บทที่ 3

การบีบข้อมูลเสียงโดยวิธีการทำนายช่วงยาวโดยการกระตุ้นด้วยพัลส์เศษค้าง (อาร์พีอี-แอลทีพี)

ในบทนี้จะกล่าวถึงวิธีการบีบข้อมูลเสียงด้วยวิธีการทำนายช่วงยาวโดยการกระตุ้นด้วยพัลส์เศษค้างหรือเรียกย่อว่า อาร์พีอี-แอลทีพี ซึ่งเป็นวิธีการเดียวกันกับวิธีเข้ารหัสข้อมูลเสียงที่ใช้ในระบบโทรศัพท์เคลื่อนจีเอสเอ็ม จัดเป็นการบีบข้อมูลเสียงที่ให้คุณภาพของสัญญาณเสียงในระดับที่ดีโดยที่อัตราความถี่ของข้อมูลเสียงที่บีบแล้วไม่สูงนัก

Degener และ Bormann (1992) ได้แสดงวิธีวิเคราะห์หาพารามิเตอร์ของสัญญาณเสียงในวิธีอาร์พีอี-แอลทีพีโดยแบ่งสัญญาณเสียงหรือข้อมูลเสียงออกเป็นช่วง ๆ โดยที่แต่ละช่วงเรียกว่าเฟรม ในหนึ่งเฟรมประกอบไปด้วยตัวอย่างสัญญาณเสียง 160 ตัวอย่างด้วยกัน อัตราการสุ่มสัญญาณคือ 8 KHz ดังนั้นในแต่ละเฟรมจะใช้เวลาทั้งสิ้น 20 ms การวิเคราะห์แบ่งได้เป็นสองช่วง ในช่วงแรกเรียกว่าการวิเคราะห์ช่วงสั้น (short-term analysis) ส่วนการวิเคราะห์ในช่วงหลังเรียกว่าการวิเคราะห์ช่วงยาว (long-term analysis) การวิเคราะห์ช่วงสั้นจะได้พารามิเตอร์แอลพีซี นั่นคือสัมประสิทธิ์ของตัวกรองซึ่งมีจำนวน 8 ตัวด้วยกัน สำหรับวิธีที่ Degener และ Bormann ใช้ในการวิเคราะห์หาสัมประสิทธิ์คือวิธีของ Schur ซึ่งในรายละเอียดจะได้กล่าวถึงต่อไป ในการวิเคราะห์ช่วงสั้นสัญญาณเสียงจะถูกนำไปเข้าตัวกรองแอลพีซีย้อนกลับ (LPC inverse filter) เพื่อหาสัญญาณเศษค้าง (residual signal) ส่วนในการวิเคราะห์ช่วงยาวจะแบ่งเฟรมข้อมูลสัญญาณเศษค้างที่ได้ออกเป็นส่วนย่อย ๆ สี่ส่วนด้วยกัน โดยที่แต่ละส่วนมีขนาด 40 ตัวอย่างสัญญาณและเรียกว่าเฟรมย่อย (sub-frame) การวิเคราะห์จะทำทีละเฟรมย่อย ผลของการวิเคราะห์จะให้พารามิเตอร์ของการทำนายช่วงยาว (long term parameter) ซึ่งจะนำไปเข้ารหัส และเก็บในหน่วยจัดเก็บหรือส่งไปในช่องทางสื่อสาร พารามิเตอร์ดังกล่าวยังใช้สำหรับการสร้างสัญญาณกระตุ้นที่เหมาะสมสำหรับการคลายสัญญาณเสียงคืนด้วย รายละเอียดการทำงานของวิธีนี้มีดังนี้

การบีบข้อมูลเสียงด้วยวิธีอาร์พีอี-แอลทีพี

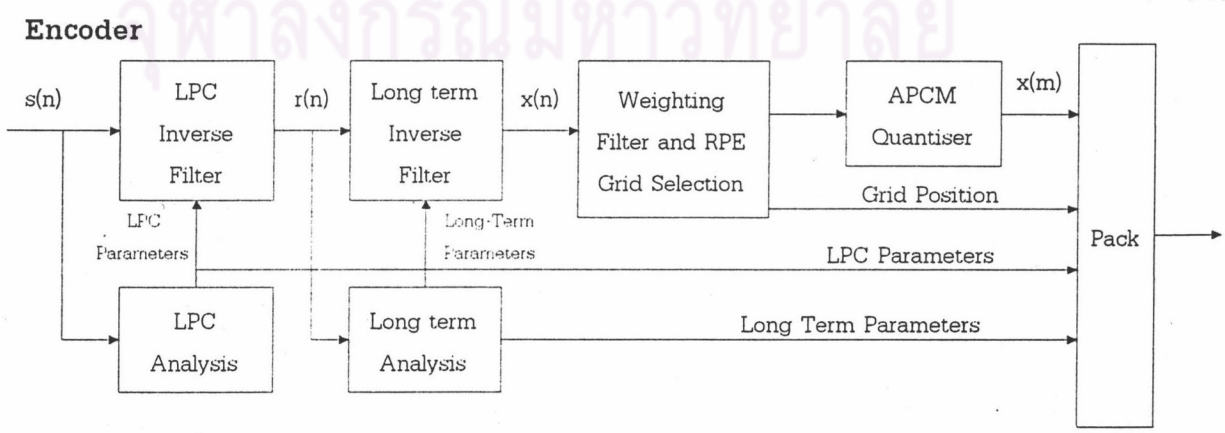
ในรูปที่ 3. 1 แสดงแผนผังการทำงานของการบีบข้อมูลเสียงโดยวิธีอาร์พีอี-แอลทีพี ในรูปสัญญาณเสียง s(n) ถูกนำไปวิเคราะห์เพื่อหาพารามิเตอร์แอลพีซีซึ่งได้แก่ สัมประสิทธิ์ของตัวกรองแอลพีซีจำนวนแปดตัวโดยกระบวนการ LPC Analysis ที่แสดงในรูป สัมประสิทธิ์ที่หาได้จะนำไปใช้ในตัวกรองแอลพีซีย้อนกลับหรือตัวกรองสังเคราะห์ดังที่ได้กล่าวมาแล้วในบทที่ 2 ซึ่งก็คือ LPC Inverse Filter ในรูป โดยมีฟังก์ชันโอนย้ายดังนี้

LPC Inverse Filter: $A(z) = 1 - \sum_{i=1}^{8} a_i z^{-i}$

ในทางปฏิบัติจะใช้ตัวกรองผลึก (lattice filter) ซึ่งมีฟังก์ชันโอนย้ายสมมูลกับฟังก์ชันโอนย้ายข้างบนแต่สัมประสิทธิ์ที่ใช้ในการคำนวณเป็นสัมประสิทธิ์การสะท้อน สำหรับในรายละเอียดจะได้กล่าวถึงในส่วนถัดไป สัญญาณที่ได้หลังจากการกรองสัญญาณเสียงด้วยตัวกรองแอลพีซีย้อนกลับคือสัญญาณเศษค้าง r(n) สัญญาณเศษค้างจะถูกนำไปวิเคราะห์ต่อเพื่อหาพารามิเตอร์ช่วงยาวโดยกระบวนการวิเคราะห์ช่วงยาว ซึ่งก็คือ Long term Analysis ในรูป พารามิเตอร์ที่ได้จากการวิเคราะห์จะนำไปใช้ในตัวกรองย้อนกลับช่วงยาว (Long term Inverse Filter) เพื่อคำนวณหาสัญญาณกระตุ้น x(n) ตัวกรองย้อนกลับช่วงยาวมีฟังก์ชันโอนย้ายดังนี้

Long term Inverse Filter: $P(z) = 1 - bz^{-lag}$

โดยที่ lag และ b เป็นพารามิเตอร์ช่วงยาวซึ่งจะได้กล่าวถึงรายละเอียดและวิธีการหาในช่วงต่อไป สำหรับสัญญาณกระตุ้น x(n) จะถูกกรองด้วยตัวกรองเวท (weighting filter) และทำการลดอัตราสุ่ม (down sampling) ลงเพื่อเป็นการลดปริมาณข้อมูล ก่อนที่จะนำไปเข้ารหัสด้วยวิธีเอพีซีเอ็ม (APCM - Adaptive Pluse Code Modulation)

