การพัฒนาโปรแกรมคอมพิวเตอร์สำหรับงานวิเคราะห์สายอากาศแบบจานสะท้อนคลื่น ระยะที่ 1

โดย

ฉัตรชัย ไวยาพัฒนกร

โครงการวิจัยเลขที่ 136-MRD-2538

ทุนส่งเสริมการวิจัยวิศวกรรมศาสตร์

ปี 2539

สถาบันวิจัยและพัฒนาของคณะวิศวกรรมศาสตร์

คณะวิศวกรรมศาสตร์

จุฬาลงกรณ์มหาวิทยาลัย

กรุงเทพฯ

มกราคม 2540

การพัฒนาโปรแกรมคอมพิวเตอร์สำหรับงานวิเคราะห์สายอากาศแบบจานสะท้อนคลื่น ระยะที่ 1

โดย
ฉัตรชัย ไวยาพัฒนกร
วุฒิ ปรัชญาดุษฎีบัณฑิต

โครงการวิจัยเลขที่ 136-MRD-2538
ทุนส่งเสริมการวิจัยวิศวกรรมศาสตร์
ปี 2539

สถาบันวิจัยและพัฒนาของคณะวิศวกรรมศาสตร์
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย
กรุงเทพฯ
มกราคม 2540

บทคัดย่อ

การตรวจสอบสมรรถนะของสายอากาศจานสะท้อนคลื่นที่ได้รับการออกแบบสามารถกระทำได้ทั้งโดยการสร้างแล้วดำเนินการวัดในย่านทดสอบ หรือการวิเคราะห์ด้วยวิธีการทางคณิตศาสตร์ การวิเคราะห์ทางคณิตศาสตร์มีข้อดีในเรื่องของค่าใช้จ่ายและเวลาในการดำเนินการ การพัฒนาโปรแกรมคอมพิวเตอร์เป็นวิธีการเพิ่มประสิทธิภาพและประสิทธิผลในการวิเคราะห์ที่ดียิ่ง โครงการนี้ได้พัฒนาโปรแกรมคอมพิวเตอร์สำหรับการวิเคราะห์สายอากาศชนิดจานสะท้อนเดี่ยวสมมาตร โปรแกรมที่ได้รับการพัฒนามีความสามารถในการวิเคราะห์เพื่อการคำนวณแบบรูปการแผ่พลังงาน และอัตราขยายของสายอากาศชนิดจานสะท้อนเดี่ยวสมมาตรรูปพาราโบลิก โดยมีการคำนวณผลกระทบเนื่องจากการเลี้ยวเบนที่ขอบของจานสะท้อนด้วยทฤษฎีการเลี้ยวเบนเชิงเรขาคณิต ผลการคำนวณแบบรูปการแผ่พลังงานและอัตราขยายของสายอากาศจานสะท้อนคลื่นด้วยโปรแกรมที่พัฒนาขึ้นได้รับการตรวจสอบความแม่นยำโดยการเปรียบเทียบกับโปรแกรมมาตรฐานและผลงานวิจัยที่ได้รับการตีพิมพ์ในวารสารวิชาการที่มีมาตรฐานระดับสากล และพบว่ามีความแม่นยำเป็นที่น่าพอใจ

Abstract

Evaluation of a reflector antenna performance after its design can be achieved by constructing the antenna and performing measurements in a test range, or mathematically analyse the antenna. Mathematical analysis has advantage in terms of both expenditure and time of operation. The development of a computer program is an efficient and effective way of performing the analysis. This project has developed a computer program for the analysis of axially symmetric parabolic reflector antennas. The program has been developed to be analytically capable of calculating antenna's far field radiation pattern and gain. Calculation of effects due to diffraction at the edge of the reflector using geometrical theory of diffraction has also been included. Accuracies of results of calculation using the program developed have been assessed by comparing with results from standard software and published work of high quality in international publications. It is found to be satisfactory.

กิตติกรรมประกาศ

ผู้วิจัยขอขอบคุณ คุณ ศุภเชษฐ์ เพิ่มพูนวัฒนาสุข และ คุณ ธนากร ลิขิตาภิวัฒน์ ในฐานะที่ทั้งสองท่านเป็นกำลังสำคัญในการทำงานจนกระทั่งโครงการสำเร็จลุล่วงด้วยดี นอกจากนี้ขอขอบคุณคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ให้ความสนับสนุนทางการเงินกับโครงการนี้

สารบัญ

1 บทนำ

ความเป็นมาของปัญหา

สายอากาศนับเป็นอุปกรณ์ตัวหนึ่งของระบบสื่อสารที่มีความสำคัญมาก สมรรถนะของสายอากาศจะเป็นตัวบ่งสมรรถนะของระบบโดยรวม โดยเฉพาะอย่างยิ่งในระบบสื่อสารที่ช่องสัญญาณมีความจุสูงและมีการส่งผ่านข้อมูลด้วยอัตราเร็วสูง เช่น ระบบเชื่อมโยงด้วยสัญญาณจุลวัฏภาคพื้นดิน(microwave terrestrial link) ระบบสื่อสารผ่านดาวเทียม ฯลฯ ในระบบเหล่านี้การเปลี่ยนแปลงเชิงสมรรถนะของสายอากาศก่อให้เกิดผลกระทบที่ค่อนข้างรุนแรงได้

ที่จริงแล้ววิทยาการทางสายอากาศได้รับการพัฒนาอย่างต่อเนื่องมาช้านานแล้วในต่างประเทศโดยเฉพาะประเทศที่พัฒนาแล้ว แต่สำหรับประเทศไทยนั้นการพัฒนาในด้านนี้ยังล้าหลังอยู่มากทีเดียว ทำให้ไม่อาจมีการพัฒนาของอุตสาหกรรมสายอากาศได้อย่างจริงจัง การออกแบบและสร้างสายอากาศมักใช้วิธีลองผิดลองถูกเหมือนอย่างสังคมดั้งเดิมที่เพิ่งเริ่มสร้างวิทยาการต่างๆ หรือลอกเลียนผลิตภัณฑ์ต่างประเทศที่มีขายในท้องตลาด ด้วยเหตุนี้การพัฒนาโปรแกรมคอมพิวเตอร์สำหรับงานวิเคราะห์สายอากาศแบบจานสะท้อนคลื่นจึงมีความจำเป็นอย่างยิ่งเพื่อช่วยยกระดับงานวิศวกรรมภายในประเทศ และยังจะเป็นผลสืบเนื่องให้ประเทศไทยมีความสามารถในการแข่งขันในตลาดโลกที่ดีขึ้นในระยะยาว

โครงการนี้ได้พัฒนาโปรแกรมคอมพิวเตอร์สำหรับการวิเคราะห์สายอากาศแบบจานสะท้อนคลื่น โดยในช่วงเวลาของโครงการนี้โปรแกรมที่สร้างขึ้นมีความสามารถเบื้องต้นในการวิเคราะห์เพื่อการคำนวณแบบรูปการแผ่พลังงานและอัตราขยายของสายอากาศแบบจานสะท้อนคลื่นที่มีโครงสร้างง่ายที่สุด นั่นคือแบบจานสะท้อนคลื่นเดี่ยวที่มีการป้อนสัญญาณที่หน้าจานอย่างสมมาตร ในการนี้ไม่มีการศึกษาผลกระทบที่เนื่องมาจากโครงสร้างของระบบป้อนสัญญาณ การผิดรูปของผิวจาน และการเลื่อนเชิงตำแหน่งของสายอากาศป้อน มีการศึกษาการใช้วิธีการคำนวณที่ทำให้สามารถรวมผลกระทบจากการเลี้ยวเบนที่ขอบ คือ ทฤษฎีการเลี้ยวเบนเชิงเรขาคณิต

ประโยชน์ที่ได้รับ

ผลลัพธ์ของโครงการนี้ คือ โปรแกรมคอมพิวเตอร์สำหรับการวิเคราะห์สายอากาศแบบจานสะท้อนรูปพาราโบลิกชนิดจานสะท้อนคลื่นเดี่ยวที่มีการป้อนสัญญาณที่หน้าจานอย่างสมมาตร โปรแกรมสามารถทำการคำนวณแบบรูปการแผ่พลังงานและอัตราขยายของสายอากาศโดยรวมผลกระทบจากการเลี้ยวเบนที่ขอบ โปรแกรมนี้ช่วยเพิ่มพูนประสิทธิภาพการวิเคราะห์และออกแบบสายอากาศจานสะท้อนคลื่นชนิดจานสะท้อนเดี่ยวที่มีการป้อนสัญญาณที่หน้าจานอย่างสมมาตร นอกจากนี้ยังเป็นประโยชน์ในการศึกษาเบื้องต้นเพื่อการออกแบบและสร้างย่านทดสอบสายอากาศแบบกระชับ

โครงร่างของรายงานฉบับนี้

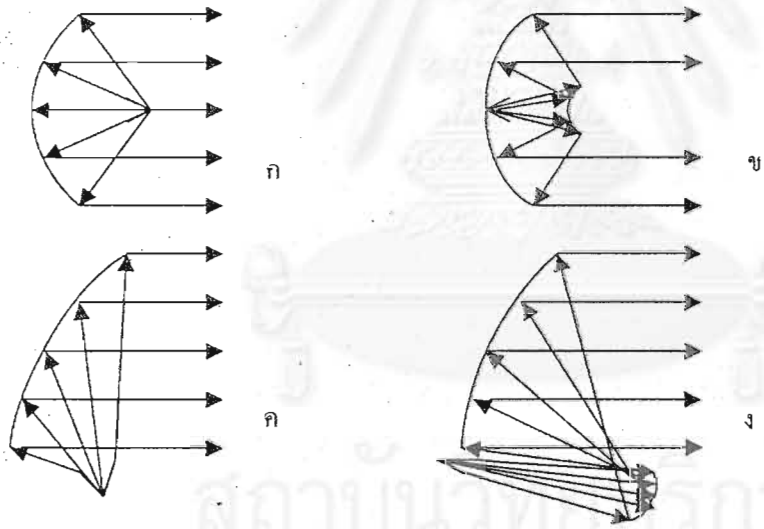
รายงานฉบับนี้กล่าวถึงการวิเคราะห์สายอากาศจานสะท้อนคลื่นในบทที่ 2 การคำนวณและแบบจำลองคณิตศาสตร์สำหรับการคำนวณในบทที่ 3 ในบทที่ 4 และ 5 เป็นรายงานผลการตรวจสอบความสามารถของโปรแกรมที่พัฒนาขึ้น และตัวอย่างการใช้งานโปรแกรม สำหรับบทสรุปและข้อเสนอแนะของงานวิจัยนี้มีกล่าวไว้ในบทที่ 6 ส่วนคู่มือการใช้งานโปรแกรมและรายการของโปรแกรมที่พัฒนาขึ้นมีแนบอยู่ในภาคผนวกของรายงานนี้

2 การวิเคราะห์สายอากาศจานสะท้อนคลื่น

แนวคิดหลักในการวิเคราะห์ที่ใช้ในโครงการนี้ คือ แนวคิดเชิงรังสีของทัศนศาสตร์เรขาคณิต โดยการคำนวณผลกระทบจากการเลี้ยวเบนที่ขอบจานสะท้อนใช้ทฤษฎีการเลี้ยวเบนเชิงเรขาคณิต ซึ่งเป็นส่วนขยายของทัศนศาสตร์เรขาคณิต ในลำดับแรกจะได้กล่าวถึงประยุกต์วิทยาสายอากาศชนิดจานสะท้อนคลื่นก่อน แล้วจึงนำเสนอเกี่ยวกับการวิเคราะห์ต่อไป

ประยุกต์วิทยาสายอากาศชนิดจานสะท้อนคลื่น

สายอากาศชนิดจานสะท้อนคลื่นจัดเป็นสายอากาศประเภทช่องเปิด สายอากาศชนิดนี้มีความเหมาะสมอย่างยิ่งสำหรับการใช้งานในย่านความถี่จุลวัฏ(microwave frequencies) มีอัตราขยายและความสามารถเชิงทิศทางที่ดีมาก สายอากาศชนิดนี้โดยทั่วไปแล้วมีโครงสร้างซึ่งประกอบด้วยส่วนสำคัญหลักๆ 2 ส่วน คือ จานสะท้อนคลื่นซึ่งอาจมีมากกว่า 1 จานก็ได้ และสายอากาศป้อนหรือตัวป้อนซึ่งทำหน้าที่ป้อนกำลังคลื่นเข้าสู่จานสะท้อน รูปที่ 1 แสดงลักษณะโครงสร้างเชิงเรขาคณิตและหลักการทำงานของสายอากาศชนิดจานสะท้อนคลื่นแบบต่างๆ



รูปที่ 1 ลักษณะโครงสร้างเชิงเรขาคณิตและหลักการทำงานของสายอากาศชนิดจานสะท้อนคลื่นแบบต่างๆ
ก) ชนิดจานสะท้อนเดี่ยวสมมาตร ข)ชนิดจานสะท้อนคู่สมมาตร ค)ชนิดจานสะท้อนเดี่ยวไม่สมมาตร
ง) ชนิดจานสะท้อนคู่ไม่สมมาตร

แบบ ก เป็นชนิดจานสะท้อนเดี่ยวสมมาตรซึ่งเป็นแบบที่โครงการนี้สนใจ รังสีของคลื่นเมื่อออกจากตัวป้อน ณ จุดรวมศูนย์หรือจุดโฟกัส(focal point) จะวิ่งเข้าสู่จานสะท้อนในทิศทางต่างๆ แต่เมื่อกระทบผิวจานสะท้อนออกมารังสีทั้งหมดจะถูกบังคับให้วิ่งขนานกันออกไป แบบ ข เป็นชนิดจานสะท้อนคู่สมมาตร การจัดโครงสร้างเชิงเรขาคณิตดังแบบ ข มีข้อดีที่สามารถลดระยะรวมศูนย์หรือระยะโฟกัสในกรณีที่ต้องการอัตราขยายสูงมากๆ และยังมีส่วนช่วยเพิ่มประสิทธิภาพช่องเปิด(aperture efficiency)ได้อีกด้วย อย่างไรก็ดีแบบนี้มีข้อเสียที่เนื่องมาจากการบดบังสัญญาณของจานสะท้อนรอง

แบบ ค เป็นชนิดจานสะท้อนเดี่ยวไม่สมมาตร แบบนี้มีข้อดีที่ไม่ปรากฏผลเสียจากการบดบังสัญญาณของตัวป้อนและโครงสร้างที่รองรับตัวป้อนดังเช่นที่เกิดขึ้นกับแบบ ก แต่ก็ได้รับผลกระทบจากการเพิ่มขึ้นของระดับโพลาไรเซชันไขว้เนื่องจากการจัดวางตัวป้อนในลักษณะที่ไม่สมมาตร ปัญหานี้แก้ไขได้โดยการเพิ่มจานสะท้อนรองดังแสดงในแบบ ง สายอากาศแบบ ค และ ง มีจุดด้อยเมื่อเทียบกับแบบ ก และ ข ตรงที่การจัดวางจานสะท้อนและตัวป้อนกระทำได้ยากกว่า และอ่อนไหวต่อการกระทบเชิงกลมากกว่า การออกแบบและสร้างจึงต้องพิถีพิถันมาก

สำหรับกรณีของสายอากาศชนิดจานสะท้อนเดี่ยวสมมาตรดังแสดงในรูปที่ 1ก นั้น มีปัจจัยต่างๆที่ส่งผลกระทบต่อสมรรถนะในการทำงานดังนี้

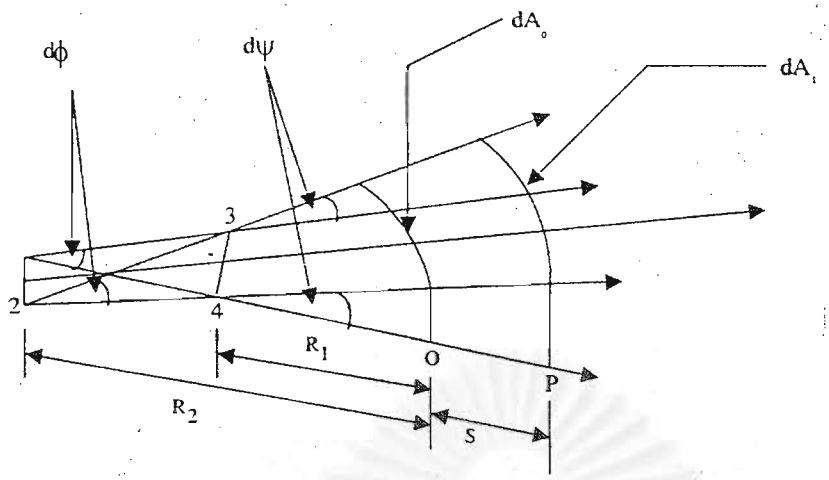
- การเลี้ยวเบนที่ขอบจานสะท้อน
- การเลื่อนตำแหน่งของตัวป้อนจากจุดรวมศูนย์
- การบดบังสัญญาณเนื่องจากตัวป้อนและโครงสร้างที่รองรับมัน
- การผิดรูปของผิวจานสะท้อน

โครงการนี้มีวัตถุประสงค์ในการพัฒนาโปรแกรมคอมพิวเตอร์ที่สามารถวิเคราะห์ผลกระทบที่เกิดเนื่องจากการเลี้ยวเบนที่ขอบจานสะท้อนของสายอากาศชนิดจานสะท้อนเดี่ยวสมมาตรเท่านั้น การเลี้ยวเบนที่ขอบของจานสะท้อนเป็นปรากฏการตามธรรมชาติที่เกิดขึ้น โปรแกรมคอมพิวเตอร์ที่พัฒนาขึ้นจะช่วยให้ศึกษาผลกระทบจากการการเลี้ยวเบนที่ขอบจานสะท้อนรวมทั้งหามาตรการลดผลกระทบได้อย่างมีประสิทธิภาพยิ่งขึ้น

<u>วิธีวิเคราะห์สายอากาศชนิดจานสะท้อนคลื่น</u>

การวิเคราะห์สายอากาศชนิดจานสะท้อนสามารถทำได้ด้วยวิธีการต่างๆ[3,10] โครงการนี้เลือกวิธีวิเคราะห์ปัญหาการเลี้ยวเบนที่ขอบจานสะท้อนโดยดำเนินตามวิธีทฤษฎีการเลี้ยวเบนเชิงเรขาคณิต[4,9] ทฤษฎีนี้เป็นส่วนขยายของทัศนศาสตร์เรขาคณิตที่ไม่สามารถรวมผลที่เกิดจากการเลี้ยวเบนไว้ในการคำนวณด้วย

ทัศนศาสตร์เรขาคณิตพิจารณาการเคลื่อนที่ไปของคลื่นในตัวกลางในลักษณะของรังสีของลำพลังงานคลื่น กำลังงานคลื่นที่ปรากฏในลำพลังงานเดียวกันจะคงที่เสมอตามหลักการอนุรักษ์พลังงาน นอกจากนี้แนวทางเดินของรังสี คือ แนวที่เส้นทางเชิงทัศนศาสตร์มีความคงตัวตามหลักการของแฟร์มาต์ที่ดัดแปลงแล้ว[11] ลำรังสีที่ใช้ในการคำนวณด้วยทัศนศาสตร์เรขาคณิตมีแสดงในรูปที่ 2



รูปที่ 2 ลำรังสีสำหรับการคำนวณด้วยทัศนศาสตร์เรขาคณิต

จากรูปที่ 2 สามารถเขียนสมการแสดงหลักการอนุรักษ์พลังงานได้ดังนี้

$$S_0 dA_0 = S_1 dA_1 \qquad (1)$$

โดยที่ S_0 และ S_1 คือ ความหนาแน่นกำลังงาน ณ จุด O และ P ตามลำดับ

โดยทั่วไปสามารถเขียน S ได้ในรูปของ E ดังนี้

$$S(r) = \frac{E(r)^2}{2Z} \qquad (2)$$

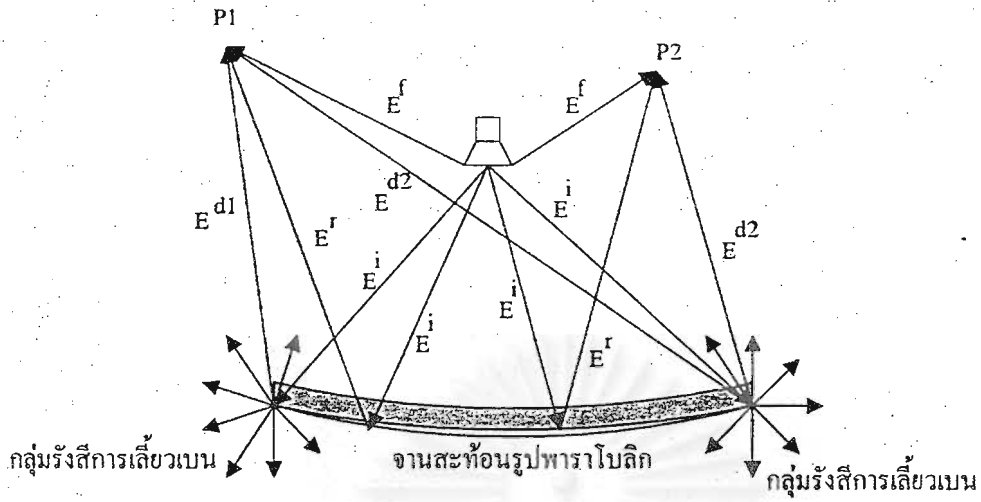
โดยที่ Z คืออิมพีแดนซ์ลักษณะสมบัติของตัวกลางที่คลื่นเคลื่อนที่ไป

จากสมการ (1) และ (2) จะได้ความสัมพันธ์ซึ่งเป็นไปตามหลักการอนุรักษ์พลังงานดังนี้

$$E_p = E_o \frac{R_1 R_2}{(R_1 + s)(R_2 + s)} \qquad (3)$$

จุดอ่อนของวิธีทัศนศาสตร์เรขาคณิตก็คือ การขาดความสามารถในการทำนายผลของการเลี้ยวเบน JB Keller[4] จึงได้เสนอทฤษฎีการเลี้ยวเบนเชิงเรขาคณิตเพื่อเสริมความสามารถดังกล่าว ทฤษฎีนี้พิจารณาการเลี้ยวเบนที่เกิดขึ้นโดยแทนด้วยรังสีการเลี้ยวเบน ซึ่งรังสีการเลี้ยวเบนหาได้จากผลคูณระหว่างสัมประสิทธิ์การเลี้ยวเบนกับค่าความเข้มสนามไฟฟ้าของรังสีตกกระทบ ณ จุดเลี้ยวเบน

รูปที่ 3 แสดงปัญหาการวิเคราะห์สายอากาศชนิดจานสะท้อนเดี่ยวรูปพาราโบลิกโดยรวมผลที่เกิดจากการเลี้ยวเบนที่ขอบจานด้วย



รูปที่ 3 รังสีสมมติสำหรับการวิเคราะห์สายอากาศชนิดจานสะท้อนเดี่ยวรูปพาราโบลิก

จากรูปที่ 3 ณ จุดสังเกต P_1 และ P_2 ค่าความเข้มสนามไฟฟ้าสามารถคำนวณได้ดังสมการต่อไปนี้

$$E_{P_1} = E^f + E^r + E^{d1} + E^{d2} \qquad (4)$$

$$E_{P_2} = E^f + E^r + E^{d2} \qquad (5)$$

โดยที่

E^f คือ ค่าความเข้มสนามไฟฟ้าจากสายอากาศป้อน

E^i, E^r คือ ค่าความเข้มสนามไฟฟ้าที่ตกกระทบลงบนและสะท้อนจากผิวจานสะท้อน

E^{d1}, E^{d2} คือ ค่าความเข้มสนามไฟฟ้าเนื่องจากการเลี้ยวเบน

ถ้าให้ E^d แทนสนามการเลี้ยวเบนใดๆ สามารถคำนวณค่าของมันได้ดังนี้

$$E^d = DE^i \qquad (6)$$

โดยที่

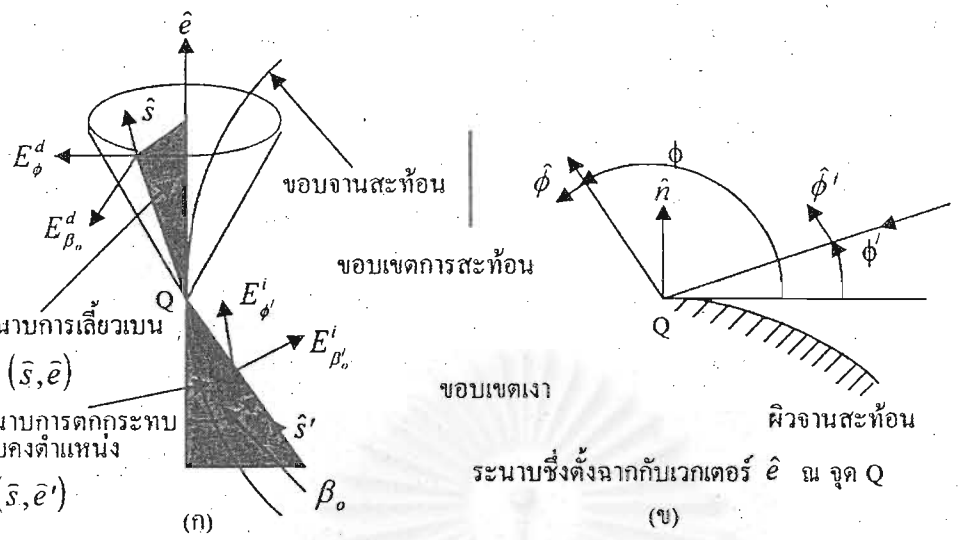
D คือ ค่าสัมประสิทธิ์การเลี้ยวเบน

โดยทั่วไปจะพิจารณาค่า E^d โดยแยกเป็น 2 องค์ประกอบที่ตั้งฉากกัน การคำนวณ E^d จาก E^i เป็นไปตามสูตรต่อไปนี้

$$E^d_{\beta_0}(P) = -E^i_{\beta_0'}(Q) D_s(\phi, \phi'; \beta_0') A(s) e^{-jks} \qquad (7)$$

$$E^d_{\phi}(P) = -E^i_{\phi'}(Q) D_h(\phi, \phi'; \beta_0') A(s) e^{-jks} \qquad (8)$$

ค่าแสดงทิศทางต่างๆ β_0, ϕ, β_0' และ ϕ' เป็นดังปรากฏในรูปที่ 4



รูปที่ 4 เรขาคณิตของการวิเคราะห์การเลี้ยวเบนที่ขอบจานสะท้อน

$$D_{h,s}(\phi,\phi';\beta_o') = -\frac{e^{-j\frac{\pi}{4}}}{2\sqrt{2\pi k}\sin\beta_o'}\left\{\begin{matrix}F\left[kL^i a(\phi-\phi')\right]\\ \cos\left[(\phi-\phi')/2\right]\end{matrix}\pm\begin{matrix}F\left[kL^r a(\phi+\phi')\right]\\ \cos\left[(\phi+\phi')/2\right]\end{matrix}\right\} \quad (9)$$

$$L^i = [ss'/(s+s')]\sin^2\beta_o' \quad (10)$$

$$L^r = [s\rho_1^r/(s+\rho_1^r)\sin^2\beta_o \quad (11)$$

$$a(\phi\pm\phi'') = 2\cos^2[(\phi\pm\phi'')/2] \quad (12)$$

และ

$$F[x] = 2j\sqrt{x}e^{jx}\int_{\sqrt{x}}^{\infty} e^{-j\tau^2}d\tau \quad (13)$$

ค่าของ ρ_1^r ซึ่งเป็นรัศมีความโค้งหลักของหน้าคลื่นสะท้อน ณ จุด Q บนระนาบการตกกระทบ สามารถคำนวณได้ดังสูตรต่อไปนี้

$$\frac{1}{\rho_1^r} = \frac{1}{\rho_1^i} + \frac{2}{\rho_{g1}\cos\theta^i} \quad (14)$$

โดยที่

ρ_1^i คือ รัศมีความโค้งหลักของหน้าคลื่นตกกระทบ ณ จุด Q บนระนาบการตกกระทบ โดยมีขนาดเท่ากับระยะ s' จากแหล่งกำเนิดถึงจุด Q

ρ_{g1} คือ รัศมีความโค้งของผิว ณ จุด Q ในบริเวณระนาบการตกกระทบ

^{i} คือ มุมตกกระทบ ณ จุด Q

สำหรับค่าตัวประกอบการลดทอน $A(s)$ ซึ่งแสดงผลของการบานออกของหน้าคลื่นเมื่อมีการแผ่กระจายออกจากแหล่งกำเนิด โดยในกรณีนี้ คือ จากตำแหน่งที่เกิดการเลี้ยวเบน สามารถคำนวณได้ดังสูตรต่อไปนี้

$$A(s) = \sqrt{\rho/[s(\rho+s)]} \quad (15)$$

ซึ่งระยะคอสติก(caustic) ρ จากตำแหน่งการเลี้ยวเบน ณ จุด Q บนขอบโค้งสามารถคำนวณได้ดังนี้

$$\frac{1}{\rho} = \frac{1}{\rho'_e} - \frac{n_e \cdot (s' - s)}{a_e \sin^2 \beta''_o} \qquad (16)$$

โดย

ρ'_e คือ รัศมีความโค้งของหน้าคลื่นตกกระทบ ณ จุด Q บนระนาบซึ่งประกอบด้วยเวกเตอร์ $\hat{e}$ และ $\hat{s}'$

a_e คือ รัศมีความโค้งของขอบ ณ จุด Q .

$\hat{n}$ คือ เวกเตอร์หนึ่งหน่วยในทิศตั้งฉากกับขอบ ณ จุด Q และชี้ออกจากจุดศูนย์กลางความโค้ง

พจน์อื่นๆ นอกเหนือจากที่กล่าวมามีนิยามดังต่อไปนี้

D_s คือ สัมประสิทธิ์การเลี้ยวเบนอย่างอ่อน

D_h คือ สัมประสิทธิ์การเลี้ยวเบนอย่างแข็ง

s' คือ ระยะจากแหล่งกำเนิดถึงจุด Q

s คือ ระยะจากจุด Q ถึงจุดสังเกต P

$\hat{e}$ คือ เวกเตอร์หนึ่งหน่วยในแนวสัมผัสกับขอบ ณ จุด Q

$\hat{s}'$ คือ เวกเตอร์หนึ่งหน่วยในแนวของรังสีตกกระทบ ณ จุด Q

$\hat{s}$ คือ เวกเตอร์หนึ่งหน่วยในแนวของรังสีการเลี้ยวเบน ณ จุด Q

$\hat{\phi}, \hat{\phi}'$ คือ เวกเตอร์หนึ่งหน่วยบนระนาบที่ตั้งฉากกับขอบ ณ จุด Q ดังแสดงในรูปที่ 4

$\hat{\beta}'_o$ คือ เวกเตอร์หนึ่งหน่วยระนาบซึ่งประกอบด้วย $\hat{e}$ และ $\hat{s}'$

$\hat{\beta}_o$ คือ เวกเตอร์หนึ่งหน่วยระนาบซึ่งประกอบด้วย $\hat{e}$ และ $\hat{s}$

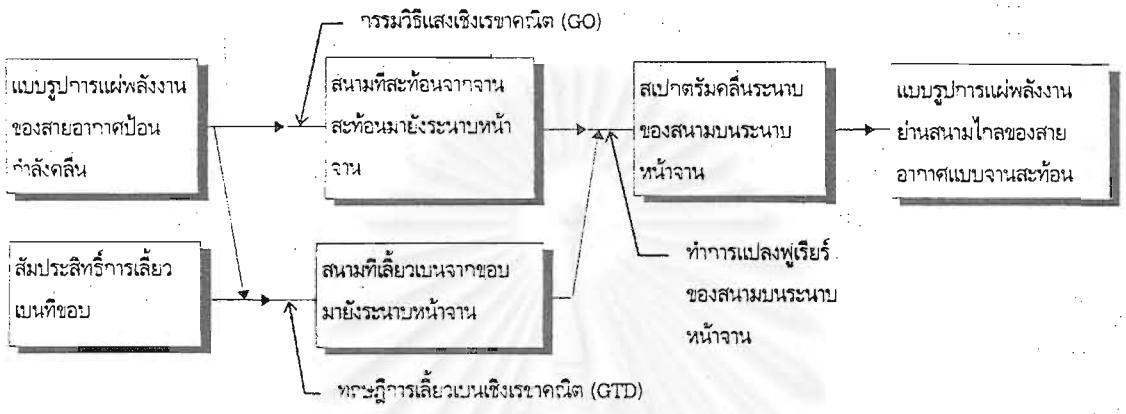
β_o คือ มุมระหว่างแนวตกกระทบและแนวที่สัมผัสกับขอบ

สรุป

สายอากาศจานสะท้อนเป็นสายอากาศที่มีอัตราขยายสูงเหมาะสำหรับการใช้งานย่านความถี่จุลวัฏ สำหรับโครงการวิจัยนี้เลือกพัฒนาโปรแกรมสำหรับวิเคราะห์สายอากาศชนิดจานสะท้อนเดี่ยวสมมาตร โดยพิจารณาผลกระทบเนื่องจากการเลี้ยวเบนที่ขอบเพียงอย่างเดียว กรรมวิธีวิเคราะห์ที่ใช้ คือ ทัศนศาสตร์เรขาคณิตและทฤษฎีการเลี้ยวเบนเชิงเรขาคณิต

3 การคำนวณและแบบจำลองคณิตศาสตร์สำหรับการคำนวณ

การคำนวณดำเนินไปตามกรรมวิธีวิเคราะห์ที่ได้กล่าวไว้ในบทที่ 2 โดยตัวโปรแกรมที่พัฒนาขึ้นมีผังขั้นตอนการคำนวณดังแสดงในรูปที่ 1



รูปที่ 1 ขั้นตอนการคำนวณหาแบบรูปการแผ่พลังงานย่านสนามไกลของสายอากาศชนิดจานสะท้อนคลื่น

อนึ่งในการคำนวณนี้ทำการแปลงฟูริเยร์โดยวิธีการอินทิเกรตเชิงตัวเลขตามแบบมาตรฐานทั่วไปมิได้ใช้การแปลงฟูริเยร์อย่างเร็ว จึงอาจมีผลต่อความเร็วของการคำนวณบ้าง

แบบจำลองคณิตศาสตร์ต่างๆที่ใช้ในการคำนวณ

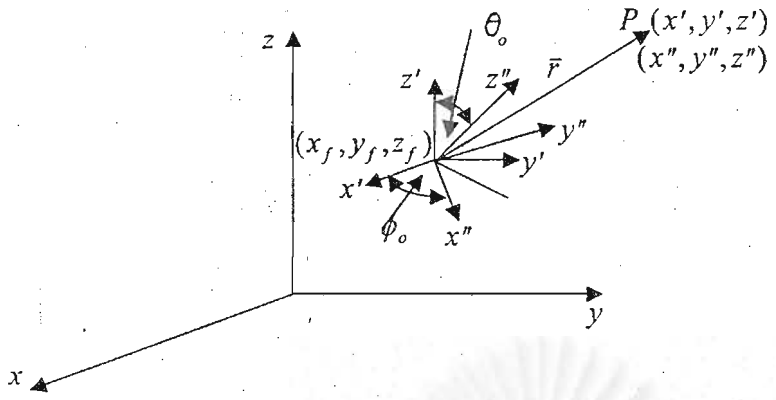
สำหรับแบบจำลองคณิตศาสตร์ต่างๆที่เกี่ยวข้องมีรายละเอียดดังนี้

1. สายอากาศป้อนกำลังคลื่น

การคำนวณแบบรูปการแผ่พลังงานย่านสนามไกลของระบบสายอากาศชนิดจานสะท้อนเดี่ยวรูปพาราโบลิกแบบสมมาตร สามารถใช้สายอากาศป้อนกำลังคลื่นที่มีโพลาไรเซชันแบบเชิงเส้นหรือแบบวงกลมก็ได้ แต่ในที่นี้เลือกใช้สายอากาศที่มีโพลาไรเซชันแบบเชิงเส้น ซึ่งแบบรูปการแผ่พลังงานย่านสนามไกลเป็นดังสมการ (1.1)

$$\bar{E}_{feed}(r'',\theta'',\phi'') = \left[f_E(\theta'')\sin(\phi'' - \phi_{pol})\bar{a}_{\theta''} + f_H(\theta'')\cos(\phi'' - \phi_{pol})\,\bar{a}_{\phi''} \right] \frac{e^{-jk_0 r''}}{r''} \qquad (1.1)$$

โดยที่ ϕ_{pol} เป็นค่าปัจจัยที่ใช้กำหนดโพลาไรเซชัน ซึ่ง $\phi_{pol} = 0$ โพลาไรเซชันของสายอากาศอยู่ในแนวแกน y'' และ $\phi_{pol} = -\pi$ 2 โพลาไรเซชันของสายอากาศอยู่ในแนวแกน x'' และ $f_E(\theta'')$ คือ สนามไฟฟ้าที่เกิดจากการวัดรอบแกนของสายอากาศในระนาบ $\phi'' - \phi_{pol} = \pi / 2$ และ $f_H(\theta'')$ คือ สนามไฟฟ้าที่เกิดจากการวัดรอบแกนของสายอากาศในระนาบ $\phi'' - \phi_{pol} = 0$ ซึ่งเป็นค่าปัจจัยที่ใช้กำหนดชนิดของสายอากาศป้อนกำลังคลื่นดังต่อไปนี้



รูปที่ 2 การวางตัวของสายอากาศป้อนกำลังคลื่นที่ตำแหน่ง (x_f,y_f,z_f) ในระบบพิกัด $x''y''z''$

1. เมื่อ $f_E(\theta'') = \cos^{q_E}\theta''$ และ $f_H(\theta'') = \cos^{q_H}\theta''$ เรียกว่าสายอากาศชนิดโคไซน์กำลังต่าง ๆ และถ้า $q_E = 1$ และ $q_H = 0$ เรียกว่าสายอากาศขั้วคู่ขนาดสั้น(short dipole)

2. ถ้า $f_E(\theta'') = 1+\cos\theta''$ และ $f_H(\theta'') = 1+\cos\theta''$ เรียกว่าแหล่งกำเนิดฮอยเกน(Huygens' source)

สายอากาศป้อนกำลังคลื่นสามารถวางตัวที่ตำแหน่งและแกนพิกัดใด ๆ ดังรูปที่ 2 ทำให้ในการหาแบบรูปการแผ่พลังงานย่านสนามไกลซึ่งอยู่ในระบบพิกัด xyz ต้องทำการหาความสัมพันธ์ของระบบพิกัดของสายอากาศป้อนกำลังคลื่นกับระบบพิกัดของแบบรูปการแผ่พลังงานย่านสนามไกล

