



TESIS - SM 2310

## **Pembuatan Software Radar Cross Section Target**

MUKHAIMY GAZALI  
NRP. 1204 201 001

DOSEN PEMBIMBING  
Dr. Basuki Widodo, M.Sc.

PROGRAM STUDI MAGISTER  
JURUSAN MATEMATIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
INSTITUT TEKNOLOGI SEPULUH NOPEMBER  
SURABAYA  
2007

**PEMBUATAN SOFTWARE  
RADAR CROSS SECTION TARGET**

**T E S I S**

**Tesis ini disusun untuk memenuhi salah satu syarat memperoleh gelar  
Magister Sains (M.Si.)**

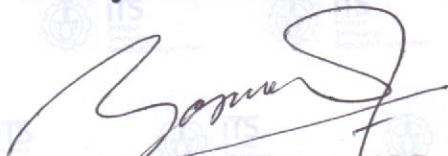
**di  
Institut Teknologi Sepuluh Nopember Surabaya**

*oleh:*

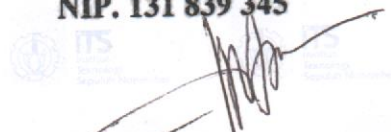
**MUKHAIMY GAZALI**  
**NRP. 1204 201 001**

**Tanggal Ujian : 8 Februari 2007**  
**Periode Wisuda : Maret 2007**

**Disetujui oleh Tim Penguji Tesis:**

  
1. **Dr. Basuki Widodo, M.Sc.**  
**NIP. 131 839 345**

**(Pembimbing)**

  
2. **Dr. M. Isa Irawan, M.T.**  
**NIP. 131 843 894**

**(Penguji)**

  
3. **Drs. Soetrisno, M.IKomp.**  
**NIP. 131 629 805**


**(Penguji)**

  
4. **Drs. Nurul Hidayat, M.Kom.**  
**NIP. 131 835 479**

**(Penguji)**



**Direktur Program Pascasarjana,**

  
**Prof. Ir. Happy Ratna S., M.Sc., Ph.D.**  
**NIP. 130 541 829**

# PEMBUATAN SOFTWARE RADAR CROSS SECTION TARGET

Nama mahasiswa : Mukhaimy Gazali  
NRP : 1204 201 001  
Pembimbing : Dr. Basuki Widodo, M.Sc.

## ABSTRAK

Sebelum obyek *stealth* diproduksi, perlu diketahui perkiraan besarnya *Radar Cross Section* (RCS) yang terjadi pada obyek. Namun software untuk perhitungan RCS tidak mudah diperoleh dan dimodifikasi sesuai keperluan pengguna. Untuk itu pembuatan software RCS target mutlak diperlukan.

Pada tesis ini dibuat software RCS target, yang selanjutnya disebut targetRCS2. Metode yang digunakan untuk prediksi RCS pada targetRCS2 adalah metode *Physical Optics* (PO) yang berbasiskan potongan-potongan segitiga. Untuk membedakan bagian (potongan) yang teriluminasi dengan bagian bayangan digunakan prinsip *back-face culling*. Selanjutnya Model obyek direpresentasikan dengan menggunakan OpenGL, yang mana dipakai sebagai software interface untuk hardware grafik. Perhitungan RCS dilakukan untuk bentuk geometris sederhana, yaitu pelat, kubus, limas dan bola. Untuk bentuk geometri yang kompleks, perhitungan RCS dilakukan pada salah satu model yaitu Frigate. Hasil perhitungan dari targetRCS2 divalidasikan dengan software POFACETS. Pada bentuk geometris sederhana, yaitu pelat, kubus, limas dan bola, hasil validasi tidak memperlihatkan perbedaan yang berarti antara hasil perhitungan pada targetRCS2 dan POFACETS. Sedangkan pada model Frigate, hasil perhitungan RCS dengan targetRCS2 dan POFACETS terlihat perbedaan antara 0,05569 dBsm (decibels square meter) sampai 16,14 dBsm.

**Kata Kunci** : Radar Cross Section, Physical Optics, Back-face Culling, OpenGL.

# TARGET RADAR CROSS SECTION SOFTWARE CONSTRUCTION

By : Mukhaimy Gazali  
Student Identity Number : 1204 201 001  
Supervisor : Dr. Basuki Widodo, M.Sc.

## ABSTRACT

Before stealth object is produced, it is important to know the level of Radar Cross Section (RCS) that will occur on the object. However it is not easy to obtain software for RCS calculation, and the software does not always be able to be modified according with user need. Therefore, it is important to develop target RCS software.

In this thesis we develop target RCS software, called targetRCS2. targetRCS2 applies Physical Optics (PO) method as RCS prediction method, base on triangular patch. To determine whether a patch either illuminated part or shadow part, we use the principles of back-face culling. The object is represented by using OpenGL as interface software to graphics hardware. The RCS calculation will be conducted by using simple geometric form, such as plate, cube, pyramid and sphere. For complex geometric form, RCS calculation will be conducted by using Frigate model. Calculation from targetRCS2 will be compared toward POFACETS. For simple geometric forms, which are plate, cube, pyramid and sphere, validation result does not show any significant difference between targetRCS2 and POFACETS. On Frigate model, RCS calculation using targetRCS2 and POFACETS show difference between 0.05569 dBsm (decibels square meter) and 16.14 dBsm.

**Keywords :** Radar Cross Section, Physical Optics, Back-face Culling, OpenGL.

# Kata Pengantar

Segala puji bagi Allah SWT yang telah memberikan karunia-Nya, berupa kekuatan iman, ilmu pengetahuan dan kesehatan, sehingga penulis dapat menyelesaikan Tesis dengan judul:

## PEMBUATAN SOFTWARE RADAR CROSS SECTION TARGET

Tesis ini merupakan hasil penelitian sebagai salah satu syarat untuk memperoleh gelar Magister Sains (M.Si.) pada Program Studi Magister Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Teknologi Sepuluh Nopember Surabaya.

Penulis menyadari bahwa tesis ini dapat diselesaikan tidak lepas dari bimbingan, arahan, petunjuk, dorongan moral maupun bantuan materil dari berbagai pihak. Untuk itu, pada kesempatan ini penulis mengucapkan terima kasih dan penghargaan yang sebesar-besarnya kepada:

1. DR. Basuki Widodo, selaku pembimbing tesis.
2. DR. Erna Apriliani, selaku Koordinator Program Studi Matematika PPs ITS dan Dosen Wali penulis.
3. Dosen-dosen Matematika ITS lainnya, selaku staf pengajar Program Studi Matematika PPs ITS.
4. Rekan-rekan mahasiswa S2 Matematika terutama angkatan 2004 dan 2005.
5. Pak BOSS Salman Alfarisi, atas semua saran, kritikan dan kerja keras membantu "adik"-nya menyelesaikan Tesis ini.
6. Dieky Adzkia. From "Everybody Love Dieky", may ALLAH bless you. Selaku tempat curhat (securhat-curhatnya) dan seluruh bantuannya mulai dari awal hingga akhir pengerjaan tesis ini. You are one of my best brothers.
7. Ditha (Aditya Maharani), my little sister. Yang banyak bantu kerjaan dan berbagi cerita dengan penulis.
8. Pupus Rikna Afriswari. You help me at the beginning and the middle this thesis working process. You are the one who push me to finish this thesis as soon as possible. You are still my friend.

9. Dessy Dwi Setyawati. You saved my days from big crash. I don't know how to thank you. Your presence those days mean a lot to me. Hope this friendship last forever.
10. Keluarga besar H. Darwis dan H. M. Yusuf, atas semua dukungannya.
11. **Save The Best for Last.** mama Wardah M. Yusuf dan abah Adenan Darwis, atas semua yang telah Mama dan Abah berikan selama ini.
12. Semua pihak yang telah banyak membantu dan tidak tersebut namanya.

Akhirnya penulis berharap dan memohon, semoga bantuan semua pihak mendapat ridho dan bernilai amal ibadah serta mendapat balasan yang setimpal dari Allah SWT. Semoga tesis ini dapat memberikan manfaat dan menambah wawasan keilmuan serta mendapat ridho dari Allah SWT

Surabaya, 14 Februari 2007

Penulis

# Daftar Isi

<b>1</b>	<b>Pendahuluan</b>	<b>1</b>
1.1	Latar Belakang . . . . .	1
1.2	Rumusan Masalah . . . . .	2
1.3	Batasan Masalah . . . . .	2
1.4	Tujuan dan Manfaat . . . . .	2
1.5	Metode Penelitian . . . . .	2
1.5.1	Pembuatan Interface Awal . . . . .	3
1.5.2	Menelaah Permasalahan RCS dan Penggunaan PO . . . . .	3
1.5.3	Membuat Komponen Prediksi RCS . . . . .	3
1.5.4	Membuat Komponen untuk Membaca File DXF . . . . .	4
1.5.5	Perhitungan RCS dengan Desain Obyek dari AutoCAD . . . . .	4
<b>2</b>	<b>Kajian Pustaka</b>	<b>5</b>
2.1	<i>Scattering</i> . . . . .	5
2.2	<i>Radar Cross Section</i> . . . . .	6
2.3	<i>Physical Optics</i> . . . . .	9
2.3.1	<i>Scattering</i> pelat Segiempat dengan PO . . . . .	10
2.4	Perhitungan RCS Potongan Segitiga dengan PO . . . . .	13
2.4.1	Penggunaan Koefisien Refleksi . . . . .	15
2.4.2	Medan Listrik Insiden pada Koordinat Lokal . . . . .	16
2.4.3	RCS dengan Faktor Polarisasi Penerima . . . . .	17
2.4.4	Perhitungan Difusi . . . . .	19
2.5	Evaluasi Integral Difraksi Elektromagnetik . . . . .	21
2.6	Transformasi Koordinat . . . . .	23
2.6.1	Kosinus Arah . . . . .	26
2.7	<i>Back-face Culling</i> . . . . .	27
2.8	Alur Perhitungan RCS Target . . . . .	28
<b>3</b>	<b>Perancangan dan Pembuatan Software</b>	<b>31</b>
3.1	Spesifikasi Software . . . . .	31
3.2	File Input . . . . .	33
3.2.1	Format File .XMO . . . . .	33
3.2.2	File AutoCAD . . . . .	35
3.2.3	File StereoLithography . . . . .	36
3.2.4	File Input Sifat Permukaan . . . . .	38
3.3	Representasi Model dengan C# OpenGL Framework . . . . .	40
3.3.1	Modifikasi C# OpenGL Framework . . . . .	41
3.4	Antarmuka Pengguna . . . . .	42
3.4.1	Bagian Input . . . . .	43

3.4.2	Tampilan Model dan Hasil Perhitungan . . . . .	45
3.5	Kode Physical Optics . . . . .	48
<b>4</b>	<b>Uji Coba dan Evaluasi Software</b>	<b>50</b>
4.1	Pelat . . . . .	50
4.2	Kubus . . . . .	54
4.3	Limas . . . . .	56
4.4	Bola . . . . .	58
4.5	Frigate . . . . .	60
<b>5</b>	<b>Penutup</b>	<b>63</b>
5.1	Kesimpulan . . . . .	63
5.2	Saran . . . . .	63
<b>A</b>	<b>Source Code POCalc</b>	<b>69</b>
<b>B</b>	<b>Penjelasan Menu targetRCS2</b>	<b>100</b>
B.1	Menu File . . . . .	100
B.2	Menu Model . . . . .	101
B.3	Menu Perhitungan . . . . .	102
B.4	Menu Tools . . . . .	103
B.5	Menu Help . . . . .	104
<b>C</b>	<b>Keperluan Minimum targetRCS2</b>	<b>105</b>



# Daftar Gambar

2.1	Beberapa mekanisme <i>scattering</i> . . . . .	6
2.2	Sistem koordinat bola . . . . .	8
2.3	Scattering gelombang pada pelat segiempat . . . . .	10
2.4	Pendekatan Gaussian pada Permukaan Kasar . . . . .	19
2.5	Jarak Korelasi . . . . .	20
2.6	Geometri segitiga integrasi . . . . .	22
2.7	Pelat segitiga dengan orientasi yang bersesuaian . . . . .	24
2.8	Sudut rotasi segitiga . . . . .	24
2.9	Rotasi pertama dengan sudut $\alpha$ pada sumbu $z$ . . . . .	25
2.10	Rotasi pertama dengan sudut $\beta$ pada sumbu $z$ . . . . .	25
2.11	Kosinus Arah . . . . .	27
2.12	Pandangan arah $\vec{V}$ pada poligon dengan normal $\vec{N}$ . . . . .	28
2.13	Alur perhitungan utama . . . . .	29
2.14	Alur perhitungan tiap potongan segitiga . . . . .	30
3.1	Contoh file XMO . . . . .	35
3.2	Struktur file StL ASCII . . . . .	37
3.3	Dialog pengolahan input permukaan . . . . .	39
3.4	Dialog input material dasar . . . . .	40
3.5	Antarmuka software . . . . .	42
3.6	Input software bagian atas . . . . .	43
3.7	Input software bagian bawah . . . . .	44
3.8	Tampilan model pada software . . . . .	45
3.9	Tampilan hasil perhitungan pada software . . . . .	46
3.10	Tampilan koordinat polar pada software . . . . .	47
3.11	Tampilan koordinat kartersian pada software . . . . .	48
4.1	RCS pelat terhadap frekuensi . . . . .	51
4.2	Pola RCS pelat dari Tabel 4.3 . . . . .	54
4.3	Pola RCS kubus dari Tabel 4.4 . . . . .	56
4.4	Bentuk limas untuk bahan uji . . . . .	56
4.5	Pola RCS limas dari Tabel 4.5 . . . . .	58
4.6	Bentuk bola untuk bahan uji . . . . .	58
4.7	Pola RCS bola dari Tabel 4.6 . . . . .	60
4.8	Bentuk Frigate untuk bahan uji . . . . .	60
4.9	Pola RCS Frigate dari Tabel 4.7 . . . . .	62
B.1	targetRCS2 Menu File . . . . .	100
B.2	targetRCS2 Menu Model . . . . .	101
B.3	targetRCS2 Menu Perhitungan . . . . .	103

B.4	targetRCS2 Menu Tools . . . . .	103
B.5	targetRCS2 Menu Help . . . . .	104

# Daftar Tabel

3.1	Format data file .XMO . . . . .	34
3.2	Struktur grup kode 3DFace . . . . .	36
3.3	Format StL Binary . . . . .	37
3.4	Struktur data file LayerValue . . . . .	38
3.5	Struktur data file LayerMaterials . . . . .	39
3.6	Fungsi-fungsi pada <b>POCalc</b> . . . . .	49
4.1	Perhitungan $\sigma$ pelat terhadap frekuensi . . . . .	51
4.2	$\sigma$ pelat pada targetRCS2 dan Persamaan 2.18 . . . . .	52
4.3	$\sigma$ pelat pada targetRCS2 dan POFACETS . . . . .	53
4.4	$\sigma_{\theta\theta}(\theta, \phi = 15^\circ)$ kubus pada targetRCS2 dan POFACETS . . . . .	55
4.5	$\sigma_{\theta\theta}(\theta, \phi = 15^\circ)$ limas pada targetRCS2 dan POFACETS . . . . .	57
4.6	$\sigma_{\theta\theta}(\theta, \phi = 0^\circ)$ bola pada targetRCS2 dan POFACETS . . . . .	59
4.7	$\sigma_{\theta\theta}(\theta, \phi = 0^\circ)$ Frigate pada targetRCS2 dan POFACETS . . . . .	61
B.1	Fungsi Menu File . . . . .	100
B.2	Fungsi Menu Model . . . . .	102
B.3	Fungsi Menu Perhitungan . . . . .	102
B.4	Fungsi Menu Tools . . . . .	103
B.5	Fungsi Menu Help . . . . .	104

# BAB 1

## Pendahuluan

### 1.1 Latar Belakang

*Stealth* merupakan gabungan prinsip dan teknologi yang mampu membuat suatu obyek sulit untuk dideteksi oleh sensor seperti radar, pencari panas maupun detektor suara bahkan oleh mata manusia [10, 16]. Hal ini ditujukan agar obyek, yang memiliki kemampuan stealth tidak mudah untuk diserang.

Secara khusus teknologi *stealth* ditujukan untuk mengurangi *Radar Cross Section* (RCS) yang terjadi pada obyek [28]. RCS merupakan ukuran daya scatter pada arah tertentu ketika target teriluminasi gelombang kejadian [15]. Semakin kecil RCS pada suatu obyek, maka obyek tersebut semakin sulit untuk dideteksi. Hal ini berarti tingkat *stealth* yang dimiliki semakin besar.

Pada fase desain sangatlah penting dilakukan komputasi prediksi seberapa besar RCS yang terjadi sebelum obyek diproduksi. Oleh karena itu keberadaan software prediksi RCS mutlak diperlukan.

Pada penelitian ini akan dibuat software untuk prediksi RCS. Dimana bentuk obyek yang akan diprediksikan digambar menggunakan AutoCAD, sehingga software harus mampu untuk membaca file gambar yang dibuat dengan AutoCAD. File gambar ini dibentuk dalam format Drawing Interchange File (DXF)[1].

Metode yang digunakan untuk prediksi RCS target adalah metode Physical Optics (PO)[3, 15, 20, 23].

## 1.2 Rumusan Masalah

Permasalahan utama yang akan dibahas pada tesis ini adalah bagaimana membuat software prediksi RCS menggunakan PO berbasis potongan segitiga, untuk model yang digambar menggunakan AutoCAD.

## 1.3 Batasan Masalah

Pada tesis ini dibuat batasan masalah sebagai berikut.

1. File DXF yang dibaca merupakan format file DXF dari maksimal AutoCAD versi R12.
2. Pada proses validasi digunakan bangun geometris sederhana, seperti pelat, kubus dan limas.
3. Perhitungan *scattering* hanya melibatkan proses refleksi tunggal dan tidak melibatkan proses *scattering* lainnya.

## 1.4 Tujuan dan Manfaat

Tujuan dari tesis ini adalah membuat software prediksi RCS yang mampu membaca dan mengolah obyek yang digambar dari AutoCAD.

Sedangkan manfaat dari tesis ini adalah diperoleh nilai dan pola RCS yang terjadi pada obyek yang didesain, sehingga terlihat efektif tidaknya desain suatu obyek.

## 1.5 Metode Penelitian

Pembuatan software RCS target ini digambarkan secara umum sebagai berikut. Input utama dari software ini adalah desain obyek yang berasal dari AutoCAD dengan basis 3-dimensi. Output berupa besarnya RCS yang terjadi pada be-

berapa sudut obyek baik dengan sistem radar monostatis ataupun bistatis. Algoritma perhitungan yang digunakan adalah pendekatan PO.

Tahapan pembuatan software RCS target yang akan dibuat, diuraikan sebagai berikut.

### **1.5.1 Pembuatan Interface Awal**

Pada tahapan ini dibuat interface awal untuk software sebagaimana rancangan di atas. Software dibuat dengan menggunakan bahasa pemrograman C# dengan .Net framework 2.0 [17], dimana model akan direpresentasikan dengan OpenGL. OpenGL tersebut merupakan antarmuka software ke hardware grafik untuk menghasilkan aplikasi interaktif tiga-dimensi [25, 19]. Komponen yang digunakan untuk antarmuka OpenGL ini adalah "C# OpenGL Framework" versi 1.6 [5]. Pada tahapan ini juga dibuat komponen untuk menampilkan koordinat polar.

### **1.5.2 Menelaah Permasalahan RCS dan Penggunaan PO**

Pada tahapan ini dipelajari secara mendalam permasalahan RCS untuk target dan bagaimana cara perhitungan menggunakan PO. Dimana akan ditelaah lebih lanjut hubungan antara konsep RCS, persamaan elektromagnetik dan konsep PO. Sehingga diperoleh rumusan untuk perhitungan RCS untuk target kompleks. Target kompleks yang dimaksudkan di sini adalah bentuk target yang terdiri lebih dari satu bentuk permukaan.

### **1.5.3 Membuat Komponen Prediksi RCS**

Dari penelaahan yang telah dilakukan dibuat komponen untuk perhitungan RCS target, sebagai pemrosesan utama. Untuk pengujian dibuat file teks sebagai representasi obyek (bukan data gambar dari AutoCAD).

#### **1.5.4 Membuat Komponen untuk Membaca File DXF**

Pada tahapan ini akan ditelaah bagaimana AutoCAD menyusun struktur file untuk menyimpan bentuk gambar yang ada, ke dalam bentuk file teks (file DXF). Untuk penyederhanaan digunakan demo OpenGL Profesional[6], dimana pada proses penyederhanaan ini diperoleh file AutoCAD yang hanya berisi instruksi untuk bentuk "3Dface", "Line", dan "Point". Dari file yang telah disederhanakan ini akan dibuat fungsi-fungsi untuk merepresentasi file DXF ke komponen interface OpenGL. Bentuk "3Dface" yang diperoleh merupakan potongan-potongan (*patch*) segitiga dari bentuk awal.

#### **1.5.5 Perhitungan RCS dengan Desain Obyek dari AutoCAD**

Pada tahapan ini akan dilakukan proses simulasi prediksi RCS dari berbagai bentuk obyek yang berasal dari AutoCAD. Obyek sebelumnya ditampilkan dulu sebagai bukti bahwa proses pembacaan file AutoCAD telah benar. Kemudian dilakukan proses perhitungan dengan komponen prediksi RCS. Dari hasil perhitungan akan dianalisis besarnya RCS yang terjadi pada struktur obyek yang ada, maupun hal lainnya.

# BAB 2

## Kajian Pustaka

Selama ini telah terdapat beberapa software untuk memprediksikan RCS, diantaranya CADRCS [4], Epsilon [21], dan RadBase [27]. Namun akses untuk mendapatkan software-software ini terbatas. Selain itu software-software ini tidak dapat dimodifikasi dengan mudah sesuai dengan keperluan. Untuk itu akan dibuat software untuk prediksi RCS target yang bersesuaian dan diharapkan dapat dikembangkan berkelanjutan.

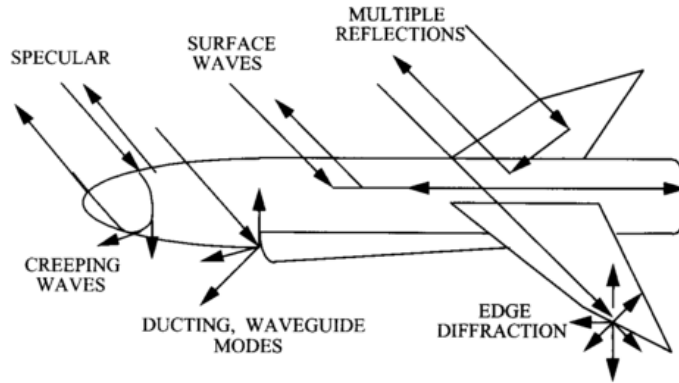
Metode yang digunakan untuk prediksi RCS target adalah metode *Physical Optics* (PO). Metode PO sering digunakan secara umum untuk menganalisa obyek kompleks pada frekuensi tinggi, karena metode ini merupakan metode yang sederhana [2].

### 2.1 *Scattering*

Ketika gelombang elektromagnetik dari radar mengenai suatu obyek, arus listrik terinduksi pada obyek. Arus ini berjalan pada permukaan obyek dan meradiasi ulang medan electromagnetic. Secara umum radiasi ini bisa menuju arah mana saja. Proses radiasi ulang ini disebut *scattering* [2]. Sifat radiasi *scatter* obyek sangat tergantung pada geometri obyek dan arah radiasi kejadian, sehingga setiap obyek bisa memiliki penanda *scattering* yang unik.

Beberapa mekanisme *scattering* yang penting diantaranya adalah refleksi, difraksi, gelombang permukaan, dan *ducting*. Sebagai ilustrasi, mekanisme *scattering* diperlihatkan pada Gambar 2.1. Pada penelitian ini hanya dibatasi pada scattering untuk refleksi tunggal.





Gambar 2.1: Beberapa mekanisme *scattering*

## 2.2 Radar Cross Section

Radar Cross Section merupakan bentuk dari observasi radar yang Persamaannya dibentuk dari fisika radiasi elektromagnetik yang dikembangkan oleh James Clerk Maxwell. Persamaan ini ditulis pada sekitar tahun 1850 yang merupakan himpunan Persamaan yang mendeskripsikan teori elektromagnetik. [16]

Bentuk Persamaan Maxwell sebagai berikut. [15, 16]

$$\begin{aligned}
 \nabla \times \vec{H} &= \vec{J} + \frac{\partial \vec{D}}{\partial t}, \\
 \nabla \times \vec{E} &= -\frac{\partial \vec{B}}{\partial t}, \\
 \nabla \cdot \vec{D} &= \rho, \\
 \nabla \cdot \vec{B} &= 0.
 \end{aligned}
 \tag{2.1}$$

dimana

$\vec{H}$  = vektor intensitas medan magnet (*vector magnetic field intensity*)

$\vec{J}$  = vektor kerapatan arus (*vector current density*)

$\vec{D}$  = vektor kerapatan flux listrik (*vector electric flux density*)

$t$  = waktu

$\vec{E}$  = vektor intensitas medan listrik (*vector electric field intensity*)

$\vec{B}$  = vektor kerapatan flux magnetik (*vector magnetic flux density*)

$\rho$  = kerapatan beban listrik (*electric charge density*)

Sedangkan pada IEEE [14, 15, 26] definisi RCS ( $\sigma$ ) dinyatakan sebagai

$$\sigma = \frac{\text{Daya yang direfleksikan ke penerima tiap unit sudut penuh}}{\text{Densitas daya kejadian}/4\pi} \quad (2.2)$$

Lebih tepatnya, RCS merupakan limit dari rasio jarak *scatterer* menuju titik dengan daya *scatterer* diukur mendekati tak hingga.

$$\sigma = \lim_{r \rightarrow \infty} 4\pi r^2 \frac{|\vec{E}_s|^2}{|\vec{E}_i|^2} = \lim_{r \rightarrow \infty} 4\pi r^2 \frac{|\vec{H}_s|^2}{|\vec{H}_i|^2} \quad (2.3)$$

dimana

$r$  = jarak ke observer

$\vec{E}_i$  = vektor medan listrik insiden

$\vec{H}_i$  = vektor medan magnet insiden

$\vec{E}_s$  = vektor medan listrik *scattering*

$\vec{H}_s$  = vektor medan magnet *scattering*

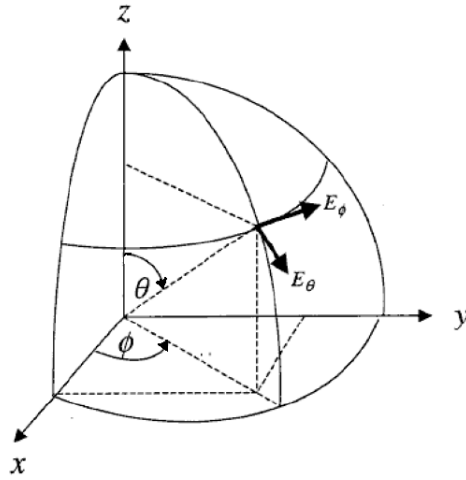
Besarnya RCS juga bisa diukur dari luasan efektif suatu obyek [14], yaitu:

$$\sigma = \frac{4\pi A_e^2}{\lambda^2} \quad (2.4)$$

dimana  $A_e$  adalah luasan efektif obyek. Persamaan 2.4 seringkali digunakan untuk memperkirakan RCS tertinggi untuk permukaan datar ketika dilihat secara langsung oleh radar.

