SANDIA REPORT
SAND2020-9501
Printed August 2020

Sandia National Laboratories

A File Format and API for Dynamic Radar Cross Section Data

Dylan A. Crocker

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

ABSTRACT

Often the Radar Cross-Section (RCS) of a target is incorrectly assumed to be a single number by those unfamiliar with electromagnetic scattering. In actuality, a target's RCS depends on many factors. These factors include radar signal frequency, radar observation angle, as well as target orientation. Another possible parameter (often not considered) is time. The RCS of targets may change over time due to movement, environmental changes, etc. In order to accurately represent the dynamic RCS of a target in a time-stepped analysis, the ability to interface with large RCS datasets efficiently is desired. To this end, a file format and API (written in C++) were developed and are described in this report.

CONTENTS

Nomenclature 7

1. Introduction **9**

2. Background **11**
2.1. Geometry Definition 13
2.2. Polarization Mismatch 15
2.3. Circular Polarization 16

3. File Format **18**
3.1. Time Table 18
3.2. Frequency Table 18
3.3. Aspect Angle Table 20
3.4. RCS Table 20

4. API **22**
4.1. Creating the RCS Data Files 22
4.2. Querying the RCS Data Files 23
4.3. Command Line Interface 24

5. Summary **25**
5.1. Summary of Features 25
5.2. Future Work 25
References 27

References **27**

LIST OF FIGURES

Figure 2-1. Radar coordinate frame (note that $\hat{h}$ is out of the page). 13
Figure 2-2. Target coordinate frame (note that $\hat{h}$ is into the page). 14
Figure 2-3. Polarization mismatch angle α (note that $\hat{k}$ is into the page). 14

Figure 3-1. Time table example. 19
Figure 3-2. Frequency table example. 19
Figure 3-3. Aspect angle table example. 20
Figure 3-4. RCS data (complex scatting length) table example. 21

Nomenclature

API Application Programming Interface

CP Circular Polarization

CSL Complex Scattering Length

FFT Fast Fourier Transform

JSON JavaScript Object Notation

LHCP Left Hand Circular Polarization

M&S Modeling and Simulation

PSM Polarization Scattering Matrix

RCS Radar Cross Section

RHCP Right Hand Circular Polarization

SQL Structured Query Language

1. INTRODUCTION

Modeling and Simulation (M&S) of complex and dynamic systems involving radars are often employed for many different applications. Military battlefield scenario simulations, airport traffic control simulations, and more recently passenger vehicle simulations are just a few M&S examples that may include radar systems. The governing equation for the simulation of a radar's ability to detect and track targets is

$$P^r = \frac{P^t G^r G^t \lambda^2 \sigma}{(4\pi)^3 R^4}. \tag{1.1}$$

which appropriately called "the radar equation" [1]. The variables in (1.1) have the following definitions:

P^r is the power received at the radar from the signal scatted from the target (Watts),
P^t is the power transmitted from the radar (Watts),
G^r is the gain of the radar receive antenna,
G^t is the gain of the radar transmit antenna,
λ is the wavelength of the transmitted signal (meters,)
σ is the Radar Cross Section (RCS) of the target (m^2) and,
R is the distance from the radar to the target in meters.

Notice that the RCS σ is the sole parameter dependent on the target. It is this parameter that fully describes the interaction between an incident wave (from the radar) and the target. Often the RCS of a target is incorrectly assumed to be a single number by those unfamiliar with electromagnetic scattering. In reality, a target's RCS depends on many factors including radar signal polarization and frequency, radar observation angle, as well as the target's orientation. Another possible parameter (often not considered) is time. The RCS of targets may change over time due to movement, environmental changes, etc. Examples include objects surrounded by time varying plasma, changes in flora due to seasons, and target configuration changes (such as a tank rotating its turret). The time variation, coupled with the other varying parameters (e.g., frequency), can cause the RCS datasets necessary for M&S scenarios to become relatively large. Furthermore, phase information is often included with the RCS (for coherent processing) which doubles the storage requirements. In order to accurately represent the dynamic RCS of a target in a time stepped M&S analysis, the ability for the M&S tools to efficiently interface with large RCS datasets is desired. To this end, a file format and API (written in C++) have been developed and are described in the subsequent sections of this report.

The subsequent sections of this report are outlined as follows. Section 2 provides the background for the RCS computations including the geometry definitions assumed and post processing capability (e.g., polarization alignment). The file format, which is based on the SQLite database file format, is detailed in Section 3. The C++ API developed for interfacing with the data is described (with examples) in Section 4. Finally, Section 5 provides a summary including future work.

2. BACKGROUND

In order to calculate the Radar Cross-Section (RCS) of a target, the incident and scattered electromagnetic (EM) fields must be computed. EM simulation tools accomplish this by applying a known incident field and computing the subsequently induced currents on the target. From the currents, the scattered fields can be computed. With both the incident and scattered fields known, the RCS of the target can is computed as

$$\sigma = \lim_{R\to\infty} 4\pi R^2 \frac{p^s}{p^i} = \lim_{R\to\infty} 4\pi R^2 \frac{|\vec{E}^s|^2}{|\vec{E}^i|^2}. \tag{2.1}$$

In (2.1), σ is the radar cross-section in units of meters squared, R is the distance from the radar to the target (meters), p^i and p^s represent incident and scattered power density (W/m^2) respectively, and $\vec{E}^i$ and $\vec{E}^s$ represent incident and scattered electric field vectors (V/m) respectively. The expression using power densities serves to clearly illustrate that RCS is a power quantity; however, the official IEEE definition of RCS is expressed in terms of the electric fields [1]. Although non-obvious from the definition, σ is independent of R due to the fact that the scattered field $\vec{E}^s$ is a function of $1/R$. The limit with respect to R is included to reinforce the fact that the equation is only valid in the far-field.