จากรูปที่ 2 ระบบพิกัด $x''y''z''$ เป็นระบบพิกัดของสายอากาศป้อนกำลังคลื่นซึ่งเกิดจากการแปลงระบบพิกัด xyz โดยการย้ายจุดกำเนิดของระบบพิกัด xyz ไปอยู่ที่ (x_f,y_f,z_f) ทำให้ระบบพิกัด $x'y'z'$ มีความสัมพันธ์กับระบบพิกัด xyz เป็นดังสมการ (1.2)

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x-x_f \\ y-y_f \\ z-z_f \end{bmatrix} \tag{1.2}$$

และระบบพิกัด $x''y''z''$ เกิดจากหมุนแกน z' รอบแกน y เป็นมุม θ_o และหมุนแกน x' รอบแกน z เป็นมุม ϕ_o ทำให้ระบบพิกัด $x''y''z''$ มีความสัมพันธ์กับระบบพิกัด $x'y'z'$ เป็นดังสมการ (1.3)

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \begin{bmatrix} \cos\theta_o\cos\phi_o & \cos\theta_o\sin\phi_o & -\sin\theta_o \\ -\sin\phi_o & \cos\phi_o & 0 \\ \sin\theta_o\cos\phi_o & \sin\theta_o\sin\phi_o & \cos\theta_o \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{1.3}$$

ดังนั้นระบบพิกัดของระบบสายอากาศป้อนกำลังคลื่นมีความสัมพันธ์กับระบบพิกัด xyz คือ

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \begin{bmatrix} \cos\theta_o\cos\phi_o & \cos\theta_o\sin\phi_o & -\sin\theta_o \\ -\sin\phi_o & \cos\phi_o & 0 \\ \sin\theta_o\cos\phi_o & \sin\theta_o\sin\phi_o & \cos\theta_o \end{bmatrix} \begin{bmatrix} x-x_f \\ y-y_f \\ z-z_f \end{bmatrix} \tag{1.4}$$

และจากแบบรูปการแผ่พลังงานย่านสนามไกลของสายอากาศป้อนกำลังคลื่นที่มีโพลาไรเซชันชนิดเชิงเส้นดังสมการ (1.1) ซึ่งเมื่อแปลงให้อยู่ในระบบพิกัด $x''y''z''$ จะได้สมการ (1.5)

$$\begin{bmatrix} E_{x''} \\ E_{y''} \\ E_{z''} \end{bmatrix} = \begin{bmatrix} \sin\theta''\cos\phi'' & \cos\theta''\cos\phi'' & -\sin\phi'' \\ \sin\theta''\sin\phi'' & \cos\theta''\sin\phi'' & \cos\phi'' \\ \cos\theta'' & -\sin\theta'' & 0 \end{bmatrix} \begin{bmatrix} 0 \\ f_E(\theta'')\sin(\phi''-\phi_{pol}) \\ f_H(\theta'')\cos(\phi''-\phi_{pol}) \end{bmatrix} \quad (1.5)$$

และเมื่อแปลงสนามไฟฟ้าในระบบพิกัด $x''y''z''$ ให้อยู่ในระบบพิกัด xyz เป็นดังสมการ (1.6)

$$\begin{bmatrix} E_x \\ E_y \\ E_z \end{bmatrix} = \begin{bmatrix} \cos\theta_o\cos\phi_o & -\sin\phi_o & \sin\theta_o\cos\phi_o \\ \cos\theta_o\sin\phi_o & \cos\phi_o & \sin\theta_o\sin\phi_o \\ -\sin\theta_o & 0 & \cos\theta_o \end{bmatrix} \begin{bmatrix} E_{x''} \\ E_{y''} \\ E_{z''} \end{bmatrix} \quad (1.6)$$

ดังนั้นสนามไฟฟ้าย่านสนามไกลของสายอากาศป้อนกำลังคลื่นที่ตำแหน่ง xyz ใด ๆ เป็นดังสมการ (1.6) และนำมาเขียนเป็นฟังก์ชันด้วยโปรแกรม MATLAB ได้ดังนี้

```
function Exyz=feed(x,y,z)
```

กำหนดตัวแปร beta zeta0 phi0 qe qh phipol huygen ให้เป็นตัวแปร global เพื่อไม่ต้องทำการส่งมาในฟังก์ชัน โดยใช้ค่าที่ถูกกำหนดไว้ในโปรแกรม sympba.m ซึ่งเป็นโปรแกรมหลัก

```
global beta zeta0 phi0 qe qh phipol
j=sqrt(-1);
```

ทำการแปลงจุด (x,y,z) ให้ไปอยู่ในระบบพิกัด $x'y'z'$ ดังสมการ (1.2)

```
xyz1=[x-xf; y-yf; z-zf];
```

ทำการแปลงจุด (x,y,z) ให้ไปอยู่ในระบบพิกัด $x''y''z''$ ดังสมการ (1.4)

```
P=[cos(zeta0)*cos(phi0) cos(zeta0)*sin(phi0) -sin(zeta0); -sin(phi0) cos(phi0) 0; sin(zeta0)*cos(phi0) sin
(zeta0)*sin(phi0) cos(zeta0)];
xyz2=P*xyz1;
```

คำนวณค่า r'' θ'' ϕ'' ของระบบพิกัดของสายอากาศป้อนกำลังคลื่น $x''y''z''$

```
r2=norm(xyz2);
zetaf=acos(xyz2(3)/r2);
phif=atan2(xyz2(2),xyz2(1));
```

หาค่าสนามไฟฟ้าตามแบบรูปการแผ่พลังงานย่านสนามไกลดังสมการ (1.1)

```
Erzp2=[0; 0; 0];
if huygen=1
   Erzp2=[0; (1+cos(zetaf))*sin(phif-phipol); (1+cos(zetaf))*cos(phif-phipol)]*exp(-j*beta*r2)/r2;
else
   Erzp2=[0; cos(zetaf)^qe*sin(phif-phipol); cos(zetaf)^qh*cos(phif-phipol)]*exp(-j*beta*r2)/r2;
end
```

ทำการแปลงสนามไฟฟ้าในระบบพิกัดทรงกลม $r''\theta''\phi''$ ให้ไปอยู่ในระบบพิกัดคาร์ทีเซียน $x''y''z''$ ดังสมการ (1.5)

```
G=[sin(zetaf)*cos(phif) cos(zetaf)*cos(phif) -sin(phif); sin(zetaf)*sin(phif) cos(zetaf)*sin(phif) cos(phif); cos
(zetaf) -sin(zetaf) 0];
Exyz2=G*Erzp2;
```

ทำการแปลงสนามไฟฟ้าในระบบพิกัดคาร์ทีเซียน $x''y''z''$ ให้ไปอยู่ในระบบพิกัดคาร์ทีเซียน xyz ดังสมการ (1.6)

```
Exyz=P'*Exyz2;
```

2. สนามไฟฟ้าที่สะท้อนจากจานสะท้อนมายังระนาบหน้าจาน

สนามไฟฟ้าที่สะท้อนจากจานสะท้อนซึ่งเป็นพื้นผิวตัวนำสมบูรณ์มายังจุดสังเกต P(xp,yp,zp) ใด ๆ สามารถหาได้ดังสมการ (2.1)

$$\bar{E}^r(P)=\left[-\bar{E}^i(Q_R)+2(\bar{E}^i(Q_R)\cdot\hat{n})\hat{n}\right]\cdot\sqrt{\frac{\rho_1^r\rho_2^r}{(\rho_1^r+s_r)(\rho_2^r+s_r)}}\cdot e^{-jk_o s_r} \tag{2.1}$$

โดยที่ ρ_1^r, ρ_2^r คือ รัศมีความโค้งหลักของหน้าคลื่นสะท้อนที่จุดสะท้อน โดยมีความสัมพันธ์กับรัศมีความโค้งหลักของหน้าคลื่นตกกระทบที่จุดสะท้อนเช่นเดียวกับสูตรการหาระยะภาพ ระยะวัตถุของกระจกและเลนส์ ซึ่งแสดงไว้โดย Kouyoumjian และ Pathak (1970) ดังสมการ (2.2)

$$\frac{1}{\rho_{1,2}^r}=\frac{1}{2}\left(\frac{1}{\rho_1^i}+\frac{1}{\rho_2^i}\right)+\frac{1}{f_{1,2}} \tag{2.2}$$

โดยที่ ρ_1^r, ρ_2^r คือ รัศมีความโค้งหลักของหน้าคลื่นตกกระทบที่จุดสะท้อนและ f_1, f_2 คือ ระยะโฟกัสซึ่งขึ้นอยู่กับหน้าคลื่นของคลื่นตกกระทบและค่าปัจจัยต่าง ๆ ของพื้นผิวสะท้อนที่จุดสะท้อนดังสมการ (2.3)

$$\frac{1}{f_{1,2}}=\frac{1}{\cos\theta_i}\left(\frac{\sin^2\theta_2}{R_1}+\frac{\sin^2\theta_1}{R_2}\right)\pm\frac{1}{2}\left\{\left(\frac{1}{\rho_1^i}-\frac{1}{\rho_2^i}\right)^2+\left(\frac{1}{\rho_1^i}-\frac{1}{\rho_2^i}\right)\frac{4}{\cos\theta_i}\left(\frac{\Theta_{22}^2-\Theta_{12}^2}{R_1}+\frac{\Theta_{21}^2-\Theta_{11}^2}{R_2}\right)+\left[4\left[\frac{1}{\cos^2\theta_i}\left(\frac{\sin^2\theta_2}{R_1}+\frac{\sin^2\theta_1}{R_2}\right)^2+\frac{4}{R_1R_2}\right]\right]\right\}^{1/2} \tag{2.3}$$

ในสมการ (2.3) เครื่องหมายบวกใช้สำหรับ f_1 เครื่องหมายลบใช้สำหรับ f_2 และ R_1, R_2 คือ รัศมีความโค้งหลักของพื้นผิวสะท้อนที่จุดสะท้อน โดยปกติสนามที่แผ่กระจายออกจากสายอากาศป้อนกำลังคลื่นมีหน้าคลื่นแบบทรงกลม ดังนั้น $\rho_1^i=\rho_2^i=s_i$ ซึ่งเมื่อแทนลงในสมการ (2.3) จะได้

$$\frac{1}{f_{1,2}}=\frac{1}{\cos\theta_i}\left(\frac{\sin^2\theta_2}{R_1}+\frac{\sin^2\theta_1}{R_2}\right)\pm\frac{1}{2}\left\{4\left[\frac{1}{\cos^2\theta_i}\left(\frac{\sin^2\theta_2}{R_1}+\frac{\sin^2\theta_1}{R_2}\right)^2+\frac{4}{R_1R_2}\right]\right\}^{1/2} \tag{2.4}$$

ดังนั้นเมื่อนำมาเขียนเป็นฟังก์ชัน radii.m เพื่อหารัศมีความโค้งได้ดังนี้

```
function [pr1,pr2]=radii(x,y,z)
```

กำหนดตัวแปร f xf yf zf ให้เป็นตัวแปร global เพื่อไม่ต้องทำการส่งมาในฟังก์ชันโดยใช้ค่าที่ถูกกำหนดไว้ในโปรแกรม sympba.m ซึ่งเป็นโปรแกรมหลัก

```
global f xf yf zf
```

หาเวกเตอร์หนึ่งหน่วยในทิศทางตกกระทบจากตำแหน่งสายอากาศป้อนกำลังคลื่น (xf,yf,zf) มายังจุดบนพื้นผิวจานสะท้อน Q_R (x,y,z)

```
si=norm([x-xf; y-yf; z-zf]);
vsi=[x-xf; y-yf; z-zf]/si;
```

หารัศมีความโค้งหลักของพื้นผิวจานสะท้อนรูปพาราโบลิก R_1, R_2 ที่จุดสะท้อน

```
R1=2*f*sqrt(1+(x*x+y*y)/(4*f*f));
R2=2*f*(1+(x*x+y*y)/(4*f*f))^(3/2);
```

หาเวกเตอร์หนึ่งหน่วยในทิศทางหลักของพื้นผิวจานสะท้อนรูปพาราโบลิก $\hat{u}_1$, $\hat{u}_2$ ที่จุดสะท้อน

```
if [y; -x; 0]==[0; 0; 0]
   u1=[0; 0; 0];
else
   u1=[y; -x; 0]/norm([y; -x; 0]);
end
if [x; y; (x*x+y*y)/(2*f)]==[0; 0; 0]
   u2=[0; 0; 0];
else
   u2=[x; y; (x*x+y*y)/(2*f)]/norm([x; y; (x*x+y*y)/(2*f)]);
end
```

หาเวกเตอร์หนึ่งหน่วยที่ตั้งฉากกับพื้นผิวจานสะท้อนรูปพาราโบลิก $\hat{n}_s$ ที่จุดสะท้อนจากฟังก์ชัน psurf.m

```
[z,ns]=psurf(x,y);
```

หารัศมีความโค้งหลักของหน้าคลื่นสะท้อนที่จุดสะท้อนดังสมการ (2.2)

```
cosi=vsi'*ns;
sin21=1-(vsi'*u1)^2;
sin22=1-(vsi'*u2)^2;
pr1=(1/si+1/cosi*(sin22/R1+sin21/R2)+sqrt(abs(1/(cosi*cosi)*(sin22/R1+sin21/R2)^2-4/(R1*R2))))^(-1);
pr2=(1/si+1/cosi*(sin22/R1+sin21/R2)-sqrt(abs(1/(cosi*cosi)*(sin22/R1+sin21/R2)^2-4/(R1*R2))))^(-1);
```

ฟังก์ชัน psurf.m เป็นฟังก์ชันที่หาเวกเตอร์หนึ่งหน่วยที่ตั้งฉากกับพื้นผิวจานสะท้อนรูปพาราโบลิกและหาตำแหน่งของพิกัด z บนพื้นผิวจานสะท้อนรูปพาราโบลิกเป็นดังนี้

```
function [z,ns]=psurf(x,y)
```

กำหนดตัวแปร f ให้เป็นตัวแปร global เพื่อไม่ต้องทำการส่งมาในฟังก์ชันโดยใช้ค่าที่ถูกกำหนดไว้ในโปรแกรม sympba.m ซึ่งเป็นโปรแกรมหลัก

```
global f
```

หาตำแหน่งพิกัด z บนพื้นผิวจานสะท้อนรูปพาราโบลิก

```
z=(x*x+y*y)/(4*f)-f;
```

หาเวกเตอร์หนึ่งหน่วยที่ตั้งฉากกับพื้นผิวจานสะท้อนรูปพาราโบลิก

```
ns=[-x; -y; 2*f]/norm([-x; -y; 2*f]);
```

จากสมการ (2.1) ต้องหาจุดสะท้อนที่ทำให้เกิดสนามไฟฟ้าที่จุดสังเกต P(xp,yp,zp) เมื่อสายอากาศป้อนกำลังคลื่นอยู่ที่ (xf,yf,zf) เพื่อหาสนามไฟฟ้าจากสายอากาศป้อนกำลังคลื่นที่ตกกระทบที่จุดนั้น ในการหาจุดสะท้อนนั้นทำได้โดยพิจารณาจากเงื่อนไขการสะท้อนคือ

1. มุมตกกระทบเท่ากับมุมสะท้อน ($\hat{s}_i \cdot \hat{n}_s = -\hat{s}_r \cdot \hat{n}_s$)
2. รังสีตกกระทบและรังสีสะท้อนต้องอยู่ในระนาบเดียวกัน ($\hat{s}_r \cdot (\hat{n}_s \times \hat{s}_i) = 0$)

และใช้สมการพื้นผิวจานสะท้อนรูปพาราโบลิกเป็นสมการช่วย ซึ่งเป็นการแก้สมการแบบไม่เชิงเส้น สำหรับในที่นี้ได้ใช้กรรมวิธีเชิงเลขที่เรียกว่า กรรมวิธีของนิวตัน (Newton Method) ซึ่งในโปรแกรม MATLAB ทำได้โดยใช้ฟังก์ชัน fsolve.m โดยกำหนดสมการเชิงเส้นไว้ในฟังก์ชัน โดยในที่นี้กำหนดไว้ในฟังก์ชัน xyz3.m ดังนี้

```
function q=xyz3(p)
```

กำหนดตัวแปร xf yf zf xp yp zp f ให้เป็นตัวแปร global เพื่อไม่ต้องทำการส่งมาในฟังก์ชันโดยใช้ค่าที่ถูกกำหนดไว้ในโปรแกรม sympba.m ซึ่งเป็นโปรแกรมหลัก

```
global xf yf zf xp yp zp f
x=p(1); y=p(2); z=(x^2+y^2)/(4*f)-f;
q=zeros(2,1);
n=[-x; -y; 2*f]/norm([-x; -y; 2*f]);
si=[x-xf; y-yf; z-zf]/norm([x-xf; y-yf; z-zf]);
sr=[xp-x; yp-y; zp-z]/norm([xp-x; yp-y; zp-z]);
```

มุมตกกระทบเท่ากับมุมสะท้อน

```
q(1)=si'*n+sr'*n;
```

รังสีตกกระทบและรังสีสะท้อนต้องอยู่ในระนาบเดียวกัน

```
q(2)=sr'*cross(si,n);
```

เมื่อได้ตำแหน่งของจุดสะท้อนสามารถหาสนามไฟฟ้าที่จุด P(xp,yp,zp) ได้ดังสมการ (2.1) ซึ่งเขียนเป็นฟังก์ชัน go.m ได้ดังนี้

```
function er=go(xp,yp,zp)
```

กำหนดตัวแปร beta xf yf zf r f ให้เป็นตัวแปร global เพื่อไม่ต้องทำการส่งมาในฟังก์ชันโดยใช้ค่าที่ถูกกำหนดไว้ในโปรแกรม sympba.m ซึ่งเป็นโปรแกรมหลัก

```
global beta xf yf zf r f
```

```
j=sqrt(-1);
```

กำหนดความละเอียดและจำนวนครั้งในการวนซ้ำในกรรมวิธีนิวตันไว้ในตัวแปร options เพื่อใช้ในฟังก์ชัน fsolve.m

```
options(1)=0; options(2)=1e-11; options(3)=1e-11; options(4)=1e-12; options(14)=5000; options(18)=1;
```

ทำการคำนวณหาจุดสะท้อนโดยกรรมวิธีนิวตัน

```
if (xp==0 & yp==0 & xf==0 & yf==0)
  X=[0; 0];
else
  X=fsolve('xyz7',[xp*0.99; yp*0.99],options);
end
xr=X(1);
yr=X(2);
```

หาเวกเตอร์หนึ่งหน่วยที่ตั้งฉากกับพื้นผิวจานสะท้อนรูปพาราโบลิก $\hat{n}_s$ ที่จุดสะท้อนจากฟังก์ชัน psurf.m

```
[zr,ns]=psurf(xr,yr);
```

หาสนามไฟฟ้าจากสายอากาศป้อนกำลังคลื่นที่ตกกระทบพื้นผิวสะท้อนที่จุด (xr,yr,zr) จากฟังก์ชัน feed.m

```
ei=feed(xr,yr,zr);
```

หารัศมีความโค้งหลักของหน้าคลื่นสะท้อนจากฟังก์ชัน radii.m

```
[pr1,pr2]=radii(xr,yr,zr,xf,yf,zf);
sr=norm([xp-xr; yp-yr; zp-zr]);
vsr=[xp-xr; yp-yr; zp-zr]/sr;
A=sqrt(1/((1+sr/pr1)*(1+sr/pr2)));
```

หาสนามไฟฟ้าสะท้อนมายังจุดสังเกต (xp,yp,zp) ดังสมการ (2.1)

```
er=(-ei+2*(ns'*ei)*ns)*A*exp(-j*beta*sr);
er=nearzero(er);
```

3. สนามไฟฟ้าที่เลี้ยวเบนจากขอบของจานสะท้อนมายังระนาบหน้าจาน

สนามไฟฟ้าที่เลี้ยวเบนจากขอบของจานสะท้อนซึ่งเป็นพื้นผิวตัวนำสมบูรณ์มายังจุดสังเกต P(xp,yp,zp) ใด ๆ สามารถหาได้ดังสมการ (3.1)

$$E^d_{\hat{\beta}_o}(P) = -E'_{\hat{\beta}'_o}(Q_D)\cdot D_s(\phi_o,\phi'_o;\beta_o)\quad \frac{\rho_d}{s_d(\rho_d+s_d)}\, e^{-jk_o s}$$

$$E^d_{\hat{\phi}_o}(P) = -E'_{\hat{\phi}'_o}(Q_D)\cdot D_h(\phi_o,\phi'_o;\beta_o)\quad \frac{\rho_d}{s_d(\rho_d+s_d)}\, e^{-jk_o s_d} \qquad (3.1)$$

โดยที่ $\bar{E}^i(Q_D)$ คือ สนามจากสายอากาศป้อนกำลังคลื่นมายังที่ขอบ (Q_D) และ $\hat{\phi}_o$, $\hat{\phi}'_o$ คือ เวกเตอร์หนึ่งหน่วยที่ตั้งฉากกับระนาบการเลี้ยวเบนและระนาบการตกกระทบตามลำดับ และ $\hat{\beta}_o$, $\hat{\beta}'_o$ คือ คือเวกเตอร์หนึ่งหน่วยที่ขนานกับระนาบการเลี้ยวเบนและระนาบการตกกระทบตามลำดับซึ่งสามารถเขียนเป็นสมการดังสมการ (3.2)

$$\hat{\phi}'_o = \hat{e}\times\hat{s}' \;\; \hat{e}\times\hat{s}', \quad \hat{\beta}'_o = \hat{s}'\times\hat{\phi}'_o \text{ และ } \hat{\phi}_o = \hat{s}\times\hat{e} \;\; \hat{s}\times\hat{e}, \quad \hat{\beta}_o = \hat{s}\times\hat{\phi}_o \tag{3.2}$$

โดยที่ $\hat{e}$ คือ เวกเตอร์หนึ่งหน่วยที่ขนานกับขอบ และ $\tilde{D}$ คือ สัมประสิทธิ์การเลี้ยวเบนเท่ากับ

$$\tilde{D} = -\hat{\beta}'_o\hat{\beta}_o D_s - \hat{\phi}'_o\hat{\phi}_o D_h \tag{3.3}$$

$$D_{sh}(\phi_o,\phi'_o;\beta_o) = -\frac{e^{-j\pi/4}}{2\;2\pi k_o\sin\beta_o}\left\{\frac{F\left[2k_oL'\cos^2\left[(\phi_o-\phi'_o)/2\right]\right]}{\cos\left[(\phi_o-\phi'_o)/2\right]} \mp \frac{F\left[2k_oL'\cos^2\left[(\phi_o+\phi'_o)/2\right]\right]}{\cos\left[(\phi_o+\phi'_o)/2\right]}\right\} \tag{3.4}$$

$$L' = \frac{s(\rho'_e+s)\rho'_1\rho'_2\sin^2\beta_o}{\rho'_e(\rho'_1+s)(\rho'_2+s)} \quad \text{และ} \quad L^r = \frac{s(\rho^r_e+s)\rho^r_1\rho^r_2\sin^2\beta_o}{\rho^r_e(\rho^r_1+s)(\rho^r_2+s)} \tag{3.5}$$

$$F[x] = 2j\;\; xe^{jx}\int_x^\infty e^{-j\tau^2}d\tau \tag{3.6}$$

โดยที่ ρ'_1, ρ'_2 คือ รัศมีหลักของความโค้งของคลื่นตกกระทบที่จุดเลี้ยวเบนและ ρ^r_1, ρ^r_2 คือ รัศมีหลักของความโค้งของคลื่นสะท้อนจากพื้นผิวที่จุดเลี้ยวเบนและ ρ^r_e คือ ระยะ caustic ของรังสีการเลี้ยวเบนในทิศทางของรังสีที่สะท้อนจากพื้นผิวที่จุดเลี้ยวเบนและ ρ_a คือ ระยะทาง caustic ของรังสีการเลี้ยวเบนเป็นดังนี้

$$\frac{1}{\rho_d} = \frac{1}{\rho'_e} - \frac{\hat{n}_e\cdot(\hat{s}'-\hat{s})}{R_E\sin\beta_o} \tag{3.7}$$

$$\frac{1}{\rho^r_e} = \frac{1}{\rho'_e} - \frac{2(\hat{n}\cdot\hat{n}_e)(\hat{s}'\cdot\hat{n})}{R_E\sin^2\beta_o} \tag{3.8}$$

โดยที่ ρ'_e คือ รัศมีความโค้งของหน้าคลื่นตกกระทบที่จุดเลี้ยวเบนในระนาบที่ประกอบด้วยรังสีตกกระทบ ($\hat{s}'$) กับเวกเตอร์หนึ่งหน่วยที่ขนานกับขอบ ($\hat{e}$) และ R_E คือ รัศมีความโค้งของขอบ และทิศทางของเวกเตอร์หนึ่งหน่วยที่ขนานกับขอบเป็นดังนี้

$$\hat{t} = \hat{e}\times\hat{n} \tag{3.9}$$

โดยที่ $\hat{t}$ คือ เวกเตอร์หนึ่งหน่วยที่ขนานพื้นผิวของจานสะท้อนและ $\hat{n}_s$ คือ เวกเตอร์หนึ่งหน่วยที่ตั้งฉากกับพื้นผิวของจานสะท้อนและ $\hat{n}_e$ คือเวกเตอร์หนึ่งหน่วยที่ตั้งฉากกับขอบและมีทิศพุ่งออกจากศูนย์กลางความโค้ง ดังนั้นสนามการเลี้ยวเบนที่ขอบดังสมการ (3.1) เขียนเป็นฟังก์ชัน gtd.m ได้ดังนี้

```
function ed=gtd(xp,yp,zp)
```

กำหนดตัวแปร beta r xf yf zf ให้เป็นตัวแปร global เพื่อไม่ต้องทำการส่งมาในฟังก์ชันโดยใช้ค่าที่ถูกกำหนดไว้ในโปรแกรม sympba.m ซึ่งเป็นโปรแกรมหลัก

```
global r xf yf zf
```

กำหนดความละเอียดและจำนวนครั้งในการวนซ้ำของกรรมวิธีนิวตันไว้ในตัวแปร options เพื่อใช้ในฟังก์ชัน fsolve.m

```
options(1)=0; options(2)=1e-11; options(3)=1e-11; options(4)=1e-12; options(14)=5000; options(18)=1;
```

กำหนดค่าเริ่มต้นเพื่อใช้ในการวนซ้ำโดยกำหนดมุมเริ่มต้นไว้ครบทั้ง 4 จตุภาค

```
esp1=[-0.05];
esp2=[pi/2-0.05];
esp3=[pi-0.05];
esp4=[3*pi/2-0.05];
```

ทำการคำนวณหาจุดเลี้ยวเบนที่ขอบโดยกรรมวิธีนิวตันจากฟังก์ชัน fsolve โดยกำหนดสมการไม่เชิงเส้นไว้ในฟังก์ชัน xyz4.m

```
if (xp==0 & yp==0 & xf==0 & yf==0)
   X1=[0];
   A1=[r*cos(X1(1)); r*sin(X1(1))]';
else
  X1=fsolve('xyz4',esp1',options);
  A1=[r*cos(X1(1)); r*sin(X1(1))];
  X2=fsolve('xyz4',esp2',options);
  A2=[r*cos(X2(1)); r*sin(X2(1))];
  X3=fsolve('xyz4',esp3',options);
  A3=[r*cos(X3(1)); r*sin(X3(1))];
  X4=fsolve('xyz4',esp4',options);
  A4=[r*cos(X4(1)); r*sin(X4(1))];
end

xd=A1(1); yd=A1(2);
[zd,ns]=psurf(xd,yd);
ed=[0; 0; 0]; ed1=[0; 0; 0]; ed2=[0; 0; 0]; ed3=[0; 0; 0]; ed4=[0; 0; 0];
```

หาสนามไฟฟ้าที่เลี้ยวเบนจากขอบมายังจุดสังเกต (xp,yp,zp) จากฟังก์ชัน diffrac.m

```
[ed1,check]=diffrac(xp,yp,zp,xd,yd,zd,ns);
```

ตรวจสอบว่าจุด (xp,yp,zp) เป็นจุด Cuastic หรือไม่ ถ้าเป็นตัวแปร check เท่ากับ 1 ถ้าไม่เป็นตัวแปร check เท่ากับ 0 และตรวจสอบว่า ค่าจุดเลี้ยวเบนที่ขอบที่ได้จากกรรมวิธีนิวตันจากค่าเริ่มต้นทั้ง 4 ว่าซ้ำกันหรือไม่ ถ้าซ้ำก็ไม่ต้องทำการหาสนามไฟฟ้าที่เลี้ยวเบนจากขอบ

```
if check==0
```

```
  ed=ed1;
  if (abs(A1(1)-A2(1))>10^(-10) | abs(A1(2)-A2(2))>10^(-10))
    xd=A2(1); yd=A2(2); [zd,ns]=psurf(xd,yd);
    [ed2,check]=diffrac(xp,yp,zp,xd,yd,zd,xf,yf,zf,ns);
    ed=ed+ed2;
  else
    if (abs(A1(1)-A3(1))>10^(-10) | abs(A1(2)-A3(2))>10^(-10)) & (abs(A2(1)-A3(1))>10^(-10) | abs(A2
(2)-A3(2))>10^(-10))
      xd=A3(1); yd=A3(2); [zd,ns]=psurf(xd,yd);
      [ed3,check]=diffrac(xp,yp,zp,xd,yd,zd,xf,yf,zf,ns);
      ed=ed+ed3;
    else
      if (abs(A1(1)-A4(1))>10^(-10) | abs(A1(2)-A4(2))>10^(-10)) & (abs(A2(1)-A4(1))>10^(-10) | abs
(A2(2)-A4(2))>10^(-10)) & ...
        (abs(A3(1)-A4(1))>10^(-10) | abs(A3(2)-A4(2))>10^(-10))
        xd=A4(1); yd=A4(2); [zd,ns]=psurf(xd,yd);
        [ed4,check]=diffrac(xp,yp,zp,xd,yd,zd,xf,yf,zf,ns);
        ed=ed+ed4;
      end
    end
  end
else
  ed=ed1;
end
```

เช่นเดียวกับการหาสนามไฟฟ้าที่สะท้อนจากจานสะท้อน การหาสนามการเลี้ยวเบนก็ต้องทำการหาจุดเลี้ยวเบนที่ขอบซึ่งสามารถหาได้จากเงื่อนไขการเลี้ยวเบนคือ $\hat{s}_i \cdot \hat{e} = \hat{s}_d \cdot \hat{e}$ และมีสมการช่วยคือ สมการพื้นผิวสะท้อนรูปพาราโบลิกและสมการขอบ ซึ่งกรณีสายอากาศชนิดจานสะท้อนเดี่ยวสมมาตรที่มีเส้นผ่าศูนย์กลางเท่ากับ D ทำให้สามารถเขียน xd และ yd อยู่ในระบบเชิงขั้วซึ่งทำให้ลดตัวแปรที่ไม่ทราบเหลือเพียงเป็นฟังก์ชันของมุมเท่านั้น ดังนั้นเมื่อใช้กรรมวิธีนิวตันคำนวณหาจากฟังก์ชัน fsolve.m ทำให้กำหนดสมการแบบไม่เชิงเส้นไว้ในฟังก์ชัน xyz4.m ดังนี้

```
function q=xyz4(p)
```

กำหนดตัวแปร xf yf zf xp yp zp f ให้เป็นตัวแปร global เพื่อไม่ต้องทำการส่งมาในฟังก์ชันโดยใช้ค่าที่ถูกกำหนดไว้ในโปรแกรม sympba.m ซึ่งเป็นโปรแกรมหลัก

```
global xf yf zf xp yp zp f r
phia=p(1); x=r*cos(phia); y=r*sin(phia); z=(x^2+y^2)/(4*f)-f;
q=zeros(1,1);
```

```
e=[r*sin(phia); -r*cos(phia); 0];
ve=e/norm(e);
si=[x-xf; y-yf; z-zf]/norm([x-xf; y-yf; z-zf]);
sd=[xp-x; yp-y; zp-z]/norm([xp-x; yp-y; zp-z]);
```

เงื่อนไขการเลี้ยวเบนดังสมการ

```
q(1)=si'*ve-sd'*ve;
```

สำหรับการคำนวณค่าสนามไฟฟ้าที่เลี้ยวเบนจากจุดเลี้ยวเบน (xd,yd,zd) มายังจุดสังเกต (xp,,yp,zp) หาได้จากฟังก์ชัน diffrac.m ดังนี้

```
function [ed,check]=diffrac(xp,yp,zp,xd,yd,zd,ns)
```

กำหนดตัวแปร beta xf yf zf r ให้เป็นตัวแปร global เพื่อไม่ต้องทำการส่งมาในฟังก์ชันโดยใช้ค่าที่ถูกกำหนดไว้ในโปรแกรม sympba.m ซึ่งเป็นโปรแกรมหลัก

```
global beta xf yf zf r
j=sqrt(-1);
phi=atan2(yd,xd);
```

หาเวกเตอร์หนึ่งหน่วยในทิศทางตกกระทบจากสายอากาศป้อนกำลังคลื่นมายังจุดเลี้ยวเบนที่ขอบ

```
si=norm([xd-xf; yd-yf; zd-zf]);
vsi=[xd-xf; yd-yf; zd-zf]/si;
```

หาเวกเตอร์หนึ่งหน่วยในทิศทางเลี้ยวเบนจากขอบไปยังจุดสังเกต

```
sd=norm([xp-xd; yp-yd; zp-zd]);
vsd=[xp-xd; yp-yd; zp-zd]/sd;
```

หาอนุพันธ์อันดับหนึ่งของเวกเตอร์ในทิศทางความโค้งของขอบจากจุดศูนย์กลางความโค้งไปยังจุดเลี้ยวเบน

```
vr1=[-r*sin(phi); r*cos(phi); 0];
```

หาอนุพันธ์อันดับสองของเวกเตอร์ในทิศทางความโค้งของขอบจากจุดศูนย์กลางความโค้งไปยังจุดเลี้ยวเบน

```
vr2=[-r*cos(phi); -r*sin(phi); 0];
```

หาเวกเตอร์หนึ่งหน่วยที่ขนานกับขอบ

```
ve=-vr1/norm(vr1);
```

หาเวกเตอร์หนึ่งหน่วยที่ตั้งฉากกับขอบโดยที่ $\hat{b} = \hat{e} \times \hat{n}_e$

```
vb=cross(vr1,vr2)/norm(cross(vr1,vr2));
```

หาเวกเตอร์หนึ่งหน่วยที่ตั้งฉากกับขอบและมีทิศทางพุ่งออกจากจุดศูนย์กลางความโค้งของขอบ

```
vne=cross(vb,ve);
```

รัศมีความโค้งของขอบ

```
Re=(norm(vr1)^3)/norm(cross(vr1,vr2));
```

หาเวกเตอร์ที่ขนานกับพื้นผิวจานสะท้อนรูปพาราโบลิกดังสมการ (3.9)

```
vts=cross(ve,ns);
```

หาเวกเตอร์หนึ่งหน่วยที่ตั้งฉากกับระนาบการตกกระทบดังสมการ (3.2)

vphioi=cross(ve,vsi)/norm(cross(ve,vsi));

หาเวกเตอร์หนึ่งหน่วยที่ขนานกับระนาบการตกกระทบ

vbetaoi=cross(vsi,vphioi);

หาเวกเตอร์หนึ่งหน่วยที่ตั้งฉากกับระนาบการเลี้ยวเบน

vphiod=cross(vsd,ve)/norm(cross(vsd,ve));

หาเวกเตอร์หนึ่งหน่วยที่ขนานกับระนาบการเลี้ยวเบน

vbetaod=cross(vsd,vphiod);

หาค่า ไซน์ฟังก์ชันของมุมตกกระทบหรือมุมเลี้ยวเบน

sinbo=sqrt(1-(vsd'*ve)^2);

รัศมีความโค้งของหน้าคลื่นตกกระทบที่จุดเลี้ยวเบนในระนาบที่ประกอบด้วยรังสีตกกระทบ ($\hat{s}'$) กับเวกเตอร์หนึ่งหน่วยที่ขนานกับขอบ ($\hat{e}$) เนื่องจากคลื่นที่มาจากสายอากาศป้อนกำลังคลื่นเป็นหน้าคลื่นทรงกลมดังนั้น รัศมีความโค้งของหน้าคลื่นในทุกระนาบเท่ากับระยะทางที่คลื่นเคลื่อนที่จากสายอากาศป้อนกำลังคลื่นมายังจุดเลี้ยวเบน

pei=si;

หาระยะทาง caustic ของรังสีการเลี้ยวเบน (pd)

vsi_sd=vsi-vsd;

pd=(1/pei-(vne'*vsi_sd)/(Re*sinbo*sinbo))^(-1);

ทำการตรวจสอบว่าจุดสังเกตว่าเป็นจุด caustic หรือไม่ ซึ่งจะเป็นจุด caustic เมื่อ sd+pd=0 แต่เนื่องจากในการคำนวณโดยโปรแกรมนั้นค่า sd+pd ที่คำนวณจะไม่เท่ากับศูนย์พอดี แต่จะเกิดออฟเซตดังนั้นในการตรวจสอบเงื่อนไขจึงต้องทำการเผื่อออฟเซตไว้ 10^{-8}

if abs(sd+pd)>10^(-8)

ถ้าจุดสังเกตไม่เป็นจุด Caustic ทำการคำนวณหามุม ϕ_o ϕ_o'

```
  if vts'*vphioi<=0
     phioi=acos(ns'*vphioi);
  else
     phioi=2*pi-acos(ns'*vphioi);
  end

  if vts'*vphiod<=0
     phiod=acos(ns'*vphiod);
  else
    phiod=2*pi-acos(ns'*vphiod);
  end
```

รัศมีความโค้งของหน้าคลื่นตกกระทบที่จุดเลี้ยวเบน เนื่องจากคลื่นที่มาจากสายอากาศป้อนกำลังคลื่นเป็นหน้าคลื่นทรงกลมดังนั้น รัศมีความโค้งของหน้าคลื่นในทุกระนาบเท่ากับระยะทางที่คลื่นเคลื่อนที่จากสายอากาศป้อนกำลังคลื่นมายังจุดเลี้ยวเบน

```
p1i=si;
p2i=si;
```

หาระยะ caustic ของรังสีการเลี้ยวเบนในทิศทางของรังสีสะท้อนจากพื้นผิวที่จุดเลี้ยวเบนดังสมการ (3.8)

```
per=(1/pei-(2*(ns'*vne)*(vsi'*ns))/(Re*sinbo*sinbo))^(-1);
```

หารัศมีความโค้งหลักของหน้าคลื่นสะท้อน

```
[p1r,p2r]=radii(xd,yd,zd,xf,yf,zf);
```

