

Simulation of Radar Profiles for Satellites

Daniel Topa
daniel.topa@hii.com

*Huntington Ingalls Industries
Mission Technologies*

November 25, 2024

Abstract

A brief survey of characterizing the three dimensional radar cross section of satellites. The process of finding the optimal Fourier expression for each band is explored and different success measures are presented.

Contents

1 Writ Large	1
1.1 Toy Satellite Model	4
2 Mathematical Precís	4
2.1 Models of Radar Cross Section	4
2.2 Models of Increasing Fidelity	4
2.3 Running the Code	4
2.4 Radar	8
2.5 Process	8
3 Overview: Modeling Radar Cross Section	8
3.1 Radar	8
3.2 Process	9
4 Computing the Radar Cross Section	9
4.1 Maxwell's Equations	9
4.2 Method of Moments	10
4.3 Mercury MoM Example	10
5 Fourier Expansion	10
5.1 Why Fourier?	10
5.2 Orthogonality	11
5.3 Projection	12
5.4 Discrete Topology	12
5.5 Special Case: The Mean	12

5.5.1	The Method of Least Squares	12
5.5.2	Continuum to discrete	13
5.6	Linear Independence	13
5.7	Quality of Fit	13
6	File Types	13
6.1	Geometry Files *.obj	13
6.2	New Efforts	16
7	Running Mercury Method of Moments	16
7.1	Inputs	16
7.2	Outputs	18
8	Additional Information	20
8.1	YouTube Videos	20
8.2	Further Reading	20
A	Mercury Method of Moments: Data Formats	21
A.1	Numeric Results	21
B	Mercury Method of Moments: Software Toolkit	21
B.1	<code>rcsharvester.f08</code>	21
B.1.1	Class Electric Fields: <code>m-class-electric-fields.f08</code>	21
B.1.2	<code>m-class-electric-fields.f08</code>	21
B.1.3	Class Data File: <code>m-class-data-file.f08</code>	23
C	Mercury Method of Moments: Distribution and Rights	28
C.1	Distribution Letter for Software	28
C.2	Copyright Statement by the Author	30
C.3	Legal Statement	30
C.4	Obtaining Software and Documentation	32
C.5	Distribution Contents	32
C.5.1	Executables	32
C.5.2	Documentation	32
D	Using Python to Create Spreadsheets	32
D.1	Inputs	33
D.1.1	Main	33
D.1.2	Class Test Object	36
D.1.3	Excel Details	39
D.2	Outputs	43

1 Writ Large

Radar is an invaluable tool for space domain awareness which can characterize not only location, but also orientation of satellites in space. It can answer where antennas are pointed and how a space craft is maneuvering. The open question is one of interpreting the radar return section. The key value is the amount of energy returned by reflection, and this is the essence of the radar cross section (res).

Figure 1 shows a sample radar cross section measurement for a vintage A-26 Invader. The strength of the return signal varies over several orders of magnitude depending upon the viewing

angle.

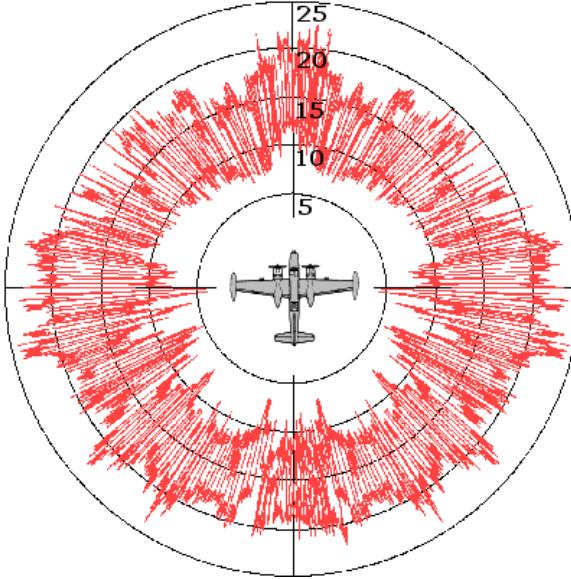


Figure 1: A radar cross section measurement for the A-26 Invader taken from the Wikipedia article on Radar Cross Section show energy change where 0 db represents 1 milliwatt return energy.

A simplified aircraft model in figure 2 presents requisite variation without over-complications and makes the point that even a stealthy aircraft can present large cross sections at proper view angles as seen in table 2.

Look angle

The purpose of this analysis is to start with a CAD model of a structure and simulate the radar cross section, a sample show in figure 3.

render the radar cross section into a form like

$$\begin{aligned}\sigma_3(\theta) = & 35.237 \pm 0.012 + (1.675 \pm 0.018) \cos \theta + (-3.434 \pm 0.018) \cos 2\theta + (-0.866 \pm 0.018) \cos 3\theta \\ & + (5.386 \pm 0.018) \cos 4\theta + (-1.280 \pm 0.018) \cos 5\theta + (1.379 \pm 0.018) \cos 6\theta + (-0.675 \pm 0.018) \cos 7\theta\end{aligned}$$

where θ is an azimuthal angle with $\theta = 0$ representing a nose-one view and $\pm\pi$ representing a tail-on view

1.1 Toy Satellite Model

A toy satellite model is much easier to start with for developing a data analysis can reduction chain. The model below (figure /reftab:toy-sat) features a spherical body and readily distinguishable solar panels.

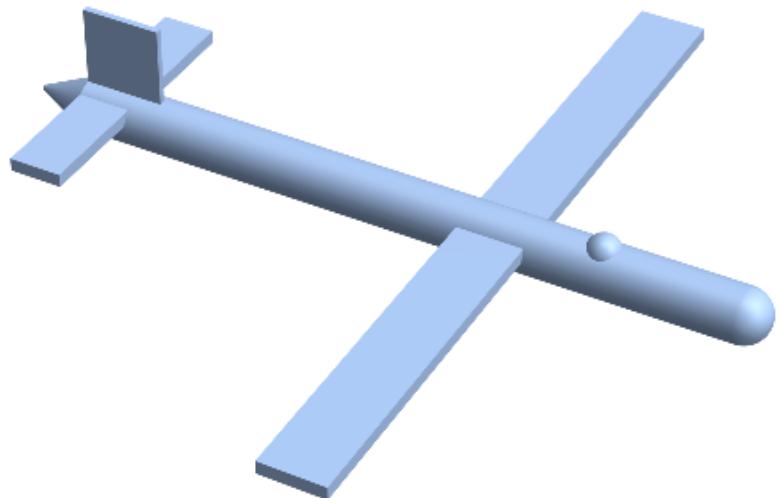


Figure 2: Toy model for aircraft. Although simplistically rendered, this representation is adequate for over the horizon radars.

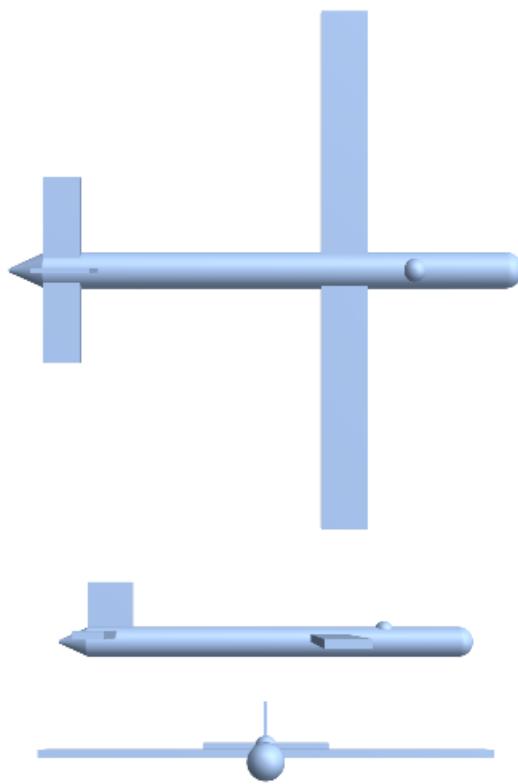


Table 1: Different aircraft views present very different cross-sectional areas.

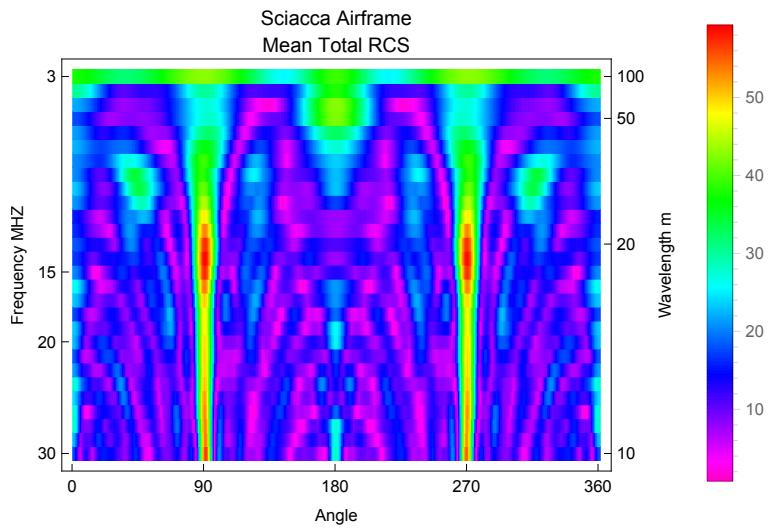


Figure 3: Output for a radar backscatter simulation for radar frequencies from 3 – 30 MHz, corresponding to wavelengths of 10 – 100 m.

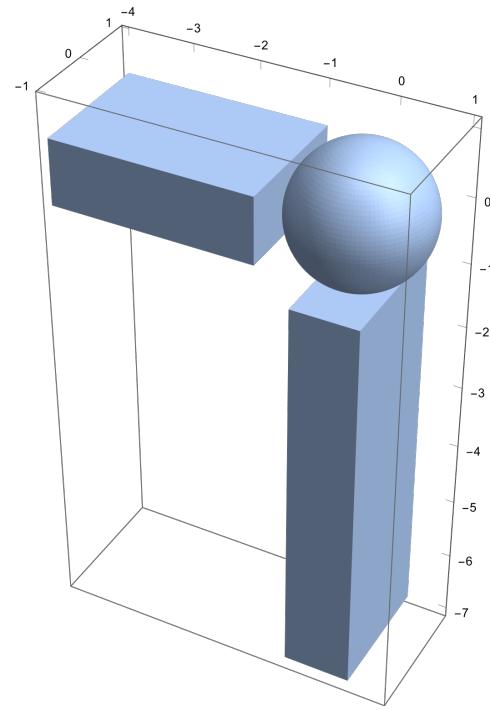
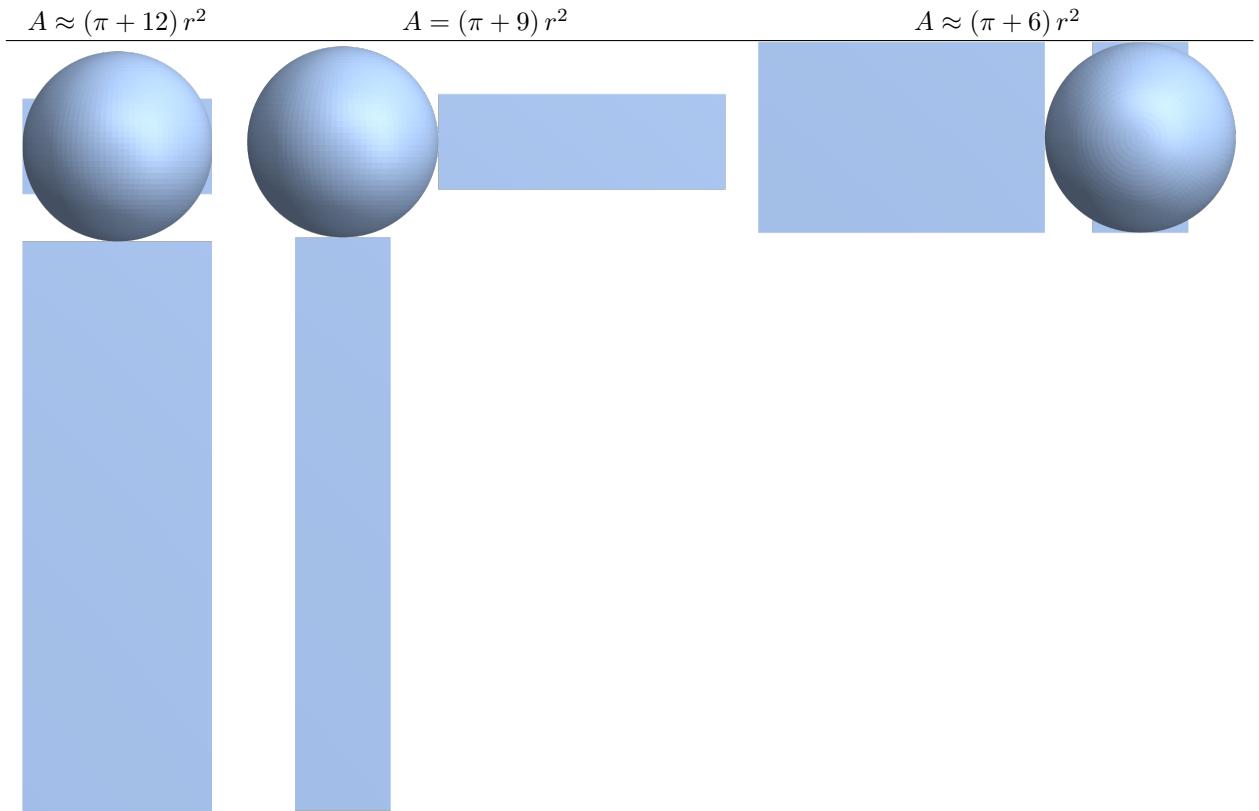
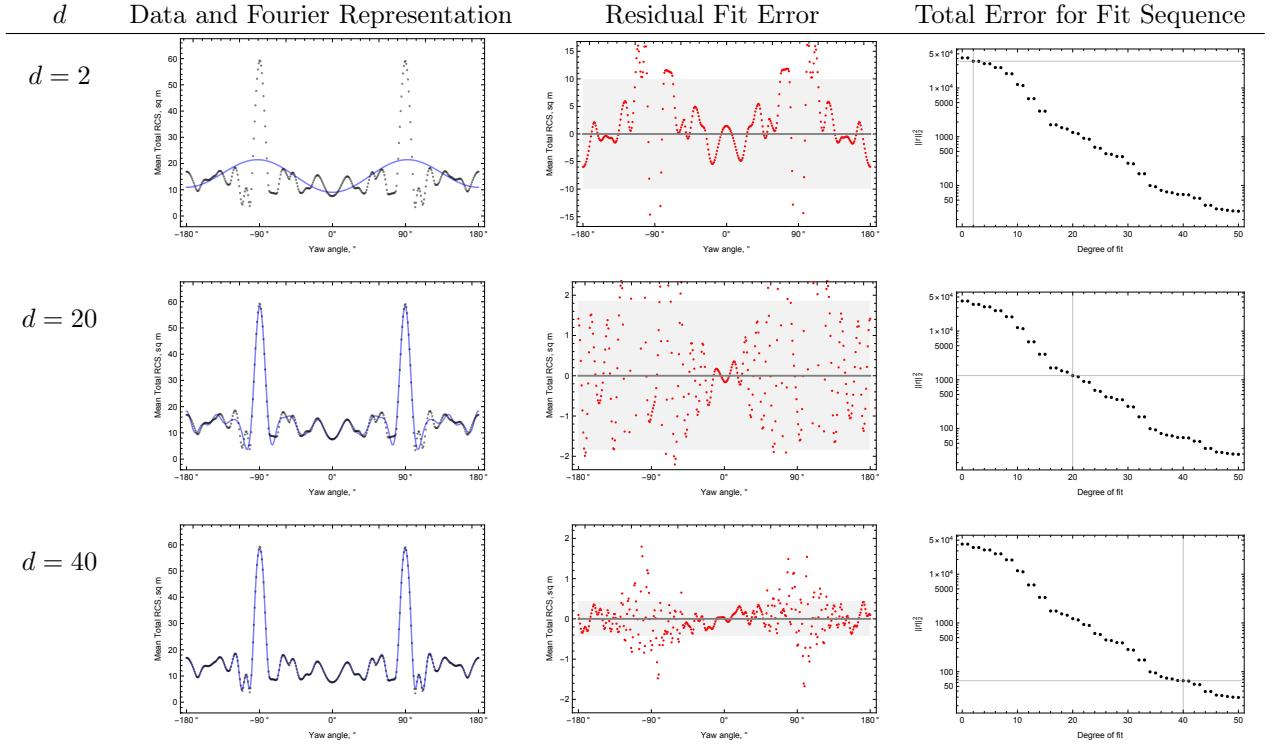