Due to the far-field separation between the target and radar, $\vec{E}^i$ and $\vec{E}^s$ are plane waves (have no field component in the direction of propagation) and the electric field vector resides in a 2D plane (discussed further in Section 2.1). Thus, the polarization of the field can be described by two orthogonal basis vectors. Often, vertical $\hat{v}$ and horizontal $\hat{h}$ bases are used, which may be considered equivalent to the $\hat{\theta}$ and $\hat{\phi}$ unit vectors defined in the radar's spherical coordinate system. Other bases may be used such as Left and Right circular polarization. Unless specified, general orthogonal polarization basis vectors $\hat{e}_1$ and $\hat{e}_2$ will be used as place holders. Given the polarization of the fields, the RCS is in general a 2×2 dyad[1] as defined by

$$[\sigma] = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}. \tag{2.2}$$

The scattered field $\vec{E}^s$ is determined by the interaction of the incident field $\vec{E}^i$ and the target geometry (essentially induced currents re-radiating). This interaction is dependent on the orientation of the target relative to the incoming signal as well as signal polarization and

[1]Note that the terminology in the dyad components (e.g., σ_{21}) is in the Receive-Transmit format. That is, σ_{21} refers to the RCS for an incident field of $\hat{e}_1$ polarization and a scattered field of $\hat{e}_2$ polarization.

frequency[2]. The interaction is described mathematically by the Polarization Scattering Matrix (PSM) [**S**] as defined by

$$\begin{bmatrix} \hat{e}_1 \cdot \vec{E}^s \\ \hat{e}_2 \cdot \vec{E}^s \end{bmatrix} = [\mathbf{S}] \begin{bmatrix} \hat{e}_1 \cdot \vec{E}^i \\ \hat{e}_2 \cdot \vec{E}^i \end{bmatrix}. \tag{2.3}$$

Where

$$[\mathbf{S}] = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \tag{2.4}$$

and each component S_{ij} represents the ratio of the scattered field with polarization i to the incident field with polarization j. In general, the values of a target's PSM are complex since they contain phase information. The PSM is related to the RCS through

$$[\mathbf{S}] = \begin{bmatrix} |S_{11}|e^{j\phi_{11}} & |S_{12}|e^{j\phi_{12}} \\ |S_{21}|e^{j\phi_{21}} & |S_{22}|e^{j\phi_{22}} \end{bmatrix} = (4\pi R^2)^{-1/2} \begin{bmatrix} \sqrt{\sigma_{11}}e^{j\phi_{11}} & \sqrt{\sigma_{12}}e^{j\phi_{12}} \\ \sqrt{\sigma_{21}}e^{j\phi_{21}} & \sqrt{\sigma_{22}}e^{j\phi_{22}} \end{bmatrix}. \tag{2.5}$$

Where ϕ represents the field phase information which is necessary for coherent processing (e.g., converting from frequency to time/range domain). Since RCS is a power quantity, it does not contain phase information by definition. Due to this, EM engineers often define a parameter γ referred to as the Complex Scattering Length (CSL)[3]. The relationship between CSL and RCS is described by

$$\gamma = \sqrt{\sigma}e^{j\phi}, \tag{2.6}$$

and conversely

$$\sigma = |\gamma|^2. \tag{2.7}$$

It is the target CSL that is stored in the RCS data files described in this report.

[2]The electrical size of a target is defined in wavelengths λ which is a function of frequency: $\lambda = c/f$ where c is the propagation velocity of the wave. Due to this relationship, the RCS of a target can vary with frequency as it is effectively changing size electrically.

[3]RCS is an area term usually expressed in units of m^2. Subsequently, CSL is a length quantity and has units of meters ($\sqrt{(\mathrm{m}^2)} = \mathrm{m}$).

2.1. Geometry Definition

In order to properly interface RCS datasets with M&S applications, the geometrical assumptions must be clearly defined. The assumed radar coordinate system is illustrated in Fig. 2-1. The direction of the propagating radar signal is defined by the $\hat{k}$ unit vector. Since the radar to target distance is large, the signal is regarded as a plane wave (no element of the fields are in the direction of propagation). The linear polarization basis $\hat{v}$ and $\hat{h}$ is chosen to define the polarization of the radar signal since this is how the data will be stored in the RCS data files. The vertical polarization $\hat{v}$ lies in the $\hat{k}\hat{z}$-plane (or $\hat{k}\hat{\theta}$-plane in standard spherical coordinates) and the horizontal polarization $\hat{h}$ lies normal to the $\hat{k}\hat{z}$-plane (or in the $\hat{k}\hat{\phi}$-plane in standard spherical coordinates). These vertical and horizontal polarizations are respectively equivalent to the TM_z and TE_z polarizations that are often used in the literature [2].

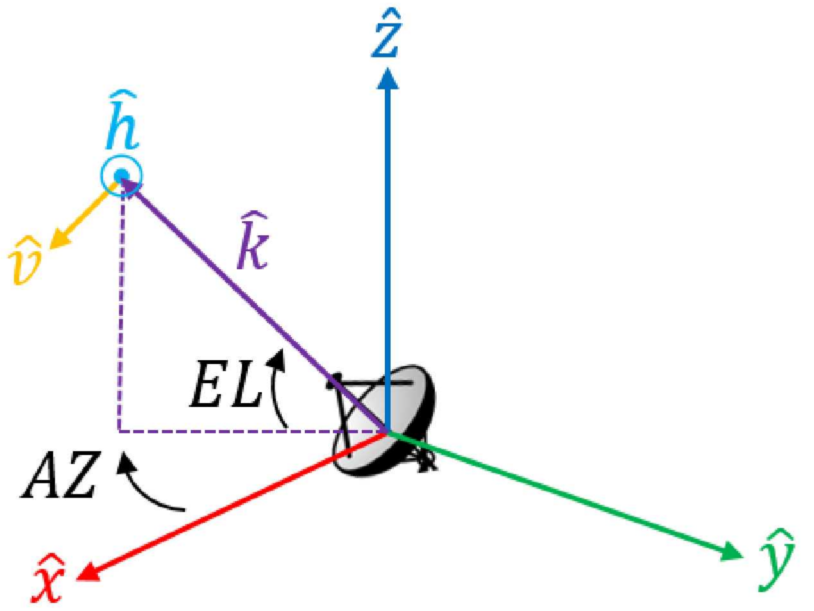


Figure 2-1. Radar coordinate frame (note that $\hat{h}$ is out of the page).