RCS juga merupakan fungsi dari berbagai faktor yang melibatkan konfigurasi target dan komposisi materialnya, frekuensi dan panjang gelombang, polarisasi pemancar dan penerima, dan aspek target (orientasi sudut target) relatif terhadap radar. Sehingga, secara umum,  $\sigma$  dapat dinyatakan sebagai  $\sigma_{pq}(\theta, \phi)$  [9], dimana  $q$  dan  $p$  masing-masing merupakan polarisasi insiden dan

penerima. Sedangkan  $(\theta, \phi)$  merupakan sudut pada koordinat bola (*spherical*) sebagaimana terlihat pada Gambar 2.2.



Gambar 2.2: Sistem koordinat bola

Perhitungan RCS pada dasarnya adalah permasalahan mendapatkan medan listrik *scattering* dari suatu target. Jika arus yang terinduksi pada target oleh gelombang insiden dapat ditentukan, maka integral radiasi dapat digunakan untuk menghitung medan listrik *scattering*.

Bentuk umum Persamaan integral radiasi untuk medan listrik [14], dinyatakan dengan

$$\vec{E}(r, \theta, \phi) = \frac{-jk\eta}{4\pi r} e^{-jkr} \iiint_V \vec{J} e^{jkg} dV \quad (2.5)$$

dimana

$k$  = konstanta propagasi =  $2\pi/\lambda$

$\lambda$  = panjang gelombang (*wavelength*)

$\eta$  = impedansi intristik medium

$\vec{J}$  = densitas arus volume

$g = \vec{r}' \cdot \hat{r}$

$\vec{r}'$  merupakan vektor posisi dari titik pusat, yaitu:

$$\vec{r}' = \hat{x}x' + \hat{y}y' + \hat{z}z' \quad (2.6)$$

dan  $\hat{r}$  merupakan unit vektor pada arah titik observasi, yaitu:

$$\hat{r} = \hat{x}u + \hat{y}v + \hat{z}w \quad (2.7)$$

dimana  $u, v, w$  adalah kosinus arah titik observasi.

$$\begin{aligned} u &= \sin \theta \cos \phi \\ v &= \sin \theta \sin \phi \\ w &= \cos \theta \end{aligned} \quad (2.8)$$

Nilai RCS dari target berkisar antara jutaan  $m^2$  (kapal) hingga bilangan fraksi yang sangat kecil (burung dan serangga). Oleh karena kisaran yang sangat besar, ilmuwan di bidang radar seringkali mengubah (konversi) nilai RCS ke dalam skala logaritmik.[15] Nilai RCS dibentuk dalam satuan dBsm. Proses konversi dilakukan dengan

$$\sigma(\text{dBsm}) = 10 \log_{10} (\sigma(\text{m}^2)) \quad (2.9)$$

### 2.3 *Physical Optics*

Pendekatan PO merupakan salah satu pendekatan RCS yang baik untuk target tiga-dimensi berukuran besar pada frekuensi tinggi. [3]. Prinsip utama PO adalah penggunaan arus *Geometrical Optics* untuk bagian permukaan target yang teriluminasi. Sedangkan untuk bagian bayangan arusnya nol. Pernyataan

dapat dinyatakan dalam bentuk matematika sebagai:

$$\vec{J}_s = \begin{cases} 2\hat{n} \times \vec{H}_i, & \text{untuk bagian yang teriluminasi} \\ 0, & \text{untuk bagian yang bayangan} \end{cases} \quad (2.10)$$

dimana

$\vec{J}_s$  = vektor arus permukaan

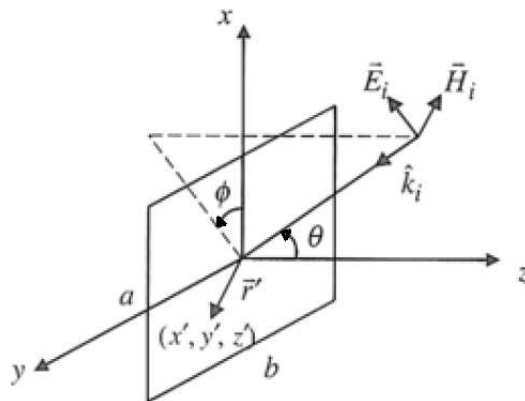
$\hat{n}$  = normal permukaan

Arus permukaan ini selanjutnya digunakan pada integral radiasi untuk menghitung medan *scatter* dari target.

Salah satu pendekatan perhitungan *scattering* frekuensi tinggi adalah mendekomposisi target kompleks dalam bentuk potongan-potongan pelat segitiga [14]. Selanjutnya nilai RCS target diperoleh dari total perhitungan medan *scatter* dari potongan-potongan segitiga tersebut.

### 2.3.1 *Scattering* pelat Segiempat dengan PO

Dalam subbab ini diambil pelat segiempat sebagai contohnya. Pelat segiempat ini memiliki dimensi  $a$  dan  $b$ , seperti yang diperlihatkan pada Gambar 2.3.



Gambar 2.3: Scattering gelombang pada pelat segiempat

Menurut [14], vektor medan listrik insiden ditentukan oleh konstanta  $E_{0\theta}$  dan  $E_{0\phi}$  dinyatakan sebagai

$$\vec{E}_i = \left( E_{0\theta} \hat{\theta} + E_{0\phi} \hat{\phi} \right) e^{-j\vec{k}_i \cdot \vec{r}} \quad (2.11)$$

dimana  $\vec{k}_i = k \hat{k}_i$ .  $\hat{k}_i$  = propagasi gelombang menuju titik pusat, sehingga  $\hat{k}_i = -\hat{r}$ .

Jika suatu insiden terpolarisasi memotong garis listrik (*Transverse Electric* / TE) atau polarisasi insiden terjadi terhadap  $\theta$ , maka  $E_{0\theta} \neq 0, E_{0\phi} = 0$ .

Sedangkan jika suatu insiden terpolarisasi memotong garis magnet (*Transverse Magnetic* / TM) atau polarisasi insiden terjadi terhadap  $\phi$ , maka  $E_{0\theta} = 0, E_{0\phi} \neq 0$ .

Intensitas magnetik (vektor medan magnet insiden) dapat dinyatakan sebagai

$$\vec{H}_i = \frac{-\hat{r} \times \vec{E}_i}{\eta} = - \left( E_{0\theta} \hat{\phi} + E_{0\phi} \hat{\theta} \right) \frac{e^{-j\vec{k}_i \cdot \vec{r}}}{\eta} \quad (2.12)$$

Menurut [14], arus pada pelat didekati dengan

$$\vec{J}_s \approx -2\hat{z} \times \left( E_{0\theta} \hat{\phi} + E_{0\phi} \hat{\theta} \right) \frac{e^{-j\vec{k}_i \cdot \vec{r}}}{\eta} \quad (2.13)$$

Vektor dan hasil kali vektor yang diperlukan untuk mengevaluasi Persamaan 2.13 adalah  $\vec{r}' =$  vektor posisi suatu titik  $(x', y', z')$  pada permukaan  $= \hat{x}x' + \hat{y}y'$

$$\begin{aligned}
-\vec{k}_i &= \hat{x} \sin \theta \cos \phi + \hat{y} \sin \theta \sin \phi + \hat{z} \cos \theta \\
-\hat{k}_i \cdot \vec{r} &= k (x' \sin \theta \cos \phi + y' \sin \theta \sin \phi) \\
\hat{z} \times \hat{\theta} &= -\hat{x} \sin \theta \cos \phi + \hat{y} \sin \theta \sin \phi \\
\hat{z} \times \hat{\phi} &= -\hat{x} \cos \phi - \hat{y} \sin \phi
\end{aligned}$$

Sehingga nilai pendekatan untuk arus menjadi

$$\vec{J}_s \approx \frac{-2e^{jkh}}{\eta} [\hat{x} (E_{0\theta} \cos \phi - E_{0\phi} \cos \theta \sin \phi) + \hat{y} (E_{0\theta} \sin \phi - E_{0\phi} \cos \theta \cos \phi)] \quad (2.14)$$

dimana  $h = x' \sin \theta \cos \phi + y' \sin \theta \sin \phi$ .

Karena itu Persamaan 2.5 dapat ditulis dengan,

$$\begin{aligned}
E_\theta(P) &= \frac{-jk\eta}{4\pi r} e^{-jkr} \iint_S \frac{2e^{jkh}}{\eta} \\
&[(E_{0\theta} \cos \phi - E_{0\phi} \cos \theta \sin \phi) \cos \theta_P \cos \phi_P \\
&+ (E_{0\theta} \sin \phi - E_{0\phi} \cos \theta \cos \phi) \cos \theta_P \sin \phi_P] e^{jkg} dx' dy' \quad (2.15)
\end{aligned}$$

dimana kuantitas titik observasi dinyatakan dengan  $P$  dan

$$g = x' \sin \theta_P \cos \phi_P + y' \sin \theta_P \sin \phi_P = \vec{r}' \cdot \hat{r} \quad (2.16)$$

Pada *scattering* monostatis, maka berlaku

$$\theta_P = \theta$$

$$\phi_P = \phi$$

$$g = h$$

Selanjutnya diasumsikan bahwa gelombang insiden terpolarisasi terhadap  $\theta$ , yaitu ( $E_{0\theta} = 0$ ) adalah

$$E_\theta(r, \theta, \phi) = \frac{jk}{2\pi r} e^{-jkr} \int_{-a/2}^{a/2} \int_{-b/2}^{b/2} E_{0\theta} \cos \theta e^{j2kh} dy' dx' \quad (2.17)$$

atau dapat ditulis,

$$E_{\theta}(r, \theta, \phi) = \frac{jk}{2\pi r} e^{-jkr} E_{0\theta} \cos \theta \int_{-a/2}^{a/2} e^{j2kx'u} dx' \int_{-b/2}^{b/2} e^{j2ky'v} dy' \quad (2.18)$$

dimana  $u$  dan  $v$  adalah kosinus arah  $x$  dan  $y$ . Dengan mengevaluasi integralnya diperoleh,

$$E_{\theta}(r, \theta, \phi) = \frac{jk}{2\pi r} e^{-jkr} E_{0\theta} \cos \theta ab \operatorname{sinc}^2(kau) \operatorname{sinc}^2(kbv) \quad (2.19)$$

dimana  $\operatorname{sinc}(x) = \sin(x)/x$ . Selanjutnya perhitungan RCS yang dibentuk oleh Persamaan 2.3, dengan luas pelat  $A$  (yaitu  $A = ab$ ) adalah:

$$\sigma = \lim_{r \rightarrow \infty} 4\pi r^2 \frac{|\vec{E}_S|^2}{|\vec{E}_i|^2} = \lim_{r \rightarrow \infty} 4\pi r^2 \frac{|\vec{E}(r, \theta, \phi)|^2}{|\vec{E}_{0\theta}|^2} \quad (2.20)$$

atau dapat ditulis,

$$\sigma_{\theta\theta}(\theta, \phi) = 4\pi r^2 \left[ \frac{A^2 k^2}{4\pi^2 r^2} \cos^2 \theta \right] \cdot \operatorname{sinc}^2(kau) \cdot \operatorname{sinc}^2(kbv) \quad (2.21)$$

atau,

$$\sigma_{\theta\theta}(\theta, \phi) = \frac{4\pi A^2}{\lambda^2} \cos^2 \theta \cdot \operatorname{sinc}^2(kau) \cdot \operatorname{sinc}^2(kbv) \quad (2.22)$$

## 2.4 Perhitungan RCS Potongan Segitiga dengan PO

Untuk target ukuran besar, target kompleks biasanya didekomposisi menjadi bentuk geometris sederhana. Dalam penelitian ini bentuk geometris yang digunakan adalah pelat segitiga. Hasil perhitungan RCS untuk tiap bagian segitiga ini akan dijumlahkan untuk mendapatkan RCS target secara keseluruhan. Penjumlahan RCS ini dilakukan dengan cara koheren [14].



Perhitungan RCS secara koheren, yaitu:

$$\sigma = \lim_{r \rightarrow \infty} 4\pi r^2 \left| \vec{E}_{S_1} + \vec{E}_{S_2} + \dots + \vec{E}_{S_M} \right|^2 \quad (2.23)$$

dimana  $M$  merupakan jumlah potongan segitiga. Pada perhitungan RCS ini,  $\left| \vec{E}_i \right|$  dibuat dalam keadaan gelombang amplitudo unit, yaitu  $\left| \vec{E}_i \right| = 1$  [9].

Jika polarisasi insiden terjadi terhadap  $\theta$ , maka

$$\begin{aligned} E_{0\theta} &= 1 \\ E_{0\phi} &= 0 \end{aligned} \quad (2.24)$$

Jika polarisasi insiden terjadi terhadap  $\phi$ , maka

$$\begin{aligned} E_{0\theta} &= 0 \\ E_{0\phi} &= 1 \end{aligned} \quad (2.25)$$

Sedangkan untuk medan listrik *scatter* potongan segitiga, yaitu:

$$\vec{E}_s = \frac{-jk\eta}{4\pi r} e^{-jk r} \iint_S \vec{J}_s e^{jk g} dS \quad (2.26)$$

menggunakan Persamaan 2.10 dan 2.12 dan selanjutnya diperoleh Persamaan arus PO, yaitu:

$$\vec{J}_s = \frac{2}{\eta} \hat{n} \times \left( \vec{E}_{0\theta} \hat{\theta} + \vec{E}_{0\phi} \hat{\phi} \right) e^{-j\vec{k}_i \cdot \vec{r}} = \frac{2}{\eta} \hat{n} \times \left( \vec{E}_{0\theta} \hat{\theta} + \vec{E}_{0\phi} \hat{\phi} \right) e^{jk h} \quad (2.27)$$

dimana  $h = x'u + y'v + z'w$ .

Karena itu medan listrik *scatter* menjadi,

$$\vec{E}_s = \frac{-j}{\lambda r} e^{-jk r} \underbrace{\left[ \hat{n} \times \left( \vec{E}_{0\theta} \hat{\theta} + \vec{E}_{0\phi} \hat{\phi} \right) \right]}_{\text{faktor polarisasi } \Upsilon} \underbrace{\iint_S e^{jk(g+h)} dS}_{I_c} \quad (2.28)$$

Selanjutnya untuk perhitungan  $I_c$  akan dijelaskan pada subbab 2.5.

Untuk tiap potong segitiga, nilai RCS diperoleh dari

$$\sigma = \lim_{r \rightarrow \infty} 4\pi r^2 \frac{|\vec{E}_s|^2}{|\vec{E}_i|^2} = \lim_{r \rightarrow \infty} 4\pi r^2 \frac{\left| \frac{-j}{\lambda r} \cdot e^{-jkr} \cdot \Upsilon \cdot I_c \right|^2}{1} \quad (2.29)$$

atau dapat dinyatakan dengan

$$\sigma = \frac{4\pi}{\lambda^2} |\Upsilon \cdot I_c|^2 \quad (2.30)$$

RCS target untuk keseluruhan potongan segitiga secara koheren digunakan Persamaan 2.23 dan 2.30 yaitu:

$$\sigma = \frac{4\pi}{\lambda^2} \left| \sum_{m=1}^M \Lambda_m \right|^2 \quad (2.31)$$

dimana

$$\Lambda_m = \Upsilon_m (I_c)_m \quad (2.32)$$

### 2.4.1 Penggunaan Koefisien Refleksi

Menurut [9] arus permukaan pada koordinat kartesian lokal dengan koefisien refleksi  $\Gamma$  adalah,

$$\begin{aligned} J_{x''} &= \frac{2}{\eta} \left( -E_{i\theta''} \cdot \cos \phi'' \cdot \Gamma_{\parallel} + E_{i\phi''} \cdot \sin \phi'' \cdot \Gamma_{\perp} \right) e^{jkh} \\ J_{y''} &= \frac{2}{\eta} \left( -E_{i\theta''} \cdot \sin \phi'' \cdot \Gamma_{\parallel} - E_{i\phi''} \cdot \cos \phi'' \cdot \Gamma_{\perp} \right) e^{jkh} \end{aligned} \quad (2.33)$$

$$J_{z''} = 0$$

dimana

$$\begin{aligned} \Gamma_{\parallel} &= \frac{-\eta \cos \theta''}{2R_S + \eta \cos \theta''} \\ \Gamma_{\perp} &= \frac{-\eta}{2R_S \cos \theta'' + \eta} \end{aligned} \quad (2.34)$$

Untuk  $\theta''$  adalah sudut permukaan bola lokal dan  $R_s$  merupakan hambatan permukaan (*surface resistance*). Pada penghantar listrik sempurna (*perfect electric conductor*), nilai  $R_s = 0$  atau  $|\Gamma| = 1$ .

Sudut permukaan bola lokal diperoleh dari

$$\begin{aligned}\theta'' &= \arcsin \left( \sqrt{(u'')^2 + (v'')^2} \right) \\ \phi'' &= \arctan \left( \frac{v''}{u''} \right)\end{aligned}\tag{2.35}$$

Nilai  $u''$  dan  $v''$  diperoleh dari matrik kosinus arah, lihat Persamaan 2.64.

Faktor polarisasi untuk Persamaan 2.33, dinyatakan dengan,

$$\begin{aligned}\Upsilon_{x''} &= (-E_{i\theta''} \cdot \cos \phi'' \cdot \Gamma_{\parallel} + E_{i\phi''} \cdot \sin \phi'' \cdot \Gamma_{\perp}) \\ \Upsilon_{y''} &= (-E_{i\theta''} \cdot \sin \phi'' \cdot \Gamma_{\parallel} - E_{i\phi''} \cdot \cos \phi'' \cdot \Gamma_{\perp}) \\ \Upsilon_{z''} &= 0\end{aligned}\tag{2.36}$$

Penanda ( $''$ ) digunakan untuk melambangkan variabel berada pada koordinat lokal. Pada [9], dijelaskan bahwa tiap titik pada koordinat global dibentuk ke dalam koordinat lokal, dimana pada koordinat lokal, normal segitiga  $\hat{n} = \hat{z}''$ . Proses transformasi selanjutnya dijelaskan pada subbab 2.6.

### 2.4.2 Medan Listrik Insiden pada Koordinat Lokal

Untuk mendapatkan medan listrik insiden pada koordinat bola lokal, yaitu  $E_{i\theta''}$  dan  $E_{i\phi''}$ , dilakukan proses transformasi dengan langkah-langkah berikut ini:

Transformasi koordinat medan insiden koordinat bola global ke koordinat kartesian global dengan persamaan,

$$\begin{bmatrix} E_{ix} \\ E_{iy} \\ E_{iz} \end{bmatrix} = \begin{bmatrix} \sin \theta \cos \phi & \cos \theta \cos \phi & -\sin \phi \\ \sin \theta \sin \phi & \sin \theta \cos \phi & \cos \phi \\ \cos \theta & -\sin \theta & 0 \end{bmatrix} \begin{bmatrix} E_{0r} \\ E_{0\theta} \\ E_{0\phi} \end{bmatrix} \quad (2.37)$$

Pada permasalahan ini  $E_{0r} = 0$ , sehingga diperoleh

$$\begin{aligned} E_{ix} &= E_{0\theta} \cdot (\cos \theta \cos \phi) - E_{0\phi} \sin \phi \\ E_{iy} &= E_{0\theta} \cdot (\sin \theta \cos \phi) + E_{0\phi} \cos \phi \\ E_{iz} &= -E_{0\theta} \cdot \sin \theta \end{aligned} \quad (2.38)$$

Medan listrik insiden pada koordinat kartesian global ditransformasikan ke dalam koordinat kartesian lokal, yaitu:

$$\begin{bmatrix} E_{ix''} \\ E_{iy''} \\ E_{iz''} \end{bmatrix} = T'' T' \begin{bmatrix} E_{ix} \\ E_{iy} \\ E_{iz} \end{bmatrix} \quad (2.39)$$

dimana  $T'$  dan  $T''$  masing-masing diperoleh dari Persamaan 2.62 dan 2.63.

Bentuk medan listrik insiden pada koordinat kartesian lokal ditransformasikan ke koordinat bola lokal untuk mendapatkan  $E_{i\theta''}$  dan  $E_{i\phi''}$  adalah,

$$\begin{aligned} E_{i\theta''} &= E_{ix''} \cos \theta'' \cos \phi'' + E_{iy''} \cos \theta'' \sin \phi'' - E_{iz''} \sin \theta'' \\ E_{i\phi''} &= -E_{ix''} \sin \phi'' + E_{iy''} \cos \phi'' \end{aligned} \quad (2.40)$$

### 2.4.3 RCS dengan Faktor Polarisasi Penerima

Karena proses perhitungan tiap segitiga dilakukan pada koordinat lokal, maka hasil kali faktor polarisasi  $\Upsilon$  dan  $I_c$ , yaitu  $\Lambda$ , juga dihitung pada koordinat

lokal. Nilai  $\Lambda$  pada koordinat kartesian lokal adalah.

$$\begin{aligned}\Lambda_{x''} &= \Upsilon_{x''} \cdot I_c \\ \Lambda_{y''} &= \Upsilon_{y''} \cdot I_c \\ \Lambda_{z''} &= \Upsilon_{z''} \cdot I_c\end{aligned}\tag{2.41}$$

Setelah nilai  $\Lambda$  pada koordinat kartesian lokal dihitung, nilai  $\Lambda$  dibentuk kembali pada koordinat kartesian global, yaitu:

$$\begin{aligned}\Lambda_x &= (T')^{-1}(T'')^{-1}\Lambda_{x''} \\ \Lambda_y &= (T')^{-1}(T'')^{-1}\Lambda_{y''} \\ \Lambda_z &= (T')^{-1}(T'')^{-1}\Lambda_{z''}\end{aligned}\tag{2.42}$$

Nilai  $\Lambda$  pada koordinat bola global

$$\begin{aligned}\Lambda_\theta &= \Lambda_x \cos \theta \cos \phi + \Lambda_y \cos \theta \sin \phi - \Lambda_z \sin \theta \\ \Lambda_\phi &= -\Lambda_x \sin \phi + \Lambda_y \cos \phi\end{aligned}\tag{2.43}$$

Nilai  $\Lambda$  koordinat bola global ini digunakan untuk penentuan nilai RCS pada polarisasi penerima  $\theta$  atau  $\phi$ . Nilai RCS dengan polarisasi insiden  $q$  dan polarisasi penerima  $\theta$  adalah,

$$\sigma_{\theta q} = \frac{4\pi}{\lambda^2} \left| \sum_{m=1}^M (\Lambda_\theta)_m \right|^2\tag{2.44}$$

Nilai RCS dengan polarisasi insiden  $q$  dan polarisasi penerima  $\phi$  adalah,

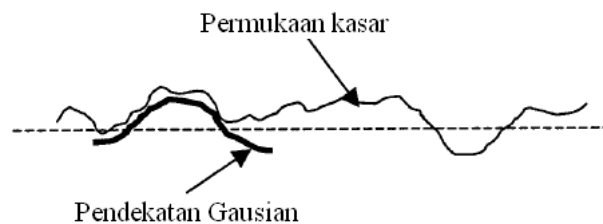
$$\sigma_{\phi q} = \frac{4\pi}{\lambda^2} \left| \sum_{m=1}^M (\Lambda_\phi)_m \right|^2\tag{2.45}$$

#### 2.4.4 Perhitungan Difusi

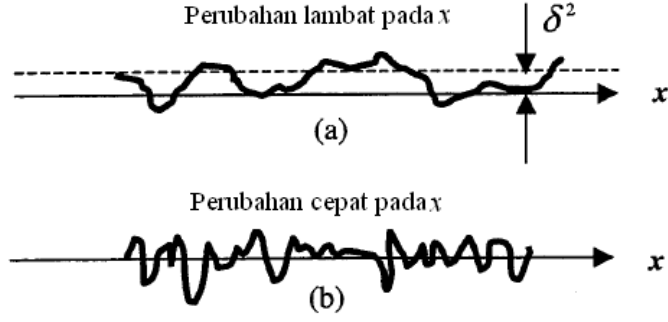
Target radar bisa jadi memiliki kekasaran permukaan atau ketidak-sempurnaan material, lihat Gambar 2.4 dan 2.5. Untuk itu ditambahkan perhitungan komponen difusi pada perhitungan RCS. Secara sederhana perhitungan RCS total dinyatakan dengan,

$$\sigma_{Total} = \sigma_{PO} + \sigma_{difusi} \quad (2.46)$$

Penambahan komponen difusi dilakukan dengan diasumsikan penyimpangan permukaan merupakan bentukan bentuk Gaussian sebagaimana diperlihatkan pada Gambar 2.4. Untuk memperluas bentuk Gaussian dikarakteristikan dengan jarak korelasi  $c$  (dalam meter), yang merupakan jarak rata-rata dimana penyimpangan terjadi. Jarak korelasi besar mengakibatkan perubahan yang lambat pada kesalahan permukaan. Sedangkan jarak korelasi kecil mengakibatkan kesalahan perubahan cepat. Variasi ketidak-tentuan ini dinotasikan dengan  $\delta^2$  diperlihatkan pada Gambar 2.5. Pada Gambar 2.5 bagian (a) memiliki jarak korelasi besar, sedangkan pada bagian (b) memiliki jarak korelasi kecil.



