บทที่ 4

การพัฒนาตัวประมวลผลระดับต้นของไฟไนต์เอลิเมนต์ซึ่งนำเข้าข้อมูลพิกัดจากภาพ

วิธีการไฟไนต์เอลิเมนต์เป็นวิธีเชิงเลขที่มีการนำไปใช้กว้างขวางมากที่สุดในทางวิศวกรรม เนื่องจากสามารถใช้ได้กับปัญหาที่ลักษณะรูปร่างซับซ้อน อาจประกอบด้วยวัสดุหลายชนิด หรือมีสภาวะขอบเขตและโหลดอย่างไรก็ได้ ทำให้ผู้ออกแบบสามารถศึกษาการแปรค่าพารามิเตอร์ต่าง ๆ เช่น รูปร่าง วัสดุ โหลด เพื่อคัดเลือกแบบที่ดีและเหมาะสมที่สุด จากความสามารถในการใช้งานที่กว้างขวางเหล่านี้ทำให้วิธีการไฟไนต์เอลิเมนต์มีความเหมาะสมที่จะใช้เป็นเครื่องมือในการออกแบบและวิเคราะห์ปัญหาทางวิศวกรรมที่นับวันจะมีความซับซ้อนมากขึ้น (ปราโมทย์ เดชะอำไพ, 2537)

โปรแกรมไฟไนต์เอลิเมนต์ต่าง ๆ แม้จะดูแตกต่างกัน แต่จะมีโมดูลต่าง ๆ คล้ายกันดังนี้

1. โมดูลจุดต่อ (node module) ทำหน้าที่กำหนดพิกัดให้แก่จุดต่อ รวมทั้งองศาความเป็นอิสระ (degree of freedom)

2. โมดูลสภาวะขอบ (boundary conditions module) ทำหน้าที่กำหนดสภาวะขอบเขตให้แก่จุดต่อที่ถูกต้อง

3. โมดูลคุณสมบัติวัสดุ (material properties module) เก็บคุณสมบัติของวัสดุชนิดต่าง ๆ ไว้ในห้องสมุดวัสดุ (materials library)

4. โมดูลเอลิเมนต์ (element module) เก็บเอลิเมนต์ชนิดต่าง ๆ ไว้ในห้องสมุดเอลิเมนต์ คำนวณเมตริกซ์ของเอลิเมนต์ต่าง ๆ (element matrix) และประกอบเมตริกซ์เหล่านั้นเข้าในเมตริกซ์รวมใหญ่

5. โมดูลโหลด (loading module) ทำหน้าที่กำหนดโหลดที่ได้รับจากภายนอกให้แก่จุดต่อและเอลิเมนต์ที่เหมาะสม

6. โมดูลคำตอบ (solution module) ทำหน้าที่หาคำตอบของระบบสมการที่ได้จากกระบวนการประกอบของโมดูลเอลิเมนต์

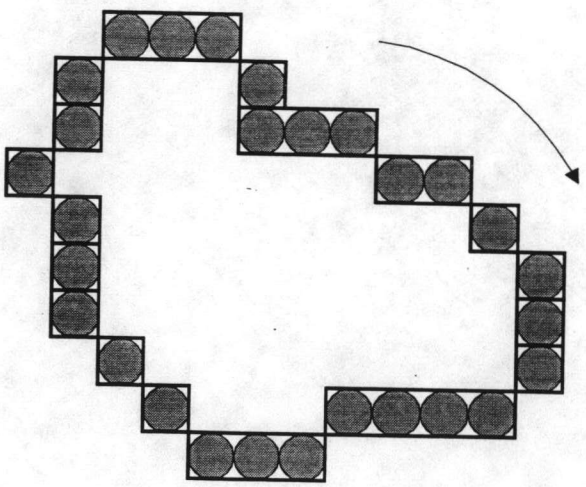
ในช่วงก่อนปี 1970 กระบวนการวิเคราะห์ด้วยวิธีไฟไนต์เอลิเมนต์ส่วนใหญ่จะเสียเวลาและค่าใช้จ่ายจำนวนมากไปกับการสร้างและเตรียมเมชด้วยมือ หลังจากนั้นจึงมีการพัฒนาตัว

ประมวลผลระดับต้นและตัวประมวลผลระดับปลายในช่วงกลางของทศวรรษ 1970 การประมวลผลระดับต้น (preprocessing) โดยใช้ตัวประมวลผลระดับต้น (preprocessor) จะช่วยให้ผู้ใช้ทำงานได้อย่างสะดวกในการป้อนข้อมูลซึ่งเกี่ยวข้องกับ 5 โมดูลแรกที่กล่าวมา กระบวนการประมวลผลระดับปลาย (postprocessor) เป็นส่วนของการแสดงผลลัพธ์ในรูปของกราฟฟิกเพื่อช่วยเสริมในด้านความเข้าใจและการตีความหลังการวิเคราะห์แก้สมการหาค่าต่าง ๆ แล้ว (Zeid, 1991)

ในบทนี้จะกล่าวถึงหลักการและทฤษฎีที่เกี่ยวข้องในการสร้างตัวประมวลผลระดับต้นอย่างง่ายที่ใช้ในงานวิจัยนี้

การตามรอยขอบเขตพื้นที่เพื่อให้ได้จุดแนวขอบ

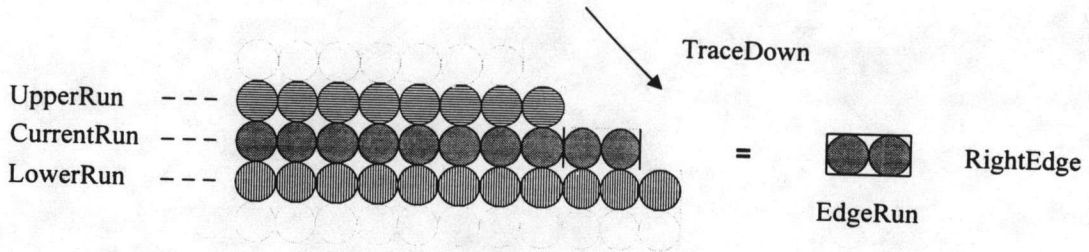
ในระหว่างการระบายนั้นจะเก็บข้อมูลของช่วงแนวนอน (run) ต่างๆไว้ด้วย กลุ่มของจุดที่จุดปลายทั้งสองข้าง X1 และ X2 จะเป็นส่วนหนึ่งของเส้นขอบเขตวัตถุนั่นเอง เส้นขอบวัตถุที่สามารถนำมาใช้ได้นั้นจะต้องมีการเรียงลำดับข้อมูลอย่างถูกต้อง ในงานวิจัยนี้จะใช้ลำดับข้อมูลวนตามเข็มนาฬิกา อัลกอริทึมจะเริ่มจากช่วงแนวนอนสุดท้ายในบรรทัด (line) ที่เป็นจุดเริ่มต้นของการระบาย