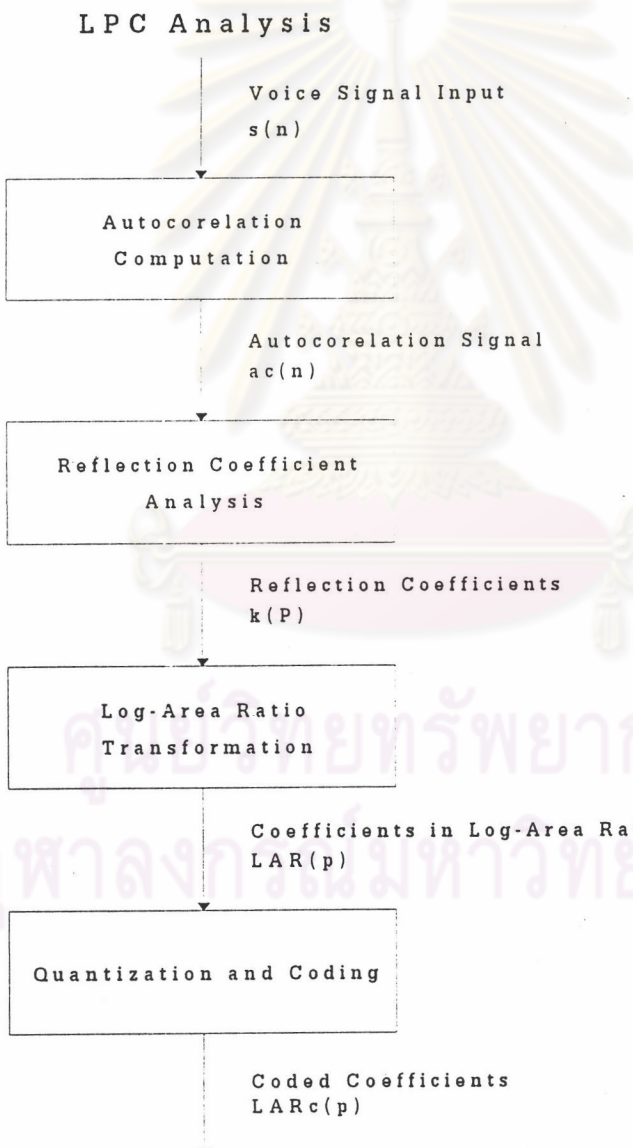


รูปที่ 3.1 แสดงหลักการทำงานของการบีบข้อมูลเสียงโดยวิธี อาร์พีอี-แอลทีพี

จะได้กล่าวถึงรายละเอียดการทำงานในแต่ละกระบวนการดังต่อไปนี้

1. <u>การวิเคราะห์แอลพีซี</u> (LPC Analysis) การวิเคราะห์แอลพีซีประกอบด้วยขั้นตอนย่อยหลายขั้นตอนด้วยกันดังที่แสดงในรูปที่ 3.2 กล่าวคือ

- การคำนวณออโตโครีเลชัน (Autocorelation Computation)
- การวิเคราะห์สัมประสิทธิ์การสะท้อน (Reflection Coefficient Analysis)
- การแปลงอัตราส่วนล็อกแอเรีย (Log-Area Ratio Transformation) และ
- ควอนไตเซชันและการเข้ารหัส (Quantization and Coding)



รูปที่ 3.2 แสดงขั้นตอนย่อยของการวิเคราะห์แอลพีซี

1.1 <u>การคำนวณออโตโครีเลชัน</u> (autocorrelation computation) ของสัญญาณเสียง การคำนวณหาออโตโครีเลชันของสัญญาณทำได้โดย การหาผลรวมของ s[n]·s[n+lag] สำหรับทุก ๆ n โดยที่ s เป็นแอเรย์ของข้อมูลเสียงหรือเฟรมของข้อมูลเสียงที่มีขนาดเท่ากับ window ในที่นี้ window คือ 160 ส่วน n เป็นดัชนีของแอเรย์ดังกล่าวมีค่าตั้งแต่ 0 จนถึง window-1 และ lag เป็นตำแหน่งของออโตโครีเลชันที่ต้องการหาในที่นี้ lag มีค่าตั้ง 0 ถึง P ในกรณี n+lag > window-1 ถือว่า s[n+lag] มีค่าเป็นศูนย์ ผลลัพธ์ของการคำนวณจะได้แอเรย์ของออโตโครีเลชันที่มีขนาด P+1 โดยที่ P เป็นจำนวนโพลหรือจำนวนของสัมประสิทธิ์ตัวกรองแอลพีซี ออโตโครีเลชันตำแหน่งที่ lag หรือ ac[lag] เขียนได้เป็น

$$ac[lag] = \sum_{n=0}^{window-lag}(s[n]\cdot s[n+lag]) \qquad \text{lag} = 0 \ldots P \qquad (25)$$

สัญญาณออโตโครีเลชันที่ได้จะถูกส่งไปยังส่วนถัดไปเพื่อวิเคราะห์หาสัมประสิทธิ์การสะท้อน

1.2 <u>การวิเคราะห์สัมประสิทธิ์การสะท้อน</u> (reflection coefficient analysis) ในหัวข้อการบีบข้อมูลเสียงด้วยวิธีแอลพีซี ได้กล่าวถึงการวิเคราะห์หาสัมประสิทธิ์การสะท้อนด้วยวิธีเลวินสัน-เดอบินไปแล้ว นอกจากวิธีเลวินสัน-เดอบินแล้ว Kondos (1995) ยังได้กล่าวถึงวิธีการอื่น ๆ ที่สามารถใช้ในการวิเคราะห์หาสัมประสิทธิ์การสะท้อนหรือสัมประสิทธิ์ทำนายได้แก่ วิธีโคแวเรียนซ์และวิธีผลึก ในที่นี้จะกล่าวถึงอีกวิธีหนึ่งซึ่งเสนอโดย Schur เนื่องจากเป็นวิธีที่ใช้ในการวิจัยบีบข้อมูลเสียงด้วยวิธีอาร์พีอี-แอลทีพีในครั้งนี้ Degener และ Bormann (1992) ได้กล่าวถึงวิธีของ Schur ไว้ว่ามีความสัมพันธ์กับวิธีของเลวินสัน-เดอบิน แต่สามารถทำงานได้เร็วกว่าบนเครื่องที่มีสถาปัตยกรรมแบบขนาน โดยที่ วิธีของ เลวินสัน-เดอบินจะต้องใช้เวลาเป็นอัตราส่วนกับ P·log(P) แต่วิธีของ Schur จะใช้เวลาเป็นอัตราส่วนกับ P เท่านั้น การวิเคราะห์ของ Schur สามารถเขียนเป็นฟังก์ชันในภาษาซีดังนี้

```
#define    P_MAX    8
/* returns the minimum mean square error    */
double schur(
    double const * ac,          /*  in: [0...p] autocorrelation values      */
    double       * ref)         /* out: [0...p-1] reflection coefficients   */
{
    int i, m;  double r, error = ac[0], G[2][P_MAX];

    if (ac[0] == 0.0) {
        for (i = 0; i < P_MAX; i++) ref[i] = 0;
        return 0;
    }

    /* Initialize the rows of the generator matrix G to ac[1...p]. */
    for (i = 0; i < P_MAX; i++)
        G[0][i] = G[1][i] = ac[i + 1];
```

```
    for (i = 0;;) {
        /* Calculate this iteration's reflection coefficient and error. */
        ref[i] = r = -G[1][0] / error;
        assert( ref[i] <= 1.0 );
        error += G[1][0] * r;

        if (++i >= P_MAX) return error;

        /* Update the generator matrix. Unlike Levinson-Durbin's summing of
         * reflection coefficients, this loop could be executed in parallel
         * by p processors in constant time.
         */
        for (m = 0; m < P_MAX - i; m++) {
            G[1][m] = G[1][m + 1] + r * G[0][m];
            G[0][m] = G[1][m + 1] * r + G[0][m];
        }
    }
}
```