Figure 4: A toy satellite model designed to present a spherical backscatter and resonances at 6, 3, 2, and 1 meter wavelengths.

Table 2: Contrasting views of a toy satellite present very different cross sectional areas, A .





2 Mathematical Precís

2.1 Models of Radar Cross Section

$$\sigma_\nu(\alpha) \approx \frac{a_0}{2} + \sum_{k=1}^{\frac{d}{2}} a_k \cos k\alpha + b_k \sin k\alpha \quad (1)$$

Amplitudes and Errors for $\nu = 3$ MHz and $d = 7$:

$$\begin{aligned} \sigma_3(\theta) = & a_0 + a_1 \cos \theta + a_2 \cos 2\theta + a_3 \cos 3\theta \\ & + a_4 \cos 4\theta + a_5 \cos 5\theta + a_6 \cos 6\theta + a_7 \cos 7\theta \end{aligned}$$

$$\begin{aligned} \sigma_3(\theta) = & 35.237 \pm 0.012 + (1.675 \pm 0.018) \cos \theta + (-3.434 \pm 0.018) \cos 2\theta + (-0.866 \pm 0.018) \cos 3\theta \\ & + (5.386 \pm 0.018) \cos 4\theta + (-1.280 \pm 0.018) \cos 5\theta + (1.379 \pm 0.018) \cos 6\theta + (-0.675 \pm 0.018) \cos 7\theta \end{aligned}$$

2.2 Models of Increasing Fidelity

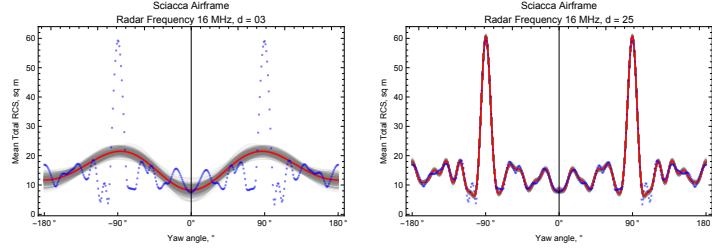
2.3 Running the Code

`./MMoM_4.1.12 sample.geo`

2.4 Radar

[14] [14] Working with CAF files, producing output, compressing data. [15] Topa [15]

Table 3: Seeing the uncertainties in the fit parameters.



2.5 Process

- | | |
|--|------------------------------|
| 1 Create CAD model | CAD software |
| 2 Convert CAD to *.obj | CAD software |
| 3 Convert *.obj to *.facet | Mathematica, Fortran |
| 4 Input properties to <code>materials.lib</code> | VIM |
| 5 Set radar frequencies | VIM |
| 6 Simulate radar irradiation | Mercury MoM |
| 7 Harvest reflection values from output | Mathematica, Fortran, Python |
| 8 Describe RCS as a series of amplitudes | Not written |

Table 4: Start with a CAD model and construct a Radar Cross Section model

3 Overview: Modeling Radar Cross Section

3.1 Radar

Wave speed equation

$$\lambda\nu = c \quad (2)$$

- (A) Build a CAD model of the satellite (*.cad)
- (B) Seal the CAD mesh
- (C) Create geometry file (*.geo)
- (D) Irradiate object with Mercury MoM
- (E) Harvest backscatter
- (F) Construct RCS
- (G) Resolve RCS measurements into spherical harmonics

3.2 Process

- (A) Build a CAD model of the satellite (*.cad)
- (B) Seal the CAD mesh
- (C) Create geometry file (*.geo)
- (D) Irradiate object with Mercury MoM
- (E) Harvest backscatter
- (F) Construct RCS
- (G) Resolve RCS measurements into spherical harmonics

band	ν	λ
HF	3 – 30 MHz	10 – 1 m
UHF	30 – 300 MHz	0.1 – 0.01 m
VHF	300 – 1000 MHz	0.01 – 0.03 m
L	1 – 2 GHz	30 – 15 mm
S	2 – 4 GHz	15 – 7.5 mm
C	4 – 8 GHz	7.5 – 3.7 mm
X	8 – 12 GHz	3.7 – 2.5 mm
Ku	12 – 18 GHz	2.5 – 1.7 mm
K	18 – 27 GHz	1.7 – 1.1 mm
Ka	27 – 40 GHz	1.1 – 0.75 mm
V	40 – 75 GHz	0.75 – 0.4 mm
W	75 – 110 GHz	0.4 – 0.27 mm
mm	110 – 300 GHz	0.27 – 0.1 mm

Table 5: IEEE Standard Designations for Radar Bands ([2]).

4 Computing the Radar Cross Section

The computation of the radar cross section is based on a simple scenario guided by Maxwell's equations. A radar illuminates the target with radar energy. The target, acting like an antenna, reradiates the energy and illuminates the radar receiver. The radar cross section is measurement of the difference between energy absorbed and energy radiated.

One form of the radar cross section equation is

$$\sigma = 4\pi r^2 \frac{S_r}{S_t} \quad (3)$$

where r is the distance to the target, S_t is the energy *intercepted* by the target and S_r is the energy *radiated* by the target. The result has units of area and can be compared to an ideally reflecting sphere.

The areal measure σ is computed through numerical simulation by the program Mercury MoM.

4.1 Maxwell's Equations

The mathematical foundation for the radar cross section computation is the set of Maxwell's equations. Start with two conservative fields on a simply connected domain, the electric field $\mathbf{E}: \mathbb{R}^3 \mapsto \mathbb{R}^3$, and the magnetic field $\mathbf{B}: \mathbb{R}^3 \mapsto \mathbb{R}^3$. We can relax the requirements for the electric current $\mathbf{J}: \mathbb{R}^3 \mapsto \mathbb{R}^3$. The differential form of Maxwell's equations in SI units is

$$\begin{aligned} \nabla \cdot \mathbf{E} &= \frac{\rho}{\epsilon_0} & \nabla \times \mathbf{E} &= -\partial_t \mathbf{B} \\ \nabla \cdot \mathbf{B} &= 0 & \nabla \times \mathbf{B} &= \mu_0 (\mathbf{J} + \epsilon_0 \partial_t \mathbf{E}) \end{aligned} \quad (4)$$

The fundamental physical constant in SI units are given in table 6.

4.2 Method of Moments

There is a rich mathematical toolkit for the solution of Maxwell's equations. Mercury MoM uses the method of moments (MoM), also known as the boundary element method (BEM).

ϵ_0	vacuum electric permittivity	$8.854 \times 10^{-12} \text{ F m}^{-1}$
μ_0	vacuum magnetic permeability	$1.257 \times 10^{-7} \text{ N A}^{-2}$

Table 6: Constants used in Maxwell's equations

Maxwell's equations in integral form

$$\begin{aligned}
\nabla \cdot \mathbf{E} &= \frac{\rho}{\epsilon_0} \quad \Rightarrow \quad \oint_{d\Omega} \mathbf{E} \cdot d\mathbf{S} = \int_{\Omega} \rho dV \\
\nabla \times \mathbf{E} &= -\partial_t \mathbf{B} \quad \Rightarrow \quad \oint_{d\Omega} \mathbf{B} \cdot d\mathbf{S} = 0 \\
\nabla \cdot \mathbf{B} &= 0 \quad \Rightarrow \quad \oint_{d\Sigma} \mathbf{E} \cdot d\mathbf{l} = -\frac{d}{dt} \mathbf{B} \cdot d\mathbf{S} \\
\nabla \times \mathbf{B} &= \mu_0 (\mathbf{J} + \epsilon_0 \partial_t \mathbf{E}) \quad \Rightarrow \quad \oint_{d\Sigma} \mathbf{B} \cdot d\mathbf{l} = \mu_0 \int_{\Sigma} \left(\mathbf{J} + \epsilon_0 \frac{d}{dt} \mathbf{E} \right) \cdot d\mathbf{S}
\end{aligned} \tag{5}$$

The method of moments creates dense matrices which implies quadratic growth in storage and computation time as the problem scales up.

4.3 Mercury MoM Example

The power of Mercury MoM is the ability to simultaneously solve equations with millions of unknowns using a patented low rank approximation method for the linear system.

Internally, the Mercury MoM package resolves the signal emitted from the target into two complex electric fields: θ for the vertical component and ϕ for the horizontal component. As these fields are complex, they each have orthogonal real and imaginary components.

The prescription for the mean total radar cross section is taken from the Sciacca report as the average of the total field energy

$$\langle \sigma_T \rangle = \frac{1}{2} (\theta^* \theta + \theta^* \phi + \phi^* \theta + \phi^* \phi), \tag{6}$$

and is assigned areal units of squared meters m^2 .

5 Fourier Expansion

5.1 Why Fourier?

What is the best way to approximate information in the cross sections curves shown in figure ??? Given that the data is periodic over the unit circle,

$$\sigma_\nu(\alpha, \beta) = \sigma_\nu(\alpha + 2\pi, \beta) \tag{7}$$

an obvious choice is the Fourier series. For example, a d th order approximation can be written as

$$\sigma_\nu(\alpha, \beta_0 = \frac{\pi}{12}) \approx \frac{a_0}{2} + \sum_{k=1}^d a_k \cos k\alpha + b_k \sin k\alpha. \tag{8}$$

What does this series represent? It represents a sequence of oscillations about the mean value.

There are three powerful theoretical reasons which support this choice. The first is the convergence of the series. The Weierstrass Approximation Theorem states that the polynomial

approximation of smooth curves converges *uniformly*. That is, we can define a maximum error $\epsilon > 0$ and we are guaranteed the existence of $N \in \mathbb{Z}^+$ such that

$$\left\| \sigma_\nu(\alpha, \beta_0) - \frac{a_0}{2} - \sum_{k=1}^N a_k \cos k\alpha + b_k \sin k\alpha \right\|_\infty \leq \epsilon. \quad (9)$$

The ability to choice a maximum error is both remarkable and useful. However, the application to trigonometric polynomials, while valid, is not immediate and left for another document.