The RCS data for a target is often defined in the target's coordinate frame, which is denoted by primed variables as illustrated in Fig. 2-2. In this coordinate frame the propagation of the incident radar signal is in the $-\hat{k}'$ direction. When querying the RCS data, the incident propagation unit vector is required as an input (see Section 4.2) and must be equal to $-\hat{k}'$.

A discrepancy between the polarization vectors defined in the radar and target coordinate frames will occur when the coordinate frames are rotated relative to each other. This polarization "misalignment" can be described by an angle α, which is defined as a right hand rotation about $\hat{k}$ (or $-\hat{k}'$) from $\hat{v}'$ to $\hat{v}$ (or $\hat{h}'$ to $\hat{h}$) as illustrated in Fig. 2-3. Compensation for such a mismatch is described in the next section.

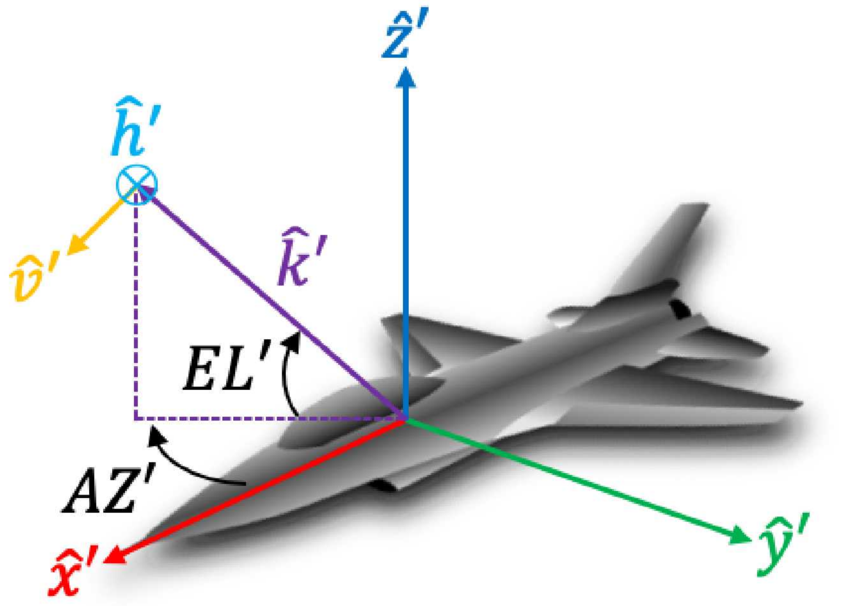


Figure 2-2. Target coordinate frame (note that $\hat{h}$ is into the page).

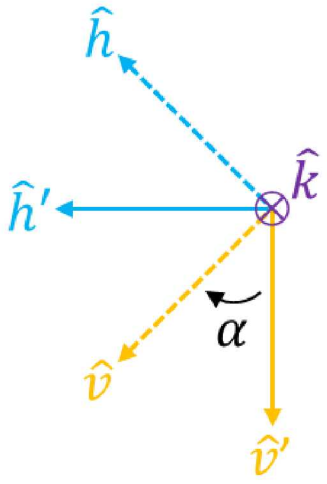


Figure 2-3. Polarization mismatch angle α (note that $\hat{k}$ is into the page).

2.2. Polarization Mismatch

The complex scattering matrix $[\mathbf{S}]$ can be obtained from the complex scattering length $[\gamma]$ (the values stored in the RCS file) [3] as

$$[\mathbf{S}]' = \begin{bmatrix} a_{v'v'} & a_{v'h'} \\ a_{h'v'} & a_{h'h'} \end{bmatrix} = (4\pi R^2)^{-1/2}[\gamma]' = \begin{bmatrix} \gamma_{v'v'} & \gamma_{v'h'} \\ \gamma_{h'v'} & \gamma_{h'h'} \end{bmatrix}. \tag{2.8}$$

Note, the primed variables indicate the target coordinate frame and R is the distance between the radar and target. The scattering matrix is used to calculate the scattered field of an object given an incident field

$$\begin{bmatrix} E^s_{v'} \\ E^s_{h'} \end{bmatrix} = [\mathbf{S}]' \begin{bmatrix} E^i_{v'} \\ E^i_{h'} \end{bmatrix}. \tag{2.9}$$

We define a rotation matrix $[\mathbf{R}]$ such that the polarization basis of the electric fields can be transformed as

$$\begin{bmatrix} E^s_{v} \\ E^s_{h} \end{bmatrix} = [\mathbf{R}] \begin{bmatrix} E^s_{v'} \\ E^s_{h'} \end{bmatrix}. \tag{2.10}$$

Where $[\mathbf{R}]$ is the passive right-hand rotation matrix

$$[\mathbf{R}] = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}. \tag{2.11}$$

We can now calculate the scattering in the radar's coordinate frame using a similarity transform [4] as

$$\begin{bmatrix} E^s_{v} \\ E^s_{h} \end{bmatrix} = [\mathbf{R}][\mathbf{S}]'[\mathbf{R}]^{-1} \begin{bmatrix} E^i_{v} \\ E^i_{h} \end{bmatrix}. \tag{2.12}$$

The complex scattering matrix may be transformed (e.g., rotated) as long as the transformation is unitary [3] since unitary transforms preserve inner products. The transformation shown in (2.12) is a unitary transform since $[\mathbf{R}]$ is unitary[4] (i.e., $[\mathbf{R}]^T = [\mathbf{R}]^H = [\mathbf{R}]^{-1}$).