Gambar 2.4: Pendekatan Gaussian pada Permukaan Kasar



Gambar 2.5: Jarak Korelasi

Menurut [3], perhitungan RCS dengan komponen difusi adalah,

$$\sigma = \frac{4\pi}{\lambda^2} e^{-4k^2\delta^2} \left[ \left| \sum_{m=1}^M \Lambda_m \right|^2 + \sqrt{1 - e^{-4k^2\delta^2}} \sum_{m=1}^M |(\Lambda_D)_m| \right] \quad (2.47)$$

dimana

$$\Lambda_D = \Upsilon \cdot E_D \quad (2.48)$$

$\delta^2$  merupakan standar deviasi kekasaran permukaan materi.  $E_D$  merupakan komponen difusi pada tiap segitiga. Komponen difusi ini dihitung dengan menggunakan formula berikut ini,

$$E_D = 4\pi k^2 c^2 \delta^2 \cdot S_c \cdot \cos^2 \theta \cdot \exp \left( \left[ \frac{c^2 \pi \cdot \sin \theta}{\lambda} \right]^2 \right) \quad (2.49)$$

dimana  $c$  merupakan jarak korelasi.

RCS untuk polarisasi insiden  $q$  dan polarisasi penerima  $\theta$  dapat dinyatakan dengan

$$\sigma_{\theta q} = \frac{4\pi}{\lambda^2} e^{-4k^2\delta^2} \left[ \left| \sum_{m=1}^M (\Lambda_{\theta})_m \right|^2 + \sqrt{1 - e^{-4k^2\delta^2}} \sum_{m=1}^M |(\Lambda_{D\theta})_m| \right] \quad (2.50)$$

Sedangkan RCS untuk polarisasi insiden  $q$  dan polarisasi penerima  $\phi$  dinyatakan

dengan

$$\sigma_{\phi q} = \frac{4\pi}{\lambda^2} e^{-4k^2\delta^2} \left[ \left| \sum_{m=1}^M (\Lambda_{\phi})_m \right|^2 + \sqrt{1 - e^{-4k^2\delta^2}} \sum_{m=1}^M |(\Lambda_{D\phi})_m| \right] \quad (2.51)$$

## 2.5 Evaluasi Integral Difraksi Elektromagnetik

Penyelesaian numerik untuk integrasi dalam bentuk

$$I_c = \int_0^1 \int_0^{1-p} C(p, q) e^{jD(p, q)} dq dp \quad (2.52)$$

dimana  $C(p, q) = C_p p + C_q q + C_0$ ,  $D(p, q) = D_p p + D_q q + D_0$  dan geometri integrasi ditunjukkan pada Gambar 2.6. Penyelesaian ini ditunjukkan pada [18].

Untuk kasus perhitungan medan *scatter* dengan  $I_c = \iint_S e^{jk(g+h)} dS$ , pada [9] dijelaskan bahwa

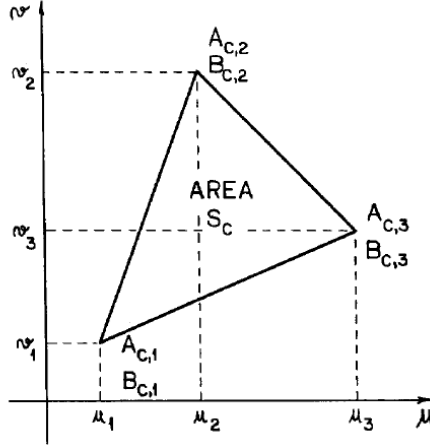
$$C_p = C_q = 0, C_0 = 1 \quad (2.53)$$

dan

$$\begin{aligned} D_p &= k [(x_1 - x_3)(u + u_i) + (y_1 - y_3)(v + v_i) + (z_1 - z_3)(w + w_i)] \\ D_q &= k [(x_2 - x_3)(u + u_i) + (y_2 - y_3)(v + v_i) + (z_2 - z_3)(w + w_i)] \\ D_0 &= k [x_3(u + u_i) + y_3(v + v_i) + z_3(w + w_i)] \end{aligned} \quad (2.54)$$

$u_i$ ,  $v_i$  dan  $w_i$  merupakan kosinus arah titik insiden. Pada kasus radar mono-statis ditulis  $u = u_i$ ,  $v = v_i$  dan  $w = w_i$ .





Gambar 2.6: Geometri segitiga integrasi

Bentuk umum penyelesaian  $I_c$  selanjutnya secara analitis digunakan persamaan berikut ini,

$$I_c = 2S_c e^{jD_0} \left\{ e^{jD_p} \left[ \frac{C_0}{D_p(D_q - D_p)} \right] - e^{jD_q} \left[ \frac{C_0}{D_p(D_q - D_p)} \right] - \frac{C_0}{D_p D_q} \right\} \quad (2.55)$$

dimana  $S_c$  merupakan luasan segitiga.

Untuk menghindari kesalahan numerik yang besar, digunakan penyelesaian-penyelesaian berikut ini:

1. Kasus 1: Jika  $|D_p| < L_t$  dan  $|D_q| \geq L_t$ , tanpa batasan pada  $|D_q - D_p|$ , maka

$$I_c = 2S_c \frac{e^{jD_0}}{jD_q} \sum_{n=0}^N \frac{(jD_p)^n}{n!} \left\{ \frac{-C_0}{n+1} + e^{jD_q} C_0 G(n, -D_q) \right\} \quad (2.56)$$

2. Kasus 2: Jika  $|D_p| < L_t$  dan  $|D_q| < L_t$  tanpa batasan pada  $|D_q - D_p|$ , maka

$$I_c = 2S_c e^{jD_0} \sum_{n=0}^N \sum_{m=0}^M \frac{C_0 (jD_p)^n (jD_q)^m}{(n+m+2)!} \quad (2.57)$$

3. Kasus 3: Jika  $|D_p| \geq L_t$  dan  $|D_q| < L_t$  tanpa batasan pada  $|D_q - D_p|$ ,

maka

$$I_c = 2S_c e^{jD_0} e^{jD_p} \sum_{n=0}^N \left[ \frac{(jD_q)^n}{n!} \cdot \frac{C_0}{n+1} \cdot G(n+1, -D_p) \right] \quad (2.58)$$

4. Kasus 4: Jika  $|D_p| \geq L_t$ ,  $|D_q| \geq L_t$ ,  $|D_q - D_p| < L_t$ , maka

$$I_c = 2S_c \frac{e^{jD_0}}{jD_q} \sum_{n=0}^N \frac{[j(D_p - D_q)]^n}{n!} \left\{ (-C_0) G(n, D_q) + \frac{e^{jD_q} C_0}{n+1} \right\} \quad (2.59)$$

dimana

$$G(n, w) = \int_0^1 s^n e^{jws} ds. \quad (2.60)$$

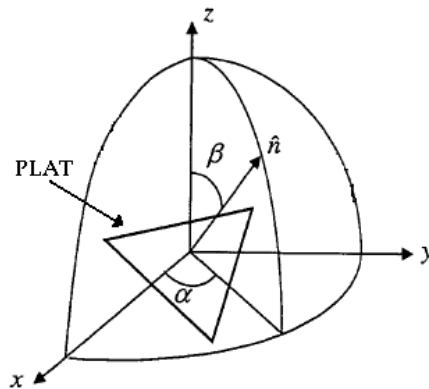
Fungsi  $G(n, w)$  dapat dievaluasi secara rekursif dengan relasi

$$G(n, w) = \frac{e^{jw} - nG(n-1, w)}{jw}, \quad (2.61)$$

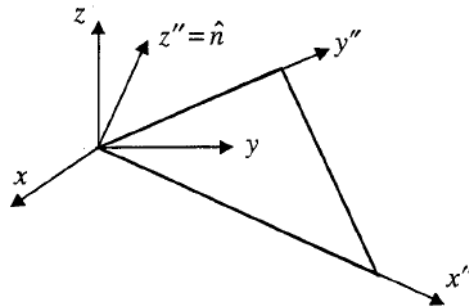
untuk  $n \geq 1$  dan nilai awal  $G(0, w) = (e^{jw} - 1) / (jw)$ . Untuk mendapatkan hasil akurat dapat dibentuk  $L_t = 0.05$  dan  $N, M = 2$  [18].

## 2.6 Transformasi Koordinat

Secara umum, sistem koordinat lokal segitiga tidak sama dengan sistem koordinat global. Satu rangkaian transformasi harus dilakukan untuk memperoleh hubungan antara variabel koordinat lokal dan global serta vektor unit. Gambar 2.7 menunjukkan suatu segitiga dengan arah yang bersesuaian. Dimana koordinat global dinyatakan dalam  $(x, y, z)$  dan koordinat lokal dalam  $(\tilde{x}, \tilde{y}, \tilde{z})$ . Gambar 2.8 menunjukkan sumbu  $\tilde{z}$  searah dengan normal segitiga, dan sumbu  $\tilde{x}$  dan  $\tilde{y}$  searah dengan sisi segitiga.

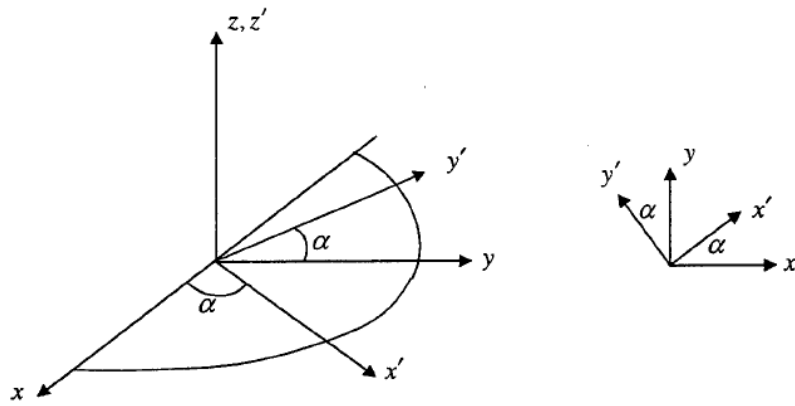


Gambar 2.7: Pelat segitiga dengan orientasi yang disesuaikan

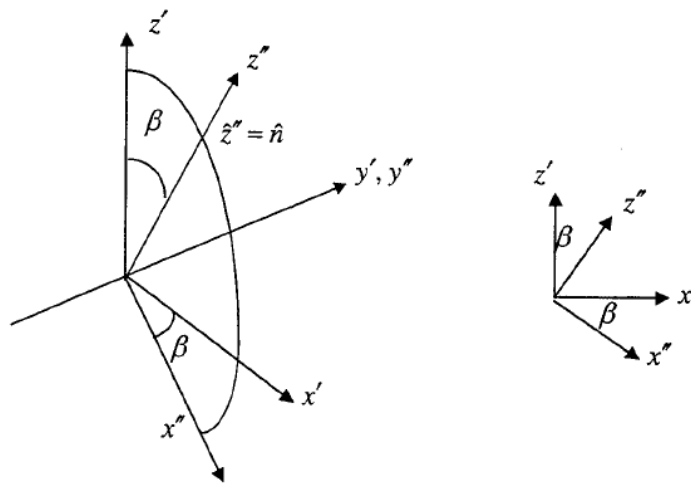


Gambar 2.8: Sudut rotasi segitiga

Dua variabel utama yang ada pada rangkaian rotasi variabel koordinat global ini adalah  $\alpha$  dan  $\beta$ .  $\alpha$  merupakan rotasi terhadap sumbu  $z$ , dan  $\beta$  merupakan rotasi terhadap sumbu  $y$ . Secara komputasi numerik, nilai  $\alpha = \text{atan2}(n_y, n_x)$  dan  $\beta = \cos^{-1}(\hat{z} \cdot \hat{n})$ .  $\text{atan2}$  merupakan fungsi perhitungan arcus tangen yang memberikan kuadran yang disesuaikan. Rotasi dengan sudut  $\alpha$  ditunjukkan pada Gambar 2.9, dan Rotasi dengan sudut  $\beta$  ditunjukkan pada Gambar 2.10.



Gambar 2.9: Rotasi pertama dengan sudut  $\alpha$  pada sumbu  $z$



Gambar 2.10: Rotasi pertama dengan sudut  $\beta$  pada sumbu  $z$

Matriks transformasi yang dibentuk dari rotasi pada gambar 2.9 adalah sebagai berikut:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{T'} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (2.62)$$

Matriks transformasi yang dibentuk dari rotasi pada gambar 2.10 adalah sebagai berikut:

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix}}_{T''} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \quad (2.63)$$

Matriks kosinus arah

$$\begin{bmatrix} u'' \\ v'' \\ w'' \end{bmatrix} = T'' T' \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (2.64)$$

dimana nilai  $u, v, w$  dijelaskan pada Persamaan 2.8, dan

$$\begin{aligned} u'' &= \sin \theta'' \cos \phi'' \\ v'' &= \sin \theta'' \sin \phi'' \\ w'' &= \cos \theta'' \end{aligned} \quad (2.65)$$

### 2.6.1 Kosinus Arah

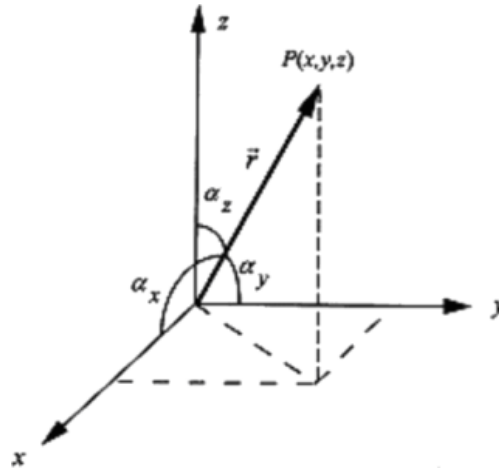
Kosinus arah (*Direction Cosines*) merupakan kuantitas yang melibatkan koordinat bola. Untuk vektor posisi dari titik pusat ke titik  $P$ , kosinus arah merupakan kosinus dari tiga sudut yang dibentuk vektor yang berkaitan dengan sumbu kartesian. Menurut [14] sudut-sudut yang diperlihatkan pada Gambar 2.11 dapat didefinisikan sebagai berikut:

$$x \text{ kosinus arah: } u \text{ atau } \cos \alpha \text{ atau } \cos \alpha_x = \sin \theta \cos \phi$$

$$y \text{ kosinus arah: } v \text{ atau } \cos \beta \text{ atau } \cos \alpha_y = \sin \theta \sin \phi$$

$$z \text{ kosinus arah: } w \text{ atau } \cos \gamma \text{ atau } \cos \alpha_z = \cos \theta$$

Pada penulisan ini, notasi  $u, v, w$  masing-masing digunakan untuk merepresentasikan kosinus arah  $x, y, z$ .



Gambar 2.11: Kosinus Arah

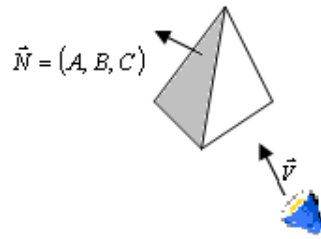
## 2.7 *Back-face Culling*

Pada aplikasi komputer grafik, *back-face culling* digunakan untuk menentukan apakah suatu poligon dari obyek grafis berada pada bagian yang terlihat oleh kamera (mata) atau tidak. [8, 12, 31] Dengan prinsip ini, bisa digunakan untuk membedakan bagian obyek yang teriluminasi dengan bagian bayangan pada PO.

Jika vektor  $\vec{N}$  merupakan vektor normal dari suatu poligon dengan komponen kartesian  $(A, B, C)$ , dan  $\vec{V}$  vektor arah pandangan, poligon merupakan bayangan jika

$$\vec{V} \cdot \vec{N} > 0 \quad (2.66)$$

Untuk poligon yang dilihat pada arah  $\vec{V}$  dengan normal  $\vec{N}$  diperlihatkan pada Gambar 2.12.



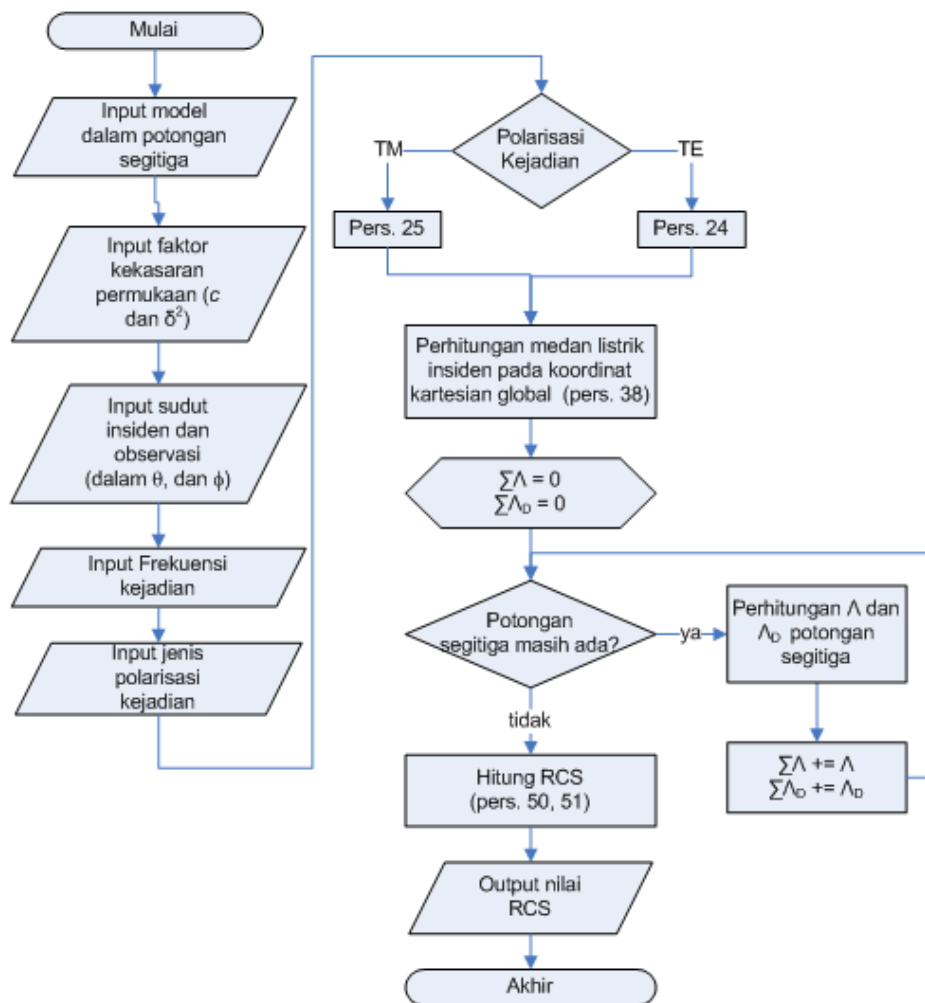
Gambar 2.12: Pandangan arah  $\vec{V}$  pada poligon dengan normal  $\vec{N}$

## 2.8 Alur Perhitungan RCS Target

Untuk mempermudah proses perhitungan pada pembuatan software ini dibuat diagram alur proses perhitungan. Proses perhitungan RCS target dalam diagram alur, dibentuk dalam dua bagian. Diagram ini terdiri atas alur utama dan alur untuk perhitungan tiap segitiga. Alur utama merupakan Gambaran proses perhitungan secara keseluruhan. Alur utama diperlihatkan pada Gambar 2.13. Sedangkan alur perhitungan tiap segitiga diperlihatkan pada Gambar 2.14.

Alur utama nantinya akan dihasilkan nilai RCS ( $\sigma$ ) dari desain obyek pada sudut  $\theta$  dan  $\phi$  tertentu. Pada alur utama, diinputkan data sebagai berikut:

1. Model dalam potongan segitiga.
2. Sudut insiden dan observasi dalam  $\theta$  dan  $\phi$ .
3. Frekuensi kejadian
4. Jenis polarisasi kejadian.
5. Faktor kekasaran permukaan (jarak korelasi  $c$  dan standar deviasi).

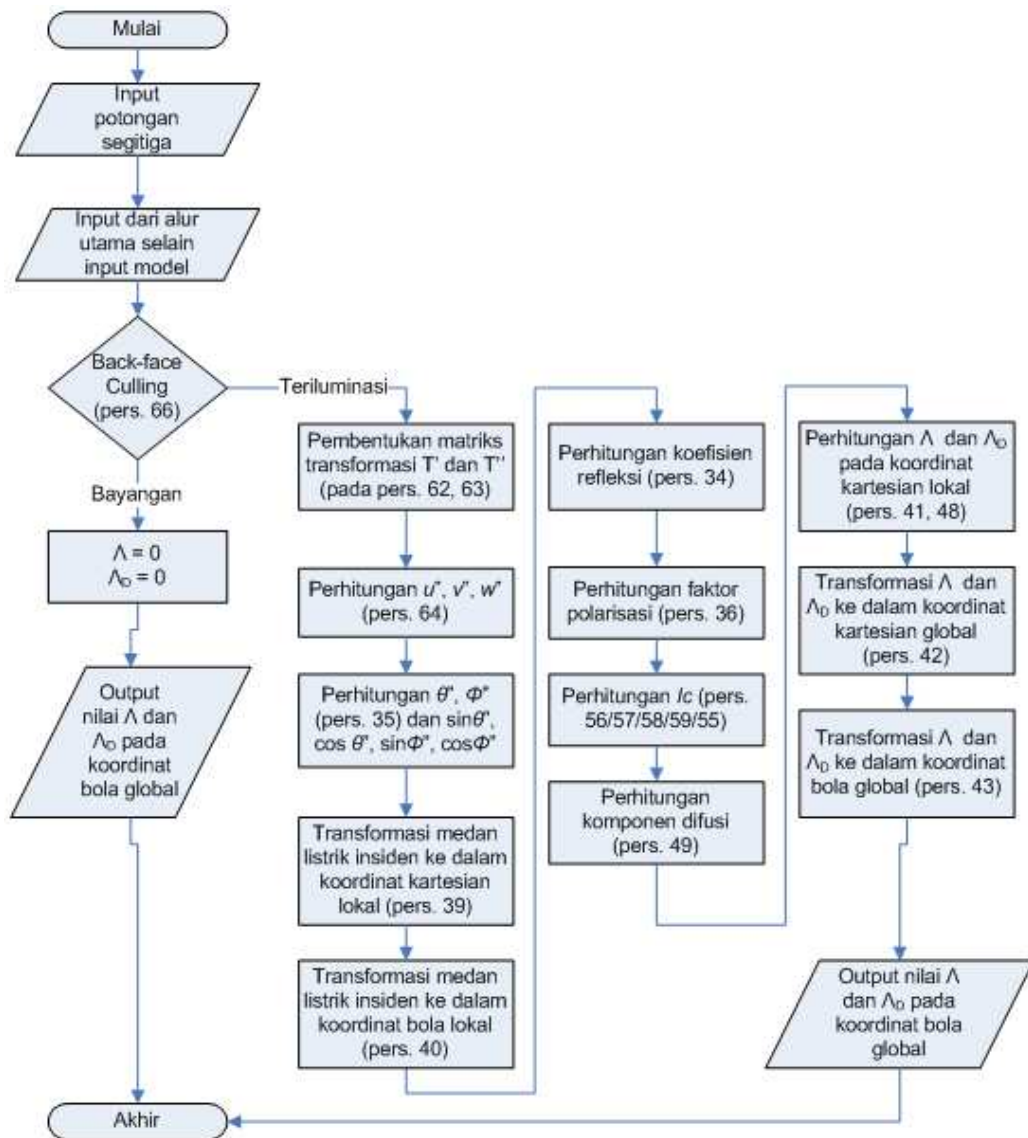


Gambar 2.13: Alur perhitungan utama

Di alur utama terdapat perhitungan  $\Lambda$  untuk tiap segitiga, yang dijelaskan pada alur perhitungan segitiga.

Alur tiap potongan segitiga akan menghasilkan nilai  $\Lambda$  untuk segitiga. Pada alur ini diinputkan potongan segitiga (tunggal) dan input dari alur utama selain data model keseluruhan.





Gambar 2.14: Alur perhitungan tiap potongan segitiga

# BAB 3

## Perancangan dan Pembuatan Software

Pada bab ini akan dibahas hal-hal yang berkaitan dengan perancangan dan pembuatan software RCS target yang selanjutnya diberi nama targetRCS2. Software dibuat dengan menggunakan bahasa pemrograman C# dengan .Net Framework 2.0 [17]. Untuk keperluan komputasi numerik, seperti permasalahan matriks dan vektor digunakan pustaka dnAnalytic [7].

Hal yang akan dibahas pada bab ini meliputi spesifikasi dasar software, format file input, representasi model, antarmuka pengguna (*user interface*), Modifikasi C# OpenGL, dan pembentukan kode PO.

### 3.1 Spesifikasi Software

Pada sub-bab ini dijelaskan kerangka acuan untuk pembuatan software, yang meliputi kemampuan, input dan output yang ada pada software.

Kemampuan software yang berkaitan dengan perhitungan RCS dijabarkan sebagai berikut:

1. Perhitungan RCS berdasarkan jenis polarisasi, yaitu polarisasi kejadian dan polarisasi penerima radar.
2. Penggunaan model radar monostatis atau bistatis.
3. Hasil perhitungan dapat dilihat di layar atau disimpan pada file teks.
4. Software mampu mendapatkan pola RCS obyek. Pola ini bisa berdasarkan kisaran sudut observasi tertentu, atau kisaran frekuensi tertentu.

5. Hambatan dan kekasaran permukaan mempunyai peranan pada perhitungan.
6. Pola hasil perhitungan RCS dapat dilihat pada koordinat kartesian dan koordinat polar.

Kemampuan software yang berkaitan dengan model obyek yang dijadikan input adalah sebagai berikut:

1. Membaca file input dari AutoCAD yang telah disederhanakan.
2. Menampilkan model obyek dari file input secara 3-dimensi.

Penjelasan mengenai file input yang digunakan terdapat pada sub-bab 3.2.

Berdasarkan acuan kemampuan software di atas dibentuk input dan output dasar untuk software. Input pada software meliputi hal-hal sebagai berikut:

1. Input model
2. Sudut observasi dan sudut kejadian / insiden
3. Frekuensi
4. Koefisien refleksi
5. Kekasaran permukaan berdasarkan nilai jarak korelasi dan standar deviasi pada permukaan
6. Polarisasi kejadian

Sedangkan output yang dihasilkan oleh software adalah sebagai berikut:

1. Menampilkan obyek dalam tampilan 3-dimensi dengan OpenGL
2. Hasil perhitungan RCS ditampilkan ke layar, atau disimpan ke dalam file teks.
3. Sketsa hasil perhitungan RCS pada koordinat kartesian dan koordinat polar

## 3.2 File Input

Pada proses pengerjaan PO ini file input merupakan data mengenai potongan-potongan segitiga yang membangun obyek. Karena untuk penyeleksian bagian yang bayangan menggunakan metode *back-face culling*, maka susunan titik setiap segitiga yang membangun obyek harus dibuat mengikuti aturan tangan kanan.