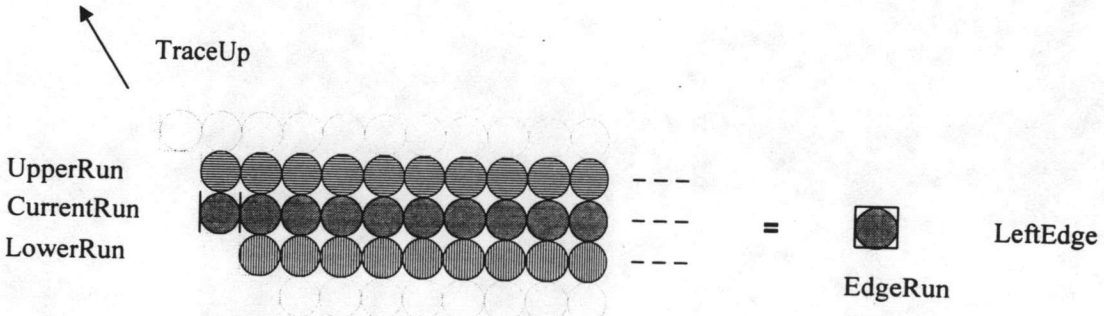
รูปที่ 4.1 ช่วงแนวนอนของเส้นขอบซึ่งมีลำดับข้อมูลวนตามเข็มนาฬิกา

อัลกอริทึมจะตรวจหาช่วงแนวนอนที่มีส่วนเกยกัน (intersect) ทางด้านบนและด้านล่างโดยเริ่มในทิศลงด้านล่างก่อน เรียกว่าการตามรอยในทิศลง (TraceDown) เลือกเอาข้อมูลของช่วงแนวนอนด้านล่างเฉพาะในส่วนที่ถัดจากค่า X2 ของช่วงแนวนอนที่กำลังทำการตรวจสอบไปจนสุดด้านขวา(ในที่นี้จะเรียกว่าเป็นช่วงแนวนอนด้านขวาของช่วงแนวนอนที่กำลังทำการตรวจสอบ) เมื่อเก็บข้อมูลช่วงแนวนอนด้านขวานั้นเข้า list แล้วก็จะเลื่อนการดำเนินการนั้นมาที่ช่วงแนวนอนที่เกยกันด้านใต้แล้วเริ่มดำเนินการเช่นเดียวกันที่กล่าวมา เมื่อดำเนินการจนไม่มีช่วงแนวนอนเกยกันด้านล่างแล้วก็จะตรวจหาช่วงแนวนอนที่อยู่เกยกันด้านบน เลือกเฉพาะช่วงข้อมูลที่อยู่ทางซ้ายของช่วงแนวนอนที่กำลังตรวจสอบไปจนถึงจุดเริ่มต้นของช่วงแนวนอนด้านบน (ในที่นี้จะเรียกว่าช่วงแนวนอนด้านซ้าย) เก็บช่วงแนวนอนด้านซ้ายนี้เข้า list ทิศทางนี้เรียกว่าการตามรอยในทิศขึ้น (TraceUp) ดำเนินการในทิศทางนี้ไปจะกระทั่งไม่มีช่วงแนวนอนที่เกยกันด้านบนจึงเปลี่ยนมาดำเนินการในทิศลง เมื่ออัลกอริทึมทำงานมาจนครบรอบที่จุดเริ่มต้นแล้วก็จะหยุด ข้อมูลที่อยู่ใน list จะเป็นข้อมูลของช่วงแนวนอนที่เป็นเส้นขอบของวัตถุที่มีการเรียงลำดับข้อมูลในทิศตามเข็มนาฬิกา

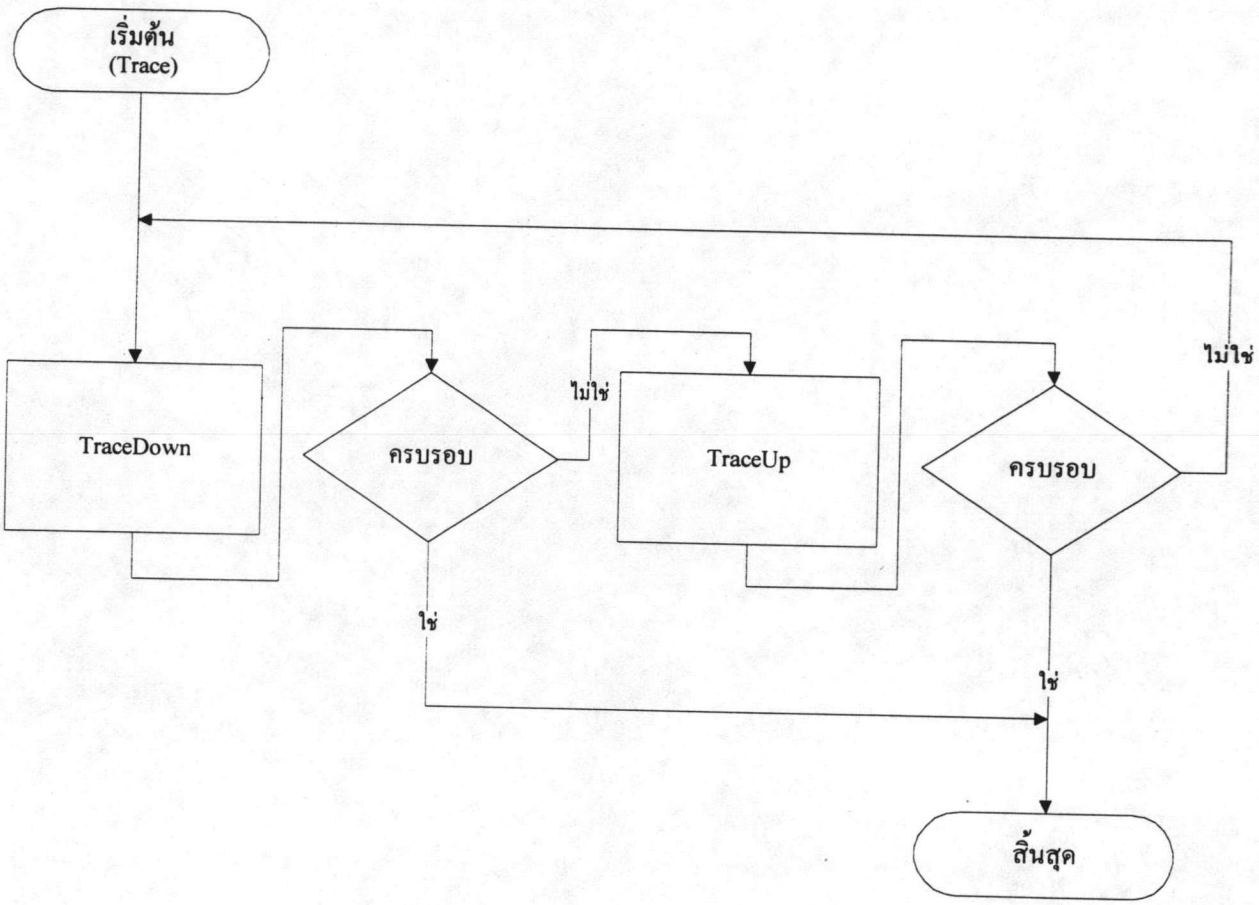
แผนภูมิแสดงการทำงานของอัลกอริทึมที่กล่าวมาแสดงในรูปที่ 4.4, 4.5 และ 4.6