The API utilizes the following transformation when querying the data in order to return RCS data defined in the radar coordinate frame:

[4]The rotation matrix $[\mathbf{R}]$ is real valued and has the property $[\mathbf{R}]^T = [\mathbf{R}]^{-1}$ hence it is orthogonal [5]. Orthogonal matrices are a special case of the more general unitary matrices, which are complex valued and have the property $[\mathbf{R}]^H = [\mathbf{R}]^{-1}$ [5]. The superscript H is used to define the conjugate transpose or Hermitian transpose.

$$[\gamma] = \begin{bmatrix} \gamma_{vv} & \gamma_{vh} \\ \gamma_{hv} & \gamma_{hh} \end{bmatrix} = [\mathbf{R}][\gamma]'[\mathbf{R}]^{-1} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \gamma_{v'v'} & \gamma_{v'h'} \\ \gamma_{h'v'} & \gamma_{h'h'} \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}. \tag{2.13}$$

The components of $[\gamma]$ are then calculated by the following equations (actually implemented in the API):

$$\gamma_{vv} = \gamma_{v'v'}\cos^2\alpha - (\gamma_{h'v'} + \gamma_{v'h'})\sin\alpha\cos\alpha + \gamma_{h'h'}\sin^2\alpha, \tag{2.14}$$

$$\gamma_{vh} = \gamma_{v'h'}\cos^2\alpha - (\gamma_{h'h'} - \gamma_{v'v'})\sin\alpha\cos\alpha - \gamma_{h'v'}\sin^2\alpha, \tag{2.15}$$

$$\gamma_{hv} = \gamma_{h'v'}\cos^2\alpha - (\gamma_{h'h'} - \gamma_{v'v'})\sin\alpha\cos\alpha - \gamma_{v'h'}\sin^2\alpha, \tag{2.16}$$

and

$$\gamma_{hh} = \gamma_{h'h'}\cos^2\alpha + (\gamma_{h'v'} + \gamma_{v'h'})\sin\alpha\cos\alpha + \gamma_{v'v'}\sin^2\alpha. \tag{2.17}$$

RCS values σ are then computed by $|\gamma|^2$.

2.3. Circular Polarization

The RCS data can also be referenced to a circularly polarized basis[5]. If the $\hat{h}$ ($\hat{\phi}$) component is delayed 90° from the $\hat{v}$ ($\hat{\theta}$) component, the electric field will rotate with a right handed sense around $\hat{k}$ [2]. Similarly, a 90° advance will produce a left hand rotation. The conversion between linear and circular polarized basis functions is given by

$$\begin{bmatrix} E_R^i \\ E_L^i \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -j \\ 1 & j \end{bmatrix} \begin{bmatrix} E_v^i \\ E_h^i \end{bmatrix}, \tag{2.18}$$

and

$$\begin{bmatrix} E_v^i \\ E_h^i \end{bmatrix} = \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -j \\ 1 & j \end{bmatrix} \right)^{-1} \begin{bmatrix} E_R^i \\ E_L^i \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -j & j \end{bmatrix} \begin{bmatrix} E_R^i \\ E_L^i \end{bmatrix}. \tag{2.19}$$

When converting linear scattering to circular scattering a conjugation of the matrix must be performed

[5]Note that the CP basis is defined in the radar coordinate frame.

$$\begin{bmatrix} E_R^s \\ E_L^s \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -j \\ 1 & j \end{bmatrix}^* \begin{bmatrix} E_v^s \\ E_h^s \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & j \\ 1 & -j \end{bmatrix} \begin{bmatrix} E_v^s \\ E_h^s \end{bmatrix}. \tag{2.20}$$

This is done in order to account for the change in rotation sense (RHCP is now referenced to the $-\hat{k}$ direction) [1] [2]. Given a linearly polarized scattering matrix, (2.3) can then be rewritten for circular polarization as

$$\begin{bmatrix} E_R^s \\ E_L^s \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & j \\ 1 & -j \end{bmatrix} \begin{bmatrix} a_{vv} & a_{vh} \\ a_{hv} & a_{hh} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -j & j \end{bmatrix} \begin{bmatrix} E_R^i \\ E_L^i \end{bmatrix}. \tag{2.21}$$

It follows that the linear polarized CSL matrix (values actually stored in the RCS data files) can be converted to a circularly polarized basis by

$$\begin{bmatrix} \gamma_{RR} & \gamma_{RL} \\ \gamma_{LR} & \gamma_{LL} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & j \\ 1 & -j \end{bmatrix} \begin{bmatrix} \gamma_{vv} & \gamma_{vh} \\ \gamma_{hv} & \gamma_{hh} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -j & j \end{bmatrix}. \tag{2.22}$$