สำหรับโปรแกรมการวิเคราะห์แอลพีซีด้วยวิธีของ Schur ในภาษาแอสเซมบลีของ ADSP2101 นั้นได้แสดงไว้ในส่วนภาคผนวก

1.3 <u>การแปลงอัตราส่วนล็อกแอเรีย</u> (log-area ratio transformation) Kondos (1995) ได้กล่าวว่าการตรวจสอบเสถียรภาพของตัวกรองจากสัมประสิทธิ์การสะท้อนทำได้โดยการตรวจสอบว่าค่าสมบูรณ์ของสัมประสิทธิ์จะต้องไม่มากไปกว่า 1 นั่นคือ ตัวกรองแอลพีซีจะมีเสถียรภาพเมื่อ

$$|K_i| \leq 1.0 \qquad (26)$$

โดยที่ K_i เป็นสัมประสิทธิ์การสะท้อนที่ i

ดังนั้นในการสร้างระบบสำหรับการบีบข้อมูลเสียงโดยทั่วไปจึงนิยมใช้สัมประสิทธิ์การสะท้อนมากกว่าสัมประสิทธิ์ทำนายเพราะสามารถตรวจสอบเสถียรภาพของตัวกรองได้นั่นเอง แต่อย่างไรก็ตาม สัมประสิทธิ์การสะท้อนก็ยังไม่เหมาะที่จะนำไปควอนไตซ์โดยตรงเนื่องจากสเป็กตรัลเซนซิทิวิตี (spectral sensitivity) ของ K_i มีค่าไม่สม่ำเสมอ หมายความว่าขนาดของ K_i ที่มีค่าใกล้หนึ่งต้องการความเที่ยงตรงในการควอนไตซ์มากกว่าค่า K_i ที่ห่างไปจากหนึ่ง ดังนั้นจึงมีความจำเป็นที่จะต้องใช้ฟังก์ชันบางชนิดในการแปลง K_i ให้มีการกระจายค่าอย่างสม่ำเสมอก่อนที่จะควอนไตซ์ ฟังก์ชันที่นิยมใช้ได้แก่ ฟังก์ชันอัตราส่วนล็อกแอเรีย (log-area ratio - LAR) และ ฟังก์ชันอินเวอร์สไซน์ (inverse sine - IS)

ฟังก์ชันอัตราส่วนล็อกแอเรียสำหรับแปลง K เป็นอัตราส่วนล็อกแอเรีย ได้แก่

$$g_i = \log\left(\frac{1-k_i}{1+k_i}\right) \qquad (27)$$

ส่วนฟังก์ชันที่ใช้ในการแปลงอัตราส่วนล็อกแอเรียเป็น K ได้แก่

$$k_i = \left(\frac{1-10^{g_i}}{1+10^{g_i}}\right) \qquad (28)$$

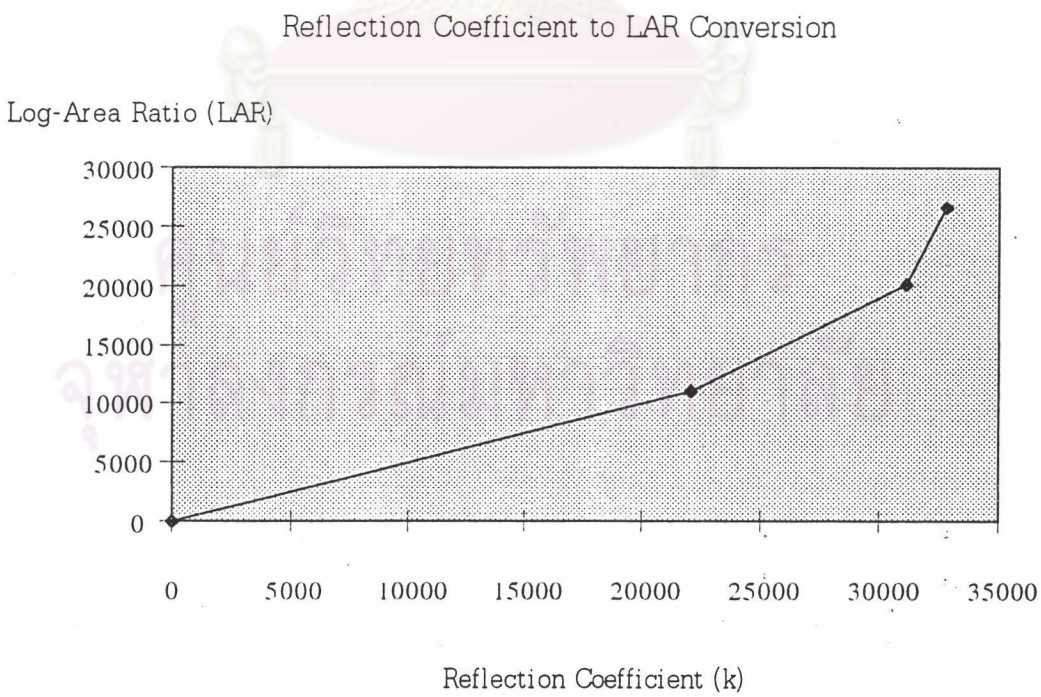
สำหรับการแปลงแบบอินเวอร์สไซน์ก็จะอาศัยฟังก์ชันอินเวอร์สไซน์และไซน์ตามลำดับ โดยสามารถเขียนเป็นสมการได้ดังต่อไปนี้

$$s_i = \sin^{-1}(k_i) \qquad (29)$$

สำหรับการแปลงสัมประสิทธิ์การสะท้อนไปเป็นรูปอินเวอร์สไซน์

$$k_i = \sin(s_i) \qquad (30)$$

สำหรับการแปลงรูปอินเวอร์สไซน์ไปเป็นสัมประสิทธิ์การสะท้อน



รูปที่ 3.3 แสดงการแปลงสัมประสิทธิ์การสะท้อนไปเป็นอัตราส่วนล็อกแอเรียโดยการประมาณค่า

Degener และ Bormann (1992) ได้ใช้วิธีล็อกการิธึมในแบบการประมาณค่าในงานวิจัยของเขา และวิธีการประมาณนี้ก็มีความเหมาะสมในการประมวลผลแบบทันทีซึ่งทำให้การคำนวณทำได้อย่างรวดเร็ว วิธีประมาณค่าของของฟังก์ชันดังกล่าวทำได้โดยการแบ่งค่าของสัมประสิทธิ์การสะท้อนเป็น 3 ช่วง แต่ละช่วงก็จะประมาณด้วยเส้นตรงหนึ่งเส้น โดยที่ความชันของเส้นตรงทั้ง 3 เส้นเพิ่มขึ้นตามลำดับ การประมาณค่าดังกล่าวแสดงเป็นความสัมพันธ์ระหว่าง LAR กับสัมประสิทธิ์การสะท้อนดังรูปที่ 3.3

1.4 ควอนไตเซชันและการเข้ารหัส (quantization and coding) เมื่อได้อัตราส่วนล็อกแอเรียมาแล้วขั้นตอนต่อไปคือการควอนไตซ์อัตราส่วนล็อกแอเรียและการเข้ารหัสเพื่อให้ได้ข้อมูลที่ใช้จำนวนบิตน้อย ๆ สำหรับการจัดเก็บหรือส่งไปในช่องนำสัญญาณ สมการที่ใช้ในการควอนไตซ์และเข้ารหัส LAR ที่ใช้ในการวิจัยคือ

$$quantized\_LAR = \frac{\frac{A * LAR}{32768} + B + 256}{512} \qquad (31)$$

โดยที่ LAR เป็นอัตราส่วนล็อกแอเรียที่ต้องการควอนไตซ์และเข้ารหัส
quantized_LAR เป็นอัตราส่วนล็อกแอเรียที่เข้ารหัสแล้ว

LAR#	A	B	ค่าสูงสุดของ quantizedLAR	ค่าต่ำสุดของ quantizedLAR	จำนวนบิตที่ต้องใช้
LAR1	20480	0	31	-32	6
LAR2	20480	0	31	-32	6
LAR3	20480	2048	15	-16	5
LAR4	20480	-2560	15	-16	5
LAR5	13964	94	7	-8	4
LAR6	15360	-1792	7	-8	4
LAR7	8534	-341	3	-4	3
LAR8	9036	-1144	3	-4	3