หาค่าปัจจัยที่กำหนดในทรานซิชันฟังก์ชันดังสมการ (3.5)

```
Li=(sd*(1+sd/pei)*sinbo*sinbo)/((1+sd/p1i)*(1+sd/p2i));
Lr=(sd*(1+sd/per)*sinbo*sinbo)/((1+sd/p1r)*(1+sd/p2r));
```

หาสนามไฟฟ้าที่เลี้ยวเบนที่ขอบตามสมการ (3.1)

```
ei=feed(xd,yd,zd,xf,yf,zf);
ephioi=vphioi'*ei;
ebetaoi=vbetaoi'*ei;

A=sqrt(1/(sd*(1+sd/pd)));
```

หาค่าของทรานซิชันฟังก์ชันดังสมการ (3.6) โดยการประมาณจากฟังก์ชัน fresnel.m

```
f1=fresnel(beta*Li*2*cos((phiod-phioi)/2)^2)/cos((phiod-phioi)/2);
f2=fresnel(beta*Lr*2*cos((phiod+phioi)/2)^2)/cos((phiod+phioi)/2);
Ds=-exp(-j*pi/4)/(sqrt(8*pi*beta)*sinbo)*(f1-f2);
Dh=-exp(-j*pi/4)/(sqrt(8*pi*beta)*sinbo)*(f1+f2);
ephiod=-ephioi*Dh*A*exp(-j*beta*sd);
ebetaod=-ebetaoi*Ds*A*exp(-j*beta*sd);
ed=ephiod*vphiod+ebetaod*vbetaod;
check=0;
else
```

เมื่อจุดสังเกตเป็นจุด caustic สนามไฟฟ้าที่เลี้ยวเบนจากขอบสามารถหาได้โดยกรรมวิธีกระแสสมมูลที่ขอบ โดยทำการหาค่าแหล่งกระแสสมมูลที่ทำให้เกิดสนามไฟฟ้าที่จุดสังเกตนั้นจากฟังก์ชัน eec.m แล้วส่งเข้าทำการอินทิเกรตแหล่งกระแสสมมูลจากฟังก์ชัน sim1.m

```
[Ie,Me,np]=eec(xp,yp,zp);
xx=[xp yp zp];
ed=sim1(Ie,Me,0,2*pi,np,xx);
check=1;
end
```

จากการคำนวณหาสนามไฟฟ้าเลี้ยวเบนจากขอบมีการประมาณค่าทรานซิชันฟังก์ชันที่ปรากฏอยู่ในสัมประสิทธิ์การเลี้ยวเบนซึ่งการประมาณดังกล่าวเป็นดังฟังก์ชัน fresnel.m ดังนี้

```
function F=fresnel(x)
```

```
% Boersma's constant
a(1)=1.595769140; a(2)=-0.000001702; a(3)=-6.808568854; a(4)=-0.000576361; a(5)=6.920691902;
a(6)=-0.016898657; a(7)=-3.050485660; a(8)=-0.075752419; a(9)=0.850663781; a(10)=-0.025639041;
a(11)=-0.150230960; a(12)=0.034404779;
b(1)=-0.000000033; b(2)=4.255387524; b(3)=-0.000092810; b(4)=-7.780020400; b(5)=-0.009520895;
b(6)=5.075161298; b(7)=-0.138341947; b(8)=-1.363729124; b(9)=-0.403349276; b(10)=0.702222016;
b(11)=-0.216195929; b(12)=0.019547031;
c(1)=0; c(2)=-0.024933975; c(3)=0.000003936; c(4)=0.005770956; c(5)=0.000689892; c(6)=-0.009497136;
c(7)=0.011948809; c(8)=-0.006748873; c(9)=0.000246420; c(10)=0.002102967; c(11)=-0.001217930;
c(12)=0.000233939;
d(1)=0.199471140; d(2)=0.000000023; d(3)=-0.009351341; d(4)=0.000023006; d(5)=0.004851466;
d(6)=0.001903218; d(7)=-0.017122914; d(8)=0.029064067; d(9)=-0.027928955; d(10)=0.016497308;
d(11)=-0.005598515; d(12)=0.000838386;


xs=abs(x);
x1=sqrt(2*xs/pi);
x2=pi*(x1^2)/2;
f1=0;
j=sqrt(-1);
if (x2>=0) & (x2<4)
  for n=1:12
     f1=f1+(a(n)+j*b(n))*(x2/4)^(n-1);
  end
  f1=f1*exp(-j*x2)*sqrt(x2/4);
else
  for n=1:12
     f1=f1+(c(n)+j*d(n))*(4/x2)^(n-1);
  end
  f1=(1-j)/2+f1*exp(-j*x2)*sqrt(4/x2);
end
F=j*sqrt(2*pi*xs)*exp(j*xs)*(0.5-j*0.5-f1);

if (x<0)
  F=conj(F);
end
```

เมื่อเกิดจุด caustic ขึ้นสนามไฟฟ้าที่เลี้ยวเบนจากขอบมายังจุดนั้นหาได้โดยใช้กรรมวิธีกระแสสมมูลที่ขอบซึ่งแหล่งกระแสสมมูลเป็นดังสมการ (3.10) และหาได้จากฟังก์ชัน eec.m ดังนี้

$$I_e(\phi) = -e^{-j\pi/4}\sqrt{\frac{8\pi}{k}}\, Y_o(\bar{E}^i \cdot \hat{e}) D_s$$

$$M_e(\phi) = -e^{-j\pi/4}\sqrt{\frac{8\pi}{k}}\, Z_o(\bar{H}^i \cdot \hat{e}) D_h \qquad (3.10)$$

```
function [Ie,Me,np]=eec(xp,yp,zp)
global beta r f xf yf zf Zo Yo
j=sqrt(-1);
Lamda=2*pi/beta;
np=ceil(2*pi*r/(Lamda/8));
if rem(np,2)~=0
   np=np+1;
end
dphi=2*pi/np;
mp=1;
for phi=0:dphi:2*pi
    xd=r*cos(phi); yd=r*sin(phi);
    [zd,ns]=psurf(xd,yd);
    ei=feed(xd,yd,zd,xf,yf,zf);
    si=norm([xd-xf; yd-yf; zd-zf]);
    vsi=[xd-xf; yd-yf; zd-zf]/si;
    [Ds,Dh,ve]=dc(xp,yp,zp,xd,yd,zd,xf,yf,zf,ns);
    hi=Yo*cross(vsi,ei);
    Ie(mp)=-exp(-j*pi/4)*sqrt(8*pi/beta)*Yo*(ve'*ei)*Ds;
    Me(mp)=-exp(-j*pi/4)*sqrt(8*pi/beta)*Zo*(ve'*hi)*Dh;
    mp=mp+1;
end
```

สนามไฟฟ้าที่เลี้ยวเบนจากขอบมายังจุด caustic นั้นหาได้ดังสมการ (3.11) โดยทำการอินทิเกรตแหล่งกระแสกระแสสมมูลที่หาได้จากฟังก์ชัน eec.m โดยใช้กรรมวิธีซิมสันดังฟังก์ชัน sim1.m

$$\bar{E}(P) = \frac{jkZ_o}{4\pi}\int_0^{2\pi}\left[\hat{s}_r \times (\hat{s}_r \times \hat{e}) I_e(\phi) + Y_o(\hat{s}_r \times \hat{e}) M_e(\phi)\right]\frac{e^{-jks_r}}{s_r}\bar{r}''(\phi)\, d\phi \qquad (3.11)$$

โดยที่ $\bar{r}''(\phi)$ คือ อนุพันธ์อันดับที่สองเทียบกับ ϕ ของเวกเตอร์ในทิศทางความโค้งของขอบจากจุดศูนย์กลางความโค้งไปยังจุดเลี้ยวเบน $\bar{r}(\phi) = r\cos\phi\,\bar{a}_x + r\sin\phi\,\bar{a}_y$

```
function area=sim1(E1,E2,a,b,n,xx)
h=(b-a)/n;
x0=[0; 0; 0];
x0=expr1(E1(1),E2(1),a,xx)+expr1(E1(n+1),E2(n+1),b,xx);
x1=[0; 0; 0];
x2=[0; 0; 0];
for xi=2:n
    x=a+(xi-1)*h;
    if rem(xi-1,2)==0
      x2=x2+expr1(E1(xi),E2(xi),x,xx);
    else
      x1=x1+expr1(E1(xi),E2(xi),x,xx);
    end
end
area=h*(x0+2*x2+4*x1)/3;
```

ในฟังก์ชัน sim1.m มีการเรียกใช้ฟังก์ชัน expr1.m เพื่อทำการคำนวณค่าของฟังก์ชันที่ถูกทำการอินทิเกรตที่จุดต่าง ๆ ที่แบ่งภายในขอบเขตของการอินทิเกรตดังนี้

```
function value=expr1(E1,E2,x,xx)
global beta r f Zo Yo
j=sqrt(-1);
xd=r*cos(x); yd=r*sin(x);
sr=norm([xx(1)-xd; xx(2)-yd; xx(3)-((xd^2+yd^2)/(4*f)-f)]);
vsr=[xx(1)-xd; xx(2)-yd; xx(3)-((xd^2+yd^2)/(4*f)-f)]/sr;
vr1=[-r*sin(x); r*cos(x); 0];
vr2=[-r*cos(x); -r*sin(x); 0];
ve=-vr1/norm(vr1);
value=j*beta*Zo/(4*pi)*(cross(vsr,cross(vsr,ve))*E1+Yo*cross(vsr,ve)*E2)*exp(-j*beta*sr)/sr*norm(vr2);
```

4. แบบรูปการแผ่พลังงานย่านสนามไกล

แบบรูปการแผ่พลังงานย่านสนามไกลของสายอากาศชนิดจานสะท้อนเดี่ยวรูปพาราโบลิกสามารถหาได้ดังสมการ (4.1)

$$\bar{E}(r,\theta,\phi) = \frac{jk_o e^{-jk_o r}}{2\pi r}\left[\left(f_x\cos(\phi)+f_y\sin(\phi)\right)\bar{a}_\theta + \left(f_x\sin(\phi)+f_y\cos(\phi)\right)\cos(\theta)\bar{a}_\phi\right] \qquad (4.1)$$

โดยที่ f_x และ f_y คือ สเปกตรัมคลื่นระนาบซึ่งเกิดจากการแปลงฟูริเยร์สนามบนระนาบหน้าจานดังสมการ (4.2)

$$f_x(k_x,k_y) = \iint_S E_{ap}(x,y)e^{jk_x x + jk_y y}\,dxdy$$
$$f_y(k_x,k_y) = \iint_S E_{ap}(x,y)e^{jk_x x + jk_y y}\,dxdy \qquad (4.2)$$

โดยที่ $k_x = k_o \sin(\theta)\cos(\phi)$, $k_y = k_o \sin(\theta)\sin(\phi)$ ซึ่งแสดงอยู่ในโปรแกรมหลัก sympba.m ดังนี้

ทำการลบตัวแปรต่าง ๆ ในหน่วยความจำ

```
clear;
```

กำหนด beta xf yf zf xp yp zp r f zeta0 phi0 qe qh phipol Zo Yo ให้เป็นตัวแปรแบบ global variable เพื่อให้ฟังก์ชันต่าง ๆ เรียกใช้

```
global beta xf yf zf xp yp zp r f zeta0 phi0 qe qh phipol Zo Yo
```

ทำการกำหนดค่าปัจจัยต่าง ๆ ของสายอากาศชนิดจานสะท้อนเดี่ยวรูปพาราโบลิกแบบสมมาตร

```
%Freq=input('Operating frequency (Hz) : ');
Freq=4*10^9;
%FtoD=input('F/D ratio : ');
FtoD=0.37;
Lamda=(3*10^8)/Freq;
%D=input('Diameter of paraboloid antenna (m) : ');
D=10*Lamda;
f=FtoD*D;
beta=2*pi/Lamda;
r=D/2;
Zo=sqrt(4*pi*10^(-7)/(10^(-9)/(36*pi)));
Yo=1/Zo;
```

ทำการกำหนดค่าปัจจัยต่าง ๆ ของสายอากาศป้อนกำลังคลื่น

กำหนดค่ามุมที่หมุนแกน z ออกไปโดยทำการหมุนรอบแกน y ดังรูปที่ 2

```
%zeta0=input('Orientation of feed about y axis in counter clockwise direction (degree) : ');
zeta0=pi;
```

กำหนดค่ามุมที่หมุนแกน x'' ออกไปโดยทำการหมุนรอบแกน z ดังรูปที่ 2

```
%phi0=input('Orientation of feed about z axis in counter clockwise direction (degree) : ');
phi0=0;
```

กำหนดชนิดของสายอากาศป้อนกำลังคลื่น ถ้า huygen=1 สายอากาศป้อนกำลังคลื่นเป็นแบบฮอยเกนถ้าเป็น 0 สายอากาศป้อนกำลังคลื่นเป็นแบบโคไซน์กำลังต่าง ๆ

```
huygen=1;
```

```
if huygen=0
```

กำหนดค่ากำลังของสายอากาศชนิดโคไซน์กำลังต่าง ๆ

```
%qe=input('Order of cosine pattern of feed in E-plane : ');
qe=1;        %y-polarized dipole qe=1
%qh=input('Order of cosine pattern of feed in H-plane : ');
qh=1;        %y-polarized dipole qh=0
end
```

กำหนดโพลาไรเซชันของสายอากาศป้อนกำลังคลื่น ถ้า phipol=0 เป็นสายอากาศที่มีโพลาไรเซชันในแนวแกน y ถ้า phipol=-pi/2 เป็นสายอากาศที่มีโพลาไรเซชันในแนวแกน x

```
%phipol=input('Polarization of feed : 0 -> y polarized, -pi/2 -> x polarized : ');
phipol=0;


% Input parameter to check case
```

ทำการรวมสนามไฟฟ้าจากการเลี้ยวเบนที่ขอบเข้าไปในระนาบหน้าจานเมื่อ Para1=1 ไม่รวมสนามไฟฟ้าจากการเลี้ยวเบนที่ขอบเข้าไปในระนาบหน้าจานเมื่อ Para1=0

```
Para1=1;
```

ทำการรวมสนามไฟฟ้าจากสายอากาศป้อนกำลังคลื่นเข้าไปในบริเวณย่านสนามใกล้เมื่อ Para2=1 ไม่รวมสนามไฟฟ้าจากสายอากาศป้อนกำลังคลื่นเข้าไปในบริเวณย่านสนามใกล้เมื่อ Para2=0

```
Para2=0;
```

กำหนดให้สายอากาศป้อนกำลังวางอยู่ที่จุดโฟกัส

```
  xf=0; yf=0; zf=0;
```

คำนวณจำนวนจุดบนระนาบหน้าจานเพื่อทำการหาสนามไฟฟ้าเพื่อนำไปหาสเปกตรัมคลื่นระนาบสำหรับคำนวณแบบรูปการแผ่พลังงานย่านสนามใกล้ โดยทำการแบ่งช่วงของจุดห่างกัน 1/8 ของความยาวคลื่นที่ใช้งาน

```
% Setup increasing step of observation point on aperture plane
s=ceil((r-(-r))/(Lamda/8)); t=ceil((r-(-r))/(Lamda/8));
if rem(s,2)~=0
  s=s+1;
end
if rem(t,2)~=0
  t=t+1;
end
StepX=(r-(-r))/s;  StepY=(r-(-r))/t;
```

หาสนามไฟฟ้าบนระนาบหน้าจาน

ทำการกำหนดค่าให้สนามไฟฟ้าบนหน้าจานเป็นศูนย์

```
if Para1==0
   Eax=zeros(s+1,t+1); Eay=zeros(s+1,t+1); Eaz=zeros(s+1,t+1);
else
   Eax=zeros(s+1,t+1); Eay=zeros(s+1,t+1); Eaz=zeros(s+1,t+1);
   Edx=zeros(s+1,t+1); Edy=zeros(s+1,t+1); Edz=zeros(s+1,t+1);
end
```

คำนวณหาสนามไฟฟ้าบนระนาบหน้าจานโดยกำหนดให้ขนาดของระนาบหน้าจานเท่ากับภาพฉายของจานสะท้อน

```
n=1;
m=1;
zp=zf;
for xp=-r:StepX:r
    if (xp==-r) | (xp==r)
      m=s/2+1; yp=0;
      Ego=go(xp,yp,zp);
      Eax(n,m)=Ego(1); Eay(n,m)=Ego(2); Eaz(n,m)=Ego(3);
      if Para1==1
        Ed=gtd(xp,yp,zp);
        Edx(n,m)=Ed(1); Edy(n,m)=Ed(2); Edz(n,m)=Ed(3);
      end
    else
      m=1;
      for yp=-r:StepY:r
```

ทำการตรวจสอบว่าจุดสังเกต xp, yp อยู่ในบริเวณภาพฉายหรือไม่ ถ้าอยู่ก็ทำการคำนวณหาสนามไฟฟ้าที่สะท้อนหรือสนามเลี้ยวเบนตาม Para1 ที่กำหนดไว้ ถ้าไม่อยู่สนามไฟฟ้าที่ตำแหน่งนั้นก็จะเป็นตามที่ได้กำหนดไว้ข้างต้น

```
          if (yp>=-sqrt(r^2-xp^2) & yp<=sqrt(r^2-xp^2))
            Ego=go(xp,yp,zp);
            Eax(n,m)=Ego(1); Eay(n,m)=Ego(2); Eaz(n,m)=Ego(3);
            if Para1==1
              Ed=gtd(xp,yp,zp);
              Edx(n,m)=Ed(1); Edy(n,m)=Ed(2); Edz(n,m)=Ed(3);
            end
          end
          m=m+1;
      end
```

```
    end
    fprintf('Aperture Field %5.2f%s\n',(n*100/((r-(-r))/StepX+1)),' %');
    n=n+1;
end
```

ทำการเก็บค่าสนามไฟฟ้าบนระนาบหน้าจานลงในไฟล์

```
if Para1==1
  save c:\user\joe\aperture\t00c11d1.mat Eax Eay Eaz Edx Edy Edz
  Eax=Eax+Edx; Eay=Eay+Edy; Eaz=Eaz+Edz;
  clear Edx Edy Edz
else
  Emax1=max(20*log10(abs(Eay(s/2+1,:))));
  Emax2=max(20*log10(abs(Eay(:,s/2+1))));
  Emax3=max(20*log10(abs(Eax(s/2+1,:))));
  Emax4=max(20*log10(abs(Eax(:,s/2+1))));
  save c:\user\joe\aperture\t00c11g1.mat Emax1 Emax2 Emax3 Emax4
end

x=-r:StepX:r;
y=-r:StepY:r;
```

ทำการหาแบบรูปการแผ่พลังงานโดยหาสเปกตรัมคลื่นระนาบของสนามบนระนาบหน้าจาน

```
k=1;
```

กำหนดค่าระนาบของจุดสังเกต ϕ ใด ๆ

```
%phi=input('Plane of observation in phi angle (degrees) : ');
phi=0;
```

กำหนดค่ามุม θ เริ่มต้น

```
%zetaStr=input('Start elevation angle (degree) : ');
zetaStr=0;
```

กำหนดค่ามุม θ สุดท้าย

```
%zetaStp=input('Stop elevation angle (degree) : ');
zetaStp=90;
```

กำหนดช่วงของการเพิ่มขึ้นของมุม θ

```
dzeta=0.5;
phiR=phi*pi/180;
for zeta=zetaStr:dzeta:zetaStp
    zetaR=zeta*pi/180;
```

```
    fx(k)=sim2(Eax,-r,r,s/2,t/2,zetaR,phiR,'-r','r');
    fy(k)=sim2(Eay,-r,r,s/2,t/2,zetaR,phiR,'-r','r');
    rp=200*(D^2)/Lamda;
    Efzeta(k)=j*beta*exp(-j*beta*rp)/(2*pi*rp)*(fx(k)*cos(phiR)+fy(k)*sin(phiR));
    Efphi(k)=j*beta*exp(-j*beta*rp)/(2*pi*rp)*(fy(k)*cos(phiR)-fx(k)*sin(phiR))*cos(zetaR);
    if Para2==1
      xp=rp*sin(zetaR)*cos(phiR);
      yp=rp*sin(zetaR)*sin(phiR);
      zp=rp*cos(zetaR);
      Exyzi=feed(xp,yp,zp);
      P=[sin(zetaR)*cos(phiR) sin(zetaR)*sin(phiR) cos(zetaR); ...
          cos(zetaR)*cos(phiR) cos(zetaR)*sin(phi*pi/180) -sin(zetaR); ...
          -sin(phiR) cos(phiR) 0];
      Erzpi=P*Exyzi;
      Efr(k)=Erzpi(1);
      Efzeta(k)=Efzeta(k)+Erzpi(2);
      Efphi(k)=Efphi(k)+Erzpi(3);
  end
  fprintf('Far Field %5.2f%s\n',(k*100/((zetaStp-zetaStr)/dzeta+1)),' %');
  k=k+1;
end
if phipol=0
  Eco=Efzeta*sin(phiR)+Efphi*cos(phiR);
  Ecx=Efzeta*cos(phiR)-Efphi*sin(phiR);
else
  Ecx=Efzeta*sin(phiR)+Efphi*cos(phiR);
  Eco=Efzeta*cos(phiR)-Efphi*sin(phiR);
end
zeta=zetaStr:dzeta:zetaStp;
ทำการเก็บค่าปัจจัยและสนามไฟฟ้าบนระนาบหน้าจานและสนามไฟฟ้าย่านสนามไกล
if Para1==1
  save c:\user\joe\matfile\s00c11d1.mat zeta0 phi0 qe qh phipol Eax Eay Eaz D Freq f x y s t StepX StepY
Efzeta Efphi Eco Ecx zeta phi timeuse
else
  save c:\user\joe\matfile\s00c11g1.mat zeta0 phi0 qe qh phipol Eax Eay Eaz D Freq f x y s t StepX StepY
Efzeta Efphi Eco Ecx zeta phi timeuse
```

```
end
```

จากโปรแกรมหลัก sympba.m ข้างต้น การคำนวณหาแบบรูปการแผ่พลังงานย่านสนามไกลดังสมการ (4.1) สเปกตรัมคลื่นระนาบในสมการ (4.2) หาได้จากฟังก์ชัน sim2.m ดังนี้

```
function area=sim2(E,a,b,n,m,xx,yy,eqn1,eqn2)
global r
h=(b-a)/(2*n);
Y0=0;
Y1=0;
Y2=0;
for xi=0:(2*n)
    x=a+xi*h;
    c=eval(eqn1); d=eval(eqn2);
    k=(d-c)/(2*m);
    X0=0;
    X1=0;
    X2=0;
    if (k~=0)
      X0=expr2(E(xi+1,1),x,c,xx,yy)+expr2(E(xi+1,2*m+1),x,d,xx,yy);
      for yj=1:(2*m-1)
          y=c+yj*k;
          Q=expr2(E(xi+1,yj+1),x,y,xx,yy);
          if rem(yj,2)==0
            X1=X1+Q;
          else
            X2=X2+Q;
          end
      end
    end
    L=(X0+2*X1+4*X2)*k/3;
    if (xi==0) | (xi==(2*n))
      Y0=Y0+L;
    elseif rem(xi,2)==0
      Y1=Y1+L;
    else
      Y2=Y2+L;
    end
```

```
end
area=(Y0+2*Y1+4*Y2)*h/3;
```

ในฟังก์ชัน sim2.m มีการเรียกใช้ฟังก์ชัน expr2.m เพื่อทำการคำนวณค่าของฟังก์ชันที่ถูกทำการอินทิเกรตที่จุดต่าง ๆ ที่แบ่งภายในขอบเขตของการอินทิเกรตดังนี้

```
function value=expr2(Etemp,x,y,xx,yy)
global beta
j=sqrt(-1);
value=Etemp*exp(j*beta*(sin(xx)*cos(yy)*x+sin(xx)*sin(yy)*y));
```

สรุป

การคำนวณแบบรูปการแผ่พลังงานของสายอากาศตามวิธีวิเคราะห์ที่นำเสนอในบทที่ 2 เริ่มต้นด้วยการคำนวณสนามที่ระนาบหน้าจานสะท้อนที่มาจากตัวป้อนโดยไม่คิดผลจากการเลี้ยวเบน แล้วจึงนำการคิดผลดังกล่าวเข้ามารวมด้วย จากนั้นจึงไปสู่การคำนวณแบบรูปการแผ่พลังงานที่ย่านสนามไกล ซึ่งทำโดยคำนวณสเปกตรัมคลื่นระนาบของสายอากาศ

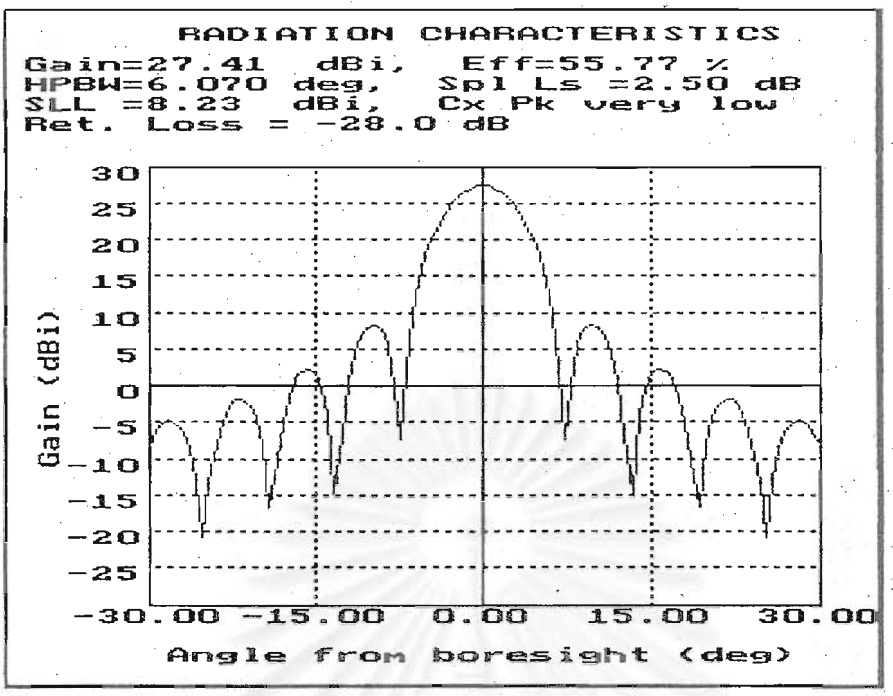
4 การตรวจสอบความสามารถของโปรแกรมที่พัฒนาขึ้น

ความสามารถของโปรแกรมที่พัฒนาขึ้นจำเป็นต้องได้รับการตรวจสอบ ทั้งนี้เพื่อยืนยันว่าโปรแกรมสามารถทำงานได้ตามข้อกำหนด ผลการคำนวณที่ได้มีความน่าเชื่อถือในระดับสูง ปกติแล้วการตรวจสอบความสามารถของโปรแกรมสามารถทำได้ทั้งด้วยการเปรียบเทียบผลกับโปรแกรมมาตรฐานที่มีอยู่แล้ว หรือเทียบผลกับผลการคำนวณที่ได้รับการตีพิมพ์ในวารสารวิชาการที่มีคุณภาพสูงเป็นที่ยอมรับของวงวิชาการ อีกทางออกหนึ่ง คือ การเปรียบเทียบผลการคำนวณกับการวัดจริง วิธีหลังนี้มีค่าใช้จ่ายในการดำเนินการสูงจึงไม่ค่อยเหมาะสมเท่าใดนัก สำหรับโครงการนี้ได้เลือกที่จะตรวจสอบความสามารถของโปรแกรมที่พัฒนาขึ้นโดยการเปรียบเทียบผลกับโปรแกรมมาตรฐานที่มีอยู่แล้ว และเทียบผลกับผลการคำนวณที่ได้รับการตีพิมพ์ในวารสารวิชาการที่มีคุณภาพสูงเป็นที่ยอมรับของวงวิชาการ

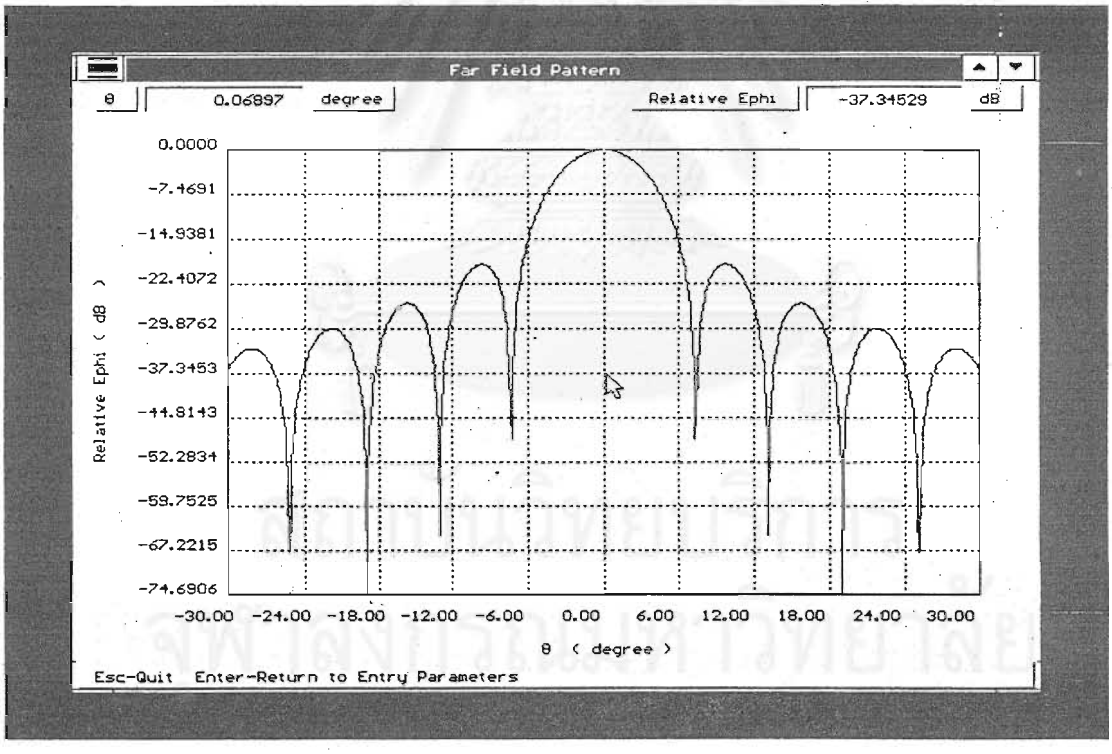
ผลการเปรียบเทียบความสามารถของโปรแกรมที่พัฒนาขึ้นกับโปรแกรมที่มีอยู่แล้ว

ผู้วิจัยได้ดำเนินการเปรียบเทียบผลการคำนวณที่ได้จากโปรแกรมที่พัฒนาขึ้นกับโปรแกรม RASCAL ซึ่งพัฒนาโดยคณะนักวิจัยของมหาวิทยาลัยแคลิฟอร์เนียใต้ RASCAL เป็นโปรแกรมที่ใช้สำหรับการวิเคราะห์และสังเคราะห์สายอากาศจานสะท้อนด้วยกรรมวิธีทัศนศาสตร์กายภาพและทัศนศาสตร์เรขาคณิต ความสนใจเกี่ยวกับลักษณะสมบัติของแบบรูปการแผ่พลังงานจำกัดอยู่เฉพาะบริเวณพูประธานและพูข้างที่อยู่ใกล้เคียง ดังนั้น RASCAL จึงไม่สนใจการวิเคราะห์ปัญหาการเลี้ยวเบน การเปรียบเทียบผลการคำนวณจึงทำเฉพาะกรณีไม่คิดผลกระทบจากการเลี้ยวเบน ผู้วิจัยได้ดำเนินการเปรียบเทียบแบบรูปการแผ่พลังงานของสายอากาศชนิดจานสะท้อนคลื่นที่มีเส้นผ่านศูนย์กลาง 1 เมตร ความถี่ 3 GHz F/D ratio = 0.42 โดยใช้โปรแกรม RASCAL และโปรแกรมที่พัฒนาขึ้นเอง โดยมีผลการเปรียบเทียบแสดงอยู่ในรูปที่ 3

จากรูปที่ 3 พบว่าแบบรูปการแผ่พลังงานที่คำนวณด้วยโปรแกรมที่พัฒนาขึ้นกับโปรแกรม RASCAL มีความใกล้เคียงกันอย่างมาก ระดับพูข้างอันดับแรกอยู่ที่ประมาณ 20 dB ต่ำกว่าพูประธานทั้งกรณีของผลที่ได้จาก RASCAL และที่ได้จากโปรแกรมที่พัฒนาขึ้น โดยต้องทำการเปรียบเทียบระดับของพูต่างๆด้วยขนาดเชิงสัมพัทธ์ ดังนั้นสามารถกล่าวได้ว่าโปรแกรมที่พัฒนาขึ้นมีขีดความสามารถเทียบได้กับโปรแกรม RASCAL



(ก)

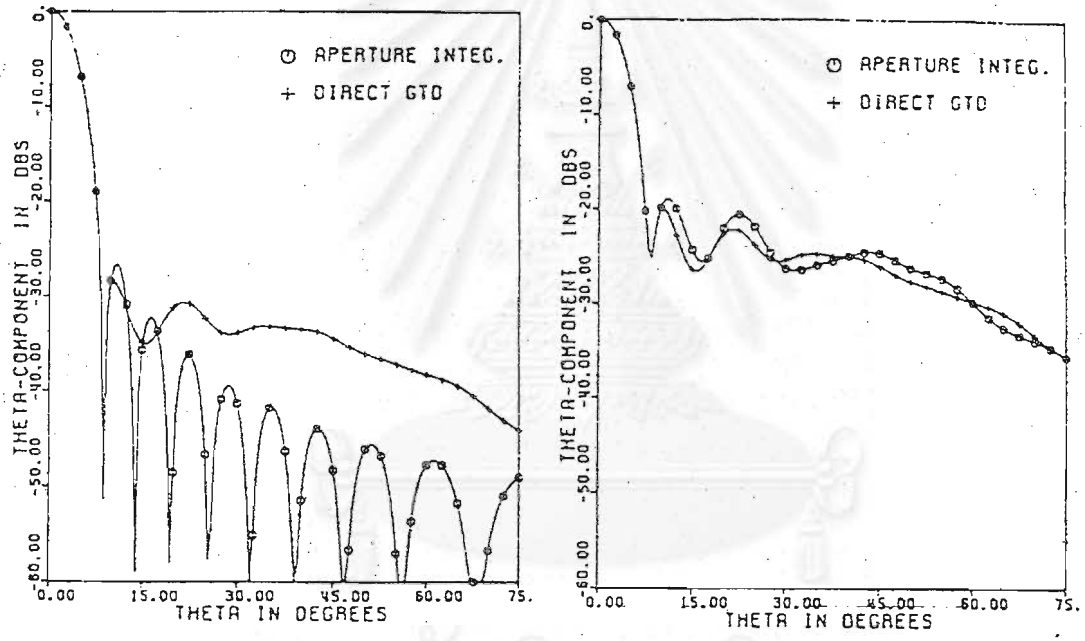


(ข)

รูปที่ 3 เปรียบเทียบแบบรูปการแผ่พลังงานของสายอากาศชนิดจานสะท้อนคลื่นที่มีเส้นผ่านศูนย์กลาง 1 เมตร ความถี่ 3 GHz F/D ratio = 0.42 โดยใช้โปรแกรม RASCAL (ก) และโปรแกรมที่พัฒนาขึ้นเอง (ข)

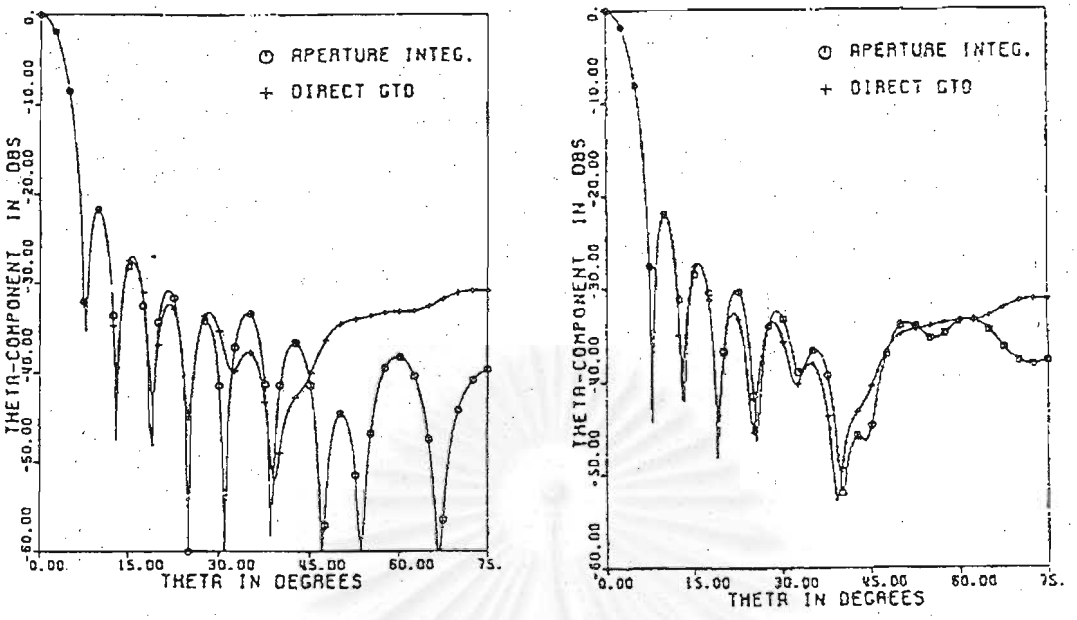
การตรวจสอบความแม่นยำในการคำนวณด้วยโปรแกรมที่พัฒนาขึ้น