This transformation (Eg. 2.22) is implemented by the following equations in the API when CP is selected during a query:

$$\gamma_{RR} = \frac{1}{2}((\gamma_{vv} - \gamma_{hh}) + j(\gamma_{vh} + \gamma_{hv})), \tag{2.23}$$

$$\gamma_{RL} = \frac{1}{2}((\gamma_{vv} + \gamma_{hh}) - j(\gamma_{vh} - \gamma_{hv})), \tag{2.24}$$

$$\gamma_{LR} = \frac{1}{2}((\gamma_{vv} + \gamma_{hh}) + j(\gamma_{vh} - \gamma_{hv})), \tag{2.25}$$

and

$$\gamma_{LL} = \frac{1}{2}((\gamma_{vv} - \gamma_{hh}) - j(\gamma_{vh} + \gamma_{hv})). \tag{2.26}$$

3. FILE FORMAT

The dynamic RCS datasets have the potential to be quite large (multiple GB). This is due to the fact that the datasets contain full polarimetric complex RCS data for multiple aspect angles, frequencies, and time steps (as discussed in the previous sections). The potential size of the datasets makes simple text parsing (loading ASCII files into memory and searching for the required data) undesirable. A fast and flexible method for storing and querying the RCS data was desired and therefore SQLite [6] was chosen as the base file format.

SQLite was chosen because it is light weight (does not require large computational resources), portable, serverless (it is an API and does not require a separate database server process) and can handle large datasets efficiently. It utilizes B-Tree data structures [7] and can therefore perform queries in $O(\log n)$ time while loading and parsing text files would at best be $O(n)$ and require more memory.

The SQLite data file stores the RCS data as complex scattering length (CSL) in the target coordinate frame using the vertical and horizontal ($\hat{v}$ and $\hat{h}$) polarization basis. The data is organized into four tables: time, frequency, aspect angle, and RCS. Each table is discussed in more detail below.

3.1. Time Table

The time table has unique identifier, start time, and stop time columns. The start and stop time (units of seconds) indicate the time period in the simulation where the corresponding RCS data is applicable. An example of the time table is shown in Fig. 3-1 with the SQL statement used to create the table shown in Listing 3.1.

3.2. Frequency Table

The frequency table has unique identifier and frequency (GHz) columns. The table stores all the unique frequencies for which the RCS data is defined. An example frequency table is shown in Fig. 3-2. The SQL statement that creates the table is shown in Listing 3.2.

```sql
CREATE TABLE t_table (uid INTEGER PRIMARY KEY, start REAL NOT NULL,
    end REAL NOT NULL);
```

Code Listing 3.1. Time table SQL `CREATE` statement.

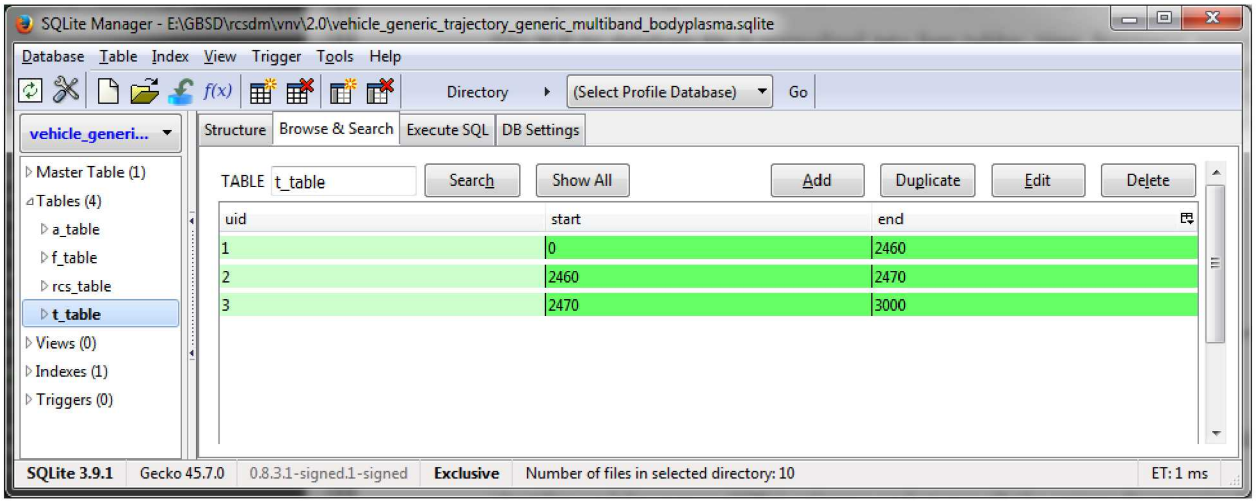


TABLE t_table

uid	start	end
1	0	2460
2	2460	2470
3	2470	3000

Figure 3-1. Time table example.