ตารางที่ 3.1 แสดงค่า A B ค่าสูงสุด ต่ำสุด และจำนวนบิตที่ใช้ในการควอนไตซ์และเข้ารหัสอัตราส่วนล็อกแอเรีย

ค่าของ A และ B สำหรับอัตราส่วนล็อกแอเรียทั้งแปดตัวนั้นได้แสดงไว้ในตารางที่ 3.1 ถ้าหากค่าที่ได้จากการคำนวณมีค่ามากกว่าค่าสูงสุดที่แสดงไว้ในตารางก็จะถูกจำกัดให้มีค่า

เท่ากับค่าสูงสุดในตาราง หรือในทำนองเดียวกันถ้าค่าที่คำนวณได้น้อยกว่าค่าต่ำสุดที่แสดงไว้ในตารางก็จะถูกจำกัดให้มีค่าเท่ากับค่าต่ำสุดในตารางเช่นกัน

2. ตัวกรองแอลพีซีย้อนกลับ (LPC inverse filter) ตัวกรองแอลพีซีย้อนกลับรับเฟรมสัญญาณเสียง และสัมประสิทธิ์การสะท้อนซึ่งอยู่ในรูปล็อกแอเรียที่เข้ารหัสไว้แล้ว (coded log-area ratio - LARc) ทั้งแปดจำนวนเข้ามา เพื่อใช้ในการสร้างสัญญาณเศษค้าง (residual signal) ในขั้นแรก LARc จะถูกแปลงกลับไปอยู่ในรูปอัตราส่วนล็อกแอเรีย (LAR) เสียก่อน และต่อจากนั้นจึงถูกแปลงกลับไปเป็นสัมประสิทธิ์การสะท้อน

การแปลง LARc ให้อยู่ในรูปอัตราส่วนล็อกแอเรียทำได้โดยใช้สมการดังต่อไปนี้

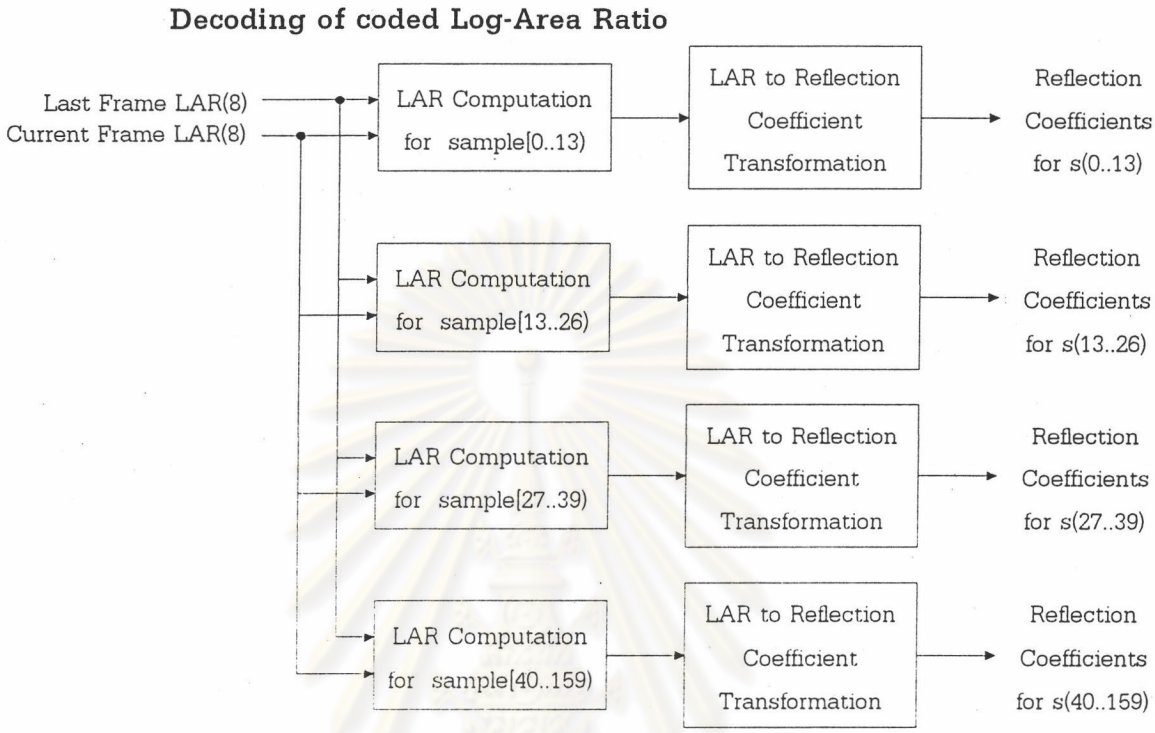
$$LAR = \frac{[(LARc + MIC) * 1024 - B * 2] * INVA}{16384} \qquad (32)$$

โดยที่ค่าของ B MIC และ INVA ของการคำนวณ LAR แต่ละตัวมีค่าดังตารางที่ 3.2

LAR#	B	MIC	INVA
LAR1	0	-32	13107
LAR2	0	-32	13107
LAR3	2048	-16	13107
LAR4	-2560	-16	13107
LAR5	94	-8	19223
LAR6	-1792	-8	17476
LAR7	-341	-4	31454
LAR8	-1144	-4	29708

ตารางที่ 3.2 แสดงค่า B MIC และ INVA ซึ่งใช้ในการแปลง LARc เป็น LAR

เมื่อได้ LAR ทั้งแปดตัวมาแล้วมันก็จะถูกนำไปเฉลี่ยกับ LAR ในเฟรมที่แล้วในอัตราส่วนที่เหมาะสมเพื่อคำนวณหาสัมประสิทธิ์การสะท้อนอีก 4 ชุด การวิเคราะห์และสังเคราะห์สัญญาณเสียงในแต่ละเฟรมก็จะใช้สัมประสิทธิ์การสะท้อนทั้งสี่ชุดมาทำการคำนวณ การทำเช่นนี้ทำให้สัญญาณที่ได้จากการสังเคราะห์มีความต่อเนื่องกลมกลืนมากขึ้นโดยเฉพาะในช่วงที่เป็นจุดเชื่อมต่อระหว่างเฟรม อัตราส่วนในการเฉลี่ยมีลักษณะดังนี้



รูปที่ 3.4 แสดงการคำนวณหาสัมประสิทธิ์การสะท้อนทั้ง 4 ชุดจาก LAR ในเฟรมปัจจุบันและ LAR ในเฟรมที่แล้ว

ให้ lLAR(i) เป็น LAR ตัวที่ i ของเฟรมที่แล้ว และ
pLAR(i) เป็น LAR ตัวที่ i ของเฟรมปัจจุบัน
s(m..n) เป็นสัญญาณเสียงในแอเรย์ตั้งแต่ตัวที่ m ถึงตัวที่ n

LAR ตัวที่ i ในช่วงที่ 1 สำหรับ s(0..13):

$$LAR(i)=0.75*lLAR(i) + 0.25*pLAR(i)$$

LAR ตัวที่ i ในช่วงที่ 2 สำหรับ s(13..26):

$$LAR(i)=0.50*lLAR(i) + 0.50*pLAR(i)$$

LAR ตัวที่ i ในช่วงที่ 3 สำหรับ s(27..39):

$$LAR(i)=0.25*lLAR(i) + 0.75*pLAR(i)$$

LAR ตัวที่ i ในช่วงที่ 4 สำหรับ s(40..159):

$$LAR(i)=0.00*lLAR(i) + 1.00*pLAR(i)$$

สัมประสิทธิ์การสะท้อนในแต่ละชุดจะถูกนำไปใช้ในตัวกรองผลึกเพื่อคำนวณหาสัญญาณเศษค้าง (residual signal) สำหรับแต่ละช่วงเฟรมย่อย