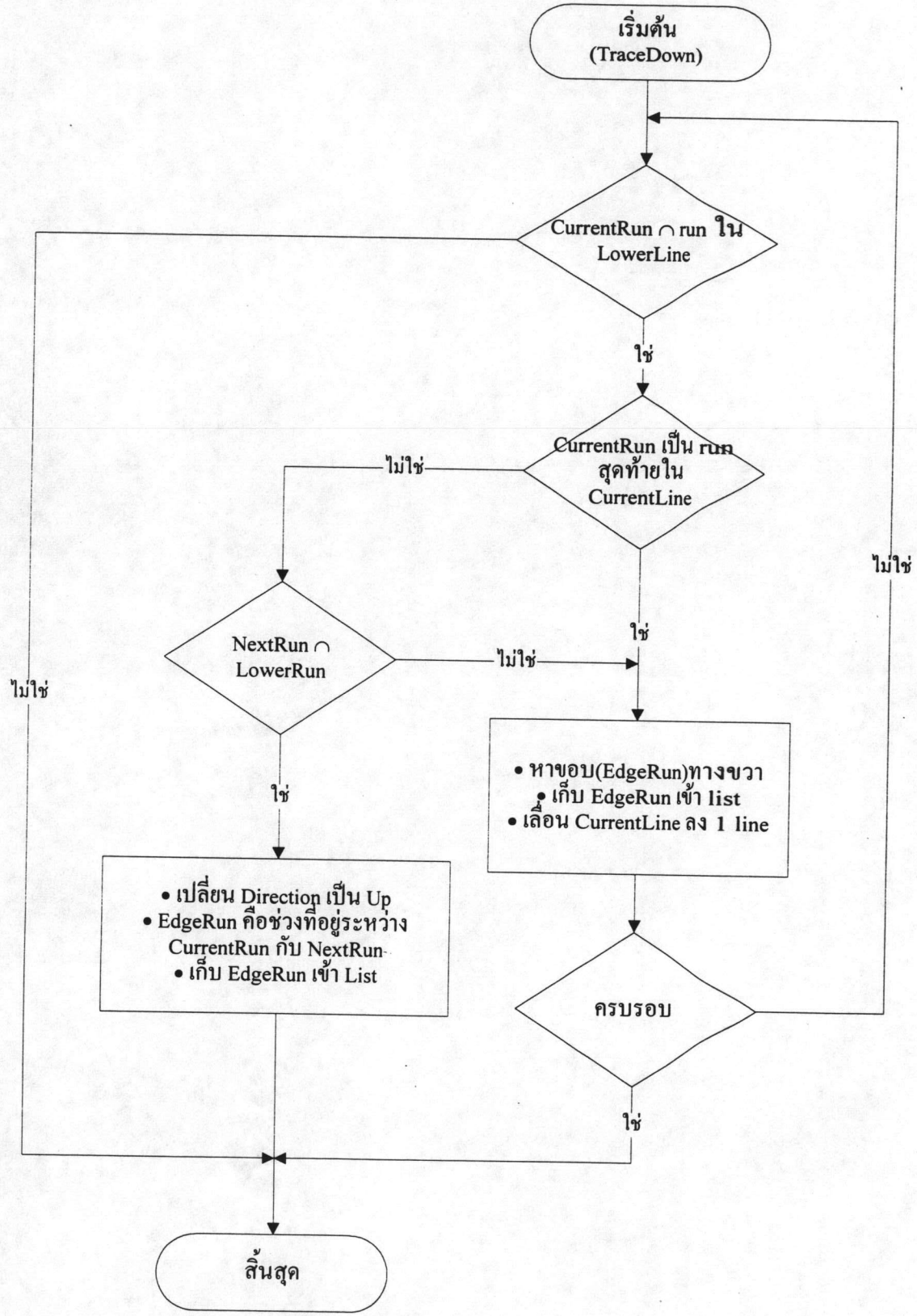
รูปที่ 4.2 การตามรอยในทิศลง (TraceDown)