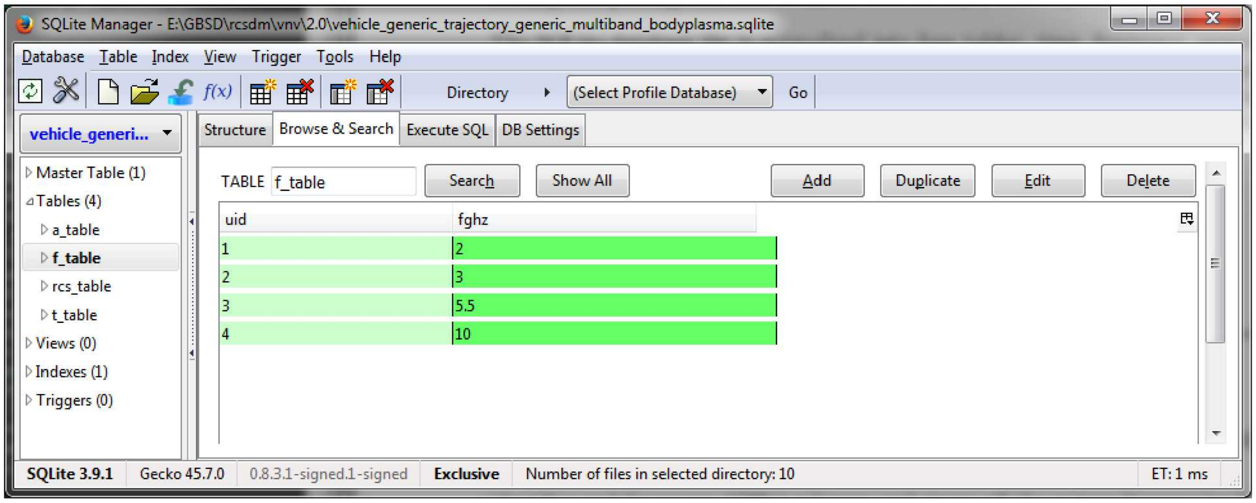


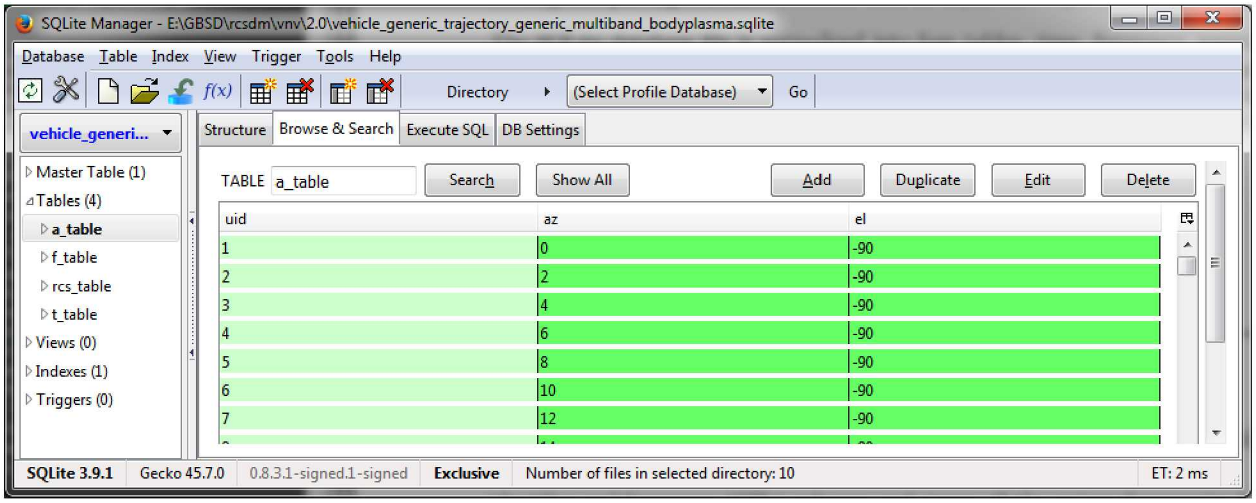
TABLE f_table

uid	fghz
1	2
2	3
3	5.5
4	10

Figure 3-2. Frequency table example.

```
CREATE TABLE f_table (uid INTEGER PRIMARY KEY, fghz REAL NOT NULL);
```

Code Listing 3.2. Frequency table SQL `CREATE` statement.



uid	az	el
1	0	-90
2	2	-90
3	4	-90
4	6	-90
5	8	-90
6	10	-90
7	12	-90

Figure 3-3. Aspect angle table example.

```
CREATE TABLE a_table (uid INTEGER PRIMARY KEY, az REAL NOT NULL,
    el REAL NOT NULL);
```

Code Listing 3.3. Frequency table SQL CREATE statement.

3.3. Aspect Angle Table

The aspect angle table has unique identifier, azimuth (deg), and elevation (deg) columns. The data is stored in the target's coordinate frame as illustrated by Fig. 2-2. A single unique identifier is used to represent a unique aspect angle defined by both azimuth and elevation. An example of the aspect angle table is shown in Fig. 3-3. Listing 3.3 displays the SQL statement used to create the table.

3.4. RCS Table

The RCS table has columns for unique identifier, time identifier, aspect angle identifier, frequency identifier, and both real and imaginary components of the complex scattering lengths (CSL) for all linear polarizations (VV, VH, HV, and HH)[1]. The unique identifiers from the time, aspect angle, and frequency tables are used to index the RCS table which may be very large. In order to decrease the query time, indexes are built on the RCS and aspect angle tables. An example of the RCS data table is shown in Fig. 3-4. The SQL statement used for the table's creation is shown in Listing 3.4.

[1]Note that the Receive-Transmit notation is used when denoting RCS data polarization e.g., VH indicates transmit H and receive V.