File input ini berasal dari file format data model (.XMO) yang merupakan file representasi obyek khusus untuk penelitian ini (bentukan sendiri). File input juga berasal dari format file AutoCAD (.DXF) dan file StereoLithography (StL). File DXF disederhanakan dalam bentuk 3DFace yang merupakan data potongan-potongan segitiga.

### 3.2.1 Format File .XMO

File .XMO merupakan file dengan format Extensible Markup Language (XML). XML merupakan teknologi yang digunakan untuk mendeskripsikan data, ditulis dalam bentuk teks sederhana agar dapat didistribusikan melalui internet [11] [24]. Dokumen ini menggunakan serangkaian *tag* untuk mendeskripsikan berbagai elemen data. *Tag* merupakan kata sederhana pada kurung siku (< >). Hampir semua elemen XML memiliki *tag* pembuka dan *tag* penutup. Suatu *tag* dapat memuat teks atau *tag* lainnya.

File .XMO dibuat berdasarkan format data yang diperlihatkan pada Tabel 3.1.

Tabel 3.1: Format data file .XMO

No	Field	Tipe	Keterangan
1	ID	auto number	Sebagai dasar penomoran potongan segitiga
2	X1	float	nilai X titik ke-1
3	Y1	float	nilai Y titik ke-1
4	Z1	float	nilai Z titik ke-1
5	X2	float	nilai X titik ke-2
6	Y2	float	nilai Y titik ke-2
7	Z2	float	nilai Z titik ke-2
8	X3	float	nilai X titik ke-3
9	Y3	float	nilai Y titik ke-3
10	Z3	float	nilai Z titik ke-3
11	Color	int16	warna potongan berdasarkan index pewarnaan AutoCAD.

Format warna menggunakan pengindeksan warna AutoCAD, jumlah warna yang digunakan sebanyak 256 warna. Pewarnaan ini bisa digunakan untuk membedakan sifat jenis permukaan. Format penyimpanan permukaan dijelaskan pada subbab 3.2.4.

Sebagai contoh file XMO untuk menyimpan satu segitiga dengan koordinat (0,0,0), (1,1,0), (0,1,0) diperlihatkan pada Gambar 3.1.

```

<?xml version="1.0" standalone="yes"?>
<MaterialsDataSet
xmlns="http://tempuri.org/MaterialsDataSet.xsd">
  <Facets>
    <ID>1</ID>
    <X1>0</X1>
    <Y1>0</Y1>
    <Z1>0</Z1>
    <X2>1</X2>
    <Y2>1</Y2>
    <Z2>0</Z2>
    <X3>0</X3>
    <Y3>1</Y3>
    <Z3>0</Z3>
    <Color>12</Color>
  </Facets>
</MaterialsDataSet>

```

Gambar 3.1: Contoh file XMO

### 3.2.2 File AutoCAD

Format DXF merupakan *tagged data* yang merepresentasikan seluruh informasi gambar dari AutoCAD [1]. *Tagged data* maksudnya adalah setiap elemen data pada file didahului oleh bilangan bulat yang disebut kode grup. Suatu nilai kode grup mengindikasikan tipe elemen data yang mengikuti.

File DXF yang digunakan pada penelitian ini disederhanakan dengan hanya memuat entitas 3Dface, dan spesifikasi paling mendasar dari penulisan file DXF. Spesifikasi dasar ini ditujukan agar AutoCAD masih bisa membaca file DXF yang telah dibuat, apabila pembuatan file dilakukan diluar software AutoCAD.

Struktur grup kode yang digunakan untuk 3Dface diperlihatkan pada Tabel 3.2.

Tabel 3.2: Struktur grup kode 3DFace

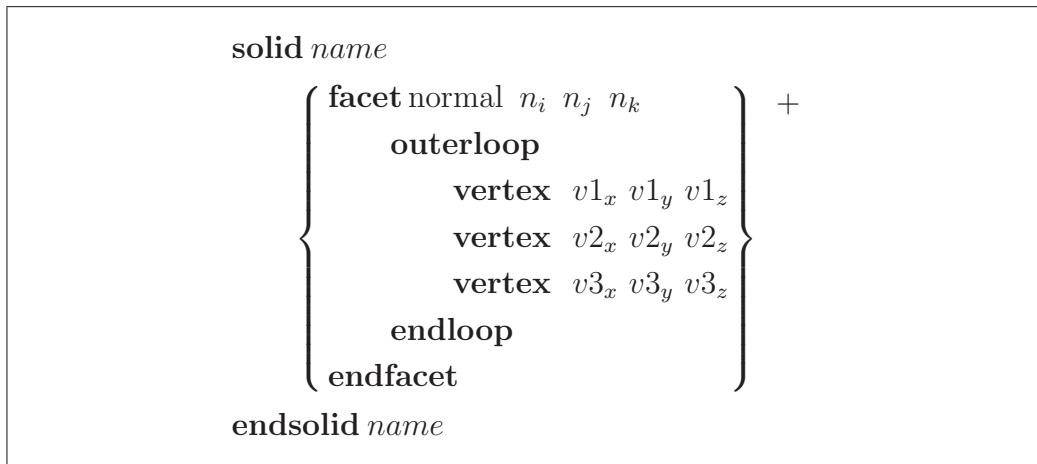
Kode Grup	Keterangan
100	Subclass penanda (AcDbFace)
62	Nomor warna
10	Titik pertama, nilai X
20, 30	Titik pertama, nilai Y dan Z
11	Titik kedua, nilai X
21, 31	Titik kedua, nilai Y dan Z
12	Titik ketiga, nilai X
22, 32	Titik ketiga, nilai Y dan Z
13	Titik keempat, nilai X, (nilai titik keempat bisa sama dengan titik ketiga apabila pada penggambaran hanya diperlukan tiga titik)
23, 33	Titik keempat, nilai Y dan Z
70	Penanda adanya edge yang tidak terlihat

### 3.2.3 File StereoLithography

File StereoLithography (StL) merupakan file yang berisi representasi segitiga dari suatu permukaan geometri dalam 3-dimensi yang dibuat oleh 3D System [29]. Dimana permukaan dibentuk dalam potongan-potongan kecil segitiga (*facet*). Setiap facet dinyatakan dengan arah tegak-lurusnya, dan tiga titik yang merepresentasikan *vertex* segitiga.

File StL bisa disimpan dalam bentuk format ASCII maupun bentuk *binary*. Dalam bentuk ASCII, format file StL diperlihatkan pada Gambar 3.2.

Tulisan yang ditebalkan (*bold face*) mengindikasikan kata kunci, dan harus ditulis dalam huruf kecil. Notasi  $\{ \}^+$  menyatakan bahwa bagian tersebut bisa diulang lebih dari satu kali. Simbol dengan huruf miring *italics* merupakan variabel dalam nilai tertentu. Data numerik pada untuk **facet normal** dan **vertex** merupakan tipe data float presisi tunggal (*single precision floats*).



Gambar 3.2: Struktur file StL ASCII

Format file StL dalam bentuk binary diperlihatkan pada Tabel 3.3. Bagian nomor 3 - 15 bisa pada Tabel 3.3 diulang lebih dari satu kali.

Tabel 3.3: Format StL Binary

NO	Jumlah Byte	Type Data	Keterangan
1	80	ASCII	Header. (Tidak ada ketentuan)
2	4	integer	Jumlah facets pada file
3	4	float	$i$ untuk normal
4	4	float	$j$ untuk normal
5	4	float	$k$ untuk normal
6	4	float	$x$ untuk vertex ke-1
7	4	float	$y$ untuk vertex ke-1
8	4	float	$z$ untuk vertex ke-1
9	4	float	$x$ untuk vertex ke-2
10	4	float	$y$ untuk vertex ke-2
11	4	float	$z$ untuk vertex ke-2
12	4	float	$x$ untuk vertex ke-3
13	4	float	$y$ untuk vertex ke-3
14	4	float	$z$ untuk vertex ke-3
15	2	integer	Atribut byte (diset 0



### 3.2.4 File Input Sifat Permukaan

File input sifat permukaan ini digunakan untuk menyimpan data material yang membangun permukaan obyek. Format file ini menggunakan format file XML. File input permukaan ini terdiri atas file LayerValue dan LayerMaterials. File LayerValue digunakan untuk menyimpan data utama permukaan. File LayerMaterials digunakan untuk menyimpan data detail material-material yang menyusun permukaan beserta sifat-sifatnya.

Struktur untuk data yang disimpan pada file LayerValue, diperlihatkan pada Tabel 3.4.

Tabel 3.4: Struktur data file LayerValue

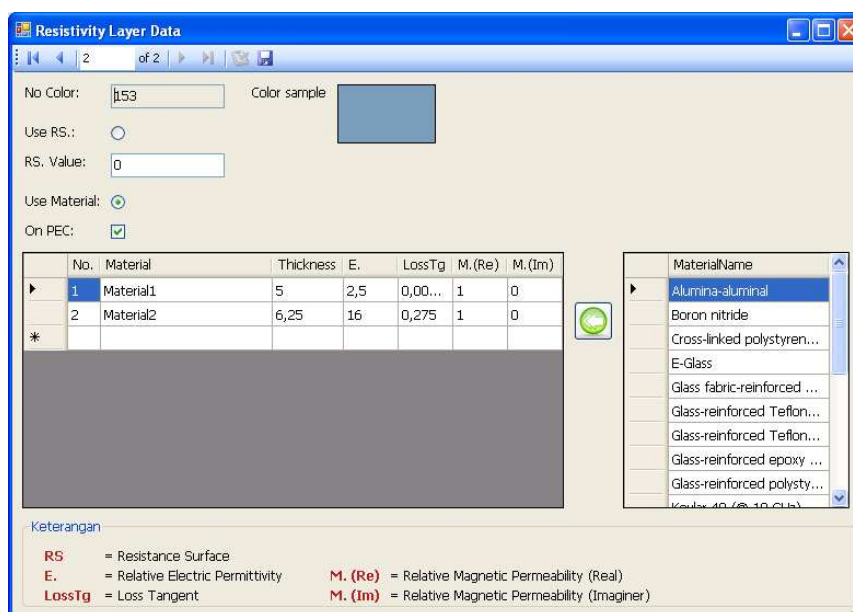
No	Field	Tipe	Keterangan
1	NoColor	integer	No warna permukaan sesuai index pewarnaan AutoCAD
2	UseResistanceSurface	boolean	Status perhitungan resistansi permukaan menggunakan data ResistanceSurfaceValue
3	ResistanceSurfaceValue	double	Nilai ResistanceSurfaceValue untuk perhitungan resistansi permukaan
4	UseMaterial	boolean	Status perhitungan resistansi permukaan menggunakan data material penyusun permukaan
5	OnPEC	boolean	Status bagian terdalam dari permukaan merupakan PEC

Struktur untuk data yang disimpan pada file LayerMaterials, diperlihatkan pada Tabel 3.5.

Tabel 3.5: Struktur data file LayerMaterials

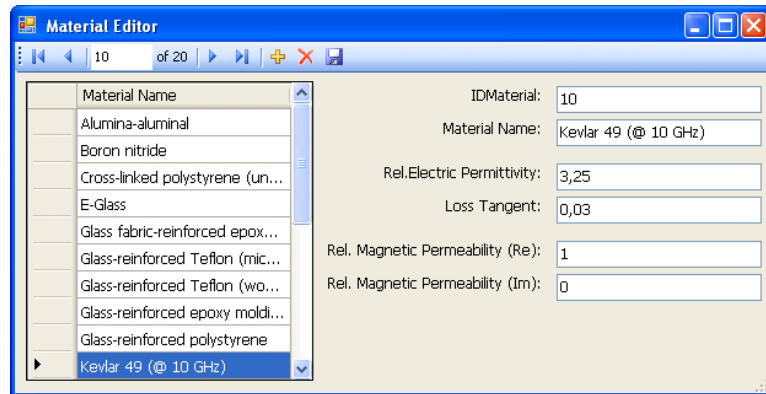
No.	Field	Tipe	Keterangan
1	NoColor	integer	nomor warna permukaan
2	MaterialNumber	integer	Nomor urut jenis material penyusun permukaan
3	MaterialName	teks	nama material
4	Relative-ElectricPermittivity	double	
5	LossTangent	double	
6	RelativeMagnetic-PermeabilityReal	double	
7	RelativeMagnetic-PermeabilityImaginer	double	
8	Thickness	double	Ketebalan material (dalam mm)

File input permukaan ini tidak mesti harus ada, apabila pengerjaan perhitungan hanya menggunakan data tahanan permukaan. Untuk pengolahan data input permukaan dibuat dialog sebagaimana diperlihatkan pada Gambar 3.3.



Gambar 3.3: Dialog pengolahan input permukaan

Inputan material dasar yang disediakan dibuat dengan menggunakan database MS-Access untuk memudahkan manajemen data material. Untuk pengolahan data input layar dibuat dialog sebagaimana diperlihatkan pada Gambar 3.4.



Gambar 3.4: Dialog input material dasar

### 3.3 Representasi Model dengan C# OpenGL Framework

OpenGL merupakan software interface untuk hardware grafik [19, 25, 30]. Interface ini terdiri atas lebih dari 150 perintah yang berbeda untuk menspesifikasikan obyek dan operasi yang diperlukan untuk menghasilkan aplikasi 3-dimensi yang interaktif. OpenGL diimplementasikan dalam berbagai platform hardware grafis.

OpenGL tidak menyediakan perintah level-tinggi, yaitu untuk mendeskripsikan obyek 3-dimensi. Untuk merepresentasikan bentuk yang relatif kompleks, seperti mobil, pesawat, atau molekul, model dibentuk dari bentuk geometris sederhana, yaitu : titik, garis, dan poligon.

Pustaka OpenGL yang digunakan pada software adalah C# OpenGL Framework [5] edisi *basic*. Versi ini dianggap sudah cukup untuk merepresentasikan model, dan berbagai operasi sederhana lainnya.

### 3.3.1 Modifikasi C# OpenGL Framework

Modifikasi Framework ini dilakukan untuk membantu proses perhitungan RCS menggunakan PO. Bagian yang dimodifikasi adalah *class* TriangularFace sebagai penyimpan data potongan segitiga. Penambahan ini meliputi penambahan data *public*, dan fungsi pengolahannya. Data yang ditambahkan pada C# OpenGL Framework sebagai berikut:

```
// public Data
public double area;      // luas segitiga
public double alpha;    // sudut rotasi
public double beta;     // sudut elevasi
public double sinA;     // sin (alpha)
public double cosA;     // cos (alpha)
public double sinB;     // sin (beta)
public double cosB;     // cos (beta)
public short autoCadColor;
```

Fungsi yang ditambahkan pada C# OpenGL Framework sebagai berikut:

```
void CalcAreaNAngle()
{
    double triLenght = normal.Length ();
    area = Math.Abs(0.5 * triLenght);

    double xt = (double)normal.x / triLenght;
    double yt = (double)normal.y / triLenght;
    double zt = (double)normal.z / triLenght;

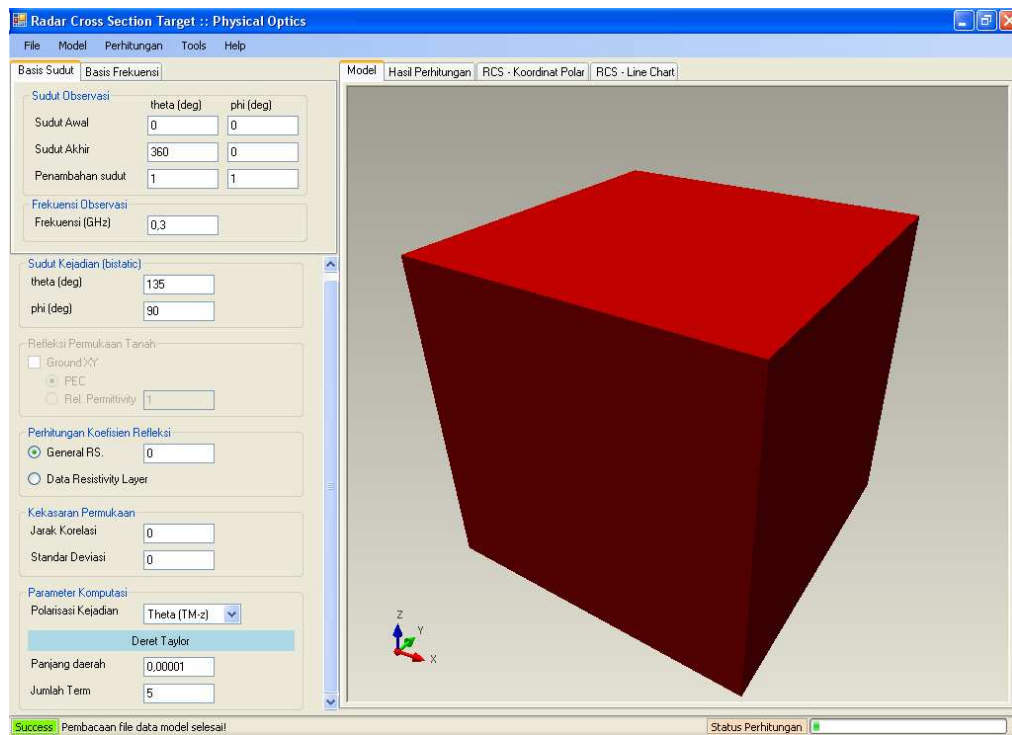
    alpha = Math.Atan2(yt, xt);
    beta = Math.Acos(zt);
    sinA = Math.Sin(alpha);
    cosA = Math.Cos(alpha);
    sinB = Math.Sin(beta);
    cosB = Math.Cos(beta);
}
```

### 3.4 Antarmuka Pengguna

Tampilan antarmuka software secara umum diperlihatkan pada Gambar 3.5.

Pada Gambar 3.5, software terbagi atas:

1. Menu, pada bagian atas software.
2. Input perhitungan RCS, pada bagian kiri software.
3. Output, pada bagian kanan software. Pada bagian output meliputi: tampilan obyek, hasil perhitungan, sketsa hasil perhitungan pada koordinat kartesian dan polar.



Gambar 3.5: Antarmuka software

Penjelasan menu yang ada pada bagian atas software dapat dilihat pada lampiran B.

### 3.4.1 Bagian Input

Bagian input terdiri atas dua bagian, bagian atas dan bagian bawah. Input bagian atas digunakan untuk data sudut observasi dan frekuensi. Input bagian bawah digunakan untuk data lainnya.

Basis Sudut		Basis Frekuensi	
<b>Sudut Observasi</b>			
Sudut Awal	theta (deg)	phi (deg)	
<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	
Sudut Akhir	<input type="text" value="360"/>	<input type="text" value="0"/>	
Penambahan sudut	<input type="text" value="1"/>	<input type="text" value="1"/>	
<b>Frekuensi Observasi</b>			
Frekuensi (GHz)	<input type="text" value="0,3"/>		

Basis Sudut		Basis Frekuensi	
<b>Frekuensi Observasi</b>		<b>Sudut Observ.</b>	
Frekuensi Awal (GHz)	<input type="text" value="0,1"/>	theta (deg)	<input type="text" value="0"/>
Frekuensi Akhir	<input type="text" value="10"/>	phi (deg)	<input type="text" value="0"/>
Penambahan Frekuensi	<input type="text" value="0,1"/>		

Gambar 3.6: Input software bagian atas

Input bagian atas diperlihatkan pada Gambar 3.6. Bagian input ini terdiri dua halaman yang digunakan untuk menentukan perhitungan RCS berbasis sudut atau berbasis frekuensi. Pada basis sudut, nilai sudut berada pada interval tertentu dengan besar frekuensi tetap. Sedangkan pada basis frekuensi, nilai frekuensi berada pada interval tertentu dengan arah sudut tetap.

Input bagian bawah diperlihatkan pada Gambar 3.7. Input bagian bawah ini terdiri atas beberapa bagian, yaitu sebagai berikut:

1. Sudut Kejadian (bistatik)

Bagian ini digunakan untuk perhitungan radar bistatik. Dimana pada perhitungan ini sudut kejadian (insiden) tetap, sedangkan sudut observasi bisa berubah (perhitungan berbasis sudut).

2. Refleksi Permukaan Tanah

Bagian ini digunakan untuk perhitungan RCS dengan asumsi obyek berada pada bidang XY (tanah) yang memantulkan gelombang radar. Bagian ini untuk pengembangan software selanjutnya.

The image shows a software interface with several input sections:

- Sudut Kejadian (bistatic):** theta (deg) is 135, phi (deg) is 90.
- Refleksi Permukaan Tanah:** Ground XY is unchecked, PEC is selected, and Rel. Permittivity is 1.
- Perhitungan Koefisien Refleksi:** General RS. is selected with a value of 0, and Data Resist. Layer has a Set Data button.
- Kekasaran Permukaan:** Jarak Korelasi is 0, and Standar Deviasi is 0.
- Parameter Komputasi:** Polarisasi Kejadian is Theta (TM-z), Deret Taylor is selected, Panjang daerah is 0,00001, and Jumlah Term is 5.

Gambar 3.7: Input software bagian bawah

### 3. Perhitungan Koefisien Refleksi

Pada bagian ini terdapat dua pilihan, yaitu seluruh permukaan dianggap memiliki tahanan permukaan yang sama (*General RS*) atau tiap jenis permukaan bisa memiliki tahanan permukaan yang berbeda.

### 4. Kekasaran Permukaan

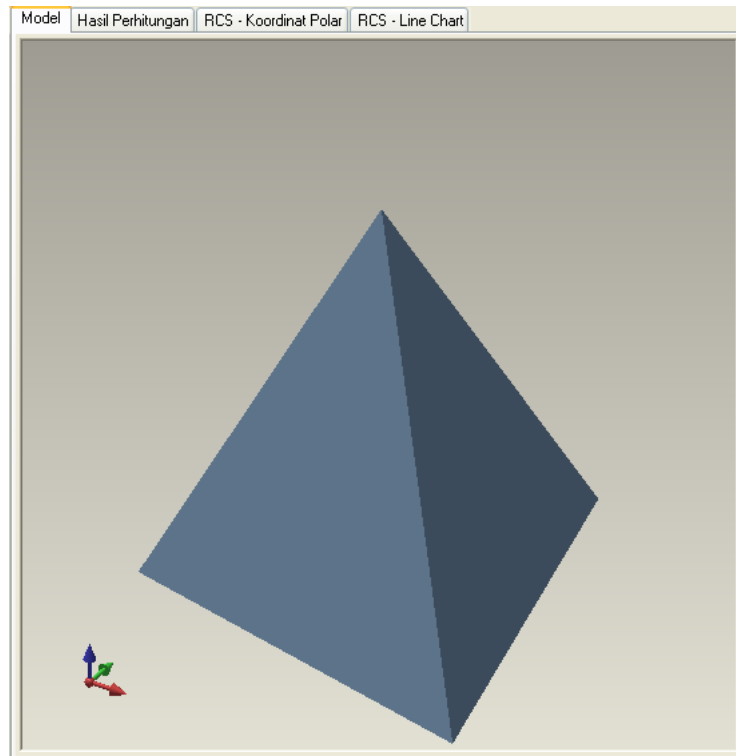
Bagian ini digunakan untuk input jarak korelasi dan standar deviasi, apabila permukaan dianggap memiliki tingkat kekasaran tertentu yang sama.

### 5. Parameter komputasi

Bagian ini digunakan untuk menentukan jenis polarisasi kejadian (insiden). Pada bagian ini juga digunakan untuk input panjang daerah ( $L_t$ ) dan jumlah suku (*term*), pada perhitungan  $I_c$ .

### 3.4.2 Tampilan Model dan Hasil Perhitungan

Tampilan Model dan Hasil Perhitungan merupakan bagian output utama software. Tampilan model diperlihatkan pada Gambar 3.8, dimana pada kasus ini digunakan bentuk limas.



Gambar 3.8: Tampilan model pada software

Hasil perhitungan RCS dalam bentuk tabel diperlihatkan pada Gambar 3.9. Pada Gambar 3.9, dengan menggunakan nilai  $\phi = 0^\circ$  dan nilai  $\theta$  tertentu dapat dilihat nilai RCS yang dalam dBsm. RCStheta digunakan untuk menunjukkan nilai RCS dengan polarisasi penerima  $\theta$  ( $\sigma_{\theta q}$ ), sedangkan RCSphi digunakan untuk menunjukkan nilai RCS dengan polarisasi penerima  $\phi$  ( $\sigma_{\phi q}$ ).  $q$  merupakan polarisasi insiden yang ditentukan pada proses input. Sebagai contoh pada  $\phi = 0$ ,  $\theta = 0,000$  diperoleh  $\sigma_{\theta q} = -7,917449E-001$  dBsm dan  $\sigma_{\phi q} = -1,60000E+002$  dBsm. Tombol Save digunakan untuk menyimpan hasil perhitungan dalam bentuk file teks.