จากการที่ไม่สามารถตรวจสอบความแม่นยำในการคำนวณกรณีคิดผลกระทบจากการเลี้ยวเบนกับ RASCAL ได้ ผู้วิจัยจึงได้ดำเนินการสืบค้นผลการคำนวณสำหรับกรณีปัญหาตัวอย่างที่ได้รับการตีพิมพ์ในวารสารวิชาการที่เชื่อถือได้เพื่อเป็นผลการคำนวณอ้างอิงในการเปรียบเทียบกับผลที่ได้จากโปรแกรมที่พัฒนาขึ้น เพื่อให้เห็นว่าขั้นตอนการวิเคราะห์ที่ใช้ดังรูป 1 มีความถูกต้องในเกณฑ์ที่เชื่อถือได้ ในที่นี้จึงทำการเปรียบเทียบผลการวิเคราะห์ที่ได้กับงานวิจัยของ M. S. A. Sanad และ L. Shafi (1986) ซึ่งได้ทำการวิเคราะห์สายอากาศชนิดจานสะท้อนเดี่ยวรูปพาราโบลิกแบบสมมาตรขนาด 10 เท่าของความยาวคลื่น ค่าอัตราส่วนของระยะโฟกัสต่อขนาดสายอากาศเท่ากับ 0.37 และใช้สายอากาศชนิดฮอยเกนส์กับสายอากาศชนิดไดโพลขนาดสั้นมากเป็นสายอากาศป้อนกำลังคลื่น โดยใช้ทฤษฎีการเลี้ยวเบนเชิงเรขาคณิตมาหาสนามไฟฟ้าย่านสนามไกลโดยตรงเปรียบเทียบกับการขยายแนวความคิดของการอินทิเกรตสนามบนระนาบ (extended aperture integration method) ซึ่งได้ทำการรวมสนามเนื่องจากการเลี้ยวเบนและสนามจากสายอากาศป้อนกำลังคลื่นเข้าไปบนระนาบหน้าจานแล้วทำการอินทิเกรตสนามบนระนาบเพื่อหาแบบรูปการแผ่พลังงานย่านสนามไกลได้ผลเป็นดังรูปที่ 4 และ 5



ก) เฉพาะสนามเนื่องจากการสะท้อนที่รวมเข้าไปบนระนาบ ข) รวมทั้งสนามเนื่องจากการเลี้ยวเบนและจากสายอากาศป้อนก

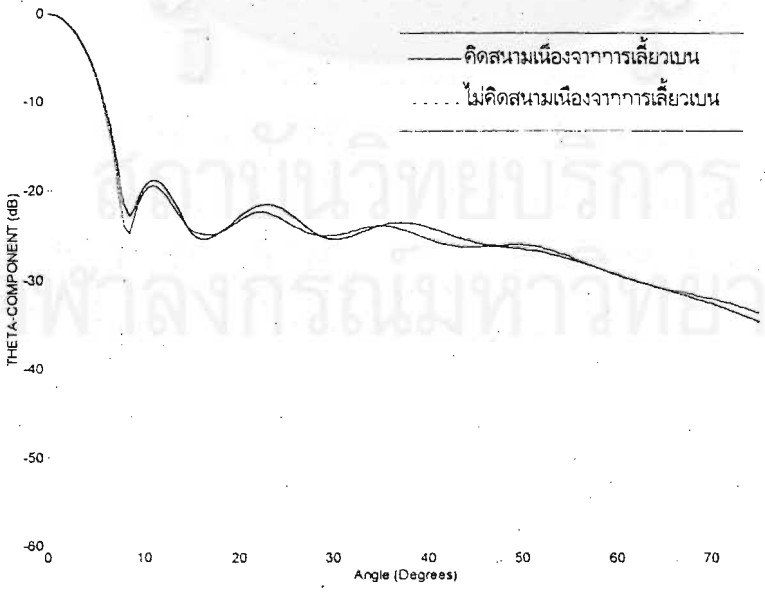
รูป 4 แบบรูปการแผ่พลังงานย่านสนามไกลของสายอากาศชนิดจานสะท้อนเดี่ยวรูปพาราโบลิกแบบสมมาตร (สายอากาศชนิดไดโพลขนาดสั้นมากเป็นสายอากาศป้อนกำลังคลื่น) ที่มา : M. S. A. Sanad และ L. Shafi (1986)



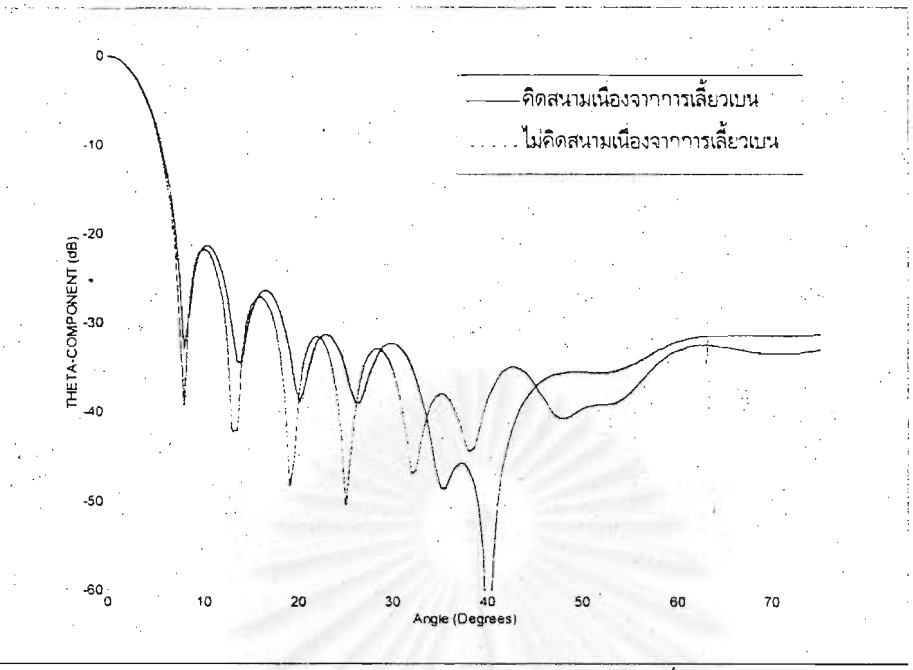
ก) เฉพาะสนามเนื่องจากการสะท้อนที่รวมเข้าไปบนระนาบ ข) รวมทั้งสนามเนื่องจากการเลี้ยวเบนและจากสายอากาศป้อนกำลังคลื่น

รูปที่ 3 แบบรูปการแผ่พลังงานย่านสนามไกลของสายอากาศชนิดจานสะท้อนเดี่ยวรูปพาราโบลิกแบบสมมาตร (สายอากาศชนิดฮอยเกนเป็นสายอากาศป้อนกำลังคลื่น) ที่มา : M. S. A. Sanad และ L. Shafi (1986)

เมื่อใช้ขั้นตอนการวิเคราะห์ดังแสดงในบทที่ 3 มาวิเคราะห์ระบบสายอากาศชนิดจานสะท้อนเดี่ยวรูปพาราโบลิกแบบสมมาตรข้างต้น โดยทำการรวมสนามจากสายอากาศป้อนกำลังคลื่นเข้าไปโดยตรงที่ย่านสนามไกลได้ผลดังรูปที่ 4 และ 5



รูปที่ 4 แบบรูปการแผ่พลังงานย่านสนามไกลของสายอากาศชนิดจานสะท้อนเดี่ยวรูปพาราโบลิกแบบสมมาตร (สายอากาศชนิดไดโพลขนาดสั้นมากเป็นสายอากาศป้อนกำลังคลื่น) ตามขั้นตอนการวิเคราะห์ในบทที่ 3



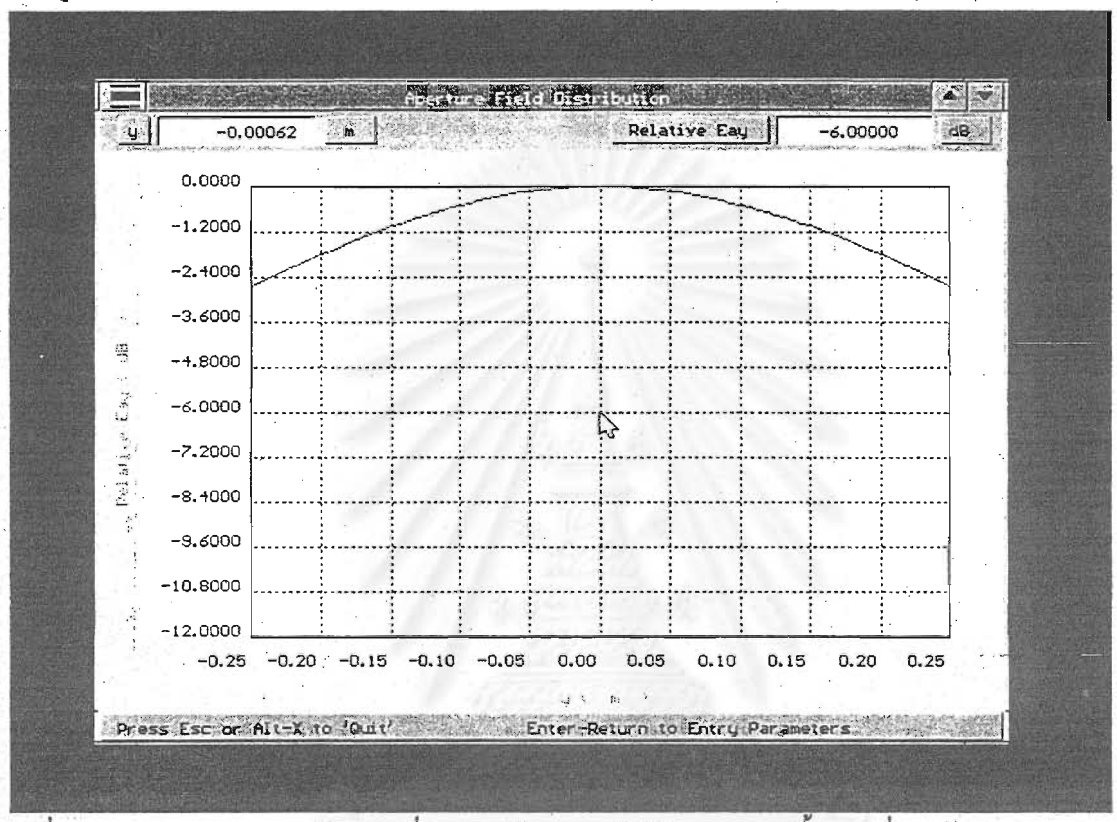
รูปที่ 7 แบบรูปการแผ่พลังงานย่านสนามไกลของสายอากาศชนิดจานสะท้อนเดี่ยวรูปพาราโบลิกแบบสมมาตร (สายอากาศชนิดฮอยเกนส์เป็นสายอากาศป้อนกำลังคลื่น) ตามขั้นตอนการวิเคราะห์ดังรูปที่ 1

เมื่อเปรียบเทียบผลการวิเคราะห์ในงานวิจัยของ M. S. A. Sanad และ L. Shafi (1986) ดังรูปที่ 4 และ 5 กับผลการวิเคราะห์ตามขั้นตอนในบทที่ 3 เมื่อพิจารณารูปที่ 6 และ 7 พบว่า ผลที่ได้จากการวิเคราะห์ตามขั้นตอนการวิเคราะห์ในบทที่ 3 มีแนวโน้มใกล้เคียงกันกับการวิเคราะห์โดยใช้ทฤษฎีการเลี้ยวเบนเชิงเรขาคณิตโดยตรง ดังนั้นแผนภาพขั้นตอนการวิเคราะห์ในบทที่ 3 ให้ผลการวิเคราะห์อยู่ในเกณฑ์ที่เชื่อถือได้

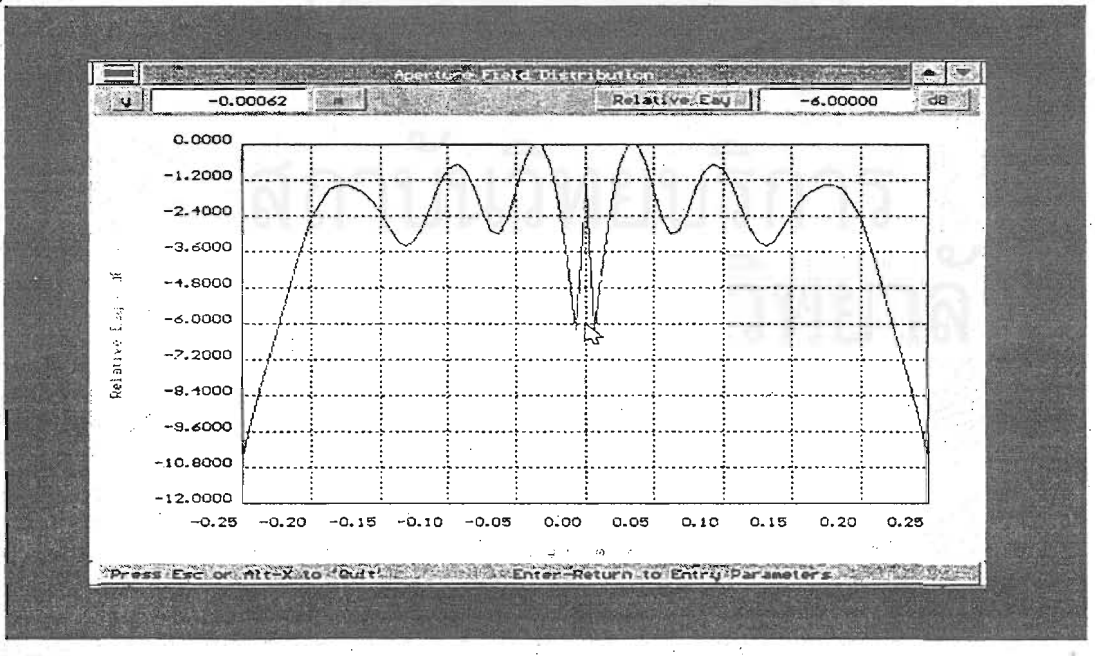
5 ตัวอย่างการใช้งานโปรแกรม

ตัวอย่างการใช้งานโปรแกรม

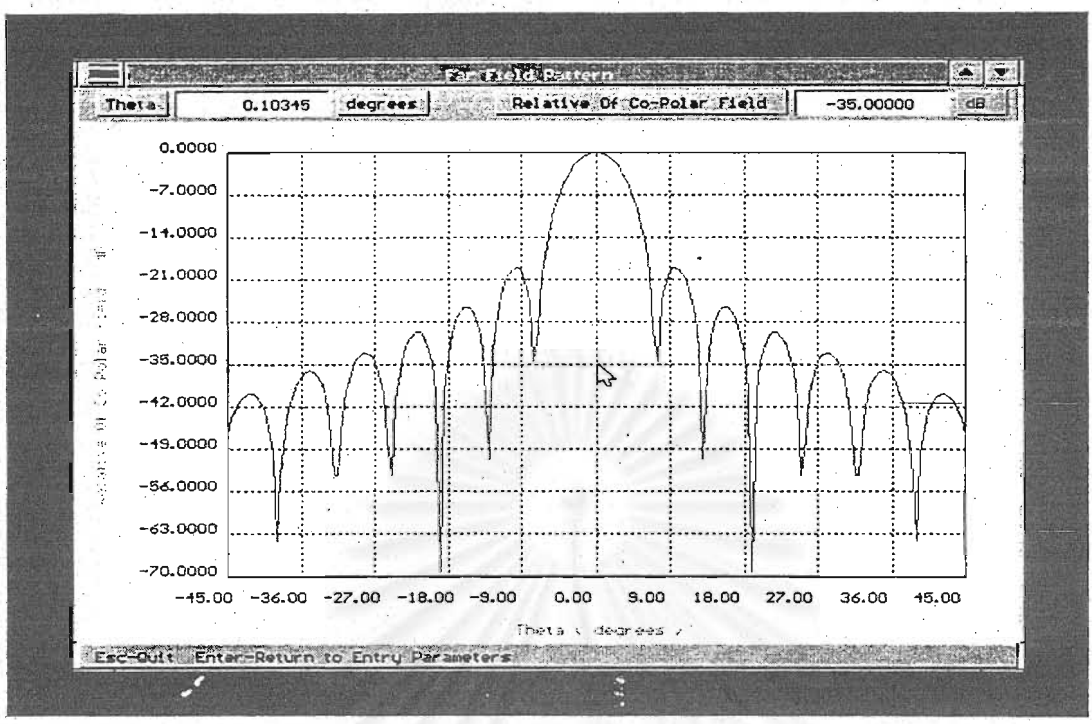
ในส่วนนี้จะนำเสนอตัวอย่างปัญหาที่ทำการคำนวณโดยโปรแกรมที่พัฒนาขึ้น สายอากาศที่เป็นกรณีตัวอย่างมีขนาดเส้นผ่านศูนย์กลาง 0.5 เมตร ความถี่ปฏิบัติการ 6 จิกะเฮิรทซ์ อัตราส่วนทางยาวโฟกัสต่อเส้นผ่านศูนย์กลาง 0.42 ใช้สายอากาศป้อนชนิดแหล่งกำเนิดแบบจุด ผลการคำนวณแสดงในรูปที่ 1- 4



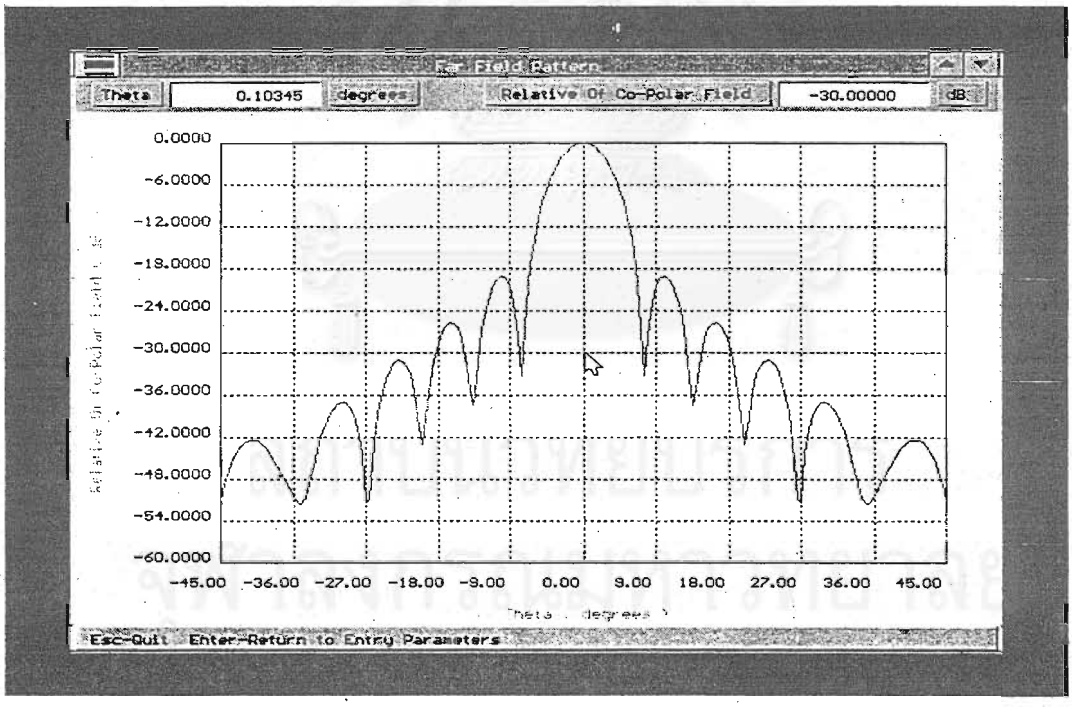
รูปที่ 1 ภาพการกระจายความเข้มสนามที่ระนาบหน้าจานกรณีไม่คิดผลจากการเลี้ยวเบนที่ขอบ



รูปที่ 2 ภาพการกระจายความเข้มสนามที่ระนาบหน้าจานกรณีคิดผลจากการเลี้ยวเบนที่ขอบ



รูปที่ 3 แบบรูปการแผ่พลังงานย่านสนามไกลกรณีไม่คิดผลจากการเลี้ยวเบนที่ขอบ



รูปที่ 4 แบบรูปการแผ่พลังงานย่านสนามไกลกรณีคิดผลจากการเลี้ยวเบนที่ขอบ

ผลการคำนวณในรูปที่ 1- 4 แสดงให้เห็นความแตกต่างระหว่างกรณีคิดและไม่คิดผลกระทบจากการเลี้ยวเบนที่ขอบจานสะท้อน ผลดังกล่าวมีระดับนัยสำคัญสูงเมื่อพิจารณาภาพการกระจายความเข้มของสนามที่ระนาบหน้าจาน แต่กลับไม่น่ากังวลนักเมื่อพิจารณาแบบรูปการแผ่พลังงานย่านสนามไกล

6 สรุปและข้อเสนอแนะ

โครงการได้พัฒนาโปรแกรมคอมพิวเตอร์สำหรับการวิเคราะห์สายอากาศจานสะท้อนชนิดจานเดี่ยวที่มีการป้อนสัญญาณอย่างสมมาตร โปรแกรมสามารถคำนวณภาพการกระจายความเข้มของสนามที่ระนาบหน้าจานและแบบรูปการแผ่พลังงานที่ย่านสนามไกลได้ นอกจากนี้ยังสามารถคำนวณผลกระทบอันเนื่องมาจากการเลี้ยวเบนที่ขอบจานสะท้อนได้อีกด้วย การคำนวณการเลี้ยวเบนได้ใช้ทฤษฎีการเลี้ยวเบนเชิงเรขาคณิตซึ่งเป็นส่วนขยายของทัศนศาสตร์เรขาคณิต สำหรับชนิดของสายอากาศป้อนที่สามารถวิเคราะห์ได้ด้วยโปรแกรมนี้ได้แก่ แหล่งกำเนิดแบบจุด แหล่งกำเนิดฮอยเกน สายอากาศขั้วคู่ ตัวป้อนที่มีแบบรูปการแผ่พลังงานชนิดฟังก์ชันโคไซน์ยกกำลัง

โปรแกรมนี้มีจุดอ่อนที่เห็นได้ชัด คือ ความเร็วในการประมวลผล ทั้งนี้อาจแก้ไขได้ในโอกาสต่อไปด้วยการปรับปรุงต่างๆ ดังนี้

ก. คิดผลกระทบจากการเลี้ยวเบนไปรวมที่ย่านสนามไกลโดยตรงเลยแทนที่จะนำมารวมกับสนามตามวิธีทัศนศาสตร์เรขาคณิตที่บริเวณระนาบหน้าจาน

ข. ในการคำนวณแบบรูปการแผ่พลังงานย่านสนามไกลด้วยการหาสเปกตรัมคลื่นระนาบควรใช้การแปลงฟูริเยร์อย่างเร็วแทนวิธีการหาผลรวมตามสูตรการอินทิเกรตเชิงตัวเลขแบบมาตรฐานที่ใช้อยู่

นอกเหนือไปจากนี้ควรพัฒนาให้โปรแกรมสามารถรับข้อมูลเกี่ยวตัวป้อนซึ่งได้จากผลการวัดตัวป้อนนั้นๆโดยตรงได้ด้วย และควรพัฒนาต่อไปให้สามารถวิเคราะห์ผลกระทบเนื่องจากปัญหาอื่นๆอีก เช่น การผิดรูปของผิวจาน การเลื่อนเชิงตำแหน่งของตัวป้อน และการบดบังหน้าจาน

เอกสารอ้างอิง

1.Clarricoats, P.J.B., "Some recent advances in microwave reflector antennas," Proc. IEE, Vol.126, No. 1, January, 1979, pp. 9-25

2.Rusch, W.V.T., "The Current State of the Reflector Antenna Art," IEEE Trans. on Antennas and Propagation, Vol. AP-32, No.4, April, 1984, pp. 313-329

3.Sletten, C.J., Reflector and Lens, Chapter 4, pp. 187-213, Artech House, USA

4.Keller, J.B., "Geometrical Theory of Diffraction," Journal of the Optical Society of America, Vol. 52, No. 2, February, 1962, pp. 116-130

5.Rudge, A.W., et al., The Handbook of Antenna Design, Peter Peregrinus Ltd., IEE London, UK, 1986

6.Wood, P.J., Reflector antenna analysis, Peter Peregrinus Ltd., IEE London, UK, 1980.

7.Rahmat-Samii, Y.,"Satellite Antenna Design and Modelling," 19th QMW ANTENNA SYMPOSIUM, 25-26 March.1993

8.ศุภเชษฐ์ เพิ่มพูนวัฒนาสุข, ฉัตรชัย ไวยาพัฒนกร, สมศักดิ์ ปัญญาแก้ว, "การศึกษาผลกระทบจากปรากฏการณ์เลี้ยวเบนที่ขอบของจานสะท้อนคลื่นที่มีต่อแบบรูปการแผ่พลังงานย่านสนามไกลของสายอากาศชนิดจานสะท้อนเดี่ยวสมมาตรโดยใช้ทฤษฎีการเลี้ยวเบนเชิงเรขาคณิต,"การประชุมวิชาการวิศวกรรมไฟฟ้าครั้งที่ 18, พฤศจิกายน 2538, หน้า 322-327

9. Rusch, W.V.T., Sφrensen, O., "The Geometrical of Diffraction for Axially Symmetric Reflectors," IEEE Trans on Antennas and Propagation, Vol. AP-23, No. 5, May, 1975, pp. 414-418

ภาคผนวก

คู่มือการใช้งานฉบับร่าง

โปรแกรมคอมพิวเตอร์วิเคราะห์ระบบสายอากาศชนิดจานสะท้อนคลื่น

(Reflector Antenna System Analysis)

โปรแกรมคอมพิวเตอร์วิเคราะห์ระบบสายอากาศชนิดจานสะท้อนคลื่น เป็นโปรแกรมที่ใช้ในการคำนวณและแสดงค่าของสนามไฟฟ้าที่ระนาบหน้าจาน และที่ย่านสนามไกล โปรแกรมคอมพิวเตอร์วิเคราะห์ระบบสายอากาศชนิดจานสะท้อนคลื่นเป็นผลงานของกลุ่มวิทยาการแม่เหล็กไฟฟ้าประยุกต์ ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ซึ่งมีวิธีการใช้ที่ไม่ยากนัก ดังต่อไปนี้

การเข้าสู่โปรแกรม

โปรแกรมคอมพิวเตอร์วิเคราะห์ระบบสายอากาศชนิดจานสะท้อนคลื่นนั้นสามารถที่จะเรียกใช้งานได้บน DOS โดยพิมพ์คำสั่ง

MAINMENU <Enter>

หลังจากนั้นบนหน้าจอคอมพิวเตอร์จะปรากฏรายการสำหรับการคำนวณดังรูป

Reflector Antenna System Analysis

<<<<<<<<MAIN MENU>>>>>>>>

<*>1>Calculate Aperture Field of Symmetric Parabolic Reflector

<*>2>Plot Aperture Field Distribution

<*>3>Calculate Far Field of Symmetric Parabolic Reflector

<*>4>Plot Far Field Radiation Pattern

<*> ESC ..... Exit to DOS/Windows

Department of Electrical Engineering

Faculty of Engineering

Chulalongkorn University

รูปที่ ผ1 หน้าจอหลักแสดงรายการสำหรับการคำนวณในการวิเคราะห์สายอากาศชนิดจานสะท้อนเดี่ยวสมมาตร

ผู้ใช้สามารถที่จะใช้โปรแกรมคอมพิวเตอร์วิเคราะห์ระบบสายอากาศชนิดจานสะท้อนคลื่น ดังกล่าวนี้เพื่อ

1.คำนวณสนามไฟฟ้าที่ระนาบหน้าจานสะท้อนคลื่น (Calculate Aperture Field of Symetrical Parabolic Antenna) โดยการกดปุ่มบนแป้นพิมพ์ หมายเลข 1 หรืออักษร "C"

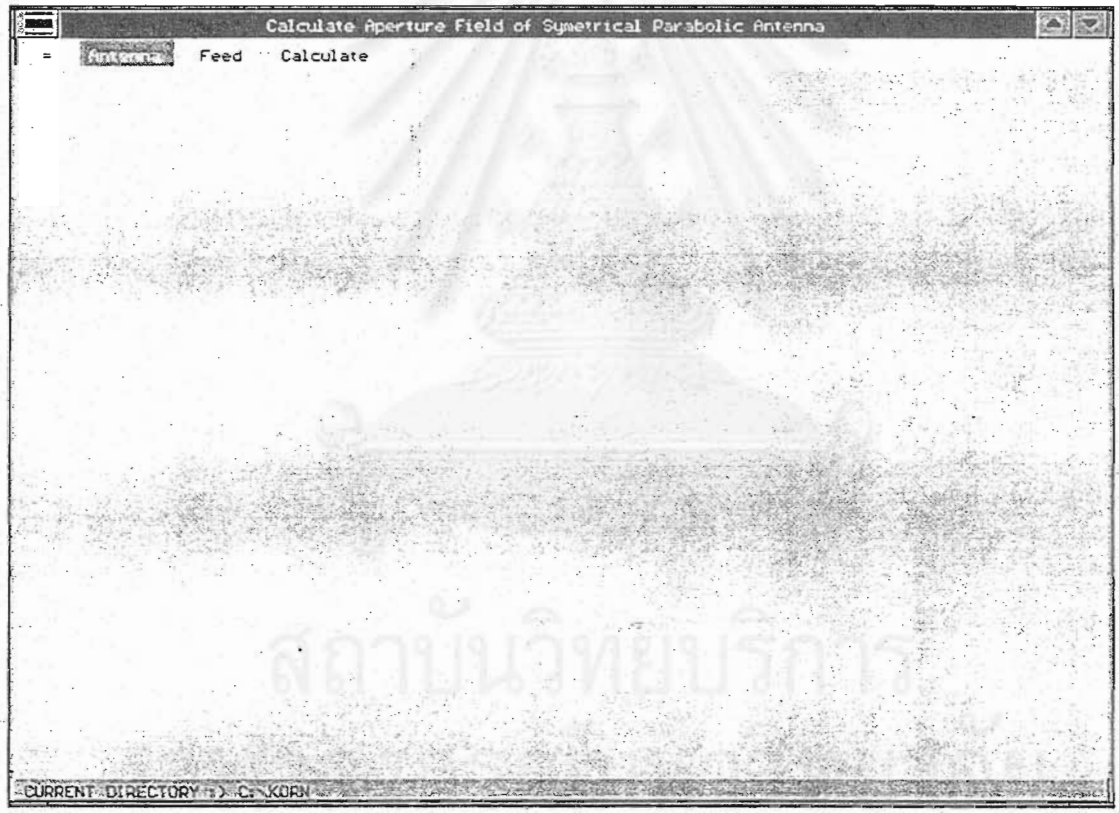
2.แสดงภาพการกระจายความเข้มสนามไฟฟ้าที่ระนาบหน้าจานสะท้อนคลื่น(Plot Aperture Field of Symetrical Parabolic Antenna) โดยการกดปุ่มบนแป้นพิมพ์ หมายเลข 2 หรืออักษร "P"

3.คำนวณสนามไฟฟ้าย่านสนามไกล (Calculate Far Field of Symetrical Parabolic Antenna) โดยการกดปุ่มบนแป้นพิมพ์ หมายเลข 3 หรืออักษร "L"

4.แสดงภาพแบบรูปการแผ่พลังงานย่านสนามไกล (Plot Far Field Radiation Pattern of Symetrical Parabolic Antenna) โดยการกดปุ่มบน keyboard หมายเลข 4 หรืออักษร "O"

การคำนวณสนามไฟฟ้าที่ระนาบหน้าจานสะท้อนคลื่น

เมื่อเลือกเมนูข้อ 1. Calculate Aperture Field of Symetrical Parabolic Reflector หน้าจอที่ปรากฏจะเป็นดังรูปที่ ผ2



รูปที่ ผ2 หน้าจอที่ปรากฏเมื่อเลือกเมนูข้อ 1

แล้วให้สังเกตที่ menu bar หากเลื่อน cursor ไปที่ช่องซ้ายสุด และกดปุ่ม Enter จะปรากฏ menu ย่อยขึ้นมา ได้แก่

- Change Dir สำหรับเปลี่ยน Directory ที่จะใช้เก็บข้อมูลการคำนวณ เป็น Directory อื่นที่ไม่ใช่ Directory ปัจจุบัน

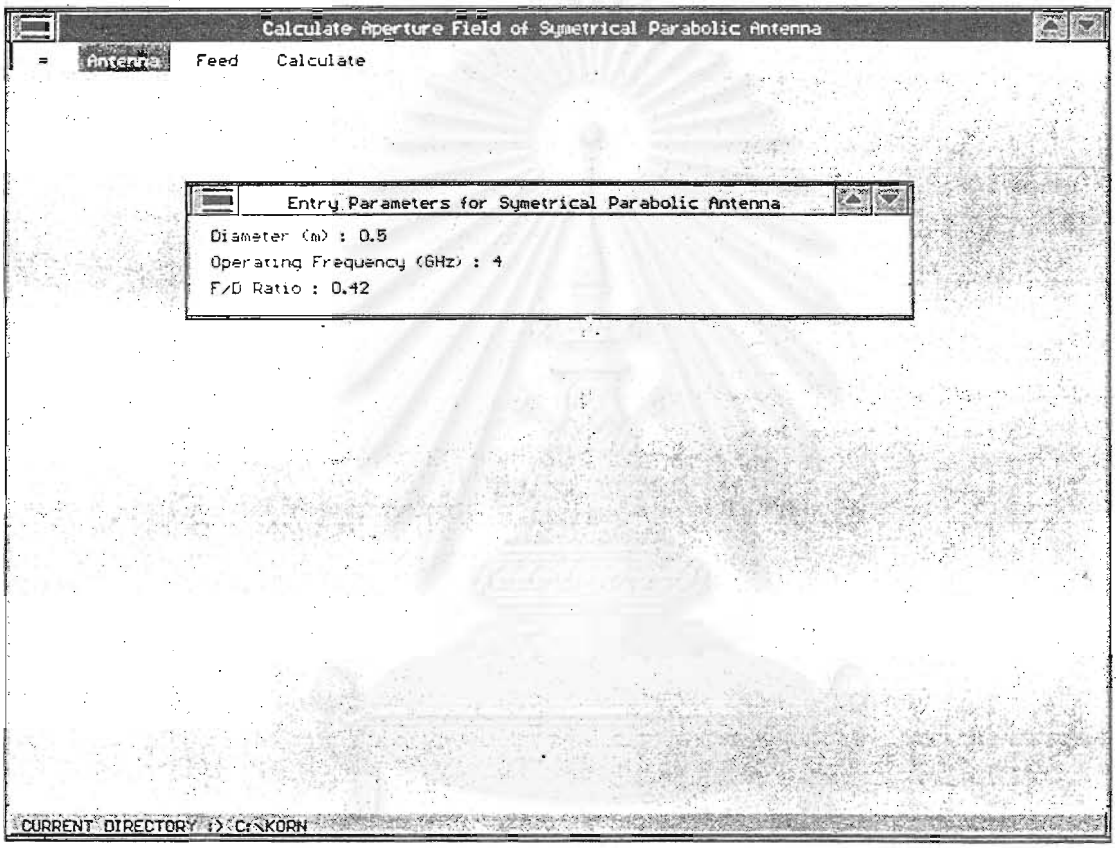
- Exit สำหรับออกจากการคำนวณสนามไฟฟ้าหน้าจานสะท้อนคลื่น กลับสู่ mainmenu

ที่ menu bar ANTENNA เราสามารถที่จะใส่ค่า parameter ของจานสะท้อนคลื่น โดยเลือกที่ menu ย่อย PARAMETER ซึ่งจะปรากฏวินโดว์ให้ใส่ค่า parameter ดังต่อไปนี้

Diameter ค่าจะต้องอยู่ในช่วง 0.1 - 2.0 เมตร

Operating Frequency ค่าจะต้องอยู่ช่วง 1 - 20 GHz

F/D Ratio รูปที่ ผ3 แสดงหน้าจอดังกล่าว



รูปที่ ผ3 หน้าจอแสดงการป้อนข้อมูลเบื้องต้นสำหรับการวิเคราะห์สายอากาศ

ใน menu bar ANTENNA นี้ผู้ใช้สามารถตรวจสอบค่า parameter ที่ได้ใส่เข้าไป โดยเลือก menu ย่อย CONFIGURATION ซึ่งเมื่อเลือกแล้วจะปรากฏหน้าต่างแสดงค่าต่าง ๆ ต่อไปนี้

ขนาดเส้นผ่านศูนย์กลางของจานสะท้อนคลื่น (Diameter of Reflector)

ความถี่ปฏิบัติการ(Operating Frequency)

อัตราส่วนความยาวเส้นผ่านศูนย์กลางต่อความยาวคลื่น (D/Lamda)

อัตราส่วนความยาวโฟกัสต่อความยาวเส้นผ่านศูนย์กลาง (F/D Ratio)

ค่าความยาวโฟกัส(Focus)

ชนิดของตัวป้อน (Feed Type)

นอกจากนี้ใน menu ANTENNA นี้ยังมี menu ย่อย MEMORY INFORMATION ซึ่งจะแสดงให้ผู้ใช้ทราบข้อมูลเกี่ยวกับหน่วยความจำ

ในการพิจารณาเรื่องแบบรูปการแผ่พลังงานนั้น ชนิดของตัวป้อน (Feed) นับว่ามีส่วนสำคัญที่จะทำให้แบบรูปการแผ่พลังงานที่ได้มีความแตกต่างกันออกไป ผู้ใช้สามารถที่จะกำหนดชนิดของตัวป้อน โดยเลือก menu FEED ที่อยู่บน menu bar ซึ่งโปรแกรมจะมี menu ย่อย ให้เลือกดังนี้