The second reason is the Riesz–Fischer theorem. While often cited as evidence that the Lebesgue space L^2 is complete, it also establishes that an infinite series whose terms converge quadratically represents an L^2 function. And so the amplitudes, the a and b values, will eventually converge at least quadratically.

The final reason is the often-overlooked fact that of all the orthogonal polynomials, only the trigonometric polynomials are orthogonal in both the continuous topology of L^2 and and the discrete orthogonality of l^2 . Having an orthogonal basis decouples the problem and can be solved mode-by-mode, obviating the need to solve a large linear system.

5.2 Orthogonality

Given non-zero integers m and n , the expression of orthogonality in L^2 can be written as

$$\begin{aligned} \int_{-\pi}^{\pi} \sin(mx) \sin(nx) dx &= \pi \delta_n^m \\ \int_{-\pi}^{\pi} \cos(mx) \cos(nx) dx &= \pi \delta_n^m \\ \int_{-\pi}^{\pi} \cos(mx) \sin(nx) dx &= 0 \\ \int_{-\pi}^{\pi} \sin(mx) dx &= 0 \\ \int_{-\pi}^{\pi} \cos(mx) dx &= 0 \end{aligned} \quad (10)$$

where δ_n^m is the Kronecker delta tensor. To complete the toolkit to solve the linear system, recall that

$$\int_{-\pi}^{\pi} 1 dx = 2\pi. \quad (11)$$

The linear system decouples and amplitudes for each mode can be computed independently:

$$\begin{aligned} a_0 &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) dx \\ a_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos kx dx \\ b_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin kx dx \end{aligned} \quad (12)$$

where $k = 1, 2, 3, \dots$

5.3 Projection

Consider the low-order approximation

$$f(\varphi) \approx \frac{a_0}{2} + a_1 \cos \varphi + a_2 \cos 2\varphi. \quad (13)$$

To compute the amplitudes $a = \{a_0, a_1, a_2\}$, use Hilbert's projection theorem, which is tantamount to using the method of least squares. Multiply both side of the equation by $\cos j\varphi$ and integrate over the domain to isolate the contribution of a_j . For example, to isolate the component a_2 ,

$$\begin{aligned} \int_{-\pi}^{\pi} f(\varphi) \cos 2\varphi d\varphi &\approx \int_{-\pi}^{\pi} \left(\frac{a_0}{2} + a_1 \cos \varphi + a_2 \cos 2\varphi \right) \cos 2\varphi d\varphi. \\ &= a_2 \int_{-\pi}^{\pi} \cos^2 2\varphi d\varphi. \end{aligned} \quad (14)$$

The result is the rather general formula

$$a_2 = \frac{1}{\pi} \int_{-\pi}^{\pi} f(\varphi) \cos 2\varphi d\varphi. \quad (15)$$

Because the basis functions are orthogonal, the modes are decoupled and the amplitudes can be computed independently.

5.4 Discrete Topology

The continuum formulation in L^2 is useful for understanding the critical components of the theory and makes a natural starting point for discussing the discrete topology of l^2 . The moment we discretize the problem in a computer code, we switch from the continuum to a discrete space. Out of all the known orthogonal polynomials, only two are orthogonal in both L^2 and l^2 , and these are the sine and cosine functions.

However, because of sampling mesh that was used in Mercury MoM, the data vectors are *not* orthogonal. But resolution is deferred until the end.

5.5 Special Case: The Mean

The Fourier series resolves a function into successively higher oscillations about a constant value. This constant value is the mean, and it is a least squares solution and in this effort, is a valuable bellwether.

5.5.1 The Method of Least Squares

$$r^2 = \sum_{k=1}^m (\sigma(\alpha_k, \beta_0) - \mu)^2 \quad (16)$$

$$D_\mu r^2 = -2(\sigma(\alpha_k, \beta_0) - \mu) = 0 \quad (17)$$

The least squares solution for the parameter μ is simply the average of the data:

$$\mu = m^{-1} \sigma(\alpha_k, \beta_0). \quad (18)$$

Compare to a_0

5.5.2 Continuum to discrete

The continuum formulation translates to discrete spaces using the idea of the integral as a Riemann sum:

$$\int_{-\pi}^{\pi} \cos j\alpha d\alpha \Rightarrow \sum_{k=1}^m \cos(j\alpha_k) \Delta_k. \quad (19)$$

Here Δ_k , the interval length, corresponds to the Riemann integration measure $d\alpha$. A basic result from the calculus is that the average $\langle f(x) \rangle$ of an integrable function over a finite, ordered interval (a, b) is

$$\langle f(x) \rangle = \frac{\int_b^a f(x) dx}{b - a}, \quad (20)$$

where the interval length $L = b - a$. The connection to the discrete case is clear with the realization that the Mercury MoM results are spaced at regular intervals, that is,

$$\Delta_k = \Delta.$$

Then the interval length is

$$L = \sum_{k=1}^m \Delta_k = \sum_{k=1}^m \Delta = m\Delta. \quad (21)$$

$$\frac{\int_b^a f(x) dx}{L} \Rightarrow \frac{\sum_{k=1}^m f(x)\Delta}{m\Delta} = \frac{\sum_{k=1}^m f(x)}{m} \quad (22)$$

5.6 Linear Independence

5.7 Quality of Fit

6 File Types

Standard file types

1. *.obj
2. *.txt

Intrinsic file types

1. *.geo
2. *.facet

6.1 Geometry Files *.obj

The geometry files are the primary input to Hg MoM.

1. Mathematica: Import/Export format OBJ
2. Wikipedia: Wavefront .obj file
3. All3DP: The OBJ File Format – Simply Explained

Listing 1: The *.obj file for the sphere with 20 facets.

```

1 # Created with the Wolfram Language : www.wolfram.com
2 mtllib sphere-d050-01.mtl
3
4 # 12 vertex positions
5 v 13.81966018676758 42.53253936767578 22.36067962646484
6 v -13.81966018676758 42.53253936767578 -22.36067962646484
7 v -36.18033981323242 26.28655624389648 22.36067962646484
8 v -36.18033981323242 -26.28655624389648 22.36067962646484
9 v -13.81966018676758 -42.53253936767578 -22.36067962646484
10 v 13.81966018676758 -42.53253936767578 22.36067962646484

```

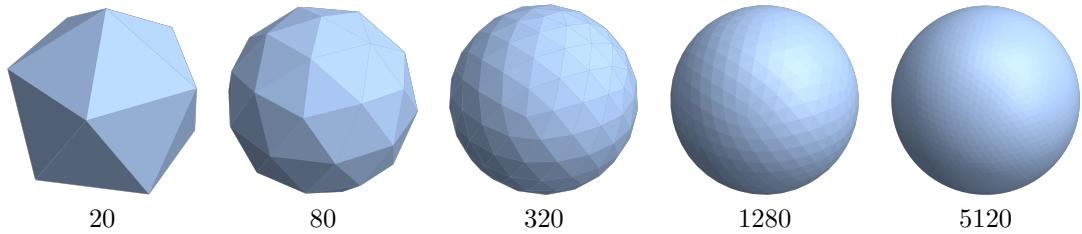


Table 7: A sphere resolved into facets in an *.obj file. The application irradiates each facet and aggregates the output.

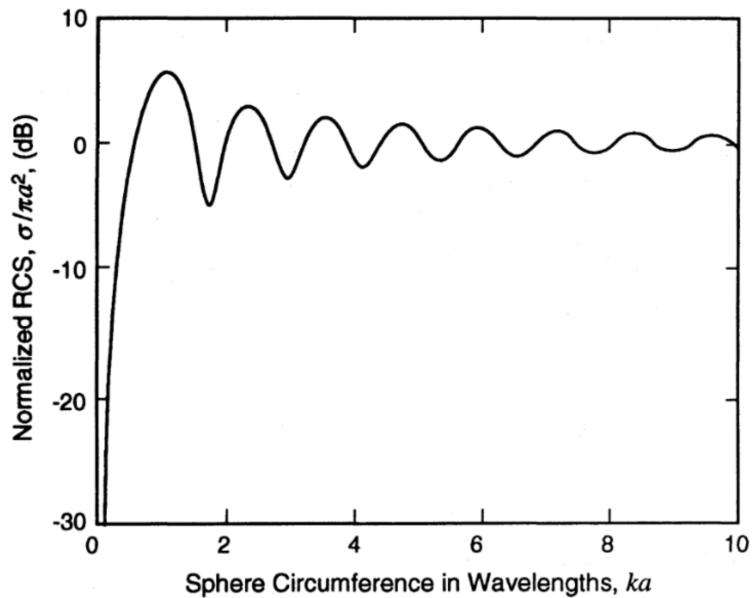


Figure 4.2. Sphere RCS backscatter as a function of ka as computed by the algorithm shown in Listing 4.1.

Table 8: Closed form solution for spherical scattering taken from [7, p. 123].

```

11 v 36.18033981323242 26.28655624389648 -22.36067962646484
12 v -44.72135925292969 0 -22.36067962646484
13 v 44.72135925292969 0 22.36067962646484
14 v 36.18033981323242 -26.28655624389648 -22.36067962646484
15 v 0 0 50
16 v 0 0 -50
17
18 # 0 UV coordinates
19
20 # 0 vertex normals
21
22 # Mesh `` with 20 faces
23 usemtl DefaultMaterial
24 f 1/ 2/ 3/
25 f 4/ 5/ 6/
26 f 2/ 1/ 7/
27 f 4/ 3/ 8/
28 f 9/ 6/ 10/
29 f 3/ 4/ 11/
30 f 2/ 7/ 12/
31 f 3/ 2/ 8/
32 f 2/ 12/ 8/
33 f 12/ 5/ 8/
34 f 7/ 1/ 9/
35 f 5/ 4/ 8/
36 f 6/ 5/ 10/
37 f 5/ 12/ 10/
38 f 12/ 7/ 10/
39 f 7/ 9/ 10/
40 f 4/ 6/ 11/
41 f 6/ 9/ 11/
42 f 9/ 1/ 11/
43 f 1/ 3/ 11/

```

Listing 2: The `*.facet` file for the sphere with 20 facets.

```

1 facimusFacet.f08 2020-06-25 11:34:36
2      1
3 <partName>
4 0
5      12
6      13.819660    42.532539    22.360680
7      -13.819660   42.532539   -22.360680
8      -36.180340   26.286556   22.360680
9      -36.180340   -26.286556   22.360680
10     -13.819660   -42.532539  -22.360680
11     13.819660   -42.532539  22.360680
12     36.180340   26.286556  -22.360680
13     -44.721359   0.000000  -22.360680
14     44.721359   0.000000  22.360680
15     36.180340   -26.286556 -22.360680
16     0.000000   0.000000  50.000000
17     0.000000   0.000000 -50.000000
18      1
19 <partName>
20      3 20 0 0 0 0 0
21      1 2 3 0
22      4 5 6 0
23      2 1 7 0
24      4 3 8 0
25      9 6 10 0
26      3 4 11 0
27      2 7 12 0
28      3 2 8 0
29      2 12 8 0
30      12 5 8 0
31      7 1 9 0
32      5 4 8 0
33      6 5 10 0

```

```

34      5    12    10    0
35     12     7    10    0
36      7     9    10    0
37      4     6    11    0
38      6     9    11    0
39      9     1    11    0
40      1     3    11    0

```

6.2 New Efforts

7 Running Mercury Method of Moments

7.1 Inputs

Consider an example with the sphere.