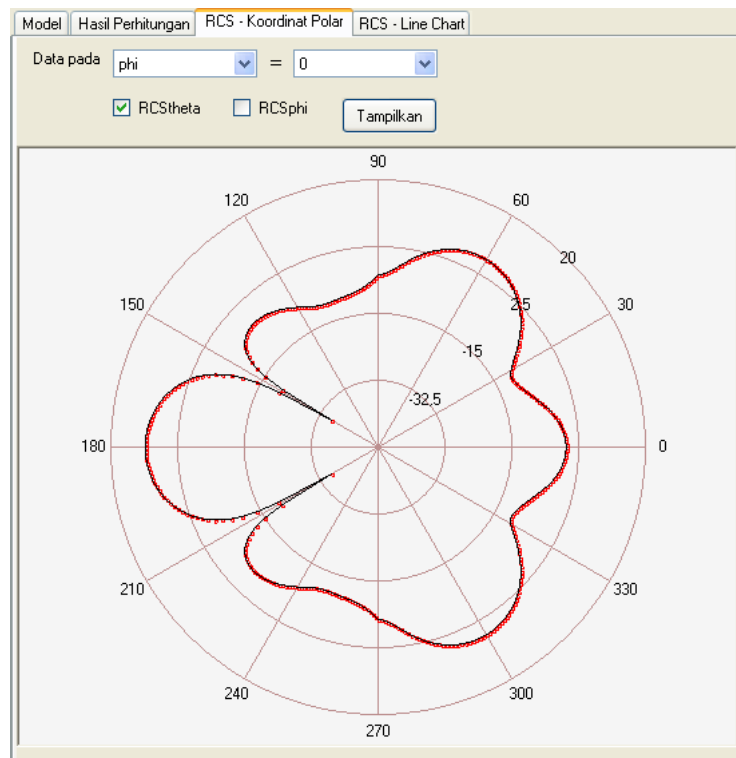


theta	RCStheta	RCSphi
0,000	-7,917459E-001	-1,600000E+002
1,000	-8,152225E-001	-1,600000E+002
2,000	-8,856265E-001	-1,600000E+002
3,000	-1,002874E+000	-1,600000E+002
4,000	-1,166802E+000	-1,600000E+002
5,000	-1,377136E+000	-1,600000E+002
6,000	-1,633430E+000	-1,600000E+002
7,000	-1,935002E+000	-1,600000E+002
8,000	-2,280829E+000	-1,600000E+002
9,000	-2,669424E+000	-1,600000E+002
10,000	-3,098677E+000	-1,600000E+002
11,000	-3,565673E+000	-1,600000E+002
12,000	-4,066482E+000	-1,600000E+002
13,000	-4,595959E+000	-1,600000E+002
14,000	-5,147578E+000	-1,600000E+002
15,000	-5,713386E+000	-1,600000E+002
16,000	-6,284136E+000	-1,600000E+002
17,000	-6,849723E+000	-1,600000E+002
18,000	-7,399930E+000	-1,600000E+002
19,000	-7,925462E+000	-1,600000E+002
20,000	-8,419016E+000	-1,600000E+002

Gambar 3.9: Tampilan hasil perhitungan pada software

Sketsa nilai RCS pada koordinat polar diperlihatkan pada Gambar 3.10. Pada Gambar 3.10 disketsakan nilai / pola RCStheta pada  $\phi = 0^\circ$ . Untuk menampilkan pola RCSphi dapat dilakukan dengan memilih *checkbox* RCSphi kemudian menekan tombol Tampilkan. Pada gambar, nilai RCS berada pada kisaran -30 dBsm sampai 20 dBsm. Skala 0, 30, 60 dan seterusnya merupakan nilai sudut  $\theta$ . Apabila *combobox* pada bagian atas dipilih theta, maka skala pada gambar koordinat polar merupakan nilai sudut  $\phi$ .

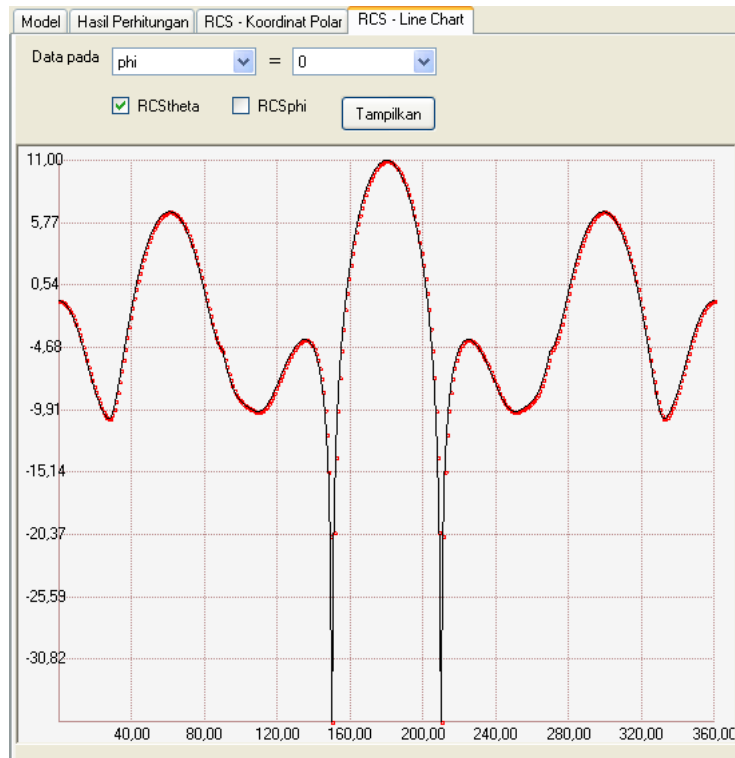
Tampilan koordinat polar ini hanya ditujukan untuk perhitungan RCS berbasis sudut.



Gambar 3.10: Tampilan koordinat polar pada software

Sketsa nilai RCS pada koordinat kartesian diperlihatkan pada Gambar 3.11. Pada Gambar 3.11 disketsakan nilai / pola RCStheta pada  $\phi = 0^\circ$  pada koordinat kartesian. Untuk menampilkan pola RCSphi dapat dilakukan dengan memilih *checkbox* RCSphi kemudian menekan tombol Tampilkan. Pada perhitungan berbasis sudut, sumbu datar menyatakan besaran sudut (dalam  $\theta$  atau  $\phi$ ), dan sudut tegak merupakan besaran RCS. Apabila *combobox* pada bagian atas dipilih theta, sumbu datar menyatakan besaran sudut  $\phi$ . Sedangkan apabila *combobox* pada bagian atas dipilih phi, sumbu datar menyatakan besaran sudut  $\theta$ .

Tampilan koordinat kartesian dapat digunakan untuk perhitungan RCS berbasis sudut maupun berbasis frekuensi. Pada perhitungan RCS berbasis frekuensi sumbu datar menyatakan besaran frekuensi, dan sudut tegak merupakan besaran RCS.



Gambar 3.11: Tampilan koordinat kartesian pada software

### 3.5 Kode Physical Optics

Kode PO dibentuk dalam satu *class* tersendiri, yaitu **POCalc**. *Class* ini digunakan untuk memodularkan dan menyimpan seluruh proses perhitungan RCS menggunakan PO. Pada **POCalc** dibentuk fungsi-fungsi, sebagaimana yang diperlihatkan pada Tabel 3.6.

Tabel 3.6: Fungsi-fungsi pada **POCalc**

No.	Nama Fungsi	Keterangan
1	ComplexPowUInt	memangkatkan bilangan kompleks sebesar bilangan bulat positif tertentu
2	facetRCSBiStatic	perhitungan RCS bagian potongan segitiga secara bistatis
3	facetRCSMonoStatic	perhitungan RCS bagian potongan segitiga secara monostatis
4	factorial	perhitungan $n!$
5	HitungBiStatic	perhitungan RCS secara bistatis berbasis sudut
6	HitungBiStaticFreq	perhitungan RCS secara bistatis berbasis frekuensi
7	HitungMonoStatic	perhitungan RCS secara monostatis berbasis sudut
8	HitungMonoStaticFreq	perhitungan RCS secara monostatis berbasis frekuensi
9	Ic_Eval	perhitungan integral pada difraksi elektromagnetik
10	POCalc	Definisi dasar untuk class Physical Optics
11	Pow2	memangkatkan bilangan sebesar dua
12	RCLayer	perhitungan koefisien refleksi secara paralel (sejajar) atau tegak lurus
13	recG	perhitungan fungsi G secara rekursif, fungsi ini untuk mendukung perhitungan Ic_Eval
14	ReflCoeff	perhitungan koefisien refleksi dan sudut transmisi tiap lapis permukaan, juga untuk mendukung fungsi RCLayer

# BAB 4

## Uji Coba dan Evaluasi Software

Pada bab ini akan diujikan beberapa bentuk bangun geometris sederhana, yaitu pelat, kubus, limas dan bola. Untuk model yang kompleks digunakan model kapal frigate [13]. Perbandingan hasil perhitungan dilakukan dengan menggunakan beberapa perumusan RCS yang ada dan dengan membandingkan hasil pada POFACETS dan targetRCS2.

### 4.1 Pelat

Pelat yang diujikan berdimensi  $1 \text{ m} \times 1 \text{ m}$ , dan dibentuk dari 2 segitiga. Dengan menggunakan Persamaan 2.4, frekuensi  $f = 300 \text{ MHz}$  ( $\lambda = C/f = 3.10^8 \text{ m.s}^{-1} / 3.10^8 \text{ Hz} = 1 \text{ m}$ ), RCS pelat sebagai berikut.

$$\sigma = \frac{4\pi A^2}{\lambda^2} = \frac{4\pi(1.1)^2}{1^2} = 12.5663706 \text{ m}^2 = 10.9920986 \text{ dBsm}$$

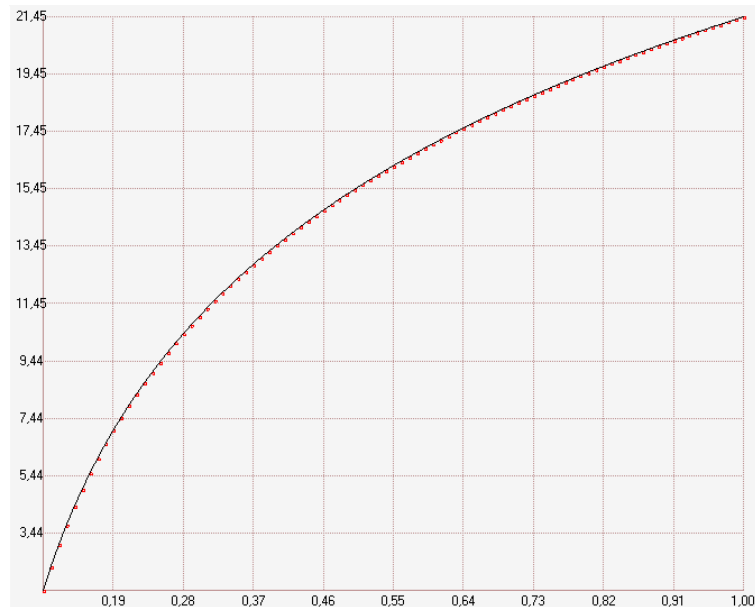
Nilai ini diperoleh dengan menggunakan software Matlab. Dengan targetRCS2 diperoleh RCS pelat sebesar 10.9920986 dBsm. Berarti tidak ada perbedaan antara hasil RCS dengan Persamaan 2.4 dan targetRCS2.

Hasil perhitungan Persamaan 2.4 dengan targetRCS2 (dalam dBsm), pada frekuensi tertentu (dalam GHz), diperlihatkan pada Tabel 4.1.

Tabel 4.1: Perhitungan  $\sigma$  pelat terhadap frekuensi

Frekuensi	targetRCS2	Persamaan 2.4	Selisih
0,1	1,4496735	1,4496735	0
0,2	7,4702735	7,4702735	0
0,3	10,9920986	10,9920986	0
0,4	13,4908734	13,4908734	0
0,5	15,4290736	15,4290736	0
0,6	17,0126986	17,0126986	0
0,7	18,3516343	18,3516343	0
0,8	19,5114733	19,5114733	0
0,9	20,5345237	20,5345237	0
1	21,4496735	21,4496735	0

Dari Tabel 4.1, terlihat bahwa dengan presisi 7 angka di belakang koma, hasil perhitungan menggunakan targetRCS2 dan Persamaan 2.4 tidak menunjukkan adanya perbedaan.



Gambar 4.1: RCS pelat terhadap frekuensi

Grafik hasil perhitungan pelat dari Tabel 4.1 diperlihatkan pada Gambar 4.1. Pada grafik, sumbu datar merepresentasikan frekuensi dalam GHz, dan sumbu tegak merepresentasikan nilai RCS dalam dBsm.

Menggunakan Persamaan *scattering* segiempat PO (Persamaan 2.18) dan targetRCS2, pada sudut  $\phi = 10^\circ$  dengan  $f = 0,3$  GHz diperoleh hasil perhitungan  $\sigma_{\theta\theta}(\theta, \phi = 10^\circ)$ , sebagaimana yang diperlihatkan pada Tabel 4.2.

Tabel 4.2:  $\sigma$  pelat pada targetRCS2 dan Persamaan 2.18

$\theta$	targetRCS2	Persamaan 2.18	Selisih
-80	-36,1546033	-36,1546033	0
-70	-22,0999448	-22,0999448	0
-60	-12,8997874	-12,8997874	0
-50	-7,4015170	-7,4015170	0
-40	-6,6327296	-6,6327296	0
-30	-26,9303594	-26,9303594	0
-20	2,3754348	2,3754348	0
-10	9,0662705	9,0662705	0
0	10,9920986	10,9920986	0
10	9,0662705	9,0662705	0
20	2,3754348	2,3754348	0
30	-26,9303594	-26,9303594	0
40	-6,6327296	-6,6327296	0
50	-7,4015170	-7,4015170	0
60	-12,8997874	-12,8997874	0
70	-22,0999448	-22,0999448	0
80	-36,1546033	-36,1546033	0

Dari Tabel 4.2 terlihat bahwa tidak ada perbedaan antara hasil perhitungan targetRCS2 dan Persamaan 2.18.

Perbandingan hasil perhitungan  $\sigma_{\theta\theta}(\theta, \phi = 0^\circ)$  pada  $f = 0,3$  GHz, menggunakan POFACETS dan targetRCS2 diperlihatkan pada Tabel 4.3.

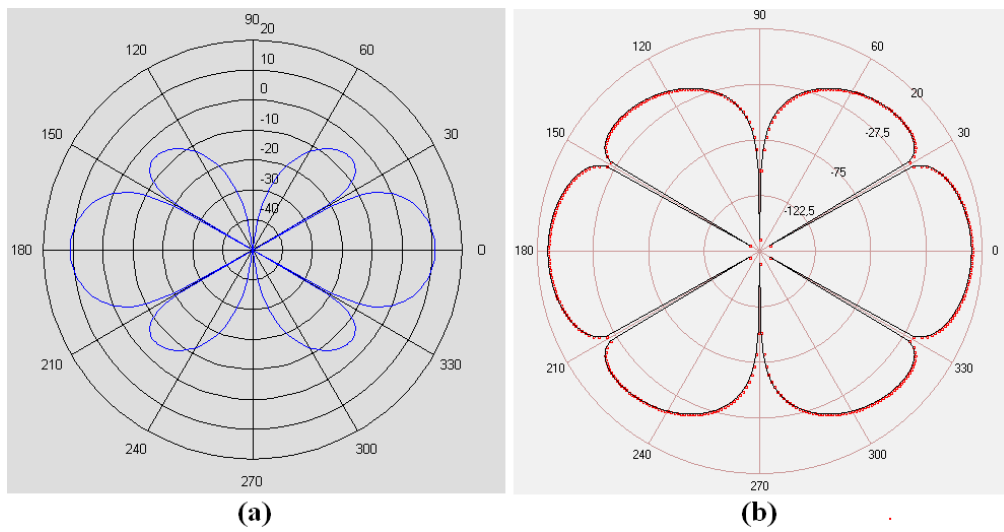
Tabel 4.3:  $\sigma$  pelat pada targetRCS2 dan POFACETS

$\theta$	targetRCS2	POFACETS	Selisih
-80	-36,1546033	-36,1546015	1,8E-06
-70	-22,0999448	-22,0999448	0
-60	-12,8997874	-12,8997874	0
-50	-7,4015170	-7,4015170	0
-40	-6,6327296	-6,6327296	0
-30	-26,9303594	-26,9303592	2E-07
-20	2,3754348	2,3754348	0
-10	9,0662705	9,0662705	0
0	10,9920986	10,9920986	0
10	9,0662705	9,0662705	0
20	2,3754348	2,3754348	0
30	-26,9303594	-26,9303592	2E-07
40	-6,6327296	-6,6327296	0
50	-7,4015170	-7,4015170	0
60	-12,8997874	-12,8997874	0
70	-22,0999448	-22,0999448	0
80	-36,1546033	-36,1546015	1,8E-06

Pada perbandingan targetRCS2 dengan POFACETS terdapat perbedaan hingga 1,8E-06. Hal ini bisa disebabkan karena perbedaan akurasi pada software.

Hasil perhitungan RCS pelat yang ada pada Tabel 4.3 dalam koordinat polar, diperlihatkan pada Gambar 4.2. Dimana bagian (a) pada Gambar 4.2 adalah hasil dari POFACETS, dan bagian (b) adalah hasil dari targetRCS2.





Gambar 4.2: Pola RCS pelat dari Tabel 4.3

## 4.2 Kubus

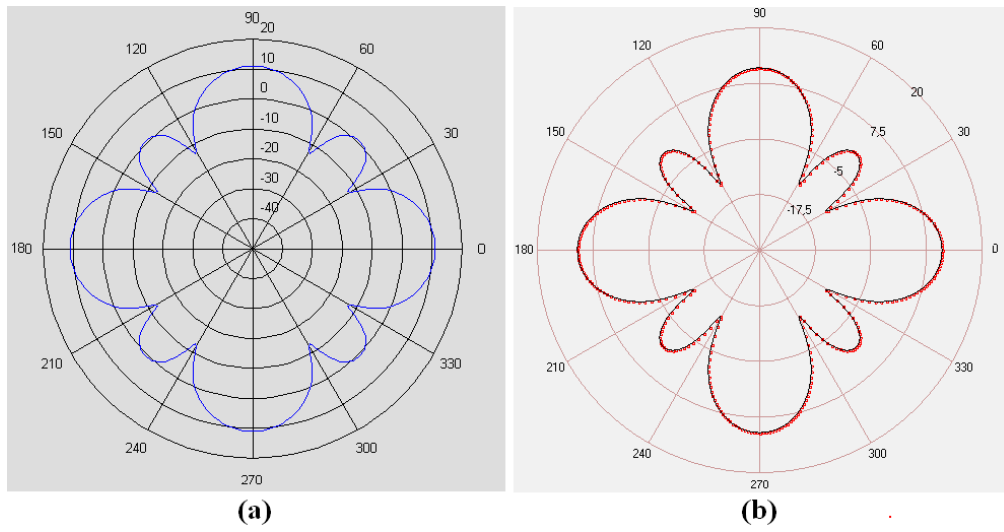
Kubus yang diujikan berdimensi  $1\text{ m} \times 1\text{ m} \times 1\text{ m}$ , dan dibentuk dari 12 segitiga.

Perbandingan hasil perhitungan  $\sigma_{\theta\theta}(\theta, \phi = 15^\circ)$  pada frekuensi = 0,3 GHz, menggunakan POFACETS dan targetRCS2 diperlihatkan pada Tabel 4.4

Tabel 4.4:  $\sigma_{\theta\theta}(\theta, \phi = 15^\circ)$  kubus pada targetRCS2 dan POFACETS

$\theta$	targetRCS2	POFACETS	Selisih
0	10,9920986	10,9920986	0
20	2,5952406	2,5952406	0
40	-1,5125614	-1,5125614	0
60	-13,8828656	-13,8828656	0
80	5,2219285	5,2219285	0
100	5,2219285	5,2219285	0
120	-13,8828656	-13,8828656	0
140	-1,5125614	-1,5125614	0
160	2,5952406	2,5952406	0
180	10,9920986	10,9920986	0
200	2,5952406	2,5952406	0
220	-1,5125614	-1,5125614	0
240	-13,8828656	-13,8828656	0
260	5,2219285	5,2219285	0
280	5,2219285	5,2219285	0
300	-13,8828656	-13,8828656	0
320	-1,5125614	-1,5125614	0
340	2,5952406	2,5952406	0
360	10,9920986	10,9920986	0

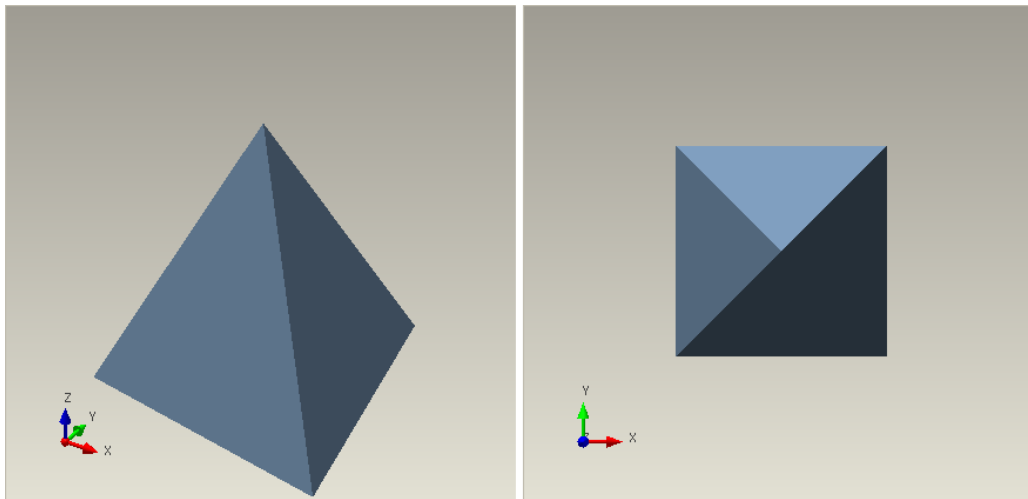
Dari hasil perhitungan targetRCS2 dengan POFACETS pada Tabel 4.4 tidak terlihat adanya perbedaan. Hasil perhitungan pada Tabel 4.4 dalam koordinat polar, diperlihatkan pada Gambar 4.3.



Gambar 4.3: Pola RCS kubus dari Tabel 4.4

### 4.3 Limas

Limas yang dimaksud di sini adalah bangun yang diperlihatkan pada Gambar 4.4. Dimensi limas, alas =  $1\text{ m} \times 1\text{ m}$ , tinggi =  $1\text{ m}$ . Limas dibentuk dari 6 segitiga.



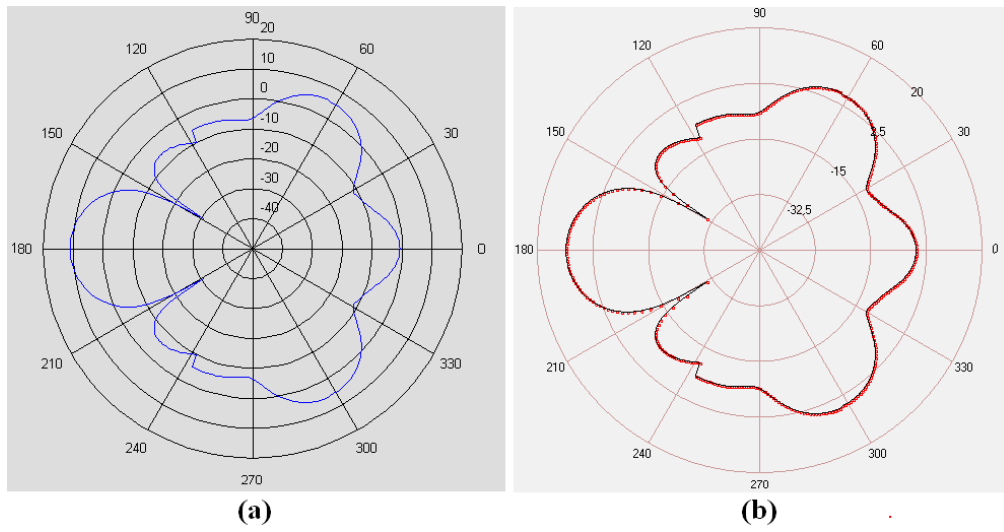
Gambar 4.4: Bentuk limas untuk bahan uji

Perbandingan hasil perhitungan  $\sigma_{\theta\theta}(\theta, \phi = 30^\circ)$  limas pada frekuensi = 0,3 GHz, menggunakan targetRCS2 dan POFACETS diperlihatkan pada Tabel 4.5.

Tabel 4.5:  $\sigma_{\theta\theta}(\theta, \phi = 15^\circ)$  limas pada targetRCS2 dan POFACETS

$\theta$	targetRCS2	POFACETS	Selisih
0	-0,7917459	-0,7917459	0
20	-9,4103971	-9,4103971	0
40	-5,8238946	-5,8238946	0
60	2,6381591	2,6381591	0
80	-5,3435978	-5,3435978	0
100	-11,8048847	-11,8048846	1E-07
120	-13,9099026	-13,9099026	0
140	-17,5430159	-17,5430159	0
160	2,9445575	2,9445575	0
180	10,9920986	10,9920986	0
200	2,9445575	2,9445575	0
220	-17,5430159	-17,5430159	0
240	-13,9099026	-13,9099026	0
260	-11,8048847	-11,8048846	1E-07
280	-5,3435978	-5,3435978	0
300	2,6381591	2,6381591	0
320	-5,8238946	-5,8238946	0
340	-9,4103971	-9,4103971	0
360	-0,7917459	-0,7917459	0

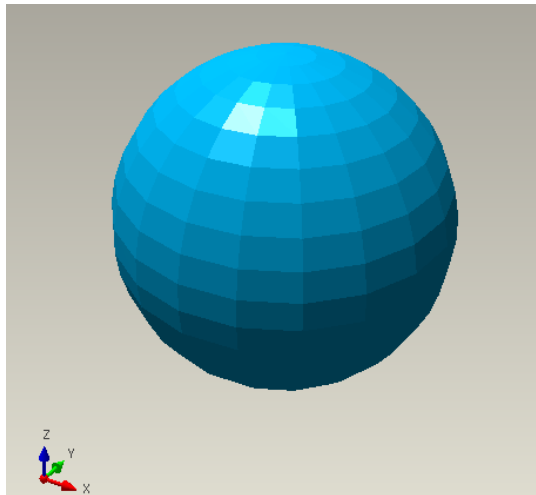
Dari hasil perhitungan targetRCS2 dengan POFACETS pada Tabel 4.5 hampir terlihat adanya perbedaan. Perbedaan hanya terlihat pada  $\theta = 100^\circ$  dan  $\theta = 260^\circ$  sebesar 1E-7 dBsm, dimana perbedaan ini sangat kecil. Hasil perhitungan pada Tabel 4.5 dalam koordinat polar, diperlihatkan pada Gambar 4.5.



Gambar 4.5: Pola RCS limas dari Tabel 4.5

#### 4.4 Bola

Bola yang diujikan dengan jari-jari 1 m, dengan bentuk yang diperlihatkan pada Gambar 4.6. Bola dibentuk dari 760 segitiga.



