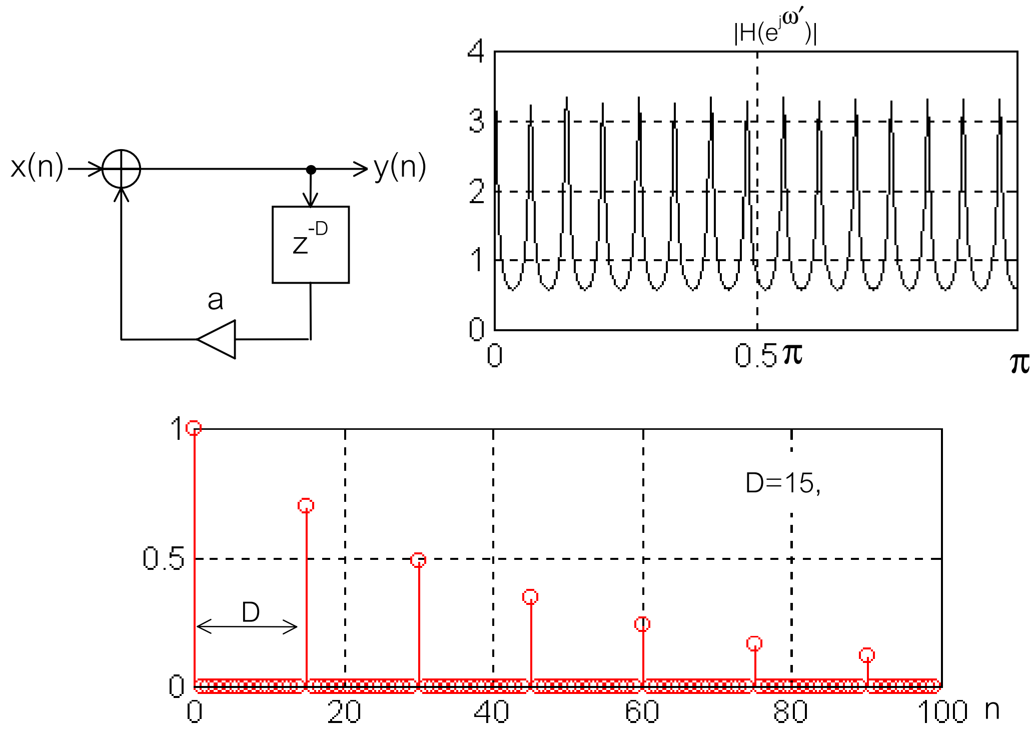
$$h(n) = \delta(n) + a\delta(n-D) + a^2\delta(n-2D) + \ldots \qquad (12.7)$$

แปลง z ได้ $$H(z) = 1 + az^{-D} + a^2z^{-2D} + \ldots \qquad (12.8)$$

โดยใช้อนุกรมเลขาคณิตกับสมการที่ 12.7 จะได้ว่า H(z) ลู่เข้าสู่ฟังก์ชั่นที่เป็น IIR ดังนี้

$$H(z) = 1 + az^{-R} + (az^{-R})^2 + (az^{-R})^3 + \ldots \qquad H(z) = \frac{1}{1 - az^{-D}} \tag{12.9}$$

ตัวกำเนิดเสียงสะท้อนแบบพื้นฐานนี้ มีโพลอยู่ D ตำแหน่งโดยที่ไม่มีศูนย์อยู่เลย ทำให้ผลตอบสนองเชิงความถี่มีลักษณะที่เป็นยอดแหลมตามตำแหน่งของโพล ดังแสดงในรูปที่ 12.4 การใช้ตำกำเนิดเสียงสะท้อนแบบนี้โดด ๆ จะทำให้ได้เสียงที่ฟังดูไม่เป็นธรรมชาตินัก



รูปที่ 12.4 โครงสร้าง และผลตอบสนองต่ออิมพัลส์ของตัวกำเนิดเสียงสะท้อนแบบพื้นฐาน

2) <u>ตัวกำเนิดเสียงสะท้อนแบบผ่านทุกความถี่</u> (all-pass reverberator) ตัวกำเนิดเสียงสะท้อนนี้มีลักษณะพิเศษ คือ มีผลตอบสนองเชิงความถี่ที่ผ่านทุกความถี่เท่ากับ 1 เท่ากัน ($|H(e^{j\omega'})| = 1$) มีฟังก์ชั่นถ่ายโอน คือ

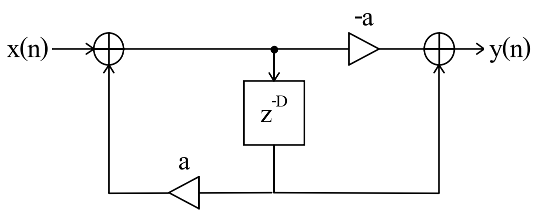
$$H(z) = \frac{-a + z^{-D}}{1 - az^{-D}} \tag{12.10}$$

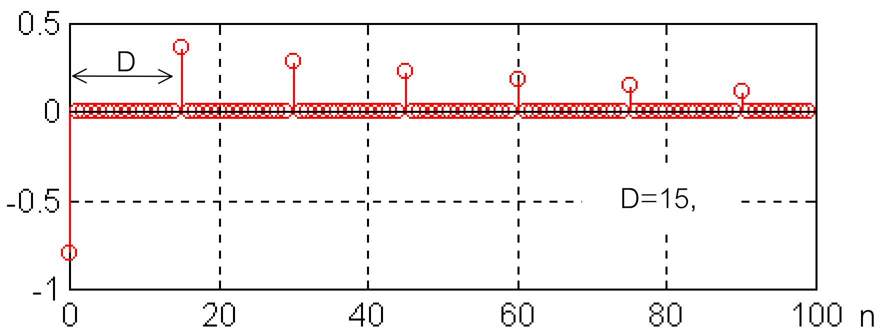
โครงสร้าง direct form 2 ของระบบนี้ รวมทั้งผลตอบสนองต่ออิมพัลส์แสดงดังในรูปที่ 12.5 ซึ่งเราสามารถวิเคราะห์หาสมการของ h(n) ได้ โดยแตก H(z) ในสมการที่ 12.10 ออกเป็นผลหาร และเศษของการหารได้ ดังนี้

$$H(z) = A + \frac{B}{1 - az^{-D}}, \quad \text{โดย } A = -\frac{1}{a} \text{ และ } B = \frac{1-a^2}{a} \qquad (12.11)$$

จะได้ $$h(n) = (A+B)\delta(n) + Ba\delta(n-D) + Ba^2\delta(n-2D) + \ldots \qquad (12.12)$$

จะเห็นได้ว่า h(n) มีลักษณะคล้าย ๆ กับตัวกำเนิดเสียงสะท้อนแบบแรก เพียงแต่มีอิมพัลส์ลูกแรกเป็นค่าติดลบ





***รูปที่ 12.5** โครงสร้าง และผลตอบสนองต่ออิมพัลส์ของตัวกำเนิดเสียงสะท้อนแบบผ่านทุกความถี่*

สำหรับตัวกำเนิดเสียงสะท้อนในห้องของ Schroeder ใช้ตัวกำเนิดเสียงสะท้อนในข้อ 1 และ 2 มาต่อกันเป็นโครงสร้างดังในรูปที่ 12.6 ซึ่งถ้าเลือก a_i, G_i, และ D_i ที่เหมาะสม ก็จะทำให้เกิดเสียงสะท้อนเสมือนอยู่ในห้องแสดงดนตรีขนาดใหญ่ได้ ตัวอย่างของผลตอบสนองต่ออิมพัลส์ของระบบนี้แสดงอยู่ในรูปเดียวกัน โดยได้ใช้ค่าพารามิเตอร์ต่าง ๆ ดังนี้

$D_1 = 29, D_2 = 37, D_3 = 44, D_4 = 50, D_5 = 27,$ และ $D_6 = 31$

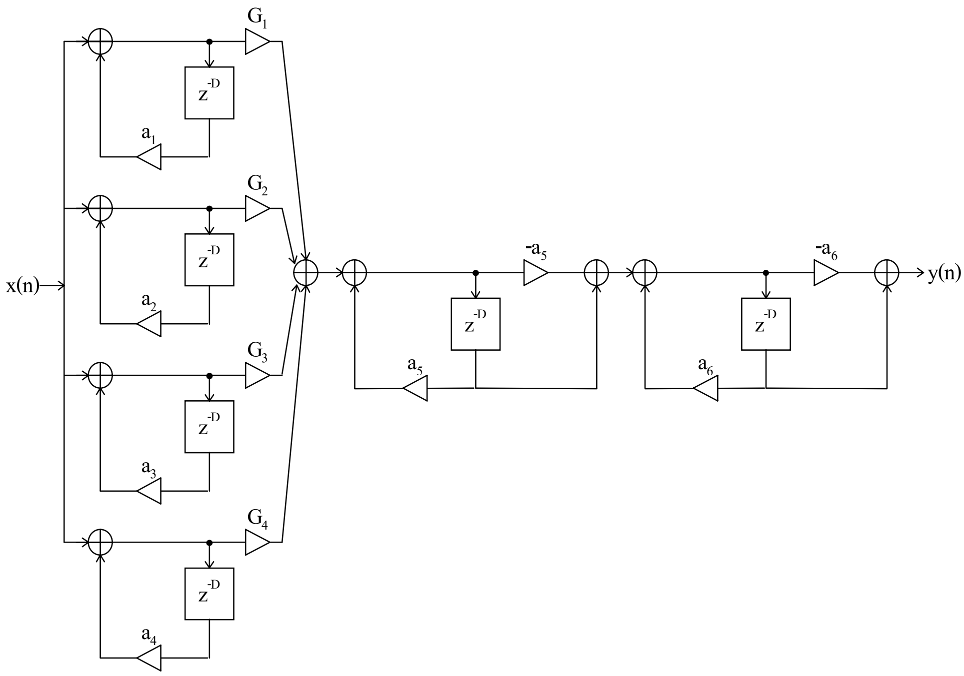
$G_1 = 1, G_2 = 0.9, G_3 = 0.8,$ และ $G_4 = 0.7$

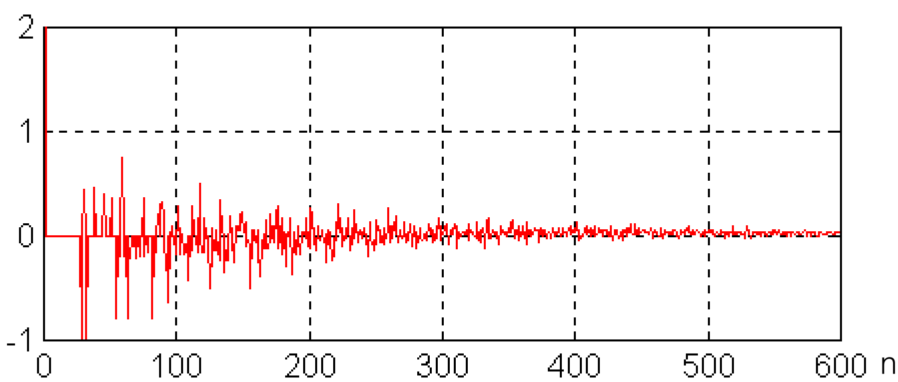
$a_1 = a_2 = a_3 = a_4 = a_5 = a_6 = 0.8$

ตัวกำเนิดเสียงสะท้อนในห้อง สามารถสร้างด้วยตัวกรอง FIR ก็ได้ ซึ่งถึงแม้มีข้อเสีย คือ ต้องใช้อันดับของตัวกรองสูง ทำให้ใช้การประมวลผลที่มาก แต่ก็มีข้อดี คือ สามารถควบคุมค่าของ h(n) แต่ละค่าได้โดยอิสระต่อกัน ไม่ยึดติดกับโมเดลเหมือนการใช้ตัวกรอง IIR ทำให้เราสามารถใช้ผล

ตอบสนองต่ออิมพัลส์จริง ๆ ของโรงแสดงดนตรีต้นแบบที่ต้องการมาเป็น h(n) ของตัวประมวลผลได้ เพื่อให้ได้เสียงเสมือนอยู่โรงแสดงดนตรีต้นแบบจริง ๆ

ผลตอบสนองต่ออิมพัลส์ของโรงแสดงดนตรีต้นแบบนั้น สามารถหาได้โดยการทำการทดลองวัดมาจากโรงแสดงดนตรีนั้นจริง ๆ (มีขั้นตอนวิธีเหมือนการหาผลตอบสนองต่ออิมพัลส์ของเสียงสามมิติ ซึ่งจะได้อธิบายในตอนต่อไป) หรือใช้ซอฟท์แวร์พิเศษจำลองโรงดนตรีที่ต้องการขึ้นมาทั้งขนาด, รูปร่าง, และสัมประสิทธิ์การสะท้อนเสียง แล้วคำนวณหาผลตอบสนองต่ออิมพัลส์ที่จะเกิดขึ้นได้ การสร้างเสียงสะท้อนในห้องโดยใช้ตัวกรอง FIR นี้ ปัจจุบันได้มีผู้ทำได้โดยใช้อันดับของตัวกรองถึง 262,144 โดยใช้อัตราการสุ่มเท่ากับ 48 kHz[22] นั่นหมายถึง สามารถจำลองผลตอบสนองต่ออิมพัลส์ได้ยาวถึงประมาณ 5 วินาทีครึ่งทีเดียว



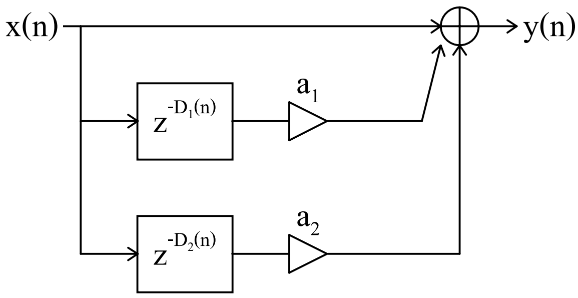


รูปที่ 12.6 *โครงสร้าง และผลตอบสนองต่ออิมพัลส์ของตัวจำลองเสียงสะท้อนในห้อง*

<u>เสียงคอรัส</u> (Chorus)[1],[4]

คอรัสเป็นการจำลองการประสานเสียง สมมติว่าเรามีเสียงเพลงซึ่งเล่นด้วยเครื่องดนตรีเพียงชิ้นเดียว (เล่นคนเดียว) เราสามารถจำลองเสียงประสานให้เหมือนเป็นการเล่นเครื่องดนตรีชนิดเดียวกัน และเพลงเดียวกันด้วยคนหลาย ๆ คนพร้อมกันได้

โครงสร้างง่าย ๆ ของตัวกำเนิดเสียงคอรัสแสดงดังในรูปที่ 12.7 ซึ่งเป็นการจำลองเครื่องดนตรีที่เล่นโดยคนสามคน สัมประสิทธิ์ของการล้าหลังของคนเล่นคนที่สอง และสาม คือ D_1 และ D_2 เป็นค่าไม่คงที่ โดยจะเปลี่ยนแปลงช้า ๆ ตามเวลา เราจึงเขียนแทนว่าเป็น $D_1(n)$ และ $D_2(n)$ เพื่อบอกว่าค่าทั้งสองเป็นตัวแปรที่แปรตาม n การที่เรามีการล้าหลังที่ไม่คงที่เติมให้กับเสียงของผู้เล่นคนที่สอง และสามเช่นนี้ เป็นการจำลองให้เหมือนกับว่า ผู้เล่นทั้งสามคนนี้เล่นเพลงไม่พร้อมกันทีเดียว แต่มีความช้าเร็วเหลื่อมกันบ้างตลอดเวลา ทำให้ได้เสียงที่ออกมาคล้ายเสียงประสานที่ไพเราะได้



รูปที่ 12.7 *ตัวอย่างโครงสร้างของตัวกำเนิดเสียงคอรัส*

<u>เสียงสามมิติ</u> (3-Dimemsional Sound)[20],[21]

หัวข้อนี้เป็นหัวข้อสุดท้ายของการปรับแต่งคุณลักษณะของเสียง และก็เป็นการประยุกต์ใช้งานที่พิสดารที่สุด วัตถุประสงค์ก็คือ เราต้องการปรับเสียงที่เป็นโมโนธรรมดา ให้กลายเป็นเสียงที่เหมือนมาจากทิศทางใดทิศทางหนึ่ง ซึ่งไม่เพียงเฉพาะ ทิศทาง 360 องศาในแนวราบเท่านั้น แต่รวมถึงมุมที่มาจากทิศทางด้านบน และด้านล่างตัวเราด้วย สิ่งที่พิเศษ ก็คือ เราจะสร้างเสียงสามมิตินี้โดยใช้เพียงแค่ลำโพงหน้าคู่เดียว หรือใช้หูฟัง (headphone) เท่านั้น ไม่มีการใช้ลำโพงเซอร์ราวด์ (ลำโพงหลัง) เหมือนในระบบดอร์บี้ในโรงภาพยนต์แต่อย่างใด ๆ การทำเสียงสามมิติได้ในทุกทิศทางเช่นนี้ได้ทำให้ภาพยนตร์สามมิติ ซอฟท์แวร์เกมส์ และซอฟท์แวร์อื่น ๆ สามารถเลียนแบบสถานการณ์ในสิ่งแวดล้อมได้เหมือนจริงยิ่งขึ้น ซึ่งเทคโนโลยีพวกนี้เรียกกันเป็นศัพท์เทคนิคว่า virtual reality

ความเป็นไปได้ของการทำเสียงสามมิติเกิดจาก แนวความคิดว่า เนื่องจากคนเราสามารถแยะแยะทิศทางของเสียงได้ โดยใช้หู (ซึ่งเปรียบเสมือนเป็นเซนเซอร์รับเสียง) เพียงสองข้างเท่านั้น ดังนั้นเราก็น่าจะสามารถจำลองการเกิดเสียงที่มาจากทิศทางใด ๆ ได้โดยใช้เพียงแค่สองลำโพงเช่นกัน ซึ่งคำกล่าวนี้ จริง ๆ แล้วก็มิได้ขัดแย้งกับกฎของฟิสิกส์ หรือทางไฟฟ้าแต่อย่างใด

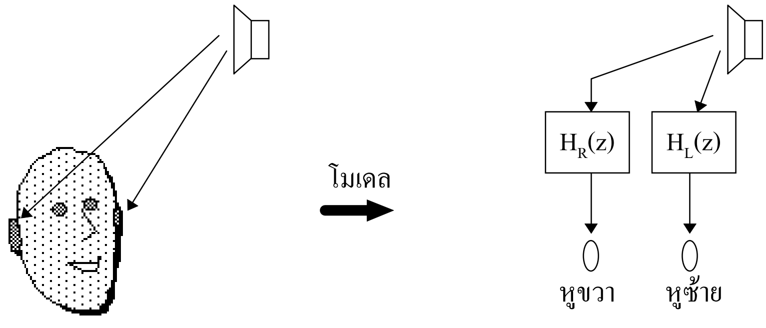
การที่คนเราสามารถแยกแยะทิศทางของเสียงได้สามมิติ ทั้งที่มีเพียงแค่สองหู (ไม่ได้มีหูหน้าหลัง หรือบนล่างด้วย) แสดงว่ายังองค์ประกอบบางสิ่งบางอย่างที่ทำให้ช่วยเราให้มีสัมผัสในการแยกแยะทิศทางได้ องค์ประกอบเหล่านี้ ได้แก่ การตอบสนองของใบหู และหัวคนต่อเสียงที่เข้ามา และสิ่งแวดล้อมอื่น ๆ รอบตัวที่เรารับรู้ได้จากทางอื่น เช่น การได้เห็นร่วมด้วย ตัวอย่างของการตอบสนองของหัวคนต่อเสียงในทิศทางต่าง ๆ ที่เราเข้าใจได้ง่าย ๆ ได้แก่