จากบทก่อนเราทราบว่าตัวกรองวิเคราะห์มีฟังก์ชันโอนย้ายดังนี้

$$A(z) = 1 - \sum_{i=1}^{P} a_i z^{-i}$$

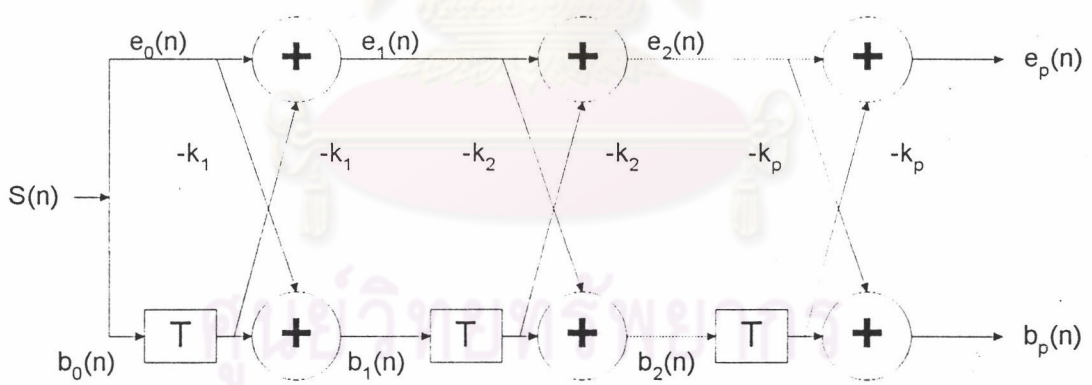
และสัญญาณเศษค้าง r(z) ที่ได้จากการกรองสัญญาณเสียงรับเข้าด้วยตัวกรองวิเคราะห์ A(z) คือ

$$r(z) = S(z)A(z)$$

โดยขั้นตอนวิธีของเลวินสัน-เดอบินทำให้ทราบความสัมพันธ์ระหว่างสัมประสิทธิ์การสะท้อนและสัมประสิทธิ์การทำนายดังนี้

$$\alpha'_j = \alpha_i - k'_i \alpha_{i-j} \quad j=1,2,3, \ldots, i-1$$

ด้วยการเริ่มต้นด้วยสมการข้างต้นสามสมการนี้ Rabiner และ Schafer ได้แสดงให้เห็นว่าตัวกรองวิเคราะห์ดังกล่าวสามารถสร้างขึ้นได้โดยใช้สัมประสิทธิ์การสะท้อนในตัวกรองผลึก ซึ่งมีแผนผังการไหลของสัญญาณ (signal flow diagram) ดังรูปที่ 3.5 (Rabiner และ Shafer, 1978 อ้างถึงใน Frerking, 1994)



รูปที่ 3.5 แผนผังการไหลของสัญญาณในตัวกรองวิเคราะห์แบบผลึก

จากในรูป b_i มีชื่อเรียกว่าลำดับค่าผิดพลาดของการทำนายย้อนกลับ (backward prediction error sequence) รูปเครื่องหมายบวกในวงกลมหมายถึงการรวมสัญญาณ ส่วนรูปอักษร T ในสี่เหลี่ยมหมายถึงการหน่วงสัญญาณในเชิงเวลาไปหนึ่งตัวอย่าง ดังนั้นจากรูป ลำดับค่าผิดพลาดของการทำนายย้อนกลับจึงมีค่าเป็น

$$b_i(n) = b_{i-1}(n-1) - k_i e_{i-1}(n) \qquad (33)$$

ส่วน e_i เรียกว่าลำดับค่าผิดพลาดของการทำนายไปข้างหน้า (forward prediction error sequence) ซึ่งก็คือสัญญาณเศษค้างที่เราต้องการหา จากรูป e_i มีค่าเป็น

$$e_i = e_{i-1}(n) - k_i b_{i-1}(n-1) \qquad (34)$$

จากสมการทั้งสองสามารถเขียนเป็นฟังก์ชันในภาษาซีซึ่งแสดงการคำนวณหาสัญญาณเศษค้างจากสัญญาณเสียงโดยใช้ตัวกรองวิเคราะห์แบบผลึกได้ดังต่อไปนี้

```
#define     P_MAX       8
static void short_term_analysis(
    double const * ref, /*  in:    [0...p-1] reflection coefficients   */
    int            n,   /*         # of samples                        */
    double const * in,  /*         [0...n-1] input samples             */
    double       * out) /* out:    [0...n-1] short-term residual       */
{
    double sav, s, ui; int i;
    static double u[P_MAX];

    while (n--) {
        sav = s = *in++;
        for (i = 0; i < P_MAX; i++) {
            ui   = u[i];
            u[i] = sav;

            sav  =  ui + ref[i] * s;
            s    =  s  + ref[i] * ui;
        }
        *out++ = s;
    }
}
```

สำหรับการคำนวณหาสัญญาณเศษค้างในภาษาแอสเซมบลีของ ADSP2101 นั้นได้แสดงไว้ในภาคผนวก เนื่องจากสัมประสิทธิ์การสะท้อน ref[i] ที่คำนวณได้จากวิธีเลวินสัน-เดอบินเป็นค่าลบดังนั้นในโปรแกรมจึงใช้การบวกแทนการลบ สำหรับขั้นตอนต่อไปเป็นการหาสัญญาณกระตุ้นจากสัญญาณเศษค้างซึ่งประกอบด้วยขั้นตอนย่อยสองขั้นตอนคือการวิเคราะห์ช่วงยาวและการกรองสัญญาณเศษค้างเพื่อหาสัญญาณกระตุ้น จะกล่าวต่อไปในหัวข้อ 3 และ 4 ตามลำดับ

3. <u>การวิเคราะห์ช่วงยาว (long term analysis)</u> การวิเคราะห์ช่วงยาวเป็นการวิเคราะห์เพื่อหาพารามิเตอร์ซึ่งจะต้องใช้ในตัวกรองย้อนกลับช่วงยาวหรือเรียกย่อว่าตัวกรองแอลทีไอ (long-term inverse filter - LTI) ก่อนอื่นจะกล่าวถึงตัวกรองทำนายช่วงยาวหรือตัวกรองแอลทีพี (long-term predictor filter-LTP) เสียก่อน ตัวกรองแอลทีพีเป็นตัวสร้างสัญญาณเศษค้างจากสัญญาณกระตุ้นและเป็นขั้นตอนการทำงานที่สำคัญขั้นตอนหนึ่งในการคลายข้อมูลเสียง ตัวกรองแอลทีพีซึ่งใช้สัญลักษณ์ 1/P(z) ถูกกำหนดด้วยสมการตัวกรองต่อไปนี้

ตัวกรองแอลทีพี: $$\frac{1}{P(z)} = \frac{1}{1 - \sum_{i=-I}^{I} b_i z^{-(D+i)}} \qquad (35)$$

โดยที่

D เป็นตัวชี้ไปยังตำแหน่งในแอเรย์ครอสโครีเลชันช่วงยาว (long-term cross correlation) ซึ่งปกติก็คือคาบของพิทช์ของสัญญาณเสียง หรือค่าทวีคูณของมัน ในบางครั้งก็เรียกว่าแล็ก (lag)

b_i เป็นสัมประสิทธิ์อัตราขยายแอลทีพี (LTP gain coefficient)

i เป็นจำนวนแทพ (tap) ของตัวกรอง ซึ่งทั่วไปมักจะให้ i มีค่าเป็นศูนย์นั่นคือมีหนึ่งแทพ หรือให้ i มีค่าเป็นหนึ่งนั่นคือมีสามแทพ ในการวิจัยนี้ เลือก i เป็นศูนย์

พารามิเตอร์แอลทีพีซึ่งก็คือ b_i และ D จะมีค่าแปรเปลี่ยนไปตามเวลาเช่นเดียวกับพารามิเตอร์ของตัวกรองช่วงสั้น Kondos (1995) ได้กล่าวถึงการปรับตัวตามเวลาของพารามิเตอร์แอลทีพี ว่าจะเปลี่ยนเร็วกว่าพารามิเตอร์ช่วงสั้น นั่นคือจะมีการแปรค่าในทุก ๆ 5 ถึง 10 ms นั่นเป็นสาเหตุที่ทำให้ต้องแบ่งเฟรมข้อมูลเสียงออกเป็นเฟรมย่อย (sub-frame) โดยแบ่งเป็นสี่ส่วนเท่า ๆ กัน ดังนั้นในแต่ละเฟรมย่อยจะมีตัวอย่างสัญญาณ 40 ตัวอย่าง หรือถ้าคิดเป็นช่วงเวลาก็จะได้เฟรมย่อยละ 5 ms เนื่องจากเฟรมข้อมูลกินเวลา 20 ms นั่นเอง การวิเคราะห์ช่วงยาวจะทำทีละเฟรมย่อยจนครบทั้งสี่เฟรมย่อยดังนั้นภายหลังการวิเคราะห์จะทำให้ได้พารามิเตอร์ช่วงยาวจำนวนสี่ชุดด้วยกัน