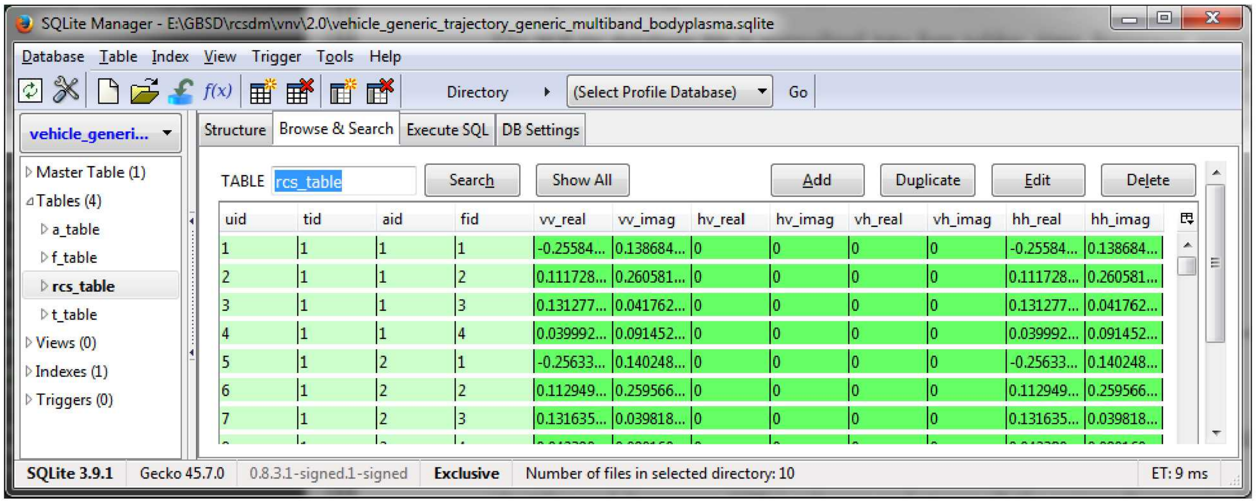

SQLite Manager - E:\GBSD\rcsdm\vnv\2.0\vehicle_generic_trajectory_generic_multiband_bodyplasma.sqlite

TABLE rcs_table

uid	tid	aid	fid	vv_real	vv_imag	hv_real	hv_imag	vh_real	vh_imag	hh_real	hh_imag
1	1	1	1	-0.25584...	0.138684...	0	0	0	0	-0.25584...	0.138684...
2	1	1	2	0.111728...	0.260581...	0	0	0	0	0.111728...	0.260581...
3	1	1	3	0.131277...	0.041762...	0	0	0	0	0.131277...	0.041762...
4	1	1	4	0.039992...	0.091452...	0	0	0	0	0.039992...	0.091452...
5	1	2	1	-0.25633...	0.140248...	0	0	0	0	-0.25633...	0.140248...
6	1	2	2	0.112949...	0.259566...	0	0	0	0	0.112949...	0.259566...
7	1	2	3	0.131635...	0.039818...	0	0	0	0	0.131635...	0.039818...

SQLite 3.9.1 | Gecko 45.7.0 | 0.8.3.1-signed.1-signed | Exclusive | Number of files in selected directory: 10 | ET: 9 ms

Figure 3-4. RCS data (complex scatting length) table example.

```
CREATE TABLE rcs_table (uid INTEGER PRIMARY KEY, tid INTEGER, aid INTEGER,
    fid INTEGER, vv_real REAL NOT NULL, vv_imag REAL NOT NULL,
    hv_real REAL NOT NULL, hv_imag REAL NOT NULL, vh_real REAL NOT NULL,
    vh_imag REAL NOT NULL, hh_real REAL NOT NULL, hh_imag REAL NOT NULL);
```

Code Listing 3.4. RCS table SQL `CREATE` statement.

4. API

An Application Programing Interface (API) was developed to provide M&S applications the ability to interface with the RCS data. C++ was chosen as the implementation language for its speed, ability to interface with the SQLite C API, and the fact that many M&S tools are written in C++ (e.g., AFSIM [8]). The developed API additionally provides capability for creating the SQLite RCS data files from the datasets generated by electromagnetic analysis tools.

The source code for the API, a Command Line Interface (CLI) application, as well as Verification & Validation (V&V) scripts (written in Python) are maintained in a Git repository located on the internal Sandia network[1].

4.1. Creating the RCS Data Files

The RCS data file creation is handled by the class `DatabaseCreator` defined in the files `database_creator.cc/.h`. The class requires only two inputs for creating the RCS data file: a path to the directory containing the data files to compile (*.field files[2]) and the name of the configuration file. Example usage is given in Listing 4.1. Note that all code in the API is defined in the `rcsdm` namespace.

```cpp
using namespace rcsdm;
std::string data_dir = "/home/user/data";
std::string config_file = "config_file.json";
auto dbcreator = DatabaseCreator(data_dir, config_file);
dbcreator.Build();
```

Code Listing 4.1. Example usage of the `DatabaseCreator` class.

The configuration file specifies each RCS dataset, i.e., *.field file, with its applicable time duration. Note that the configuration file is expected to be in the same directory as the data files. The configuration file also specifies the name of the conglomerate dataset that will be used to name the resulting SQLite data file. The data in the configuration file is formatted in the widely used Javascript Object Notation (JSON) file format for easy parsing. An example configuration file is shown in Listing 4.2.

For more details on the usage of the `DatabaseCreator` class, refer to the code documentation in `database_creator.h` (see source code).

[1]`https://gitlab.sandia.gov/applied-electromagnetics/rcs-data-manager`

[2]Several RCS simulation tools output data in the *.field file format. This format is desired because the data contains phase information (CSL data values). However, if a simulation tool does not utilize the field file format, a custom script may be developed to convert the files.

```
{
    "datasetname" : "dynamic_rcs_target"
    "fielddatasets" : [
        {
            "filename" : "initial_rcs.field",
            "starttime" : 0,
            "endtime" : 1200
        },
        {
            "filename" : "modified_rcs_01.field",
            "starttime" : 1200,
            "endtime" : 1800
        },
        {
        "filename" : "modified_rcs_01.field",
        "starttime" : 1800,
        "endtime" : 2400
        }
    ]
}
```

Code Listing 4.2. Example of a configuration file (JSON format).

4.2. Querying the RCS Data Files

Querying the RCS database is accomplished through the `DatabaseInterface` class defined in the files `database_interface.cc/.h`. Instantiation of the class requires a path to the database file given as a `std::string`. There are multiple options for querying the database; two options are demonstrated in Listing 4.3. The query methods require a time step in seconds, the observation aspect angle specified as either azimuth and elevation angles (both in degrees) or as the unit vector $-\hat{k}'$ (shown in Fig. 2-2), the frequency of the radar (GHz), the desired RCS polarization, and, if applicable, the polarization mismatch angle (deg). Details on the geometry are given in Section 2.1. Additional query methods include the ability to query complex (CSL) data as well as wideband (multiple frequencies) data queries.