ก) ความแตกต่างของความดังของเสียงที่มาถึงหูแต่ละข้าง (Interaural Intensity Difference หรือ IID) เช่น เสียงที่มาทางด้านซ้ายจะกระทบหูซ้ายด้วยความดังมากกว่าเสียงที่กระทบหูขวา

ก) ความแตกต่างของเวลาที่เสียงมาถึงหูแต่ละข้าง (รวมถึงเฟสที่แตกต่างกันด้วย) (Interaural Time Difference หรือ ITD) เช่น เสียงที่มาทางด้านซ้ายจะกระทบหูซ้ายก่อนหูขวา หรือการที่คนรับรู้ทิศทางของเสียงความถี่ต่ำยากกว่า ก็เนื่องจากเสียงความถี่ต่ำ จะมีความยาวคลื่นมาก ทำให้ความแตกต่างของเฟสที่รับรู้ที่หูทั้งสองน้อยลง

ถึงแม้เราจะมีความเข้าใจถึงการรับรู้ทิศทางมากขึ้น แต่ก็ยังไม่ถึงขั้นที่จะสามารถสรุปเป็นสมการคณิตศาสตร์ หรือฟังก์ชั่นออกมาได้ว่าเสียงที่มาจากทิศทางต่าง ๆ เป็นอย่างไร แต่อย่างไรก็ตาม ความเข้าใจที่ยังไม่สมบูรณ์ก็ไม่ใช่ปัญหาที่กีดกั้นเทคโนโลยีของการสร้างเสียงสามมิติ เนื่องจาก เราสามารถทำการทดลองวัดฟังก์ชั่นถ่ายโอนของเสียงที่มาจากทิศทางต่าง ๆ ได้ แล้วเก็บสิ่งที่วัดได้เป็นฐานข้อมูลที่สามารถเรียกออกมาใช้เพื่อกำเนิดเสียงสามมิติในทิศทางต่าง ๆ ต่อไป

ก่อนอื่นต้องมีความเข้าใจเกี่ยวกับฟังก์ชั่นถ่ายโอนที่จะทำการวัดก่อน สมมติว่าเราอยู่ในห้องทดลองที่ไม่มีเสียงอื่นใด นอกจากเสียงจากแหล่งกำเนิดที่ทดลองเท่านั้น ถ้าตั้งแหล่งกำเนิดเสียงไว้ที่ทิศทางใดทิศทางหนึ่งของผู้ฟัง เสียงที่ผู้ฟังได้ยินที่หูซ้าย และหูขวา จะสามารถโมเดลได้ว่า เกิดจากการนำเสียงจากแหล่งกำเนิดผ่านฟังก์ชั่นถ่ายโอน $H_R(z)$ ได้เป็นเสียงที่ได้ยินที่หูขวา และผ่าน $H_L(z)$ ได้เป็นเสียงที่ได้ยินที่หูซ้าย ดังแสดงในรูปที่ 12.8 ฟังก์ชั่นทั้งสองนี้ จะรวมเอารายละเอียดเกี่ยวกับการได้ยินในทิศทางนั้น ๆ ไว้ทั้งหมด ซึ่งได้แก่ IID กับ ITD ที่ได้กล่าวมาแล้ว และปฏิสัมพันธ์ที่เสียงกระทำกับหัว และใบหูของผู้ฟัง เป็นต้น ได้มีการตั้งชื่อฟังก์ชั่นนี้ว่า ฟังก์ชั่นถ่ายโอนอันเนื่องมาจากหัวคน (Head-Related Transfer Function หรือ HRTF)



รูปที่ 12.8 *แนวความคิดของฟังก์ชั่นถ่ายโอนเนื่องมาจากหัวคน (HRTF)*

เราจะโมเดลให้ HRTF เป็นตัวกรองแบบ FIR ที่มีสัมประสิทธิ์คงที่ และสามารถทำการทดลองเพื่อวัดค่าสัมประสิทธิ์ของ HRTF ที่มุมของแหล่งกำเนิดเสียงต่าง ๆ ได้ โดยนำเอาไมโครโฟนสองตัวไปติดที่ในรูหูทั้งสองข้างของหุ่น หรือคนที่จะเป็นแบบ จากนั้นส่งเสียงที่เตรียมไว้จากแหล่งกำเนิดที่ทิศทางที่ต้องการ และระหว่างที่เล่นเสียงนั้นก็บันทึกเสียงจากไมโครโฟนทั้งสอง เมื่อได้เสียงที่บันทึกไว้ ก็คือ เรารู้ทั้งสัญญาณขาเข้า และขาออกของ HRTF แล้ว ก็สามารถใช้เทคนิคทางการประมวลผลสัญญาณ เพื่อวิเคราะห์หาสัมประสิทธิ์ของ HRTF ได้

ในการจำลองสร้างเสียงสามมิติขึ้นมา ก็เพียงแต่นำเอาเสียงโมโนปกติ ผ่านตัวกรอง $H_L(z)$ และ $H_R(z)$ เสียงที่เป็นสัญญาณขาออกก็เป็นเสียงที่พร้อมจะไปเล่นออกที่ลำโพงซ้าย และขวาตามลำดับ เพื่อให้เกิดเป็นเสียงที่เสมือนมาจากทิศทางที่ต้องการได้

หลักการวัดเพื่อหา HRTF นี้ถึงแม้จะง่ายมาก แต่การวัดจริงก็มีข้อปลีกย่อยที่ต้องระวัง และควบคุมให้ดีเพื่อให้ได้ค่าที่ถูกต้อง เป็นโชคดีที่เราอาจไม่จำเป็นต้องไปทำการทดลองวัด HRTF เอง เพราะที่ผ่านมาได้มีผู้เคยทดลองวัด HRTF แล้ว และบางทีก็ได้เผยแพร่ค่าสัมประสิทธิ์ที่วัดได้แก่สาธารณชน ที่แพร่หลายมากก็คือ การวัดโดยใช้หุ่นรูปหัวคนโดยห้องปฏิบัติการ MIT Media ในปี 1994 ซึ่งมีรายละเอียดของการวัด และค่าสัมประสิทธิ์ที่วัดได้ใน [20]

อุปกรณ์ช่วยได้ยิน (Hearing Aids)[17],[18]

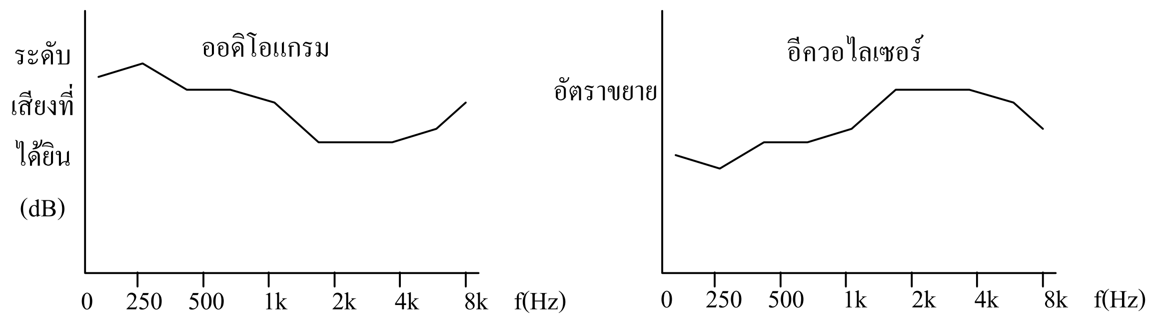
อุปกรณ์นี้ใช้กับผู้ป่วยที่มีปัญหาในการได้ยิน โดยจะรับเสียงจากภายนอก จากนั้นขยายและส่งไปยังหูฟังที่เสียบอยู่ที่หูของผู้ป่วย ซึ่งนอกจากทำหน้าที่ขยายเสียงธรรมดาแล้ว หน้าที่ที่สำคัญกว่านั้นก็คือ การปรับแต่งสัญญาณเสียงให้มีความเหมาะสมต่อการได้ยินของผู้ป่วยแต่ละราย การปรับแต่งที่สำคัญ ได้แก่

1. การลดสัญญาณรบกวน ในภาวะที่มีสัญญาณรบกวน อุปกรณ์ช่วยได้ยินควรจะสามารถลดสัญญาณรบกวนนั้นได้ เนื่องจากอุปกรณ์ช่วยได้ยินส่วนใหญ่เน้นที่การรับเสียงคนพูด จึงอาจใช้อัลกอริธึมเพื่อพยายามลดสัญญาณรบกวนที่ไม่ใช่เสียงพูดออก ซึ่งก็ควรเป็นอัลกอริธึมแบบปรับตัวได้ (adaptive) เพราะเราไม่ทราบลักษณะที่แน่นอนของสัญญาณล่วงหน้า

2. การขยายแบบปรับค่าได้อัตโนมัติ (Automatic Gain Control) อุปกรณ์ช่วยได้ยินควรมีความสามารถในการปรับอัตราขยายได้เองโดยอัตโนมัติ โดยมีอัตราขยายสูงในช่วงขนาดสัญญาณขาเข้าต่ำ และเมื่อขนาดสัญญาณขาเข้าใหญ่เกินระดับที่ตั้งไว้ก็จะลดอัตราขยายลงเพื่อไม่ให้สัญญาณขาออกเกิดความเพี้ยน

3. การปรับสเปกตรัมของเสียง (Frequency Shaping) ทำหน้าที่เหมือนกับอีควอไลเซอร์เสียงที่ได้อธิบายมาในหัวข้อก่อนหน้านี้ เนื่องจากผู้ป่วยจะมีความสามารถในการได้ยินในแต่ละความถี่แตกต่างกัน ซึ่งแพทย์สามารถตรวจสอบ และวัดระดับการได้ยินของผู้ป่วยที่ความถี่ต่าง ๆ ออกมาได้

เป็นกราฟที่เรียกว่า ออดิโอแกรม (audiogram) ซึ่งกราฟนี้ก็จะแตกต่างกันระหว่างหูซ้าย และขวา และแตกต่างกันในผู้ป่วยแต่ละคน เราจะใช้อีควอไลเซอร์ 2 ตัวแยกกันสำหรับหูซ้าย และขวา เพื่อปรับระดับการได้ยินในแต่ละย่านความถี่ให้อยู่ในระดับที่เหมาะสม ซึ่งอาจใช้ตัวกรองแบบ FIR หรือ IIR ก็ได้ดังที่ได้กล่าวมาแล้ว การใช้ตัวกรองดิจิตอลสำหรับหน้าที่นี้มีข้อดีที่เห็นได้ชัด คือ เราสามารถใช้ฮาร์ดแวร์เดียวกันสำหรับผู้ป่วยทุก ๆ คนได้ โดยเพียงแต่ปรับค่าสัมประสิทธิ์ของอีควอไลเซอร์ให้เหมาะสมกับออดิโอแกรมของผู้ป่วยแต่ละรายเท่านั้น ทำให้สะดวก และรวดเร็วต่อการใช้งานมาก



***รูปที่ 12.9** ตัวอย่างของออดิโอแกรม และผลตอบสนองเชิงความถี่ของอีควอไลเซอร์ที่ต้องการ*

การกรองสัญญาณฮาร์มอนิกใน UPS[19]

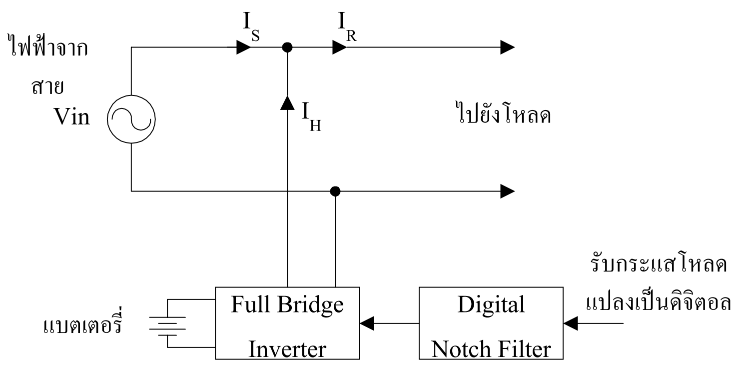
ปัญหาของ UPS หรือแหล่งจ่ายกำลังไม่หยุดชะงัก (uninterruptible power supply) เมื่อนำไปจ่ายโหลดที่ไม่เป็นเชิงเส้น ก็คือ โหลดนั้นจะดึงกระแสที่ทำให้มีความถี่ฮาร์มอนิกเกิดขึ้นในสาย ซึ่งถ้าหากไม่กำจัดทิ้ง กระแสฮาร์โมนิกเหล่านี้ก็อาจไปทำความเสียหายให้กับอุปกรณ์อื่น ๆ ได้

UPS ในปัจจุบันส่วนใหญ่มีส่วนประกอบของแบตเตอรี่ และวงจรอินเวอร์เตอร์สำหรับแปลงไฟตรงเป็นไฟสลับ ซึ่งจะทำงานจ่ายกระแสเฉพาะเมื่อไฟฟ้าดับ ส่วนในภาวะปกติตัวอินเวอร์เตอร์จะไม่ได้ใช้งาน ดังนั้น ในภาวะปกติเราสามารถนำเอาอินเวอร์เตอร์นี้มากำเนิดกระแสฮาร์มอนิก (I_H) เพื่อชดเชยเข้าไปในไฟฟ้าจากสาย (I_S) ดังแสดงในรูปที่ 12.10 ซึ่งถ้ากระแส I_H นี้ตรงกับส่วนของฮาร์มอนิกในกระแสที่ไปยังโหลด (I_R) ก็จะทำให้กระแสที่มาจากสาย คือ I_S เป็นรูปคลื่นซายน์สมบูรณ์ หรือไม่มีความถี่ฮาร์มอนิกหลงเหลืออยู่ ซึ่งเป็นภาวะที่เราต้องการ

เทคนิคที่ทำให้ตัวอินเวอร์เตอร์กำเนิดกระแสฮาร์มอนิกออกมาได้พอดีกับที่โหลดดึงไป ทำได้โดยการสุ่มค่ากระแสจากโหลดแล้วแปลงไปเป็นดิจิตอล ซึ่งกระแสโหลดที่ได้วัดได้นี้จะประกอบด้วยความถี่พื้นฐาน (50 เฮริตซ์) และ ความถี่ฮาร์มอนิกปนกันอยู่ จากนั้นใช้ตัวกรองดิจิตอลแบบตัดความถี่ (digital notch filter) กรองเอาเฉพาะความถี่พื้นฐานออก เหลือเพียงสัญญาณฮาร์มอนิกซึ่งจะนำไปใช้ควบคุมวงจรอินเวอร์เตอร์ให้กำเนิดกระแสฮาร์มอนิกที่ต้องการออกมาได้ การเลือกใช้ตัวกรองดิจิตอลเพื่อตัดความถี่พื้นฐานในที่นี้มีข้อดีกว่าการใช้ตัวกรองแอนะลอก คือ

1. เราต้องการให้กระแส I_H ตรงพอดีกับส่วนของฮาร์มอนิกที่มีอยู่ในกระแส I_R นั่นหมายความว่า ตัวกรองตัดความถี่ที่ใช้ต้องมีความคมที่ดี และมีเฟสที่เป็นเชิงเส้นสมบูรณ์ เพื่อไม่ให้สัญญาณขาออกของตัวกรองมีความผิดเพี้ยนทางรูปร่าง (หรือทางเฟสนั่นเอง) ตัวกรองชนิดนี้ทำได้ง่าย และแม่นยำโดยใช้ตัวกรองดิจิตอลแบบ FIR แต่จะทำได้ยาก และแพงกว่าด้วยตัวกรองแอนะลอก[19]

2. ในปัจจุบัน UPS มักมีส่วนควบคุมที่ทำด้วยดิจิตอลอยู่แล้ว เช่น การควบคุมวงจรอินเวอร์เตอร์ (ด้วยสัญญาณ PWM), การควบคุมการชาร์ตแบตเตอรี่, การควบคุมการติดต่อสื่อสารกับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม, การควบคุมระดับแรงดันไฟ, และการควบคุมคีย์กับจอแสดงผล (ถ้ามี) ซึ่งถ้าหากใช้ตัวประมวลผลสัญญาณสำหรับทำตัวกรองดิจิตอลแล้ว ฟังก์ชั่นทางดิจิตอลเหล่านี้ก็สามารถรวมให้ทำโดยตัวประมวลผลสัญญาณได้ และก็จะส่งผลให้ต้นทุนของ UPS มีราคาถูก



รูปที่ 12.10 *การกรองสัญญาณฮาร์มอนิกใน UPS โดยใช้การประมวลผลสัญญาณดิจิตอล*

การสร้างสัญญาณ (Waveform Generators)[1],[23]

ตัวกรองดิจิตอล IIR สามารถนำมาทำเป็นออสซิลเลเตอร์ดิจิตอลได้ ด้วยหลักการคล้ายคลึงกับระบบแอนะลอก กล่าวคือ เมื่อตัวกรองอยู่ในสภาวะเสถียรภาพแบบวิกฤต (critically stable) ผลตอบสนองต่ออิมพัลส์ของมันจะไม่ลู่ลงสู่ศูนย์ และก็จะไม่ใหญ่ขึ้นจนเป็นอนันต์ แต่จะคงรูปร่างสัญญาณที่เป็นคาบอย่างหนึ่งไว้ไปเรื่อย ๆ เช่น สัญญาณซายน์ เป็นต้น สภาวะเสถียรภาพแบบวิกฤตของตัวกรองดิจิตอลก็คือ สภาวะที่โพลของระบบมีขนาดเท่ากับหนึ่งนั่นเอง

ขอยกตัวอย่างโครงสร้างของตัวกรองแบบ direct form ที่กำเนิดสัญญาณซายน์ที่ความถี่ ω_0' ซึ่งก็คือ ตัวกรองที่มีผลตอบสนองต่ออิมพัลส์เป็นสัญญาณซายน์ที่ความถี่ดังกล่าวนั่นเอง เขียนเป็นสมการของผลตอบสนองต่ออิมพัลส์ได้ว่า

$$h(n) = \sin(\omega_0' n) \tag{12.13}$$

ซึ่งเมื่อแปลง z จะได้ฟังก์ชั่นถ่ายโอน คือ

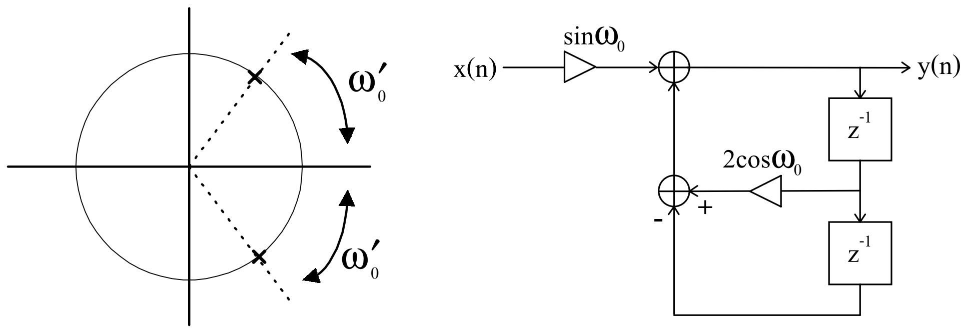
$$H(z) = \frac{\sin(\omega_0')z^2}{z^2 - 2\cos(\omega_0')z + 1} \qquad (12.14)$$