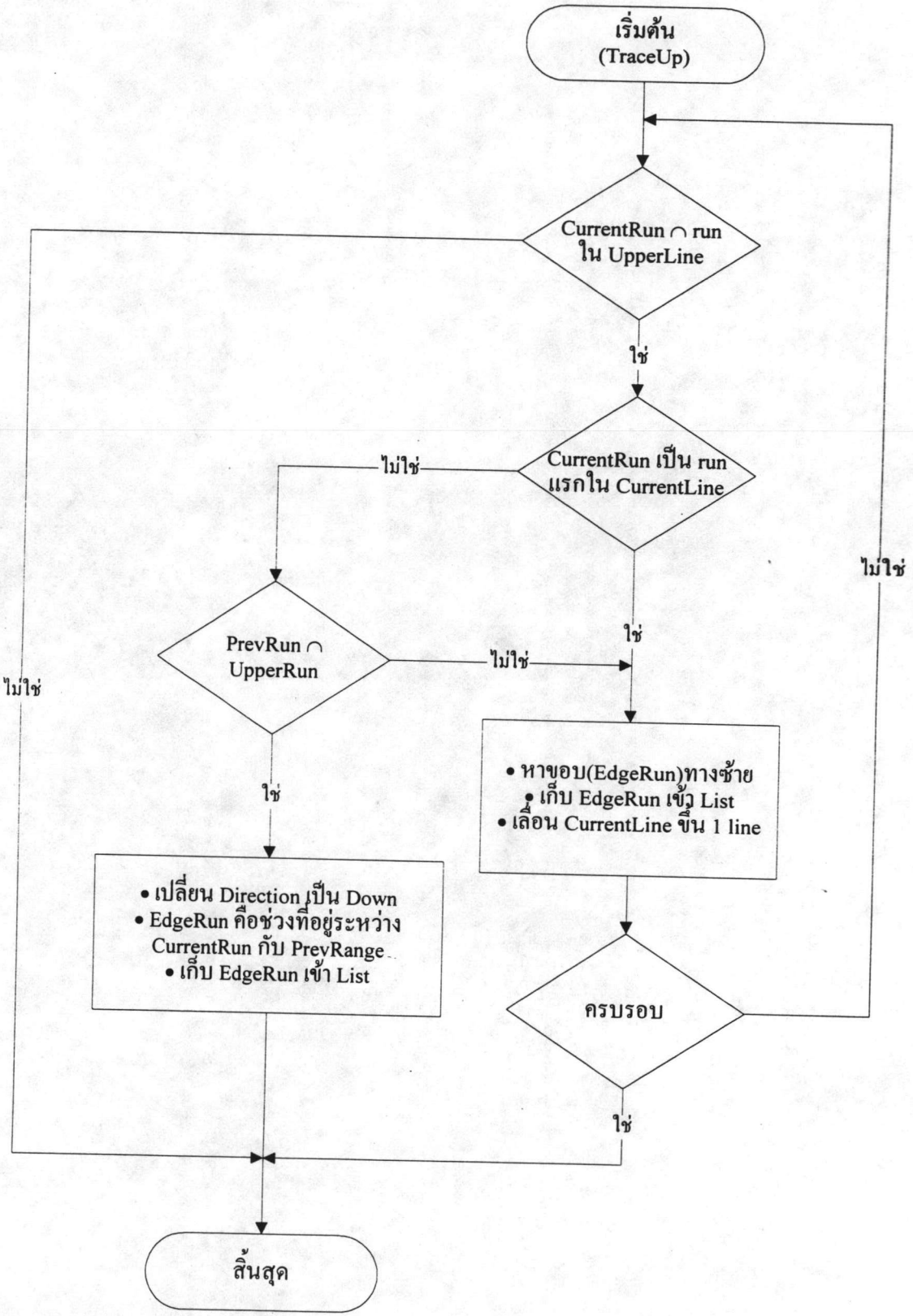
รูปที่ 4.3 การตามรอยในทิศขึ้น (TraceUp)



รูปที่ 4.4 แผนภูมิแสดงการตามรอยขอบ



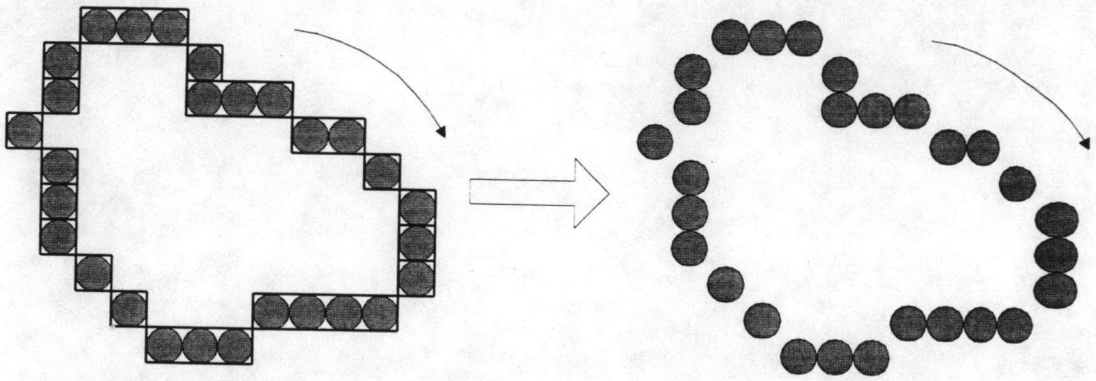
รูปที่ 4.5 แผนภูมิการตามรอยในทิศลง (TraceDown)



รูปที่ 4.6 แผนภูมิการตามรอยในทิศขึ้น (TraceUp)

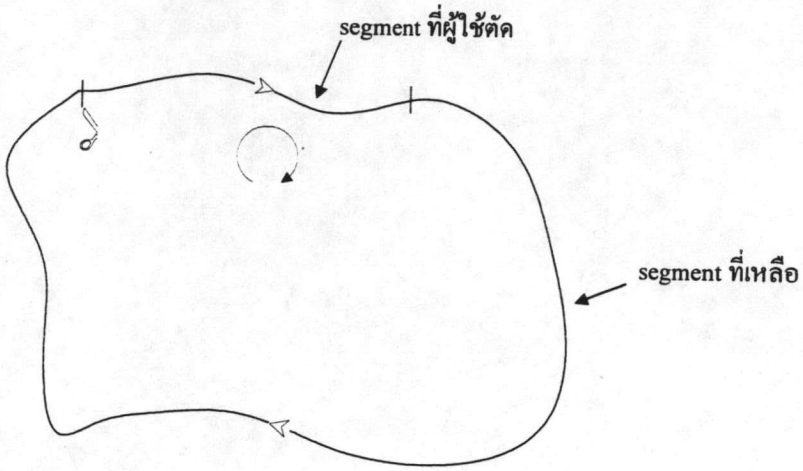
การตัดแนวขอบเป็นส่วนย่อย (segmentation)