The query functions shown in Listing 4.3 return the RCS (σ) in units of dBsm; however, the RCS data file stores CSL (γ) in units of meters. The CSL values queried from the file are converted to RCS in dBsm via

$$\sigma_{dBsm} = 20 * \log_{10} |\gamma_m|. \tag{4.1}$$

Further details (including additional query methods and options) are provided in the documentation within the `database_interface.h` header file (see the source code).

```cpp
using namespace rcsdm;
std::string path = "/home/user/data/dynamic_rcs_target.sqlite";
auto dbinterface = DatabaseInterface(path);

double time = 2000;      // Time step (s)
double freq = 10;        // Radar frequency (GHz)
double az = 0;           // Azimuth aspect angle (deg)
double el = -35;         // Elevation aspect angle (deg)
double pol_angle = 10;   // Polarization mismatch angle (deg)
polarization_t pol = VV; // RCS polarization type

// Use AZ and EL
double rcs = dbinterface.QueryRcsDbsm(time, freq, az, el, pol, pol_angle);

double xu = 0.8192; // x component of -k' unit vector
double yu = 0.0;    // y component of -k' unit vector
double zu = 0.7877; // z component of -k' unit vector

// Use unit vector -k'
rcs = dbinterface.QueryRcsDbsm(time, freq, xu, yu, zu, pol, pol_angle);
```

Code Listing 4.3. Example usage of the `DatabaseInterface` class.

```
rcsdm build --input /home/user/data/config_file.json
```

Code Listing 4.4. Example usage of the CLI to create a database.

4.3. Command Line Interface

A simple CLI application was also developed for testing the API as well managing an RCS dataset. The CLI is most useful for database creation as it allows the creation of datasets to be scripted (in a shell script for instance) as shown in Listing 4.4.

More information can be found by running `rcsdm -h`.

5. SUMMARY

This report describes both a file format and the corresponding API that were developed for the purpose of integrating large dynamic RCS datasets with M&S applications. The file format is based on SQLite which provides a fast and flexible method for storing and querying the RCS data. The database is normalized and has appropriate indexes built in, which results in a negligible performance hit when querying the data in real time (< 0.01 sec per query[1]). The API is written in C++ and provides methods for both building and querying the RCS data. Functionality is also included to modify the polarization basis in order to compensate for target and radar coordinate frame offsets or convert from linear to circular polarization. Features and future work are summarized below.

5.1. Summary of Features

- Combine multiple RCS datasets (*.field files) containing values for multiple frequencies, aspect angles, and all linear polarization combinations into a singe file where each dataset can be indexed based on time.
- Query data quickly with negligible impact on overall simulation time. Queries are based on simulation time step, frequency, aspect angle, and polarization.
- Queries can return either CSL values (which include phase information) or RCS magnitude (dBsm). Single frequency or wideband queries may be made.
- Queries can be specified to modify the polarization basis as needed e.g., rotating linear basis or converting to circular basis.
- Extensive debugging capability is built into each object (`DatabaseCreator` and `DatabaseInterface`) via status variables that can be checked.
- Nearest neighbor interpolation is used when query parameters do not match those in the dataset. The range of the interpolation may be set in `DatabaseInterface` object variables.

5.2. Future Work

- Expand the time table to allow for each dataset to have multiple start and stop times. This would allow data sets within the file to be applied to multiple time periods.

[1]Tested on a 650 MB datafile using a 2.9 GHz processor.

- Wideband processing capabilities e.g., utilize FFT to produce time/range profiles.
- Add support for bistatic RCS. This could potentially lead to significantly larger datasets.
- Add flexibility to the dataset creation e.g., process *.field files with dissimilar frequencies, aspect angles, etc. to allow the unification of data produced by different simulation runs.

REFERENCES

[1] M. A. Richards, J. A. Scheer, and W. A. Holm, *Principles of Modern Radar: Basic Principles*. Edison, NJ: SciTech, 2010.

[2] C. A. Balanis, *Advanced engineering electromagnetics*. John Wiley & Sons, 2nd ed., 2012.

[3] G. T. Ruck, *Radar Cross Section Handbook*. New York, USA: Plenum Press, 1970.

[4] E. W. Weisstein, "Similarity Transformation." `http://mathworld.wolfram.com/SimilarityTransformation.html`. Accessed: 2017-03-13.

[5] P. Hlawiczka, *Matrix Algebra for Electronic Engineers*. New York, USA: Hayden Publishing, 1965.

[6] "SQLite." `http://sqlite.org/`. Accessed: 2017-09-12.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. Cambridge, USA: MIT press, third ed., 2009.

[8] P. D. Clive, J. A. Johnson, M. J. Moss, J. M. Zeh, B. M. Birkmire, and D. D. Hodson, "Advanced framework for simulation, integration and modeling (afsim)," in *Proceedings of the 13th International Conference on Scientific Computing*, pp. 73–77, 2015.

DISTRIBUTION

Email—Internal (encrypt for OUO)

Name	Org.	Sandia Email Address
Jay Barton	05345	jbarto@sandia.gov
Dylan A. Crocker	06773	dacrock@sandia.gov
John R. Dickinson	06773	jrdicki@sandia.gov
Thomas E. Roth	05345	teroth@sandia.gov
Ann M. Raynal	05344	amrayna@sandia.gov
Eric A. Shields	06773	eashiel@sandia.gov
Joshua Speciale	02665	jspecia@sandia.gov
R. Derek West	05346	rdwest@sandia.gov
Mathew W. Young	05345	mwyoung@sandia.gov
Technical Library	01177	libref@sandia.gov

Sandia National Laboratories