จะเห็นได้ว่า ตัวกรองนี้มีอันดับสอง และมีโพลที่มีขนาดเท่ากับ 1 สองตัวอยู่ที่ความถี่ $\pm\omega_0'$ ดังแสดงตำแหน่งของโพลในรูปที่ 12.11 เมื่อวิเคราะห์ต่อไปจะได้ว่าสมการผลต่างของตัวกรองนี้ คือ

$$y(n) = a_0x(n) - b_1y(n-1) - y(n-2) \qquad (12.15)$$

โดยที่ $a_0 = \sin(\omega_0)$ และ $b_1 = -2\cos(\omega_0)$

จะเห็นได้ว่า มีเทอมของสัญญาณขาเข้าเพียงเทอมเดียว คือ $a_0x(n)$ และสัมประสิทธิ์ a_0 ก็มีใช้อยู่ที่ตำแหน่งเดียวนี้ นั่นคือ a_0 เป็นเหมือนตัวสเกลขนาดของสัญญาณขาเข้า ดังนั้นมันเป็นค่าที่กำหนดขนาดของสัญญาณขาออกด้วย (เนื่องจากระบบเป็นเชิงเส้น) ส่วน b_1 ถูกใช้เป็นสัมประสิทธิ์ที่คูณตัวป้อนกลับ เพราะฉะนั้น b_1 จะเป็นตัวกำหนดตำแหน่งของโพล ซึ่งก็คือ กำหนดความถี่ในการแกว่งของสัญญาณขาออกนั่นเอง



รูปที่ 12.11 *ตำแหน่งโพล และ แผนภาพโครงสร้าง direct form 1 ของตัวกำเนิดสัญญาณซายน์*

ข้อดีของออสซิลเลเตอร์สัญญาณซายน์แบบนี้ก็คือ ทำได้ง่าย เพราะต้องการตัวคูณแต่ตัวเดียวเท่านั้น (ตัวคูณ a_0 ที่สัญญาณ x(n) นั้น ไม่จำเป็นต้องใช้ เพราะ เราจะใช้ x(n) เป็นสัญญาณอิมพัลส์ซึ่งมีค่าที่ n=0 จุดเดียว ดังนั้น $a_0x(n)$ จะใช้เป็นเพียงค่าเริ่มต้นที่ใช้กระตุ้นระบบเท่านั้น) อีกทั้งยังสามารถเปลี่ยนแปลงขนาด และความถี่ได้ง่าย โดยเปลี่ยนค่าสัมประสิทธิ์เพียงสองค่าเท่านั้น เราสามารถคำนวณค่าสัมประสิทธิ์ที่จะใช้กำเนิดความถี่ต่าง ๆ ไว้ล่วงหน้าก่อนแล้วเก็บไว้ในตาราง จากนั้น ตอนใช้งานก็ดึงค่าเหล่านี้มาโปรแกรมให้กับออสซิลเลเตอร์ ทำให้สามารถสร้างเป็นตัวออสซิลเลเตอร์ที่ปรับความถี่ได้ตามต้องการ

แต่ออสซิลเลเตอร์ชนิดนี้ก็มีปัญหาที่สำคัญ คือ เรื่องความคลาดเคลื่อนเมื่อนำไปใช้กับระบบเลขจำนวนเต็ม ซึ่งถ้าหากว่าความคลาดเคลื่อนมีมาก จะสามารถทำให้ความถี่ที่ออกมาคลาดเคลื่อนไปได้ หรือทำให้ระบบเสียเสถียรภาพก็เป็นไปได้เช่นกัน ใน [23] ได้วิเคราะห์ความคลาดเคลื่อนออสซิลเลเตอร์ชนิดนี้ในภาวะการทำงานที่จำนวนบิตต่าง ๆ และพบว่ามันสามารถทำงานได้ดี โดยให้กำเนิดสัญญาณซายน์ที่มีความเพี้ยนต่ำ ในช่วงความถี่กลาง ๆ คือ ตั้งแต่ $\omega_0' = 0.1\pi$ ถึงความถี่ประมาณ 0.8π แต่จะมีความเพี้ยนมาในช่วงความถี่ต่ำ ๆ และ สูง ๆ

นอกจากสัญญาณซายน์ เรายังสามารถกำเนิดสัญญาณดิจิตอลรายคาบใด ๆ ได้ โดยใช้ตัวกรอง IIR ที่มีฟังก์ชั่นถ่ายโอนดังต่อไปนี้

$$H(z) = \frac{b_0 z^D + b_1 z^{D-1} + \ldots + b_{D-2} z^2 + b_{D-1} z}{z^D - 1} \qquad (12.16)$$

ระบบนี้มีอันดับเท่ากับ D ซึ่งมีโพลอยู่ D ค่าที่มีขนาดเท่ากับหนึ่งเช่นกัน อยู่รอบวงกลมหนึ่งหน่วย ระบบนี้จะให้ผลตอบสนองต่ออิมพัลส์ที่เป็นรายคาบโดยที่ค่าใน 1 คาบ คือ

$$h(n)_{1\ \text{คาบ}} = [\, b_0 \quad b_1 \quad b_2 \ \ldots \ b_{D-1} \,] \qquad (12.17)$$

ขอพิสูจน์โดยการยกตัวอย่าง ระบบที่มี D=4 ซึ่งจะมีฟังก์ชั่นถ่ายโอน ดังนี้

$$H(z) = \frac{b_0 z^4 + b_1 z^3 + b_2 z^2 + b_3 z}{z^4 - 1}$$

หรือ $$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 - z^{-4}} \qquad (12.18)$$

$$Y(z)\,[1-z^{-4}] = X(z)\,[b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}]$$

จะได้สมการผลต่างของระบบ คือ

$$y(n) = y(n-4) + b_0 x(n) + b_1\, x(n-1) + b_2\, x(n-2) + b_3\, x(n-3) \qquad (12.19)$$

ในที่นี้เราจะวิเคราะห์หา h(n) โดยการแทนค่าให้ x(n) ในสมการนี้เป็นอิมพัลส์ $\delta(n)$ และให้ y(n) เป็น h(n) ซึ่งในกรณีนี้จะง่ายกว่าวิธีการแปลง z ผกผัน จะได้สมการที่ 12.19 กลายเป็น

$$h(n) = h(n-4) + b_0\delta(n) + b_1\delta(n-1) + b_2\delta(n-2) + b_3\delta(n-3) \qquad (12.20)$$

ที่เวลา 4 ขั้นแรก จาก n=0 ถึง n=3 สัญญาณ h(n-4) ซึ่งคือ ขาออกของระบบย้อนหลังไป 4 ขั้นเวลาจะยังไม่มีค่า ดังนั้น สัญญาณขาออกเป็นผลจากสัญญาณขาเข้าอย่างเดียว สมการผลต่างใน (12.20) จะกลายเป็น

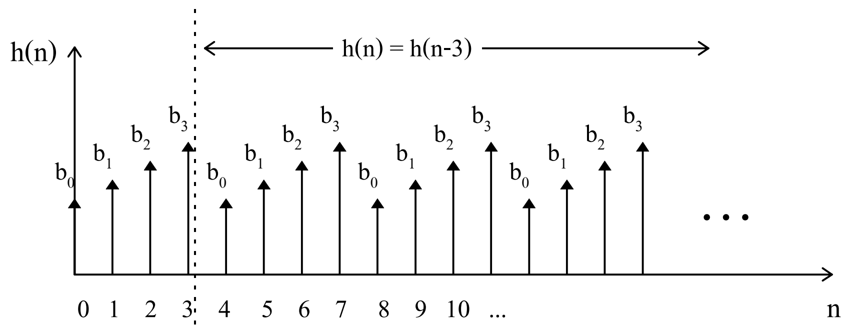
$$h(n) = b_0\delta(n) + b_1\delta(n-1) + b_2\delta(n-2) + b_3\delta(n-3) \qquad (12.21)$$

ซึ่งจะทำให้ได้ผลตอบต่ออิมพัลส์ใน 4 ขั้นเวลาแรกเป็น $[b_0 \;\; b_1 \;\; b_2 \;\; b_3]$

สำหรับในเวลาเริ่มต้นจาก n=4 เป็นต้นไป คราวนี้ค่า $\delta(n)$, $\delta(n-1)$, $\delta(n-2)$, และ $\delta(n-3)$ จะกลายเป็นศูนย์แทน ส่วนค่า h(n-4) จะไม่เป็นศูนย์ ดังนั้น สมการผลต่างจะกลายเป็น

$$h(n) = h(n-4) \qquad (12.22)$$

นั่นคือ ผลตอบสนองต่ออิมพัลส์จะซ้ำกับค่าเดิมของมันก่อนหน้านั้น 4 ค่า คือ h(4)=h(0), h(5)=h(1), h(6)=h(2), ... เป็นเช่นนี้ไปเรื่อย ๆ จนถึงเวลาอนันต์ ดังนั้น จะเห็นได้ว่า h(n) จะเกิดเป็นคาบขึ้นหลังเวลา n=4 ดังแสดงในรูปที่ 12.12



รูปที่ 12.12 *การเกิดคาบของ h(n) ของระบบตามสมการที่ 12.18*

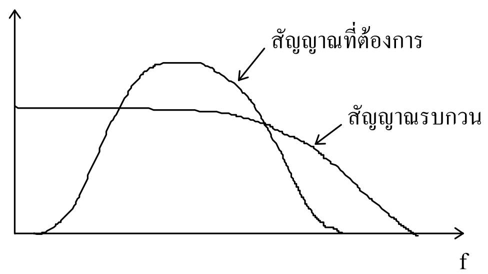
ตัวกรองปรับตัวได้ (Adaptive Filters)[2], [10]

เราได้เห็นมาแล้วว่า ตัวกรองความถี่ทั่ว ๆ ไป เป็นตัวกรองแบบมีสัมประสิทธิ์คงที่ หรือ เป็นแบบไม่แปรตามเวลา สำหรับตัวกรองแบบปรับตัวได้จะเป็นตัวกรองแบบที่มีสัมประสิทธิ์ปรับเปลี่ยนตลอดเวลา โดยอาศัยเงื่อนไขทางสถิติของสัญญาณ และของโมเดลของสิ่งแวดล้อมที่สร้างขึ้นในการคำนวณหาค่าสัมประสิทธิ์ ตัวกรองแบบปรับตัวได้นี้ทำให้ขอบเขตของการประยุกต์ใช้การประมวลผลสัญญาณดิจิตอลขยายวงออกไปกว้างมาก ขอยกตัวอย่าง ภาวะที่เราไม่สามารถใช้ตัวกรองดิจิตอลธรรมดาได้ หรือใช้ได้แต่จะให้ผลที่ไม่ดีนัก ซึ่งจะให้ผลที่ดีขึ้นได้ถ้าใช้ตัวกรองแบบปรับตัวได้ เช่น

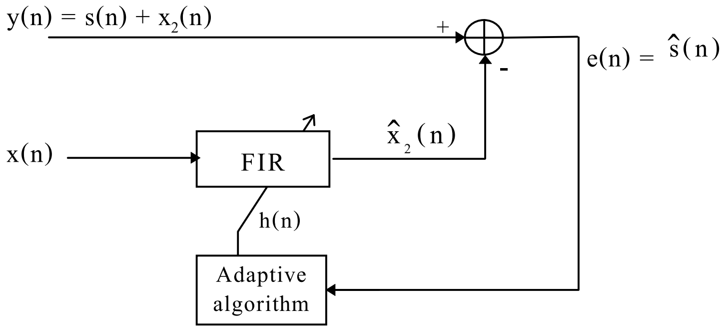
1. เมื่อสัญญาณรบกวน และสัญญาณที่เราต้องการอยู่ในย่านความถี่เดียวกัน เช่นในรูปที่ 12.13 ซึ่งก็ค่อนข้างชัดเจนว่าถ้าใช้ตัวกรองความถี่แบบธรรมดากรอง และต้องการให้สัญญาณรบกวนส่วนใหญ่หมดไป จะทำให้พลังงานส่วนใหญ่สัญญาณที่ต้องการสูญหายไปด้วย

2. เมื่อไม่รู้ลักษณะที่แน่นอนของสัญญาณรบกวน เช่น ไม่รู้ว่าสัญญาณรบกวนอยู่ในย่านความถี่ใด หรือไม่รู้ว่าสัญญาณรบกวนเข้ามาในทิศทางใดในกรณีของสายอากาศแบบอะเรย์ เป็นต้น

3. เมื่อสัญญาณที่ต้องการมีความผิดเพี้ยนที่ไม่รู้ลักษณะที่แน่นอน เช่น กรณีของโมเดม หรือโทรศัพท์มือถือ ที่มีการส่งสัญญาณผ่านช่องสัญญาณ สัญญาณที่ตัวรับได้รับจะถูกทำให้ผิดเพี้ยนไปด้วยฟังก์ชั่นถ่ายโอนของช่องสัญญาณที่ไม่รู้ค่าแน่นอน และอาจมีค่าแปรตามเวลาด้วย ซึ่งเราต้องการใช้ตัวกรองเพื่อกรองเอาความผิดเพี้ยนนี้ออกไป



***รูปที่ 12.13** ตัวอย่างของสเปกตรัมของสัญญาณรบกวนที่อยู่ในย่านเดียวกับสัญญาณที่ต้องการ*



***รูปที่ 12.14** โครงสร้างพื้นฐานของตัวกรองแบบปรับตัวได้*

รูปที่ 12.14 แสดงโครงสร้างพื้นฐานของตัวกรองแบบปรับตัวได้ ซึ่งประกอบด้วยตัวกรองแบบ FIR ซึ่งมีค่าสัมประสิทธิ์ปรับเปลี่ยนได้ โดยรับค่าสัมประสิทธิ์มาจากการประมวลผลของอัลกอริธึมในการปรับตัว (adaptive algorithm) ซึ่งก็มีด้วยกันหลากหลายวิธีในการคำนวณหาค่าสัมประสิทธิ์นี้ การศึกษาถึงอัลกอริธึมเหล่านี้ รวมอยู่ในสาขาของการประมวลผลสัญญาณดิจิตอลขั้นสูง ซึ่งบางครั้งก็เรียกว่า การประมวลผลสัญญาณทางสถิติ (statistical signal processing) หรือ การประมวลผลสัญญาณแบบปรับตัวได้ (adaptive signal processing) หรือ ทฤษฎีการประมาณค่า (estimation theory)

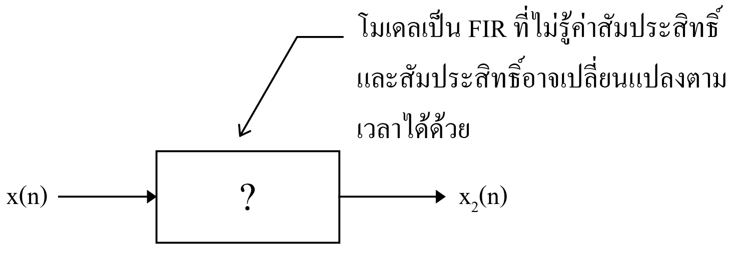
สำหรับการทำความเข้าใจโดยละเอียดถึงการทำงานของอัลกอริธึมในการปรับตัวต่าง ๆ นั้น จำเป็นที่ผู้ศึกษาต้องมีความเข้าใจในทฤษฎีทางสถิติ ทฤษฎีเกี่ยวกับสัญญาณแรนดอม (random signal) และทฤษฎีคณิตศาสตร์ของเมตริกซ์ และพีชคณิตเชิงเส้น ซึ่งเกินขอบเขตที่หนังสือเล่มนี้จะกล่าวถึงได้ ในที่นี้จะได้อธิบายในเชิงแนะนำถึง อัลกอริธึมในการปรับตัวชนิดหนึ่งที่ชื่อว่า วิธี LMS (Least Mean Square) ซึ่งเป็นอัลกอริธึมที่ง่ายที่สุดเมื่อเทียบกับตัวอื่น ๆ แต่ก็มีการใช้งานอย่างกว้างขวาง ซึ่งหวังว่าคงจะพอให้เป็นแนวทาง และเป็นแรงกระตุ้นให้ผู้ที่สนใจได้ศึกษาต่อไปได้

สถานการณ์แวดล้อมของโครงสร้างพื้นฐานของตัวกรองปรับตัวได้ในรูปที่ 12.14 เป็นดังนี้

1. เราสามารถวัดสัญญาณมาได้สองสัญญาณ คือ x(n) และ y(n) โดยที่ y(n) เป็นสัญญาณที่ผสมกันระหว่าง s(n) กับ $x_2(n)$ วัตถุประสงค์ของตัวกรองก็คือ <u>ต้องการแยก s(n) ออกจาก $x_2(n)$</u> โดยไม่จำเป็นต้องรู้ย่านความถี่ หรือลักษณะทางสถิติของสัญญาณทั้งสองล่วงหน้า

2. เงื่อนไขที่จำเป็น คือ s(n) นั้นต้องไม่มีความสัมพันธ์กับ $x_2(n)$ ในทางสถิติ (statistical independent) หรือสัมพันธ์กันน้อยมาก เช่น เป็นสัญญาณเสียงที่มีแหล่งกำเนิด 2 แหล่งที่ไม่เกี่ยวข้องกัน ส่วน x(n) ซึ่งเป็นสัญญาณอีกสัญญาณหนึ่งที่วัดได้นั้น ต้องมีความสัมพันธ์กับ $x_2(n)$

3. เราโมเดลความสัมพันธ์ระหว่าง x(n) กับ $x_2(n)$ ว่า สัมพันธ์กันด้วยฟังก์ชั่นถ่ายโอนของตัวกรอง FIR ตัวหนึ่งที่ไม่รู้ค่าสัมประสิทธิ์ ไม่รู้ค่าอันดับ และสัมประสิทธิ์อาจเปลี่ยนแปลงตามเวลาได้อย่างช้า ๆ โดยที่ x(n) เป็นสัญญาณขาเข้าของตัวกรองนี้ และ $x_2(n)$ เป็นขาออกดังแสดงในรูปที่ 12.15 โมเดลนี้นอกจากง่ายต่อการคำนวณแล้ว ยังเป็นโมเดลที่ดีสำหรับสิ่งแวดล้อมหลาย ๆ อย่าง เช่น ใช้เป็นโมเดลของช่องสัญญาณต่าง ๆ ที่เสียง หรือคลื่นแม่เหล็กไฟฟ้าเดินทางผ่าน ฯลฯ



รูปที่ 12.15 *โมเดลความสัมพันธ์ระหว่างสัญญาณ x(n) กับ $x_2(n)$*

4. เราจะนำ x(n) ผ่านตัวกรอง FIR ที่สร้างขึ้นซึ่งมีสัมประสิทธิ์ h(n) และอันดับเท่ากับ N ให้สัญญาณขาออกของตัวกรอง คือ $\hat{x}_2(n)$ ถ้าหากว่า เราสามารถปรับสัมประสิทธิ์ h(n) ใกล้เคียงกับสัมประสิทธิ์ที่เป็นโมเดลระหว่าง x(n) กับ $x_2(n)$ ได้ $\hat{x}_2(n)$ ก็จะเป็นสัญญาณที่ใกล้เคียงกับ $x_2(n)$ และเมื่อนำ $\hat{x}_2(n)$ ลบออกจาก y(n) ตามรูปที่ 12.14 สัญญาณที่เหลืออยู่ซึ่งจะเรียกว่า สัญญาณความคลาดเคลื่อน e(n) ก็จะมีค่าใกล้เคียงกับ s(n) นั่นคือ เราสามารถแยก $x_2(n)$ ออกจาก s(n) ได้สำเร็จ