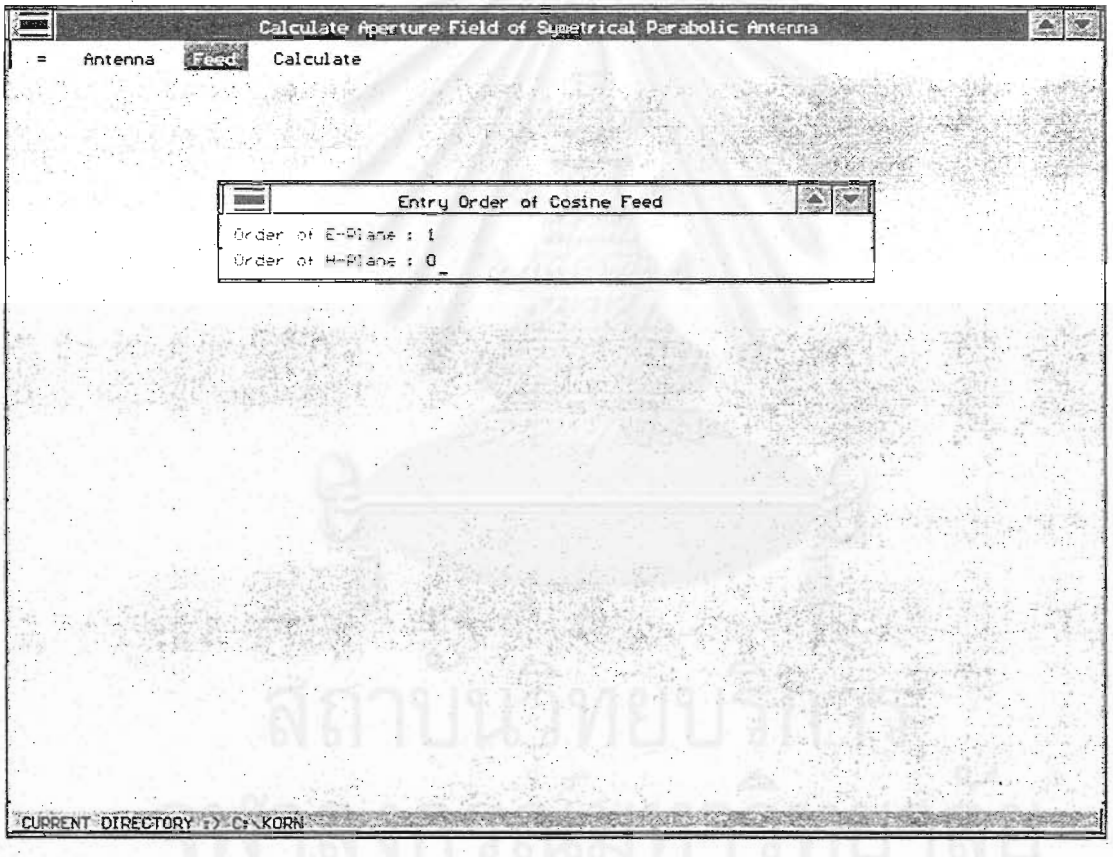
Point Source

Short Dipole

Huygen Source

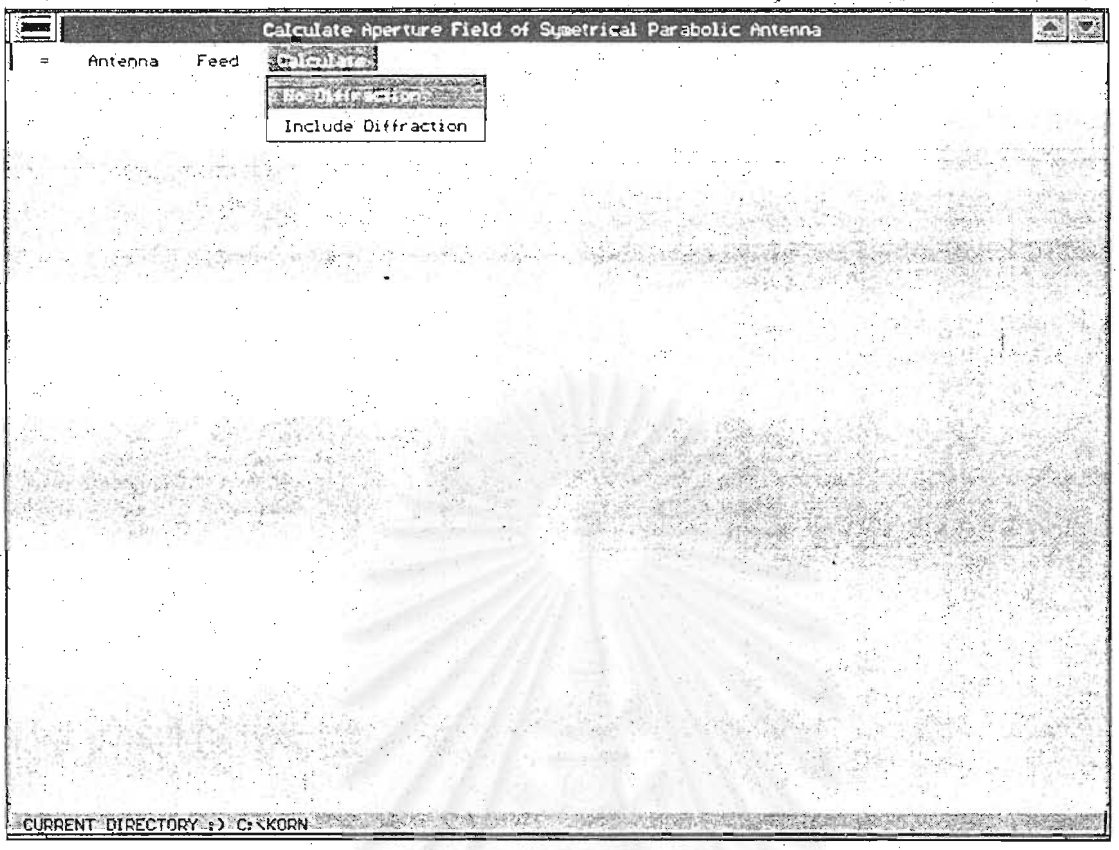
Cosine Order n

รูปที่ ผ4 แสดงตัวอย่างกรณีเลือกตัวป้อนชนิดฟังก์ชันโคไซน์ยกกำลัง



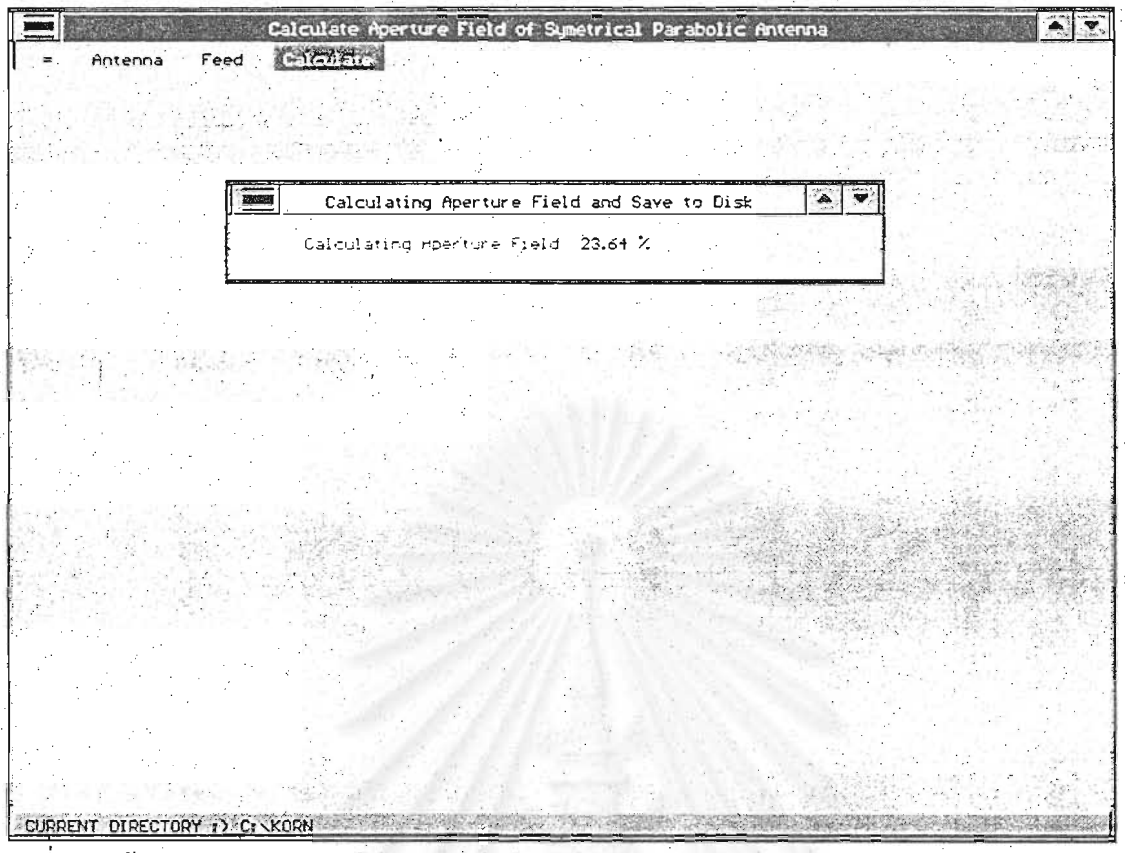
รูปที่ ผ4 หน้าจอสำหรับการเลือกสายอากาศป้อนโดยกำหนดลักษณะแบบรูปการแผ่พลังงาน

เมื่อค่าต่าง ๆที่มีความจำเป็นต่อการคำนวณได้รับการป้อนเข้าสู่โปรแกรมเรียบร้อยแล้ว ผู้ใช้ก็สามารถจะสั่งให้โปรแกรมเริ่มทำการคำนวณโดยการเลือก CALCULATE ที่ menu bar ซึ่งเมื่อเลือกแล้วจะปรากฏ menu ย่อย ให้ผู้ใช้กำหนดว่าต้องการให้ผลการคำนวณรวมผลของการเลี้ยวเบน(Include Diffraction) หรือไม่รวมผลของการเลี้ยวเบน(No Diffraction) ดังแสดงในรูปที่ ผ5



รูปที่ ผ5 หน้าจอสำหรับสั่งให้ทำการคำนวณ

จากนั้นโปรแกรมจะให้ผู้ใช้ใส่ชื่อแฟ้มข้อมูลที่จะใช้เก็บข้อมูลผลการคำนวณ เมื่อโปรแกรมเริ่มการคำนวณ สังเกตตัวเลขแสดงการคำนวณ หากตัวเลขเพิ่มขึ้นจนกระทั่งถึง 100 % แสดงว่าการคำนวณเสร็จสมบูรณ์และสามารถกลับสู่ Mainmenu ได้ รูปที่ ผ6 เป็นหน้าจอแสดงสถานะการคำนวณ



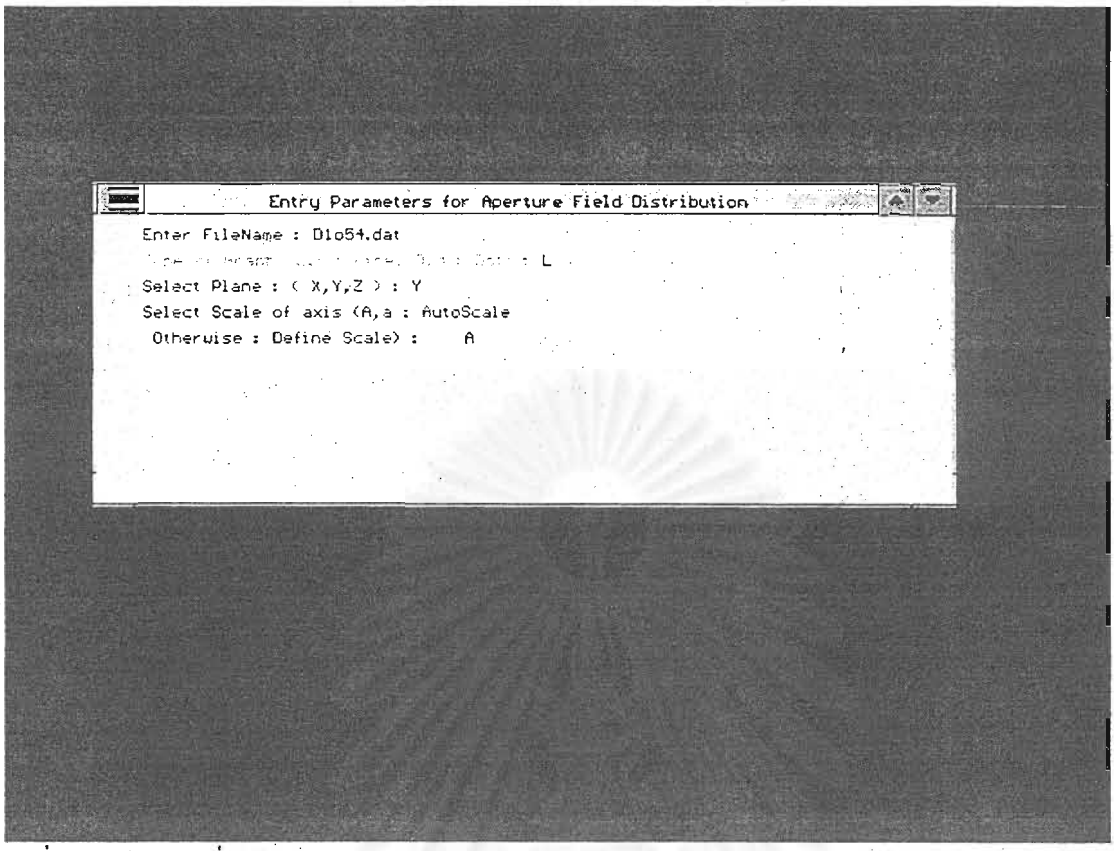
รูปที่ ผ6 หน้าจอแสดงสถานะขณะโปรแกรมกำลังทำการคำนวณ

การแสดงภาพการกระจายความเข้มสนามไฟฟ้าที่ระนาบหน้าจานสะท้อนคลื่น

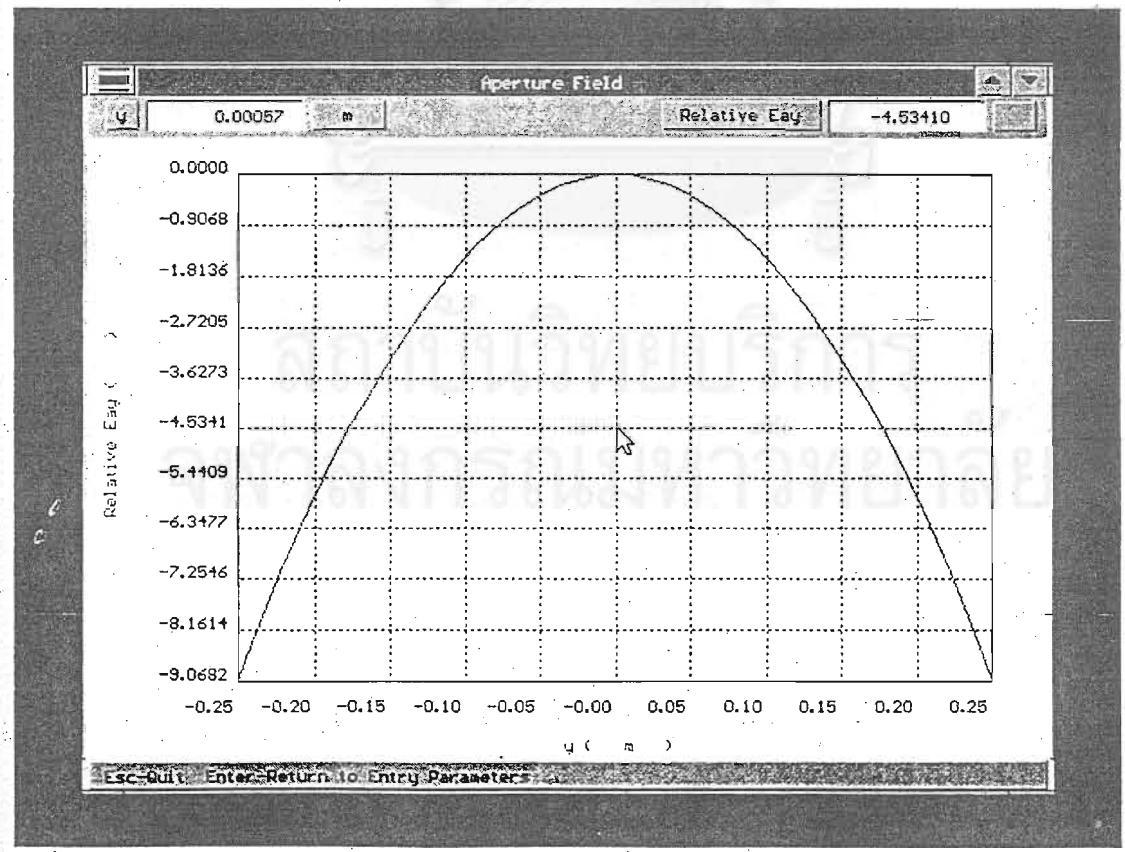
เมื่อผู้ใช้เลือกเมนู Plot Aperture Field of Symetrical Parabolic Antenna จะปรากฏหน้าจอขึ้นมาใหม่ โดยมีหน้าต่าง โดยจะต้องใส่ชื่อของแฟ้มข้อมูลที่ผู้ใช้ใช้ในการเก็บข้อมูลที่ได้จากการคำนวณสนามไฟฟ้าที่ระนาบหน้าจานสะท้อนคลื่น

จากนั้นโปรแกรมจะให้ผู้ใช้เลือกรูปแบบการแสดงผล ซึ่งจะแสดงเป็นกราฟที่ต่อเนื่องเป็นเส้น หรือเป็นจุดก็ได้ตามการเลือกใน Type Graph สิ่งต่อไปที่ผู้ใช้จะต้องกำหนดคือระนาบที่สนใจ และมาตราส่วน(scale) ตามลำดับ รูปที่ ผ7 แสดงหน้าจอสำหรับการจัดการแสดงผลการคำนวณ

เสร็จแล้วโปรแกรมก็จะแสดงภาพการกระจายความเข้มสนามไฟฟ้าที่ระนาบหน้าจานสะท้อนคลื่น ซึ่งสามารถอ่านค่าสนามที่ตำแหน่งต่าง ๆ ได้โดยตรง โดยใช้ ปุ่มลูกศร หรือ mouse เลื่อนไปยังจุดที่ต้องการ รูปที่ ผ8เป็นตัวอย่างภาพการกระจายความเข้มสนามไฟฟ้าที่ระนาบหน้าจานสะท้อน



รูปที่ ผ7 หน้าจอการสั่งให้โปรแกรมทำการแสดงผลการคำนวณในลักษณะรูปภาพ



รูปที่ ผ8 หน้าจอแสดงภาพการกระจายความเข้มสนามไฟฟ้าที่ระนาบหน้าจานสะท้อน

การคำนวณแบบรูปการแผ่พลังงานย่านสนามไกล

จากเมนูหลักของโปรแกรม เลือกตัวเลือกที่ 3 จะเข้าสู่เมนู Calculate Far Field Radiation Pattern ซึ่งเป็นเมนูที่ใช้สำหรับเปิดแฟ้มข้อมูลที่บันทึกในส่วนของการคำนวณสนามหน้าจาน นำข้อมูลนั้นมาคำนวณแบบรูปการแผ่พลังงานย่านสนามไกล แล้วบันทึกข้อมูลที่คำนวณได้ลงในแฟ้มข้อมูล ซึ่งมีขั้นตอนการปฏิบัติดังนี้

1. เลือกไปที่เมนู Open File & Calculate เมื่อกด Enter จะเข้าสู่เมนูที่ใช้สำหรับใส่ชื่อแฟ้มข้อมูลที่ได้จากการคำนวณสนามหน้าจาน
2. เมื่อใส่ชื่อแฟ้มที่ถูกต้องแล้ว จะเข้าสู่เมนูที่ให้ใส่ค่าพารามิเตอร์ต่าง ๆ ดังนี้

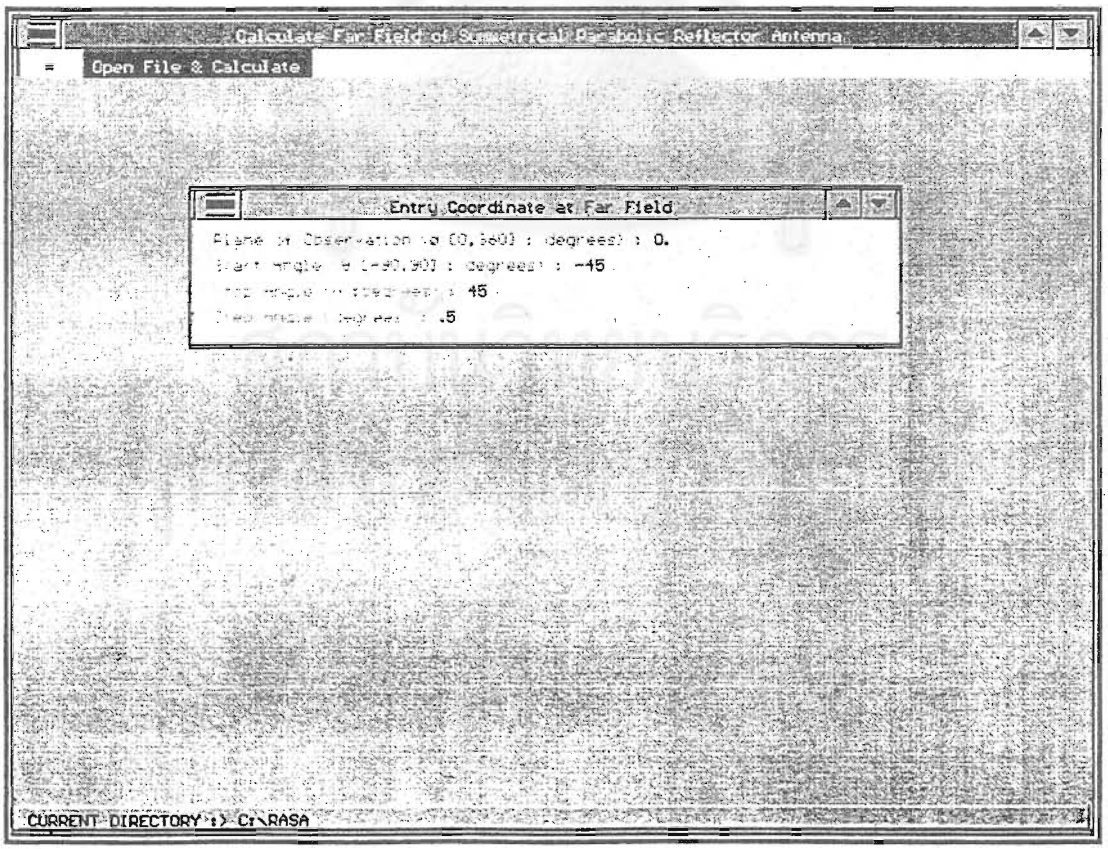
   Plane of Observation (ϕ ; 0 - 360 : degree) เป็นระนาบสังเกตสำหรับแสดง แบบรูปการแผ่พลังงานย่านสนามไกล ดังรูป ผ2

   Start Angle (θ : -90 - 90 : degree) เป็นมุมเริ่มต้นที่จะคำนวณสนามไฟฟ้าที่ระยะสนามไกล

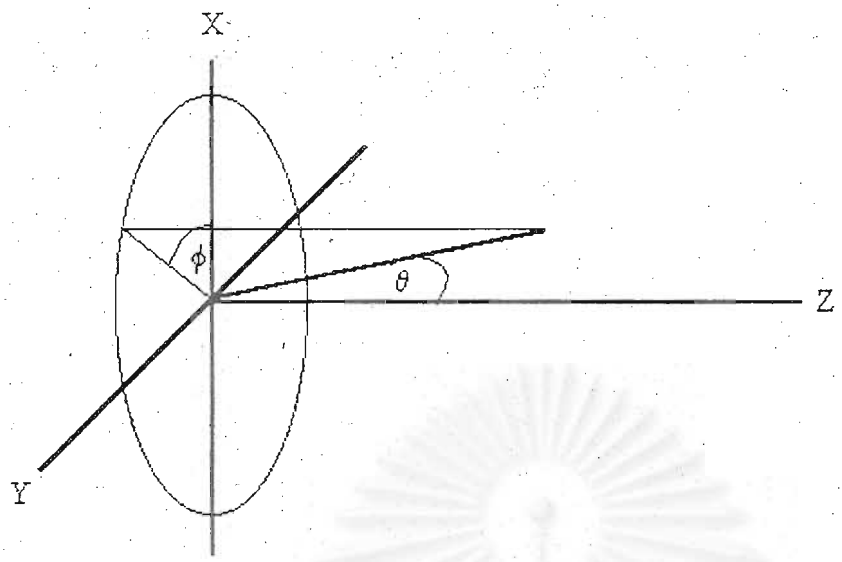
   Stop Angle (θ ; -90 - 90 : degree) เป็นมุมสุดท้ายที่จะคำนวณสนามไฟฟ้าที่ระยะสนามไกล

   Step Angle (degree) เป็นการกำหนดความละเอียดในการคำนวณ ว่าจะให้คำนวณค่าถัดไปที่มุมเพิ่มขึ้นจากเดิมเท่าใด ค่า Step Angle นี้ไม่ควรกำหนดให้มีค่าน้อยเกินไป เพราะจะทำให้โปรแกรมใช้เวลาในการคำนวณนาน ค่าที่เหมาะสมควรอยู่ในช่วง 0.1 - 1 องศา

รูปที่ ผ9 แสดงหน้าจอสำหรับกำหนดค่าปัจจัยให้โปรแกรมคำนวณแบบรูปการแผ่พลังงานย่านสนามไกล



รูปที่ ผ9 หน้าจอสำหรับกำหนดค่าปัจจัยให้โปรแกรมคำนวณแบบรูปการแผ่พลังงานย่านสนามไกล



รูปที่ ผ10 ระนาบสังเกตสำหรับการแสดงแบบรูปการแผ่พลังงานย่านสนามไกล

3. เมื่อใส่พารามิเตอร์ต่าง ๆ เรียบร้อยแล้ว โปรแกรมจะกำหนดให้เก็บข้อมูลลงใน แฟ้ม โดยใส่ชื่อแฟ้มตามต้องการ จากนั้นโปรแกรมจะเริ่มคำนวณค่า โดยมีตัวเลขเป็นเปอร์เซ็นต์แสดงให้เห็นว่าคำนวณค่าไปมากน้อยเพียงใดแล้ว ซึ่งขั้นตอนนี้จะใช้เวลานานพอสมควร

4. เมื่อต้องการออกไปสู่เมนูหลักของโปรแกรม ทำได้โดยเลือกเมนู exit

การแสดงแบบรูปการแผ่พลังงานย่านสนามไกล

จากเมนูหลักของโปรแกรม เลือกตัวเลือกที่ 4 จะเข้าสู่เมนูที่ให้ใส่พารามิเตอร์ต่าง ๆ ดังนี้

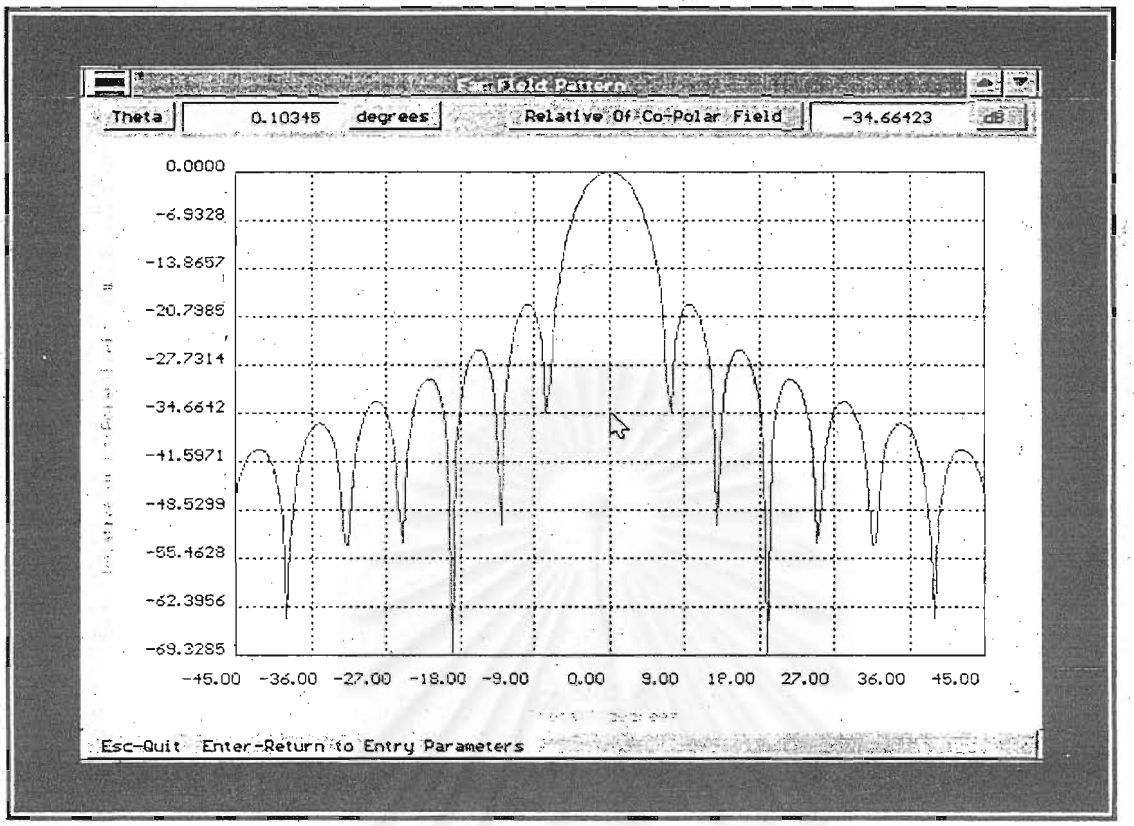
Enter FileName ใส่ชื่อแฟ้มที่บันทึกผลในส่วนการคำนวณแบบรูปการแผ่พลังงานย่านสนามไกล

Type of Graph เลือกชนิดของกราฟว่าเป็นกราฟเส้น (L,l) หรือเป็นจุด (D,d)

Select Polarisation เลือกดูผลกรณีโพลาไรเซชันร่วม(C)หรือโพลาไรเซชันไขว้(X)

Select Scale of Axis เลือกว่าจะใช้สเกลที่จัดให้อัตโนมัติ (A,a) หรือใช้สเกลที่เลือกเอง (D,d) ถ้าเลือกสเกลเอง จะมีการกำหนดค่าสูงสุดและค่าต่ำสุดของแกน x และแกน y

เมื่อใส่ค่าพารามิเตอร์เรียบร้อยแล้ว โปรแกรมจะแสดงแบบรูปการแผ่พลังงานที่สนามไกล ซึ่งสามารถอ่านค่าที่จุดต่าง ๆ ได้ โดยอาศัยปุ่มบังคับทิศทางเลื่อนแถบเส้นสีฟ้าในแนวแกนตั้งและแกนนอนให้ไปตัดกันที่จุดที่ต้องการ หรือใช้เมาส์เลื่อนไปที่จุดที่ต้องการ เมื่อต้องการออกไปสู่เมนูหลักของโปรแกรม กด Esc ถ้าต้องการพล็อตแบบรูปการแผ่พลังงานที่สนามไกลอีก กด Enter รูปที่ ผ11 เป็นตัวอย่างหน้าจอแสดงแบบรูปการแผ่พลังงานย่านสนามไกล



รูปที่ ผ11 ตัวอย่างหน้าจอแสดงแบบรูปการแผ่พลังงานย่านสนามไกล

รายการของโปรแกรมที่พัฒนาขึ้น

รายการของโปรแกรมที่พัฒนาขึ้นมีปรากฏอยู่ในภาคผนวกของรายงานฉบับสมบูรณ์นี้ สำหรับส่วนต่างๆของโปรแกรมมีดังต่อไปนี้

1.ส่วนคำนวณภาพการกระจายความเข้มสนามบริเวณระนาบหน้าจาน ส่วนนี้มีชื่อเรียกว่า Program CAF สามารถคำนวณผลทั้งกรณีคำนึงถึงและไม่คำนึงถึงผลกระทบจากการเลี้ยวเบนที่ขอบ นอกจากส่วนการคำนวณแล้วยังมีส่วนสำหรับเชื่อมโยงกับผู้ใช้เพื่อให้ผู้ใช้สามารถป้อนข้อมูลเกี่ยวกับค่าปัจจัยของสายอากาศที่ต้องการวิเคราะห์ และระบุแฟ้มสำหรับเก็บข้อมูลผลการคำนวณด้วย

2.ส่วนแสดงผลการคำนวณภาพการกระจายความเข้มสนามบริเวณระนาบหน้าจาน ส่วนนี้มีชื่อเรียกว่า Program PAF โดยนำผลการคำนวณจาก 1 มาแสดงที่หน้าจอ ดังนั้นในตอนเริ่มต้นจึงต้องเปิดแฟ้มข้อมูลที่ต้องการแสดงผลก่อน การตรวจสอบค่าผลการคำนวณ ณ ตำแหน่งต่างๆทำได้ด้วยการใช้ Mouse เลื่อนไปบนเส้นโค้งภาพการกระจายความเข้มสนามบริเวณระนาบหน้าจาน โปรแกรมส่วนนี้จึงทำงานหลักในการจัดการเตรียมหน้าจอสำหรับแสดงผล การอ่านแฟ้มข้อมูลและการวาดภาพการกระจายความเข้มสนามบริเวณระนาบหน้าจาน

3ส่วนคำนวณแบบรูปการแผ่พลังงานย่านสนามไกลส่วนนี้มีชื่อเรียกว่า Program NearFieldTo-FarField โดยนำผลจาก 1 มาทำการคำนวณต่อ วิธีคำนวณที่ใช้ คือ การแปลงฟูริเยร์เพื่อหาสเปกตรัมคลื่นระนาบซึ่งเป็นสิ่งเดียวกับแบบรูปการแผ่พลังงานย่านสนามไกล หลังจากได้ผลการคำนวณแล้วก็จัดเก็บข้อมูลเข้าแฟ้มเพื่อการแสดงผลต่อไป