Listing 3: “tabula-rasa.geo”

```

1  ego
2
3 !Mercury MoM input file, VIE/SIE Version 4.x compatible (VIE/Dual Sided SIE)
4
5 &MM_MOM
6   bUseACA = .TRUE.,
7   bSolve_ACA = .TRUE.,
8   bOutOfCore = .TRUE.,
9   bNormalizeToWaveLength = .FALSE.,
10  bNormalize          = .FALSE.,
11  dCloseLambda = 0.100000,
12  ACA_Factor_Tol = 0.000010,
13  ACA_RHS_Tol = 0.000100,
14  Point_Tolerance = 0.001000,
15  nLargestBlockSize = 400,
16  MemorySize_GB = -1.000000,
17  stackSize_GB = -1.000000,
18  nFillThreads = -1,
19  nFillMKLThreads = 1,
20  nLUThreads = -1,
21  nLUMKLThreads = 1,
22  nRHSTThreads = 1,
23  nRHSMKLThreads = 1,
24  bOutputACAGrouping = .FALSE.,
25  bOutputRankFraction = .FALSE.,
26  bLimitLUColumns = .FALSE.,
27  Lop_Admissibility = WEAK,
28  Kop_Admissibility = CLOSE
29 /
30
31 &Scratch_Memory
32   Scratch_RankFraction_Z = 0.300000,
33   Scratch_RankFraction_LU = 0.600000,
34   Scratch_RankFraction_RHS = 0.500000,
35   Scratch_RankFraction_Solve = 1.000000,
36   MemoryFraction_Z = 0.950000,
37   MemoryFraction_Scratch_LU = 0.500000,
38   MemoryFraction_LU = 1.000000,
39   MemoryFraction_RHS = 0.500000,
40   MemoryFraction_Solve = 0.900000,
41 /
42
43 &QUADRATURE
44   NTRISELF = 7,
45   NTRINEAR = 3,
46   NTRIFAR = 3,
47   NTETSELF = 11,

```

```

48  NTETNEAR    = 4,
49  NTETFAR     = 4,
50  NQGAUSS    = 4,
51 /
52
53 FREQUENCY
54   mhz
55   nu-mhz  nu-mhz  1 !Freq Start, Freq Stop, Number of Frequencies
56
57 Excitation
58   MONOSTATIC
59
60 Angle Cut
61   1
62   0.000000 359.000000 360
63   AZIMUTH
64   90.000000
65
66 Boundary Conditions
67 PEC-Materials.lib
68 4
69 V_FREE_SPACE => Free_Space
70 V_PEC => PEC
71 V_PMC => PMC
72 V_NULL => NULL
73 1
74 0 BC_PEC V_FREE_SPACE
75
76 SIE
77 myFacet.facet
78 m
79
80 Geometry_End
81
82 ! Fiducial run

```

Listing 4: A simple *.facet file

```

1 facimusFacet.f08 2020-06-25 11:34:36
2   1
3 <partName>
4 0
5   12
6   13.819660    42.532539    22.360680
7   -13.819660   42.532539   -22.360680
8   -36.180340   26.286556    22.360680
9   -36.180340   -26.286556   22.360680
10  -13.819660   -42.532539  -22.360680
11  13.819660   -42.532539   22.360680
12  36.180340    26.286556  -22.360680
13  -44.721359    0.000000  -22.360680
14  44.721359    0.000000   22.360680
15  36.180340   -26.286556  -22.360680
16  0.000000     0.000000   50.000000
17  0.000000     0.000000  -50.000000
18   1
19 <partName>
20   3    20    0    0    0    0    0
21   1    2     3    0
22   4    5     6    0
23   2    1     7    0
24   4    3     8    0
25   9    6    10    0
26   3    4    11    0
27   2    7    12    0
28   3    2     8    0
29   2   12     8    0
30   12   5     8    0
31    7    1     9    0

```

32	5	4	8	0
33	6	5	10	0
34	5	12	10	0
35	12	7	10	0
36	7	9	10	0
37	4	6	11	0
38	6	9	11	0
39	9	1	11	0
40	1	3	11	0

7.2 Outputs

untitled text 4

```

1 Freq = 10.00E+00 MHz
2 Lambda = 29.98E-00 m
3 k = 209.56E-03 m-1
4
5
6 BACKSCATTER RCS RESULTS .....
7
8 Theta, Phi, Theta-Theta (complex efield), Phi-Theta (complex efield), Theta-Phi (complex efield), Phi-Phi (complex efield)
9
10 90.0000, 0.0000, ( 0.2333228E-01, 0.1410688E-03), ( 0.8868390E-04, 0.5353989E-05), ( 0.8868393E-04, 0.5354086E-05), ( 0.2275015E-01, 0.1270486E-03)
11 90.0000, 1.0000, ( 0.2333244E-01, 0.1402642E-03), ( 0.8605096E-04, 0.5255909E-05), ( 0.8604993E-04, 0.5248534E-05), ( 0.2274918E-01, 0.1264947E-03)
12 90.0000, 2.0000, ( 0.2333250E-01, 0.1393928E-03), ( 0.8339065E-04, 0.5144119E-05), ( 0.8338990E-04, 0.5151524E-05), ( 0.2274821E-01, 0.1259348E-03)
13 90.0000, 3.0000, ( 0.2333247E-01, 0.1384604E-03), ( 0.8070502E-04, 0.5039624E-05), ( 0.8070488E-04, 0.5032761E-05), ( 0.2274726E-01, 0.1253780E-03)
14 90.0000, 4.0000, ( 0.2333289E-01, 0.1374878E-03), ( 0.7799496E-04, 0.4922949E-05), ( 0.7799467E-04, 0.4916138E-05), ( 0.2274633E-01, 0.1248239E-03)
15 90.0000, 5.0000, ( 0.2333302E-01, 0.1364364E-03), ( 0.7526141E-04, 0.4795374E-05), ( 0.7526139E-04, 0.4787108E-05), ( 0.2274541E-01, 0.1242470E-03)
16 90.0000, 6.0000, ( 0.2333315E-01, 0.1353522E-03), ( 0.7250279E-04, 0.4655677E-05), ( 0.7250353E-04, 0.4652742E-05), ( 0.2274450E-01, 0.1236636E-03)
17 90.0000, 7.0000, ( 0.2333320E-01, 0.1342022E-03), ( 0.6972432E-04, 0.4524583E-05), ( 0.6972397E-04, 0.4519492E-05), ( 0.2274361E-01, 0.1230940E-03)
18 90.0000, 8.0000, ( 0.2333339E-01, 0.1333009E-03), ( 0.6692425E-04, 0.4379934E-05), ( 0.6692352E-04, 0.4374713E-05), ( 0.2274274E-01, 0.1225080E-03)
19 90.0000, 9.0000, ( 0.2333352E-01, 0.1323333E-03), ( 0.6436104E-04, 0.4208323E-05), ( 0.6436093E-04, 0.4197709E-05), ( 0.2274180E-01, 0.1219030E-03)
20 90.0000, 10.0000, ( 0.2333365E-01, 0.1313757E-03), ( 0.6174190E-04, 0.4040028E-05), ( 0.6174189E-04, 0.4029747E-05), ( 0.2274107E-01, 0.1214930E-03)
21 90.0000, 11.0000, ( 0.2333371E-01, 0.1302903E-03), ( 0.5840381E-04, 0.3920618E-05), ( 0.5840380E-04, 0.3918686E-05), ( 0.2274003E-01, 0.1207031E-03)
22 90.0000, 12.0000, ( 0.2333380E-01, 0.1276849E-03), ( 0.5552738E-04, 0.3757901E-05), ( 0.5552594E-04, 0.3756007E-05), ( 0.2273949E-01, 0.1200838E-03)
23 90.0000, 13.0000, ( 0.2333388E-01, 0.12652230E-03), ( 0.5263346E-04, 0.3593899E-05), ( 0.5263261E-04, 0.3591580E-05), ( 0.2273871E-01, 0.1194466E-03)
24 90.0000, 14.0000, ( 0.2333395E-01, 0.1247289E-03), ( 0.4972452E-04, 0.3426847E-05), ( 0.4972324E-04, 0.3422775E-05), ( 0.2273798E-01, 0.1188115E-03)
25 90.0000, 15.0000, ( 0.2333402E-01, 0.1231622E-03), ( 0.4679928E-04, 0.3249850E-05), ( 0.4679913E-04, 0.3247067E-05), ( 0.2273727E-01, 0.1181715E-03)
26 90.0000, 16.0000, ( 0.2333408E-01, 0.1215549E-03), ( 0.4366077E-04, 0.3071021E-05), ( 0.4366053E-04, 0.3071021E-05), ( 0.2273658E-01, 0.1175054E-03)
27 90.0000, 17.0000, ( 0.2333414E-01, 0.1199663E-03), ( 0.4098046E-04, 0.2891643E-05), ( 0.4098083E-04, 0.2891672E-05), ( 0.2273593E-01, 0.1168209E-03)
28 90.0000, 18.0000, ( 0.2333418E-01, 0.1182040E-03), ( 0.3794509E-04, 0.2791195E-05), ( 0.3794405E-04, 0.2696849E-05), ( 0.2273530E-01, 0.1161288E-03)
29 90.0000, 19.0000, ( 0.2333422E-01, 0.1164673E-03), ( 0.3497020E-04, 0.2512710E-05), ( 0.3497056E-04, 0.2501005E-05), ( 0.2273469E-01, 0.1154252E-03)
30 90.0000, 20.0000, ( 0.2333425E-01, 0.1146772E-03), ( 0.3198408E-04, 0.2319652E-05), ( 0.3198424E-04, 0.2317885E-05), ( 0.2273412E-01, 0.1146960E-03)
31 90.0000, 21.0000, ( 0.2333428E-01, 0.1128494E-03), ( 0.2898948E-04, 0.2112308E-05), ( 0.2898854E-04, 0.2120017E-05), ( 0.2273357E-01, 0.1139418E-03)
32 90.0000, 22.0000, ( 0.2333431E-01, 0.11109812E-03), ( 0.2598479E-04, 0.1916728E-05), ( 0.2598479E-04, 0.1916728E-05), ( 0.2273306E-01, 0.1131798E-03)
33 90.0000, 23.0000, ( 0.2333438E-01, 0.1093430E-03), ( 0.2283058E-04, 0.1770430E-05), ( 0.2283058E-04, 0.1770430E-05), ( 0.2273258E-01, 0.1126160E-03)
34 90.0000, 24.0000, ( 0.2333443E-01, 0.1071143E-03), ( 0.1956589E-04, 0.1524590E-05), ( 0.1956589E-04, 0.1524590E-05), ( 0.2273213E-01, 0.1115701E-03)
35 90.0000, 25.0000, ( 0.2333449E-01, 0.1051199E-03), ( 0.1693341E-04, 0.1322052E-05), ( 0.1693341E-04, 0.1322052E-05), ( 0.2273177E-01, 0.1107367E-03)
36 90.0000, 26.0000, ( 0.2333452E-01, 0.10329011E-03), ( 0.13209011E-04, 0.1108616E-05), ( 0.13209011E-04, 0.1108623E-05), ( 0.2273123E-01, 0.1098675E-03)
37 90.0000, 27.0000, ( 0.2333456E-01, 0.10180452E-03), ( 0.1087428E-04, 0.9874643E-06), ( 0.1087428E-04, 0.9874643E-06), ( 0.2273096E-01, 0.1089735E-03)
38 90.0000, 28.0000, ( 0.2333452E-01, 0.9894501E-04), ( 0.7839484E-05, 0.7839482E-06), ( 0.7839484E-05, 0.7806624E-06), ( 0.2273064E-01, 0.1088477E-03)
39 90.0000, 29.0000, ( 0.2333450E-01, 0.9680697E-04), ( 0.4801968E-05, 0.4874063E-06), ( 0.4801968E-05, 0.4862241E-06), ( 0.2273035E-01, 0.1070995E-03)
40 90.0000, 30.0000, ( 0.2333416E-01, 0.9465464E-04), ( 0.1765148E-05, 0.1764301E-06), ( 0.1765148E-05, 0.1762104E-06), ( 0.2273009E-01, 0.1061108E-03)
41 90.0000, 31.0000, ( 0.2333411E-01, 0.9246088E-04), ( 0.12730341E-05, 0.7069519E-07), ( 0.12730341E-05, 0.7069519E-07), ( 0.2272987E-01, 0.1050739E-03)
42 90.0000, 32.0000, ( 0.2333405E-01, 0.90233387E-04), ( 0.13098902E-05, 0.6741837E-07), ( 0.13098902E-05, 0.6741837E-07), ( 0.2272967E-01, 0.1040102E-03)
43 90.0000, 33.0000, ( 0.2333399E-01, 0.8799594E-04), ( 0.1343749E-05, 0.5359628E-06), ( 0.1343749E-05, 0.5359628E-06), ( 0.2272952E-01, 0.1029183E-03)
44 90.0000, 34.0000, ( 0.2333392E-01, 0.8569709E-04), ( 0.1355683E-05, 0.5586803E-06), ( 0.1355683E-05, 0.5586803E-06), ( 0.2272940E-01, 0.1017700E-03)
45 90.0000, 35.0000, ( 0.2333385E-01, 0.8340531E-04), ( 0.1340611E-04, 0.7677730E-06), ( 0.1340611E-04, 0.7677730E-06), ( 0.2272931E-01, 0.1005938E-03)
46 90.0000, 36.0000, ( 0.2333381E-01, 0.8107616E-04), ( 0.1646228E-04, 0.6359825E-06), ( 0.1646228E-04, 0.6359825E-06), ( 0.2272925E-01, 0.9957707E-04)
47 90.0000, 37.0000, ( 0.2333378E-01, 0.7873635E-04), ( 0.1747474E-04, 0.5117308E-06), ( 0.1747474E-04, 0.5117308E-06), ( 0.2272918E-01, 0.9810371E-04)
48 90.0000, 38.0000, ( 0.2333375E-01, 0.7643654E-04), ( 0.2246019E-04, 0.4137767E-05), ( 0.2246019E-04, 0.4137767E-05), ( 0.2272925E-01, 0.9677792E-04)
49 90.0000, 39.0000, ( 0.2333373E-01, 0.7394435E-04), ( 0.2546455E-04, 0.15829728E-05), ( 0.2546455E-04, 0.15829728E-05), ( 0.2272929E-01, 0.9583405E-04)
50 90.0000, 40.0000, ( 0.2333372E-01, 0.7153424E-04), ( 0.2846110E-04, 0.1786374E-05), ( 0.2846110E-04, 0.1786374E-05), ( 0.2272937E-01, 0.9402471E-04)
51 90.0000, 41.0000, ( 0.2333373E-01, 0.6909868E-04), ( 0.3144751E-04, 0.1982437E-05), ( 0.3144751E-04, 0.1982437E-05), ( 0.2272949E-01, 0.9257005E-04)
52 90.0000, 42.0000, ( 0.2333371E-01, 0.6664295E-04), ( 0.3442489E-04, 0.2177858E-05), ( 0.3442489E-04, 0.2177858E-05), ( 0.2272963E-01, 0.9106311E-04)
53 90.0000, 43.0000, ( 0.2333373E-01, 0.6418363E-04), ( 0.3733910E-04, 0.23738527E-05), ( 0.3733910E-04, 0.23738527E-05), ( 0.2272981E-01, 0.8959324E-04)
54 90.0000, 44.0000, ( 0.2333291E-01, 0.6171301E-04), ( 0.4034362E-04, 0.2561433E-05), ( 0.4034362E-04, 0.2561433E-05), ( 0.2273003E-01, 0.8788771E-04)
55 90.0000, 45.0000, ( 0.2333278E-01, 0.5922416E-04), ( 0.4328624E-04, 0.2751016E-05), ( 0.4328605E-04, 0.2742658E-05), ( 0.2273027E-01, 0.8622059E-04)
56 90.0000, 46.0000, ( 0.2333264E-01, 0.5670952E-04), ( 0.4621203E-04, 0.2930805E-05), ( 0.4621203E-04, 0.2936432E-05), ( 0.2273055E-01, 0.8449983E-04)
57 90.0000, 47.0000, ( 0.2333250E-01, 0.5419380E-04), ( 0.4912426E-04, 0.3104662E-05), ( 0.4912537E-04, 0.3112095E-05), ( 0.2273086E-01, 0.8272303E-04)
58 90.0000, 48.0000, ( 0.2333236E-01, 0.5167195E-04), ( 0.5202224E-04, 0.3285702E-05), ( 0.5202224E-04, 0.3284841E-05), ( 0.2273119E-01, 0.80890130E-04)
59 90.0000, 49.0000, ( 0.2333221E-01, 0.4914165E-04), ( 0.5490336E-04, 0.3457679E-05), ( 0.5490287E-04, 0.3454552E-05), ( 0.2273156E-01, 0.7991491E-04)
60 90.0000, 50.0000, ( 0.2333216E-01, 0.4659749E-04), ( 0.5774821E-04, 0.3628128E-05), ( 0.5765651E-04, 0.3636154E-05), ( 0.2273174E-01, 0.7776268E-04)
61 90.0000, 51.0000, ( 0.2333208E-01, 0.4408797E-04), ( 0.6081237E-04, 0.3768658E-05), ( 0.6081237E-04, 0.3768803E-05), ( 0.2273240E-01, 0.7585774E-04)
62 90.0000, 52.0000, ( 0.2333174E-01, 0.4150420E-04), ( 0.6343987E-04, 0.3945118E-05), ( 0.6344026E-04, 0.3940577E-05), ( 0.2273285E-01, 0.7301238E-04)