ตัวที่จะคำนวณหาสัมประสิทธิ์ h(n) ก็คือ อัลกอริธึมปรับตัวได้ ซึ่งรับค่า e(n) และ x(n) จากนั้นคำนวณ h(n) ป้อนให้กับตัวกรอง FIR และจะปรับค่า h(n) ไปเรื่อย ๆ ทุก ๆ ขั้นเวลา จนกระทั่ง h(n) ลู่เข้าสู่ค่าที่ถูกต้อง

อัลกอริธึม LMS และเงื่อนไขการทำงานของมัน

เมื่อพิจารณาสัญญาณความคลาดเคลื่อน e(n) จากรูปที่ 12.14 จะได้ว่า

$$e(n) = y(n) - \hat{x}_2(n) = s(n) + x_2(n) - \hat{x}_2(n) \qquad (12.23)$$

พิจารณาว่าทุกสัญญาณเป็นสัญญาณแรนดอม เมื่อหาค่าคาดหวัง (หรือค่าเฉลี่ยทางสถิติ) ของกำลังสองของทั้งสมการจะได้

$$E[e^2(n)] = E[(s(n) + x_2(n) - \hat{x}_2(n))^2] \qquad (12.24)$$

ขอละตัวชี้เวลา n เพื่อให้สมการง่ายต่อการมอง ดังนี้

$$E[e^2] = E[(s + x_2 - \hat{x}_2)^2] \qquad (12.25)$$

กระจายเทอมกำลังสองด้านขวามือของสมการออกมาจะได้

$$E[e^2] = E[s^2 + 2s(x_2 - \hat{x}_2) + (x_2 - \hat{x}_2)^2]$$
$$E[e^2] = E[s^2] + 2E[s x_2] - 2E[s\hat{x}_2] + E[(x_2 - \hat{x}_2)^2] \qquad (12.26)$$

เนื่องจาก s กับ x_2 และ s กับ $\hat{x}_2$ ไม่มีความสัมพันธ์กัน จากทฤษฎีทางสถิติที่ว่าค่าคาดหวังของผลคูณของสองสัญญาณที่ไม่สัมพันธ์กันจะเท่ากับศูนย์ จะได้ว่า เทอมที่สอง และสามของฝั่งขวาของสมการเป็นศูนย์ ดังนั้น สมการที่ 12.26 จะลดเหลือเพียง

$$E[e^2] = E[s^2] + E[(x_2 - \hat{x}_2)^2] \tag{12.27}$$

ใช้ความรู้ที่ว่า ค่าคาดหวังของกำลังสองของสัญญาณ ก็คือ ค่ากำลังเฉลี่ยของสัญญาณ สมการนี้จะตีความได้ว่า กำลังเฉลี่ยของสัญญาณ e(n) เท่ากับกำลังเฉลี่ยของสัญญาณ s(n) บวกกับกำลังเฉลี่ยของผลต่างระหว่าง $x_2(n)$ กับ $\hat{x}_2(n)$ เราต้องการให้เทอมสุดท้ายนี้มีค่าน้อยที่สุดเท่าที่จะทำได้ เพราะนั่นจะหมายถึงว่า สัญญาณ $x_2(n)$ กับ $\hat{x}_2(n)$ มีความแตกต่างกันน้อยที่สุดเท่าที่จะทำได้ และก็จะทำให้ตัวกรองทำงานได้ตามที่ต้องการ

สัญญาณ $\hat{x}_2(n)$ นั้นขึ้นกับค่า h(n) เพราะ เป็นขาออกของตัวกรอง FIR โดยสมมติว่า อันดับของตัวกรองเท่ากับ N จะได้

$$\hat{x}_2(n) = h_0(n)x(n) + h_1(n)x(n-1) + \dots + h_{N-1}(n)x(n-N+1) \tag{12.28}$$

ในการวิเคราะห์ทางคณิตศาสตร์ของตัวกรองปรับตัวได้ นิยมเขียนสัญญาณให้อยู่ในรูปของเวคเตอร์ และเมตริกซ์เพื่อให้มองดูกระชับ ดังนั้น ขอนิยามเวคเตอร์ของสัมประสิทธิ์ที่เวลา n คือ $\vec{h}(n)$ และเวคเตอร์ของสัญญาณขาเข้าที่เวลา n คือ $\vec{x}(n)$ ซึ่งมีค่าดังต่อไปนี้

$$\vec{h}(n) = \begin{bmatrix} h_0(n) \\ h_1(n) \\ \vdots \\ h_{N-1}(n) \end{bmatrix} \quad \text{และ} \quad \vec{x}(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-N+1) \end{bmatrix} \tag{12.29}$$

ทั้งสองเวคเตอร์มีขนาด N แถว 1 คอลัมน์ สังเกตว่า $\vec{x}(n)$ คือ เวคเตอร์ที่ประกอบขึ้นจากสัญญาณขาเข้าปัจจุบัน และสัญญาณขาเข้าที่ย้อนหลังไป N ตัว เมื่อเราได้นิยามดังนี้แล้ว จะสามารถเขียนสมการที่ 12.29 ได้ใหม่ในรูปแบบการคูณกันของเวคเตอร์ คือ

$$\hat{x}_2(n) = \vec{h}(n)^T \vec{x}(n) \tag{12.30}$$

สำหรับสัญญาณ e(n) ก็ขึ้นกับ h(n) เพราะเป็นผลต่างของ y(n) กับ $\hat{x}_2(n)$ ส่วนสัญญาณ s(n) นั้นเป็นขาเข้าที่ไม่ขึ้นกับ h(n) ดังนั้น สมการที่ 12.27 มีเฉพาะเทอม $E[e^2]$ และเทอม $E[(x_2 - \hat{x}_2)^2]$ เท่านั้นที่ขึ้นกับค่า h(n) ดังนั้น ถ้ารู้ค่าพารามิเตอร์ต่าง ๆ ของระบบทั้งหมด เราจะสามารถแก้สมการหาค่า h(n) ที่จะทำให้ **$E[(x_2 - \hat{x}_2)^2]$ มีค่าต่ำสุด**เท่าที่จะทำได้ ซึ่งก็จะคือจุดเดียวกับที่ค่า **$E[e^2]$ มีค่าต่ำสุด**เท่าที่จะทำได้ เขียนเป็นสมการได้ว่า

$$E[\, e^2 \,]_{\text{ต่ำสุดจากการปรับค่า } h(n)} = E[\, s^2 \,] + E[\, (x_2 - \hat{x}_2)^2 \,]_{\text{ต่ำสุดจากการปรับค่า } h(n)} \qquad (12.31)$$

การแก้สมการนี้เพื่อหาค่า h(n) โดยใช้เงื่อนไขว่า เป็น h(n) ที่ทำให้ค่า $E[\, e^2 \,]$ ต่ำสุด จะง่ายกว่าใช้เงื่อนไขว่า ให้ค่า $E[(x_2 - \hat{x}_2)^2]$ ต่ำสุด ซึ่งผลตอบของการแก้สมการนี้เป็นที่รู้จักกันโดยทั่วไปในชื่อว่า <u>ตัวกรองวีนเนอร์</u> (Wiener Filter) ในที่นี้จะขอไม่กล่าวถึงการแก้สมการนี้ และผลตอบที่ได้ แต่ขอให้ผู้อ่านมีความเข้าใจว่า ตัวกรองวีนเนอร์นี้จะใช้ค่า h(n) ที่ดีที่สุดทุก ๆ ค่าเวลา n แต่ปัญหาของตัวกรองวีนเนอร์ก็คือ ค่า h(n) นี้มีการคำนวณที่ยุ่งยาก และจำเป็นต้องรู้ค่าพารามิเตอร์ทางสถิติของสัญญาณ x(n) และ y(n) ด้วย ซึ่งในทางปฏิบัติมักจะไม่ทราบค่าแน่นอน ด้วยเหตุผลนี้ ทำให้ตัวกรองวีนเนอร์แทบจะไม่มีที่นำมาใช้งานได้ในทางปฏิบัติ

อย่างไรก็ตาม ตัวกรองวีนเนอร์นี้มีประโยชน์มากในแง่ทฤษฎี เพราะมันคือเป้าหมายที่ตัวกรองปรับตัวได้แบบอื่น ๆ ต้องการไปถึง อัลกอริธึมปรับตัวได้ที่เป็นที่นิยมได้แก่ LMS, RLS, และ Kalman ล้วนแล้วแต่มีผลตอบที่สามารถพิสูจน์ได้ว่า ลู่เข้าสู่ผลตอบของตัวกรองวีนเนอร์ทั้งสิ้น กล่าวคือ อัลกอริธึมเหล่านี้ไม่ได้ให้ผลตอบที่เหมือนกับผลตอบของตัวกรองวีนเนอร์ทุก ๆ ขั้นเวลา แต่ถ้าถูกใช้ในสภาวะที่ลักษณะทางสถิติของสัญญาณขาเข้าเปลี่ยนแปลงไม่เร็วนัก เมื่อเวลาผ่านไปสักระยะหนึ่งมันจะสามารถให้ผลตอบที่ลู่เข้าสู่ผลตอบของตัวกรองวีนเนอร์ได้

ลักษณะสำคัญของการคำนวณสัมประสิทธิ์ด้วยอัลกอริธึมเหล่านี้ ที่ปรับปรุงจากตัวกรองวีนเนอร์ คือ

1. ไม่จำเป็นต้องรู้ค่าพารามิเตอร์ทางสถิติของสัญญาณขาเข้า
2. มีการคำนวณที่สามารถปรับไปใช้ในการคำนวณแบบเวลาจริงได้

ขอยกตัวอย่างอัลกอริธึมแบบ LMS หรือ Least Mean Square ที่ถูกคิดค้นขึ้นเมื่อปี ค.ศ. 1960 โดยวิโดรว์ และฮอฟฟ์ (Widrow and Hoff) มีที่มาโดยตรงจากตัวกรองวีนเนอร์ และการแก้สมการโดยวิธี steepest descent ซึ่งทำโดยการสมมติผลตอบ (ค่าสัมประสิทธิ์) เริ่มต้นขึ้นมาเป็นค่าอะไรก็ได้ จากนั้น ทุก ๆ ขั้นเวลา เมื่อได้สัญญาณขาเข้าใหม่ ก็จะคำนวณหาทิศทางที่จะเปลี่ยนค่าสัมประสิทธิ์นั้นให้เข้าใกล้กับผลตอบที่ถูกต้อง แล้วเปลี่ยนค่าสัมประสิทธิ์ตามทิศทางนั้น ซึ่งวิธีนี้ ปรากฏว่าทำให้การคำนวณง่ายกว่าการคำนวณหาสัมประสิทธิ์โดยตรงมาก ซึ่งทิศทางที่ดีที่สุดที่จะใช้เปลี่ยนค่าสัมประสิทธิ์ตามวิธี LMS นี้ คือ

$$\vec{\Delta} = 2\, e(n)\, \vec{x}(n) \qquad (12.32)$$

เราจะหาค่าสัมประสิทธิ์ใหม่สำหรับเวลา n+1 จากค่าสัมประสิทธิ์ปัจจุบัน และเวคเตอร์ทิศทางได้ด้วยการคำนวณอย่างตรงไปตรงมา ดังนี้

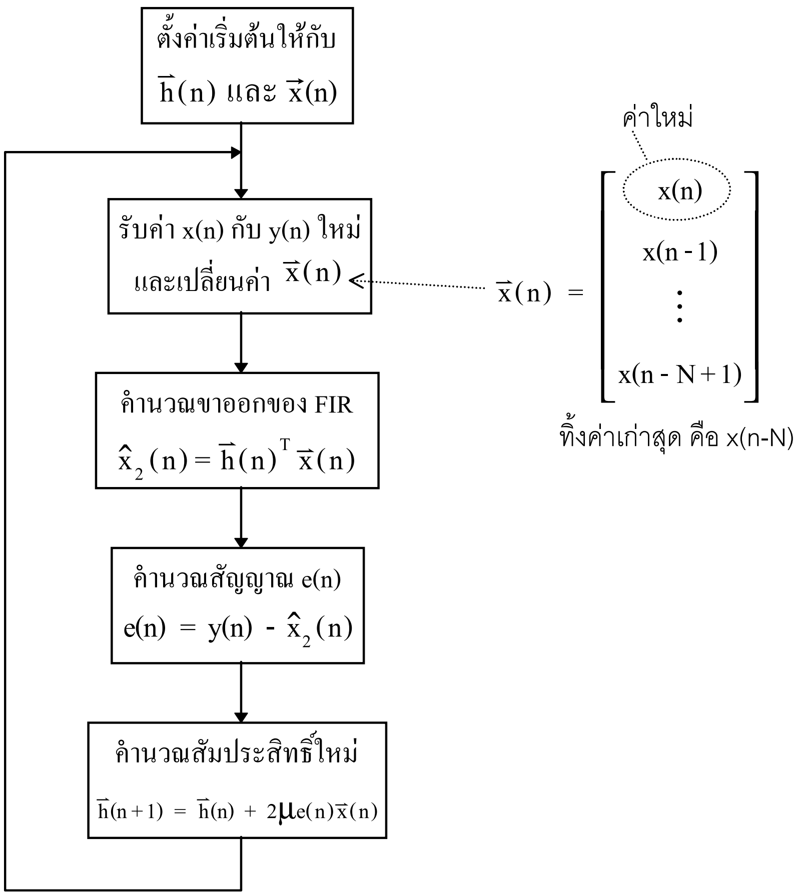
$$\vec{h}(n+1) = \vec{h}(n) + \mu\vec{\Delta} \tag{12.33}$$

โดยที่ค่า μ คือ ตัวที่ควบคุมว่า สัมประสิทธิ์จะมีค่าเปลี่ยนแปลงตามเวคเตอร์ทิศทางเร็วมากน้อยแค่ไหน โดยทั่วไป μ มีค่าประมาณ 0.001 ถึง 0.1 ถ้าค่า μ มีค่าน้อยเกินไปก็จะส่งผลให้สัมประสิทธิ์ลู่เข้าสู่ค่าที่ถูกต้องช้า แต่ถ้าค่า μ มีค่ามากเกินไปก็อาจทำให้สัมประสิทธิ์ไม่สามารถลู่เข้าสู่ค่าที่ถูกต้องได้ หรือถึงแม้จะลู่ได้แต่ก็อาจจะเปลี่ยนแปลงไปมา ไม่คงตัวอยู่ที่ภาวะสมดุลที่ดีที่สุด ดังนั้นการเลือกค่า μ ที่เหมาะสมมีผลต่อประสิทธิผลการทำงานของระบบมาก

เมื่อรวมเอาสมการที่ 12.32 และ 12.33 เข้าด้วยกัน ก็จะได้สมการสำหรับคำนวณหาค่าสัมประสิทธิ์ที่เวลาใด ๆ คือ

$$\vec{h}(n+1) = \vec{h}(n) + 2\mu e(n)\vec{x}(n) \tag{12.34}$$

เราสามารถสรุปกระบวนการทำงานของตัวกรองปรับตัวได้ ที่ใช้ LMS เป็นตัวคำนวณค่าสัมประสิทธิ์ ได้ดังรูปที่ 12.16



รูปที่ 12.16 *แผนภาพแสดงการทำงานของตัวกรองปรับตัวได้*

โปรแกรมที่ 12.1 เป็นฟังก์ชั่นที่ใช้คำนวณตัวกรองปรับตัวได้ โดยจะรับค่าของเวคเตอร์สัญญาณ x(n), สัญญาณ y(n), ค่า μ , และค่าอันดับ จากนั้นประมวลผลตามแผนภาพในรูปที่ 12.16 แล้วคืนค่าสัมประสิทธิ์ และเวคเตอร์ของสัญญาณ e(n) โดยสัมประสิทธิ์จะคืนค่าเป็นเมตริกซ์ที่มีจำนวนแถวเท่ากับขนาดของเวคเตอร์ $\vec{x}(n)$ และมีจำนวนคอลัมน์เท่ากับขนาดของเวคเตอร์ $\vec{h}(n)$ โดยที่แต่ละแถวเป็นค่าสัมประสิทธิ์ $\vec{h}(n)$ ที่แต่ละเวลา ตั้งแต่เวลาเริ่มต้นจนถึงค่าเวลาสุดท้ายของการประมวลผล ซึ่งทำให้เราสามารถนำไปวิเคราะห์ต่อไปได้ว่า สัมประสิทธิ์ที่หาได้มีการเปลี่ยนแปลงตามเวลาอย่างไร

สำหรับโปรแกรมที่ 12.2 เป็นตัวอย่างของการเรียกใช้ฟังก์ชั่น lms (โปรแกรมที่ 12.1) โดยสมมติให้สัญญาณ x(n) และ s(n) เป็นสัญญาณแบบ white gaussian ซึ่งสร้างโดยใช้ฟังก์ชั่น randn ใน MATLAB จากนั้นสมมติค่าสัมประสิทธิ์ของสิ่งแวดล้อม คือ เวคเตอร์ h_env แล้วสร้างสัญญาณ x_2(n) ด้วยการนำสัญญาณ x(n) ผ่านตัวกรอง FIR ที่มีค่าสัมประสิทธิ์นี้ เราจะสมมติว่าเราวัดได้เฉพาะสัญญาณ y(n) และ x(n) โดยผ่านสัญญาณทั้งสองให้กับฟังก์ชั่น LMS เพื่อแยกเอา s(n) ออกมาจาก y(n) ค่าที่ฟังก์ชั่นคืนกลับมา คือ เมตริกซ์ h และเวคเตอร์ err ซึ่งถ้าตัวกรองทำหน้าที่ของมันได้สำเร็จจะได้ว่า h มีแถวสุดท้ายใกล้เคียงกับค่าสัมประสิทธิ์ของสิ่งแวดล้อม และ err ก็มีค่าใกล้เคียงกับ s(n)

วิธีที่จะวิเคราะห์ให้เห็นชัดจะต้องดูความแตกต่างระหว่างสัมประสิทธิ์ที่ LMS หาได้ที่เวลาต่าง ๆ เทียบกับสัมประสิทธิ์ของสิ่งแวดล้อม เราจะนิยามค่าความแตกต่างนี้เป็นค่าพารามิเตอร์ตัวเดียวที่แต่ละเวลา n เรียกค่านี้ว่า ค่านอร์มของความแตกต่างสัมประสิทธิ์ (coefficient error norm) ซึ่งเป็นค่าผลรวมของกำลังสองความแตกต่างของสัมประสิทธิ์ นอร์แมลไลซ์ด้วยผลรวมกำลังสองของสัมประสิทธิ์สิ่งแวดล้อม ดังนี้

$$\text{h\_norm}(n) = \frac{\sum_{i=0}^{\text{order-1}} (h_i(n) - h\_env_i(n))^2}{\sum_{i=0}^{\text{order-1}} h\_env_i(n)^2} \qquad (12.35)$$

```
function [h,err]=lms(x,y,mu,order);
h(1,:)=zeros(1,order);
xvec=zeros(1,order);
for i=1:length(x)
   xvec(2:order) = xvec(1:order-1);
   xvec(1)=x(i);                    % รับค่า x(n) ใหม่ใส่ในเวคเตอร์
   x2_est = h(i,:)*xvec';
   err(i) = y(i) - x2_est;
   h(i+1,:) = h(i,:) + 2*mu*xvec*err(i);
end
h(i+1,:)=[];
```