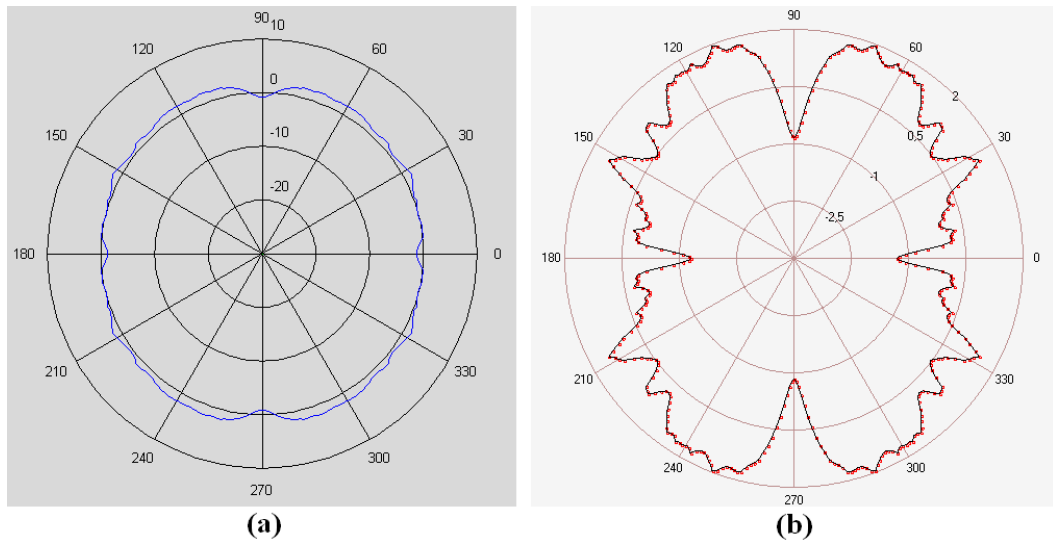
Gambar 4.6: Bentuk bola untuk bahan uji

Perbandingan hasil perhitungan  $\sigma_{\theta\theta}(\theta, \phi = 0^\circ)$  bola pada frekuensi = 0,3 GHz, menggunakan targetRCS2 dan POFACTS diperlihatkan pada Tabel 4.6.

Tabel 4.6:  $\sigma_{\theta\theta}(\theta, \phi = 0^\circ)$  bola pada targetRCS2 dan PO-FACETS

$\theta$	targetRCS2	POFACETS	Selisih
0	-1,2975143	-1,2975131	1,20E-06
20	0,2892447	0,2892463	1,60E-06
40	0,8407608	0,8407608	0
60	1,6623159	1,6623141	1,80E-06
80	1,3682730	1,3682756	2,60E-06
100	1,3682730	1,3682756	2,60E-06
120	1,6623159	1,6623141	1,80E-06
140	0,8407608	0,8407608	0
160	0,2892447	0,2892463	1,60E-06
180	-1,2975143	-1,2975131	1,20E-06
200	0,2892447	0,2892463	1,60E-06
220	0,8407608	0,8407608	0
240	1,6623159	1,6623141	1,80E-06
260	1,3682730	1,3682756	2,60E-06
280	1,3682730	1,3682756	2,60E-06
300	1,6623159	1,6623141	1,80E-06
320	0,8407608	0,8407608	0
340	0,2892447	0,2892463	1,60E-06
360	-1,2975143	-1,2975131	1,20E-06

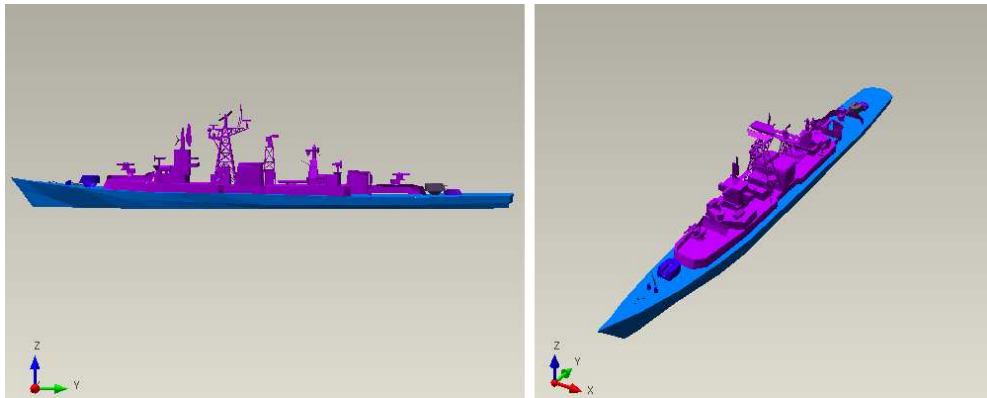
Dari hasil perhitungan targetRCS2 dengan POFACETS pada Tabel 4.6 terlihat adanya perbedaan antara 1,20E-06 dBsm sampai 2,60E-06 dBsm. Hasil perhitungan pada Tabel 4.6 dalam koordinat polar, diperlihatkan pada Gambar 4.7.



Gambar 4.7: Pola RCS bola dari Tabel 4.6

## 4.5 Frigate

Frigate yang diujikan berasal dari data INRIA, dengan bentuk yang diperlihatkan pada Gambar 4.8. Frigate ini dibentuk dari 4157 segitiga.



Gambar 4.8: Bentuk Frigate untuk bahan uji

Perbandingan hasil perhitungan  $\sigma_{\theta\theta}(\theta, \phi = 0^\circ)$  frigate pada frekuensi = 3 GHz, menggunakan targetRCS2 dan POFACTS diperlihatkan pada Tabel 4.7.

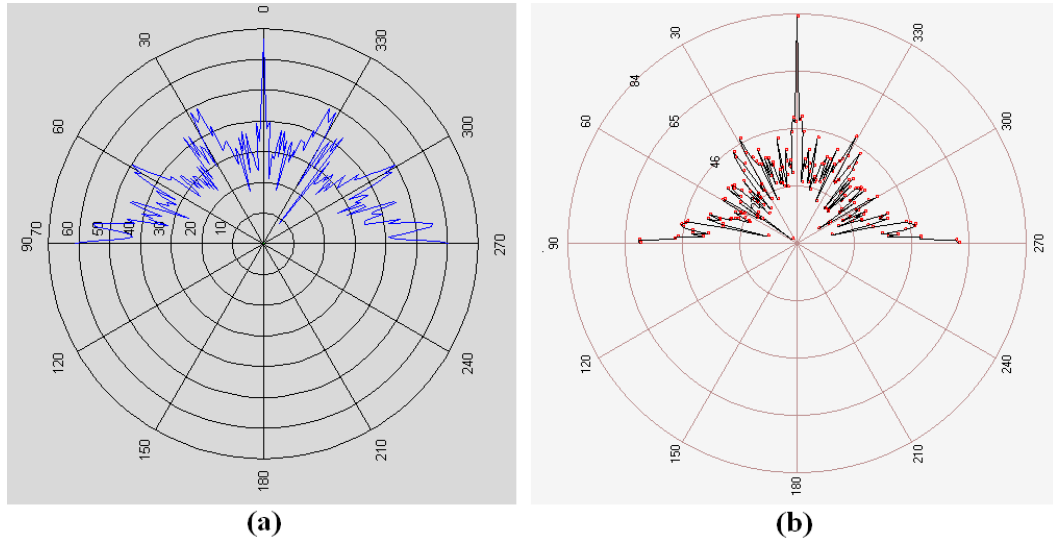
Tabel 4.7:  $\sigma_{\theta\theta}(\theta, \phi = 0^\circ)$  Frigate pada targetRCS2 dan POFACETS

$\theta$	targetRCS2	POFACETS	Selisih
-90	61,5383098	60,2764892	1,262E+00
-80	45,9389771	45,3682108	5,708E-01
-70	30,3697526	35,4932606	5,124E+00
-60	38,2779489	45,8146820	7,537E+00
-50	35,2745063	34,7123587	5,621E-01
-40	29,8573256	25,5678624	4,289E+00
-30	41,0223612	41,0780547	5,569E-02
-20	29,4884250	35,8682604	6,380E+00
-10	28,2323882	28,8370071	6,046E-01
0	82,8101314	66,6667716	1,614E+01
10	26,8252716	23,5115665	3,314E+00
20	28,9321012	35,4644890	6,532E+00
30	42,4649191	38,7751016	3,690E+00
40	27,4363616	22,9868057	4,450E+00
50	35,6906948	34,5858833	1,105E+00
60	36,5846922	45,8423878	9,258E+00
70	31,5987306	34,3639127	2,765E+00
80	45,8757690	45,0451333	8,306E-01
90	60,2853480	61,5300613	1,245E+00

Dari hasil perhitungan targetRCS2 dengan POFACETS pada Tabel 4.7 terlihat adanya perbedaan. Perbedaan RCS perhitungan frigate pada targetRCS2 dan POFACETS antara  $5,569E-02$  dBsm sampai  $1,614E+01$  dBsm. Perbedaan terbesar terdapat pada saat  $\theta = 0$ . Hasil perhitungan seluruh model, terlihat bahwa semakin banyak segitiga yang dihitung, perbedaan antara targetRCS2 dan POFACETS jadi semakin besar. Hal ini bisa disebabkan adanya perbedaan dalam penentuan bagian yang teriluminasi dan bagian bayangan, tingkat akurasi perhitungan pada tiap software, dan kemungkinan adanya perhitungan tambahan pada POFACETS.



Hasil perhitungan pada Tabel 4.7 dalam koordinat polar, diperlihatkan pada Gambar 4.9.



Gambar 4.9: Pola RCS Frigate dari Tabel 4.7

# BAB 5

## Penutup

### 5.1 Kesimpulan

Dari penelitian ini dapat disimpulkan beberapa hal. Hasil perhitungan menggunakan targetRCS2 pada pelat nilainya tidak berbeda dengan hasil dari perumusan RCS dasar. Nilai RCS dari targetRCS2 untuk bangun pelat, kubus, limas dan bola tidak memperlihatkan perbedaan yang berarti dengan nilai RCS dari POFACETS. Namun untuk bentuk yang lebih kompleks, yaitu frigate hasil perhitungan RCS dengan targetRCS2 dan POFACETS terlihat perbedaan antara  $5,569E-02$  dBsm sampai  $1,614E+01$  dBsm. Hal ini bisa disebabkan adanya perbedaan dalam penentuan bagian yang teriluminasi dan bagian bayangan, tingkat akurasi perhitungan pada tiap software, dan kemungkinan adanya perhitungan tambahan pada POFACETS.

### 5.2 Saran

Beberapa hal yang dapat dikembangkan dari penelitian ini, diantaranya sebagai berikut.

1. Diharapkan adanya data hasil penujian yang benar-benar akurat (resmi) untuk perbandingan hasil penelitian.
2. Penggunaan metode lain ditambahkan pada perkembangan software (selain metode PO).
3. Perhitungan untuk terjadi refleksi pada tanah, apabila obyek berada pada permukaan tanah, begitu pula dengan refleksi pada air.

4. Penggunaan metode pendeteksian daerah bayangan selain *back-face culling*, karena pada metode *back-face culling* segitiga harus mengikuti aturan tangan kanan. Apabila pembentukan segitiga titiknya tidak mengikuti aturan tangan kanan maka akan terjadi kesalahan pemrosesan dalam penentuan bagian bayangan dari obyek.
5. Penambahan kemungkinan adanya refeksi multi, difraksi, dan pengaruh gelombang permukaan pada proses *scattering* obyek.
6. Pembacaan file AutoCAD dikembangkan untuk entitas AutoCAD yang lain, sehingga ketergantungan dengan penggunaan software lain untuk penyederhanaan model tidak diperlukan lagi.

## Daftar Pustaka

- [1] Autodesk Inc., (2005), *DXF Reference*, Autodesk Inc., United States of America.
- [2] Bradley, C.J. (2004), *The Calibration Of Bistatic Radar Cross Section Measurements*, Tesis Master, Air Force Institute of Technology.
- [3] Chartzigeorgiadis, F. dan Jenn, D.C. (2004), "A (MATLAB) Physical-Optics RCS Prediction Code", *IEEE Transactions on Antennas and Propagation*, Vol. 46, No. 4.
- [4] CSS Denmark, (2005), "(PC) based Radar Cross Section Simulation Software", <http://www.cadrcs.com>.
- [5] devDept, (2006), "C# OpenGL Framework", <http://www.csharpopenglframework.com>.
- [6] devDept, (2006), "C# OpenGL Framework Profesional Version Demo", [http://www.csharpopenglframework.com/professional\\_edition.html](http://www.csharpopenglframework.com/professional_edition.html).
- [7] dnAnalytics, (2006), "dnAnalytics, providing open source numerical tools for the .NET framework", <http://www.dnanalytics.net>.
- [8] Game Programming Wikipedia, (2006), "3D:Backface Culling", [http://gpwiki.org/index.php/3D:Backface\\_Culling](http://gpwiki.org/index.php/3D:Backface_Culling).

- [9] Garrido, Jr., E.E. (2000), *Graphical User Interface For a Physical Optics Radar Cross Section Prediction Code*, Tesis Master, Naval Postgraduate School.
- [10] GlobalSecurity.org, (2005), "F-22 Raptor Stealth", <http://www.globalsecurity.org/military/systems/aircraft/f-22-stealth.htm>.
- [11] Harris, A. (2002), *Microsoft C# Programming for Absolute Beginner*, Premier Press, United States of America.
- [12] Hearn, D. dan P.Baker, M. (1994), *Computer Graphics*, Second edition, Prentice Hall, New Jersey.
- [13] INRIA Rocquencourt, (2006), "frigate3 From 3D Meshes Research Database", <http://www-c.inria.fr/gamma/download/affichage.php?dir=WATERCRA&name=frigate3>.
- [14] Jenn, D.C. (2005), *Radar and Laser Cross Section Engineering*, Second edition, AIAA, Virginia.
- [15] Knott, E.F., Shaeffer, J.F. dan Tuley, M.T. (2004), *Radar Cross Section*, Second edition, SciTech Publishing Inc., Boston.
- [16] Lynch, Jr, D. (2004), *Introduction to RF Stealth*, SciTech Publishing Inc., Boston.
- [17] Microsoft Corporation, (2006), "Visual C# Developer Center", <http://msdn.microsoft.com/vcsharp/>.
- [18] Moreira, F.J.S. dan Prata, Jr., A. (1994), "A Self-Checking Predictor-Corrector Algorithm for Efficient Evaluation of Reflector Antennas Radiation Integrals", *IEEE Transactions on Antennas and Propagation*, Vol.

- 42, No. 2, <http://www.cpdee.ufmg.br/~fernando/artigos/ieeeAP94.pdf>.
- [19] Neider, J. dan Davis, T. (1997), *OpenGL Programming Guide*, Second edition, Addison Wesley Longman Inc., United States of America.
- [20] Ozturk, A.K. (2002), *Implementation of Physical Theory of Diffraction For Radar Cross Section Calculations*, Tesis Master, Bilkent University.
- [21] Roke Manor Research Limited, (2005), "Epsilon - Radar Cross Section Prediction Tool", <http://www.roke.co.uk/sensors/epsilon>.
- [22] Sarabandi, K. (2003), "A Radar Cross-Section Model for Power Lines at Millimeter-Wave Frequencies", *IEEE Transactions on Antennas and Propagation*, Vol. 51, No. 9.
- [23] Sefi, S. dan Ooppelstrup, J. (2004), "Physical Optics And Nurbs For RCS Calculations", *Computational Electromagnetic 2004*, <http://www.nada.kth.se/~sandy/RAPPORTS/emb04.pdf>.
- [24] Sharp, J. dan Jagger, J. (2003), *Microsoft Visual C# .NET Step by Step*, Microsoft Press, Washington.
- [25] Silicon Graphics, Inc. (2006), "OpenGL - The Industry Standard for High Performance Graphics", <http://www.opengl.org>.
- [26] Skolnik, M.I. (1970), *Radar Handbook*, McGraw-Hill, Inc, New York.
- [27] Surface Optics Corporation, (2005), "RadBase (Radar Cross Section Simulation and Analysis)", <http://www.surfaceoptics.com/index.htm>.
- [28] Wikipedia, (2005), "Stealth technology", Wikimedia Foundation, Inc, [http://en.wikipedia.org/wiki/Stealth\\_technology](http://en.wikipedia.org/wiki/Stealth_technology).

- [29] Wikipedia, (2006), "StL File Format", Wikimedia Foundation, Inc, [http://en.wikipedia.org/wiki/STL\\_\(file\\_format\)](http://en.wikipedia.org/wiki/STL_(file_format)).
- [30] Wright, Jr, R.S. dan Sweet, M. (1996), *OpenGL SuperBible*, Waite Group Press, United States of America.
- [31] Zhang, H. dan Hoff, K. (1997), "Fast Backface Culling Using Normal Mask", Symposium on Interactive 3D Graphics, <http://www.cs.unc.edu/~zhangh/backface.pdf>.

# Lampiran A

## Source Code POCalc

```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4 using dnA.Math;
5 using System.Collections;
6 using openglFramework;
7 using System.IO;
8
9 namespace TargetRCS2
10 {
11     class POCalc
12     {
13         // Tambahan data
14         Type trifaceType = typeof(openglFramework.TriangularFace);
15         public POCalc()
16         {
17             Console.WriteLine("POCalc init!!!");
18         }
19
20         #region Data_Utama_PO
21         ComplexDouble jn = new ComplexDouble(0.0, 1.0);
22
23         public Form1 fParent;
24         public MaterialsDataSet materialsDataSet1;
25
26         public enum Polarisasi { TM, TE };
27         public enum TipeKoefisienRefleksi { General, LayerType };
28
29         public double theta0;    // theta start
30         public double theta1;    // theta end
31         public double del_theta;
32         int itheta;
33         public double phi0;      // phi start
34         public double phi1;      // phi end
35         public double del_phi;
36         int iphi;
37         public Polarisasi polarisasi_kejadian;
38         public TipeKoefisienRefleksi tipe_koef_ref;
```



```

39
40     public double frekuensi;
41     // public double wave;    // wave = 3e8 / frekuensi
42     double wave;
43
44     // Data untuk basis frekuensi
45     public double F_Theta;
46     public double F_Phi;
47     public double F_frekuensi0;
48     public double F_frekuensi1;
49     public double F_del_frekuensi;
50
51     // deret Taylor
52     public double panjang_daerah;
53     public int jumlah_term;
54
55     // Kekasaran permukaan
56     public double jarak_korelasi;
57     public double standar_deviasi;
58
59     // Sudut Kejadian (bistatic)
60     public double theta_kejadian = 0;
61     public double phi_kejadian = 0;
62
63     public double Rs = 0; // Resistivity untuk setiap segitiga
64
65     // Tambahan variabel
66     double rad = Math.PI / 180.0;
67     double C = 3E8;    // Konstanta kecepatan cahaya
68     double e_fs = 8.854E-12; // permittivity of free space
69     double m_fs = 4 * Math.PI * 1E-7; // permeability of free space
70     public double snum = 1E-16; // mendekati eps
71     double snum_db;
72
73     ArrayList entitas_segitiga = new ArrayList();
74     int entitas_segitiga_count;
75
76     double rcsTemp;
77     double[,] RCStheta;
78     double[,] RCSphi;
79
80     double Co = 1;
81     #endregion
82
83     #region Data_Perhitungan
84     double corel; // normalisasi panjang gelombang
85     double delstd;
86     double delsq; // variance

```

```

87     double bk;
88     double cfac1;
89     double cfac2;
90
91     ComplexDouble E0t = 0.0;
92     ComplexDouble E0p = 0.0;
93     ComplexDouble Et2 = 0.0;
94     ComplexDouble Ep2 = 0.0;
95     #endregion
96
97     // Variabel Global II untuk mengurangi beban pembuatan variabel
98     #region Variabel_Global_II
99     double st; // sin thetaR
100    double ct; // cos thetaR
101    double sp; // sin phiR
102    double cp; // cos phiR
103    double u;
104    double u2;
105    double uu;
106    double v;
107    double v2;
108    double vv;
109    double w;
110    double w2;
111    double ww;
112
113    double st2;
114    double ct2;
115    double sp2;
116    double cp2;
117    double theta_2;
118    double phi_2;
119
120    // Tambahan-tambahan bistatik
121    double cpi;
122    double spi;
123    double cti;
124    double sti;
125
126    double ui;
127    double vi;
128    double wi;
129
130    double vi2;
131    double ui2;
132    double wi2;
133
134    double uui;

```

```

135     double vvi;
136     double wwi;
137
138     double sti2;
139     double cti2;
140     double spi2;
141     double cpi2;
142     double theta_i2;
143     double phi_i2;
144
145     DoubleMatrix D0 = new DoubleMatrix(1, 3);
146     DoubleMatrix D0i = new DoubleMatrix(1, 3);
147
148     // _InF ==> Incident Field
149     ComplexDoubleMatrix EO_InF = new ComplexDoubleMatrix(1, 3);
150     ComplexDoubleMatrix E2_InF;
151
152     ComplexDouble Jx2;
153     ComplexDouble Jy2;
154
155     ComplexDouble Ic;
156
157     double Edif; // untuk penambahan komponen difusi
158     ComplexDoubleMatrix Es2 = new ComplexDoubleMatrix(3, 1);
159     ComplexDoubleMatrix Es1;
160     ComplexDoubleMatrix Es0;
161     ComplexDoubleMatrix Ed2 = new ComplexDoubleMatrix(3, 1);
162     ComplexDoubleMatrix Ed1;
163     ComplexDoubleMatrix Ed0;
164
165     double ndotk;
166     double nidotk;
167
168     // matrix transformasi koordinat kartesian ke koordinat silinder
169     public DoubleMatrix T1 = new DoubleMatrix(3, 3, 0.0);
170     // matrix transformasi koordinat silinder ke koordinat shperical
171     public DoubleMatrix T2 = new DoubleMatrix(3, 3, 0.0);
172
173     DoubleMatrix D1 = null;
174     DoubleMatrix D2 = null;
175     DoubleMatrix D1i = null;
176     DoubleMatrix D2i = null;
177
178     double DpX; // untuk fase pada vertex segitiga
179     double DqX;
180     double DoX;
181
182     ComplexDouble RCperp; // Reflection Koefisien

```

```

183     ComplexDouble RCpara;
184     #endregion
185
186     #region Variabel_Perhitungan_ReflCoeff
187     ComplexDoubleMatrix M1para = new ComplexDoubleMatrix(2, 2);
188     ComplexDoubleMatrix M1perp = new ComplexDoubleMatrix(2, 2);
189     ComplexDoubleMatrix M2para = new ComplexDoubleMatrix(2, 2);
190     ComplexDoubleMatrix M2perp = new ComplexDoubleMatrix(2, 2);
191     ComplexDoubleMatrix Mpara;
192     ComplexDoubleMatrix Mperp;
193     #endregion
194
195     public void SetEntitasData (ArrayList arl)
196     {
197         entitas_segitiga = arl;
198         entitas_segitiga_count = entitas_segitiga.Count;
199     }
200
201     public double[,] getRCSttheta()
202     {
203         return RCSttheta;
204     }
205
206     public double[,] getRCSphi()
207     {
208         return RCSphi;
209     }
210
211     public void HitungMonoStatic()
212     {
213         snum_db = 10 * Math.Log10(snum);
214
215         double thetaR;    // untuk proses looping (iterator sudut)
216         double phiR;
217         TriangularFace tri_face;
218         facetRCSData fRCS;
219
220         RCLayerResult layerRC;
221         layerRC.RCpara1 = -1;
222         layerRC.RCperp1 = -1;
223
224         wave = C / (frekuensi * 1E9);
225         corel = jarak_korelasi / wave;
226         delstd = standar_deviasi;
227         delsq = Pow2(delstd);
228         bk = 2 * Math.PI / wave;
229         cfac1 = Math.Exp(-4 * Pow2(bk) * delsq);
230         cfac2 = 4 * Math.PI * Pow2(bk * corel) * delsq;

```

```

231
232     if (polarisasi_kejadian == Polarisasi.TM)
233     {
234         EOt = 1;
235         EOp = 0;
236     }
237     else if (polarisasi_kejadian == Polarisasi.TE)
238     {
239         EOt = 0;
240         EOp = 1;
241     }
242
243     itheta = (int)(Math.Floor((theta1 - theta0) / del_theta) + 1.0);
244     iphi = (int)(Math.Floor((phi1 - phi0) / del_phi) + 1.0);
245     RCStheta = new double[iphil, itheta];
246     RCSphi = new double[iphil, itheta];
247
248     // Perjalanan dimulai
249     ComplexDouble sumt = 0;
250     ComplexDouble sump = 0;
251     double sumdt = 0.0;
252     double sumdp = 0.0;
253
254     int i1;
255     int i2;
256     for (i1 = 0; i1 < iphil; ++i1) // awal perjalanan perhitungan
257     {
258         for (i2 = 0; i2 < itheta; ++i2)
259         {
260             fParent.ProgressBar1.Increment(1);
261
262             thetaR = (theta0 + i2 * del_theta) * rad;
263             phiR = (phi0 + i1 * del_phi) * rad;
264
265             // ----- facetRCS untuk MonoStatic
266             // Sudut global dan arah
267             st = Math.Sin(thetaR); ct = Math.Cos(thetaR);
268             sp = Math.Sin(phiR); cp = Math.Cos(phiR);
269             u = st * cp;
270             v = st * sp;
271             w = ct;
272             DO[0, 0] = u;
273             DO[0, 1] = v;
274             DO[0, 2] = w;
275             uu = ct * cp;
276             vv = ct * sp;
277             ww = -st;
278

```

```

279     EO_InF[0, 0] = EOt * uu - EOp * sp;
280     EO_InF[0, 1] = EOt * vv + EOp * cp;
281     EO_InF[0, 2] = EOt * ww;
282
283     // memulai loop untuk setiap segitiga
284     sumt = 0;
285     sump = 0;
286     sumdt = 0;
287     sumdp = 0;
288
289     for (int m = 0; m < entitas_segitiga_count; ++m)
290     {
291         if (entitas_segitiga[m].GetType() != trifaceType)
292             continue;
293
294         tri_face = (TriangularFace)entitas_segitiga[m];
295         fRCS = facetRCSMonoStatic(thetaR, phiR, tri_face, EO_InF);
296
297         sumt = sumt + fRCS.Ets;
298         sump = sump + fRCS.Eps;
299         sumdt = sumdt + ComplexMath.Absolute(fRCS.Etd);
300         sumdp = sumdp + ComplexMath.Absolute(fRCS.Epd);
301     }
302
303     rcsTemp = 4 * Math.PI * cfac1
304         * (Pow2(ComplexMath.Absolute(sumt))
305           + Math.Sqrt(1 - Pow2(cfac1)) * sumdt)
306         / Pow2(wave);
307     if (rcsTemp <= snum)
308         RCStheta[i1, i2] = snum_db;
309     else
310         RCStheta[i1, i2] = 10 * Math.Log10(rcsTemp);
311
312     rcsTemp = 4 * Math.PI * cfac1
313         * (Pow2(ComplexMath.Absolute(sump))
314           + Math.Sqrt(1 - Pow2(cfac1)) * sumdp)
315         / Pow2(wave);
316     if (rcsTemp <= snum)
317         RCSphi[i1, i2] = snum_db;
318     else
319         RCSphi[i1, i2] = 10 * Math.Log10(rcsTemp);
320     }
321 }
322 }
323
324 public void HitungMonoStaticFreq()
325 {
326     snum_db = 10 * Math.Log10(snum);