4.ส่วนแสดงผลการคำนวณแบบรูปการแผ่พลังงานย่านสนามไกล ส่วนนี้มีชื่อเรียกว่า Program PFF โดยนำผลการคำนวณจาก 3 มาแสดงที่หน้าจอ ส่วนนี้จะคล้ายคลึงกับส่วนที่สองแต่มีข้อแตกต่างที่ลักษณะการแสดงผลที่ต้องแสดงการเปลี่ยนแปลงค่าความเข้มสนามเทียบกับตำแหน่งเชิงมุม

```
{Update 28 Nov 1996}
{Calculate Aperture Field}
Program CAF;
Uses Crt,Dos,Graph,UseGraph,KeyGraph,Menu1,Math,VectPlex;
Const Max1 = 600;
Type
  CArr1     = Array[1..Max1] Of Complex;
  AFHType   = Record
                Header : String[8];
                DtoLamda,Frequency,F_D,Dia,StepR,StepF : Real;
                SizeR,SizeF : Word;
                FeedType : String;
              End;
  AFType    = Record
                Eax,Eay,Eaz : CArr1;
              End;
Var
  Window : WinType;
  AFH : AFHType;
  AF : AFType;
  AFHfile : File Of AFHType;
  AFfile : File Of AFType;

  D,Freq,FtoD,Lamda,Beta,f,r : Real;
  xf,yf,zf,xp,yp,zp,StepX,StepY,xfinal,yfinal : Real;
  s,t,m,n : Word;
  qe,qh : Integer;
  Ego,Ed : CVec;
  PointS,ShortD,Cosine,Huygen : Boolean;
  Para,SetStartY,Save,NewFile,Quit : Boolean;
  Para1,Check : Byte;
  FeedStr,FileName,qeStr,qhStr : String;
  Regs : Registers;
  Ch : Char;
  StrTemp : String;

Procedure ErrorMessage(Text : String; Var ChTemp : Char);
Var Block : BlockType;
    Menu : MenuType;
    Pic1 : Pointer;
    No : Byte;
Begin
  No:=1;
  With Block do
  Begin
    Header:='Error'; StartX:=155; StartY:=150; EndX:=475; EndY:=250;
    Menu[1]:='  Ok  ';
```

```
    OpenBlock(Block,Pic1,9,15,15);
    SetColor(0);
    OutTextXY(StartX+20,StartY+30,Text);
    HSelectMenu(Menu,1,StartX+135,EndY-45,0,7,15,0,25,15,12,No,ChTemp);
    Repeat
      SelectMenuH(Menu,1,StartX+135,EndY-45,0,7,15,0,25,15,12,No,ChTemp);
    Until ChTemp in [#13,#27,#45];
  End;
  CloseBlock(Block,Pic1);
End;

Procedure Help(Var ChTemp : Char);
Begin
  ErrorMessage('Cannot locate help file, HELP.TXT',ChTemp);
End;

Procedure ChangeDir(Window : WinType);
Var Block  : BlockType;
    ChTemp : Char;
    Pic1   : Pointer;
    StartText,EndText,Len,Len1 : Word;
    Path,PathTemp : String;
    Result,StepCornerX,StepCornerY,WidthX,WidthY : Byte;
Begin
  StepCornerX:=25; StepCornerY:=20; WidthX:=4; WidthY:=3;
  GetDir(0,Path);
  With Block do
  Begin
    Header:='Change Directory';
    StartX:=20; StartY:=55; EndX:=617; EndY:=156;
  End;
  Repeat
    OpenBlock(Block,Pic1,3,15,15);
    Line3D(24,76,'',0,1,77,589,15,0,7,0,0,1,8,3);
    Line3D(26,82,'',0,1,21,585,0,15,13,0,0,1,8,3);
    Line3D(27,108,'',0,1,20,287,0,15,15,0,0,1,8,3);
    Line3D(320,108,'',0,1,20,289,0,15,15,0,0,1,8,3);
    Line3D(27,132,'Enter or Esc-Quit',0,1,15,583,0,15,7,0,0,1,8,2);
    OutTextXY(35,87,'CHANGE TO DIRECTORY :) ');
    MoveTo(35+TextWidth('CHANGE TO DIRECTORY :) ')+10,87);
    ReadString(Path,15,9,13,1,40);
    If Path=#27 then GetDir(0,Path);
    {$I-} ChDir(Path); {$I+}
    If IOResult<>0 then
    Begin
      SetColor(4);
      OutTextXY(65,113,'' DIRECTORY NOT FOUND '');
```

```
      Line3D(27,132,'Esc-Quit Enter-Continue',0,1,15,583,0,15,7,0,0,1,8,2);
      Result:=1;
      ChTemp:=ReadKey;
    End
    Else Result:=0;
    CloseBlock(Block,Pic1);
  Until ((Result=0) or (ChTemp=#27));
  GetDir(0,Path);
  PathTemp:='Current Directory :) '+Path;
  UpCaseStr(PathTemp);
  With Window do
  Begin
    StartText:=StartX+WidthX; EndText:=EndX-WidthX;
    Len:=TextWidth(PathTemp); Len:=(EndText-StartText-Len) div 2;
    Len1:=TextWidth(PathTemp)+StartText;
    If (Len<0) or (Len1>EndText) then
    Begin
      While Len1>EndText do
      Begin
        Len1:=TextWidth(PathTemp)+StartText;
        Delete(Path,Length(PathTemp),1);
      End;
      Line3D(StartX+WidthX,EndY-WidthY-StepCornerY+6,PathTemp,0,0,
          StepCornerY-6,EndX-StartX-2*WidthX,15,0,7,0,1,1,8,2);
    End
    Else
    Begin
      Line3D(StartX+WidthX,EndY-WidthY-StepCornerY+6,PathTemp,0,1,
          StepCornerY-6,EndX-StartX-2*WidthX,15,0,7,0,1,1,8,2);
    End;
  End;
End;

Procedure CheckQuit(Var ChTemp : Char);
Var Block  : BlockType;
    Menu   : MenuType;
    No     : Byte;
    Pic1   : Pointer;
Begin
  With Block do
  Begin
    Header:='Exit Parabolic Reflector Antenna';
    StartX:=100; StartY:=100; EndX:=530; EndY:=210;
    No:=1; Menu[1]:='Ok'; Menu[2]:='Cancel';
    OpenBlock(Block,Pic1,3,15,15);
    SetColor(4); SetTextStyle(0,0,3); OutTextXY(StartX+33,134,'!');
    CirCle(StartX+42,145,15);
```

```
    SetColor(0); SetTextStyle(2,0,4);
    OutTextXY(StartX+70,142,'This will end your Parabolic Reflector Antenna session');
  End;
  HSelectMenu(Menu,2,235,170,0,7,15,0,60,20,12,No,ChTemp);
  Repeat
    SelectMenuH(Menu,2,235,170,0,7,15,0,60,20,12,No,ChTemp);
  Until ChTemp in [#13,#27];
  If ChTemp=#13 then
  Begin
    Case No Of
      1 : Quit:=True;
    End;
  End;
  CloseBlock(Block,Pic1);
  SetTextStyle(2,0,4);
End;

Procedure MemInfo;
Var Block  : BlockType;
    Regs   : Registers;
    MemStr : String;
    DosMem : Word;
    Pic1   : Pointer;
Begin
  With Block do
  Begin
    Header:='Memory Information';
    StartX:=20; StartY:=50; EndX:=617; EndY:=163;
  End;
  Intr($12,Regs); DosMem:=Regs.AX;
  OpenBlock(Block,Pic1,6,15,5);
  Line3D(24,70,'',0,1,90,589,15,0,7,0,0,1,8,3);
  Line3D(27,73,'',0,1,13,582,15,0,3,0,0,1,8,3);
  Line3D(27,89,'',0,1,29,582,0,15,15,0,0,1,8,3);
  Line3D(27,121,'',0,1,13,582,15,0,3,0,0,1,8,3);
  Line3D(27,137,'',0,1,17,582,0,15,15,0,0,1,8,3);
  SetColor(15);
  OutTextXY(35,74,'Memory :) ');
  OutTextXY(35,122,'Installed Memory :) ');
  SetColor(0);
  Str(MaxAvail,MemStr);
  OutTextXY(38,93,'Maximum Contineous Memory : '+MemStr+' Bytes');
  Str(MemAvail,MemStr);
  OutTextXY(38,105,'Memory Available : '+MemStr+' Bytes');
  Str(DosMem,MemStr);
  OutTextXY(35,141,'Base : '+MemStr+' KBytes');
  OutTextXY(285,141,'Extended : ');
```

```
  Repeat Until KeyPressed;
  CloseBlock(Block,Pic1);
End;

{Input Parameter of Reflector Antenna}
Procedure ParaA(Var D,Freq,FtoD : Real);
Var Block : BlockType;
    Pic1  : Pointer;
    Str1  : Str80;
Begin
  With Block do
  Begin
    Header:='Entry Parameters for Symmetrical Parabolic Reflector Antenna';
    StartX:=85; StartY:=100; EndX:=545; EndY:=180;
    OpenBlock(Block,Pic1,14,0,15);
    Repeat
      SetColor(9);
      Str1:='Diameter [0.1-2.0 m] : '; OutTextXY(StartX+20,125,Str1);
      MoveTo(StartX+20+TextWidth(Str1),125); ReadRealNum(D,0,14,15,1,30);
    Until ((D>=0.1) and (D<=2.0));
    Freq:=Freq/Power(10,9);
    Repeat
      SetColor(9);
      Str1:='Operating Frequency [1-10 GHz] : '; OutTextXY(StartX+20,140,Str1);
      MoveTo(StartX+20+TextWidth(Str1),140); ReadRealNum(Freq,0,14,15,1,30);
    Until ((Freq>=1) and (Freq<=10));
    Freq:=Freq*Power(10,9);
    Repeat
      SetColor(9);
      Str1:='F/D Ratio : '; OutTextXY(StartX+20,155,Str1);
      MoveTo(StartX+20+TextWidth(Str1),155); ReadRealNum(FtoD,0,14,15,1,30);
    Until ((FtoD>0) and (FtoD<=5));
  End;
  CloseBlock(Block,Pic1);
End;

Procedure Configuration;
Var Block : BlockType;
    Pic1  : Pointer;
    Str1,Str2 : Str80;
Begin
  With Block do
  Begin
    Header:='Configuration of Symmetrical Parabolic Reflector Antenna';
    StartX:=105; StartY:=100; EndX:=525; EndY:=310;
    OpenBlock(Block,Pic1,14,0,15); SetColor(9);
    Line3D(StartX+3,StartY+20,'',0,1,186,414,15,0,7,0,0,1,8,3);
```

```
  Repeat Until KeyPressed;
  CloseBlock(Block,Pic1);
End;

{Input Parameter of Reflector Antenna}
Procedure ParaA(Var D,Freq,FtoD : Real);
Var Block : BlockType;
    Pic1  : Pointer;
    Str1  : Str80;
Begin
  With Block do
  Begin
    Header:='Entry Parameters for Symmetrical Parabolic Reflector Antenna';
    StartX:=85; StartY:=100; EndX:=545; EndY:=180;
    OpenBlock(Block,Pic1,14,0,15);
    Repeat
      SetColor(9);
      Str1:='Diameter [0.1-2.0 m] : '; OutTextXY(StartX+20,125,Str1);
      MoveTo(StartX+20+TextWidth(Str1),125); ReadRealNum(D,0,14,15,1,30);
    Until ((D>=0.1) and (D<=2.0));
    Freq:=Freq/Power(10,9);
    Repeat
      SetColor(9);
      Str1:='Operating Frequency [1-10 GHz] : '; OutTextXY(StartX+20,140,Str1);
      MoveTo(StartX+20+TextWidth(Str1),140); ReadRealNum(Freq,0,14,15,1,30);
    Until ((Freq>=1) and (Freq<=10));
    Freq:=Freq*Power(10,9);
    Repeat
      SetColor(9);
      Str1:='F/D Ratio : '; OutTextXY(StartX+20,155,Str1);
      MoveTo(StartX+20+TextWidth(Str1),155); ReadRealNum(FtoD,0,14,15,1,30);
    Until ((FtoD>0) and (FtoD<=5));
  End;
  CloseBlock(Block,Pic1);
End;

Procedure Configuration;
Var Block : BlockType;
    Pic1  : Pointer;
    Str1,Str2 : Str80;
Begin
  With Block do
  Begin
    Header:='Configuration of Symmetrical Parabolic Reflector Antenna';
    StartX:=105; StartY:=100; EndX:=525; EndY:=310;
    OpenBlock(Block,Pic1,14,0,15); SetColor(9);
    Line3D(StartX+3,StartY+20,'',0,1,186,414,15,0,7,0,0,1,8,3);
```

```
    Line3D(StartX+6,StartY+24,'',0,1,13,407,15,0,3,0,0,1,8,3);
    Line3D(StartX+6,StartY+39,'',0,1,95,407,0,15,15,0,0,1,8,3);
    Line3D(StartX+6,StartY+138,'',0,1,13,407,15,0,3,0,0,1,8,3);
    Line3D(StartX+6,StartY+153,'',0,1,40,407,0,15,15,0,0,1,8,3);
    SetColor(15);
    OutTextXY(StartX+9,StartY+25,'Antenna :) ');
    OutTextXY(StartX+9,StartY+139,'Feed :) ');
    SetColor(9); Str(D:8:3,Str1);
    OutTextXY(StartX+10,StartY+45,'Diameter of Reflector');
    MoveTo(StartX+10+TextWidth('Diameter of Reflector'),StartY+45);
    SetColor(0); OutText(' : '+Str1+' m'); SetColor(9);
    Str(Freq/Power(10,9):8:3,Str1);
    OutTextXY(StartX+10,StartY+60,'Operating Frequency');
    MoveTo(StartX+10+TextWidth('Diameter of Reflector'),StartY+60);
    SetColor(0); OutText(' : '+Str1+' GHz'); SetColor(9);
    Str(D/Lamda:8:3,Str1);
    OutTextXY(StartX+10,StartY+75,'D/Lamda');
    MoveTo(StartX+10+TextWidth('Diameter of Reflector'),StartY+75);
    SetColor(0); OutText(' : '+Str1); SetColor(9);
    Str(FtoD:8:3,Str1);
    OutTextXY(StartX+10,StartY+90,'F/D Ratio');
    MoveTo(StartX+10+TextWidth('Diameter of Reflector'),StartY+90);
    SetColor(0); OutText(' : '+Str1); SetColor(9);
    Str(f:8:3,Str1);
    OutTextXY(StartX+10,StartY+105,'Focus Distance  ');
    MoveTo(StartX+10+TextWidth('Diameter of Reflector'),StartY+105);
    SetColor(0); OutText(' : '+Str1+' m'); SetColor(9);
    OutTextXY(StartX+10,StartY+155,'Feed Type : '+FeedStr);
    If Cosine then
    Begin
      SetColor(0);
      Str(qe,Str1); Str(qh,Str2);
      OutTextXY(StartX+10,StartY+170,'Order E-Plane : '+Str1+'  Order H-Plane : '+Str2);
    End;
    SetColor(15);
  End;
  Repeat Until KeyPressed;
  CloseBlock(Block,Pic1);
End;

Procedure NoParaA(Var ChTemp : Char);
Begin
  ErrorMessage('You must enter parameter in Antenna Menu first.',ChTemp);
End;

Procedure ParaF(Var qe,qh : Integer);
Var Block : BlockType;
```

```
    Pic1  : Pointer;
    Str1 : Str80;
Begin
  With Block do
  Begin
    Header:='Entry Order of Cosine Feed';
    StartX:=125; StartY:=100; EndX:=415; EndY:=160;
    OpenBlock(Block,Pic1,14,0,15); SetColor(9);
    Str1:='Order of E-Plane : '; OutTextXY(StartX+10,125,Str1);
    MoveTo(StartX+10+TextWidth(Str1),125); ReadIntNum(qe,0,14,15,1,30);
    SetColor(9);
    Str1:='Order of H-Plane : '; OutTextXY(StartX+10,140,Str1);
    MoveTo(StartX+10+TextWidth(Str1),140); ReadIntNum(qh,0,14,15,1,30);
  End;
  CloseBlock(Block,Pic1);
End;

Procedure InitPara;
Begin
  Para:=False;
  PointS:=False; ShortD:=True; Cosine:=False; Huygen:=False;
  qe:=1; qh:=0;
  FeedStr:='Short Dipole';
  xf:=0; yf:=0; zf:=0; {Assume Feed at Origin}
  Save:=False;
End;

Procedure NearZeroV(Var E : CVec);
Var Zero : Real;
Begin
  Zero:=1e-10;
  If Abs(E[1].Re)<Zero then E[1].Re:=0;
  If Abs(E[1].Im)<Zero then E[1].Im:=0;
  If Abs(E[2].Re)<Zero then E[2].Re:=0;
  If Abs(E[2].Im)<Zero then E[2].Im:=0;
  If Abs(E[3].Re)<Zero then E[3].Re:=0;
  If Abs(E[3].Im)<Zero then E[3].Im:=0;
End;

Procedure NearZero(Var E : Complex);
Var Zero : Real;
Begin
  Zero:=1e-10;
  If Abs(E.Re)<Zero then E.Re:=0;
  If Abs(E.Im)<Zero then E.Im:=0;
End;
```

```
Procedure RevsAxis(x,y,z : Real; Var xdat,ydat,zdat : Real);
Begin
  xdat:=-x; ydat:=y; zdat:=-z;
End;

Procedure RectToSpher(x,y,z : Real; Var r,zeta,phi : Real);
Begin
  r:=Sqrt(x*x+y*y+z*z);
  zeta:=Atan(Sqrt(x*x+y*y),z);
  phi:=Atan(y,x);
End;

Procedure VSpherToRect(E : CVec; zeta,phi : Real; Var Ei : CVec);
Var CVTemp : CVec;
    CTemp : Complex;
Begin
  MulRC(Sin(zeta)*Cos(phi),E[1],CVTemp[1]);
  MulRC(Cos(zeta)*Cos(phi),E[2],CVTemp[2]);
  MulRC(-Sin(phi),E[3],CVTemp[3]);
  PlusC(CVTemp[1],CVTemp[2],CTemp); PlusC(CTemp,CVTEmp[3],CTemp);
  Ei[1]:=CTemp;
  MulRC(Sin(zeta)*Sin(phi),E[1],CVTemp[1]);
  MulRC(Cos(zeta)*Sin(phi),E[2],CVTemp[2]);
  MulRC(Cos(phi),E[3],CVTemp[3]);
  PlusC(CVTemp[1],CVTemp[2],CTemp); PlusC(CTemp,CVTEmp[3],CTemp);
  Ei[2]:=CTemp;
  MulRC(Cos(zeta),E[1],CVTemp[1]);
  MulRC(-Sin(zeta),E[2],CVTemp[2]);
  MulRC(0,E[3],CVTemp[3]);
  PlusC(CVTemp[1],CVTemp[2],CTemp); PlusC(CTemp,CVTEmp[3],CTemp);
  Ei[3]:=CTemp;
End;

Procedure Feed(xr,yr,zr : Real; Var Ei : CVec);
Var E : CVec;
    eksi : Complex;
    xdat,ydat,zdat,rdat,zetadat,phidat : Real;
Begin
  {Change x y z Coordinate on Surface to xdat ydat zdat Coordinate}
  RevsAxis(xr,yr,zr,xdat,ydat,zdat);
  RectToSpher(xdat,ydat,zdat,rdat,zetadat,phidat);
  CExpo(Beta*rdat,eksi);

  {Define Feed Pattern}
  If PointS or ShortD or Cosine Then
  Begin
    E[1].Re:=0; E[1].Im:=0; {R-Component in Feed Pattern is zero at Far Field}
```

```
    E[2].Re:=Power(Cos(zetadat),qe)*Sin(phidat); E[2].Im:=0;
    E[3].Re:=Power(Cos(zetadat),qh)*Cos(phidat); E[3].Im:=0;
  End;
  If Huygen Then
  Begin
    E[1].Re:=0; E[1].Im:=0; {R-Component in Feed Pattern is zero at Far Field}
    E[2].Re:=Power(1+Cos(zetadat),qe)*Sin(phidat); E[2].Im:=0;
    E[3].Re:=Power(1+Cos(zetadat),qh)*Cos(phidat); E[3].Im:=0;
  End;

  MulRC(1/rdat,eksi,eksi);
  MulC(E[2],eksi,E[2]);
  MulC(E[3],eksi,E[3]);

  {Change Electric Field in xdat ydat zdat Coordinate to x y z Coordinate}
  VSpherToRect(E,zetadat,phidat,Ei);
  MulRC(-1,Ei[1],Ei[1]);
  MulRC(-1,Ei[3],Ei[3]);
End;

Procedure Surface(x,y : Real; Var z : Real; Var ns : RVec);
Var Temp : Real;
Begin
  z:=(x*x+y*y)/(4*f)-f; Temp:=Sqrt(x*x+y*y+4*f*f);
  ns[1]:=-x/Temp; ns[2]:=-y/Temp; ns[3]:=(2*f)/Temp;
End;

Procedure Go(xp,yp,zp : Real; Var Ego : CVec);
Var xr,yr,zr,sr : Real;
    Ei,CVTemp1,CVTemp2 : CVec;
    ns : RVec;
    Dot,eksr : Complex;
Begin
  xr:=xp; yr:=yp; Surface(xr,yr,zr,ns);
  Feed(xr,yr,zr,Ei);
  DotCVRV(Ei,ns,Dot); MulRC(2,Dot,Dot);
  MulRCV(-1,Ei,CVTemp1); MulCRV(Dot,ns,CVTemp2);
  PlusCV(CVtemp1,CVTemp2,Ego);
  sr:=Sqrt((xp-xr)*(xp-xr)+(yp-yr)*(yp-yr)+(zp-zr)*(zp-zr));
  CExpo(Beta*sr,eksr);
  MulCCV(eksr,Ego,Ego);
  NearZeroV(Ego);
End;

Procedure UCrossRV(v1,v2 : RVec; Var v3 : RVec);
Var RVTemp : RVec;
    RTemp : Real;
```

```
Begin
  CrossRV(v1,v2,RVTemp); NormRV(RVTemp,RTemp); UnitV(RVTemp,RTemp,v3);
End;

Procedure Diffrac(xp,yp,zp,xd,yd,zd : Real; Var Check : Byte; Var Ed : CVec);
Var phi : Double;
    Temp,RTemp,Near90 : Real;
    si,sd,Re,sinbetai,pei,pd,phii,phid,p1i,p2i,per,Li,Lr : Real;
    A,Ephii,Ebetai,Ephid,Ebetad,f1,f2,Ds,Dh,CTemp : Complex;
    vsi,usi,vsd,usd,vr1,vr2,ue,ub,une,uts,uns : RVec;
    uphii,ubetai,uphid,ubetad,usi_sd,RVTemp : RVec;
    Ei,CVTemp1,CVTemp2 : CVec;
Begin
  phi:=Atan(yd,xd); Surface(xd,yd,zd,uns);
  vsi[1]:=xd-xf; vsi[2]:=yd-yf; vsi[3]:=zd-zf;
  NormRV(vsi,si); UnitV(vsi,si,usi);
  vsd[1]:=xp-xd; vsd[2]:=yp-yd; vsd[3]:=zp-zd;
  NormRV(vsd,sd); UnitV(vsd,sd,usd);

  If Abs(Sin(phi))<1e-15 then RTemp:=0 Else RTemp:=Sin(phi);
  If Abs(Cos(phi))<1e-15 then Temp:=0 Else Temp:=Cos(phi);

  vr1[1]:=-r*RTemp; vr1[2]:= r*Temp; vr1[3]:=0;
  vr2[1]:=-r*Temp; vr2[2]:=-r*RTemp; vr2[3]:=0;
  NormRV(vr1,RTemp); UnitV(vr1,-RTemp,ue);
  UCrossRV(vr1,vr2,ub); UCrossRV(ub,ue,une);
  NormRV(vr1,RTemp); Temp:=Power(RTemp,3); CrossRV(vr1,vr2,RVTemp);
  NormRV(RVTemp,RTemp); Re:=Temp/RTemp;
  UCrossRV(ue,uns,uts);
  UCrossRV(ue,usi,uphii); UCrossRV(usi,uphii,ubetai);
  UCrossRV(usd,ue,uphid); UCrossRV(usd,uphid,ubetad);
  DotRV(usi,ue,RTemp); sinbetai:=Sqrt(1-RTemp*RTemp);

  pei:=si; MinusRV(usi,usd,usi_sd);
  DotRV(une,usi_sd,RTemp);
  pd:=Power(1/pei-RTemp/(Re*sinbetai*sinbetai),-1);

  If Abs(sd+pd)>Power(10,-8) then
  Begin
    DotRV(uts,uphii,RTemp);
    If RTemp<=0 then
    Begin
      DotRV(uns,uphii,RTemp); phii:=ArcCos(RTemp);
    End
    Else
    Begin
      DotRV(uns,uphii,RTemp); phii:=2*Pi-ArcCos(RTemp);
```

```
End;

DotRV(uts,uphid,RTemp);
If RTemp<=0 then
Begin
  DotRV(uns,uphid,RTemp); phid:=ArcCos(RTemp);
End
Else
Begin
  DotRV(uns,uphid,RTemp); phid:=2*Pi-ArcCos(RTemp);
End;

p1i:=si; p2i:=si;
DotRV(uns,une,Temp); DotRV(usi,uns,RTemp);
per:=Power(1/pei-(2*Temp*RTemp)/(Re*sinbetai*sinbetai),-1);

Li:=(sd*sinbetai*sinbetai)/(1+sd/si);
Lr:=sd*(1+sd/per)*sinbetai*sinbetai;

Feed(xd,yd,zd,Ei);

DotCVRV(Ei,uphii,Ephii);
DotCVRV(Ei,ubetai,Ebetai);

{Check value in square root may be positive or negative}
If 1/(sd*(1+sd/pd))>=0 then
Begin
  A.Re:=Sqrt(1/(sd*(1+sd/pd)));
  A.Im:=0;
End
Else
Begin
  A.Re:=0;
  A.Im:=Sqrt(-1/(sd*(1+sd/pd)));
End;

RTemp:=Beta*Li*2*Power(Cos((phid-phii)/2),2);
Fresnel(RTemp,CTemp);
MulRC(1/Cos((phid-phii)/2),CTemp,f1);

If Abs((phid+phii)/2-Pi/2)<1e-10 then Near90:=Pi/2
Else Near90:=(phid+phii)/2;
RTemp:=Beta*Lr*2*Power(Cos(Near90),2);
Fresnel(RTemp,CTemp);
MulRC(1/Cos(Near90),CTemp,f2);

CExpo(Pi/4,CTemp);
```

```
    RTemp:=-1/(Sqrt(8*Pi*Beta)*sinbetai);
    MulRC(RTemp,CTemp,Ds);
    MinusC(f1,f2,CTemp); MulC(Ds,CTemp,Ds);
    CExpo(Pi/4,CTemp);
    RTemp:=-1/(Sqrt(8*Pi*Beta)*sinbetai);
    MulRC(RTemp,CTemp,Dh);
    PlusC(f1,f2,CTemp); MulC(Dh,CTemp,Dh);

    MulC(Ephii,Dh,CTemp); MulC(CTemp,A,CTemp); CExpo(Beta*sd,Ephid);
    MulC(CTemp,Ephid,Ephid);MulRC(-1,Ephid,Ephid);
    MulC(Ebetai,Ds,CTemp); MulC(CTemp,A,CTemp); CExpo(Beta*sd,Ebetad);
    MulC(CTemp,Ebetad,Ebetad);MulRC(-1,Ebetad,Ebetad);

    MulCRV(Ephid,uphid,CVTemp1); MulCRV(Ebetad,ubetad,CVTemp2);
    PlusCV(CVTemp1,CVTemp2,Ed);
    Check:=0;
  End
  Else
  Begin
    Ed[1].Re:=0; Ed[1].Im:=0;
    Ed[2].Re:=0; Ed[2].Im:=0;
    Ed[3].Re:=0; Ed[3].Im:=0;
    Check:=1;
  End;
End;

{Calculate Diffracted Field by Two Point GTD}
Procedure Gtd(xp,yp,zp : Real; Var Ed : CVec);
Var Check : Byte;
    Ed1,Ed2 : CVec;
    xd,yd,zd : Real;
    phi : Double;
    ns : RVec;
Begin
  phi:=Atan(yp,xp);
  xd:=r*Cos(phi); yd:=r*Sin(phi);
  If Abs(xd)<1e-15 then xd:=0;
  If Abs(yd)<1e-15 then yd:=0;
  Diffrac(xp,yp,zp,xd,yd,zd,Check,Ed1);
  If Check=0 then
  Begin
    Ed:=Ed1;
    xd:=-xd; yd:=-yd; Surface(xd,yd,zd,ns);
    Diffrac(xp,yp,zp,xd,yd,zd,Check,Ed2);
    PlusCV(Ed,Ed2,Ed);
  End
  Else Ed:=Ed1;
```

```
  NearZeroV(Ed);
End;

{Initial Aperture Field : Assign Value for sending to Plot Graph}
Procedure InitAFH(Var AFH : AFHType);
Begin
  With AFH do
  Begin
    Header:='APERTURE';
    DtoLamda:=D/Lamda; Frequency:=Freq; F_D:=FtoD; Dia:=D;
    StepR:=StepY; StepF:=StepX; FeedType:=FeedStr;
    SizeF:=s; {X-Axis}  SizeR:=t; {Y-Axis}
  End;
End;

Procedure InitAF(Var AF : AFType);
Begin
  With AF do
  Begin
    For m:=1 to Max1 do
    Begin
      Eax[m].Re:=0; Eax[m].Im:=0;
      Eay[m].Re:=0; Eay[m].Im:=0;
      Eaz[m].Re:=0; Eaz[m].Im:=0;
    End;
  End;
End;

Function FileExist(FName : String) : Boolean;
Var fi : Text;
Begin
  {$I-} Assign(fi,FName); Reset(fi); Close(fi); {$I+}
  FileExist:=(IOResult=0) and (FName<>'');
End;

Procedure SaveAFtoDisk(Var FileName : String; Var Save,NewFile : Boolean);
Var Block : BlockType;
    Menu  : MenuType;
    Pic1  : Pointer;
    Dir   : DirStr;
    Name  : NameStr;
    Ext   : ExtStr;
    No    : Byte;
    Str1  : Str80;
    ChTemp : Char;
Begin
  Repeat
```

```
  No:=1;
  With Block do
  Begin
    Header:='Save Aperture Field to Disk';
    StartX:=90; StartY:=100; EndX:=540; EndY:=160;
    OpenBlock(Block,Pic1,14,0,15); SetColor(9);
    Str1:='Save Aperture Field to FileName : '; OutTextXY(StartX+25,130,Str1);
    MoveTo(StartX+25+TextWidth(Str1),130); ReadString(FileName,0,14,15,1,50);
  End;
  FSplit(FileName,Dir,Name,Ext);
{   If Ext='' then
  Begin
    Ext:='.afp'; Filename:=Filename+Ext;
  End;}
  CloseBlock(Block,Pic1);
  If FileExist(FileName) then
  Begin
    With Block do
    Begin
      Header:='Warning';
      StartX:=160; StartY:=100; EndX:=460; EndY:=200;
      OpenBlock(Block,Pic1,9,15,15);
      Menu[1]:=' Yes '; Menu[2]:=' No ';
      SetColor(0);
        OutTextXY(StartX+25,StartY+30,'File '+Name+Ext+' already exists. Overwrite?');
      HSelectMenu(Menu,2,StartX+90,EndY-45,0,7,15,0,40,15,12,No,ChTemp);
      Repeat
        SelectMenuH(Menu,2,StartX+90,EndY-45,0,7,15,0,40,15,12,No,ChTemp);
      Until ChTemp in [#13,#27,#45];
    End;
    If ChTemp=#13 then
    Begin
      Case No Of
        1 : Begin Save:=True; NewFile:=False; End;
        2 : Begin Save:=False; NewFile:=False; End;
      End;
    End;
    CloseBlock(Block,Pic1);
  End
  Else
  Begin
    If (FileName=#27) or (FileName=#45) or (FileName=#13) then
    Begin
      Save:=False; NewFile:=False;
    End
    Else
    Begin
```

```
      FSplit(FileName,Dir,Name,Ext);
      Save:=True; NewFile:=True;
    End;
  End;
 Until (FileName=#13) or (FileName=#27) or (FileName=#45) or NewFile or
    (ChTemp in [#13,#27,#45]);
End;

Procedure Calculate(Var ChTemp : Char);
Var Block : BlockType;
  Pic1 : Pointer;
  Str1,Str2 : Str80;
Begin
 s:=Round((r-(-r))/(Lamda/8)); t:=Round((r-(-r))/(Lamda/8));
 If Odd(s) then s:=s+1;
 If Odd(t) then t:=t+1;
 StepX:=(r-(-r))/s; StepY:=(r-(-r))/t;
 SaveAFtoDisk(FileName,Save,NewFile);
 If (Save or NewFile) and ((FileName<>#27) or (FileName<>#45)) and
   (FileName<>'') then
 Begin
  InitAFH(AFH);
  If NewFile then
  Begin
   Assign(AFHfile,FileName);
   ReWrite(AFHfile); Write(AFHfile,AFH); Close(AFHfile);
   Assign(AFfile,FileName); ReSet(AFfile);
  End
  Else
  Begin
   Assign(AFHfile,FileName);
   ReSet(AFHfile); Seek(AFHfile,0); Write(AFHfile,AFH); Close(AFHfile);
   Assign(AFfile,FileName); ReSet(AFfile);
  End;
  n:=1; zp:=zf; xp:=-r; xfinal:=-r+s*StepX; yfinal:=-r+t*StepY;
  With Block do
  Begin
   Header:='Calculating Aperture Field and Save to Disk';
   StartX:=120; StartY:=100; EndX:=500; EndY:=180;
   OpenBlock(Block,Pic1,14,0,15); SetColor(9);
   Str1:='Calculating Aperture Field '; OutTextXY(StartX+70,140,Str1);
  End;
  Repeat
   InitAF(AF);
   With AF do
   Begin
    If (xp=-r) or (xp=xfinal) then
```

```
        Begin
        m:=Trunc(s/2)+1; yp:=0;
        Go(xp,yp,zp,Ego);
        Eax[m]:=Ego[1]; Eay[m]:=Ego[2]; Eaz[m]:=Ego[3];
        If Para1=1 then
        Begin
          Gtd(xp,yp,zp,Ed);
          PlusC(Eax[m],Ed[1],Eax[m]);
          PlusC(Eay[m],Ed[2],Eay[m]);
          PlusC(Eaz[m],Ed[3],Eaz[m]);
        End;
      End
      Else
      Begin
        m:=1; If n=Trunc(t/2)+1 then SetStartY:=True Else SetStartY:=False;
        yp:=-r;
        Repeat
          If (yp>=-Sqrt(xfinal*xfinal-xp*xp)) and
             (yp<= Sqrt(xfinal*xfinal-xp*xp)) or SetStartY then
          Begin
              Go(xp,yp,zp,Ego);
            Eax[m]:=Ego[1]; Eay[m]:=Ego[2]; Eaz[m]:=Ego[3];
            If Para1=1 then
            Begin
              Gtd(xp,yp,zp,Ed);
              PlusC(Eax[m],Ed[1],Eax[m]);
              PlusC(Eay[m],Ed[2],Eay[m]);
              PlusC(Eaz[m],Ed[3],Eaz[m]);
            End;
          End;
          m:=m+1; yp:=-r+(m-1)*StepY;
        Until yp>yfinal;
      End;
      Seek(AFfile,n); Write(AFfile,AF);
      n:=n+1; xp:=-r+(n-1)*StepX;
    End;
    SetColor(0);
    Bar(Block.StartX+70+TextWidth(Str1),140,Block.StartX+70+TextWidth(Str1)+50,155);
    Str((n-1)/(s+1)*100:5:2,Str2);
    MoveTo(Block.StartX+70+TextWidth(Str1),140); OutText(Str2+' %');
  Until xp>xfinal;
  Close(AFfile);
  Repeat Until KeyPressed;
  CloseBlock(Block,Pic1);
 End
 Else ChTemp:=FileName[1];
End;
```

```
Procedure WinH;
Begin
  With Window do
  Begin
    MaxCol:=4;
    MenuH[1].Menu:=' = ';
    MenuH[2].Menu:=' Antenna ';
    MenuH[3].Menu:=' Feed ';
    MenuH[4].Menu:=' Calculate ';
  End;
End;

Procedure WinV;
Begin
  With Window do
  Begin
    MenuH[1].MaxRow:=3;
    MenuH[1].MenuV[1]:=' Change Dir ';
    MenuH[1].MenuV[2]:=' Help ';
    MenuH[1].MenuV[3]:=' Exit ';

    MenuH[2].MaxRow:=3;
    MenuH[2].MenuV[1]:=' Parameter ';
    MenuH[2].MenuV[2]:=' Configuration ';
    MenuH[2].MenuV[3]:=' Memory Information ';

    MenuH[3].MaxRow:=4;
    MenuH[3].MenuV[1]:=' Point Source ';
    MenuH[3].MenuV[2]:=' Short Dipole ';
    MenuH[3].MenuV[3]:=' Huygen Source ';
    MenuH[3].MenuV[4 ]:=' Cosine order n ';

    MenuH[4].MaxRow:=2;
    MenuH[4].MenuV[1]:=' No Diffraction ';
    MenuH[4].MenuV[2]:=' Include Diffraction ';
  End;
End;

Begin {Main}
  OpenGraph;
  ChangeColor(2,50);   ChangeColor(14,79); ChangeColor(11,46);
  ChangeColor(10,126); ChangeColor(13,43); ChangeColor(12,69);
  ChangeColor(9,116);  ChangeColor(5,42);  ChangeColor(6,84);
  ChangeColor(3,49);
  InitialWin(Window); WinH; WinV; Quit:=False;
  InputValueToWin(Window,0,0,GetMaxX,GetMaxY,
```

```
        'Calculate Aperture Field of Symmetrical Parabolic Reflector Antenna');
LoadWin(Window,9,15,1,14);
MenuHor(Window,0,15,15,3,1);
InitPara;
Repeat
 HorMenu(Window,0,15,15,3,1,Ch);
 With Window do
 Begin
  If (Ch=#13) and (MenuH[Choice].MenuV[1]<>'') then
  Begin
   Repeat
      VerMenu(Window,0,15,15,3,1,0,15,15,3,1,Ch);
    Until Ch in [#13,#27,#45];
   If Ch=#13 then
   Begin
    Case Choice Of
     1 : Begin
         Case MenuH[Choice].ChoiceSub Of
          1 : ChangeDir(Window);
          2 : Help(Ch);
          3 : CheckQuit(Ch);
         End; {Menu Choice 1}
        End;
     2 : Begin
         Case MenuH[Choice].ChoiceSub Of
          1 : Begin
              ParaA(D,Freq,FtoD); Para:=True;
              f:=FtoD*D; {Focus Distance}
              Lamda:=(3*Power(10,8))/Freq; {Operating Wavelenght}
              Beta:=2*Pi/Lamda; {Propagation Constant}
              r:=D/2;
             End;
          2 : Begin
               If Para then Configuration Else NoParaA(Ch);
              End;
          3 : MemInfo;
         End; {Menu Choice 2}
        End;
     3 : Begin
         Case MenuH[Choice].ChoiceSub Of
          1 : Begin
               PointS:=True; ShortD:=False;
               Huygen:=False; Cosine:=False;
               qe:=0; qh:=0; FeedStr:='Point Source';
               AFH.FeedType:=FeedStr;
              End;
          2 : Begin
```

```
                PointS:=False; ShortD:=True;
                Huygen:=False; Cosine:=False;
              qe:=1; qh:=0; FeedStr:='Short Dipole';
              AFH.FeedType:=FeedStr;
            End;
          3 : Begin
              PointS:=False; ShortD:=False;
              Huygen:=True;  Cosine:=False;
              qe:=1; qh:=1; FeedStr:='Huygen Source';
              AFH.FeedType:=FeedStr;
            End;
          4 : Begin
              PointS:=False; ShortD:=False;
              Huygen:=False; Cosine:=True;
              ParaF(qe,qh);
              Str(qe,qeStr); Str(qh,qhStr); {Send FeedType to Plot Graph}
              FeedStr:='Cosine Feed';
              AFH.FeedType:=FeedStr;
            End;
        End; {Menu Choice 3}
      End;
    4 : Begin
        Case MenuH[Choice].ChoiceSub Of
          1 : Begin
              If Para then
              Begin
                  Para1:=0; Calculate(Ch);
              End
              Else NoParaA(Ch);
            End;
          2 : Begin
              If Para then
              Begin
                Para1:=1; Calculate(Ch);
              End
              Else NoParaA(Ch);
            End;
        End; {Menu Choice 4}
      End; {Of Case}
      End;
  End;
 End;
End;
If Ch=#45 then CheckQuit(Ch);
Until Quit;
CloseGraph;
End. {Main}
```

```
{Update 4 Dec 1996}
{Plot Aperture Field From Disk}
Program PAF;
Uses Dos,Crt,Graph,UseGraph,KeyGraph,Math,VectPlex,Menu1,Open3D;
Const MaxV = 600;
      Max1 = 600;
      AFformat = 'APERTURE';
Type
  VecArr = Array[1..MaxV] of Real;
  CArr1  = Array[1..Max1] Of Complex;
  AFHType = Record
              Header : String[8];
              DtoLamda,Frequency,F_D,Dia,StepR,StepF : Real;
              SizeR,SizeF : Word;
              FeedType : String;
            End;
  AFType = Record
             Eax,Eay,Eaz : CArr1;
           End;
Var
  AFH : AFHType;
  AF : AFType;
  AFHfile : File of AFHType;
  AFfile : File of AFType;
  XValue,YValue : VecArr;
  Block : BlockType;
  D,Start,Step : Real;
  FileName,Auto,XLable,YLable,GraphType,Path,Name : String;
  UnitX,UnitY : Str20;
  MinX1,MaxX1,MinY1,MaxY1,MinX,MaxX,MinY,MaxY : Real;
  x1,y1,x2,y2,n : Word;
  Regs : Registers;
  SetDL,Ch,PlaneChoice : Char;
  Checkout,Format : Boolean;

Function FileExist(FName : String) : Boolean;
Var fi : Text;
Begin
  {$I-} Assign(fi,FName); Reset(fi); Close(fi); {$I+}
  FileExist:=(IOResult=0) and (FName<>'');
End;

Procedure ErrorMessage(HText,Text1,Text2,Text3 : String);
Var Block : BlockType;
    Menu : MenuType;
    Pic1 : Pointer;
    No : Byte;
```

```
    ChTemp : Char;
Begin
  No:=1;
  With Block do
  Begin
    Header:=HText;
    StartX:=135; EndX:=520; StartY:=150; EndY:=270;
    Menu[1]:=' Ok ';
    OpenBlock(Block,Pic1,9,15,15);
    SetColor(0);
    OutTextXY(StartX+20,StartY+30,Text1);