ข้อมูลที่ได้จากการตามรอยแล้วจะอยู่ในรูปของช่วงแนวนอนที่เป็นเส้นขอบเขตของวัตถุ แต่ยังไม่สามารถนำมาใช้ได้เนื่องจากการนำมาใช้ต้องการอยู่ในรูปของจุดที่เรียงกัน list ของช่วงแนวนอนที่ได้ (EdgeList) จะต้องถูกตามรอยซ้ำเพื่อให้ได้ list ของจุด (PointList) เพื่อนำมาใช้เป็นเส้นขอบเขตวัตถุที่ผู้ใช้โปรแกรม (PPROCESS.EXE) สามารถตัดเอาส่วนใดมาใช้ก็ได้ โครงสร้างข้อมูลของ PointList เป็น circular double link-list ทำให้การเข้าถึงข้อมูลเป็นไปได้ 2 ทิศทางคือ ไป-กลับและสามารถท่องไปได้อีกเมื่อไปจนถึงปลายข้อมูลข้างหนึ่งแล้ว ทิศทางไปข้างหน้าถูกกำหนดไว้ให้ตรงกับทิศทางตามเข็มนาฬิกาและทิศทางย้อนกลับตรงกับทวนเข็มนาฬิกา



รูปที่ 4.7 การแปลงข้อมูลจาก list ของช่วงแนวนอนเป็น Pointlist

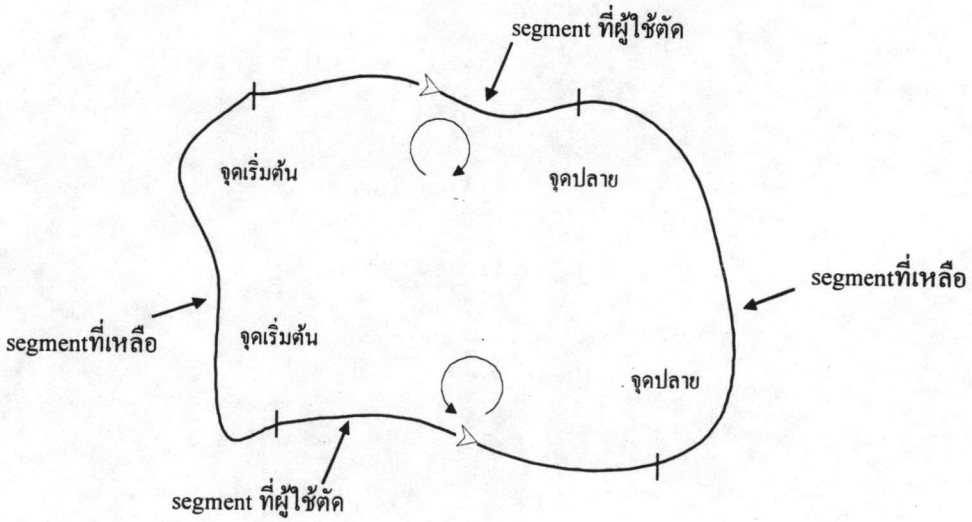
เส้นขอบของวัตถุที่ได้นี้จะเป็นจุดต่อเนื่องกันไปเป็นเส้นเดียว มีทิศตามเข็มนาฬิกา ผู้ใช้โปรแกรมสามารถตัดข้อมูลนี้เป็นส่วน ๆ เพื่อให้ได้ข้อมูลเฉพาะในส่วนที่จะวิเคราะห์ เครื่องมือที่ใช้คือ tailor tool ที่ปรากฏในแถบเครื่องมือ (tool bar) ในการตัดจะต้องกำหนดทิศทางของการตามรอยข้อมูลเป็นทวนเข็มนาฬิกา หรือตามเข็มนาฬิกา เส้นของข้อมูลที่ตัดออกมาในงานวิจัยนี้จะเรียกว่าส่วนย่อย (segment) วงรอบของข้อมูลที่มีโครงสร้างเป็นวงกลมจะกลายสภาพเป็นส่วนย่อย จะเห็นได้ว่าในการตัดครั้งแรก วงข้อมูลของวัตถุจะกลายสภาพเป็นส่วนย่อย 2 อัน อันแรกเป็นส่วนย่อยที่ผู้ใช้ตัดออกมาเพื่อนำไปใช้งาน อีกอันหนึ่งเป็นส่วนย่อยที่เหลือจากการตัด



รูปที่ 4.8 การตัดแนวขอบ

ส่วนย่อยที่ผู้ใช้ตัดออกมาเพื่อเป็นส่วนหนึ่งของพื้นที่ที่จะทำการวิเคราะห์จะไม่สามารถถูกตัดแบ่งได้อีก และจะถูกเก็บไว้ใน segmentList ส่วนย่อยที่เหลือสามารถถูกตัดได้อีก

เมื่อผู้ใช้ตัดส่วนย่อยได้ 2 ชิ้นที่เป็นด้านตรงข้ามของพื้นที่สี่เหลี่ยมที่จะทำการวิเคราะห์แล้ว ก็จะทำการเชื่อมส่วนย่อยเหล่านั้นเข้าด้วยกันโดยใช้เครื่องมือชื่อว่า spline tool เพื่อให้ได้รูปปิดที่มีด้านสี่ด้าน รูปปิดนี้จะใช้ในการแบ่งพื้นที่ต่อไป



รูปที่ 4.9 การตัดแนวขอบเพื่อใช้ในการแบ่งพื้นที่

การเลือกเก็บข้อมูลจากแนวขอบย่อยเพื่อสร้างเส้นโค้งบี-สไปลน์

ส่วนย่อยที่ผู้ใช้ตัดออกมามีจำนวนข้อมูล(จุด)เป็นจำนวนมาก ประมาณ 200 - 1000 จุด การกำหนดตำแหน่งในส่วนย่อยด้วยพารามิเตอร์ทำได้ยาก โดยทั่วไปการกำหนดพารามิเตอร์ t เพื่อใช้บอกตำแหน่งในเส้นโค้งคือ ให้ t=0 เป็นจุดเริ่มต้น, t=0.5 เป็นตำแหน่งกึ่งกลางของเส้นโค้ง และ t=1 เป็นจุดปลาย หากกำหนดพารามิเตอร์ให้กับส่วนย่อยโดยใช้จำนวนจุดเป็นตัวกำหนดพารามิเตอร์ เช่นส่วนย่อยมีจำนวนข้อมูล 1000 จุด ถ้ากำหนดให้ t=0.25, 0.5, 0.75 ซึ่งตรงกับจุดที่ 250, 500, 750 เส้นโค้งตามตำแหน่งพารามิเตอร์เหล่านี้อาจจะไม่ได้มีความยาวเป็น 1 ใน 4, ครึ่งหนึ่ง และสามในสี่ของเมื่อดูด้วยตา หรืออาจพูดได้ว่าความยาวของเส้นโค้งไม่มีความสัมพันธ์แบบเชิงเส้นกับค่าพารามิเตอร์นั่นเอง ทั้งนี้เนื่องจากข้อมูลที่ได้จากการ scan มีลักษณะของขอบหยักไปมา