โปรแกรมที่ 12.1 *ฟังก์ชั่น lms.m สำหรับคำนวณตัวกรองปรับตัวได้แบบ FIR*

```
mu=0.005;
order=9;
h_env=[0.1054,0.1839,0.2596,0.2500,0.1558,0.0613,0.0151,0.005,0.002];
                         % สมมติ w นี้เป็นสัมประสิทธิ์ของสิ่งแวดล้อม
N=500;                   % จำนวนจุดที่จะจำลองการทำงาน
x=randn(1,N);            % สมมติ x(n) เป็นสัญญาณประเภท white gaussian
x2=filter(h_env,1,x);    % x(n) ผ่านสิ่งแวดล้อมซึ่งเป็น FIR มีสัมประสิทธิ์ h ได้ขาออกเป็น x2(n)
s = 0.5*randn(1,N); % สมมติ s(n) เป็นสัญญาณประเภท white gaussian เช่นกัน
y = s + x2;
[h,err] = lms(x,y,mu,order);

hend = h(N,:);
h_norm = (h-ones(N,1)*h_env).^2;
h_norm = sum(h_norm');
hh = h_env*h_env';
h_norm = h_norm/hh;
plot(h_norm)
```

โปรแกรมที่ 12.2 *โปรแกรม runlms.m ตัวอย่างการเรียกใช้งานฟังก์ชั่น lms.m*

ส่วนสุดท้ายของโปรแกรมที่ 12.2 จะคำนวณหาค่า h_norm นี้ แล้ววาดออกมา ซึ่งจะทำให้เห็นได้ว่า ค่านี้มีการเปลี่ยนแปลงตามเวลาอย่างไร ถ้าค่า h_norm ลู่เข้าสู่ศูนย์ก็แสดงว่า สัมประสิทธิ์ที่ LMS หาได้จะลู่เข้าสู่ค่าสัมประสิทธิ์ของสิ่งแวดล้อม นั่นคือ ตัวกรองปรับตัวได้สามารถทำงานสำเร็จได้ตามที่เราต้องการ

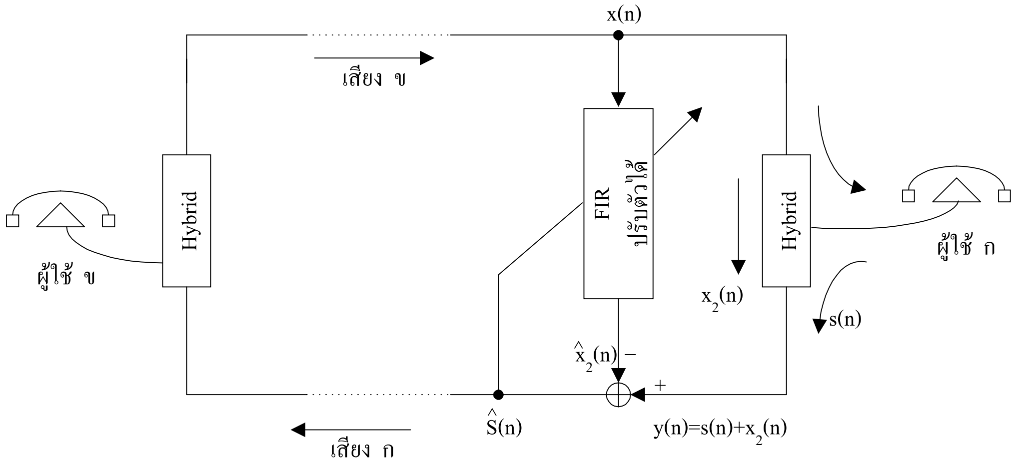
การหักล้างเสียงสะท้อน (Echo Cancellation) [2], [10]

การประยุกต์ใช้งานที่สำคัญอันหนึ่งของตัวกรองปรับตัวได้ คือ การหักล้างเสียงสะท้อนซึ่งมีที่ใช้งานอยู่ในหลายสถานการณ์ ในที่นี้ขอยกตัวอย่างสถานการณ์ที่สำคัญ 2 สถานการณ์ด้วยกัน คือ

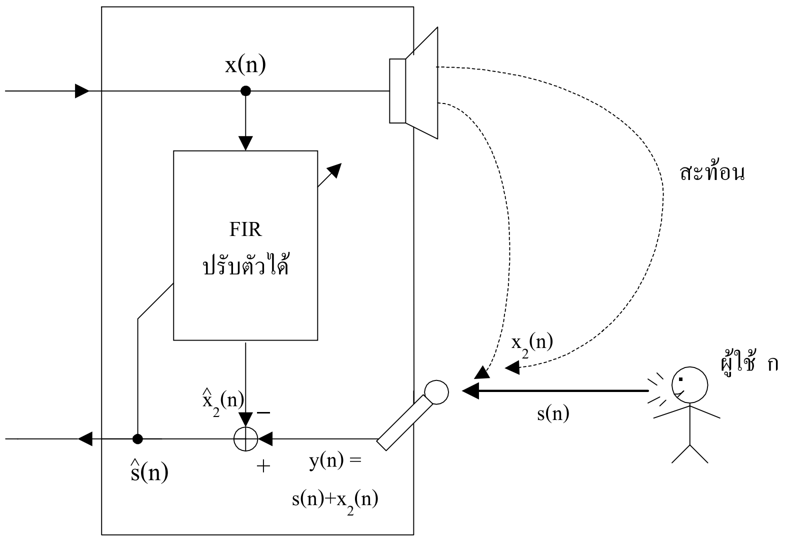
1. เสียงสะท้อนที่เกิดขึ้นในระบบโทรศัพท์ทางไกล เสียงสะท้อนนี้เกิดจากวงจรหม้อแปลงไฮบริด (hybrid transformer) ที่อยู่ที่ชุมสายโทรศัพท์ ซึ่งทำหน้าที่แยกสัญญาณเสียงขาไป และกลับซึ่งอยู่ในสายคู่เดียวกัน (สายจากชุมสายไปยังบ้านของผู้ใช้ หรือ local loop) เพื่อส่งไปกับสาย 2 คู่ที่เป็นสายเชื่อมชุมสาย (trunk) สำหรับโทรทางไกล สาเหตุที่ต้องแยกช่องสัญญาณส่งและรับออกจากกันเนื่องจากการส่งระยะไกลต้องการการขยายสัญญาณ หรือไม่ก็ต้องการส่งเป็นแบบดิจิตอลซึ่งต้องกระทำกับแต่ละช่องสัญญาณแยกกัน ในรูปที่ 12.17 แสดงระบบกรองเสียงสะท้อนที่ฝั่งของผู้ใช้ ก เนื่องจากวงจรไฮบริดมีความไม่เป็นอุดมคติ จึงทำให้เสียงของผู้ใช้ ข บางส่วนหลุดลอดผ่านวงจรกลับไปสู่ผู้ใช้ ข ซึ่งถ้าหากไม่กรองทิ้ง ผู้ใช้ ข ก็จะได้ยินเสียงของเขาเองสะท้อนกลับมา ทำให้เกิดความรำคาญได้

2. เสียงสะท้อนที่เกิดขึ้นในระบบโทรศัพท์แบบไม่ยกหู (speakerphone) หรืออุปกรณ์ประเภทประชุมผ่านวีดีโอทางไกล (video conference) การสื่อสารเหล่านี้ใช้อุปกรณ์ที่มีไมโครโฟนรับเสียงของผู้พูดซึ่งเป็นไมโครโฟนที่รับเสียงได้กว้างโดยผู้ใช้ไม่ต้องพูดจ่ออยู่ที่ตำแหน่งของ

ไมโครโฟน ขณะเดียวกันก็มีลำโพงเพื่อส่งเสียงของผู้พูดอีกฝั่งหนึ่งให้ดังออกมา ดังนั้น เสียงที่ดังออกมาจากลำโพงก็สามารถเล็ดลอดไปเข้าไมโครโฟนได้ และบางส่วนก็สะท้อนสิ่งต่าง ๆ ก่อนเข้าไมโครโฟน ดังแสดงในรูปที่ 12.18



รูปที่ 12.17 *การกรองเสียงสะท้อนในระบบโทรศัพท์ทางไกล*



รูปที่ 12.18 *การกรองเสียงสะท้อนในโทรศัพท์แบบไม่ยกหู*

ในทั้งสองสถานการณ์ เราสามารถโมเดลเสียงที่สะท้อนกลับ ($x_2(n)$) ได้ว่า เป็นเสียงของผู้ใช้อีกฝั่งหนึ่ง ($x(n)$) ที่ผ่านตัวกรอง FIR ที่ไม่รู้ค่าสัมประสิทธิ์ตัวหนึ่ง จากนั้นก็ใช้ตัวกรอง FIR ปรับตัวได้ที่ได้กล่าวถึงในหัวข้อที่แล้ว เพื่อหักล้างเสียงสะท้อนนี้ทิ้งเสีย

สัญญาณขาเข้าของตัวกรองปรับตัวได้ คือ 1) สัญญาณที่มาจากผู้ใช้ ข ($x(n)$) และ 2) สัญญาณที่มาจากผู้ใช้ ที่ถูกผสมกับเสียงที่สะท้อนกลับ ($y(n) = s(n) + x_2(n)$) ตัวกรองปรับตัวได้จะคำนวณหา $\hat{x}_2(n)$ เพื่อมาหักล้างกับ $x_2(n)$ ดังนั้น จะได้สัญญาณขาออก คือ $e(n)$ มีค่าประมาณเท่ากับ $s(n)$ ซึ่งคือเสียงของผู้ใช้ ก อย่างเดียวเพื่อส่งไปหาผู้ใช้ ข

ในทางปฏิบัติ ยังมีเทคนิคเพิ่มเติมอีกจากการใช้ LMS ปกติอีก เพื่อทำให้ได้ผลดียิ่งขึ้น ได้แก่

1) การใช้อัลกอริธึม Normalized LMS แทน LMS ปกติ เพื่อชดเชยผลของการที่สัญญาณเสียงมีกำลังงานที่เปลี่ยนแปลงตามเวลาซึ่งไม่เหมาะกับการใช้ LMS ปกติ

2) การใช้เทคนิคให้ LMS ทำงานคำนวณหาค่าสัมประสิทธิ์เฉพาะเมื่อเวลาไม่มีเสียงพูดมาจากผู้ใช้ ก และให้ LMS หยุดทำงานโดยค้างค่าสัมประสิทธิ์เดิมไว้ถ้ามีเสียงพูดจากผู้ใช้ ก มา เทคนิคนี้ช่วยป้องกันการที่เสียงบางส่วนของผู้ใช้ ก จะถูกหักล้างไปด้วยในการใช้ LMS ปกติ

ภาคผนวก ก

การใช้งาน Matlab

ในส่วนนี้จะได้แนะนำการใช้ Matlab สำหรับผู้เริ่มต้น โดยคำสั่งต่าง ๆ ที่ใช้นี้ได้ทดลองกับ Matlab for Windows เวอร์ชั่น 4.2 ผู้ที่เคยมีประสบการณ์บ้างกับการเขียนโปรแกรมภาษาสูงต่าง ๆ เช่น ปาสคาล หรือ ซี จะพบว่า การเขียนโปรแกรมใน Matlab ทำได้ง่ายมาก

บรรทัดที่มีเครื่องหมาย >> (ซึ่งเป็นตัวแทนเครื่องหมาย prompt ใน Matlab) หมายถึง คำสั่งที่จะต้องพิมพ์ป้อนให้ Matlab ส่วนบรรทัดที่มีการย่อหน้าลึกเพิ่มเข้าไปอีกเล็กน้อยก็คือผลลัพธ์ที่ Matlab ตอบสนองออกมา

เริ่มรู้จักกับ Matlab ในฐานะเป็นเครื่องคิดเลข

เราเริ่มต้นมอง Matlab อย่างง่าย ๆ ในฐานะเป็นเหมือนเครื่องคิดเลข หรือเป็นโต๊ะทดลองทางคณิตศาสตร์ตัวหนึ่งที่มีฟังก์ชั่น และการกระทำทางคณิตศาสตร์ครบถ้วน สัญลักษณ์สำหรับการกระทำทางคณิตศาสตร์เบื้องต้น ได้แก่ + บวก, - ลบ, * คูณ, / หาร, และ ^ ยกกำลัง สมมติว่าเราต้องการคำนวณ 3(2+4) สามารถพิมพ์เข้าไปได้ทันทีดังนี้

>> 3*(2+4) ซึ่ง Matlab จะตอบออกมาในบรรทัดต่อมาว่า ..

 ans = 18

ซึ่งจะเห็นว่าลักษณะการเขียนประโยคทางคณิตศาสตร์ก็เหมือนกับภาษาสูงทั่ว ๆ ไป โดยจะกระทำในวงเล็บก่อน แล้วค่อยทำข้างนอก ถ้าไม่มีวงเล็บ Matlab จะเรียงลำดับการกระทำดังนี้

1. ยกกำลัง
2. คูณ และ หาร
3. บวก และ ลบ

เช่น ถ้าสั่งว่า 3^2*5/4/2+1 จะมีค่าเท่ากับ $\frac{3^2}{2 \cdot 4} \cdot$ 1 ซึ่งได้ค่าเท่ากับ 6.625 เป็นต้น

สำหรับรูปแบบตัวเลขทางวิทยาศาสตร์ที่มีการคูณด้วยกำลังของสิบ เช่น 3×10^{-11} ก็สามารถป้อนเข้าไปได้โดยใช้ว่า 3e-11

การใช้ตัวแปรใน Matlab

เราสามารถเก็บค่าต่าง ๆ ไว้ในตัวแปรได้โดยใช้เครื่องหมาย = และสามารถเรียกใช้ตัวแปรนั้นต่อไปได้อย่างสะดวก ตัวอย่างเช่น

\>> a=2 เก็บ 2 ไว้ในตัวแปร a

a = 2

\>> b=3*(a+4) เก็บผลลัพธ์ไว้ในตัวแปร b

b = 18

ถ้าเราไม่ต้องการให้ Matlab แสดงค่าตอบสนองออกมา ให้ใส่เครื่องหมายเซมิโคลอน (**;**) ไว้ท้ายคำสั่ง เช่น ถ้าสั่งว่า

\>> a = 2**;** จะให้ผลเหมือนกับการสั่งข้างต้น เพียงแต่ไม่แสดงผลตอบสนองออกมา

เราสามารถตั้งตัวแปรขึ้นมาใช้ได้ไม่จำกัดจำนวน (จำกัดอยู่ที่หน่วยความจำของเครื่องคอมพิวเตอร์) การตั้งชื่อตัวแปรสามารถใช้ตัวอักษร a ถึง z และ A ถึง Z, ตัวเลข 0 ถึง 9, และสัญลักษณ์ _ โดยมีข้อแม้ว่าจะต้องใช้ตัวอักษรอย่างน้อย 1 ตัวนำหน้าเสมอ เช่น a11_79 เป็นต้น

ถ้าต้องการดูว่าขณะนั้นมีตัวแปรอะไรอยู่บ้างให้ใช้คำสั่ง **who** และถ้าต้องการดูว่าตัวแปรหนึ่ง ๆ เก็บค่าอะไรอยู่ ก็ให้พิมพ์ชื่อตัวแปรนั้นแล้วกด enter เช่น

.>> who ขอดูว่ามีตัวแปรอะไรบ้าง

Your variables are:

a b

\>> a ขอดูว่า a เก็บค่าอะไรอยู่

a = 1

ตัวแปรที่สร้างไว้จะไม่หายไปจนกว่าจะปิดโปรแกรม Matlab การลบตัวแปรทิ้งทำได้โดยใช้คำสั่ง **clear** โดยการสั่ง clear เฉย ๆ จะเป็นการลบตัวแปรทั้งหมดทิ้ง ถ้าต้องการลบเฉพาะบางตัวก็ให้สั่ง clear แล้วตามตัวชื่อของตัวแปร เช่น

\>> clear a b จะลบเฉพาะตัวแปร a และ b เท่านั้น

การเรียกคำสั่งเก่ามาใช้ใหม่

คำสั่งเก่าที่เคยพิมพ์แล้ว สามารถเรียกกลับขึ้นมาใหม่ได้โดยไม่ต้องพิมพ์ใหม่ ด้วยการกดปุ่มลูกศรขึ้น และลง เมื่อเจอคำสั่งที่ต้องการแล้วก็สามารถกดปุ่มลูกศรซ้ายและขวาเพื่อเข้าไปแก้ไขได้ลักษณะเช่นเดียวกับการใช้ doskey ใน DOS

เมตริกซ์ และเวคเตอร์

การป้อนค่าเมตริกซ์ใน Matlab สามารถกระทำได้ง่ายมาก เช่น ถ้าต้องการสร้างตัวแปร a ให้เก็บเมตริกซ์ $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ สามารถสั่งได้ ดังนี้

>> a = [1 2 3; 4 5 6; 7 8 9];

หรือ a = [1, 2, 3; 4, 5, 6; 7, 8, 9] ก็ได้

สรุปว่า เครื่องหมายคอมม่า (,) หรือ วรรค ใช้แบ่งระหว่างคอลัมน์ และ เครื่องหมายเซมิโคลอน (;) ใช้แบ่งระหว่างแถว (สังเกตว่าการเติมวรรคมากน้อยไม่มีผลต่อ Matlab)

ส่วนเวคเตอร์ก็คือ เมตริกซ์ที่มีแถวเดียว (เรียกว่า เวคเตอร์แถว) หรือเมตริกซ์ที่มีคอลัมน์เดียว (เรียกว่า เวคเตอร์คอลัมน์) ซึ่งสามารถป้อนค่าได้โดยวิธีเดียวกันเช่น

>> a = [1 2 3 4 5 6 7];   เวคเตอร์แถว

>> a = [1; 2; 3; 4; 5; 6; 7];   เวคเตอร์คอลัมน์

สำหรับการสร้างเวคเตอร์แถวที่มีค่าเป็นเชิงเส้น   สามารถทำได้อย่างสะดวกอีกวิธีหนึ่งโดยใช้เครื่องหมายโคลอน (:) เช่น

>> a = 1:9   (หรือ a=[1:9])   จะให้ค่า a = [1 2 3 4 5 6 7 8 9]

>> a = 1 : 0.5 : 5   จะให้ค่า a = [1 1.5 2 2.5 3 3.5 4 4.5 5]

>> a = 3 : -2 : -10   จะให้ค่า a = [3 1 -1 -3 -5 -7 -9]

สรุปว่า รูปแบบการใช้คือ   [ค่าแรก : ค่าที่เพิ่มขึ้น : ค่าสุดท้าย] โดยสามารถละ [ ] ได้ และถ้าละค่ากลางก็จะถือว่า ค่าที่เพิ่มขึ้นเป็น 1

การกระทำทางเมตริกซ์ (Matrix operations)

การบวกลบคูณหาร และยกกำลังของเมตริกซ์ สามารถกระทำได้โดยใช้สัญลักษณ์เหมือนการกระทำกับค่าสเกลาปกติ   ข้อควรระวัง คือ ขนาดของเมตริกซ์ที่มากระทำกันต้องถูกต้องตามกฎของการกระทำนั้น ๆ เช่น ถ้าเอา a*b โดย a และ b เป็นเมตริกซ์   จะได้ว่าจำนวนแถวของ a จะต้องเท่ากับจำนวนคอลัมน์ของ b เสมอ และ ถ้าเอา a^2 จะต้องใช้ a ที่เป็นเมตริกซ์จัตุรัส (square) เสมอ เป็นต้น

การกระทำที่จะขอแนะนำเพิ่มเติมในที่นี้ คือ transpose ซึ่งทำได้โดยใช้สัญลักษณ์เครื่องหมายคำพูดขีดเดียว (') ดังนี้

>> b=a'   ให้เมตริกซ์ b เป็น transpose ของเมตริกซ์ a

หมายเหตุ ถ้าสมาชิกใน a เป็นจำนวนเชิงซ้อน การกระทำนี้จะเป็น conjugate transpose คือ นอกจาก transpose แล้ว ยังแปลงสมาชิกทุกตัวใน a เป็นค่าคอนจูเกตด้วย (เปลี่ยนเครื่องหมายของค่าจินตภาพ) ถ้าต้องการ transpose ธรรมดาโดยไม่มีคอนจูเกตให้ใช้เครื่องหมาย .' แทน ดังนี้

>> b=a.'