ตัวกรอง แอลทีพี หรือ 1/P(z) เป็นตัวกรองที่สร้างสัญญาณเศษค้างจากสัญญาณกระตุ้น ซึ่งมีบทบาทในการถอดรหัสสัญญาณเศษค้างกลับมาในตอนคลายข้อมูลเสียง จะได้กล่าวถึงตัวกรองแอลทีพีอีกครั้งในส่วนที่เกี่ยวกับการคลายข้อมูลเสียง สำหรับตัวกรองแอลทีไอก็คือส่วนกลับของตัวกรองแอลทีพีนั่นเอง การทำงานของตัวกรองแอลทีไอก็จะกลับกับตัวกรองแอลทีพี กล่าวคือมันจะรับสัญญาณเศษค้างเป็นอินพุทแล้วทำการคำนวณหาสัญญาณกระตุ้นก่อนที่จะหาพารามิเตอร์ช่วงยาว D หรือแล็ก จะต้องคำนวณหาสัญญาณครอสโครีเลชัน (cross corelation) ระหว่างสัญญาณเศษค้างของเฟรมย่อยปัจจุบันกับสัญญาณเศษค้างในเฟรมย่อยก่อนนี้ก่อน ค่าของครอสโครีเลชันถูกบรรจุในแอเรย์ที่มีขนาดเป็นสองเท่าของเฟรมย่อย ในที่นี้คือ 80 ดัชนีที่ชี้ไปยังค่าที่ใหญ่ที่สุดในแอเรย์นี้ก็คือ lag นั่นเอง สำหรับการหาสัมประสิทธิ์แอลทีพีหรือ b_i คำนวณได้จากสมการต่อไปนี้

$$b_i = \frac{\max\_cc}{energy} \qquad (36)$$

โดยที่ max_cc เป็นค่าสูงสุดในสัญญาณครอสโครีเลชัน แล็กจะถูกกำหนดให้เป็นดัชนีที่ชี้ไปยัง max_cc ส่วน energy เป็นค่าพลังงานของสัญญาณเศษค้าง ซึ่งก็คือผลรวมของสัญญาณเศษค้างแต่ละตัวยกกำลังสอง ในการคำนวณจริงสามารถใช้ฟังก์ชันในการคำนวณหาออโตโครีเลชันมาใช้ได้ โดย energy เป็นค่าของออโตโครีเลชันตัวแรกในแอเรย์ กรณีที่ energy มีค่าเป็นศูนย์ b_i จะถูกกำหนดให้มีค่าเป็นศูนย์ โปรแกรมภาษาซีต่อไปนี้แสดงการวิเคราะห์หาพารามิเตอร์ช่วงยาวดังกล่าว

```
#define     SUBWIN                  40
/* Calculate long-term prediction lag and gain. */
static void long_term_parameters(
    double const * d,           /*  in: [0.....SUBWIN-1] samples       */
    double const * prev,        /*  [-3*SUBWIN+1...0] past signal  */
    int          * lag_out,     /*  out: LTP lag                        */
    double       * gain_out)    /*  LTP gain                          */
{
    int      i, lag;
    double   cc[2 * SUBWIN], maxcc, energy;

    /* Find the maximum correlation with lags SUBWIN...3*SUBWIN-1
     * between this frame and the previous ones.
     */
    crosscorrelation(SUBWIN, d, prev - SUBWIN, 2 * SUBWIN, cc);
    maxcc = cc[lag = 0];

    for (i = 1; i < 2 * SUBWIN; i++)
        if (cc[i] > maxcc) maxcc = cc[lag = i];

    *lag_out = lag + SUBWIN;

    /* Calculate the gain from the maximum correlation and
     * the energy of the selected SUBWIN past samples.
     */
    autocorrelation(SUBWIN, prev - *lag_out, 1, &energy);
    *gain_out = energy ? maxcc / energy : 0.0;
}
/* Compute the autocorrelation
 *   ac(l) = Σ  x(i) * x(i-l)  for all i
 * for lags l between 0 and lag-1, and x(i) == 0 for i < 0 or i >= n
 */
void autocorrelation(
    int   n, double const * x,  /*  in: [0...n-1] samples x         */
    int lag, double       * ac) /* out: [0...lag-1] autocorrelation */
{
    double d; int i;
    while (lag--) {
        for (i = lag, d = 0; i < n; i++) d += x[i] * x[i-lag];
        ac[lag] = d;
    }
}
```

สำหรับโปรแกรมในการวิเคราะห์ช่วงยาวในภาษาแอสเซมบลีของ ADSP2101 นั้น ได้แสดงไว้ในภาคผนวก จากที่กล่าวมาจะเห็นว่าการวิเคราะห์ช่วงยาวในวิธีอาร์พีอี-แอลทีพี มีการคำนวณที่เชื่อมโยงกันระหว่างข้อมูลในเฟรมย่อยปัจจุบันและเฟรมย่อยก่อนหน้านี้ทำให้พารามิเตอร์ที่ได้มีความกลมกลืนต่อเนื่อง การทำเช่นนี้ทำให้ลดผลของความผิดเพี้ยนของสัญญาณในช่วงขอบของเฟรม เมื่อเปรียบเทียบกับโปรแกรม LPC.EXE ที่เขียนขึ้นมาสำหรับการศึกษาวิธีการบีบข้อมูลเสียงด้วยวิธีแอลพีซีที่จะกล่าวถึงในหัวข้อขั้นตอนการทำวิจัย ในโปรแกรมนั้นไม่มีการวิเคราะห์ช่วงยาว ผลของสัญญาณเสียงที่ได้จากโปรแกรม LPC.EXE ดังกล่าวจะให้สัญญาณที่มีความผิดเพี้ยนของสัญญาณค่อนข้างมากที่บริเวณรอยต่อของเฟรมการวิเคราะห์

4. <u>ตัวกรองย้อนกลับช่วงยาว (long-term inverse filter)</u> จากรูปที่ 3.1 สัญญาณเศษค้างซึ่งได้จากการกรองสัญญาณเสียงรับเข้าด้วยตัวกรองแอลพีซีย้อนกลับจะถูกกรองอีกครั้งด้วยตัวกรองแอลทีไอ จากหัวข้อก่อนสามารถเขียนตัวกรองแอลทีไอซึ่งเป็นส่วนกลับของตัวกรองแอลทีพีได้ดังนี้

ตัวกรองแอลทีไอ : $$P(z) = 1 - bz^{-lag} \qquad (37)$$

โดยที่ตัวกรองแอลทีไอถูกเขียนให้อยู่ในรูปที่ง่ายขึ้นโดยการกำหนดให้แทพของตัวกรองหรือ i มีค่าเป็นศูนย์ ดังนั้นสัมประสิทธิ์ แอลทีพี จึงมีเพียง b ตัวเดียว และในที่นี้แทน D ด้วย lag สมการข้างบนสามารถในรูปเชิงเวลาเพื่อหาสัญญาณกระตุ้นได้ดังนี้

$$y(n) = x(n) - bx(n - lag) \qquad (38)$$

โดยที่

y(n) เป็นสัญญาณกระตุ้นที่ต้องการหา

x(n) เป็นสัญญาณเศษค้าง

5. <u>การกรองสัญญาณแบบเวทและการเลือกกริด (weighting filter and RPE grid selection)</u> สัญญาณกระตุ้นที่ได้หลังจากผ่านตัวกรองแอลทีไอ จะถูกนำไปเข้าตัวกรองเวท (weighting filter) สำหรับช่วยลดสัญญาณรบกวน ตัวกรองเวทเป็นตัวกรองในประเภทที่เรียกว่า FIR (finite impulse response) มีสมการที่อธิบายการทำงานของมันดังนี้