เมื่อผู้วิจัยนำจุดทั้งหมดของ ส่วนย่อยมาเป็นจุดควบคุม (control point) ของเส้นโค้งบี-สไปลน์ เส้นโค้งบี-สไปลน์ที่ได้จะมีลักษณะคล้ายกับส่วนย่อยคือมีลักษณะหยักไปมา เมื่อกำหนดพารามิเตอร์ (parameterization) จะมีความยาวส่วนโค้งไม่เป็นเชิงเส้นกับพารามิเตอร์เช่นเดียวกัน งานทางวิศวกรรมมักต้องการเส้นโค้งที่มีลักษณะเรียบ เส้นโค้งดังกล่าวจึงไม่เหมาะที่จะนำมาใช้

ผู้วิจัยคิดว่าเส้นโค้งที่นำมาใช้ควรมีความยาวคอร์ดมีความสัมพันธ์เชิงเส้นกับค่าพารามิเตอร์ จึงใช้วิธีประมาณค่าในการหาจุดซึ่งแบ่งส่วนย่อย(ที่ได้จากการตัด)ออกเป็นส่วน ๆ ที่มีความยาวคอร์ดเท่ากัน และใช้จุดแบ่งเหล่านี้เป็นจุดควบคุมในการสร้างเส้นโค้งบี-สไปลน์เพื่อเป็นตัวแทนของส่วนย่อย โดยแนวความคิดดังนี้

ตัวอย่างการแบ่งส่วนย่อยที่มีข้อมูล 1,000 จุดออกเป็น 10 ส่วน

ขั้นแรกแบ่งส่วนย่อยโดยใช้พารามิเตอร์เป็นอัตราส่วนกับจำนวนจุดก่อน พารามิเตอร์ 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 ก็จะตรงกับจุดที่ 0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 ตามลำดับ

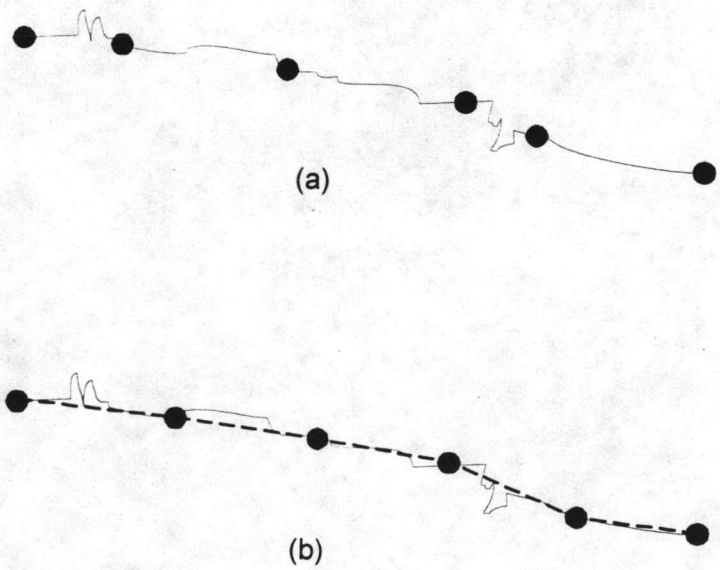
หาระยะทางระหว่างจุดเหล่านี้ (chord length) เมื่อรวมเข้าด้วยกันจะได้เป็นความยาวประมาณของเส้นโค้ง (approximate curve length) เมื่อหาความยาวประมาณเส้นโค้งด้วยจำนวนการแบ่ง (number of segment) จะได้ความยาวประมาณของคอร์ด

$$appx.chord\ length = \frac{appx.curve\ length}{number\ of\ segment}$$

หาจุดแบ่งเส้นโค้งใหม่โดยเคลื่อนที่ไปตามเส้นโค้งทีละจุด วัดระยะทางที่เคลื่อนที่ไปได้ ระยะนี้เรียกว่าความยาวคอร์ดสะสม (accumulate chord length) เมื่อความยาวคอร์ดสะสมมากกว่าหรือเท่ากับความยาวคอร์ดสะสมที่ควรจะเป็นในแต่ละช่วง (n^{th} segment x appx. chord length) จุดนี้หรือจุดที่อยู่ก่อนหน้าจะใช้เป็นจุดแบ่งเส้นโค้ง การเลือกว่าจะเป็นจุดใดนั้นขึ้นกับว่าจุดใดมีความยาวของคอร์ดสะสมใกล้เคียงกับความยาวคอร์ดที่ควรจะเป็นในแต่ละช่วงมากกว่า

Bresenham (อ้างจาก Foley and Van Dam, 1982) ได้เสนออัลกอริทึมในการวาดเส้นตรงในอุปกรณ์แบบ raster จุดต่าง ๆ ของเส้นคำนวนจากการเลือกว่าจุดใดมีระยะใกล้เคียงกับเส้นที่คำนวนได้ในทางทฤษฎีมากกว่า อัลกอริทึมดังกล่าวใช้เพียงการบวกลบตัวเลขและไม่ใช้การคูณ ซึ่งทำให้การทำงานรวดเร็ว แนวความคิดในการเลือกจุดที่ใกล้เคียงกับจุดแบ่งทางทฤษฎีของผู้วิจัยก็มาจากแนวความคิดนี้

เมื่อได้จุดแบ่งใหม่แล้ว ส่วนต่าง ๆ ที่ได้จากการแบ่งด้วยวิธีดังกล่าวจะมีความยาวคอร์ดใกล้เคียงกัน ส่วนที่มีความยาวคอรด์ต่างจากส่วนอื่น ๆ มากที่สุดคือส่วนย่อยสุดท้าย ผู้วิจัยจึงให้อัลกอริทึมทำงานเพิ่มขึ้นอีกสองรอบ ผลจากการแบ่งด้วยอัลกอริทึมดังกล่าวเมื่อมองด้วยตาแล้วเป็นที่น่าพอใจ