```

```

327
328     double thetaR;
329     double phiR;
330     int iFreq;
331     TriangularFace tri_face;
332     facetRCSData fRCS;
333
334     RCLayerResult layerRC;
335     layerRC.RCpara1 = -1;
336     layerRC.RCperp1 = -1;
337
338     if (polarisasi_kejadian == Polarisasi.TM)
339     {
340         EOt = 1;
341         EOp = 0;
342     }
343     else if (polarisasi_kejadian == Polarisasi.TE)
344     {
345         EOt = 0;
346         EOp = 1;
347     }
348
349     iFreq = (int)(Math.Floor((F_frekuensi1 - F_frekuensi0)
350         / F_del_frekuensi) + 1.0);
351     RCStheta = new double[iFreq, 1];
352     RCSphi = new double[iFreq, 1];
353
354     // Perjalanan dimulai
355     ComplexDouble sumt = 0.0;
356     ComplexDouble sump = 0.0;
357     double sumdt = 0.0;
358     double sumdp = 0.0;
359
360     thetaR = F_Theta * rad;
361     phiR = F_Phi * rad;
362
363     int i1;
364     frekuensi = F_frekuensi0;
365     for (i1 = 0; i1 < iFreq; ++i1)
366     {
367         wave = C / (frekuensi * 1E9);
368         corel = jarak_korelasi / wave;
369         delstd = standar_deviasi;
370         delsq = Pow2(delstd);
371         bk = 2 * Math.PI / wave;
372         cfac1 = Math.Exp(-4 * Pow2(bk) * delsq);
373         cfac2 = 4 * Math.PI * Pow2(bk * corel) * delsq;
374

```

```

375         fParent.ProgressBar1.Increment(1);
376
377         // ----- facetRCS untuk MonoStatic
378         // Sudut global dan arah
379         st = Math.Sin(thetaR); ct = Math.Cos(thetaR);
380         sp = Math.Sin(phiR); cp = Math.Cos(phiR);
381         u = st * cp;
382         v = st * sp;
383         w = ct;
384         DO[0, 0] = u;
385         DO[0, 1] = v;
386         DO[0, 2] = w;
387         uu = ct * cp;
388         vv = ct * sp;
389         ww = -st;
390
391         EO_InF[0, 0] = EOt * uu - EOp * sp;
392         EO_InF[0, 1] = EOt * vv + EOp * cp;
393         EO_InF[0, 2] = EOt * ww;
394
395         // memulai loop untuk setiap segitiga
396         sumt = 0;
397         sump = 0;
398         sumdt = 0;
399         sumdp = 0;
400
401         for (int m = 0; m < entitas_segitiga_count; ++m)
402         {
403             if (entitas_segitiga[m].GetType() != trifaceType)
404                 continue;
405
406             tri_face = (TriangularFace)entitas_segitiga[m];
407
408             fRCS = facetRCSMonoStatic(thetaR, phiR, tri_face, EO_InF);
409
410             sumt = sumt + fRCS.Ets;
411             sump = sump + fRCS.Eps;
412             sumdt = sumdt + ComplexMath.Absolute(fRCS.Etd);
413             sumdp = sumdp + ComplexMath.Absolute(fRCS.Epd);
414         }
415
416         // Hasil perhitungan
417         RCStheta[i1, 0] = 10 * Math.Log10(4 * Math.PI * cfac1
418             * (Pow2(ComplexMath.Absolute(sumt))
419             + Math.Sqrt(1 - Pow2(cfac1)) * sumdt)
420             / (Pow2(wave)) + snum);
421         RCSphi[i1, 0] = 10 * Math.Log10(4 * Math.PI * cfac1
422             * (Pow2(ComplexMath.Absolute(sump))

```



```

423         + Math.Sqrt(1 - Pow2(cfac1)) * sumdp)
424         / (Pow2(wave)) + snum);
425
426     frekuensi = frekuensi + F_del_frekuensi;
427 }
428 }
429
430 struct facetRCSData
431 {
432     public ComplexDouble Ets; // R = result / hasil
433     public ComplexDouble Eps;
434     public ComplexDouble Etd;
435     public ComplexDouble Epd;
436 }
437
438 private facetRCSData facetRCSMonoStatic(double thetaR, double phiR,
439     TriangularFace tri_face, ComplexDoubleMatrix EO_InF1)
440 {
441     facetRCSData hsl;
442     float[][] vert1 = tri_face.GetVertice();
443
444     // Tes apakah bagian depan teriluminasi
445     ndotk = u * tri_face.normal.x +
446         v * tri_face.normal.y +
447         w * tri_face.normal.z;
448
449     // Bagian bayangan
450     if (ndotk <= 0)
451     {
452         hsl.Etd = 0.0;
453         hsl.Epd = 0.0;
454         hsl.Ets = 0.0;
455         hsl.Eps = 0.0;
456     }
457
458     // bagian yang teriluminasi
459     else
460     {
461         DO.Transpose();
462
463         // T1 = [ca sa 0; -sa ca 0; 0 0 1];
464         // T2 = [cb 0 -sb; 0 1 0; sb 0 cb];
465         T1[0, 0] = tri_face.cosA; T1[0, 1] = tri_face.sinA;
466         T1[1, 0] = -tri_face.sinA; T1[1, 1] = tri_face.cosA;
467         T1[2, 2] = 1;
468         T2[0, 0] = tri_face.cosB; T2[0, 2] = -tri_face.sinB;
469         T2[1, 1] = 1;
470         T2[2, 0] = tri_face.sinB; T2[2, 2] = tri_face.cosB;

```

```

471
472     D1 = T1 * D0;
473     D2 = T2 * D1; // D2 --> matrix (3, 1)
474     D0.Transpose();
475     u2 = D2[0, 0];
476     v2 = D2[1, 0];
477     w2 = D2[2, 0];
478
479     // Pencarian sudut spherical pada koordinat lokal
480     st2 = Math.Sqrt(Pow2(u2) + Pow2(v2)) * Math.Sign(w2);
481     theta_2 = Math.Asin(st2);
482     ct2 = Math.Cos(theta_2);
483
484     // phi_2 = Math.Atan2(v2, u2 + snum);
485     phi_2 = Math.Atan2(v2, u2);
486     cp2 = Math.Cos(phi_2);
487     sp2 = Math.Sin(phi_2);
488
489     // "Incident field" pada koordinat kartesian lokal
490     EO_InF1.Transpose();
491     E2_InF = T2 * T1 * EO_InF1; // e2 --> matrix (3, 1)
492     EO_InF1.Transpose();
493
494     // "Incident field" pada koordinat spherical lokal
495     Et2 = E2_InF[0, 0] * ct2 * cp2 + E2_InF[1, 0] * ct2 * sp2
496           - E2_InF[2, 0] * st2;
497     Ep2 = -E2_InF[0, 0] * sp2 + E2_InF[1, 0] * cp2;
498
499
500     if (tipe_koef_ref == TipeKoefisienRefleksi.General)
501     {
502         RCperp = -1 / (2 * Rs * ct2 + 1);
503         RCpara = 0;
504         if ((2 * Rs + ct2) != 0)
505             RCpara = -ct2 / (2 * Rs + ct2);
506     }
507     else
508     {
509         RCLayerResult layerRC = RCLayer(theta_2, ct2, st2,
510             tri_face.autoCadColor, frekuensi * 1E9);
511         RCperp = layerRC.RCperp1;
512         RCpara = layerRC.RCpara1;
513     }
514
515     // Surface Current pada koordinat kartesian lokal
516     Jx2 = (-Et2 * cp2 * RCpara) + (Ep2 * sp2 * RCperp);
517     Jy2 = (-Et2 * sp2 * RCpara) - (Ep2 * cp2 * RCperp);
518     // Integral luas untuk kasus umum

```

```

519     DpX = 2 * bk * ((vert1[0][0] - vert1[2][0]) * u +
520                   (vert1[0][1] - vert1[2][1]) * v +
521                   (vert1[0][2] - vert1[2][2]) * w);
522
523     DqX = 2 * bk * ((vert1[1][0] - vert1[2][0]) * u +
524                   (vert1[1][1] - vert1[2][1]) * v +
525                   (vert1[1][2] - vert1[2][2]) * w);
526
527     DoX = 2 * bk * (vert1[2][0] * u +
528                   vert1[2][1] * v + vert1[2][2] * w);
529
530     Ic = Ic_Eval(tri_face.area, DpX, DqX, DoX, panjang_daerah);
531
532     // Tambahkan komponen difusi
533     Edif = cfac2 * tri_face.area * Pow2(ct2) *
534           Math.Exp(-Pow2(corel * Math.PI * st2 / wave));
535
536     // Komponen Scattered untuk segitiga pada koordinat lokal
537     Es2[0, 0] = Jx2 * Ic;
538     Es2[1, 0] = Jy2 * Ic;
539     Es2[2, 0] = 0;
540     Ed2[0, 0] = Jx2 * Edif;
541     Ed2[1, 0] = Jy2 * Edif;
542     Ed2[2, 0] = 0;
543
544     // Transformasi balik ke koordinat global
545     T2.Transpose();
546     Es1 = T2 * Es2; // 3x3 X 3x1 = 3x1
547     T2.Transpose();
548
549     T1.Transpose();
550     Es0 = T1 * Es1; // 3x1
551     T1.Transpose();
552
553     T2.Transpose();
554     Ed1 = T2 * Ed2;
555     T2.Transpose();
556
557     T1.Transpose();
558     Ed0 = T1 * Ed1;
559     T1.Transpose();
560
561     hsl.Ets = Es0[0, 0] * uu + Es0[1, 0] * vv + Es0[2, 0] * ww;
562     hsl.Eps = Es0[0, 0] * (-sp) + Es0[1, 0] * cp;
563     hsl.Etd = Ed0[0, 0] * uu + Ed0[1, 0] * vv + Ed0[2, 0] * ww;
564     hsl.Epd = Ed0[0, 0] * (-sp) + Ed0[1, 0] * cp;
565 }
566

```

```

567     return hsl;
568 }
569
570
571 public void HitungBiStatic()
572 {
573     snum_db = 10 * Math.Log10(snum);
574
575     double thetaR;    // untuk proses looping (iterator sudut)
576     double phiR;
577     TriangularFace tri_face;
578     facetRCSDData fRCS2;
579     RCLayerResult layerRC;
580     layerRC.RCpara1 = -1;
581     layerRC.RCperp1 = -1;
582
583     wave = C / (frekuensi * 1E9);
584     corel = jarak_korelasi / wave;
585     delstd = standar_deviasi;
586     delsq = Pow2(delstd);
587     bk = 2 * Math.PI / wave;
588     cfac1 = Math.Exp(-4 * bk * bk * delsq);
589     cfac2 = 4 * Math.PI * Pow2((bk * corel)) * delsq;
590
591     // pola loop
592     cpi = Math.Cos(phi_kejadian * rad);
593     spi = Math.Sin(phi_kejadian * rad);
594     cti = Math.Cos(theta_kejadian * rad);
595     sti = Math.Sin(theta_kejadian * rad);
596
597     ui = sti * cpi;
598     vi = sti * spi;
599     wi = cti;
600     DOi[0, 0] = ui;
601     DOi[0, 1] = vi;
602     DOi[0, 2] = wi;
603
604     uui = cti * cpi;
605     vvi = cti * spi;
606     wwi = -sti;
607
608     itheta = (int)(Math.Floor((theta1 - theta0) / del_theta) + 1.0);
609     iphi = (int)(Math.Floor((phi1 - phi0) / del_phi) + 1.0);
610     RCStheta = new double[iphil, itheta];
611     RCSphi = new double[iphil, itheta];
612
613     if (polarisasi_kejadian == Polarisasi.TM)
614     {

```

```

615         EOt = 1.0;
616         EOp = 0.0;
617     }
618     else if (polarisasi_kejadian == Polarisasi.TE)
619     {
620         EOt = 0.0;
621         EOp = 1.0;
622     }
623
624     // Perjalanan dimulai
625     ComplexDouble sumt = 0.0;
626     ComplexDouble sump = 0.0;
627     double sumdt = 0.0;
628     double sumdp = 0.0;
629
630
631     // incident field pada koordinat kartesian global
632     EO_InF[0, 0] = EOt * uui - EOp * spi;
633     EO_InF[0, 1] = EOt * vvi + EOp * cpi;
634     EO_InF[0, 2] = EOt * wwi;
635
636     int i1;
637     int i2;
638     for (i1 = 0; i1 < iphi; ++i1) // awal perjalanan perhitungan
639     {
640         for (i2 = 0; i2 < itheta; ++i2)
641         {
642             fParent.ProgressBar1.Increment(1);
643
644             thetaR = (theta0 + i2 * del_theta) * rad;
645             phiR = (phi0 + i1 * del_phi) * rad;
646
647             // Sudut global dan arah
648             st = Math.Sin(thetaR); ct = Math.Cos(thetaR);
649             sp = Math.Sin(phiR); cp = Math.Cos(phiR);
650             u = st * cp;
651             v = st * sp;
652             w = ct;
653             DO[0, 0] = u;
654             DO[0, 1] = v;
655             DO[0, 2] = w;
656             uu = ct * cp;
657             vv = ct * sp;
658             ww = -st;
659
660             // memulai loop untuk setiap segitiga
661             sumt = 0.0;
662             sump = 0.0;

```

```

663         sumdt = 0.0;
664         sumdp = 0.0;
665
666         for (int m = 0; m < entitas_segitiga.Count; ++m)
667         {
668             if (entitas_segitiga[m].GetType() != trifaceType)
669                 continue;
670
671             tri_face = (TriangularFace)entitas_segitiga[m];
672
673             fRCS2 = facetRCSBiStatic(thetaR, phiR, tri_face, EO_InF);
674
675             // Penjumlahan seluruh segitiga
676             sumt = sumt + fRCS2.Ets;
677             sump = sump + fRCS2.Eps;
678             sumdt = sumdt + ComplexMath.Absolute(fRCS2.Etd);
679             sumdp = sumdp + ComplexMath.Absolute(fRCS2.Epd);
680         }
681
682         // Hasil perhitungan
683         RCStheta[i1, i2] = 10 * Math.Log10(4 * Math.PI * cfac1
684             * (Pow2(ComplexMath.Absolute(sumt))
685             + Math.Sqrt(1 - Pow2(cfac1)) * sumdt)
686             / (Pow2(wave) + snum));
687
688         RCSphi[i1, i2] = 10 * Math.Log10(4 * Math.PI * cfac1
689             * (Pow2(ComplexMath.Absolute(sump))
690             + Math.Sqrt(1 - Pow2(cfac1)) * sumdp)
691             / (Pow2(wave) + snum));
692     }
693 }
694 }
695
696 public void HitungBiStaticFreq()
697 {
698     snum_db = 10 * Math.Log10(snum);
699
700     double thetaR;
701     double phiR;
702     int iFreq;
703     TriangularFace tri_face;
704     facetRCSData fRCS2;
705
706     RCLayerResult layerRC;
707     layerRC.RCpara1 = -1;
708     layerRC.RCperp1 = -1;
709
710     // pola loop

```

```

711     cpi = Math.Cos(phi_kejadian * rad);
712     spi = Math.Sin(phi_kejadian * rad);
713     cti = Math.Cos(theta_kejadian * rad);
714     sti = Math.Sin(theta_kejadian * rad);
715
716     ui = sti * cpi;
717     vi = sti * spi;
718     wi = cti;
719     DOi[0, 0] = ui;
720     DOi[0, 1] = vi;
721     DOi[0, 2] = wi;
722
723     uui = cti * cpi;
724     vvi = cti * spi;
725     wwi = -sti;
726     iFreq = (int)(Math.Floor((F_frekuensi1 - F_frekuensi0)
727         / F_del_frekuensi) + 1.0);
728     RCStheta = new double[iFreq, 1];
729     RCSphi = new double[iFreq, 1];
730
731     if (polarisasi_kejadian == Polarisasi.TM)
732     {
733         EOt = 1.0;
734         EOp = 0.0;
735     }
736     else if (polarisasi_kejadian == Polarisasi.TE)
737     {
738         EOp = 1.0;
739         EOt = 0.0;
740     }
741
742     ComplexDouble sumt = 0.0;
743     ComplexDouble sump = 0.0;
744     double sumdt = 0.0;
745     double sumdp = 0.0;
746
747     // Bidang kejadian pada koordinat kartesian global
748     EO_InF[0, 0] = EOt * uui - EOp * spi;
749     EO_InF[0, 1] = EOt * vvi + EOp * cpi;
750     EO_InF[0, 2] = EOt * wwi;
751
752     thetaR = F_Theta * rad;
753     phiR = F_Phi * rad;
754
755     int i1;
756     frekuensi = F_frekuensi0;
757     for (i1 = 0; i1 < iFreq; ++i1)
758     {

```

```

759     wave = C / (frekuensi * 1E9);
760     corel = jarak_korelasi / wave;
761     delstd = standar_deviasi;
762     delsq = Pow2(delstd);
763     bk = 2 * Math.PI / wave;
764     cfac1 = Math.Exp(-4 * Pow2(bk) * delsq);
765     cfac2 = 4 * Math.PI * Pow2(bk * corel) * delsq;
766
767     fParent.ProgressBar1.Increment(1);
768
769     // Sudut global dan arah
770     st = Math.Sin(thetaR); ct = Math.Cos(thetaR);
771     sp = Math.Sin(phiR); cp = Math.Cos(phiR);
772     u = st * cp;
773     v = st * sp;
774     w = ct;
775     DO[0, 0] = u;
776     DO[0, 1] = v;
777     DO[0, 2] = w;
778     uu = ct * cp;
779     vv = ct * sp;
780     ww = -st;
781
782     sumt = 0.0;
783     sump = 0.0;
784     sumdt = 0.0;
785     sumdp = 0.0;
786
787     for (int m = 0; m < entitas_segitiga.Count; ++m)
788     {
789         if (entitas_segitiga[m].GetType() != trifaceType)
790             continue;
791
792         tri_face = (TriangularFace)entitas_segitiga[m];
793
794         fRCS2 = facetRCSBiStatic(thetaR, phiR, tri_face, EO_InF);
795
796         sumt = sumt + fRCS2.Ets;
797         sump = sump + fRCS2.Eps;
798         sumdt = sumdt + ComplexMath.Absolute(fRCS2.Etd);
799         sumdp = sumdp + ComplexMath.Absolute(fRCS2.Epd);
800     }
801
802     // Hasil perhitungan
803     RCStheta[i1, 0] = 10 * Math.Log10(4 * Math.PI * cfac1
804         * (Pow2(ComplexMath.Absolute(sumt))
805         + Math.Sqrt(1 - Pow2(cfac1)) * sumdt)
806         / (Pow2(wave) + snum));

```



```

807         RCSphi[i1, 0] = 10 * Math.Log10(4 * Math.PI * cfac1
808             * (Pow2(ComplexMath.Absolute(ump))
809             + Math.Sqrt(1 - Pow2(cfac1)) * sumdp)
810             / (Pow2(wave)) + snum);
811
812         frekuensi = frekuensi + F_del_frekuensi;
813     }
814 }
815
816 private facetRCSData facetRCSBiStatic(double thetaR, double phiR,
817     TriangularFace tri_face, ComplexDoubleMatrix EO_InF1)
818 {
819     facetRCSData hsl;
820
821     // Tes apakah bagian depan teriluminasi
822     ndotk = u * tri_face.normal.x +
823         v * tri_face.normal.y +
824         w * tri_face.normal.z;
825
826     nidotk = ui * tri_face.normal.x +
827         vi * tri_face.normal.y +
828         wi * tri_face.normal.z;
829
830     // bagian bayangan
831     // if (nidotk <= 0)
832     if (nidotk <= 0 || ndotk <= 0)
833     {
834         hsl.Etd = 0.0;
835         hsl.Epd = 0.0;
836         hsl.Ets = 0.0;
837         hsl.Eps = 0.0;
838     }
839
840     // bagian yang teriluminasi
841     else
842     {
843         T1[0, 0] = tri_face.cosA; T1[0, 1] = tri_face.sinA;
844         T1[1, 0] = -tri_face.sinA; T1[1, 1] = tri_face.cosA;
845         T1[2, 2] = 1;
846         T2[0, 0] = tri_face.cosB; T2[0, 2] = -tri_face.sinB;
847         T2[1, 1] = 1;
848         T2[2, 0] = tri_face.sinB; T2[2, 2] = tri_face.cosB;
849
850         // Transformasi kuantitas insiden
851         D0i.Transpose();
852         D1i = T1 * D0i;
853         D2i = T2 * D1i; // D2 --> matrix (3, 1)
854         D0i.Transpose();

```

```

855     ui2 = D2i[0, 0];
856     vi2 = D2i[1, 0];
857     wi2 = D2i[2, 0];
858
859     // Hitung sudut spherical kejadian pada koodinat lokal
860     sti2 = Math.Sqrt(Pow2(ui2) + Pow2(vi2)) * (double)Math.Sign(wi2);
861     theta_i2 = Math.Asin(sti2);
862     cti2 = Math.Cos(theta_i2);
863
864     // phi_i2 = Math.Atan2(vi2, ui2 + snum);
865     phi_i2 = Math.Atan2(vi2, ui2);
866     cpi2 = Math.Cos(phi_i2);
867     spi2 = Math.Sin(phi_i2);
868
869     // Transformasi kuantitas observasi
870     D0.Transpose();
871     D1 = T1 * D0;
872     D2 = T2 * D1; // D2 --> matrix (3, 1)
873     D0.Transpose();
874     u2 = D2[0, 0];
875     v2 = D2[1, 0];
876     w2 = D2[2, 0];
877
878     st2 = Math.Sqrt(Pow2(u2) + Pow2(v2)) * (double)Math.Sign(w2);
879     theta_2 = Math.Asin(st2);
880     ct2 = Math.Cos(theta_2);
881
882     // phi_2 = Math.Atan2(v2, u2 + snum);
883     phi_2 = Math.Atan2(v2, u2);
884     cp2 = Math.Cos(phi_2);
885     sp2 = Math.Sin(phi_2);
886
887     float[][] vert1 = tri_face.GetVertice();
888
889     // "Incident field" pada koordinat kartesian lokal
890     EO_InF1.Transpose();
891     E2_InF = T2 * T1 * EO_InF1; // e2 --> matrix (3, 1)
892     EO_InF1.Transpose();
893
894     Et2 = E2_InF[0, 0] * cti2 * cpi2
895           + E2_InF[1, 0] * cti2 * spi2 - E2_InF[2, 0] * sti2;
896     Ep2 = -E2_InF[0, 0] * spi2 + E2_InF[1, 0] * cpi2;
897
898     if (tipe_koef_ref == TipeKoefisienRefleksi.General)
899     {
900         // koefisien refleksi (Rs dinormalisasi terhadap eta0)
901         RCperp = -1 / (2 * Rs * cti2 + 1);
902         RCpara = 0;

```

```

903         if ((2 * Rs + cti2) != 0)
904             RCpara = -cti2 / (2 * Rs + cti2);
905     }
906     else
907     {
908         RCLayerResult layerRC = RCLayer(theta_i2, cti2, sti2,
909             tri_face.autoCadColor, frekuensi * 1E9);
910         RCperp = layerRC.RCperp1;
911         RCpara = layerRC.RCpara1;
912     }
913
914     // Surface Current pada koordinat kartesian lokal
915     Jx2 = (-Et2 * cpi2 * RCpara) + (Ep2 * spi2 * RCperp);
916     Jy2 = (-Et2 * spi2 * RCpara) - (Ep2 * cpi2 * RCperp);
917
918     // Integral luasan
919     // Fase pada vertex segitiga
920     DpX = bk * ((vert1[0][0] - vert1[2][0]) * (u + ui) +
921         (vert1[0][1] - vert1[2][1]) * (v + vi) +
922         (vert1[0][2] - vert1[2][2]) * (w + wi));
923
924     DqX = bk * ((vert1[1][0] - vert1[2][0]) * (u + ui) +
925         (vert1[1][1] - vert1[2][1]) * (v + vi) +
926         (vert1[1][2] - vert1[2][2]) * (w + wi));
927
928     DoX = bk * (vert1[2][0] * (u + ui) +
929         vert1[2][1] * (v + vi) + vert1[2][2] * (w + wi));
930
931     Ic = Ic_Eval(tri_face.area, DpX, DqX, DoX, panjang_daerah);
932
933     // Tambahkan komponen difusi
934     Edif = cfac2 * tri_face.area * Pow2(ct2) *
935         Math.Exp(-Pow2(corel * Math.PI * st2 / wave));
936
937     // Komponen bidang Scattered segitiga pada koordinat lokal
938     Es2[0, 0] = Jx2 * Ic;
939     Es2[1, 0] = Jy2 * Ic;
940     Es2[2, 0] = 0;
941
942     Ed2[0, 0] = Jx2 * Edif;
943     Ed2[1, 0] = Jy2 * Edif;
944     Ed2[2, 0] = 0;
945
946     // Transformasi balik ke koordinat global
947     T2.Transpose();
948     Es1 = T2 * Es2; // 3x3 X 3x1 = 3x1
949     T2.Transpose();
950

```

```

951         T1.Transpose();
952         Es0 = T1 * Es1; // 3x1
953         T1.Transpose();
954
955         T2.Transpose();
956         Ed1 = T2 * Ed2;
957         T2.Transpose();
958
959         T1.Transpose();
960         Ed0 = T1 * Ed1;
961         T1.Transpose();
962
963         hsl.Ets = Es0[0, 0] * uu + Es0[1, 0] * vv + Es0[2, 0] * ww;
964         hsl.Eps = Es0[0, 0] * (-sp) + Es0[1, 0] * cp;
965         hsl.Etd = Ed0[0, 0] * uu + Ed0[1, 0] * vv + Ed0[2, 0] * ww;
966         hsl.Epd = Ed0[0, 0] * (-sp) + Ed0[1, 0] * cp;
967     }
968
969     return hsl;
970 }
971
972
973 #region Fungsi_Perhitungan_PO
974
975 private ComplexDouble Ic_Eval(double area, double DpX1,
976     double DqX1, double DoX1, double panjang_daerah1)
977 {
978     double DdX1;
979
980     ComplexDouble expDoX1;
981     ComplexDouble expDpX1;
982     ComplexDouble expDqX1;
983
984     DdX1 = DqX1 - DpX1;
985     expDoX1 = ComplexMath.Exp(jn * DoX1);
986     expDpX1 = ComplexMath.Exp(jn * DpX1);
987     expDqX1 = ComplexMath.Exp(jn * DqX1);
988
989     // Kasus-kasus spesial
990     int n;
991     int nn;
992     ComplexDouble sic;
993     ComplexDouble Ic1;
994
995     // kasus #1
996     if (Math.Abs(DpX1) < panjang_daerah1 &&
997         Math.Abs(DqX1) >= panjang_daerah1)
998     {