$$Y(z) = \sum_{i=0}^{N-1} h_i X(z) z^{-i}$$

โดยที่

X(z) เป็นสัญญาณอินพุท

Y(z) เป็นสัญญาณเอาต์พุทที่ได้หลังจากผ่านตัวกรอง FIR

h_i เป็นสัมประสิทธิ์ที่ i ของตัวกรอง

N เป็นจำนวนแทพของตัวกรอง

สมการตัวกรองดังกล่าวเขียนในรูปเชิงเวลาได้เป็น

$$y(n) = \sum_{i=0}^{N-1} h_i x(n-i)$$

สำหรับสัมประสิทธิ์ของตัวกรองที่ในการวิจัยในครั้งมีจำนวนทั้งสิ้น 11 ตัวนั่นคือ N=11 สำหรับค่าของสัมประสิทธิ์แต่ละตัวได้แสดงไว้ในตารางที่ 3.3 ค่าสัมประสิทธิ์ดังกล่าวเป็นจำนวนทศนิยมตายตัว (fixed point) ซึ่งใช้ในการคำนวณแบบทันที จำนวนที่สอดคล้องกันเมื่อเขียนในรูปทศนิยมจะต้องเปรียบเทียบกับ 32768 นั่นคือสัมประสิทธิ์เหล่านี้มีค่าอยู่ในช่วง -1.00 ถึง +1.00

i	0	1	2	3	4	5	6	7	8	9	10
h_i	-134	-374	0	2054	5741	8192	5741	2054	0	-374	-134

ตารางที่ 3.3 แสดงสัมประสิทธิ์ของตัวกรองเวท

สัญญาณกระตุ้นที่ผ่านตัวกรองเวทแล้วจะถูกนำไปคำนวณเพื่อหากริด (grid) ซึ่งเป็นดัชนีชี้ไปยังตำแหน่งในแอเรย์ของสัญญาณกระตุ้น x(n) โดยกริดจะมีค่าได้ตั้งแต่ศูนย์ถึงสอง การเลือกค่ากริดจะต้องทำการแบ่งสัญญาณกระตุ้นเป็นสามชุดด้วยกัน แต่ละชุดสัญญาณประกอบด้วยข้อมูลในแอเรย์ของสัญญาณกระตุ้นจำนวนชุดละ 13 ตัวอย่างสัญญาณดังนี้

ชุดแรกประกอบด้วยข้อมูลที่ชี้ด้วย

0 3 6 9 12 15 18 21 24 27 30 33 36

ชุดที่สองประกอบด้วยข้อมูลที่ชี้ด้วย

1 4 7 10 13 16 19 22 25 28 31 34 37

ชุดที่สามประกอบด้วยข้อมูลที่ชี้ด้วย

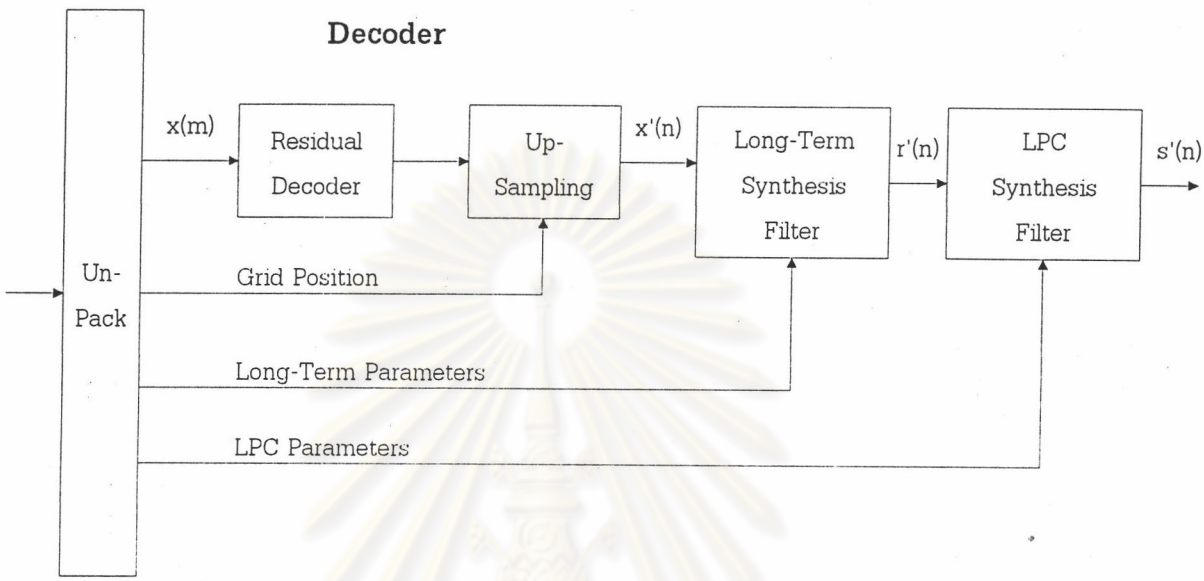
2 5 8 11 14 17 20 23 26 29 32 35 38

การนำสัญญาณกระตุ้นไปเข้ารหัสจะใช้สัญญาณเพียงชุดเดียวจากสามชุดข้างบน การเลือกว่าจะใช้สัญญาณชุดไหนจะพิจารณาจากชุดสัญญาณที่มีผลต่อการคลายสัญญาณกลับมากที่สุดนั่นคือจะเลือกชุดสัญญาณที่มีความแรงของสัญญาณมากที่สุด ความแรงของสัญญาณคำนวณได้จากกำลังของสัญญาณหรือผลรวมของทุกตัวอย่างสัญญาณยกกำลังสอง เมื่อคำนวณหาชุดสัญญาณได้แล้วก็จะกำหนดให้กริดเท่ากับตำแหน่งเริ่มต้นของสัญญาณชุดนั้น หรืออีกนัยหนึ่งกริดจะเป็นตัวชี้ไปยังจุดเริ่มต้นของชุดสัญญาณที่เลือก ฟังก์ชันสำหรับเลือกกริดเป็นดังนี้

```
void RPE_grid_selection(double *x, double xM[], int *Mc_out) {
        /* x[40]    = input excitation signal */
        /* xM[13]   = output down-sampling excitation signal */
        /* Mc_out   = grid selection */
        int m, i, Mc;
        double EM, result, temp;

        EM=0.0;
        Mc=0;

        for( m=0; m<3; m++ ) {
                result=0;
                for( i=0; i<13; i++) {
                        temp = x[m+3*i];
                        result = result + temp * temp;
                }
                if( result > EM ) {
                        Mc = m;
                        EM = result;
                }
        }

        /* down-sampling by factor of 3 to get the selected xM[0..12] */

        for( i=0; i<13; i++)
                xM[i] = x[Mc + 3*i];
        *Mc_out = Mc;
}
```

6. <u>การควอนไตซ์สัญญาณ (APCM quantiser)</u> ขั้นตอนสุดท้ายเป็นการควอนไตซ์สัญญาณกระตุ้นเพื่อให้ได้รหัสสัญญาณที่ใช้จำนวนบิตน้อย ๆ การควอนไตซ์สัญญาณกระตุ้นใช้วิธีการเข้ารหัสเอพีซีเอ็ม (Adaptive Pulse Code Modulation - APCM) วิธีการนี้มีหลักการใกล้เคียงกับวิธีพีซีเอ็มแต่ต่างกันที่วิธีพีซีเอ็มมีขนาดขั้นของการควอนไตซ์ (quantization step) ที่ตายตัว แต่วิธีเอพีซีเอ็มมีขนาดขั้นของการควอนไตซ์ปรับเปลี่ยนไปตามระดับขนาดของสัญญาณ โดยที่ถ้าสัญญาณมีขนาดสูงขนาดขั้นก็จะมีค่าสูง ถ้าสัญญาณมีขนาดต่ำขนาดขั้นก็จะต่ำตามไปด้วย การที่ขนาดขั้นของการควอนไตซ์มีขนาดปรับเปลี่ยนไปตามสัญญาณที่เข้ามาทำให้สัญญาณที่ได้หลังจากการดีควอนไตซ์ (dequantization) มีความเที่ยงตรงมากขึ้น