    OutTextXY(StartX+20,StartY+45,Text2);
    Line3D(StartX+3,EndY-20,'',0,1,17,380,15,0,7,0,0,1,8,3);
    SetColor(1);
    OutTextXY(StartX+10,EndY-18,Text3);
    HSelectMenu(Menu,1,StartX+165,EndY-50,0,7,15,0,25,15,12,No,ChTemp);
    Repeat
      SelectMenuH(Menu,1,StartX+165,EndY-50,0,7,15,0,25,15,12,No,ChTemp);
    Until ChTemp in [#13,#27,#45];
    Ch:=ChTemp;
  End;
  CloseBlock(Block,Pic1);
End;

{Utility Of Mouse}
Procedure MouseOn;
Begin
  Regs.Ax:=1; Intr($33,Regs);
End;

Procedure MouseOff;
Begin
  Regs.Ax:=2; Intr($33,Regs);
End;

Procedure SetMouseXY(x,y : Word);
Begin
  Regs.CX:=x; Regs.DX:=y;
  Regs.AX:=4; Intr($33,Regs);
End;

Procedure SetMouseLR(Left,Right : Word);
Begin
  Regs.CX:=Left; Regs.DX:=Right;
  Regs.AX:=7; Intr($33,Regs);
End;
```

```
Procedure SetMouseTB(Top,Bottom : Word);
Begin
  Regs.CX:=Top; Regs.DX:=Bottom;
  Regs.AX:=8; Intr($33,Regs);
End;

Procedure ReadMouseXY(Var x,y : Word);
Begin
  Regs.AX:=4; Intr($33,Regs);
  x:=Regs.CX; y:=Regs.DX;
End;

Procedure MouseBound(Left,Right,Top,Bottom : Word);
Begin
  SetMouseLR(Left,Right); SetMouseTB(Top,Bottom);
  SetMouseXY(Round((Left+Right)/2),Round((Top+Bottom)/2)); MouseOn;
End;

Procedure CheckPress(Var Regs : Registers);
Begin
  Regs.AX:=3; Intr($33,Regs);
End;

Procedure InitXYVector;
Var i : Word;
Begin
  For i:=1 to MaxV do
  Begin
    XValue[i]:=0; YValue[i]:=0;
  End;
End;

Procedure Design(XValue,YValue : VecArr; m : Word; MinX,MaxX,MinY,MaxY : Real;
                 UnitX,UnitY : Str20; Xlable,YLable : Str80; Block : BlockType;
                 x1,y1,x2,y2 : Word; SetDL : Char; Var ChTemp : Char);
Var CenterX,CenterY,RatioX,RatioY : Real;
    StartX,StartY,EndX,EndY : Word;
    XConstant,YConstant,ABlankX : Word;
    SetX,SetY : Integer;
    Pic1,Pic2    : Pointer;
    StepX,StepY  : Real;

  Procedure PositionXY(XTemp,YTemp : Real; Var x,y : Word);
  Begin
    x:=Round(StartX+(XTemp-MinX)*RatioX);
    y:=Round(EndY-(YTemp-MinY)*RatioY);
  End;
```

```
Procedure FillCh(Var Blank : String; C : Byte; Ch : Char);
Var i : Byte;
Begin
  Blank:='';
  For i:=1 to C do Blank:=Blank+Ch;
End;

Function PixelBlank(n : Byte) : Word;
Var Blank : String;
Begin
  FillCh(Blank,n,' ');
  PixelBlank:=TextWidth(Blank);
End;

Procedure ReadPosition(x1,y1,x2,y2 : Word; Var ChTemp : Char);
Var Str1,Str2 : Str20;
    StrTemp   : Str80;
    XReal,YReal : Real;
    XConst,YConst,x,y,xMs,yMs : Word;
Begin
  ChTemp:=#0;
  SetViewPort(StartX,StartY,EndX,EndY,ClipOn);
  SetColor(1); SetWriteMode(XORPut);
  Line(0,Round(2*CenterY-YConstant),Round(2*CenterX),
       Round(2*CenterY-YConstant));
  Line(XConstant,0,XConstant,Round(2*CenterY));
  SetWriteMode(COPYPut);
  MouseBound(StartX,EndX,StartY,EndY);
  SetMouseXY(StartX+XConstant,StartY+YConstant);
  ReadMouseXY(xMs,yMs); {Read old position to Compare with new position}
  Repeat
    If (Not KeyPressed) then {Check Condition Between KeyBoard and Mouse}
    Begin
      MouseOn;
      SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
      CheckPress(Regs);
      Case Regs.BX Of
        0 : Begin {No Presse}
              ReadMouseXY(x,y);
              XConst:=x-StartX; YConst:=y-StartY;
              XReal:=MinX+(XConst/RatioX);
              YReal:=MaxY-(YConst/RatioY);
              Str(XReal:10:5,Str1); Str(YReal:10:5,Str2);
              If (x=xMs) and (y=yMs) then
              Begin
                StrTemp:=XLable;
```

```
            SetColor(0);
            OutTextXY(x1+14+TextWidth(StrTemp)+SetX,y1+26,''+Str1);
            OutTextXY(x2-TextWidth(UnitY)-SetY-PixelBlank(ABlankX),
                  y1+26,' '+Str2);
          End
          Else
          Begin
            SetWriteMode(COPYPut);
            SetFillStyle(1,15); {SetWriteMode(COPYPut);}
            StrTemp:=XLable;
            Bar(x1+14+TextWidth(StrTemp)+7,y1+24,x1+14+
               TextWidth(StrTemp)+5+PixelBlank(ABlankX),y1+24+
               TextHeight('S')+4);
            SetColor(0);
            {Write XValue}
            OutTextXY(x1+14+TextWidth(StrTemp)+SetX,y1+26,''+Str1);
            StrTemp:=YLable;
            SetFillStyle(1,15); {SetWriteMode(COPYPut);}
            Bar(x2-TextWidth(UnitY)-12-PixelBlank(AblankX),y1+24,
            x2-TextWidth(UnitY)-15,y1+24+TextHeight('S')+4);
            SetColor(0);
            {Write YValue}
            OutTextXY(x2-TextWidth(UnitY)-SetY-PixelBlank(ABlankX),
                  y1+26,' '+Str2);
            ReadMouseXY(xMs,yMs);
          End;
        End;
    1 : ChTemp:=#13;
    2 : ChTemp:=#27;
  End;
End
Else
Begin
  SetWriteMode(COPYPut);
  MouseOff;
  SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
  XReal:=MinX+(XConstant/RatioX);
  YReal:=MinY+(YConstant/RatioY);
  Str(XReal:10:5,Str1); Str(YReal:10:5,Str2);

  StrTemp:=XLable;
  SetFillStyle(1,15); {SetWriteMode(COPYPut);}
  Bar(x1+14+TextWidth(StrTemp)+7,y1+24,x1+14+TextWidth(StrTemp)+5+
     PixelBlank(ABlankX),y1+24+TextHeight('S')+4);
  SetColor(0);
  OutTextXY(x1+14+TextWidth(StrTemp)+SetX,y1+26,''+Str1);
  StrTemp:=YLable;
```

```
      SetFillStyle(1,15); {SetWriteMode(COPYPut);}
      Bar(x2-TextWidth(UnitY)-12-PixelBlank(ABlankX),y1+24,
          x2-TextWidth(UnitY)-15,y1+24+TextHeight('S')+4);
      SetColor(0);
      OutTextXY(x2-TextWidth(UnitY)-SetY-PixelBlank(ABlankX),y1+26,' '+Str2);

      SetViewPort(StartX,StartY,EndX,EndY,ClipOn);
      ChTemp:=ReadKey;
      Case ChTemp Of
        #80 : Begin Dec(YConstant); If YConstant<0 then YConstant:=0; End;
        #72 : Begin
                Inc(YConstant);
                If YConstant>Round(2*CenterY) then YConstant:=Round(2*CenterY);
              End;
        #75 : Begin Dec(XConstant); If XConstant<0 then XConstant:=0; End;
        #77 : Begin
                Inc(XConstant);
                If XConstant>Round(2*CenterX) then XConstant:=Round(2*CenterX);
              End;
      End;
      SetColor(1); SetWriteMode(XORPut);
      Line(0,Round(2*CenterY-YConstant),Round(2*CenterX),
           Round(2*CenterY-YConstant));
      Line(XConstant,0,XConstant,Round(2*CenterY));
    End;
  Until ChTemp in [#13,#27,#45];
  SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
  SetWriteMode(COPYPut);
  MouseOff;
End;

Procedure GridLine(GridX,GridY : Boolean);
Var x1,y1,x2,y2 : Word;
    i : Byte;
Begin
  If GridX=True then
  Begin
    For i:=1 to 9 do
    Begin
      SetLineStyle(DottedLn,0,1);
      PositionXY(MinX+i*StepX,MinY,x1,y1);
      PositionXY(MinX+i*StepX,MaxY,x2,y2);
      Line(x1,y1,x2,y2);
    End;
  End;
  If GridY=True then
  Begin
```

```
    For i:=1 to 9 do
    Begin
      SetLineStyle(DottedLn,0,1);
      PositionXY(MinX,MinY+i*StepY,x1,y1);
      PositionXY(MaxX,MinY+i*StepY,x2,y2);
      Line(x1,y1,x2,y2);
    End;
  End;
  SetLineStyle(0,1,0);
End;

Procedure ScaleXY(Block : BlockType);
Var i : Byte;
    x1,y1,x2,y2 : Word;
    Str1 : Str20;
Begin
  For i:=0 to 10 do
  Begin
    PositionXY(MinX+i*StepX,MinY,x1,y1);
    Str((MinX+i*StepX):5:2,Str1);
    OutTextXY(x1-30,y1+8,Str1);
    PositionXY(MinX,MinY+i*StepY,x1,y1);
    Str((MinY+i*StepY):8:4,Str1);
    SetTextJustify(RightText,CenterText);
    OutTextXY(x1-5,y1-5,Str1);
    SetTextJustify(LeftText,TopText);
  End;
  SetColor(9);
  SetTextStyle(2,VertDir,4);
  With Block do
  Begin
    OutTextXY(StartX+10,Round(StartY+65+CenterY-TextWidth(YLable+'('+UnitY+')')/2),
          YLable+'('+UnitY+')');
    SetTextStyle(2,0,4);
    OutTextXY(Round(StartX+90+CenterX-TextWidth(XLable+'('+UnitX+')')/2),
          EndY-35,XLable+'('+UnitX+')');
  End;
End;

Procedure DrawFrame(Block : BlockType; x1,y1,x2,y2 : Word;
                    Var ChTemp : Char);
Var StrTemp : Str80;
Begin
  OpenWin(Block,Pic1,Pic2,9,15,15);
  SetFillStyle(1,7);
  Bar(x1+4,y1+20,x2-4,y1+34+TextHeight('S'));
  {Write XLable Border}
```

```
  StrTemp:=XLable;
  Line3D(x1+14,y1+23,'',0,1,TextHeight('S')+8,TextWidth(StrTemp),15,
        0,7,0,0,1,5,1);
  OutTextXY(x1+14,y1+25,StrTemp);
  {Adjust Width of RealX Value}
  Line3D(x1+14+TextWidth(StrTemp)+5,y1+23,'',0,1,TextHeight('S')+8,
        PixelBlank(ABlankX),0,15,15,0,0,1,5,1);
  {Write XUnit Border}
  Line3D(x1+14+TextWidth(StrTemp)+PixelBlank(ABlankX)+10,y1+23,
        UnitX,0,0,TextHeight('S')+8,0,15,0,7,0,0,1,0,2);
  {Write YLable Border}
  StrTemp:=YLable;
  Line3D(x2-TextWidth(UnitY)-15-PixelBlank(ABlankX)-5-10-TextWidth(StrTemp),
        y1+23,'',0,1,TextHeight('S')+8,TextWidth(StrTemp)+10,
        15,0,7,0,0,1,5,1);
  OutTextXY(x2-TextWidth(UnitY)-15-PixelBlank(ABlankX)-10-TextWidth(StrTemp),
          y1+25,StrTemp);
  {Adjust Width of RealY Value}
  Line3D(x2-TextWidth(UnitY)-15-PixelBlank(ABlankX),y1+23,'',0,1,
        TextHeight('S')+8,PixelBlank(ABlankX),0,15,15,0,0,1,5,1);
  {Write YUnit Border}
  Line3D(x2-TextWidth(UnitY)-10,y1+23,UnitY,0,0,TextHeight('S')+8,
        0,15,0,7,0,0,1,0,2);

  Line3D(x1+4,y2-20,'',0,1,17,x2-x1-8,15,0,7,0,0,1,5,1);
  OutTextXY(x1+14,y2-17,'Press Esc or Alt-X to "Quit"          Enter-Return to Entry Parameters');
  Rectangle(StartX,StartY,EndX,EndY);          { this space is proper }
  GridLine(True,True);
  ScaleXY(Block);
End;

Procedure Draw(CheckType : Char);
Var i,x1,y1,x2,y2 : Word;
Begin
 For i:=1 to m do
 Begin
   Case UpCase(CheckType) of
     'L' : Begin
           If i<>m then
           Begin
             If (XValue[i]>=MinX) and (XValue[i]<=MaxX) and
                (YValue[i]>=MinY) and (YValue[i]<=MaxY) and
                (XValue[i+1]>=MinX) and (XValue[i+1]<=MaxX) and
                (YValue[i+1]>=MinY) and (YValue[i+1]<=MaxY) then
             Begin
               PositionXY(XValue[i],YValue[i],x1,y1);
               PositionXY(XValue[i+1],YValue[i+1],x2,y2);
```

```
            SetColor(4);
            Line(x1,y1,x2,y2);
          End;
        End;
      End;
    'D' : Begin
        If (XValue[i]>MinX) and (XValue[i]<MaxX) and
          (YValue[i]>MinY) and (YValue[i]<MaxY) then
        Begin
          PositionXY(XValue[i],YValue[i],x1,y1);
          PutPixel(x1,y1,4);
        End;
      End;
  End;
 End;
 SetColor(0);
End;

Begin
 ABlankX:=15; {Adjust Width of XValue and YValue Block}
 {Adjust XValue or YValue Display : Pixel Unit}
 SetX:=15+PixelBlank(ABlankX-13);
 SetY:=20-PixelBlank(ABlankX-13);
 StartX:=x1+90; EndX:=x2-35; StartY:=y1+65; EndY:=y2-65;
 StepX:=(MaxX-MinX)/10; StepY:=(MaxY-MinY)/10;
 RatioX:=(EndX-StartX)/(MaxX-MinX);
 RatioY:=(EndY-StartY)/(MaxY-MinY);
 CenterX:=(EndX-StartX)/2; CenterY:=(EndY-StartY)/2; {Center Distance}
 XConstant:=Round(CenterX); YConstant:=Round(CenterY);
 DrawFrame(Block,x1,y1,x2,y2,ChTemp);
 SetColor(0); Draw(SetDL);
 ReadPosition(x1,y1,x2,y2,ChTemp);
 CloseWin(Block,Pic1,Pic2);
End;

Procedure EntryParameter(Var MinX,MaxX,MinY,MaxY : Real;
                         Var Auto,GraphType,FileName : String;
                         Var PlaneChoice : Char;
                         Var Checkout,Format : Boolean);
Var Block    : BlockType;
   Pic1      : Pointer;
   Plane     : String;
   Str1,Str2 : Str80;
   Dir       : DirStr;
   FName     : NameStr;
   Ext       : ExtStr;
Begin
```

```
Checkout:=False;
With Block do
Begin
  Header:='Entry Parameters for Aperture Field Distribution';
  StartX:=50; StartY:=100; EndX:=550; EndY:=290;
End;
OpenBlock(Block,Pic1,14,0,15); SetColor(9);
Str1:='Enter FileName (*.* : Browse) : '; OutTextXY(80,125,Str1);
MoveTo(80+TextWidth(Str1),125); ReadString(FileName,0,14,15,1,50);
If FileName[1] in [#27,#45] then
Begin
  CloseBlock(Block,Pic1);
  ErrorMessage('Warning','Press Enter to Entry Parameter.',' ',
               'Press Alt-X or Esc to Exit');
End
Else
Begin
  If FileName='*.*' then
  Begin
    CloseBlock(Block,Pic1);
    LoadFile('*.*','Open Aperture Field File',Path,Name,0,0,11,15,4,Ch);
    FileName:=Path+Name;
    OpenBlock(Block,Pic1,14,0,15); SetColor(9);
    Str1:='Enter FileName (*.* to Browse) : '; OutTextXY(80,125,Str1);
    MoveTo(80+TextWidth(Str1),125); ReadString(FileName,0,14,15,1,50);
  End;
  If FileExist(FileName) then
  Begin
    Assign(AFHfile,FileName);
    Reset(AFHfile); Seek(AFHfile,0); Read(AFHfile,AFH);
    Close(AFHfile);
    If AFH.Header = AFformat then
    Begin
      Repeat
        SetColor(3); GraphType:='L';
        Str1:='Type of Graph (L,l : Line, D,d : Dot) : ';
        OutTextXY(80,140,Str1);
        MoveTo(80+TextWidth(Str1),140); ReadString(GraphType,0,14,15,1,1);
      Until Upcase(GraphType[1]) in ['L','D',#27,#45];
      If GraphType[1] in [#27,#45] then
      Begin
        CloseBlock(Block,Pic1);
        ErrorMessage('Warning','Press Enter to Entry Parameter.',' ',
                     'Press Alt-X or Esc to Exit');
        Checkout:=True;
      End
      Else
```

```
Begin
  Repeat
    SetColor(4); Plane:='Y'; Str1:='Select Plane : ( X,Y,Z ) : ';
    OutTextXY(80,155,Str1);
    {SetColor(13);} MoveTo(80+TextWidth(Str1),155);
    {OutText('X,'); SetColor(4);
    OutText('Y,'); SetColor(13);
    OutText('Z'); SetColor(4);  OutText(' ) : ');}
    MoveTo(235,155); ReadString(Plane,0,14,15,1,1);
    PlaneChoice:=UpCase(Plane[1]);
  Until Upcase(PlaneChoice) in ['X','Y','Z',#27,#45];
{ Until (PlaneChoice in ['X','Y','Z']) and (PlaneChoice='Y');}
  If PlaneChoice in [#27,#45] then
  Begin
    CloseBlock(Block,Pic1);
    ErrorMessage('Warning','Press Enter to Entry Parameter.',' ',
                 'Press Alt-X or Esc to Exit');
    Checkout:=True;
  End
  Else
  Begin
    Repeat
      SetColor(6); Auto:='A';
      Str1:='Select Scale of axis (A,a : AutoScale, D,d : Define Scale) : ';
      OutTextXY(80,170,Str1);
      MoveTo(80+TextWidth(Str1),170); ReadString(Auto,0,14,15,1,1);
    Until Upcase(Auto[1]) in ['A','D',#27,#45];
    If Auto[1] in [#27,#45] then
    Begin
      CloseBlock(Block,Pic1);
      ErrorMessage('Warning','Press Enter to Entry Parameter.',' ',
                   'Press Alt-X or Esc to Exit');
      Checkout:=True;
    End
    Else
    Begin
      If UpCase(Auto[1])<>'A' then
      Begin
{       ReSet(fi); Seek(fi,0); Read(fi,AF); Close(fi);}
        SetColor(6);
        MinX:=-AFH.Dia/2;
        Str1:='Minimum of x-axis : '; OutTextXY(120,185,Str1);
        MoveTo(120+TextWidth(Str1),185);
        ReadRealNum(MinX,0,14,15,1,30);
        SetColor(6);
        MaxX:=-AFH.Dia/2+AFH.SizeR*AFH.StepR;
        Str1:='Maximum of x-axis : '; OutTextXY(120,200,Str1);
```

```
            MoveTo(120+TextWidth(Str1),200);
            ReadRealNum(MaxX,0,14,15,1,30);
            SetColor(6);
            MinY:=-40;
            Str1:='Minimum of y-axis : '; OutTextXY(120,215,Str1);
            MoveTo(120+TextWidth(Str1),215);
            ReadRealNum(MinY,0,14,15,1,30);
            SetColor(6);
            MaxY:=1;
            Str1:='Maximum of y-axis : '; OutTextXY(120,230,Str1);
            MoveTo(120+TextWidth(Str1),230);
            ReadRealNum(MaxY,0,14,15,1,30);
          End;
          CloseBlock(Block,Pic1);
        End;
      End;
    End;
    Format:=True;
  End
  Else
  Begin
    CloseBlock(Block,Pic1);
    FSplit(FileName,Dir,FName,Ext);
    ErrorMessage('Warning','File '+FName+Ext+','+' format file is incorrect',
                 'Cannot open it. Press Enter to Entry Parameter.',
                 'Press Alt-X or Esc to Exit');
    Format:=False;
  End;
 End;
End;

Procedure MaxMinValue(Value : Real; Var Max,Min : Real);
Begin
 If Value>Max then Max:=Value;
 If Value<Min then Min:=Value;
End;

Procedure SetV2(Start,Step : Real; n : Word; PlaneChoice : Char;
          Var MinX,MaxX,MinY,MaxY : Real; Var XValue,YValue : VecArr);
Var  i : Word;
   Ch : Char;
   Temp : Real;
Begin
 MaxX:=-1e9; MaxY:=-1e9; MinX:=1e9; MinY:=1e9;
 For i:=1 to n do
 Begin
```

```
    XValue[i]:=Start+(i-1)*Step;
    MaxMinValue(XValue[i],MaxX,MinX);
    Case PlaneChoice Of
      'X' : AbsC(AF.Eax[i],YValue[i]); {zero value will cause program error}
      'Y' : AbsC(AF.Eay[i],YValue[i]);
      'Z' : AbsC(AF.Eaz[i],YValue[i]);
    End;
    {If Magnitude of Aperture is very small value we will define a little constant to it.
     Then send this value to message box "Very small value"}
    If Abs(YValue[i])<=0.1 then YValue[i]:=1;
    { Critical Value : Log(0)=Undefine ,so set to Zero }
    YValue[i]:=20*Log10(YValue[i]); { Convert to dB value }
    MaxMinValue(YValue[i],MaxY,MinY);
  End;
  {Normalize Aperture Field}
  For i:=1 to n do YValue[i]:=YValue[i]-MaxY;
  MinY:=MinY-MaxY; MaxY:=MaxY-MaxY;
End;

Begin { Main }
  OpenGraph;
  ChangeColor(2,48);  ChangeColor(14,79); ChangeColor(11,46);
  ChangeColor(10,126); ChangeColor(13,43); ChangeColor(12,69);
  ChangeColor(9,116);  ChangeColor(5,42);  ChangeColor(6,84);
  ChangeColor(3,49);
  Repeat
    EntryParameter(MinX1,MaxX1,MinY1,MaxY1,Auto,GraphType,FileName,
                   PlaneChoice,Checkout,Format);
    If Checkout=False then
      If FileExist(FileName) and Format then
      Begin
        Assign(AFHfile,FileName);
        Reset(AFHfile); Seek(AFHfile,0); Read(AFHfile,AFH);
        Close(AFHfile);
        Assign(AFfile,FileName);
        Reset(AFfile);
        Seek(AFfile,Round(AFH.SizeF/2)+1); Read(AFfile,AF); { See at the middle }
        Close(AFfile);
        InitXYVector;
        n:=AFH.SizeR+1; D:=AFH.Dia; Step:=AFH.StepR;
        Start:=-D/2;
        SetV2(Start,Step,n,PlaneChoice,MinX,MaxX,MinY,MaxY,XValue,YValue);
        If UpCase(Auto[1])<>'A' then
        Begin
          MinX:=MinX1; MaxX:=MaxX1; MinY:=MinY1; MaxY:=MaxY1;
        End;
        Case PlaneChoice Of
```

```
        'X' : Begin XLable:=' x '; YLable:=' Relative Eax '; End;
        'Y' : Begin XLable:=' y '; YLable:=' Relative Eay '; End;
        'Z' : Begin XLable:=' z '; YLable:=' Relative Eaz '; End;
      End;
      UnitX:='  m  '; UnitY:=' dB  ';
      x1:=50; y1:=40; x2:=580; y2:=440;
      With Block do
      Begin
        Header:='Aperture Field Distribution';
        StartX:=x1; StartY:=y1; EndX:=x2; EndY:=y2;
      End;
      SetDL:=GraphType[1];
      Design(XValue,YValue,n,MinX,MaxX,MinY,MaxY,UnitX,UnitY,XLable,YLable,
           Block,x1,y1,x2,y2,SetDL,Ch);
    End
    Else
    If (FileName[1]<>#27) and (FileName[1]<>#45) and Format then
      ErrorMessage('Error','File Not Found.',
                   'Press Enter to Entry Parameter.',
                   'Press Alt-X or Esc to Exit');
  Until Ch in [#27,#45];
  CloseGraph;
End.
```

```
{Update 28 Nov 1996}
Program NearFieldToFarField;
Uses Crt,Dos,Graph,UseGraph,KeyGraph,Menu1,Math,VectPlex,Open3D;
Const Max1 = 600;
    AFformat = 'APERTURE';
Type
  CArr1   = Array[1..Max1] Of Complex;
  AFHType = Record
            Header : String[8];
            DtoLamda,Frequency,F_D,Dia,StepR,StepF : Real;
            SizeR,SizeF : Word;
            FeedType : String;
           End;
  AFType  = Record
            Eax,Eay,Eaz : CArr1;
           End;
  FFHType = Record
            Header : String[8];
            DtoLamda,Frequency,F_D,Dia,PhiF,ZetaStart,StepA : Real;
            SizeA : Word;
            FeedType : String;
           End;
  FFType  = Record
            Er,Ezeta,Ephi,Eco,Ecx : Complex;
           End;
Var
  Window : WinType;
  AFH : AFHType;
  AF : AFType;
  AFHfile : File Of AFHType;
  AFfile : File Of AFType;
  FFH : FFHType;
  FF : FFType;
  FFHfile : File Of FFHType;
  FFfile : File Of FFType;
  Path,Name,FileName,FeedT : String;
  Freq,Lamda,D,r,Beta,FToD,f : Real;
  Phi,ThetaStart,ThetaStop,dzeta : Real;
  qe,qh : Integer;
  Size : Word;
  fx,fy : Complex;
  NoError,Quit,Format : Boolean;
  Ch : Char;

Procedure ErrorMessage(HText,Text1,Text2,Text3 : String);
Var Block : BlockType;
    Menu : MenuType;
```

```
    Pic1 : Pointer;
    No : Byte;
    ChTemp : Char;
Begin
  No:=1;
  With Block do
  Begin
    Header:=HText;
    StartX:=135; EndX:=520; StartY:=150; EndY:=270;
    Menu[1]:=' Ok ';
    OpenBlock(Block,Pic1,9,15,15);
    SetColor(0);
    OutTextXY(StartX+20,StartY+30,Text1);
    OutTextXY(StartX+20,StartY+45,Text2);
    Line3D(StartX+3,EndY-20,'',0,1,17,380,15,0,7,0,0,1,8,3);
    SetColor(1);
    OutTextXY(StartX+10,EndY-18,Text3);
    HSelectMenu(Menu,1,StartX+165,EndY-50,0,7,15,0,25,15,12,No,ChTemp);
    Repeat
      SelectMenuH(Menu,1,StartX+165,EndY-50,0,7,15,0,25,15,12,No,ChTemp);
    Until ChTemp in [#13,#27,#45];
    Ch:=ChTemp;
  End;
  CloseBlock(Block,Pic1);
End;

Procedure CheckQuit(Var ChTemp : Char);
Var Block  : BlockType;
    Menu   : MenuType;
    No     : Byte;
    Pic1   : Pointer;
Begin
  With Block do
  Begin
    Header:='Exit Parabolic Reflector Antenna';
    StartX:=100; StartY:=100; EndX:=530; EndY:=210;
    No:=1; Menu[1]:='Ok'; Menu[2]:='Cancel';
    OpenBlock(Block,Pic1,3,15,15);
    SetColor(4); SetTextStyle(0,0,3); OutTextXY(StartX+33,134,'!');
    CirCle(StartX+42,145,15);
    SetColor(0); SetTextStyle(2,0,4);
    OutTextXY(StartX+70,142,'This will end your Parabolic Reflector Antenna session');
  End;
  HSelectMenu(Menu,2,235,170,0,7,15,0,60,20,12,No,ChTemp);
  Repeat
    SelectMenuH(Menu,2,235,170,0,7,15,0,60,20,12,No,ChTemp);
  Until ChTemp in [#13,#27];
```

```
  If ChTemp=#13 then
  Begin
    Case No Of
      1 : Quit:=True;
    End;
  End;
  CloseBlock(Block,Pic1);
  SetTextStyle(2,0,4);
End;

Procedure ChangeDir(Window : WinType);
Var Block  : BlockType;
    ChTemp : Char;
    Pic1   : Pointer;
    StartText,EndText,Len,Len1 : Word;
    Path,PathTemp : String;
    Result,StepCornerX,StepCornerY,WidthX,WidthY : Byte;
Begin
  StepCornerX:=25; StepCornerY:=20; WidthX:=4; WidthY:=3;
  GetDir(0,Path);
  With Block do
  Begin
    Header:='Change Directory';
    StartX:=20; StartY:=55; EndX:=617; EndY:=156;
  End;
  Repeat
    OpenBlock(Block,Pic1,3,15,15);
    Line3D(24,76,'',0,1,77,589,15,0,7,0,0,1,8,3);
    Line3D(26,82,'',0,1,21,585,0,15,13,0,0,1,8,3);
    Line3D(27,108,'',0,1,20,287,0,15,15,0,0,1,8,3);
    Line3D(320,108,'',0,1,20,289,0,15,15,0,0,1,8,3);
    Line3D(27,132,'Enter or Esc-Quit',0,1,15,583,0,15,7,0,0,1,8,2);
    OutTextXY(35,87,'CHANGE TO DIRECTORY :) ');
    MoveTo(35+TextWidth('CHANGE TO DIRECTORY :) ')+10,87);
    ReadString(Path,15,9,13,1,40);
    If Path=#27 then GetDir(0,Path);
    {$I-} ChDir(Path); {$I+}
    If IOResult<>0 then
    Begin
      SetColor(4);
      OutTextXY(65,113,'' DIRECTORY NOT FOUND '');
      Line3D(27,132,'Esc-Quit Enter-Continue',0,1,15,583,0,15,7,0,0,1,8,2);
      Result:=1;
      ChTemp:=ReadKey;
    End
    Else Result:=0;
    CloseBlock(Block,Pic1);
```

```
  Until ((Result=0) or (ChTemp=#27));
  GetDir(0,Path);
  PathTemp:='Current Directory :) '+Path;
  UpCaseStr(PathTemp);
  With Window do
  Begin
    StartText:=StartX+WidthX; EndText:=EndX-WidthX;
    Len:=TextWidth(PathTemp); Len:=(EndText-StartText-Len) div 2;
    Len1:=TextWidth(PathTemp)+StartText;
    If (Len<0) or (Len1>EndText) then
    Begin
      While Len1>EndText do
      Begin
        Len1:=TextWidth(PathTemp)+StartText;
        Delete(Path,Length(PathTemp),1);
      End;
      Line3D(StartX+WidthX,EndY-WidthY-StepCornerY+6,PathTemp,0,0,
           StepCornerY-6,EndX-StartX-2*WidthX,15,0,7,0,1,1,8,2);
    End
    Else
    Begin
      Line3D(StartX+WidthX,EndY-WidthY-StepCornerY+6,PathTemp,0,1,
           StepCornerY-6,EndX-StartX-2*WidthX,15,0,7,0,1,1,8,2);
    End;
  End;
End;

Procedure MemInfo;
Var Block  : BlockType;
    Regs   : Registers;
    MemStr : String;
    DosMem : Word;
    Pic1   : Pointer;
Begin
  With Block do
  Begin
    Header:='Memory Information';
    StartX:=20; StartY:=50; EndX:=617; EndY:=163;
  End;
  Intr($12,Regs); DosMem:=Regs.AX;
  OpenBlock(Block,Pic1,6,15,5);
  Line3D(24,70,'',0,1,90,589,15,0,7,0,0,1,8,3);
  Line3D(27,73,'',0,1,13,582,15,0,3,0,0,1,8,3);
  Line3D(27,89,'',0,1,29,582,0,15,15,0,0,1,8,3);
  Line3D(27,121,'',0,1,13,582,15,0,3,0,0,1,8,3);
  Line3D(27,137,'',0,1,17,582,0,15,15,0,0,1,8,3);
  SetColor(15);
```

```
  OutTextXY(35,74,'Memory :) ');
  OutTextXY(35,122,'Installed Memory :) ');
  SetColor(0);
  Str(MaxAvail,MemStr);
  OutTextXY(38,93,'Maximum Contineous Memory : '+MemStr+' Bytes');
  Str(MemAvail,MemStr);
  OutTextXY(38,105,'Memory Available : '+MemStr+' Bytes');
  Str(DosMem,MemStr);
  OutTextXY(35,141,'Base : '+MemStr+' KBytes');
  OutTextXY(285,141,'Extended : ');
  Repeat Until KeyPressed;
  CloseBlock(Block,Pic1);
End;

Procedure InitFFH(Var FFH : FFHType);
Begin
  With FFH do
  Begin
    Header:='FARFIELD';
    DtoLamda:=D/Lamda; Frequency:=Freq; F_D:=FtoD; Dia:=D; PhiF:=Phi;
    StepA:=dzeta; SizeA:=Size; ZetaStart:=ThetaStart; FeedType:=FeedT;
  End;
End;

Procedure InitFF(Var FF : FFType);
Begin
  With FF do
  Begin
    ZeroC(Er,Er); ZeroC(Ezeta,Ezeta); ZeroC(Ephi,Ephi);
    ZeroC(Eco,Eco); ZeroC(Ecx,Ecx);
  End;
End;

Function FileExist(FName : String) : Boolean;
Var fi : Text;
Begin
  {$I-} Assign(fi,FName); Reset(fi); Close(fi); {$I+}
  FileExist:=(IOResult=0) and (FName<>'');
End;

Procedure EntryAFFile(Var FileName : String; Var Format : Boolean);
Var Block : BlockType;
    Pic1  : Pointer;
    Str1  : Str80;
    Dir   : DirStr;
    FName  : NameStr;
    Ext   : ExtStr;
```

```
    StartX,StartY,EndX,EndY : Word;
Begin
  With Block do
  Begin
    Header:='Entry Aperture Field File to Calculate Far Field Pattern ';
    StartX:=100; StartY:=100; EndX:=530; EndY:=170;
    OpenBlock(Block,Pic1,14,0,15); SetColor(9);
    Str1:='Enter FileName (*.* to Browse) : ';
    OutTextXY(StartX+20,StartY+35,Str1);
    MoveTo(StartX+20+TextWidth(Str1),StartY+35);
    ReadString(FileName,0,14,15,1,50);
  End;
  If FileName[1] in [#27,#45] then
  Begin
    CloseBlock(Block,Pic1);
    ErrorMessage('Warning','Press Enter to Entry Aperture Field File.',' ',
                 'Press Alt-X or Esc to Exit');
    Format:=False;
  End
  Else
  Begin
    If FileName='*.*' then
      Begin
        CloseBlock(Block,Pic1);
        LoadFile('*.*','Open Aperture Field File',Path,Name,0,0,11,15,4,Ch);
        FileName:=Path+Name;
        StartX:=100; StartY:=100; EndX:=530; EndY:=170;
        OpenBlock(Block,Pic1,14,0,15); SetColor(9);
        Str1:='Enter FileName (*.* to Browse) : ';
        OutTextXY(StartX+20,StartY+35,Str1);
        MoveTo(StartX+20+TextWidth(Str1),StartY+35);
        ReadString(FileName,0,14,15,1,50);
      End;
      If FileExist(FileName) then
      Begin
        Assign(AFHfile,FileName);
        ReSet(AFHfile); Seek(AFHfile,0); Read(AFHfile,AFH); Close(AFHfile);
        If AFH.Header=AFformat then
        Begin
          CloseBlock(Block,Pic1);
          Format:=True;
        End
        Else
        Begin
          CloseBlock(Block,Pic1);
          FSplit(FileName,Dir,FName,Ext);
          ErrorMessage('Warning','File '+FName+Ext+','+' format file is incorrect.',
```

```
                'Cannot open it. Press Enter to Entry Parameter.',
                'Press Alt-X or Esc to Exit');
        Format:=False;
      End; {End Of Else Of Check Format File}
    End
    Else
    Begin
      CloseBlock(Block,Pic1);
      ErrorMessage('Error','File Not Found.','Press Enter to Entry Parameter.',
              'Press Alt-X or Esc to Exit');
      Format:=False;
    End; {End Of FileExist}
  End; {End Of Else of Check FileName}
End;

Procedure SaveFFtoDisk(Var FileName : String; Var Save,NewFile : Boolean);
Var Block : BlockType;
    Menu  : MenuType;
    Pic1  : Pointer;
    Dir   : DirStr;
    Name  : NameStr;
    Ext   : ExtStr;
    No    : Byte;
    Str1  : Str80;
    ChTemp : Char;
Begin
  Repeat
    No:=1;
    With Block do
    Begin
      Header:='Save Far Field to Disk';
      StartX:=90; StartY:=100; EndX:=540; EndY:=160;
      OpenBlock(Block,Pic1,14,0,15); SetColor(9);
      FileName:='';
      Str1:='Save Far Field to FileName : '; OutTextXY(StartX+25,130,Str1);
      MoveTo(StartX+25+TextWidth(Str1),130); ReadString(FileName,0,14,15,1,50);
    End;
    FSplit(FileName,Dir,Name,Ext);
{   If Ext='' then
    Begin
      Ext:='.ffp';
      Filename:=Filename+Ext
    End;}
    CloseBlock(Block,Pic1);
    If FileExist(FileName) then
    Begin
      With Block do
```

```
      Begin
        Header:='Warning';
        StartX:=160; StartY:=100; EndX:=460; EndY:=200;
        OpenBlock(Block,Pic1,9,15,15);
        Menu[1]:=' Yes '; Menu[2]:=' No ';
        SetColor(0);
        OutTextXY(StartX+25,StartY+30,'File '+Name+Ext+' already exists. Overwrite?');
        HSelectMenu(Menu,2,StartX+80,EndY-45,0,7,15,0,40,15,12,No,ChTemp);
        Repeat
          SelectMenuH(Menu,2,StartX+80,EndY-45,0,7,15,0,40,15,12,No,ChTemp);
        Until ChTemp in [#13,#27,#45];
      End;
      If ChTemp=#13 then
      Begin
        Case No Of
          1 : Begin Save:=True;  NewFile:=False; End;
          2 : Begin Save:=False; NewFile:=False; End;
        End;
      End;
      CloseBlock(Block,Pic1);
    End
    Else
    Begin
      If (FileName=#27) or (FileName=#45) or (FileName=#13) then
      Begin
        Save:=False; NewFile:=False;
      End
      Else
      Begin
        Save:=True; NewFile:=True;
      End;
    End;
  Until (FileName=#13) or (FileName=#27) or (FileName=#45) or NewFile or
        (ChTemp in [#13,#27,#45]);
End;

Procedure Expr2(ETemp : Complex; x,y,xx,yy : Real; Var Value : Complex);
Var EkP : Complex;
Begin
  CExpo(-Beta*((Sin(xx)*Cos(yy)*x)+(Sin(xx)*Sin(yy)*y)),EkP);