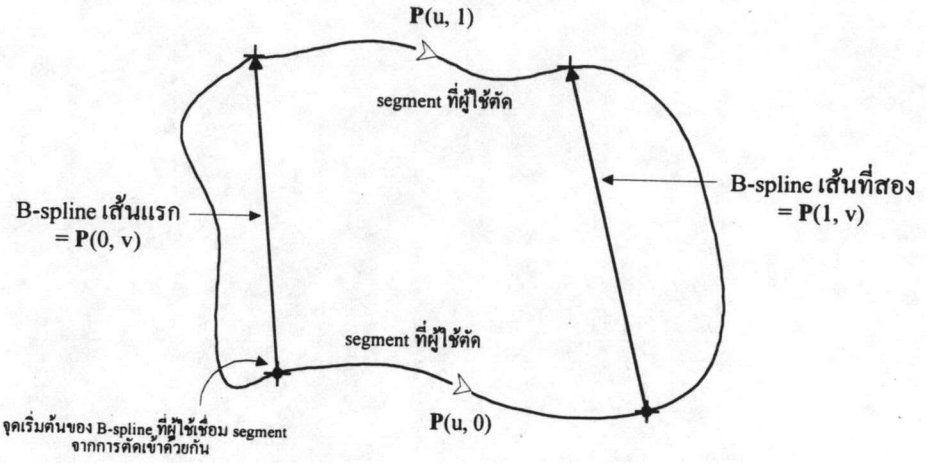


รูปที่ 4.10 (a) การแบ่งเส้นขอบตามจำนวนจุด

(b) การแบ่งโดยการเลือกจุดที่มีระยะห่างเท่า ๆ กัน

การประกอบเส้นขอบเป็นพื้นผิวคูนส์

จุดแบ่งที่ได้จากการเลือกเก็บข้อมูลส่วนย่อยจะนำมาใช้เป็นจุดควบคุมของเส้นโค้งบี-สไปลน์ เส้นโค้งบี-สไปลน์นี้จะนำมาประกอบเป็นด้านใดด้านหนึ่งในสี่ด้านของพื้นที่การวิเคราะห์ ผู้วิจัยได้กำหนดให้ส่วนย่อยที่ได้จากการตัดแบ่งเป็นด้านที่ใช้พารามิเตอร์ u ในพื้นผิวคูนส์ โปรแกรม PPROCESS กำหนดให้ผู้ใช้ตัดข้อมูลเพื่อให้เกิดเส้นโค้งที่ใช้พารามิเตอร์ u จำนวน 2 เส้น และด้าน 2 ด้านนี้กำหนดให้เป็นด้านตรงข้ามกันของพื้นที่สี่เหลี่ยม ด้าน 2 ด้านนี้จะถูกปิดด้วยเส้นโค้งที่ใช้พารามิเตอร์ v อีก 2 เส้นเพื่อให้ครบ 4 ด้านตาม topology ของพื้นผิวคูนส์ เส้นโค้งพารามิตอร์ v ถูกกำหนดให้ผู้ใช้สร้างขึ้นจาก spline tool spline tool ได้กำหนดให้ผู้ใช้สร้างเส้นโค้งสไปลน์ โดยจุดเริ่มต้นและจุดสุดท้ายจะต้องเป็นจุดปลายของเส้นโค้งพารามิเตอร์ u ในงานวิจัยนี้เส้นโค้งสไปลน์ ที่สร้างไว้จะเป็นเพียงเส้นตรงซึ่งก็คือเส้นโค้งบี-สไปลน์ที่มีจุดควบคุมเพียง 2 จุดและมีอันดับเป็น 2 เส้นโค้งสไปลน์เส้นแรกที่ผู้ใช้สร้างจะเป็นตัวกำหนดว่าเส้นโค้งแต่ละเส้นนั้นจะเป็นด้านประกอบใดในพื้นผิวคูนส์ จุดเริ่มต้นของเส้นโค้งสไปลน์พารามิเตอร์ v จะทำให้เส้นโค้งพารามิเตอร์ u ที่ติดกันเป็นเส้นโค้ง P(u, 0) และจุดปลายของเส้นโค้งสไปลน์ พารามิเตอร์ v ทำให้เส้นโค้งพารามิเตอร์ u ที่ติดกันเป็นเส้นโค้ง P(u, 1) เส้นโค้งสไปลน์ ที่สร้างขึ้นนี้กำหนดให้เป็นเส้นโค้ง P(0, v) จุดปลายเส้นโค้งพารามิเตอร์ u P(0, u) และ P(1, u) ที่มีเส้นโค้ง P(0, v) ติดต่อกันนั้นมีค่าพารามิเตอร์ u เท่ากับ 0 หรือเป็นจุดเริ่มต้นของเส้นโค้งพารามิเตอร์ u นั่นเอง เมื่อผู้ใช้สร้างเส้นโค้งสไปลน์ เพิ่มขึ้นอีก 1 เส้น เส้นโค้งสไปลน์ เส้นนั้นจะเป็นเส้นโค้ง P(1, v) จุดปลายที่เส้นโค้ง P(0, u) และ P(1, u) ติดกับเส้นโค้งสไปลน์นี้มีค่าพารามิเตอร์ u เท่ากับ 1 หรือก็คือจุดปลายของเส้นโค้งพารามิเตอร์ u ทั้งสองนั่นเอง เมื่อสร้างเส้นโค้งครบ 4 เส้นดังนี้แล้วพื้นผิวคูนส์ก็จะมีด้านต่าง ๆ ครบสมบูรณ์ตาม topology และพร้อมสำหรับการสร้างพื้นผิวและการแบ่งพื้นที่ต่อไป