การคลายข้อมูลเสียงที่บีบด้วยวิธี อาร์พีอี-แอลทีพี



รูปที่ 3.6 แสดงการคลายข้อมูลเสียงที่บีบด้วยวิธีอาร์พีอี-แอลทีพี

ในรูปที่ 3.6 แสดงวิธีการคลายข้อมูลเสียงที่บีบด้วยวิธีอาร์พี-แอลทีพี รหัสข้อมูลของพารามิเตอร์ต่าง ๆ จะถูกถอดรหัสกลับมาส่วนรหัสสัญญาณกระตุ้นจะถูกดีควอนไตซ์เพื่อสร้างสัญญาณกระตุ้น x'(n) คืนมาและถูกนำไปเข้าตัวกรองสังเคราะห์ช่วงยาว (Long-Term Synthesis Filter) เพื่อสร้างสัญญาณเศษค้าง r'(n) สัญญาณ r'(n) ที่ได้จะถูกนำไปกรองอีกครั้งด้วยตัวกรองสังเคราะห์แอลพีซี (LPC synthesis filter) และในที่สุดก็ได้สัญญาณเสียง s'(n) กลับคืนมา รายละเอียดการทำงานในแต่ละขั้นตอนมีดังนี้

1. การถอดรหัสสัญญาณกระตุ้น (Residual Decoder) ขั้นตอนนี้เป็นการนำรหัสข้อมูลที่ได้จากการเข้ารหัสด้วยวิธีเอพีซีเอ็มดังที่ได้กล่าวไว้ใน 1.6 มาทำการดีควอนไตซ์เพื่อให้ได้สัญญาณกระตุ้นคืนมา ชุดของตัวอย่างสัญญาณที่ได้จากการดีควอนไตซ์มีจำนวน 13 ตัวอย่างสัญญาณด้วยกัน

2. การสร้างสัญญาณกระตุ้น (Up Sampling) สัญญาณกระตุ้นที่ได้ในหัวข้อ 3.1 จะถูกนำไปสร้างแอเรย์ของสัญญาณกระตุ้นที่มีขนาด 40 ช่อง แอเรย์ดังกล่าวถูกหนดให้มีค่าเป็นศูนย์ในทุกช่องเมื่อตอนเริ่มต้น และถูกใส่ค่าของสัญญาณกระตุ้นลงไปในตำแหน่งที่เหมาะสม ตำแหน่งที่เริ่มใส่ค่าถูกกำหนดโดยกริดโดยกริดอาจมีค่าได้ตั้งแต่ศูนย์ถึงสอง ตำแหน่งถัดไปที่จะใส่ค่าจะห่างไปอีกสามช่อง และจะใส่ค่าสัญญาณกระตุ้นลงไปในแอเรย์เรื่อยไปตามลำดับจนครบจำนวน ขั้นตอนการทำงานของการสร้างสัญญาณกระตุ้นสามารถแสดงด้วยฟังก์ชันภาษาซีดังต่อไปนี้

```
void RPE_grid_positioning( int Mc, double *xMp, double *ep) {
          /* input:  Mc          --> Gird */
          /*         xMp         --> excitation samples */
          /* output: ep          --> excitation signal */
          int i, k;

          for( k=0; k<40; k++ )
                    ep[k] = 0.0;          /* clear excitation signal */
          for( i=0; i<13; i++ )
                    ep[ Mc + 3*i ] = xMp[i];
}
```

3. <u>การสังเคราะห์ช่วงยาว</u> (Long-Term Synthesis Filter) การสังเคราะห์ช่วงยาวจะทำให้ได้สัญญาณเศษค้าง โดยการนำสัญญาณกระตุ้นที่ได้จากหัวข้อก่อนมาผ่านตัวกรองแอลทีพี ซึ่งเป็นส่วนกลับของตัวกรองแอลทีไอตามที่กล่าวมาแล้วนั่นคือ

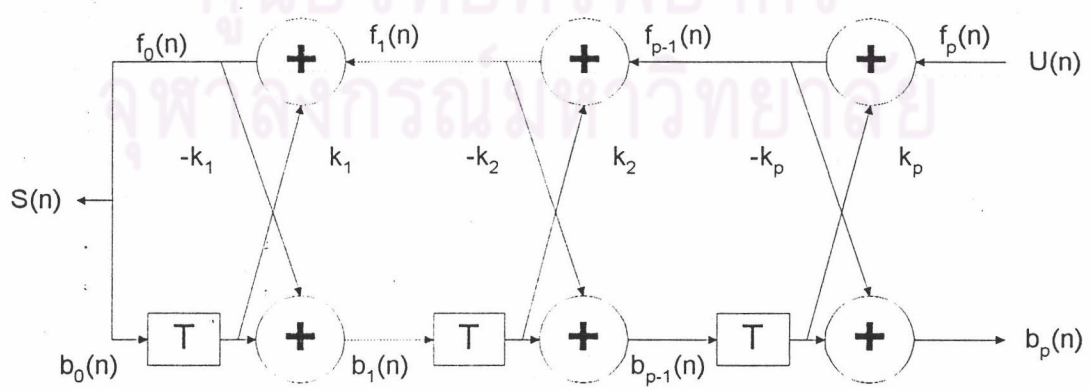
ตัวกรองแอลทีพี: $$\frac{1}{P(z)} = \frac{1}{1 - bz^{-lag}}$$

หรือเขียนในรูปเชิงเวลาได้ว่า

$$y(n) = x(n) + by(n - lag)$$

โดยที่ y(n) เป็นสัญญาณเศษค้างที่ต้องการหาและ x(n) เป็นสัญญาณกระตุ้น

4. <u>การสังเคราะห์แอลพีซี</u> (LPC Synthesis Filter) ขั้นตอนนี้สัญญาณเศษค้างจะถูกกรองด้วยตัวกรองผลึกเพื่อสร้างเป็นสัญญาณเสียงคืนมา ในรูปที่ 3.7 แสดงการไหลของสัญญาณผ่านตัวกรองผลึก ในทำนองเดียวกันกับขั้นตอนการวิเคราะห์หาสัญญาณเศษค้างเราสามารถหา 'กำลังข้างหน้า' (forward power) f_i และ 'กำลังย้อนกลับ' (backward power) b_i ได้จากแผนผังการไหลของสัญญาณดังนี้



รูปที่ 3.7 แผนผังการไหลของสัญญาณในตัวกรองสังเคราะห์ที่สร้างโดยตัวกรองผลึก

$$f_{i-1}(n) = f_i(n) + k_i b_{i-1}(n-1) \qquad \text{สำหรับ i=P, P-1, ...,2,1 (39)}$$

และ

$$b_i(n) = b_{i-1}(n-1) - k_i f_{i-1}(n) \qquad \text{สำหรับ i=1,2, ..., P} \qquad (40)$$

การคำนวณเริ่มต้นโดยการกำหนดค่า $f_p(n)$ ให้มีค่าเท่ากับสัญญาณเศษค้าง U(n) แล้วคำนวณหา $f_{p-1}(n)$ $b_p(n)$ $f_{p-2}(n)$ $b_{p-1}(n)$ $f_{p-3}(n)$ ... เรื่อยไปจนถึง $f_0(n)$ ก็จะได้ตัวอย่างสัญญาณเสียงขาออก s(n) ซึ่งมีค่าเท่ากับ $f_0(n)$ ฟังก์ชันในภาษาซีที่แสดงการคำนวณของตัวกรองสังเคราะห์แลตติสเป็นดังนี้

```
static void short_term_synthesis(
    double const * ref,          /*  in: [0...p-1] reflection coefficients   */
    int            n,            /*      # of samples                        */
    double const * in,           /*      [0...n-1] residual input            */
    double       * out)          /* out: [0...n-1] short-term signal         */
{
    double s; int i;
    static double u[P_MAX+1];

    while (n--) {
        s = *in++;
        for (i = P_MAX; i--;) {
            s     -= ref[i] * u[i];
            u[i+1] = ref[i] * s + u[i];
        }
        *out++ = u[0] = s;
    }
}
```