การกระทำที่เข้าถึงสมาชิกทุกตัวในเมตริกซ์ (Array operations)

สมมติมีเวคเตอร์ (หรือเมตริกซ์) ที่ขนาดเท่ากัน ดังนี้

>> a = [1 2 3];

>> b = [4 5 6];

และต้องการหาเวคเตอร์ ที่มีสมาชิกเป็นผลคูณของสมาชิกแต่ละตัวที่ตำแหน่งตรงกันของ a กับ b ทำได้โดยใช้เครื่องหมาย .* ดังนี้

>> a.*b

ans = [4 10 18]

การกระทำกับสมาชิกตรง ๆ ได้นี้มีประโยชน์มากในการประมวลผลสัญญาณ ซึ่งนอกจาก .* แล้ว ยังมีการกระทำในทำนองนี้อีก คือ

>> a./b เอาสมาชิกแต่ละตัวของ a กับ b มาหารกัน

>> a.^b เอาสมาชิกแต่ละตัวของ a ยกกำลังด้วยสมาชิกแต่ละตัวของ b

>> a.^3 เอาสมาชิกแต่ละตัวของ a ยกกำลังด้วย 3

>> a+3 เอาสมาชิกแต่ละตัวของ a บวกด้วย 3 (ลบก็ทำได้เช่นเดียวกัน)

สังเกตว่า ไม่มีการกระทำ .+ และ .- เพราะ การบวกลบเมตริกซ์เป็นการกระทำกับสมาชิกแต่ละตัวอยู่แล้ว

สำหรับฟังก์ชั่นของค่าสเกลาต่าง ๆ เมื่อนำมากระทำกับเมตริกซ์ ก็จะมีผลเป็นการกระทำกับสมาชิกแต่ละตัวในเมตริกซ์ และให้ผลลัพธ์เป็นเมตริกซ์ขนาดเท่าเดิม เช่น

>> sin(a) หาค่า sin ของสมาชิกแต่ละตัวของ a

การอ้างถึงสมาชิกภายในเมตริกซ์ (หรือเวคเตอร์)

ถ้ามีเมตริกซ์ $a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

เราสามารถอ้างถึงสมาชิกแต่ละตัวใน a ได้โดยใช้รูปแบบ a(แถว, คอลัมน์) เช่น a(2,3) จะได้ค่าเป็น 6 เป็นต้น ที่พิเศษไปกว่านั้นก็คือ Matlab อนุญาตให้เราอ้างค่าในเมตริกซ์ได้หลาย ๆ ค่าพร้อม ๆ กันเพื่อสร้างเป็นเมตริกซ์ หรือเวคเตอร์ใหม่ได้ เช่น

\>> a([1 2] , [2 3]) หรือ a(1:2 , 2:3) อ้างถึงแถวที่ 1 กับ 2 และ คอลัมน์ที่ 2 กับ 3

$$ans = \begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$$

\>> a(2 , [1 3]) อ้างถึงแถวที่ 2 และ คอลัมน์ที่ 1 กับ 3

ans = [4 6]

\>> a(2 , :) อ้างถึงแถวที่ 2 ทั้งแถว

ans = [4 5 6]

\>> a(: , 2) อ้างถึงคอลัมน์ที่ 2 ทั้งคอลัมน์ (สังเกตการใช้ :)

$$ans = \begin{bmatrix} 2 \\ 8 \end{bmatrix}$$

โดยสรุปก็คือ เราสามารถใช้ "เวคเตอร์เป็นตัวชี้" ได้ แทนที่จะใช้สเกลาเฉย ๆ สำหรับการอ้างค่าในเวคเตอร์ก็สามารถทำได้ในทำนองเดียวกัน เพียงแต่ใช้ตัวชี้เพียงตัวเดียว เช่น

\>> a = [1 2 3 4 5 6 7 8 9];

\>> a(3) อ้างถึงค่าที่ 3

ans = 3

\>> a(2:5) หรือ a([2:5]) อ้างถึงค่าที่ 2 ถึง 5

ans = [2 3 4 5]

\>> a(1:2:9) อ้างถึงค่าที่ 1, 3, 5, 7, และ 9

ans = [1 3 5 7 9]

จำนวนเชิงซ้อน

Matlab สามารถคำนวณระบบจำนวนเชิงซ้อนได้ โดยใช้รูปแบบเช่นเดียวกับการคำนวณปกติ โดยไม่ต้องเปลี่ยนแปลงใด ๆ และค่าเชิงซ้อนก็สามารถเป็นสมาชิกของเมตริกซ์ได้ ข้อสำคัญที่ต้องรู้จึงอยู่ที่รูปแบบการป้อนค่าจำนวนเชิงซ้อนให้กับ Matlab เท่านั้น

Matlab ใช้สัญลักษณ์ **i** และ **j** เป็นตัวแปรพิเศษภายในที่แทนค่า $\sqrt{-1}$ ซึ่งเป็นตัวคูณของส่วนจินตภาพของจำนวนเชิงซ้อน ในวิศวกรรมไฟฟ้าเราคุ้นเคยกับการใช้สัญลักษณ์ j มากกว่า i ดังนั้น จะขอยกตัวอย่างโดยใช้ j ดังนี้

>> a = 2 + j*2; ให้ค่า a เป็นจำนวนเชิงซ้อนเท่ากับ 2 + j2

>> b = 3 + j*4;

>> a*b

ans = -2.0000 +14.0000i

อย่างไรก็ตาม ในการแสดงคำตอบ Matlab จะใช้ i เป็นตัวคูณของส่วนจิตภาพเสมอซึ่งไม่เป็นปัญหาต่อการใช้งานของเราแต่อย่างใด มีข้อควรระวังอย่างหนึ่ง คือ ถ้าหากเราใช้ j เป็นตัวคูณของค่าจิตภาพ จะต้องไม่ใช้ j เป็นชื่อของตัวแปรอื่นใดทั้งสิ้น มิฉะนั้น j จะหมดค่าของ $\sqrt{-1}$ ไป

การป้อนค่าเป็นรูปแบบโพล่า สามารถทำได้โดยใช้ฟังก์ชั่น **exp( )** ซึ่งเป็นฟังก์ชั่นเอกโปเนนเชียล และใช้ **pi** ซึ่งเป็นตัวแปรพิเศษมีค่าเป็น π ดังนี้

>> a = 2*exp(j*pi/4) ให้ค่า a เป็นจำนวนเชิงซ้อนเท่ากับ $2e^{j\pi/4}$ หรือ $2\angle 45^{\circ}$

a = 1.4142 + 1.4142i

ฟังก์ชั่นที่เกี่ยวข้องกับจำนวนเชิงซ้อน ได้แก่

>> real(a) ให้ค่าส่วนจริงของ a

>> imag(a) ให้ค่าส่วนจินตภาพของ a

>> abs(a) ให้ค่าขนาดของ a

>> angle(a) ให้ค่ามุมของ a (เป็นเรเดียน)

>> conj(a) ให้ค่าคอนจูเกตของ a

ฟังก์ชั่นภายใน

Matlab มีฟังก์ชั่นพื้นฐานอยู่มากมาย และก็มีวิธีใช้บอกไว้ด้วยในซอฟท์แวร์ เราสามารถดูว่ามีฟังก์ชั่นชื่ออะไรอยู่บ้าง และใช้ทำอะไร โดยใช้เม้าส์เลือก Help ที่เมนู หรือพิมพ์ **help** ก็ได้ โดยฟังก์ชั่นต่าง ๆ ได้จัดไว้เป็นหมวดหมู่อย่างดี ในกรณีที่เรารู้ชื่อฟังก์ชั่นแต่จำวิธีใช้ไม่ได้ ก็สามารถเรียกดูวิธีใช้ได้โดยพิมพ์ help แล้วตามด้วยชื่อฟังก์ชั่น เช่น

>> help plot ขอดูวิธีใช้ฟังก์ชั่น plot

ในที่นี้ขอสรุปฟังก์ชั่นที่สำคัญบางส่วนไว้ ดังต่อไปนี้ (ให้ดูวิธีใช้โดยละเอียดจาก help)

<u>ฟังก์ชั่นเกี่ยวกับการสร้างเมตริกซ์</u>

>> zeros(n,m) ให้เมตริกซ์ที่มีสมาชิกเป็น 0 หมด ขนาด n x m

>> ones(n,m) ให้เมตริกซ์ที่มีสมาชิกเป็น 1 หมด ขนาด n x m

>> eye(n)	ให้เมตริกซ์ identity ขนาด n x n
>> rand(n,m)	ให้เมตริกซ์มีค่าแรนดอม 0 ถึง 1ขนาด n x m
>> randn(n,m)	ให้เมตริกซ์มีค่าแรนดอมแบบเกาส์เซียนขนาด n x m

<u>ฟังก์ชั่นของสเกลา</u>

sin, cos, tan, asin, acos, atan, atan2 ฟังก์ชั่นตรีโกณมิติ

sinh, cosh, tanh, asinh, acosh, atanh ฟังก์ชั่นไฮเปอร์โบลิก

exp (เอกโปเนนเชียล), log (ลอกการิธึมฐาน e), log10 (ลอกกาลิธึมฐานสิบ)

abs (หาขนาด), sign (หาเครื่องหมาย), rem (หาเศษจากการหาร)

round (ปัดทศนิยม), floor (ปัดทศนิยมทิ้ง), ceil (ปัดทศนิยมขึ้น)

sqrt (หารากที่สอง)

<u>ฟังก์ชั่นของเวคเตอร์</u>

max (หาค่ามากที่สุด), min (หาค่าน้อยที่สุด), length (หาจำนวนสมาชิก)

mean (หาค่าเฉลี่ย), median (หาค่ากลาง), std (หาค่าเบี่ยงเบนมาตรฐาน)

sum (หาผลรวม), sort (เรียงลำดับค่าจากน้อยไปมาก)

roots (หาค่ารากของโพลิโนเมียล)

<u>ฟังก์ชั่นของเมตริกซ์</u>

size (หาขนาด), eig (หาค่า eigen), det (หาค่า determinant)

โปรแกรมสคริปต์ (Script programs)

เราสามารถสั่งให้ Matlab ทำงานหลาย ๆ คำสั่งได้ทีเดียว โดยการเขียนคำสั่งเหล่านั้นไว้ในเท็กซ์ไฟล์ แล้วเก็บชื่อไฟล์นั้น**ให้มีนามสกุล .m** จากนั้นเรียกชื่อไฟล์นั้น (โดยไม่ต้องใส่ .m) ที่ prompt ของ Matlab โปรแกรมก็จะทำงานตั้งแต่คำสั่งแรกถึงคำสั่งสุดท้ายเลย

โปรแกรมประเภทนี้เรียกว่า โปรแกรมสคริปต์ ซึ่งการทำงานของโปรแกรมสคริปต์จะเทียบเท่ากับการกดคำสั่งเข้าไปใน Matlab โดยตรง ดังนั้น ตัวแปรต่าง ๆ ที่มีเรียกใช้ในโปรแกรมก็ต้องเป็นตัวแปรที่มีการตั้งไว้ก่อนเรียกโปรแกรมให้ทำงาน หรือตัวแปรที่มีการตั้งเพิ่มเติมในโปรแกรม ก็จะยังคงค้างอยู่ในหน่วยความจำหลังจากโปรแกรมทำงานเสร็จแล้ว

สำหรับการเขียน หรือแก้ไขไฟล์สคริปต์จะใช้โปรแกรมที่แก้ไขเท็กซ์ไฟล์ (text editor) ใด ๆ ก็ได้ นอกจากนี้ ยังสามารถเรียกโปรแกรมเพื่อแก้ไขไฟล์ได้จากเมนูของ Matlab โดยตรง โดยเลือกที่ <u>F</u>ile -> <u>N</u>ew หรือ <u>F</u>ile -> <u>O</u>pen M-file โปรแกรมที่ Matlab เรียกใช้ในการเปิดไฟล์ คือ notepad สำหรับผู้ที่ถนัดใช้โปรแกรมตัวอื่น ก็สามารถแก้ให้ Matlab ไปเรียกโปรแกรมที่ต้องการได้ โดยเลือกที่เมนู <u>O</u>ptions -> Editor <u>P</u>reference

<u>ข้อควรระวัง</u> ผู้ที่ใช้โปรแกรม notepad ที่อยู่ใน Windows 95 สำหรับเขียนโปรแกรมสคริปต์ อาจพบปัญหาว่า ไม่สามารถเก็บไฟล์ที่มีนามสกุลเป็น .m ได้ เนื่องจาก notepad จะเติมนามสกุล .txt ให้โดยอัตโนมัติ อาจแก้ไขได้โดยใช้เครื่องหมายคำพูดล้อมรอบชื่อไฟล์ในการตั้งชื่อไฟล์สำหรับการ save ครั้งแรก (เช่น "abc.m") หรืออาจให้ Matlab ไปเรียกใช้โปรแกรมเขียนเท็กซ์ไฟล์อื่น เช่น PFE (Programmer File Editor) ซึ่งหาได้ทั่วไปในอินเตอร์เนท

โปรแกรมสคริปต์ที่จะเรียกใช้ได้ จะต้องอยู่ในไดเรกทอรีปัจจุบัน หรือไดเรกทอรีที่ค้นหาได้ (search path) ขอให้อ่านรายละเอียดในหัวข้อ การจัดการเกี่ยวกับไดเรกทอรี และไฟล์

คำสั่งเกี่ยวกับการวนลูป และเปรียบเทียบ

Matlab มีคำสั่งสำหรับการวนลูป และเปรียบเทียบเช่นเดียวกับภาษาสูงทั่ว ๆ ไป คำสั่งเหล่านี้ มีประโยชน์มากในการเขียนในโปรแกรมสคริปต์ ซึ่งได้แก่

คำสั่ง if มีรูปแบบการใช้ คือ

```
if <เงื่อนไข>
   <คำสั่ง>
elseif <เงื่อนไข>                (elseif จะมีหลายครั้งก็ได้)
   <คำสั่ง>
else
   <คำสั่ง>
end
```

<คำสั่ง> ประกอบด้วย คำสั่ง หรือการเรียกฟังก์ชั่น หลาย ๆ คำสั่งก็ได้

<เงื่อนไข> คือ ประโยคที่เปรียบเทียบเพื่อให้ค่า 0 ถ้าเป็นเท็จ และให้ค่า 1 ถ้าเป็นจริง เช่น

```
if a == b & ok              ถ้า a เท่ากับ b และ ok เท่ากับ 1
   a = a+1;                 ให้เพิ่มค่า a ขึ้นหนึ่ง
else                        ถ้าไม่เช่นนั้น
   a = a-1;                 ให้ลดค่า a ลงหนึ่ง
end
```

การกระทำที่ใช้สำหรับเปรียบเทียบมีอยู่หลายตัว สรุปได้ดังนี้

==	เท่ากับ	~=	ไม่เท่ากับ
<	น้อยกว่า	>	มากกว่า
<=	น้อยกว่าหรือเท่ากับ	>=	มากกว่าหรือเท่ากับ

และการกระทำที่ใช้เชื่อมเงื่อนไข ได้แก่

&	และ (and)	|	หรือ (or)
~	ไม่ (not)		

การกระทำทั้งสองหมวดนี้ จริง ๆ แล้วยังมีวิธีการใช้เหมือนกับการใช้การกระทำบวก/ลบที่ได้กล่าวมาแล้ว เพียงแต่จะให้ผลลัพธ์เป็น 0 กับ 1 เท่านั้น เช่น

การเปรียบเทียบเมตริกซ์กับค่าคงที่

```
>> a = [1 2 3 4] > 2
  a = [0 0 1 1]
```

จะเอาค่าสมาชิกทุกตัวเปรียบเทียบว่ามากกว่า 2 หรือไม่ สร้างเป็นเมตริกซ์ใหม่ซึ่งเป็นผลลัพธ์ของการเปรียบเทียบ (ถ้าการเปรียบเทียบเป็นจริงให้ค่าเท่ากับหนึ่ง ถ้าเป็นเท็จให้ค่าเท่าศูนย์)

หรือ การเปรียบเทียบเมตริกซ์ กับเมตริกซ์

>> a = [1 2 3 4] == [1 4 3 5] เปรียบเทียบสมาชิกที่ตำแหน่งตรงกันว่าเท่ากันหรือไม่

  a = [1 0 1 0]

คำสั่ง while มีรูปแบบการใช้ คือ

while <เงื่อนไข>

<คำสั่ง>

end

เช่น
```
i = 10;  a=1;
while i > 1
   a = a*i;
   i = i - 1;
end
```

โปรแกรมนี้จะหาค่าสุดท้ายของ a เป็น 10! (10 แฟคทอเรียล) ซึ่งเท่ากับ 10*9*8* … 1

คำสั่ง for มีรูปแบบการใช้ คือ

for ตัวแปรลูป = ค่าต้น: ค่าที่เพิ่ม: ค่าสุดท้าย

<คำสั่ง>

end

เช่น
```
a=1;
for i = 1:10
   a = a*i;
end
```

เช่นเดียวกัน โปรแกรมนี้จะหาค่า 10! เช่นกัน ขอสังเกตความแตกต่างจากการใช้ while โดยปกติถ้าเรารู้ค่าเริ่มต้น และค่าสิ้นสุดของการวนลูป การใช้ for จะสะดวกกว่า

<u>เทคนิค</u> ถ้าต้องการหยุดการทำงานของลูป หรือของโปรแกรมขณะที่มันกำลังทำงานอยู่ ให้กด ctrl-C

โปรแกรมฟังก์ชั่น (Function programs)

นอกจากฟังก์ชั่นภายในของ Matlab แล้ว เรายังสามารถเขียนฟังก์ชั่นไว้ใช้งานเองได้ด้วย โดยเขียนเป็นเท็กซ์ไฟล์ และตั้งชื่อไฟล์ให้มีนามสกุล .m เช่นเดียวกับการเขียนโปรแกรมสคริปต์ จุดที่แตกต่างจากโปรแกรมสคริปต์ คือ

1. ฟังก์ชั่นมีการรับค่าตัวแปร และคืนค่าตัวแปร
2. ตัวแปรที่สร้างเพื่อใช้ในฟังก์ชั่นเป็นตัวแปรแบบภายในทั้งสิ้น เมื่อโปรแกรมทำงานเสร็จ ตัวแปรเหล่านั้นจะหายไป ยกเว้นตัวแปรที่คืนค่าเท่านั้น

บรรทัดแรกของโปรแกรมฟังก์ชั่น จะต้องมีรูปแบบ ดังนี้

function ตัวแปรที่คืน = ชื่อฟังก์ชั่น (ตัวแปรที่รับมา)

สมมติเราต้องการเขียนฟังก์ชั่นเพื่อหาค่าแฟคทอเรียลของค่าใด ๆ อาจเขียนได้ดังนี้

```
function out = fact(x)
out = 1;
for i = 2 : x
    out = out*i;
end
```