```

```

999         sic = 0.0;
1000
1001         for (n = 0; n <= jumlah_term; ++n)
1002         {
1003             sic = sic + ComplexPowUInt(jn * DpX1, n)
1004                 / factorial(n)
1005                 * (-Co / (n + 1) + expDqX1 * (Co * recG(n, -DqX1)));
1006         }
1007         Ic1 = expDoX1 * sic * (2 * area) / jn / DqX1;
1008     }
1009
1010     // kasus #2
1011     else if (Math.Abs(DpX1) < panjang_daerah1 &&
1012             Math.Abs(DqX1) < panjang_daerah1)
1013     {
1014         sic = 0.0;
1015
1016         for (n = 0; n <= jumlah_term; ++n)
1017             for (nn = 0; nn <= jumlah_term; ++nn)
1018             {
1019                 sic = sic + ComplexPowUInt(jn * DpX1, n)
1020                     * ComplexPowUInt(jn * DqX1, nn)
1021                     / factorial(nn + n + 2)
1022                     * Co;
1023             }
1024         Ic1 = expDoX1 * sic * (2 * area);
1025     }
1026
1027     // kasus #3
1028     else if (Math.Abs(DpX1) >= panjang_daerah1 &&
1029             Math.Abs(DqX1) < panjang_daerah1)
1030     {
1031         sic = 0.0;
1032
1033         for (n = 0; n <= jumlah_term; ++n)
1034         {
1035             sic = sic + ComplexPowUInt(jn * DqX1, n)
1036                 / factorial(n)
1037                 * Co * recG(n + 1, -DpX1) / (n + 1);
1038         }
1039         Ic1 = expDoX1 * expDpX1 * sic * (2 * area);
1040     }
1041
1042     // kasus #4
1043     else if (Math.Abs(DpX1) >= panjang_daerah1 &&
1044             Math.Abs(DqX1) >= panjang_daerah1 &&
1045             Math.Abs(DdX1) < panjang_daerah1)
1046     {

```

```

1047         sic = 0.0;
1048
1049         for (n = 0; n <= jumlah_term; ++n)
1050         {
1051             sic = sic + ComplexPowUInt(jn * DdX1, n)
1052                 / factorial(n)
1053                 * ((expDpX1 * Co / (n + 1)) + (-Co * recG(n, DqX1)));
1054         }
1055         Ic1 = expDoX1 * sic * (2 * area) / jn / DqX1;
1056     }
1057
1058     // Kasus terakhir
1059     else
1060     {
1061         Ic1 = (expDpX1 * Co / DpX1 / DdX1
1062             - expDqX1 * Co / DqX1 / DdX1
1063             - Co / DpX1 / DqX1)
1064             * expDoX1 * (2 * area);
1065     }
1066     // akhir kasus spesial
1067
1068     return Ic1;
1069 }
1070
1071 public struct RCLayerResult
1072 {
1073     public ComplexDouble RCpara1;
1074     public ComplexDouble RCperp1;
1075 }
1076
1077 private RCLayerResult RCLayer(double thetaI, double cos_tI,
1078     double sin_tI, short NoColor, double freq)
1079 {
1080     RCLayerResult hsl;
1081     hsl.RCpara1 = -1; // default value by PEC
1082     hsl.RCperp1 = -1;
1083
1084     double RSval;
1085
1086     MaterialsDataSet.LayerValueRow lvRow =
1087         materialsDataSet1.LayerValue.FindByNoColor(NoColor);
1088
1089     if (lvRow.UseResistanceSurface)
1090     {
1091         RSval = lvRow.ResistanceSurfaceValue;
1092         hsl.RCperp1 = -1 / (2 * RSval * cos_tI + 1);
1093         hsl.RCpara1 = 0;
1094     }

```

```

1095         if ((2 * RSval + cos_tI) != 0)
1096             hsl.RCpara1 = -cos_tI / (2 * RSval + cos_tI);
1097
1098         return hsl;
1099     }
1100
1101     else if (lvRow.UseMaterial)
1102     {
1103         MaterialsDataSet.LayerMaterialsRow[] lmRows =
1104             (MaterialsDataSet.LayerMaterialsRow[])
1105             materialsDataSet1.LayerMaterials.Select(
1106                 "NoColor = " + NoColor.ToString());
1107
1108         double tebalMaterial;
1109         ComplexDouble er = 1.0;
1110         ComplexDouble mr = 1.0;
1111
1112         ComplexDouble er_p = 1.0; // _p ==> sebelumnya
1113         ComplexDouble mr_p = 1.0; // untuk layer awal
1114         double theta_p = thetaI; // untuk layer awal (layer-0)
1115
1116         ReflCoeffResult RCoeff1;
1117         ComplexDouble G1para;
1118         ComplexDouble G1perp;
1119
1120         double v_ph;
1121         double lamda_ph;
1122         double phase = 0.0;
1123
1124         int nLayer = lmRows.Length; // jumlah layer
1125
1126         if (nLayer == 0) // no layer, so... use PEC
1127         {
1128             hsl.RCpara1 = -1;
1129             hsl.RCperp1 = -1;
1130         }
1131
1132         #region Composite layer
1133         else if (!lvRow.OnPEC) // Multiple Composite layer
1134         {
1135             Mpara = new ComplexDoubleMatrix(2, 2);
1136             Mpara[0, 0] = 1; Mpara[0, 1] = 0;
1137             Mpara[1, 0] = 0; Mpara[1, 1] = 1;
1138
1139             Mperp = new ComplexDoubleMatrix(2, 2);
1140             Mperp[0, 0] = 1; Mperp[0, 1] = 0;
1141             Mperp[1, 0] = 0; Mperp[1, 1] = 1;
1142

```

```

1143     int ix;
1144     for (ix = 0; ix < nLayer; ++ix)
1145     {
1146         er = lmRows[ix].RelativeElectricPermittivity -
1147             jn * lmRows[ix].RelativeElectricPermittivity *
1148             lmRows[ix].LossTangent;
1149         mr = lmRows[ix].RelativeMagneticPermeabilityReal -
1150             jn * lmRows[ix].RelativeMagneticPermeabilityImaginer;
1151         tebalMaterial = lmRows[ix].Thickness * 0.001;
1152
1153         RCoeff1 = ReflCoeff(er_p, mr_p, er, mr, theta_p);
1154         G1para = RCoeff1.gammaPara;
1155         G1perp = RCoeff1.gammaPerp;
1156         theta_p = RCoeff1.thetaT;
1157
1158         v_ph = C / Math.Sqrt(er.Real * mr.Real); // C = 3e8
1159         lamda_ph = v_ph / freq;
1160         phase = (2 * Math.PI / lamda_ph) * tebalMaterial;
1161
1162         // Formulasi Matrix
1163         M1para[0, 0] = ComplexMath.Exp(jn * phase);
1164         M1para[0, 1] = G1para * ComplexMath.Exp(-jn * phase);
1165         M1para[1, 0] = G1para * ComplexMath.Exp(jn * phase);
1166         M1para[1, 1] = ComplexMath.Exp(-jn * phase);
1167         Mpara = Mpara * M1para;
1168
1169         M1perp[0, 0] = ComplexMath.Exp(jn * phase);
1170         M1perp[0, 1] = G1perp * ComplexMath.Exp(-jn * phase);
1171         M1perp[1, 0] = G1perp * ComplexMath.Exp(jn * phase);
1172         M1perp[1, 1] = ComplexMath.Exp(-jn * phase);
1173         Mperp = Mperp * M1perp;
1174
1175         er_p = er;
1176         mr_p = mr;
1177     }
1178
1179     // koefisien refleksi material terakhir dengan udara
1180     RCoeff1 = ReflCoeff(er, mr, 1, 1, theta_p);
1181     G1para = RCoeff1.gammaPara;
1182     G1perp = RCoeff1.gammaPerp;
1183
1184     // Formulasi Matrix
1185     M1para[0, 0] = ComplexMath.Exp(jn * phase);
1186     M1para[0, 1] = G1para * ComplexMath.Exp(-jn * phase);
1187     M1para[1, 0] = G1para * ComplexMath.Exp(jn * phase);
1188     M1para[1, 1] = ComplexMath.Exp(-jn * phase);
1189     Mpara = Mpara * M1para;
1190

```



```

1191     M1perp[0, 0] = ComplexMath.Exp(jn * phase);
1192     M1perp[0, 1] = G1perp * ComplexMath.Exp(-jn * phase);
1193     M1perp[1, 0] = G1perp * ComplexMath.Exp(jn * phase);
1194     M1perp[1, 1] = ComplexMath.Exp(-jn * phase);
1195     Mperp = Mperp * M1perp;
1196
1197     // Hirung Koeffisien Refleksi
1198     hsl.RCpara1 = Mpara[1, 0] / Mpara[0, 0];
1199     hsl.RCperp1 = Mperp[1, 0] / Mperp[0, 0];
1200 }
1201 #endregion
1202
1203 #region Composite layer On PEC
1204 else if (nLayer > 0 && lvRow.OnPEC) // Composite On PEC
1205 {
1206     int ix;
1207
1208     DoubleMatrix PEC = new DoubleMatrix(2, 2);
1209     PEC[0, 0] = 1; PEC[0, 1] = 0;
1210     PEC[1, 0] = -1; PEC[1, 1] = 0;
1211     // initialize the wave matrix - parallel pol
1212     ComplexDoubleMatrix WMatrix_para =
1213         new ComplexDoubleMatrix(2, 2);
1214     WMatrix_para[0, 0] = 1; WMatrix_para[0, 1] = 0;
1215     WMatrix_para[1, 0] = 0; WMatrix_para[1, 1] = 1;
1216     // initialize the wave matrix - perpendicular pol
1217     ComplexDoubleMatrix WMatrix_perp =
1218         new ComplexDoubleMatrix(2, 2);
1219     WMatrix_perp = WMatrix_para;
1220
1221     ComplexDoubleMatrix T_para =
1222         new ComplexDoubleMatrix(2, 2);
1223     ComplexDoubleMatrix T_perp =
1224         new ComplexDoubleMatrix(2, 2);
1225
1226     double Z0 = 1; //normalized impedance of free space
1227     ComplexDouble Z_para;
1228     ComplexDouble Z_perp;
1229     ComplexDouble Z_para_prev = Z0; // untuk layer-0
1230     ComplexDouble Z_perp_prev = Z0;
1231
1232     ComplexDouble Tau_para;
1233     ComplexDouble Tau_perp;
1234
1235     ComplexDouble phaseX; // phase using ComplexDouble
1236
1237     for (ix = 0; ix < nLayer; ++ix)
1238     {

```

```

1239     er = lmRows[ix].RelativeElectricPermittivity -
1240         jn * lmRows[ix].RelativeElectricPermittivity *
1241         lmRows[ix].LossTangent;
1242     mr = lmRows[ix].RelativeMagneticPermeabilityReal
1243         - jn * lmRows[ix].RelativeMagneticPermeabilityImaginer;
1244     tebalMaterial = lmRows[ix].Thickness * 0.001;
1245
1246     // impedance of layer
1247     Z_para = ComplexMath.Sqrt((er / mr - Pow2(sin_tI)))
1248         / (er / mr * cos_tI);
1249     Z_perp = cos_tI /
1250         ComplexMath.Sqrt(er / mr - Pow2(sin_tI));
1251     G1para = (Z_para - Z_para_prev) / (Z_para + Z_para_prev);
1252     Tau_para = 1 + G1para;
1253     G1perp = (Z_perp - Z_perp_prev) / (Z_perp + Z_perp_prev);
1254     Tau_perp = 1 + G1perp;
1255     phaseX = bk * tebalMaterial *
1256         ComplexMath.Sqrt(er / mr - Pow2(sin_tI));
1257     // update untuk layer berikutnya
1258     Z_para_prev = Z_para;
1259     Z_perp_prev = Z_perp;
1260
1261     // Update the Wave Matrix
1262     T_para[0, 0] = ComplexMath.Exp(jn * phaseX);
1263     T_para[0, 1] = G1para * ComplexMath.Exp(-jn * phaseX);
1264     T_para[1, 0] = G1para * ComplexMath.Exp(jn * phaseX);
1265     T_para[1, 1] = ComplexMath.Exp(-jn * phaseX);
1266
1267     WMatrix_para = (1 / Tau_para) * WMatrix_para * T_para;
1268
1269     T_perp[0, 0] = ComplexMath.Exp(jn * phaseX);
1270     T_perp[0, 1] = G1perp * ComplexMath.Exp(-jn * phaseX);
1271     T_perp[1, 0] = G1perp * ComplexMath.Exp(jn * phaseX);
1272     T_perp[1, 1] = ComplexMath.Exp(-jn * phaseX);
1273
1274     WMatrix_perp = 1 / Tau_perp * WMatrix_perp * T_perp;
1275 }
1276
1277 // finally, the last layer is PEC;
1278 WMatrix_para = WMatrix_para * PEC;
1279 WMatrix_perp = WMatrix_perp * PEC;
1280
1281 // Hitung Koefisien Refleksi
1282 hsl.RCpara1 = WMatrix_para[1, 0] / WMatrix_para[0, 0];
1283 hsl.RCperp1 = WMatrix_perp[1, 0] / WMatrix_perp[0, 0];
1284 }
1285 #endregion
1286

```

```

1287     }
1288
1289     return hsl;
1290 }
1291
1292 struct ReflCoeffResult
1293 {
1294     public ComplexDouble gammaPara;
1295     public ComplexDouble gammaPerp;
1296     public double thetaT;
1297     public bool TIR;
1298 }
1299
1300 // Desc: Fungsi ini untuk menghitung koefisien Refleksi
1301 //       dan sudut transmisi beserta pengujian terjadinya
1302 //       Total Internal Reflection (TIR)
1303 private ReflCoeffResult ReflCoeff(ComplexDouble er1,
1304     ComplexDouble mr1, ComplexDouble er2,
1305     ComplexDouble mr2, double thetaI)
1306 {
1307     ReflCoeffResult hslX;
1308     hslX.TIR = false;
1309
1310     double sinThetaT = Math.Sin(thetaI) *
1311         Math.Sqrt(er1.Real * mr1.Real / (er2.Real * mr2.Real));
1312
1313
1314     if (sinThetaT > 1)
1315     {
1316         hslX.TIR = true;
1317         hslX.thetaT = Math.PI / 2;
1318     }
1319     else
1320     {
1321         hslX.thetaT = Math.Asin(sinThetaT);
1322     }
1323
1324     ComplexDouble n1 = ComplexMath.Sqrt(mr1 * m_fs / (er1 * e_fs));
1325     ComplexDouble n2 = ComplexMath.Sqrt(mr2 * m_fs / (er2 * e_fs));
1326
1327     hslX.gammaPerp = (n2 * Math.Cos(thetaI) - n1 * Math.Cos(hslX.thetaT))
1328         / (n2 * Math.Cos(thetaI) + n1 * Math.Cos(hslX.thetaT));
1329
1330     hslX.gammaPara = (n2 * Math.Cos(hslX.thetaT) - n1 * Math.Cos(thetaI))
1331         / (n2 * Math.Cos(hslX.thetaT) + n1 * Math.Cos(thetaI));
1332
1333     return hslX;
1334 }

```

```

1335
1336     #endregion
1337
1338
1339     #region Fungsi_Perhitungan_Tambahan
1340     // calculates the recursive function G
1341     ComplexDouble recG(int n, double w)
1342     {
1343         ComplexDouble g;
1344         ComplexDouble g0;
1345         ComplexDouble jw = jn * w;
1346         g = (ComplexMath.Exp(jw) - 1) / jw;
1347         if (n > 0)
1348         {
1349             for (int m = 0; m < n; ++m)
1350             {
1351                 g0 = g;
1352                 g = (ComplexMath.Exp(jw) - g0 * n) / jw;
1353             }
1354         }
1355
1356         return g;
1357     }
1358
1359     public static double factorial(int n)
1360     {
1361         double hsl = 1;
1362
1363         switch (n)
1364         {
1365             case 0:
1366                 hsl = 1;
1367                 break;
1368             case 1:
1369                 hsl = 1;
1370                 break;
1371             case 2:
1372                 hsl = 2;
1373                 break;
1374             case 3:
1375                 hsl = 6;
1376                 break;
1377             case 4:
1378                 hsl = 24;
1379                 break;
1380             case 5:
1381                 hsl = 120;
1382                 break;

```

```

1383         case 6:
1384             hsl = 270;
1385             break;
1386         case 7:
1387             hsl = 5040;
1388             break;
1389         case 8:
1390             hsl = 40320;
1391             break;
1392         case 9:
1393             hsl = 362880;
1394             break;
1395         case 10:
1396             hsl = 3628800;
1397             break;
1398     }
1399
1400     int i;
1401     if (n > 10)
1402     {
1403         hsl = 3628800;
1404         for (i = 11; i <= n; ++i)
1405             hsl = hsl * i;
1406     }
1407
1408     return hsl;
1409 }
1410
1411 public static ComplexDouble ComplexPowUInt(ComplexDouble x, int n)
1412 {
1413     if (n < 0)
1414         return 0.0;
1415
1416     if (n == 0)
1417         return 1;
1418
1419     if (x == 0.0)
1420         return 0.0;
1421
1422     ComplexDouble hsl = new ComplexDouble(x);
1423     ComplexDouble x2 = x * x;
1424     ComplexDouble x4 = x2 * x2;
1425     ComplexDouble x8 = x4 * x4;
1426     ComplexDouble x16 = x8 * x8;
1427
1428     int n1 = n - 1;
1429     while (n1 > 1)
1430     {

```

```

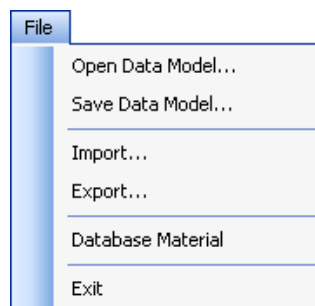
1431         if (n1 >= 16)
1432         {
1433             hsl = hsl * x16;
1434             n1 = n1 - 16;
1435         }
1436
1437         else if (n1 >= 8)
1438         {
1439             hsl = hsl * x8;
1440             n1 = n1 - 8;
1441         }
1442
1443         else if (n1 >= 4)
1444         {
1445             hsl = hsl * x4;
1446             n1 = n1 - 4;
1447         }
1448
1449         else if (n1 >= 2)
1450         {
1451             hsl = hsl * x2;
1452             n1 = n1 - 2;
1453         }
1454     }
1455
1456     if (n1 == 1)
1457         hsl = hsl * x;
1458
1459     return hsl;
1460 }
1461
1462 double Pow2(double A)
1463 {
1464     return A * A;
1465 }
1466
1467 ComplexDouble Pow2(ComplexDouble A)
1468 {
1469     return A * A;
1470 }
1471
1472 #endregion
1473 }
1474 }

```

# Lampiran B

## Penjelasan Menu targetRCS2

### B.1 Menu File



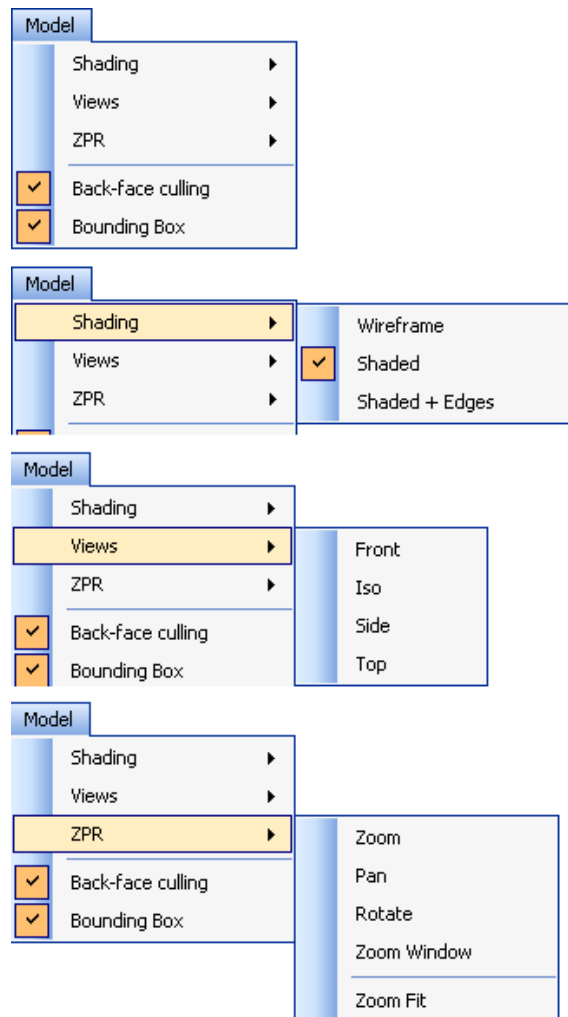
Gambar B.1: targetRCS2 Menu File

Tabel B.1: Fungsi Menu File

No	Menu	Fungsi
1	Open Data Model	Membaca file XMO
2	Save Data Model	Meyimpan file XMO
3	Import	Membaca file model tambahan, seperti file DXF, StL
4	Export	Menyimpan file model tambahan, seperti file DXF, StL
5	Database Material	Mengedit data Material sebagai input permukaan, sebagaimana yang terlihat pada Gambar 3.4
7	Exit	Keluar dari program utama

## B.2 Menu Model

Menu Model digunakan untuk pengolahan tampilan model, baik dari segi cara pandang, penentuan batas dengan kotak, mematikan dan menghidupkan mode back-face culling, dan hal lainnya. Pengolahan tampilan model ini berdasarkan fungsi yang ada pada C# OpenGL Framework.



Gambar B.2: targetRCS2 Menu Model



Tabel B.2: Fungsi Menu Model

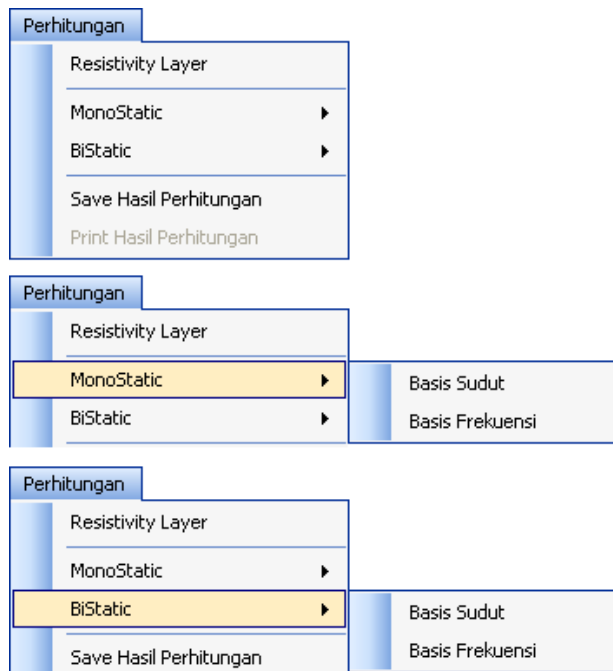
No	Menu	Fungsi
1	Shading	Pemilihan metode untuk menampilkan model
2	View	Pemilihan arah pandang model
3	ZPR	Mengubah fungsi mouse untuk melakukan <i>zoom</i> , <i>pan</i> dan <i>rotate</i>
4	Back-face Culling	Mengaktifkan / menon-aktifkan fungsi back-face culling pada OpenGL
5	Bounding Box	Mengaktifkan / menon-aktifkan kotak batasan model

### B.3 Menu Perhitungan

Menu Perhitungan digunakan untuk pengolahan perhitungan RCS secara monostatis, maupun bistatis. Pada menu ini juga disertakan menu untuk mengolah sifat permukaan obyek, serta penyimpanan hasil perhitungan.

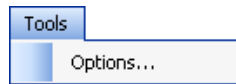
Tabel B.3: Fungsi Menu Perhitungan

No	Menu	Fungsi
1	Resistivity Layer	Mengedit data material pembentuk permukaan obyek, sebagaimana yang terlihat pada Gambar 3.3
2	MonoStatic	Melakukan perhitungan RCS secara monostatis
3	BiStatic	Melakukan perhitungan RCS secara bistatis
4	Save Hasil Perhitungan	Menyimpan hasil perhitungan dalam bentuk file teks



Gambar B.3: targetRCS2 Menu Perhitungan

## B.4 Menu Tools

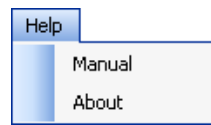


Gambar B.4: targetRCS2 Menu Tools

Tabel B.4: Fungsi Menu Tools

No	Menu	Fungsi
1	Options	Mengubah konfigurasi perhitungan pada software.

## B.5 Menu Help



Gambar B.5: targetRCS2 Menu Help

Tabel B.5: Fungsi Menu Help

No	Menu	Fungsi
1	Manual	Membuka petunjuk penggunaan software
2	About	Informasi mengenai pembuatan software

## Lampiran C

### Keperluan Minimum targetRCS2

Berikut ini merupakan daftar sistem, baik software maupun hardware komputer, yang harus dipenuhi untuk menjalankan software targetRCS2. Keperluan minimum targetRCS2 ini dibuat berdasarkan atas keperluan .Net Framework 2.0 dan sistem grafik OpenGL.

<b>Processor</b>	Processor Pentium 400 MHz, atau AMD Opteron, atau AMD Athlon
<b>Sistem Operasi</b>	Microsoft Windows 98
	Microsoft Windows 2000 SP4
	Microsoft Windows Millennium Edition
	Microsoft Windows XP Professional SP2
	Windows XP Home Edition SP2
	Windows XP Media Center Edition 2002 SP2
	Windows XP Media Center Edition 2004 SP2
	Windows XP Media Center Edition 2005
	Windows XP Starter Edition
	Microsoft Windows 2003
<b>Software Minimum</b>	.NET Framework 2.0 Redistributable
	Microsoft Data Access Components 2.8
	Windows Installer 3.0
<b>RAM</b>	128 megabytes (MB)
<b>Ruang Harddisk</b>	230 MB (200 MB untuk .Net Framework 2.0 Redistributable)
<b>Tampilan</b>	800 x 600 High Color – 16-bit
<b>Memory VGA</b>	8 MB