  MulC(ETemp,EkP,Value);
  {Value is Etemp*exp(j*Beta*(sin(xx)*cos(yy)*x+sin(xx)*sin(yy)*y));}
End;

{Integrated by Simpson Algorithm}
Procedure Sim2(Choose : Byte; a,b,c,d : Real; n,m : Word; xx,yy : Real;
               Var Area : Complex);
```

```
Var xi,yj      : Word;
    h,k,x,y    : Real;
    Y1,Y2,Y3,X1,X2,X3,Q,L : Complex;
    CTemp,CTemp1,CTemp2 : Complex;
    Ch : Char;
Begin
  h:=(b-a)/(2*n);
  ZeroC(Y1,Y1); ZeroC(Y2,Y2); ZeroC(Y3,Y3);
  For xi:=0 to 2*n do
  Begin
    x:=a+xi*h;
    k:=(d-c)/(2*m);
    Begin
      ZeroC(X1,X1); ZeroC(X2,X2); ZeroC(X3,X3);
      {Shift Index because Zero Index used to keep Header}
      Seek(AFfile,xi+1); Read(AFfile,AF);
      Case Choose Of
        1 : Begin
              Expr2(AF.Eax[1],x,c,xx,yy,CTemp1);
                Expr2(AF.Eax[2*m+1],x,d,xx,yy,CTemp2);
              End;
        2 : Begin
              Expr2(AF.Eay[1],x,c,xx,yy,CTemp1);
              Expr2(AF.Eay[2*m+1],x,d,xx,yy,CTemp2);
            End;
        3 : Begin
              Expr2(AF.Eaz[1],x,c,xx,yy,CTemp1);
              Expr2(AF.Eaz[2*m+1],x,d,xx,yy,CTemp2);
            End;
      End;
      PlusC(CTemp1,CTemp2,X1);
      For yj:=1 to 2*m-1 do
      Begin
        y:=c+yj*k;
          Case Choose of
            1 : Expr2(AF.Eax[yj+1],x,y,xx,yy,Q);
            2 : Expr2(AF.Eay[yj+1],x,y,xx,yy,Q);
            3 : Expr2(AF.Eaz[yj+1],x,y,xx,yy,Q);
          End;
          If Odd(yj) then PlusC(X3,Q,X3) Else PlusC(X2,Q,X2);
      End;
    End;
    MulRC(2,X2,CTemp1); MulRC(4,X3,CTemp2);
    PlusC3(X1,CTemp1,CTemp2,CTemp);
    MulRC(k/3,CTemp,L); {Summation L=(X1+2*X2+4*X3)*k/3;}

    If (xi=0) or (xi=(2*n)) then PlusC(Y1,L,Y1)
```

```
    Else If Odd(xi) then PlusC(Y3,L,Y3)
        Else PlusC(Y2,L,Y2);
  End;
  MulRC(2,Y2,CTemp1); MulRC(4,Y3,CTemp2);
  PlusC3(Y1,CTemp1,CTemp2,CTemp);
  MulRC(h/3,CTemp,Area);  {Area=(Y0+2*Y2+4*Y1)*h/3;}
End;

{Find Far Field Pattern from Aperture Field Distribution by Plane Wave Spectrum Method}
Procedure FarField(PhiR,ZetaStrtR,ZetaStopR,dzetaR : Real);
Var ZetaR,ZetaFinal,rp : Real; {Phi,Zzeta in radian}
    k : Word;
    Ch : Char;
    Ekrp,Factor,CTemp,CTemp1,CTemp2 : Complex;
    Block : BlockType;
    Pic1 : Pointer;
    Save,NewFile : Boolean;
    FileName : String;
    Str1,Str2 : Str80;
Begin
  SaveFFtoDisk(FileName,Save,NewFile);
  If (Save or NewFile) and ((FileName<>#27) and (FileName<>#45)) and
     (FileName<>'') then
  Begin
    If NewFile then
    Begin
      Assign(FFHfile,FileName);
      ReWrite(FFHfile); Write(FFHfile,FFH); Close(FFHfile);
      Assign(FFfile,FileName); ReSet(FFfile);
    End
    Else
    Begin
      Assign(FFHfile,FileName);
      ReSet(FFHfile); Seek(FFHfile,0); Write(FFHfile,FFH); Close(FFHfile);
      Assign(FFfile,FileName); ReSet(FFfile);
    End;
    k:=1; ZetaFinal:=ZetaStrtR+Size*dzetaR; ZetaR:=ZetaStrtR;
    rp:=200*D*D/Lamda; {Set Distance of Observer at Far Field must be > 2D^2/Lamda}
    With Block do
    Begin
      Header:='Calculating Far Field and Save to Disk';
      StartX:=120; StartY:=100; EndX:=500; EndY:=180;
      OpenBlock(Block,Pic1,14,0,15); SetColor(9);
      Str1:='Calculating Far Field  '; OutTextXY(StartX+80,140,Str1);
    End;
    Repeat
      InitFF(FF);
```

```
      SetColor(0);
{To check: Str(zetaR:3,StrTemp); OutTextXY(200,200,StrTemp); Ch:=Readkey;}
      Sim2(1,-r,r,-r,r,Trunc(AFH.SizeF/2),Trunc(AFH.SizeR/2),ZetaR,PhiR,fx);
      Sim2(2,-r,r,-r,r,Trunc(AFH.SizeF/2),Trunc(AFH.SizeR/2),ZetaR,PhiR,fy);
      CTemp.Re:=0; CTemp.Im:=1; {0+1j}
      CExpo(Beta*rp,Ekrp);
      MulRC(Beta/(2*Pi*rp),CTemp,CTemp); MulC(CTemp,Ekrp,Factor);
      {Factor of Efarfield:=j*Beta*exp(-j*Beta*rp)/(2*Pi*rp);}

      ZeroC(FF.Er,FF.Er);
      {Er=0}

      MulRC(Cos(PhiR),fx,CTemp1); MulRC(Sin(PhiR),fy,CTemp2);
      PlusC(CTemp1,CTemp2,CTemp);
      MulC(Factor,CTemp,FF.Ezeta);
      {EfZeta[k]:=Factor*(fx*Cos(PhiR)+fy*Sin(PhiR));}

      MulRC(Cos(PhiR),fy,CTemp1); MulRC(-Sin(PhiR),fx,CTemp2);
      PlusC(CTemp1,CTemp2,CTemp); MulRC(Cos(ZetaR),CTemp,CTemp);
      MulC(Factor,CTemp,FF.Ephi);
      {EfPhi[k]:=Factor*(fy*Cos(PhiR)-fx*Sin(PhiR))*Cos(ZetaR);}

      MulRC(Sin(PhiR),FF.Ezeta,CTemp1);
      MulRC(Cos(PhiR),FF.Ephi,CTemp2);
      PlusC(CTemp1,CTemp2,FF.Eco);
      {Co-Polarize : Eco[k]:=Efzeta[k]*Sin(PhiR)+Efphi[k]*Cos(PhiR);}

      MulRC(Cos(PhiR),FF.Ezeta,CTemp1);
      MulRC(-Sin(PhiR),FF.Ezeta,CTemp2);
      PlusC(CTemp1,CTemp2,FF.Ecx);
      {Cross-Polarize : Ecx[k]:=Efzeta[k]*Cos(PhiR)-Efphi[k]*Sin(PhiR);}

      Seek(FFfile,k); Write(FFfile,FF);
      k:=k+1; ZetaR:=ZetaStrtR+(k-1)*dzetaR;
      SetColor(15);
      Bar(Block.StartX+80+TextWidth(Str1),140,Block.StartX+80+TextWidth(Str1)+50,155);
      SetColor(0);
      Str((k-1)/(Size+1)*100:5:2,Str2);
      MoveTo(Block.StartX+80+TextWidth(Str1),140); OutText(Str2+' %');

    Until ZetaR>ZetaFinal;
    Close(FFfile); Close(AFfile);
    Repeat Until KeyPressed;
    CloseBlock(Block,Pic1);
  End;
End;
```

```
{Input Coordinate of Far Field}
Procedure EntryCoordinate;
Var Block : BlockType;
    Pic1  : Pointer;
    Str1  : Str80;
Begin
  With Block do
  Begin
    Header:='Entry Coordinate at Far Field';
    StartX:=105; StartY:=100; EndX:=525; EndY:=195;
  End;
  OpenBlock(Block,Pic1,14,0,15);
  Repeat
    SetColor(9); Str1:='Plane of Observation ( [0,360] : degrees) : ';
    OutTextXY(120,125,Str1);
    MoveTo(120+TextWidth(Str1),125); ReadRealNum(Phi,0.14,15,1,30);
    If Phi=360 then Phi:=0;
  Until (Phi>=0) and (Phi<=360);
  Repeat
    SetColor(9);
    Str1:='Start Angle ( [-90,90] : degrees) : '; OutTextXY(120,140,Str1);
    MoveTo(120+TextWidth(Str1),140); ReadRealNum(ThetaStart,0,14,15,1,30);
  Until (ThetaStart>=-90) and (ThetaStart<=90);
  Repeat
    SetColor(9);
    Str1:='Stop Angle ( :degrees) : '; OutTextXY(120,155,Str1);
    MoveTo(120+TextWidth(Str1),155); ReadRealNum(ThetaStop,0,14,15,1,30);
  Until (ThetaStop>ThetaStart) and (ThetaStop<=90);
  SetColor(9);
  Str1:='Step Angle (degrees) : '; OutTextXY(120,170,Str1);
  MoveTo(120+TextWidth(Str1),170); ReadRealNum(dzeta,0,14,15,1,30);
  If (ThetaStop-ThetaStart)<dzeta then
  Begin
    CloseBlock(Block,Pic1);
    ErrorMessage('Error','Step angle greater than inteval angle.',
                 ' ','Press Esc or Alt-X to Exit');
    NoError:=False;
  End
  Else
  Begin
    If dzeta<=0 then
    Begin
      CloseBlock(Block,Pic1);
      ErrorMessage('Error','Step Angle Error.',
                   ' ','Press Esc or Alt-X tp Exit');
      NoError:=False;
    End
```

```
    Else
    Begin
      NoError:=True;
      CloseBlock(Block,Pic1);
    End;
  End;
End;

Procedure WinH;
Begin
  With Window do
  Begin
    MaxCol:=2;
    MenuH[1].Menu:=' = ';
    MenuH[2].Menu:=' Open File & Calculate ';
  End;
End;

Procedure WinV;
Begin
  With Window do
  Begin
    MenuH[1].MaxRow:=3;
    MenuH[1].MenuV[1]:=' Change Dir  ';
    MenuH[1].MenuV[2]:=' Memory Information  ';
    MenuH[1].MenuV[3]:=' Exit    Alt-X  ';
  End;
End;

Begin {Main}
  OpenGraph;
  ChangeColor(2,50);  ChangeColor(14,79); ChangeColor(11,46);
  ChangeColor(10,126); ChangeColor(13,43); ChangeColor(12,69);
  ChangeColor(9,116);  ChangeColor(5,42);  ChangeColor(6,84);
  ChangeColor(3,49);
  InitialWin(Window); WinH; WinV; Quit:=False;
  InputValueToWin(Window,0,0,GetMaxX,GetMaxY,
              'Calculate Far Field of Symmetrical Parabolic Reflector Antenna');
  LoadWin(Window,9,15,1,14);
  MenuHor(Window,0,15,15,3,1);
  Repeat
    HorMenu(Window,0,15,15,3,1,Ch);
    With Window do
    Begin
      If (Ch=#13) then
      Begin
        If MenuH[Choice].MenuV[1]<>'' then
```

```
        Begin
          Repeat
            VerMenu(Window,0,15,15,3,1,0,15,15,3,1,Ch);
          Until Ch in [#13,#27,#45];
        End;
        If Ch=#13 then
        Begin
          Case Choice Of
            1 : Begin
                Case MenuH[Choice].ChoiceSub of
                    1 : ChangeDir(Window);
                    2 : MemInfo;
                    3 : CheckQuit(Ch);
                  End;
              End; {Of Choice=1}
            2 : Begin
{               LoadFile('*.*','Open Aperture File',Path,Name,0,0,11,15,4,Ch);
              FileName:=Path+Name;}
                EntryAFFile(FileName,Format);
                  If FileExist(FileName) and Format then
                  Begin
                    EntryCoordinate;
                    If NoError Then
                    Begin
                EntryAFHfile(FileName); ReSet(AFHfile);
                Seek(AFHfile,0); Read(AFHfile,AFH); Close(AFHfile);
                    Freq:=AFH.Frequency; Lamda:=3*Power(10,8)/Freq;
                    FToD:=AFH.F_D; D:=AFH.Dia; FeedT:=AFH.FeedType;
                    r:=D/2; Beta:=2*Pi/Lamda; f:=FToD*D;
                    Size:=Round((ThetaStop-ThetaStart)/dzeta);
                    If (Size mod 2)<>0 then Size:=Size+1;
                    {Set Number of Interval to Even No.}
                    dzeta:=(ThetaStop-ThetaStart)/Size;
                InitFFH(FFH);
                Assign(AFfile,FileName); ReSet(AFfile);
                    FarField(Phi*Pi/180,ThetaStart*Pi/180,ThetaStop*pi/180,dzeta*pi/180);
                    End;
                  End
              End; {Of Choice=2}
          End; {Of Case}
        End; {Of If VerMenu}
      End; {Of If HorMenu}
    End; {Of With}
    If Ch=#45 then CheckQuit(Ch);
  Until Quit;
  CloseGraph;
End. {Main}
```

```
{Update 4 Dec 1996}
{Plot Far Field From Disk}
Program PFF;
Uses Dos,Crt,Graph,UseGraph,KeyGraph,Math,VectPlex,Menu1,Open3D;
Const MaxV = 600;
      FFformat = 'FARFIELD';
Type
  VecArr  = Array[1..MaxV] of Real;
  FFHType = Record
              Header : String[8];
              DtoLamda,Frequency,F_D,Dia,PhiF,ZetaStart,StepA : Real;
                SizeA : Word;
                FeedType : String;
            End;
  FFType  = Record
                Er,Ezeta,Ephi,Eco,Ecx : Complex;
            End;
Var
  FFH : FFHType;
  FF : FFType;
  FFHfile : File Of FFHType;
  FFfile : File Of FFType;
  XValue,YValue : VecArr;
  Block : BlockType;
  D,Start,Step : Real;
  Auto,FileName,XLable,YLable,GraphType,Path,Name,Plane : String;
  UnitX,UnitY : Str20;
  MinX1,MaxX1,MinY1,MaxY1,MinX,MaxX,MinY,MaxY : Real;
  MaxYco,MinYco : Real;
  i,x1,y1,x2,y2,n : Word;
  Regs : Registers;
  SetDL,Ch : Char;
  PolarE : Char;
  Checkout,Format : Boolean;

Function FileExist(FName : String) : Boolean;
Var fi : Text;
Begin
  {$I-} Assign(fi,FName); Reset(fi); Close(fi); {$I+}
  FileExist:=(IOResult=0) and (FName<>'');
End;

{Utility Of Mouse}
Procedure MouseOn;
Begin
  Regs.Ax:=1; Intr($33,Regs);
End;
```

```
Procedure MouseOff;
Begin
  Regs.Ax:=2; Intr($33,Regs);
End;

Procedure SetMouseXY(x,y : Word);
Begin
  Regs.CX:=x; Regs.DX:=y;
  Regs.AX:=4; Intr($33,Regs);
End;

Procedure SetMouseLR(Left,Right : Word);
Begin
  Regs.CX:=Left; Regs.DX:=Right;
  Regs.AX:=7; Intr($33,Regs);
End;

Procedure SetMouseTB(Top,Bottom : Word);
Begin
  Regs.CX:=Top; Regs.DX:=Bottom;
  Regs.AX:=8; Intr($33,Regs);
End;

Procedure ReadMouseXY(Var x,y : Word);
Begin
  Regs.AX:=4; Intr($33,Regs);
  x:=Regs.CX; y:=Regs.DX;
End;

Procedure MouseBound(Left,Right,Top,Bottom : Word);
Begin
  SetMouseLR(Left,Right); SetMouseTB(Top,Bottom);
  SetMouseXY(Round((Left+Right)/2),Round((Top+Bottom)/2)); MouseOn;
End;

Procedure CheckPress(Var Regs : Registers);
Begin
  Regs.AX:=3; Intr($33,Regs);
End;

Procedure InitXYVector;
Var i : Word;
Begin
  For i:=1 to MaxV do
  Begin
    XValue[i]:=0; YValue[i]:=0;
```

```
  End;
End;

Procedure Design(XValue,YValue : VecArr; m : Word; MinX,MaxX,MinY,MaxY : Real;
                                                          UnitX,UnitY : Str20;
Xlable,YLable : Str80; Block : BlockType;
            x1,y1,x2,y2 : Word; SetDL : Char; Var ChTemp : Char);
Var CenterX,CenterY,RatioX,RatioY : Real;
                  StartX,StartY,EndX,EndY : Word;
   XConstant,YConstant,ABlankX : Word;
   SetX,SetY : Integer;
   Pic1,Pic2   : Pointer;
   StepX,StepY  : Real;

  Procedure PositionXY(XTemp,YTemp : Real; Var x,y : Word);
  Begin
    x:=Round(StartX+(XTemp-MinX)*RatioX);
    y:=Round(EndY-(YTemp-MinY)*RatioY);
  End;

  Procedure FillCh(Var Blank : String; C : Byte; Ch : Char);
  Var i : Byte;
  Begin
    Blank:='';
    For i:=1 to C do Blank:=Blank+Ch;
  End;

  Function PixelBlank(n : Byte) : Word;
  Var Blank : String;
  Begin
    FillCh(Blank,n,' ');
    PixelBlank:=TextWidth(Blank);
  End;

  Procedure ReadPosition(x1,y1,x2,y2 : Word; Var ChTemp : Char);
  Var Str1,Str2 : Str20;
     StrTemp   : Str80;
     XReal,YReal : Real;
     XConst,YConst,x,y,xMs,yMs : Word;
  Begin
    ChTemp:=#0;
    SetViewPort(StartX,StartY,EndX,EndY,ClipOn);
    SetColor(1); SetWriteMode(XORPut);
    Line(0,Round(2*CenterY-YConstant),Round(2*CenterX),
        Round(2*CenterY-YConstant));
    Line(XConstant,0,XConstant,Round(2*CenterY));
    SetWriteMode(COPYPut);
```

```
MouseBound(StartX,EndX,StartY,EndY);
SetMouseXY(StartX+XConstant,StartY+YConstant);
ReadMouseXY(xMs,yMs); {Read old position to Compare with new position}
Repeat
 If (Not KeyPressed) then {Check Condition Between KeyBoard and Mouse}
 Begin
  MouseOn;
  SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
    CheckPress(Regs);
  Case Regs.BX Of
   0 : Begin {No Presse}
       ReadMouseXY(x,y);
       XConst:=x-StartX; YConst:=y-StartY;
       XReal:=MinX+(XConst/RatioX);
       YReal:=MaxY-(YConst/RatioY);
            Str(XReal:10:5,Str1); Str(YReal:10:5,Str2);
       If (x=xMs) and (y=yMs) then
            Begin
             StrTemp:=XLable;
        SetColor(0);
            OutTextXY(x1+14+TextWidth(StrTemp)+SetX,y1+26,''+Str1);
        OutTextXY(x2-TextWidth(UnitY)-SetY-PixelBlank(ABlankX),
                  y1+26,' '+Str2);
       End
       Else
       Begin
        SetWriteMode(COPYPut);
        SetFillStyle(1,15); {SetWriteMode(COPYPut);}
            StrTemp:=XLable;
            Bar(x1+14+TextWidth(StrTemp)+7,y1+24,x1+14+
          TextWidth(StrTemp)+5+PixelBlank(ABlankX),y1+24+
          TextHeight('S')+4);
        SetColor(0);
        {Write XValue}
            OutTextXY(x1+14+TextWidth(StrTemp)+SetX,y1+26,''+Str1);
            StrTemp:=YLable;
        SetFillStyle(1,15); {SetWriteMode(COPYPut);}
            Bar(x2-TextWidth(UnitY)-12-PixelBlank(AblankX),y1+24,
              x2-TextWidth(UnitY)-15,y1+24+TextHeight('S')+4);
        SetColor(0);
        {Write YValue}
        OutTextXY(x2-TextWidth(UnitY)-SetY-PixelBlank(ABlankX),
                  y1+26,' '+Str2);
        ReadMouseXY(xMs,yMs);
       End;
     End;
   1 : ChTemp:=#13;
```

```
    2 : ChTemp:=#27;
   End;
  End
  Else
  Begin
   SetWriteMode(COPYPut);
   MouseOff;
   SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
   XReal:=MinX+(XConstant/RatioX);
   YReal:=MinY+(YConstant/RatioY);
     Str(XReal:10:5,Str1); Str(YReal:10:5,Str2);

     StrTemp:=XLable;
   SetFillStyle(1,15); {SetWriteMode(COPYPut);}
     Bar(x1+14+TextWidth(StrTemp)+7,y1+24,x1+14+TextWidth(StrTemp)+5+
   PixelBlank(ABlankX),y1+24+TextHeight('S')+4);
   SetColor(0);
     OutTextXY(x1+14+TextWidth(StrTemp)+SetX,y1+26,''+Str1);
     StrTemp:=YLable;
   SetFillStyle(1,15); {SetWriteMode(COPYPut);}
   Bar(x2-TextWidth(UnitY)-12-PixelBlank(ABlankX),y1+24,
   x2-TextWidth(UnitY)-15,y1+24+TextHeight('S')+4);
   SetColor(0);
     OutTextXY(x2-TextWidth(UnitY)-SetY-PixelBlank(ABlankX),y1+26,' '+Str2);

   SetViewPort(StartX,StartY,EndX,EndY,ClipOn);
   ChTemp:=ReadKey;
   Case ChTemp Of
    #80 : Begin Dec(YConstant); If YConstant<0 then YConstant:=0; End;
    #72 : Begin
          Inc(YConstant);
          If YConstant>Round(2*CenterY) then YConstant:=Round(2*CenterY);
         End;
    #75 : Begin Dec(XConstant); If XConstant<0 then XConstant:=0; End;
    #77 : Begin
          Inc(XConstant);
          If XConstant>Round(2*CenterX) then XConstant:=Round(2*CenterX);
         End;
   End;
   SetColor(1); SetWriteMode(XORPut);
   Line(0,Round(2*CenterY-YConstant),Round(2*CenterX),
      Round(2*CenterY-YConstant));
   Line(XConstant,0,XConstant,Round(2*CenterY));
  End;
Until ChTemp in [#13,#27,#45];
SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
SetWriteMode(COPYPut);
```

```
  MouseOff;
End;

Procedure GridLine(GridX,GridY : Boolean);
Var x1,y1,x2,y2 : Word;
    i : Byte;
Begin
  If GridX=True then
  Begin
    For i:=1 to 9 do
    Begin
      SetLineStyle(DottedLn,0,1);
      PositionXY(MinX+i*StepX,MinY,x1,y1);
      PositionXY(MinX+i*StepX,MaxY,x2,y2);
      Line(x1,y1,x2,y2);
    End;
  End;
  If GridY=True then
  Begin
    For i:=1 to 9 do
    Begin
      SetLineStyle(DottedLn,0,1);
      PositionXY(MinX,MinY+i*StepY,x1,y1);
         PositionXY(MaxX,MinY+i*StepY,x2,y2);
      Line(x1,y1,x2,y2);
    End;
  End;
  SetLineStyle(0,1,0);
End;

Procedure ScaleXY(Block : BlockType);
Var i : Byte;
    x1,y1,x2,y2 : Word;
    Str1 : Str20;
Begin
  For i:=0 to 10 do
  Begin
    PositionXY(MinX+i*StepX,MinY,x1,y1);
    Str((MinX+i*StepX):5:2,Str1);
    OutTextXY(x1-30,y1+8,Str1);
    PositionXY(MinX,MinY+i*StepY,x1,y1);
    Str((MinY+i*StepY):8:4,Str1);
    SetTextJustify(RightText,CenterText);
    OutTextXY(x1-5,y1-5,Str1);
    SetTextJustify(LeftText,TopText);
  End;
  SetColor(9);
```

```
  SetTextStyle(2,VertDir,4);
  With Block do
  Begin
    OutTextXY(StartX+10,Round(StartY+65+CenterY-TextWidth(YLable+'('+UnitY+')')/2),
          YLable+'('+UnitY+')');
    SetTextStyle(2,0,4);
    OutTextXY(Round(StartX+90+CenterX-TextWidth(XLable+'('+UnitX+')')/2),
          EndY-35,XLable+'('+UnitX+')');
  End;
End;

Procedure DrawFrame(Block : BlockType; x1,y1,x2,y2 : Word;
                Var ChTemp : Char);
Var StrTemp : Str80;
Begin
  OpenWin(Block,Pic1,Pic2,9,15,15);
  SetFillStyle(1,7);
  Bar(x1+4,y1+20,x2-4,y1+34+TextHeight('S'));
  {Write XLable Border}
  StrTemp:=XLable;
  Line3D(x1+14,y1+23,'',0,1,TextHeight('S')+8,TextWidth(StrTemp),15,
       0,7,0,0,1,5,1);
  OutTextXY(x1+14,y1+25,StrTemp);
  {Adjust Width of RealX Value}
  Line3D(x1+14+TextWidth(StrTemp)+5,y1+23,'',0,1,TextHeight('S')+8,
  PixelBlank(ABlankX),0,15,15,0,0,1,5,1);
  {Write XUnit Border}
  Line3D(x1+14+TextWidth(StrTemp)+PixelBlank(ABlankX)+10,y1+23,
       UnitX,0,0,TextHeight('S')+8,0,15,0,7,0,0,1,0,2);
  {Write YLable Border}
  StrTemp:=YLable;
  Line3D(x2-TextWidth(UnitY)-15-PixelBlank(ABlankX)-5-10-TextWidth(StrTemp),
       y1+23,'',0,1,TextHeight('S')+8,TextWidth(StrTemp)+10,
       15,0,7,0,0,1,5,1);
  OutTextXY(x2-TextWidth(UnitY)-15-PixelBlank(ABlankX)-10-TextWidth(StrTemp),
         y1+25,StrTemp);
  {Adjust Width of RealY Value}
  Line3D(x2-TextWidth(UnitY)-15-PixelBlank(ABlankX),y1+23,'',0,1,
  TextHeight('S')+8,PixelBlank(ABlankX),0,15,15,0,0,1,5,1);
  {Write YUnit Border}
  Line3D(x2-TextWidth(UnitY)-10,y1+23,UnitY,0,0,TextHeight('S')+8,
       0,15,0,7,0,0,1,0,2);

  Line3D(x1+4,y2-20,'',0,1,17,x2-x1-8,15,0,7,0,0,1,5,1);
  OutTextXY(x1+14,y2-17,'Esc-Quit  Enter-Return to Entry Parameters');
  Rectangle(StartX,StartY,EndX,EndY);
  GridLine(True,True);
```

```
  ScaleXY(Block);
End;

Procedure Draw(CheckType : Char);
Var i,x1,y1,x2,y2 : Word;
Begin
  For i:=1 to m do
  Begin
    Case UpCase(CheckType) of
      'L' : Begin
              If i<>m then
              Begin
                If (XValue[i]>=MinX) and (XValue[i]<=MaxX) and
                   (YValue[i]>=MinY) and (YValue[i]<=MaxY) and
                   (XValue[i+1]>=MinX) and (XValue[i+1]<=MaxX) and
                   (YValue[i+1]>=MinY) and (YValue[i+1]<=MaxY) then
                Begin
                  PositionXY(XValue[i],YValue[i],x1,y1);
                  PositionXY(XValue[i+1],YValue[i+1],x2,y2);
                  SetColor(4);
                  Line(x1,y1,x2,y2);
                End;
              End;
            End;
      'D' : Begin
              If (XValue[i]>MinX) and (XValue[i]<MaxX) and
                 (YValue[i]>MinY) and (YValue[i]<MaxY) then
              Begin
                PositionXY(XValue[i],YValue[i],x1,y1);
                    PutPixel(x1,y1,4);
              End;
            End;
    End;
  End;
  SetColor(0);
End;

Begin
  ABlankX:=15; {Adjust Width of XValue and YValue Block}
  {Adjust XValue or YValue Display : Pixel Unit}
  SetX:=15+PixelBlank(ABlankX-13);
  SetY:=20-PixelBlank(ABlankX-13);
  StartX:=x1+90; EndX:=x2-35; StartY:=y1+65; EndY:=y2-65;
  StepX:=(MaxX-MinX)/10; StepY:=(MaxY-MinY)/10;
  RatioX:=(EndX-StartX)/(MaxX-MinX);
  RatioY:=(EndY-StartY)/(MaxY-MinY);
  CenterX:=(EndX-StartX)/2; CenterY:=(EndY-StartY)/2;  {Center Distance}
```

```
  XConstant:=Round(CenterX); YConstant:=Round(CenterY);
  DrawFrame(Block,x1,y1,x2,y2,ChTemp);
  SetColor(0); Draw(SetDL);
  ReadPosition(x1,y1,x2,y2,ChTemp);
  CloseWin(Block,Pic1,Pic2);
End;

Procedure ErrorMessage(HText,Text1,Text2,Text3 : String);
Var Block : BlockType;
    Menu : MenuType;
    Pic1 : Pointer;
    No : Byte;
    ChTemp : Char;
Begin
  No:=1;
  With Block do
  Begin
    Header:=HText;
    StartX:=135; EndX:=520; StartY:=150; EndY:=270;
    Menu[1]:=' Ok ';
    OpenBlock(Block,Pic1,9,15,15);
    SetColor(0);
    OutTextXY(StartX+20,StartY+30,Text1);
    OutTextXY(StartX+20,StartY+45,Text2);
    Line3D(StartX+3,EndY-20,'',0,1,17,380,15,0,7,0,0,1,8,3);
    SetColor(1);
    OutTextXY(StartX+10,EndY-18,Text3);
    HSelectMenu(Menu,1,StartX+165,EndY-50,0,7,15,0,25,15,12,No,ChTemp);
    Repeat
      SelectMenuH(Menu,1,StartX+165,EndY-50,0,7,15,0,25,15,12,No,ChTemp);
    Until ChTemp in [#13,#27,#45];
    Ch:=ChTemp;
  End;
  CloseBlock(Block,Pic1);
End;

Procedure EntryParameter(Var MinX,MaxX,MinY,MaxY : Real;
                         Var Auto,GraphType,FileName : String;
                         Var PlaneChoice : Char;
                         Var Checkout,Format : Boolean);
Var Block : BlockType;
    Pic1  : Pointer;
    Str1,Str2 : Str80;
    Dir   : DirStr;
    FName  : NameStr;
    Ext   : ExtStr;
Begin
```

```
Checkout:=False;
With Block do
Begin
  Header:='Entry Parameters for Far Field Pattern';
  StartX:=50; StartY:=100; EndX:=550; EndY:=290;
End;
OpenBlock(Block,Pic1,14,0,15); SetColor(9);
Str1:='Enter FileName (*.* to Browse) : '; OutTextXY(80,125,Str1);
MoveTo(80+TextWidth(Str1),125); ReadString(FileName,0,14,15,1,50);
If FileName[1] in [#27,#45] then
Begin
  CloseBlock(Block,Pic1);
  ErrorMessage('Warning','Press Enter to Entry Parameter.',' ',
               'Press Alt-X or Esc to Exit');
  Format:=False;
End
Else
Begin
  If FileName='*.*' then
    Begin
      CloseBlock(Block,Pic1);
      LoadFile('*.*','Open Far Field File',Path,Name,0,0,11,15,4,Ch);
      FileName:=Path+Name;
      OpenBlock(Block,Pic1,14,0,15); SetColor(9);
      Str1:='Enter FileName (*.* to Browse) : '; OutTextXY(80,125,Str1);
      MoveTo(80+TextWidth(Str1),125); ReadString(FileName,0,14,15,1,50);
    End;
    If FileExist(FileName) then
      Begin
        Assign(FFHfile,FileName);
        ReSet(FFHfile); Seek(FFHfile,0); Read(FFHfile,FFH); Close(FFHfile);
        If FFH.Header = FFformat then
        Begin
          Repeat
          SetColor(3); GraphType:='L';
          Str1:='Type of Graph (L,l : Line, D,d : Dot) : ';
          OutTextXY(80,140,Str1);
          MoveTo(80+TextWidth(Str1),140); ReadString(GraphType,0,14,15,1,1);
        Until UpCase(GraphType[1]) in ['L','D',#27,#45];
        If GraphType[1] in [#27,#45] then
        Begin
          CloseBlock(Block,Pic1);
          ErrorMessage('Warning','Press Enter to Entry Parameter.',' ',
                       'Press Alt-X or Esc to Exit');
          Checkout:=True;
        End
        Else
```

```
Begin
  Repeat
    SetColor(4); Plane:='C';
    Str1:='Select Polorization ('; OutTextXY(80,155,Str1);
    MoveTo(80+TextWidth(Str1),155); OutText('C : Co-Polor');
    OutText(',X : Cross-Polar) : ');
    ReadString(Plane,0,14,15,1,1);
    PlaneChoice:=Upcase(Plane[1]);
  Until PlaneChoice in ['C','X',#27,#45];
  If PlaneChoice in [#27,#45] then
  Begin
    CloseBlock(Block,Pic1);
    ErrorMessage('Warning','Press Enter to Entry Parameter.',' ',
                 'Press Alt-X or Esc to Exit');
    Checkout:=True;
  End
  Else
  Begin
    Repeat
      SetColor(9); Auto:='A';
      Str1:='Select Scale of axis (A,a : AutoScale, D,d : Define Scale) : ';
      OutTextXY(80,170,Str1);
      MoveTo(80+TextWidth(Str1),170); ReadString(Auto,0,14,15,1,1);
    Until UpCase(Auto[1]) in ['A','D',#27,#45];
    If Auto[1] in [#27,#45] then
    Begin
      CloseBlock(Block,Pic1);
      ErrorMessage('Warning','Press Enter to Entry Parameter.',' ',
                   'Press Alt-X or Esc to Exit');
      Checkout:=True;
    End
    Else
    Begin
      If UpCase(Auto[1])<>'A' then
      Begin
        SetColor(9);
        MinX:=FFH.ZetaStart;
        Str1:='Minimum of x-axis : '; OutTextXY(120,200,Str1);
        MoveTo(120+TextWidth(Str1),200);
        ReadRealNum(MinX,0,14,15,1,30);
        SetColor(9);
        MaxX:=FFH.ZetaStart+FFH.SizeA*FFH.StepA;
        Str1:='Maximum of x-axis : '; OutTextXY(120,215,Str1);
        MoveTo(120+TextWidth(Str1),215);
        ReadRealNum(MaxX,0,14,15,1,30);
        SetColor(9);
        MinY:=-100;
```

```
            Str1:='Minimum of y-axis : '; OutTextXY(120,230,Str1);
            MoveTo(120+TextWidth(Str1),230);
            ReadRealNum(MinY,0,14,15,1,30);
            SetColor(9);
            MaxY:=0;
            Str1:='Maximum of y-axis : '; OutTextXY(120,245,Str1);
            MoveTo(120+TextWidth(Str1),245);
            ReadRealNum(MaxY,0,14,15,1,30);
          End;
          CloseBlock(Block,Pic1);
        End;
      End;
    End;
    Format:=True;
  End
  Else
  Begin
    CloseBlock(Block,Pic1);
    FSplit(FileName,Dir,FName,Ext);
    ErrorMessage('Warning','File '+FName+Ext+','+' format file is incorrect.',
                 'Cannot open it. Press Enter to Entry Parameter.',
                 'Press Alt-X or Esc to Exit');
    Format:=False;
  End; {End Of Else Of Check Format File}
 End; {End Of FileExist}
End; {End Of Else of Check FileName}
End;

Procedure MaxMinValue(Value : Real; Var Max,Min : Real);
Begin
 If Value>Max then Max:=Value;
 If Value<Min then Min:=Value;
End;

Procedure SetVFF(Start,Step : Real; n : Word; Var MinX,MaxX,MinY,MaxY : Real;
           Var XValue,YValue : VecArr; PolorE : Char);
Var  i : Word;
   Ch : Char;
   Temp : Real;
Begin
 MaxX:=-1e9; MaxY:=-1e9; MinX:=1e9; MinY:=1e9;
 For i:=1 to n do
 Begin
  MaxMinValue(XValue[i],MaxX,MinX);
  YValue[i]:=20*Log10(YValue[i]); {Convert to dB Unit}
  MaxMinValue(YValue[i],MaxY,MinY);
 End;
```

```
  If PolorE='X' then
  Begin
    For i:=1 to n do YValue[i]:=YValue[i]-MaxYco;
    {Normalize by Set Boundary of Y-Value Max=0}
    MinY:=MinY-MaxYco;
    MaxY:=MaxY-MaxYco;
  End
  Else
  Begin
    For i:=1 to n do YValue[i]:=YValue[i]-MaxY;
    {Normalize by Set Boundary of Y-Value Max=0}
    MinY:=MinY-MaxY;
    MaxY:=MaxY-MaxY;
  End;
End;

Begin
  OpenGraph;
  ChangeColor(2,48); ChangeColor(14,79); ChangeColor(11,46);
  ChangeColor(10,126); ChangeColor(13,43); ChangeColor(12,69);
  ChangeColor(9,116); ChangeColor(5,42); ChangeColor(6,84);
  ChangeColor(3,49);
  Repeat
{   LoadFile('*.*','Open Far Field File',Path,Name,0,0,11,15,4,Ch);
    FileName:=Path+Name;}
    EntryParameter(MinX1,MaxX1,MinY1,MaxY1,Auto,GraphType,
                   FileName,PolarE,Checkout,Format);
    If Checkout=False then
      If FileExist(FileName) and Format then
      Begin
        Assign(FFHfile,FileName);
        ReSet(FFHfile); Seek(FFHfile,0); Read(FFHfile,FFH); Close(FFHfile);
        Assign(FFfile,FileName); ReSet(FFfile);
        InitXYVector;
        n:=FFH.SizeA+1; D:=FFH.Dia; Step:=FFH.StepA; Start:=FFH.ZetaStart;
        For i:=1 to n do
        Begin
          Seek(FFfile,i); Read(FFfile,FF);
          XValue[i]:=FFH.ZetaStart+(i-1)*FFH.StepA; {Set Value to Array X}
          Case PolarE of
            'C' : Begin
                    AbsC(FF.Eco,YValue[i]);
                    YLable:=' Relative Of Co-Polar Field ';
                  End;
            'X' : Begin
                    AbsC(FF.Eco,YValue[i]);
                    YValue[i]:=20*Log10(YValue[i]); {Convert to dB Unit}
```

```
            MaxMinValue(YValue[i],MaxYco,MinYco);
            AbsC(FF.Ecx,YValue[i]);
            YLable:=' Relative Of Cross-Polar Field ';
          End;
        End;
      End;
    Close(FFfile);
    SetVFF(Start,Step,n,MinX,MaxX,MinY,MaxY,XValue,YValue,PolarE);
    If UpCase(Auto[1])<>'A' then
    Begin
      MinX:=MinX1; MaxX:=MaxX1; MinY:=MinY1; MaxY:=MaxY1;
    End;
    XLable:=' Theta '; UnitX:=' degrees '; UnitY:=' dB  ';
    x1:=40; y1:=30; x2:=600; y2:=450;
    With Block do
    Begin
      Header:='Far Field Pattern ';
      StartX:=x1; StartY:=y1; EndX:=x2; EndY:=y2;
    End;
    SetDL:=GraphType[1];
    Design(XValue,YValue,n,MinX,MaxX,MinY,MaxY,UnitX,UnitY,XLable,YLable,
        Block,x1,y1,x2,y2,SetDL,Ch);
  End
  Else
  If (FileName[1]<>#27) and (FileName[1]<>#45) and Format Then
    ErrorMessage('Error','File Not Found.',
            'Press Enter to Entry Parameter.',
            'Press Alt-X or Esc to Exit');
Until Ch in [#27,#45];
CloseGraph;
End.
```