```

8 Additional Information

8.1 YouTube Videos

YouTube offers useful didactic presentations and simulations.

1. The Radar cross-section of backscattering objects
2. Basic Concepts of Radar Cross Section (RCS)
3. Mie scattering
4. Mie theory (BME51 Lecture 5)
5. Mie Scattering

8.2 Further Reading

Radar rudiments

1. D. K. Barton and H.R. Ward. *Handbook of Radar Measurement*. New York, NY: Penguin Random House, 1969
2. Andrei A. Kolosov. *Over the Horizon Radar*. Artech House, 1987. ISBN: 9780890062333. URL: <https://us.artechhouse.com/Over-the-Horizon-Radar-P254.aspx>
3. Peyton Z Peebles. *Radar principles*. John Wiley & Sons, 2007

Radar cross section

1. JW Jr Crispin. *Methods of radar cross-section analysis*. Elsevier, 2013
2. Allen E Fuhs. *Radar cross section lectures*. Monterey, California, Naval Postgraduate School, 1982. URL: <https://calhoun.nps.edu/server/api/core/bitstreams/9e69ec48-4628-4243-9f9b-7e879521f7f8/content>
3. Eugene F Knott, John F Schaeffer, and Michael T Tulley. *Radar cross section*. SciTech Publishing, 2004
4. M Madheswaran and P Suresh Kumar. “Estimation of wide band radar cross section (RCS) of regular shaped objects using method of moments (MOM)”. In: *Ictact Journal on Communication Tech-nology* 3.2 (2012), pp. 536–541

Method of Moments

1. Walton C Gibson. *The method of moments in electromagnetics*. Chapman and Hall/CRC, 2021
2. Roger F Harrington. “The method of moments in electromagnetics”. In: *Journal of Electromagnetic waves and Applications* 1.3 (1987), pp. 181–200
3. Cai-Cheng Lu and Chong Luo. “Comparison of iteration convergences of SIE and VSIE for solving electromagnetic scattering problems for coated objects”. In: *Radio Science* 38.2 (2003), pp. 11–1
4. Jiade Yuan, Changqing Gu, and Guodong Han. “Efficient generation of method of moments matrices using equivalent dipole-moment method”. In: *IEEE Antennas and Wireless Propagation Letters* 8 (2009), pp. 716–719

Using Mercury MoM and post-processing

1. Daniel Topa. *Radar Cross Section Models for AFCAP Dashboard: Rapid Report 2020-02: Corrected*. Briefing. Mar. 2020
2. Daniel Topa. *Mercury Method of Moments Adjunct Visualization Tool: Trials and Tribulations*. Tech. rep. ARFL/RVB, Apr. 2020
3. Daniel Topa. *Radar Cross Section: Phase 1 Summary Report*. Tech. rep. ARFL/RVB, Apr. 2020
4. Daniel Topa. *Mercury Method of Moments: AFRL Quick Start Guide*. Tech. rep. AFRL, 2020

A Mercury Method of Moments: Data Formats

A.1 Numeric Results

The MoM RCS data is delivered in a matrix with m rows and n columns (standard matrix addressing).

$1 \leq m \leq 28$ MHz (integer steps)
 $1 \leq n \leq 90$ degrees (integer steps)

The matrix is WIDE (more columns than rows)

Frequency partition: row 1: 3 MHz row 2: 4 MHz row 28: 30 MHz

Let r index the rows. Then frequency ν is in row $= \nu - 2$

Angular partition col 1: 0 col 2: 1 . . . col 181: 180
col 1 col 2 col 3 col 181 0 1 2 . . . 180

Let c be the column index. The measurement for angle alpha is in column $c = \text{alpha} + 1$

The test asset is symmetric: $\sigma(\alpha) = \sigma(-\alpha)$

But the matrix can easily be delivered in other forms, such as the transpose (interchange rows and columns), or packed into a linear array.

Sample:

```
4.16411, 4.14247, 4.07319, 3.95637, 3.79263, 3.58287, 3.32827, 3.0303, . . .
18.2776, 18.2369, 18.1199, 17.9248, 17.6523, 17.3041, 16.8817, 16.3876, . . .
25.6306, 25.5886, 25.463, 25.2538, 24.9618, 24.5882, 24.1346, 23.6028, . . .
. . .
```

B Mercury Method of Moments: Software Toolkit

Mercury MoM produces thousands of lines of output to a `*.4112.txt` file, a mix of numbers and strings. Once the data portions are located, they can be harvested straightforwardly. However, the text messages include debug information and the text patterns are varied.

Data analysis on data sets with a large number of facets can take several hours.

B.1 rcsharvester.f08

```
! harvest the electric field values from the ASCII file *.4112.txt mixed text and numeric
lines
! compute the mean total RCS and write these values
```

B.1.1 Class Electric Fields: m-class-electric-fields.f08

B.1.2 m-class-electric-fields.f08

The primary output of the simulation are the electric fields. Lines 17-24 define the class; the remainder of the codes is for methods. The input electric field is resolved into two polarization axes: horizontal and vertical. Each of these fields are resolved into horizontal and vertical components creating four complex vectors (line 21) whose length matches the angular sample size.

The class `m-class-electric-fields.f08` reads the text file and harvests the electric fields eventually passing back a composite value (lines 65-66) for all four components of the scattering return.

```

1 ! Parses alphanumeric line from MoM * 4112.txt and extracts electric field values
2 module mClassElectricFields
3
4   use mFormatDescriptors,           only : fmt_stat, fmt_iomsg
5   use mLibraryOfConstants,         only : cZero, MoMlineLength, messageLength
6   use mPrecisionDefinitions,      only : ip, rp
7
8   implicit none
9
10  integer ( ip ) :: left = 0, right = 0
11  integer :: io_stat = 0
12  character ( len = messageLength ) :: io_msg = ""
13  character ( len = 15 )           :: number = ""
14
15  ! theta = azimuth
16  ! phi = elevation (North Pole = 0, equator = 90)
17  type :: electricFields
18    real ( rp ) :: meanTotalRCS = 0.0_rp
19    real ( rp ) :: dbsm = 0.0_rp
20    real ( rp ) :: theta = 0.0_rp, phi = 0.0_rp
21    complex ( rp ) :: thetaTheta = cZero, thetaPhi = cZero, phiTheta = cZero, phiPhi = cZero
22  contains
23    procedure, public :: gather_mean_total_rcs => gather_mean_total_rcs_sub
24  end type electricFields
25
26  private :: gather_mean_total_rcs_sub
27  private :: compute_mean_total_rcs_sub, compute_dbsm_sub, extract_electric_fields_sub
28  private :: gather_complex_field_sub, gather_real_field_sub
29
30  ! parameters
31  integer ( ip ), parameter :: mll = MoMlineLength
32  ! finger print of data line: start and stop positions for each numerical field
33  ! load matrix as columns
34  ! sample data line:
35  ! 90.0000,     0.0000,( -0.4572920E+05,  0.8350829E+05),( 0.2034567E+06, -0.9493007E+05),( 0.2034813E+06, -0.9492184E+05),( -0.1727375E+06,  0.3787291E+05)
36  integer, parameter :: endpoints ( 1 : 10, 1 : 2 ) = &
37    reshape ( [ [ 1, 14, 28, 44, 62, 78, 96, 112, 130, 146 ], &
38 [ 12, 25, 42, 58, 76, 92, 110, 126, 144, 160 ] ], [ 10, 2 ] )
39  ! constructor
40  type ( electricFields ), parameter :: electricFields0 = &
41    electricFields ( meanTotalRCS = 0.0, theta = 0.0, phi = 0.0, &
42    thetaTheta = cZero, thetaPhi = cZero, phiTheta = cZero, phiPhi = cZero )
43  contains
44
45    ! master routine: only exposed procedure
46    subroutine gather_mean_total_rcs_sub ( me, textLine )
47      class ( electricFields ), target :: me
48      character ( len = mll ), intent ( in ) :: textLine
49      call extract_electric_fields_sub ( me, textLine )
50      call compute_mean_total_rcs_sub ( me )
51      call compute_dbsm_sub ( me )
52    return
53  end subroutine gather_mean_total_rcs_sub
54
55  ! Sciacca prescription
56  subroutine compute_dbsm_sub ( me )
57    class ( electricFields ), target :: me
58    me % dbsm = 10.0_rp * log10 ( me % meanTotalRCS )
59  return
60  end subroutine compute_dbsm_sub
61
62  ! Sciacca prescription
63  subroutine compute_mean_total_rcs_sub ( me )
64    class ( electricFields ), target :: me
65    me % meanTotalRCS = abs ( me % thetaTheta ) + abs ( me % thetaPhi ) &
66    + abs ( me % phiTheta ) + abs ( me % phiPhi )
67    me % meanTotalRCS = me % meanTotalRCS / real ( 2, kind = rp )
68  return
69  end subroutine compute_mean_total_rcs_sub
70
71  subroutine extract_electric_fields_sub ( me, textLine )
72    class ( electricFields ), target :: me
73    character ( len = mll ), intent ( in ) :: textLine
74    integer ( ip ) :: position = 0
75    ! move across text line gathering numeric values
76    position = 1
77    call gather_real_field_sub &
78      ( position = position, real_value = me % theta, textLine = textLine, fmt = "( f12.4 )" )
79    call gather_real_field_sub &
80      ( position = position, real_value = me % phi,   textLine = textLine, fmt = "( f12.4 )" )
81    call gather_complex_field_sub &
82      ( position = position, complex_value = me % thetaTheta, textLine = textLine )
83    call gather_complex_field_sub &
84      ( position = position, complex_value = me % thetaPhi,   textLine = textLine )
85    call gather_complex_field_sub &
86      ( position = position, complex_value = me % phiTheta,   textLine = textLine )
87    call gather_complex_field_sub &
88      ( position = position, complex_value = me % phiPhi,     textLine = textLine )