เก็บไฟล์นี้ในชื่อ fact.m จากนั้นทดสอบการทำงานของฟังก์ชั่นนี้โดยเรียกที่ Matlab ดังนี้

```
>> fact(5)
  ans = 120
>> fact(0)
  ans = 1
```

โปรแกรมนี้ให้ผลถูกต้องตามที่ต้องการ สังเกตว่าตัวแปร i และ x เป็นตัวแปรภายในซึ่งจะไม่มีผลใด ๆ ต่อตัวแปรข้างนอกถึงแม้จะมีชื่อซ้ำกัน และจะเห็นว่าการใช้งานฟังก์ชั่นที่สร้างขึ้นเองเหมือนกับการใช้งานฟังก์ชั่นภายในของ Matlab เองทุกประการ ลองมาดูตัวอย่างของฟังก์ชั่นที่ยากขึ้น ซึ่งมีการรับค่า และคืนค่ามากกว่าหนึ่งตัว

```
function [t, out] = gensine(f, fs, N)
%This function generates a sin wave output.
%   usage   [t, out] = gensine(f, fs, N)
%      or   [out] = gensine(f, fs, N)
%where f : frequency,  fs : sampling rate,  N : number of samples
%      t : time vector,  out : output vector

if nargin==2   N=100;  end        % set default N

t = [0:N-1]/fs;                   % t is time vector.
out = sin(2*pi*f*t);              % out is sine wave vector.

if nargout==1   t = out;  end     % if there is only only 1 output, then
                                  % return "out" as "t".
```

ฟังก์ชั่นนี้สร้างเวคเตอร์ที่เป็นสัญญาณซายน์โดยการระบุความถี่ (f), อัตราสุ่ม (fs), และจำนวนจุด (N) ที่ต้องการ ฟังก์ชั่นจะคืนค่าเป็นเวคเตอร์สองตัว คือ t กับ out สังเกตว่า t กับ out ต้องอยู่ในเครื่องหมาย [ ] ซึ่งเราสามารถเรียกฟังก์ชั่นนี้ใช้ได้โดย

>> [t,x] = gensine(200, 1e5, 200);

เมื่อได้เวคเตอร์ t คือเวคเตอร์ค่าเวลา และ x คือเวคเตอร์ของค่าซายน์ ก็สามารถนำไปวาดกราฟ หรือประมวลผลต่อไปได้ ในฟังก์ชั่นนี้มีการตรวจสอบว่า ถ้าผู้ใช้ใส่ตัวแปรเข้ามาแค่สองตัวแทนที่จะเป็นสามตัวก็จะให้ N=100 เองโดยอัตโนมัติ และถ้าผู้ใช้มีตัวแปรรับแค่ตัวเดียว ค่าที่จะส่งออกจากฟังก์ชั่นจะเป็นเวคเตอร์ตัวแรกเสมอ ในที่นี้ คือ t โปรแกรมก็จะทำการสลับให้ t=out ซึ่งจะทำให้ค่าที่ออกมาเป็นเวคเตอร์ของค่าซายน์ แทนที่จะเป็นเวคเตอร์ค่าเวลา ดังนั้น ถ้าเราไม่ต้องการใช้เวคเตอร์เวลา และยอมรับค่า N ที่ 100 ก็สามารถเรียกใช้ฟังก์ชั่นนี้ได้แบบสั้นลง เช่น

>> x = gensine(200, 1e5);

ตัวแปรพิเศษที่ใช้ตรวจสอบว่า ผู้ใช้เรียกใช้ฟังก์ชั่นโดยมีตัวแปรขาเข้า และขาออกกี่ตัว ก็คือ **nargin** และ **nargout** ตามลำดับ (ย่อมาจาก number of arguments in และ number of arguments out)

โปรแกรมนี้ยังได้แนะนำสัญลักษณ์พิเศษอีกอันหนึ่ง คือ % ซึ่งใช้นำหน้าส่วนที่เป็นคำอธิบายของโปรแกรม สำหรับคำอธิบายที่อยู่ตอนต้นของโปรแกรม ยังมีลักษณะพิเศษคือ จะปรากฏออกมาเมื่อผู้ใช้ขอดู help ลองพิมพ์คำสั่งนี้ดู

>> help gensine

การจัดการเกี่ยวกับไดเรกทอรี และไฟล์

ใน Matlab มีคำสั่งที่จัดการกับเกี่ยวกับไดเรกทอรี และไฟล์บางคำสั่งที่คล้ายกับ DOS ได้แก่

pwd หรือ **cd**	ดูว่าตอนนี้อยู่ที่ไดเรกทอรีอะไร
cd <path>	ย้ายไดเรกทอรีปัจจุบัน
dir	ขอดูไฟล์ในไดเรกทอรี
type <file>	แสดงข้อความในไฟล์
delete <file>	ลบไฟล์
path	ดู และเพิ่มเติม ไดเรกทอรีสำหรับค้นหาโปรแกรม (search path)

Matlab จะสามารถเรียกโปรแกรมสคริปต์ หรือฟังก์ชั่น ที่อยู่ในไดเรกทอรีปัจจุบัน หรือในไดเรกทอรีสำหรับค้นหาเท่านั้น สามารถดูได้ว่าตอนนี้ไดเรกทอรีสำหรับค้นหามีอะไรบ้างโดยใช้คำสั่ง path เฉย ๆ และสามารถเติมไดเรกทอรีสำหรับค้นหาได้โดยใช้คำสั่ง path ดังนี้

```
>> path(path, 'c:\data\myfunc')
```

คำสั่งนี้จะทำให้ Matlab เรียกใช้โปรแกรมที่อยู่ในไดเรกทอรี c:\data\myfunc ได้โดยไม่ต้องย้ายเข้าไป แต่วิธีนี้จะเป็นการเติมชั่วคราว ถ้าปิด Matlab ก็จะหายไป วิธีเติมถาวรสามารถทำได้โดยเติมคำสั่งนี้ลงไปในไฟล์ startup.m (อยู่ใน \matlab\bin) ซึ่งเป็นโปรแกรมที่จะถูกเรียกให้ทำงานโดยอัตโนมัติทุกครั้งที่เปิด Matlab หรือใช้วิธีแก้ไขไฟล์ matlabrc.m (อยู่ใน \matlab) ก็ได้

การวาดกราฟ

กราฟกล่าวได้ว่าเป็นสิ่งที่สำคัญที่สุดในการวิเคราะห์ และแสดงผล Matlab สามารถวาดกราฟได้หลายชนิดทั้งสองมิติ และสามมิติ กราฟที่แสดงในหน้าปกของหนังสือนี้ก็วาดจาก Matlab ในที่นี้จะขอแนะนำเฉพาะการวาดกราฟสองมิติเท่านั้น

สมมติว่าเรามีเวคเตอร์ t, x, y ซึ่งเกิดจากฟังก์ชั่น gensine ที่เขียนไว้ในหัวข้อเรื่องโปรแกรมฟังก์ชั่น ดังนี้

```
>> [t,x] = gensine(50,1000);
>> [t,y] = gensine(20,1000);
```

เราต้องการวาดกราฟของ x และ y ในรูปเดียวกันโดยมี t เป็นแกนนอน สามารถทำได้ดังนี้

>> plot(t, x)	วาดโดยใช้ t เป็นแกนนอน และ x เป็นแกนตั้ง
>> hold on	ค้างรูปเอาไว้ (การวาดครั้งต่อไป จะวาดซ้อนบนรูปเดิม)
>> plot(t, y, 'b-.')	วาด y โดยคราวนี้ใช้สีน้ำเงิน และเป็นเส้นประขีดสลับจุด

>> grid	วาดเส้นกริด
>> axis([0 0.1 -1.5 1.5])	ปรับช่วงของการแสดงผลให้แกนนอนอยู่ระหว่าง 0 ถึง 0.1 และแกนตั้งอยู่ระหว่างค่า -1.5 ถึง 1.5
>> xlabel('time (sec)')	เขียนคำอธิบายแกนนอน
>> ylabel('signal (V)')	เขียนคำอธิบายแกนตั้ง
>> title('Samples of Sine Waves')	เขียนคำอธิบายบนหัวรูป
>> hold off	ยกเลิกการค้างรูป (การวาดครั้งต่อไป จะลบรูปเก่าทิ้งก่อน)

รูปที่ ก.1 แสดงผลลัพธ์ของรูปที่ได้จากคำสั่งเหล่านี้

สรุปว่าคำสั่ง plot ต้องการตัวแปรเข้า 3 ตัว คือ

plot(เวคเตอร์ของแกนนอน, เวคเตอร์ของแกนตั้ง, รูปแบบของสีและลายเส้น)

โดยอย่างน้อยเราต้องใส่ค่าเวคเตอร์ของแกนตั้งเสมอ เช่น ลองสั่ง plot(x) ดู จะพบว่า Matlab ใช้ค่า 1, 2, 3, … เป็นแกนนอน สำหรับรูปแบบของสีและลายเส้นให้ใส่เป็นข้อความภายในเครื่องหมาย '' โดยอักษรตัวแรกเป็นตัวกำหนดสี และตัวต่อไปกำหนดลายเส้น โดยมีความหมาย คือ

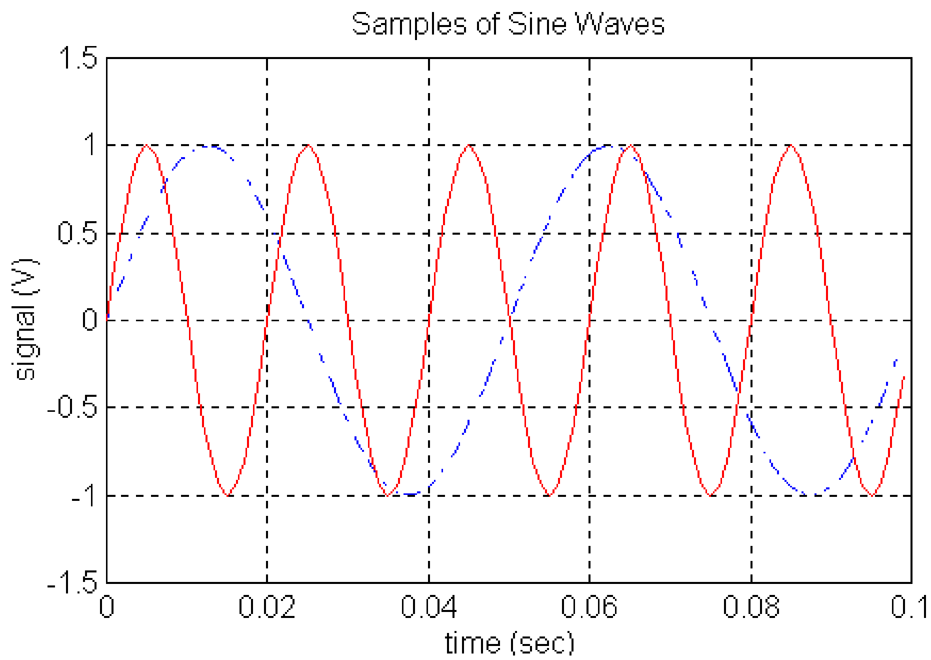
b สีน้ำเงิน, r สีแดง, k สีดำ, w สีขาว, y สีเหลือง, m สีม่วง, c สีฟ้า, g สีเขียว

- เส้นปกติ, **:** เส้นปะไข่ปลา, **--** เส้นปะขีด, **-.** เส้นปะขีดสลับจุด,

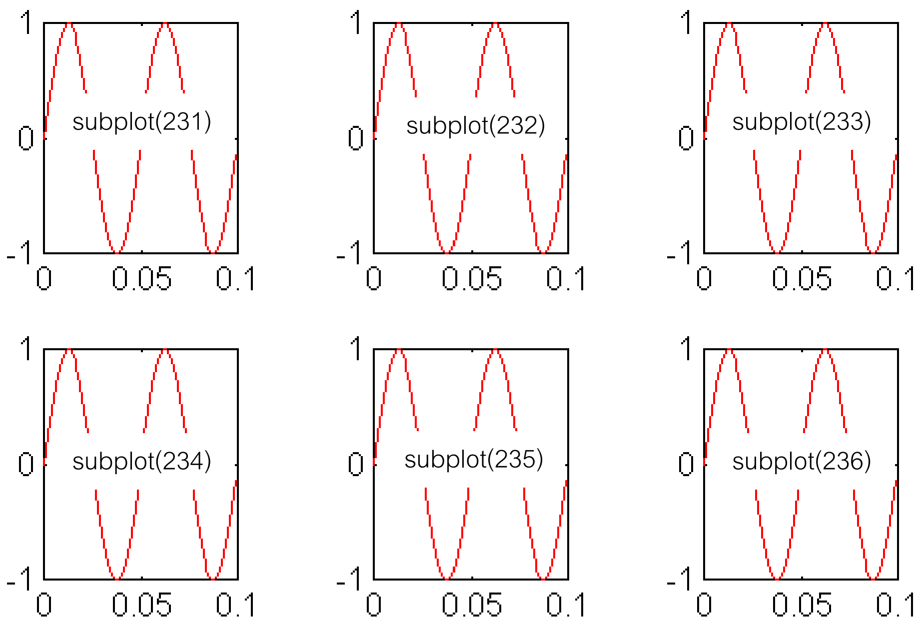
. จุด, * จุดดอกจัน, + จุดกากบาท, x จุดกากบาท, o จุดวงกลม

การวาดกราฟหลาย ๆ เส้นในรูปเดียวกันนอกจากทำโดยใช้คำสั่ง hold แล้ว ยังสามารถสั่งให้ plot ทีเดียวหลาย ๆ เส้นได้เลย โดยคำสั่งต่อไปนี้ซึ่งมีผลเหมือนข้างต้น

>> plot(t, x, t, y, 'b-.')



รูปที่ ก.1 *ตัวอย่างของกราฟที่วาดจาก Matlab*



***รูปที่ ก.2** ตัวอย่างของการใช้ subplot*

คำสั่งอื่นที่เกี่ยวกับการวาดกราฟที่สำคัญได้แก่

whitebg เปลี่ยนสีพื้นของรูปเป็นสีขาว มีประโยชน์มากถ้าจะพิมพ์รูปออกเครื่องพิมพ์ คำสั่งนี้สั่งครั้งเดียวตอนเปิด Matlab ซึ่งเราอาจใส่ไว้ใน \matlab\bin\startup.m เลยก็ได้

stem ใช้แทน plot สำหรับวาดรูปเป็นจุด และมีขีดแกนตั้งให้ด้วย

semilogx และ **semilogy** ใช้แทน plot สำหรับวาดรูปที่แกนนอน หรือตั้งเป็นสเกลล็อก

figure เปิดรูปใหม่ (ในหน้าต่างใหม่)

clf ลบรูปปัจจุบัน

zoom ขยายรูปที่แสดงผลอยู่ โดยหลังจากใช้คำสั่ง zoom แล้ว ให้กดปุ่มซ้ายของเม้าส์ค้างไว้แล้วลากเม้าส์เพื่อตีกรอบภายในบริเวณของรูปกราฟ เมื่อปล่อยปุ่มเม้าส์ รูปในกรอบก็จะถูกขยาย ถ้าต้องการหดรูปเหมือนเดิมให้กดปุ่มขวาของเม้าส์ที่บริเวณรูปนั้น

subplot(n,m,i) หรือ **subplot(nmi)** แบ่งรูปย่อยในหน้าต่างเดียวกัน ให้มี **n** x **m** รูปย่อย และชี้ที่รูปที่ i ตัวอย่างเช่น subplot(231) แบ่งเป็นรูปย่อย 6 รูป และชี้ที่รูปที่ 1 ดังแสดงในรูปที่ ก.2 รูปย่อยแต่ละรูปมีการตั้งค่าต่าง ๆ แยกอิสระต่อกันเหมือนเป็นคนละรูป ซึ่งเราต้องใช้คำสั่ง subplot ระบุ หรือใช้เม้าส์คลิกที่รูปย่อยหนึ่ง ๆ เพื่อย้ายไปกระทำกับรูปย่อยนั้น

การวัดประสิทธิภาพของโปรแกรม

การวัดประสิทธิภาพ หรือความเร็วของการทำงานของโปรแกรม หรือฟังก์ชั่น สามารถทำได้โดยใช้คำสั่ง **tic** และ **toc** ดังนี้ tic; เรียกโปรแกรม; toc เช่น

>> tic; fft(1:1024); toc วัดเวลาของการคำนวณ FFT 1024 จุด

elapsed_time = 0.0600 ใช้เวลา 0.06 วินาที

ถ้าต้องการวัดละเอียดเป็นจำนวนของ flops (floating-point operation) ที่ใช้ ก็สามารถทำได้โดยใช้คำสั่ง flops ดังนี้ **flops(0)**; เรียกโปรแกรม; **flops** เช่น

>> flops(0); fft(1:1024); flops

ans = 31050 ใช้ 31050 flops

การเก็บตัวแปร

เมื่อปิด Matlab ตัวแปรที่เราตั้งขึ้นใช้งานจะหายไปหมด ถ้าต้องการเก็บสถานะของตัวแปรทุกตัวไว้เพื่อใช้งานคราวต่อไป ให้ใช้คำสั่ง **save** และ **load** ดังนี้

>> save จะเก็บตัวแปรทุกตัวไว้ในไฟล์ matlab.mat ในไดเรกทอรี่ปัจจุบัน

ลอง clear แล้วสั่ง who ดู จากนั้นใช้คำสั่ง load ดังนี้

>> load จะเรียกตัวแปรออกจากไฟล์ matlab.mat

เราสามารถเก็บตัวแปรไว้ในไฟล์ชื่ออื่นได้โดยใช้ save แล้วตามด้วยชื่อไฟล์ การใช้ load ก็ทำได้ในทำนองเดียวกัน นอกจากนี้คำสั่ง save ยังใช้เก็บเฉพาะบางตัวแปรได้ด้วย เช่น

>> save myvar a b เก็บตัวแปร a และ b ในไฟล์ myvar.mat

>> load myvar เรียกตัวแปรทุกตัวในไฟล์ myvar.mat กลับออกมา

การจัดการเกี่ยวกับข้อความ (string)

ที่ผ่านมาได้กล่าวเฉพาะการจัดการเกี่ยวกับตัวเลข จริง ๆ แล้ว Matlab ยังสามารถจัดการเกี่ยวกับข้อความได้ด้วยในระดับหนึ่ง โดยข้อความใน Matlab จะต้องอยู่ในเครื่องหมาย ' ' เสมอ เช่น

>> a='hello' ตั้ง a เป็นตัวแปรที่เก็บข้อความ 'hello'