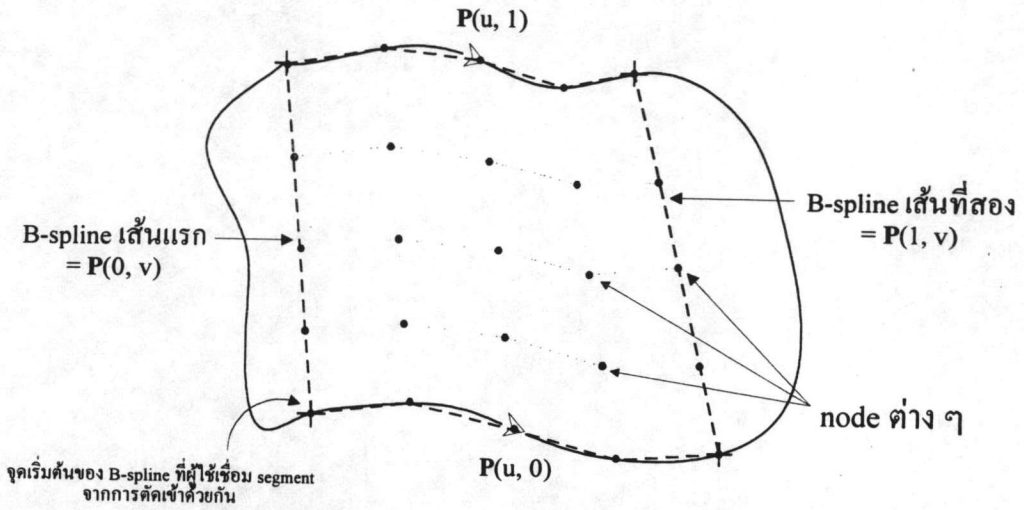
รูปที่ 4.11 การประกอบส่วนย่อยต่าง ๆ และเส้นโค้งบี-สไปลน์เป็นพื้นผิวคูนส์

<u>การใช้พื้นผิวคูนส์แบ่งพื้นที่</u>

พื้นผิวคูนส์เชิงเส้นคู่ (bilinear coon surface) ซึ่งมีสูตรเป็น

$$\mathbf{P}(u,v) = -\begin{bmatrix} -1 & (1-u) & u \end{bmatrix} \begin{bmatrix} 0 & \mathbf{P}(u,0) & \mathbf{P}(u,1) \\ \mathbf{P}(0,v) & \mathbf{P}(0,0) & \mathbf{P}(0,1) \\ \mathbf{P}(1,v) & \mathbf{P}(1,0) & \mathbf{P}(1,1) \end{bmatrix} \begin{bmatrix} -1 \\ 1-v \\ v \end{bmatrix}$$

เมื่อแทนค่า u และ v สามารถหาค่าพิกัดของจุดต่าง ๆ บนพื้นผิวได้ ในงานวิจัยนี้เส้นโค้งขอบเขตที่นำมาประกอบเป็นพื้นผิวคูนส์มีค่าอันดับ k เป็น 2 จึงมีลักษณะเป็นส่วนของเส้นตรงต่อเนื่องกันเนื่องจากเส้นโค้งบี-สไปลน์มีเป็นโพลิโนเมียลเชิงเส้นต่อกันเป็นชิ้น ๆ (piecewise linear polynomial) ถ้าแทนค่า u และ v ที่เป็นจุดแบ่งของเส้นโค้งพารามิเตอร์ u ลงในสมการ ก็จะได้จุดซึ่งเป็นตำแหน่งจุดต่อ (node) ของเมชในวิธีการไฟไนต์เอลิเมนต์ ถ้าทดลองแทนค่าพารามิเตอร์ (u และ v) ที่มีค่าระหว่างเมชใดเมชหนึ่งหลาย ๆ จุดเพื่อให้เห็นแนวโน้มขอบเขตของเมช จะเห็นว่าจุดที่อยู่ระหว่างจุดต่อ 2 จุดอยู่ในแนวเส้นตรงที่เชื่อมระหว่าง 2 จุดต่อนั้น แสดงให้เห็นว่าพื้นผิวคูนส์สามารถประมาณค่าภายใน (interpolate) เส้นขอบเขตทั้ง 4 ให้เป็นจุดบนพื้นผิวได้ เมื่อแทนค่าพารามิเตอร์ในตำแหน่งจุดต่อที่ต้องการจนหมดก็จะได้จุดต่อของวิธีการไฟไนต์เอลิเมนต์



รูปที่ 4.12 การแบ่งพื้นที่ด้วยพื้นผิวคูนส์

<u>การพัฒนาโปรแกรม PPROCESS</u>

โปรแกรม PPROCESS มีความสามารถในการนำเข้าข้อมูลพิกัดของวัตถุจากภาพเพื่อสร้างเอลิมนต์และจุดต่อ สามารถป้อนค่าสภาวะขอบเขตทางความร้อนและสมบัติทางกายภาพของวัตถุ คือ สัมประสิทธิ์การนำความร้อน (conductivity coefficient) ณ ตำแหน่งจุดต่อ (node) ต่าง ๆ เมื่อผู้ใช้ป้อนข้อมูลครบถ้วนแล้วโปรแกรมสร้างแฟ้มข้อมูลของแบบจำลองความร้อนในรูปของเท็กซ์เพื่อใช้ในโปรแกรมคำนวนด้วยวิธีการไฟไนต์เอลิเมนต์

โปรแกรม PPROCESS พัฒนาขึ้นด้วยภาษา C++ ใช้เทคนิคการโปรแกรมเชิงวัตถุ (object-oriented programming) ในการออกแบบและเขียนโปรแกรม เขียนด้วยคอมไพลเลอร์ Borland C++ 4.5 และทำงานบนไมโครซอฟท์วินโดวส์ มีส่วนติดต่อผู้ใช้แบบกราฟฟิก

แฟ้มข้อมูลต่าง ๆ ที่สร้างจากโปรแกรม PPROCESS มีรายละเอียดดังนี้

1. nodegeom.dat	พิกัด x, y ของจุดต่อต่าง ๆ
2. elmtnode.dat	จุดต่อซึ่งประกอบเป็นเอลิเมนต์ต่าง ๆ
3. conduct.dat	ค่าสัมประสิทธิ์การนำความร้อน ณ จุดต่อต่าง ๆ
4. convec.dat	ค่าสัมประสิทธิ์การพาความร้อนและอุณหภูมิของตัวกลาง ณ จุดต่อต่าง ๆ ที่มีการพาความร้อน
5. convnode.dat	คู่ของจุดต่อที่ประกอบกันเป็นด้านที่มีการพาความร้อน
6. tempfix.dat	ค่าของอุณหภูมิคงที่ ณ จุดต่อต่าง ๆ
7. nodemap.dat	การจัดเรียงของจุดต่อในแนวทาง u และ v