```

```

89      return
90  end subroutine extract_electric_fields_sub
91
92  subroutine gather_real_field_sub ( position, real_value, textLine, fmt )
93    real ( rp ),           intent ( out ) :: real_value
94    integer ( ip ),        intent ( inout ) :: position
95    character ( len = mll ), intent ( in ) :: textLine
96    character ( len = 9 ), intent ( in ) :: fmt
97    left = endpoints ( position, 1 )
98    right = endpoints ( position, 2 )
99    write ( number, fmt = 100 ) textLine ( left : right )
100   if ( io_stat /= 0 ) then
101     write ( *, fmt = '( 3g0 )' ) "Failure to WRITE string value '", trim ( textLine ( left : right ) ), "'."
102     write ( *, fmt = fmt_stat ) io_stat
103     write ( *, fmt = fmt_ioms ) trim ( io_msg )
104     stop "Error occurred in module 'mClassElectricFields', subroutine 'gather_real_field_sub'."
105   end if
106   read ( number, fmt = fmt ) real_value
107   if ( io_stat /= 0 ) then
108     write ( *, fmt = '( 5g0 )' ) "Failure to READ string value '", trim ( textLine ( left : right ) ), &
109     " as a REAL number using format descriptor ", fmt, "."
110     write ( *, fmt = fmt_stat ) io_stat
111     write ( *, fmt = fmt_ioms ) trim ( io_msg )
112     stop "Error occurred in module 'mClassElectricFields', subroutine 'gather_real_field_sub'."
113   end if
114   position = position + 1
115   return
116   100 format ( g0 )
117 end subroutine gather_real_field_sub
118
119 subroutine gather_complex_field_sub ( position, complex_value, textLine )
120   complex ( rp ),         intent ( out ) :: complex_value
121   integer ( ip ),         intent ( inout ) :: position
122   character ( len = mll ), intent ( in ) :: textLine
123   real ( rp ) :: x = 0.0_rp, y = 0.0_rp
124   call gather_real_field_sub ( position = position, real_value = x, textLine = textLine, fmt = "( e15.7 )" )
125   call gather_real_field_sub ( position = position, real_value = y, textLine = textLine, fmt = "( e15.7 )" )
126   complex_value = cmplx ( x, y )
127   return
128 end subroutine gather_complex_field_sub
129
130 end module mClassElectricFields

```

B.1.3 Class Data File: m-class-data-file.f08

```

1  module mClassDataFile
2
3  use, intrinsic :: iso_fortran_env, only : iostat_end
4  ! classes
5  use mClassAverages,          only : average, average0
6  use mClassElectricFields,    only : electricFields, electricFields0
7  use mClassMesh,              only : meshReal
8  use mAllocations,            only : allocationToolKit, allocationToolKit0
9  use mAllocationsSpecial,    only : allocate_rank_one_averages_sub
10 ! utilities
11 use mLibraryOfConstants,      only : fileNameLength, messageLength, MoMlineLength
12 ! use mBulkRCS,               only : BulkRCS, BulkRCS0
13 use mFileHandling,            only : safeopen_readonly, safeopen_writereplace
14 use mFormatDescriptors,       only : fmt_one, fmt_stat, fmt_ioms, fmt_shape2
15 use mPrecisionDefinitions,   only : ip_rp
16 use mTextFileUtilities,       only : count_lines_sub, mark_frequencies_sub, read_text_lines_sub, file_closer_sub, &
17 ! use mTextFileUtilities,       only : count_lines_sub, file_closer_sub, iostat_check_sub, mark_frequencies_sub, &
18 ! use mTextFileUtilities,       only : count_lines_sub, file_closer_sub, iostat_check_sub, mark_frequencies_sub, &
19 ! use mTextFileUtilities,       only : parse_name_sub, read_text_lines_sub
20 implicit none
21
22 ! parameters
23 integer ( ip ), parameter :: fnl = fileNameLength, msgl = messageLength, mll = MoMlineLength
24 character ( len = 9 ), parameter :: strAzimuth = "azimuth "
25 character ( len = 9 ), parameter :: strElevation = "elevation"
26 character ( len = * ), parameter :: moduleCrash = "Program crashed in module 'mClassDataFile', "
27
28 integer :: io_stat = 0
29 character ( len = msgl ) :: io_msg = ""
30
31 type :: datafile4112
32   ! rank 2
33   real ( rp ),           allocatable :: rcs_table ( : , : ) ! angle mesh length x nu mesh length
34   real ( rp ),           allocatable :: dbsm_table ( : , : ) ! angle mesh length x nu mesh length
35   ! ! rank 1
36   integer ( ip ),         allocatable :: lineNumbersFrequency ( : ), &
37   ! ! rank 1
38   type ( average ),      allocatable :: perFrequencyAverage ( : ) ! nu mesh length
39   type ( average ) ,      :: globalAverage
40   character ( len = mll ), allocatable :: lines4112Text ( : )           ! length numlines4112Text
41   ! rank 0
42   type ( electricFields ) :: eFields = electricFields0
43   type ( meshReal ) :: meshFrequency, &

```

```

44      meshFreeAngle
45      integer ( ip ) :: numFrequencies = 0, &
46          numFixedAngles = 0, &
47          numFreeAngles = 0, &
48          numMeasurements = 0, &
49          numLines4112Text = 0
50      integer ( ip ) :: io_unit = 0
51      character ( len = 9 ) :: angleFixedType = "", angleFreeType = ""
52      character ( len = fnl ) :: file4112Name = "", fileRCStxtName = "", fileRCSbinaryName = "", &
53          filedBsmTxtName = "", filedBsmBinaryName = ""
54      ! allocation tools
55      type ( allocationToolKit ) :: myKit = allocationToolKit
56      contains
57          procedure, public :: allocate_rcs_tables
58              => allocate_rcs_tables_sub, &
59                  allocate_rcsAverages
59          procedure, public :: characterize_rcs_by_frequency
59              => characterize_rcs_by_frequency_sub, &
60          procedure, public :: check_rcs_table_structure
60              => check_rcs_table_structure_sub, &
61          procedure, public :: establish_free_angle_mesh
61              => establish_free_angle_mesh_sub, &
62          procedure, public :: establish_frequency_mesh
62              => establish_frequency_mesh_sub, &
63          procedure, public :: extract_rcs_from_4112_file
63              => extract_rcs_from_4112_file_sub, &
64          procedure, public :: harvest_frequencies
64              => harvest_frequencies_sub, &
65          procedure, public :: set_file_names
65              => set_file_names_sub, &
66          procedure, public :: set_free_angle_azimuth
66              => set_free_angle_azimuth_sub, &
67          procedure, public :: set_free_angle_elevation
67              => set_free_angle_elevation_sub, &
68          procedure, public :: write_rcs_file_set
68              => write_rcs_file_set_sub, &
69          procedure, public :: write_rcs_binary
69              => write_rcs_binary_sub, &
70          procedure, public :: write_rcs_csv
70              => write_rcs_csv_sub, &
71          procedure, public :: write_dBsm_binary
71              => write_dBsm_binary_sub, &
72          procedure, public :: write_dBsm_csv
72              => write_dBsm_csv_sub, &
73          procedure, public :: write_summary_by_frequency
73              => write_summary_by_frequency_sub, &
74          procedure, public :: write_summary_for_all_frequencies
74              => write_summary_for_all_frequencies_sub
75      end type dataFile4112
76
77      private :: allocate_rcs_tables_sub, allocate_rcsAverages_sub, &
78          establish_free_angle_mesh_sub, establish_frequency_mesh_sub, extract_rcs_from_4112_file_sub, &
79          harvest_frequencies_sub, &
80          set_file_names_sub, set_free_angle_azimuth_sub, set_free_angle_elevation_sub, &
81          write_summary_by_frequency_sub, write_summary_for_all_frequencies_sub
82
83      contains
84
85          subroutine characterize_rcs_by_frequency_sub ( me )
86              class ( dataFile4112 ), target :: me
87              type ( average ), pointer :: p => null ( )
88              integer ( ip ) :: kFrequency = 0
89
90              sweep_frequencies: do kFrequency = 1, me % numFrequencies
91                  p => me % perFrequencyAverage ( kFrequency )
92                  call p % find_max_and_min ( vector = me % rcs_table ( 1 : me % numFreeAngles, kFrequency ) )
93                  call p % compute_mean_and_variance ( vector = me % rcs_table ( 1 : me % numFreeAngles, kFrequency ), &
94                      one = me % meshFreeAngle % one )
95
96              p => null ( )
96          end do sweep_frequencies
97
98          return
99      end subroutine characterize_rcs_by_frequency_sub
100
101     module subroutine write_summary_for_all_frequencies_sub ( me )
102         class ( dataFile4112 ), target :: me
103         integer ( ip ) :: kFrequency = 0, first = 0, last = 0, numConvolution = 0
104         real ( rp ), allocatable :: global_rcs ( : ), one ( : )
105         ! allocate memory for all RCS measurements
106         numConvolution = me % numFrequencies * me % numFreeAngles
107         call me % myKit % allocate_rank_one_reals ( real_array = global_rcs, index_min = 1, index_max = numConvolution )
108         call me % myKit % allocate_rank_one_reals ( real_array = one, index_min = 1, index_max = numConvolution )
109         ! load data vector
110         sweep_frequencies: do kFrequency = 1, me % meshFrequency % numMeshElements
111             first = ( kFrequency - 1 ) * me % numFreeAngles + 1
112             last = first + me % numFreeAngles - 1
113             global_rcs ( first : last ) = me % rcs_table ( 1 : me % numFreeAngles, kFrequency )
114         end do sweep_frequencies
115         ! compute extrema
116         one ( : ) = global_rcs ( : ) - global_rcs ( : ) + 1.0_rp
117         call me % globalAverage % find_max_and_min ( vector = global_rcs ( 1 : numConvolution ) )
118         call me % globalAverage % compute_mean_and_variance ( vector = global_rcs ( 1 : numConvolution ), one = one )
119         write ( *, * )
120         write ( *, fmt = 100 ) me % globalAverage % mean, &
121             me % globalAverage % standardDeviation, &
122             me % globalAverage % extrema % minValue, &
123             me % globalAverage % extrema % maxValue
124
125         return
126         100 format ( "Aggregate for all RCS measurements: mean = ", g0, " +/- ", g0, ", min = ", g0, ", max = ", g0 )
126     end subroutine write_summary_for_all_frequencies_sub
127
128     module subroutine write_summary_by_frequency_sub ( me )
129         class ( dataFile4112 ), target :: me
130         integer ( ip ) :: kFrequency = 0
131         write ( *, * )
132         sweep_frequencies: do kFrequency = 1, me % meshFrequency % numMeshElements
132             write ( *, fmt = 100 ) kFrequency, me % meshFrequency % meshValues ( kFrequency ), &
133                 me % perFrequencyAverage ( kFrequency ) % mean, &
134                 me % perFrequencyAverage ( kFrequency ) % standardDeviation, &

```

```

136
137
138      end do sweep_frequencies
139
140      return
141      100 format ( I3.3, ". nu = ", g0, ", mean RCS = ", g0, " +/- ", g0, ", min = ", g0, ", max = ", g0 )
142
143  end subroutine write_summary_by_frequency_sub
144
145  module subroutine write_rcs_file_set_sub ( me )
146    class ( datafile4112 ), target :: me
147    call me % write_rcs_csv ( )
148    call me % write_rcs_binary ( )
149    call me % write_dBsm_csv ( )
150    call me % write_dBsm_binary ( )
151
152  end subroutine write_rcs_file_set_sub
153
154  module subroutine write_rcs_binary_sub ( me )
155    class ( datafile4112 ), target :: me
156    integer ( ip ) :: io_rcs = 0
157    character ( len = msgl ) :: crashChain = ""
158
159    crashChain = moduleCrash // "subroutine 'write_rcs_binary_sub'."
160
161    open ( newunit = io_rcs, file = me % fileRCSbinaryName, action = 'WRITE', status = 'REPLACE', form = 'UNFORMATTED', &
162           iostat = io_stat, iomsq = io_msg )
163    call iostat_check_sub ( action = "UNFORMATTED OPENING", fileName = me % fileRCSbinaryName, crashChain = crashChain, &
164           iostat = io_stat, iomsq = io_msg )
165
166    write ( io_rcs, iostat = io_stat, iomsq = io_msg ) me % rcs_table ( 1 : me % meshFreeAngle % numMeshElements, &
167                                         1 : me % meshFrequency % numMeshElements )
168    call iostat_check_sub ( action = "UNFORMATTED WRITE to", fileName = me % fileRCSbinaryName, crashChain = crashChain, &
169           iostat = io_stat, iomsq = io_msg )
170    call file_closer_sub ( io_unit = io_rcs, fileName = me % fileRCSbinaryName, crashChain = crashChain )
171
172  end subroutine write_rcs_binary_sub
173
174  module subroutine write_rcs_csv_sub ( me )
175    class ( datafile4112 ), target :: me
176    integer ( ip ) :: kFrequency = 0, kFreeAngle = 0, &
177                   io_out = 0
178    character ( len = msgl ) :: crashChain = ""
179
180    crashChain = moduleCrash // "subroutine 'write_rcs_csv_sub'."
181    io_out = safeopen_writereplace ( me % fileRCStxtName )
182    ! write RCS values one row (frequency) at a time
183    sweep_frequencies: do kFrequency = 1, me % meshFrequency % numMeshElements
184      write ( io_out, fmt = me % meshFreeAngle % valuesFormatDescriptor ) ( me % rcs_table ( kFreeAngle, kFrequency ), &
185                                         kFreeAngle = 1, me % meshFreeAngle % numMeshElements )
186    call iostat_check_sub ( action = "WRITE to", fileName = me % fileRCStxtName, crashChain = crashChain, &
187           iostat = io_stat, iomsq = io_msg )
188
189    end do sweep_frequencies
190    ! close io handle
191    call file_closer_sub ( io_unit = io_out, fileName = me % fileRCStxtName, crashChain = crashChain )
192
193  end subroutine write_rcs_csv_sub
194
195  module subroutine write_dBsm_binary_sub ( me )
196    class ( datafile4112 ), target :: me
197    integer ( ip ) :: io_rcs = 0
198    character ( len = msgl ) :: crashChain = ""
199
200    crashChain = moduleCrash // "subroutine write_dBsm_binary_sub"."
201
202    open ( newunit = io_rcs, file = me % filedBsmBinaryName, action = 'WRITE', status = 'REPLACE', form = 'UNFORMATTED', &
203           iostat = io_stat, iomsq = io_msg )
204    call iostat_check_sub ( action = "UNFORMATTED OPENING", fileName = me % fileRCSbinaryName, crashChain = crashChain, &
205           iostat = io_stat, iomsq = io_msg )
206
206    write ( io_rcs, iostat = io_stat, iomsq = io_msg ) me % dBsm_table ( 1 : me % meshFreeAngle % numMeshElements, &
207                                         1 : me % meshFrequency % numMeshElements )
208    call iostat_check_sub ( action = "UNFORMATTED WRITE to", fileName = me % fileRCSbinaryName, crashChain = crashChain, &
209           iostat = io_stat, iomsq = io_msg )
210    call file_closer_sub ( io_unit = io_rcs, fileName = me % filedBsmBinaryName, crashChain = crashChain )
211
212  return
213  end subroutine write_dBsm_binary_sub
214
215  module subroutine write_dBsm_csv_sub ( me )
216    class ( datafile4112 ), target :: me
217    integer ( ip ) :: kFrequency = 0, kFreeAngle = 0, &
218                   io_out = 0
219    character ( len = msgl ) :: crashChain = ""
220
221    crashChain = moduleCrash // "subroutine 'write_dBsm_csv_sub'."
222    io_out = safeopen_writereplace ( me % filedBsmTxtName )
223    ! write RCS values one row (frequency) at a time
224    sweep_frequencies: do kFrequency = 1, me % meshFrequency % numMeshElements
225      write ( io_out, fmt = me % meshFreeAngle % valuesFormatDescriptor ) ( me % dBsm_table ( kFreeAngle, kFrequency ), &
226                                         kFreeAngle = 1, me % meshFreeAngle % numMeshElements )
227    call iostat_check_sub ( action = "WRITE to", fileName = me % filedBsmTxtName, crashChain = crashChain, &