เราสามารถเข้าถึงตัวอักษรแต่ละตัวใน a ได้ โดยมองเหมือน a เป็นเวคเตอร์ที่มีสมาชิกเป็นตัวอักษร นั่นคือ a(1) = 'h', a(2) = 'e', … เป็นต้น คำสั่งอื่น ๆ ที่เกี่ยวข้องกับการจัดการข้อความ คือ

disp แสดงข้อความ หรือค่าของตัวแปรทางหน้าจอ

abs แปลงตัวอักษร หรือข้อความเป็นตัวเลขที่เป็นรหัสแอสกี้ของมัน

eval แปลงข้อความเป็นตัวเลข และหาค่า เช่น eval('1+1') = 2

num2str, int2str แปลงตัวเลขเป็นข้อความ

fprintf เขียนข้อความออกจอ หรือลงไฟล์ โดยมีการจัดรูปแบบเหมือนฟังก์ชั่น printf ในภาษาซี ขอให้ลองศึกษาได้จากคู่มือภาษาซีทั่ว ๆ ไป

sprintf ให้ค่าข้อความโดยมีการจัดรูปแบบเช่นเดียวกับ fprintf ซึ่งข้อความนี้สามารถเก็บเป็นตัวแปร หรือใช้เป็นค่าที่ป้อนให้กับฟังก์ชั่นอื่นได้

fscanf, sscanf อ่านข้อความโดยมีการจัดรูปแบบจากไฟล์ หรือจากตัวแปร

input ให้ผู้ใช้ใส่ข้อความจากคีย์บอร์ด

ผู้ที่สนใจขอให้ดูรายละเอียดการใช้งานจากคำสั่ง help ในที่นี้จะยกตัวอย่างโปรแกรมสคริปต์ง่าย ๆ ที่รับตัวเลขจากผู้ใช้ แล้วหาค่ายกกำลังสองของเลขนั้นเพื่อแสดงให้เห็นการทำงานของคำสั่งฟังก์ชั่น input และ fprintf

```
answer = 'y';
while answer == 'y'
   a = input('Please input a number : ');
   fprintf(1, '%7.2f square is %7.2f\n', a, a^2);
   answer = input('Continue (y/n) ? ', 's');
end
```

printf ในคำสั่งนี้ใส่ค่า id ของไฟล์เป็น 1 ซึ่งจะหมายถึงการเขียนข้อความออกทางหน้าจอ

คำสั่งเกี่ยวกับเสียง

ถ้าเรามีการ์ดเสียง (sound card) ต่ออยู่กับคอมพิวเตอร์ Matlab จะสามารถเป็นโต๊ะทดลองที่ดีในการประมวลผลสัญญาณเสียง เพราะมีคำสั่งในการอ่านไฟล์เสียงที่มีนามสกุล .wav และเล่นเสียงอยู่ในตัว ได้แก่ คำสั่ง **wavread**, **wavwrite**, และ **sound** ตัวอย่างการใช้งานเช่น

```
>> [x, fs] = wavread('c:\windows\tada.wav');
```

จะทำการอ่านไฟล์ tada.wav เข้ามาเก็บในเวคเตอร์ x และอัตราการสุ่มของสัญญาณเก็บใน fs ซึ่ง x จะเป็นเวคเตอร์ที่มีความยาวเท่ากับจำนวนตัวอย่างในไฟล์ tada.wav เราสามารถนำ x ไปประมวลผลต่อไปได้ หรือเล่นเสียงฟังจาก Matlab โดยตรงก็ได้ โดยใช้คำสั่ง ดังนี้

```
>> sound(x, fs)
```

สำหรับคำสั่ง wavwrite จะทำการเก็บเวคเตอร์ใน Matlab ไปเป็นไฟล์เสียงได้ ตัวอย่างของคำสั่งต่อไปนี้เป็นการสร้างสัญญาณ DTMF ของการกดปุ่ม 1 มีความยาวครึ่งวินาที ซึ่งสัญญาณนี้

ประกอบด้วยความถี่ 697 Hz ผสมกับความถี่ 1209 Hz ซึ่งสร้างโดยฟังก์ชั่น gensine ที่เขียนไว้ในหัวข้อก่อนหน้านี้ จากนั้น เล่นเสียงออกทางลำโพง แล้วเก็บเสียงนี้ในไฟล์ชื่อ press1.wav

```
>> fs=11025;  f1=697;  f2=1209;
>> x = gensine(f1, fs, 5500) + gensine(f2, fs, 5500);
>> sound(x, fs)
>> wavwrite(x, fs, 'press1.wav')
```

เก็บ x เป็นไฟล์เสียงชื่อ press1.wav ในไดเรกทอรีปัจจุบัน

การทำกราฟที่เคลื่อนไหวได้

เรื่องนี้อาจจะเกินในส่วนเบื้องต้นไปสักหน่อย แต่เป็นเรื่องที่ไม่ยากนักถ้าศึกษาจากตัวอย่าง และมีประโยชน์ (และเพลิดเพลิน) มากในการใช้ดูผลการทดลองที่มีการเปลี่ยนแปลงตามเวลา

เทคนิคการทำภาพเคลื่อนไหวใน ทำได้โดยการวาดรูปใหม่ทับลงไปในรูปเดิม แต่เราจะไม่ใช้คำสั่ง plot ต่อ ๆ กัน เพราะรูปที่ได้จะกระพริบ และดูไม่นิ่มนวล เราจะใช้คำสั่ง plot เพียงครั้งเดียว แล้วใช้ตัวแปรชนิดพิเศษ เรียกว่า handle เป็นตัวเข้าถึงรูปที่วาดโดยคำสั่ง plot นี้ (การเข้าถึงนี้ทำได้โดยใช้คำสั่ง set เพื่อเปลี่ยนแปลงค่าของ property ต่าง ๆ ของตัวแปร) จากนั้นจะทำการวาดรูปทับโดยการเปลี่ยน property ชื่อ Ydata ของตัวแปร handle นี้

ตัวอย่างของโปรแกรมสคริปต์ข้างล่างนี้จะทำการวาดรูปสัญญาณซายน์วิ่งจากขวาไปซ้าย

```
n=0:99;
x=zeros(1,100);
h = plot(n, x, 'EraseMode', 'xor', 'color', 'c');     ตั้ง h เป็นตัวแปรเก็บ handle ของ plot
axis([0 100 -1 1]); grid on;

for i=1:1000
  x(1:99) = x(2:100);        } เปลี่ยนรูปร่างของ x เป็นรูปที่เวลาต่อไป
  x(100) = sin(0.1*i);       }
  set(h, 'Ydata', x);        ตั้ง x ให้เป็นค่าใน property Ydata ของ h
  pause(0);
end
```

ภาคผนวก ข

ฟังก์ชั่นใน Matlab DSP Toolbox

DSP Toolbox หรือ Signal Processing Toolbox เป็นชุดฟังก์ชั่นพิเศษซึ่งเพิ่มเติมเข้ามาใน Matlab (ซึ่งผู้ใช้ต้องซื้อเพิ่มเติมเอง) ฟังก์ชั่นเหล่านี้เกี่ยวข้องกับการประมวลผลสัญญาณทั้งในการวิเคราะห์ และออกแบบ ในที่นี้จะขอสรุปฟังก์ชั่นที่สำคัญซึ่งสำหรับใช้ในการประมวลผลสัญญาณขั้นพื้นฐานเท่านั้น โดยอ้างอิงกับ DSP Toolbox เวอร์ชั่น 3.0

ฟังก์ชั่นสำหรับคำนวณ และวิเคราะห์ตัวกรอง

Sinc	ฟังก์ชันซิงค์
conv	คอนโวลูชัน
fftfilt	คอนโวลูชันแบบเร็วโดยวิธี overlap-add
filter	ตัวกรองดิจิตอล
freqz	วาดผลตอบสนองเชิงความถี่ของระบบ
grpdelay	หาความเร็วกลุ่มของระบบ
impz	หาผลตอบสนองต่ออิมพัลส์ของระบบ
unwrap	ปรับเวคเตอร์ที่เป็นค่าเฟสที่ถูกจำกัดอยู่ใน $-\pi$ ถึง π ให้ขยายออกนอกย่านนี้ได้
zplane	วาดแผนภาพโพล/ศูนย์

ฟังก์ชั่นที่ใช้จัดรูปแบบของฟังก์ชั่นถ่ายโอน

residuez	กระจายฟังก์ชั่นเป็นเศษส่วนย่อย
sos2tf	แปลงจากรูปแบบผลคูณของเทอมอันดับสอง เป็นรูปแบบเศษส่วนรวม
sos2zp	แปลงจากรูปแบบผลคูณของเทอมอันดับสอง เป็นรูปแบบกระจายค่าโพลและศูนย์
tf2zp	แปลงจากรูปแบบเศษส่วนรวม เป็นรูปแบบผลคูณของเทอมอันดับสอง
zp2sos	แปลงจากรูปแบบกระจายค่าโพลและศูนย์ เป็นรูปแบบผลคูณของเทอมอันดับสอง
zp2tf	แปลงจากรูปแบบกระจายค่าโพลและศูนย์ เป็นรูปแบบเศษส่วนรวม

ฟังก์ชั่นสำหรับออกแบบตัวกรองแบบ IIR

butter	ออกแบบตัวกรอง Butterworth
cheby1	ออกแบบตัวกรอง Chebyshev type I
cheby2	ออกแบบตัวกรอง Chebyshev type II
ellip	ออกแบบตัวกรอง Elliptic

yulewalk	ออกแบบตัวกรอง Yule-Walker
buttord	หาอันดับของตัวกรอง Butterworth
cheb1ord	หาอันดับของตัวกรอง Chebyshev type I
cheb2ord	หาอันดับของตัวกรอง Chebyshev type II filter
ellipord	หาอันดับของตัวกรอง Elliptic filter

ฟังก์ชั่นสำหรับออกแบบตัวกรองแบบ FIR

fir1	ออกแบบโดยวิธีหน้าต่าง
fir2	ออกแบบโดยวิธีสุ่มความถี่
remez	ออกแบบโดยวิธี Parks-McClellan
remezord	หาอันดับของตัวกรองโดยวิธี Parks-McClellan

ฟังก์ชั่นการแปลง

fft	การแปลง FFT
fftshift	สลับผลตอบครึ่งบนของ FFT มาเป็นครึ่งล่าง
hilbert	การแปลง Hilbert
ifft	การแปลง FFT ผกผัน
psd	หา Power Spectral Density
xcorr	หา Cross-correlation
specgram	หา Spectrogram
bilinear	การแปลง Bilinear

ฟังก์ชั่นหน้าต่าง

bartlett	หน้าต่าง Bartlett
blackman	หน้าต่าง Blackman
boxcar	หน้าต่างสี่เหลี่ยม
chebwin	หน้าต่าง Chebyshev
hamming	หน้าต่าง Hamming
hanning	หน้าต่าง Hanning
kaiser	หน้าต่าง Kaiser
triang	หน้าต่าง Triangular

ฟังก์ชั่นการแปลงอัตราการสุ่ม

decimate	decimator
interp	interpolator
resample	แปลงอัตราการสุ่มด้วยอัตรา I/D เท่า

ฟังก์ชั่นเพื่อหาตัวกรองแอนะลอกต้นแบบ

besselap	ตัวกรองต้นแบบ Bessel
buttap	ตัวกรองต้นแบบ Butterworth
cheb1ap	ตัวกรองต้นแบบ Chebyshev type I
cheb2ap	ตัวกรองต้นแบบ Chebyshev ty[pe II
ellipap	ตัวกรองต้นแบบ Elliptic
lp2bp	การแปลง LPF เป็น BPF
lp2bs	การแปลง LPF เป็น BSF
lp2hp	การแปลง LPF เป็น HPF
lp2lp	การแปลง LPF เป็น LPF

ฟังก์ชั่นพิเศษเพื่อแสดงการใช้งาน

filtdemo	แสดงการออกแบบตัวกรองดิจิตอล
sosdemo	แสดงการลักษณะของฟังก์ชั่นถ่ายโอนย่อยอันดับสอง

ภาคผนวก ค

ประมวลคำศัพท์เทคนิค

ศัพท์เทคนิคในหนังสือนี้ ได้พยายามใช้ตามที่ได้มีบัญญัติไว้ในหนังสือประมวลศัพท์เทคนิคของสมาคมวิศวกรรมสถานฯ[16] อย่างไรก็ตาม มีบางคำที่ไม่ได้ใช้ตามที่บัญญัติไว้ ซึ่งได้ระบุไว้ด้วยเครื่องหมาย ** ท้ายคำศัพท์ และศัพท์อีกส่วนหนึ่ง คือ ศัพท์ที่ยังไม่ได้มีบัญญัติไว้ ได้ระบุไว้ด้วยเครื่องหมาย * ท้ายคำศัพท์

หมวดการสุ่มสัญญาณ

analog	แอนะลอก
digital	ดิจิตอล
sampling**	การสุ่ม, การสุ่มตัวอย่าง
sampling rate**, sampling frequency	อัตราการสุ่ม, ความถี่ในการสุ่ม
aliasing**	aliasing (อ่านว่า เอ-เลียด-ซิ่ง)
anti-aliasing filter*	ตัวกรองป้องกัน aliasing
reconstruction*	การสร้างสัญญาณคืน
reconstruction filter*	ตัวกรองสร้างสัญญาณคืน
quantization*	การแบ่งขั้นสัญญาณ
quantize*	แบ่งขั้นสัญญาณ
image**	สำเนา

หมวดสัญญาณ และระบบ

pulse	พัลส์
impulse	อิมพัลส์
sinc**	ซิงค์
discrete , discrete-time*	ไม่ต่อเนื่อง
continuous , continuous-time*	ต่อเนื่อง
spectrum	สเปกตรัม
real time	เวลาจริง
stability	เสถียรภาพ
stable	เสถียร
causality*	ความเป็นคอซัล

causal*	คอซัล
non-causal*	ไม่เป็นคอซัล
anticausal*	คอซัลแบบตรงข้าม
difference equation**	สมการผลต่าง
impulse response	ผลตอบสนองต่ออิมพัลส์, ผลตอบสนองต่อสัญญาณอิมพัลส์
transfer function	ฟังก์ชั่นถ่ายโอน
pole	โพล
zero	ศูนย์
frequency response	ผลตอบสนองเชิงความถี่
normalized frequency	ความถี่นอร์แมลไลซ์
transient response	ผลตอบสนองชั่วครู่
steady-state response	ผลตอบสนองสถานะอยู่ตัว
harmonic frequency	ความถี่ฮาร์มอนิก

หมวดการกระทำ และการแปลงต่าง ๆ

Fourier transform	การแปลงฟูริเยร์
Discrete-Time Fourier Transform*	การแปลงฟูริเยร์แบบเวลาไม่ต่อเนื่อง, การแปลง DTFT
Discrete Fourier Transform*	การแปลงฟูริเยร์แบบไม่ต่อเนื่อง, การแปลง DFT
Fast Fourier Transform*	การแปลง FFT
Fourier series	อนุกรมฟูริเยร์
Laplace transform	การแปลงลาปลาซ
z transform	การแปลง z
inverse z transform*	การแปลง z ผกผัน
region of convergence (ROC)*	บริเวณของการลู่เข้า
convolution	คอนโวลูชัน
linear convolution*	คอนโวลูชันแบบเชิงเส้น
circular convolution*	คอนโวลูชันแบบวงกลม
decimator*	เดซซิเมเตอร์
interpolator*	อินเตอร์โพเลเตอร์
down-sampler*	ตัวลดอัตราสุ่ม
up-sampler*	ตัวเพิ่มอัตราสุ่ม

หมวดตัวกรองดิจิตอล

order*	อันดับ
low-pass filter	ตัวกรองผ่านต่ำ (LPF)
high-pass filter	ตัวกรองผ่านสูง (HPF)
band-pass filter	ตัวกรองผ่านแถบความถี่ (BPF)

band-stop filter*	ตัวกรองตัดแถบความถี่ (BSF)
cutoff frequency	ความถี่ตัด
ripple	ความพลิ้ว
ripple factor	ตัวประกอบความพลิ้ว
attenuation	การลดทอน
pass band	แถบผ่าน
stop band	แถบหยุด
transition band*	แถบเปลี่ยน
window function*	ฟังก์ชั่นหน้าต่าง
rectangular window*	หน้าต่างแบบสี่เหลี่ยม
Hamming window*	หน้าต่างแฮมมิ่ง
Kaiser window*	หน้าต่างไคเซอร์
bilinear transform*	การแปลงไบลิเนียร์
adaptive filter*	ตัวกรองแบบปรับตัวได้

อื่น ๆ

random signal**	สัญญาณแรนดอม
fixed-point**	ระบบเลขจำนวนเต็ม
floating-point	ระบบเลขอิงดรรชนี
mantissa	แมนทิสซา
exponent	ตัวชี้กำลัง
overflow	โอเวอร์โฟล

หนังสืออ้างอิง

[1] S. J. Orfanidis, *Introduction to Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1996.

[2] E. C. Ifeachor, and B. W. Jervis, Digital Signal Processing: A Practical Approach, Addison-Wesley, New York, 1993.

[3] J. G. Proakis, and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd ed., Prentice Hall, NJ, 1996.

[4] S. K. Mitra, *Digital Signal Processing A Computer-Based Approach*, McGraw-Hill, Singapore, 1998.

[5] R. Kuc, *Introduction to Digital Signal Processing*, McGraw-Hill, Singapore, 1982.

[6] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1975.

[7] V. K. Ingle, J. G. Proakis, Digital Signal Processing Using Matlab V.4, PWS Publishing Company, Boston, 1997.

[8] W. W. Smith, and J. M. Smith, *Handbook of Real-Time Fast Fourier Transforms*, IEEE Press, Piscataway, NJ, 1995

[9] R. G. Lyons, Understanding Digital Signal Processing, Addison Wesley, Reading, MA, 1997

[10] S. Haykin, *Adaptive Filter Theory*, 3rd ed., Prentice Hall, Upper Saddle River, NJ, 1996.

[11] D. Johns and K. Martin, Analog Intergrated Circuit Design (Chapter 14 Oversampling Converters), John Wiley and Sons, Toronto, 1997

[12] K. Sigmon, Matlab Primer, 2nd ed., University of Florida, Gainesville, 1992 (เอกสารนี้มีแจกฟรีในอินเตอร์เนตที่ ftp://ftp.mathworks.com/pub/doc/primer/ หรือ ftp://ftp.ee.mut.ac.th/ pub/Matlab/)

[13] *TMS320C5x DSP Starter Kit User's Guide*, Texas Instruments, Dallas, TX, 1994

[14] *TMS320C3x User's Guide*, Texas Instruments, Dallas, TX, 1997

[15] *TMS320C5x User's Guide*, Texas Instruments, Dallas, TX, 1993

[16] ศัพท์เทคนิควิศวกรรมอิเล็กทรอนิกส์, ศัพท์เทคนิควิศวกรรมไฟฟ้าสื่อสาร, และศัพท์เทคนิควิศวกรรมคอมพิวเตอร์, สมาคมวิศวกรรมสถานแห่งประเทศไทย, มีนาคม 2539

[17] H. McAllister, N. Black, and N. Waterman, "Hearing aids - a development with digital signal processing devices," Computer and Control Engineering Journal, 1995, vol. 6, pp.283-291

[18] N. Magotra and S. Sirivara, "Real-time digital speech processing strategies for the hearing impaired," IEEE International Conference on Acoustics, Speech, and Signal Processing, 1997, vol. 2, pp. 1211 -1214.

[19] Y. Qin and S. Du, "A DSP based active power filter for line interactive UPS," Proceedings of the 1995 IEEE IECON 21st, 1995, vol. 2, pp. 884-888.

[20] B. Gardner and K. Martin, "HRTF Measurements of a KEMAR Dummy-Head Microphone," MIT Media Lab Perceptual Computing - Technical Report #280, 1994

[21] Aureal Home Page, http://www.aureal.com/, 1998.

[22] A. Reilly and D. Mcgrath, "Convolution Processing for Realistic Reverberation," Lake DSP Technical Paper, 1995. (http://www.lakedsp.com/)

[23] พรชัย ภววงษ์ศักดิ์ และนลิน สีดาห้าว, "การวิเคราะห์ความคลาดเคลื่อน และการใช้งานออสซิลเลเตอร์ดิจิตอลแบบ IIR," การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 21, 1998, หน้า 433-436