```

```

228      iostat = io_stat, iomsg = io_msg )
229
230      end do sweep_frequencies
231      ! close io handle
232      call file_closer_sub ( io_unit = io_out, fileName = me % fileRCStxtName, crashChain = crashChain )
233
234      return
235
236      subroutine set_file_names_sub ( me, file4112Name )
237          class ( dataFile4112 ), target :: me
238          character ( len = fnl ), intent ( in ) :: file4112Name
239          integer ( ip ) :: nameLength = 0
240
241          nameLength = len ( trim ( file4112Name ) )
242          me % file4112Name = trim ( file4112Name )
243          me % fileRCStxtName = trim ( file4112Name ( 1 : nameLength - 4 ) ) // ".rcs.txt"
244          me % fileRCSbinaryName = trim ( file4112Name ( 1 : nameLength - 4 ) ) // ".rcs.r32"
245          me % filedBsmTxtName = trim ( file4112Name ( 1 : nameLength - 4 ) ) // ".dBsm.txt"
246          me % filedBsmBinaryName = trim ( file4112Name ( 1 : nameLength - 4 ) ) // ".dBsm.r32"
247
248          return
249
250      end subroutine set_file_names_sub
251
252      subroutine allocate_rcsAverages_sub ( me )
253          class ( dataFile4112 ), target :: me
254          call allocate_rank_one_averages_sub ( rank_1_average = me % perFrequencyAverage, &
255                                              index_min = 1, index_max = me % num Frequencies )
256
257      return
258      end subroutine allocate_rcsAverages_sub
259
260      subroutine allocate_rcs_tables_sub ( me )
261          class ( dataFile4112 ), target :: me
262          call me % myKit % allocate_rank_two_reals ( rank_2_real_array = me % rcs_table,
263                                              & dim1_index_min = 1, dim1_index_max = me % numFreeAngles, &
264                                              dim2_index_min = 1, dim2_index_max = me % num Frequencies )
265          call me % myKit % allocate_rank_two_reals ( rank_2_real_array = me % dBsm_table,
266                                              & dim1_index_min = 1, dim1_index_max = me % numFreeAngles, &
267                                              dim2_index_min = 1, dim2_index_max = me % num Frequencies )
268
269      return
270      end subroutine allocate_rcs_tables_sub
271
272      subroutine establish_frequency_mesh_sub ( me )
273          class ( dataFile4112 ), target :: me
274          ! count lines in MoM file (e.g. 14844)
275          call count_lines_sub ( fullFileName = me % file4112Name, numLines = me % numLines4112Text )
276          ! allocate object to hold text of MoM file as a collection of text lines
277          call read_text_lines_sub ( fileName = me % file4112Name, linesText = me % Lines4112Text )
278          ! sift through text lines for " Freq "
279          call me % harvest_frequencies ( )
280
281      return
282      end subroutine establish_frequency_mesh_sub
283
284      ! sweep through character array looking for " Freq "
285      ! store these values in a temporary array until nuMesh is allocated
286      subroutine harvest_frequencies_sub ( me )
287          class ( dataFile4112 ), target :: me
288          ! pointers
289          !character ( len = m1 ), pointer :: p => null ( )
290          type ( meshReal ), pointer :: q => null ( )
291          type ( allocationToolKit ), pointer :: s => null ( )
292          ! temp arrays
293          real ( ip ) :: tempFrequencyValues ( 1 : 500 )
294          integer ( ip ) :: templateNumsFrequency ( 1 : 500 )
295          ! scalars
296          integer ( ip ) :: numfrequencies = 0, kFrequency = 0
297
298          ! find lines containing " Freq "
299          call mark_frequencies_sub ( lines4112Text = me % lines4112Text, &
300                                     numLines4112Text = me % numLines4112Text, &
301                                     tempFrequencyValues = tempFrequencyValues, &
302                                     templateNumsFrequency = templateNumsFrequency, &
303                                     numfrequencies = numfrequencies )
304
305          ! record what we have learned about the mesh
306          q => me % meshFrequency
307          q % numMeshElements = numfrequencies
308          ! allocate data objects
309          call q % allocate_mesh_real ( )
310          s => me % myKit
311          call s % allocate_rank_one_integers ( integer_array = me % lineNumbersFrequency, index_min = 1, &
312                                              index_max = q % numMeshElements )
313          s => null ( )
314          ! move temporary array data into data object
315          do kFrequency = 1, q % numMeshElements
316              q % meshValues ( kFrequency ) = tempFrequencyValues ( kFrequency )
317              me % lineNumbersFrequency ( kFrequency ) = templateNumsFrequency ( kFrequency )
318          end do
319          me % numFrequencies = q % numMeshElements
320          call q % analyze_mesh_values ( )

```

```

320      q => null ( )
321      return
322 end subroutine harvest_frequencies_sub
323
324 subroutine extract_rcs_from_4112_file_sub ( me )
325     class ( dataFile4112 ), target :: me
326     ! locals
327     !real ( rp )           :: sigma = 0._rp
328     integer ( ip )         :: kFrequency = 0, kFreeAngle = 0, linePosition = 0
329     character ( len = mll ) :: textLine
330
331     ! open *.4112.txt file, read text lines into memory
332     call read_text_lines_sub ( fileName = me % file4112Name, linesText = me % lines4112Text )
333     ! sweep and harvest RCS value
334     sweep_frequencies: do kFrequency = 1, me % numFrequencies
335         linePosition = me % lineNumbersFrequency ( kFrequency ) + 8
336         sweep_free_angles: do kFreeAngle = 1, me % numFreeAngles
337             textLine = me % lines4112Text ( linePosition )
338             call me % eFields % gather_mean_total_rcs ( textLine = textLine )
339             me % rcs_table ( kFreeAngle, kFrequency ) = me % eFields % meanTotalRCS
340             me % dBsm_table ( kFreeAngle, kFrequency ) = me % eFields % dBsm
341             linePosition = linePosition + 1
342         end do sweep_free_angles
343     end do sweep_frequencies
344
345     return
346 end subroutine extract_rcs_from_4112_file_sub
347
348 subroutine set_free_angle_elevation_sub ( me )
349     class ( datafile4112 ), target :: me
350     me % angleFreeType = strElevation
351     me % angleFixedType = strAzimuth
352     return
353 end subroutine set_free_angle_elevation_sub
354
355 subroutine set_free_angle_azimuth_sub ( me )
356     class ( datafile4112 ), target :: me
357     me % angleFreeType = strAzimuth
358     me % angleFixedType = strElevation
359     return
360 end subroutine set_free_angle_azimuth_sub
361
362 subroutine establish_free_angle_mesh_sub ( me, angle_min, angle_max, angle_count )
363     class ( datafile4112 ), target :: me
364     real ( rp ), intent ( in ) :: angle_min, angle_max
365     integer ( ip ), intent ( in ) :: angle_count
366
367     me % meshFreeAngle % meshAverage % extrema % minValue = angle_min
368     me % meshFreeAngle % meshAverage % extrema % maxValue = angle_max
369     me % meshFreeAngle % numMeshElements = angle_count
370     me % numFreeAngles = angle_count
371
372     call me % meshFreeAngle % allocate_mesh_real ( )
373     call me % meshFreeAngle % compute_real_mesh_length ( )
374     call me % meshFreeAngle % compute_real_mesh_interval ( )
375     call me % meshFreeAngle % populate_real_mesh ( )
376     call me % meshFreeAngle % populate_integer_mesh ( )
377
378     return
379 end subroutine establish_free_angle_mesh_sub
380
381 subroutine check_rcs_table_structure_sub ( me )
382     class ( datafile4112 ), target :: me
383     write ( * , * )
384     write ( * , fmt = '( g0 )' ) "# Dimensions for RCS data container # #"
385     write ( * , * )
386     write ( * , fmt = '( g0 )' ) "# Expected dimensions:"
387     write ( * , fmt = '( 2g0 )' ) "# Number of radar frequencies scanned by MoM: ", me % numFrequencies
388     write ( * , fmt = '( 4g0 )' ) "# Number of ", me % angleFreeType, " angles scanned by MoM: ", me % numFreeAngles
389     write ( * , * )
390     write ( * , fmt = '( 2g0 )' ) "# Container MoM 4112.txt file: rcs_table"
391     write ( * , fmt = '( 6g0 )' ) "# Free angle dimension = ", size ( me % rcs_table, 1 ), &
392                                " indices run from ", lbound ( me % rcs_table, 1 ), &
393                                " to ", ubound ( me % rcs_table, 1 )
394     write ( * , fmt = '( 6g0 )' ) "# Frequency dimension = ", size ( me % rcs_table, 2 ), &
395                                " indices run from ", lbound ( me % rcs_table, 2 ), &
396                                " to ", ubound ( me % rcs_table, 2 )
397     write ( * , * )
398     return
399 end subroutine check_rcs_table_structure_sub
400
401 end module mClassDataFile

```

C Mercury Method of Moments: Distribution and Rights

C.1 Distribution Letter for Software

The subsequent distribution letter was signed by Randy J. Petyak of the NASA Software Release Authority and describes terms for distribution, Government rights, and the ITAR status of the software.

December 11, 2019

Air Force Research Laboratory
RVB
3550 Aberdeen Ave SE
Kirtland Air Force Base, NM 87117-5776
Attn: Mr. Nelson Bonito

Subject: Transmittal of Mercury MoM version 4.1.12, MM_Viz Code.

This distribution letter details the terms for distribution, the Government rights in the software, and the ITAR status of the software. The software usage agreement you signed covers Mercury MoM and MMViz executable codes on both Linux 64 bit and Windows 64 bit. The Mercury MoM software is copyrighted by Matrix Compression, LLC. of which the Government retains certain rights to the software, and must be controlled as outlined in the signed Software Usage Agreement.

NASA furnishes this software under the condition that no further dissemination of the software shall be made without prior written permission of the NASA Langley Research Center. Additionally, this software has been designated as ITAR and needs appropriate protection while on the DVD or on an installed machine.

Note: The software falls under the purview of the U.S. Munitions List (USML), as defined in the International Traffic in Arms Regulations (ITAR), 22 CFR 120-130, and is export controlled. It shall not be taken out of the U.S. nor transferred to foreign nationals in the U.S. or abroad, without specific approval of a knowledgeable export control official, and/or unless an export license/license exemption is obtained/available from the United States Department of State. Violation of these regulations is punishable by fine, imprisonment, or both.

We are interested in your use of this software and the results you obtain. Please include us on your mailing list for any publications that may result from your use of this code.

If you have any additional questions related to your request, please contact me.


Randy J. Petvak
NASA Software Release Authority
(202) 358-4387

C.2 Copyright Statement by the Author

=====
MERCURY MOM(TM) (Copyrighted and Patents Issued)
MATRIX COMPRESSION TECHNOLOGIES, LLC

For licensing information contact:
John Shaeffer
3278 Hunterdon Way
Marietta, Georgia 30067
770.952.3678
Copyright 2006 Matrix Compression Technologies, LLC.

This software was developed under NASA Contracts NAS1-02057, NAS1-02117, NNL08AA00B, and NNL13AA08B, and the U.S. Government retains certain rights.

The Government, and others acting on its behalf, retain a paid-up, nonexclusive, irrevocable, worldwide license to reproduce, prepare derivative works, and perform publicly and display publicly (but not to distribute copies to the public) by or on behalf of the Government, without any obligation of confidentiality on the part of the U.S. Government. Such license extends to use by NASA contractors, and others working under agreements with the U.S. Government; provided that use of the software shall not be allowed to any person or entity where such use is not in direct performance of a contract with the United States; and provided that such use is not for internal research and development by the contractor or others that is not directly funded by the United States.

=====

C.3 Legal Statement

MERCURY MOM™
Copyrighted
US Patents: 7,742,886; 7,844,407; 8,209,138; 8,725,464

Copyright 2006 Matrix Compression Technologies, LLC.

This software was developed under NASA Contracts NAS1-02057, NAS1-02117, NNL08AA00B, and NNL13AA08B, and the U.S. Government retains certain rights.

The Government, and others acting on its behalf, retain a paid-up, nonexclusive, irrevocable, worldwide license to reproduce, prepare derivative works, and perform publicly and display publicly (but not to distribute copies to the public) by or on behalf of the Government, without any obligation of confidentiality on the part of the U.S. Government. Such license extends to use by NASA contractors, and others working under agreements with the U.S. Government; provided that use of the software shall not be allowed to any person or entity where such use is not in direct performance of a contract with the United States; and provided that such use is not for internal research and development by the contractor or others that is not directly funded by the United States.

Matrix Compression Technologies, L.L.C. expressly disclaims any and all warranties, including the warranty of non-infringement, the warranty of merchantability, and the warranty of fitness for a particular purpose. Matrix Compression Technologies, L.L.C. shall not be obligated to indemnify or pay any party for consequential damages or any other damages arising from the use of the MERCURY MOM™ software. Non-U.S. Government entities shall not distribute the MERCURY MOM™ software to any third party without the express written permission of Matrix Compression Technologies, L.L.C.

MATRIX COMPRESSION TECHNOLOGIES, LLC
John Shaeffer
3278 Hunterdon Way
Marietta, Georgia 30067
john@shaeffer.com
770.952.3678

NASA ITAR notice:

Note: The enclosed software falls under the purview of the U.S. Munitions List (USML), as defined in the International Traffic in Arms Regulations (ITAR), 22 CFR 120-130, and is export controlled. It shall not be taken out of the U.S. nor transferred to foreign nationals in the U.S. or abroad, without specific approval of a knowledgeable export control official, and/or unless an export license/license exemption is obtained/available from the United States Department of State. Violation of these regulations is punishable by fine, imprisonment, or both.

C.4 Obtaining Software and Documentation

For more information regarding this document contact the following:

Kam W. Hom
NASA
Langley Research Center
Mail Stop 207
Hampton, Virginia 23681-2199
757-864-9608
kam.w.hom@nasa.gov

or

Jeffrey A. Miller, PhD
NASA
Langley Research Center
Mail Stop 207
Hampton, Virginia 23681-2199
757-864-9611
jeffrey.allen.miller@nasa.gov

Figure 5: Contact information to request Mercury MoM Software and Documentations

C.5 Distribution Contents

C.5.1 Executables

1. Linux 64-bit
2. Windows 64-bit

C.5.2 Documentation

The distribution includes four documents in PDF which are marked as CUI:

1. User's Guide
2. Pill Tutorial
3. Code Validation Report
4. Benchmark Tests

D Using Python to Create Spreadsheets

Some users preferred to digest the radar data in spreadsheet form. The Python toolkit `xlswriter` simplifies moving the data into `*.xlsx` form.

The spreadsheets eschew row-column notation (e.g. `A4`) and makes use of variables and named ranges to simplify the interpretation of the results.

D.1 Inputs

D.1.1 Main

Main module:

```
1  #! /usr/bin/python3
2
3  # # Daniel Topa
4
5  # # Excel tools
6  # xl_new_workbook( workbook_title )
7  # xl_sheet_requirements( this_workbook )
8  # xl_sheet_generate( this_workbook, title_sheet )
9  # xl_s( this_workbook )
10 # xl_sheet_header_footer( this_worksheet )
11
12 # # imports
13 import os           # probe, change directories
14 import sys          # python version
15 import datetime     # https://stackoverflow.com/questions/415511/how-to-get-the-current-time-in-python
16 import numpy as np
17 import pandas as pd
18 import xlsxwriter    # API for Excel
19 from xlsxwriter.utility import xl_rowcol_to_cell
20 import numpy as np
21 import pandas as pd
22
23 import cls_TestObject
24
25 # # modules
26 def xl_new_workbook( testObject ):
27
28     MoMResults = xlsxwriter.Workbook( testObject.outputFile )
29     print( "output file %s" % testObject.outputFile )
30     print( "source file %s" % testObject.sourceFile )
31
32     xl_sheet_master( MoMResults, testObject ) # MoM summary
33     xl_add_data_sheets( MoMResults, testObject ) # MoM summary
34     xl_sheet_provenance( MoMResults ) # provenance sheet
35
36     return MoMResults;
37
38 # ===
39
40 def xl_add_data_sheets( this_workbook, testObject ):
41
42     format_MoM_title = this_workbook.add_format( )
43     format_MoM_title.set_bold( )
44     format_MoM_title.set_font_color( "red" )
45
46     format_MoM_head = this_workbook.add_format( )
47     format_MoM_head.set_bold( )
48
49     format_MoM_polarization = this_workbook.add_format( )
50     format_MoM_polarization.set_bold( )
51
52     number_format = this_workbook.add_format({ 'num_format': '#,##0.000' })
53
54     # https://xlsxwriter.readthedocs.io/format.html#set_center_across
55     cell_format = this_workbook.add_format()
56     cell_format.set_center_across()
57
58     for index in range( 1, 29 ):
59         # add sheet and tag header and footer
60         title = str( index + 2 ) + ' MHz'
61         print( 'adding sheet %s' % title )
62         s = xl_sheet_generate( this_workbook, title )
63         xl_sheet_header_footer( s )
64         s.write( "A1", "MoM 4.1.12 output (*.dat)", format_MoM_title )
65
66         s.write( "A3", "azimuth, ", format_MoM_head )
67         s.write( "B3", "HH, dBsm", format_MoM_head )
68         s.write( "C3", "VV, dBsm", format_MoM_head )
69         s.write( "D3", "HV, dBsm", format_MoM_head )
70         s.write( "E3", "VH, dBsm", format_MoM_head )
71
72         s.write( "H3", "mean", format_MoM_head )
73         s.write( "J3", "standard deviation", format_MoM_head )
74
75         s.write( "G4", "HH", format_MoM_polarization )
76         s.write( "G5", "VV", format_MoM_polarization )
77
78     #     AttributeError: 'str' object has no attribute '_get_xf_index'
79     #     s.write( "I4", "HII", u"\u00B1" )
80     #     s.write( "I4", "\u00B1", cell_format )
81     #     s.write( "I5", "\u00B1", cell_format )
82     #     s.set_column( "I:I", 3 )
83
84     # = AVERAGE( B5:B364 )
85     # = STDEV( B5:B364 )
```

```

86     s.write( "H4", '= AVERAGE( B5:B364)', number_format )
87     s.write( "H5", '= AVERAGE( C5:C364)', number_format )
88     s.write( "J4", '= STDEV( B5:B364 )', number_format )
89     s.write( "J5", '= STDEV( B5:B364 )', number_format )
90
91     # read in data file
92     filename = './data/sphere-005-' + testObject.resolution + '-' + str( index + 2 ).zfill(2) + '.4112.dat.txt'
93     s.write_string( "D1", filename )
94     data = pd.read_csv( filename, delimiter=r"\s+", header = None )
95     data_np = data.values
96     row = 3
97     col = 0
98     for line in range( 0, len( data_np ) ):
99         cell = xl_rowcol_to_cell( ( row, col ) )
100        s.write( row, col, data_np[ line ][ 0 ], number_format )
101        s.write( row, col + 1, data_np[ line ][ 1 ], number_format )
102        s.write( row, col + 2, data_np[ line ][ 2 ], number_format )
103        s.write( row, col + 3, data_np[ line ][ 3 ], number_format )
104        s.write( row, col + 4, data_np[ line ][ 4 ], number_format )
105        row += 1
106
107    return
108
109 # === #
110
111 def xl_sheet_generate( this_workbook, title_sheet ):
112
113     # insure every worksheet has a header and footer
114     mySheet = this_workbook.add_worksheet( title_sheet )
115     xl_sheet_header_footer( mySheet )
116
117     return mySheet;
118
119 # === #
120
121 def xl_sheet_provenance( this_workbook ):
122
123     # Define some global names.
124     this_workbook.define_name( 'c_', '=299792458' )
125     # forensic info
126     s = xl_sheet_generate( this_workbook, "provenance" )
127     # # special formats
128     # https://xlsxwriter.readthedocs.io/format.html?highlight=bold
129
130     # method 1
131     # setting the property as a dictionary of key/value pairs in the constructor
132     format_title = this_workbook.add_format( {} )
133     format_title.set_bold( )
134     format_title.set_font_color( "blue" )
135
136     # method 2
137     # passing a dictionary of properties to the add_format() constructor
138     format_time = this_workbook.add_format( { 'num_format': 'yy/mm/dd hh:mm' } ) # https://xlsxwriter.readthedocs.io/working_with_dates_and_time.html
139
140     # widen first columns
141     s.set_column( "A:A", 15 )
142     s.set_column( "B:B", 13 )
143
144     # https://xlsxwriter.readthedocs.io/worksheet.html
145     s.write_url( "A1", "https://en.wikipedia.org/wiki/Computational_electromagnetics", string = "Radar Cross Section Measurements" )
146
147     # # provenance
148     s.write( "A3", "Workbook created by", format_title )
149     s.write( "A1", tip, "boo" )
150
151     # python notebook which creates workbook
152     s.write( "A4", "python source" )
153     s.write( "B4", os.path.basename( __file__ ) ) # charlie.py
154
155     # current working directory
156     s.write( "A5", "directory" )
157     s.write( "B5", os.getcwd( ) ) # /Volumes/Tlaltecuhtli/repos/GitHub/topa-development/python/xlsx
158
159     # python version
160     s.write( "A6", "python version" )
161     s.write( "B6", sys.version ) # "3.7.0 (default, Jun 28 2018, 07:39:16) [Clang 4.0.1 (tags/RELEASE_401/final)]"
162
163     # # environment variables
164     # practise row, col notation
165     col = 0 # starting column
166     row = 7 # starting row
167     s.write( row, col, "Environment variables", format_title ); row += 1
168
169     s.write( row, col, "$USER" ) # 1127914
170     s.write( row, col + 1, os.environ[ "USER" ] ); row += 1
171
172     s.write( row, col, "$HOSTNAME" ) # Cauchy.Schwarz
173     s.write( row, col + 1, os.environ[ "HOSTNAME" ] ); row += 1
174
175     s.write( row, col, "$HOME" ) # /Users/1127914
176     s.write( row, col + 1, os.environ[ "HOME" ] ); row += 1
177

```

```

178     s.write( row, col, "timestamp" ) # 11/21/18 16:18
179     s.write( row, col + 1, datetime.datetime.now(), format_time ); row += 1
180
181     # # Excel info routines
182     # https://xlsxwriter.readthedocs.io/working_with_formulas.html
183
184     row += 1 # jump
185     s.write( row, col, "XL info function", format_title ); row += 1
186
187     s.write( row, col, "platform" ) # mac
188     s.write_formula( row, col + 1, '= INFO( "system" )' ); row += 1
189
190     s.write( row, col, "recalculation mode" ) # Automatic
191     s.write_formula( row, col + 1, '= INFO( "recalc" )' ); row += 1
192
193     s.write( row, col, "active sheets" ) # 1
194     s.write_formula( row, col + 1, '= INFO( "numfile" )' ); row += 1
195
196     s.write( row, col, "cursor" ) # $A:$A$1
197     s.write_formula( row, col + 1, '= INFO( "origin" )' ); row += 1
198
199     s.write( row, col, "XL release" ) # 16.16
200     s.write_formula( row, col + 1, '= INFO( "release" )' ); row += 1
201
202     s.write( row, col, "application directory" ) # /Users/dantopa/Library/Containers/com.microsoft.Excel/Data/Documents/
203     s.write_formula( row, col + 1, '= INFO( "directory" )' ); row += 1
204
205     s.write( row, col, "operating systems" ) # Macintosh (Intel) Version 10.13.3 (Build 17D47)
206     s.write_formula( row, col + 1, '= INFO( "osversion" )' ); row += 1
207
208     return
209
210 # === ===== #
211
212 def xl_sheet_header_footer( this_worksheet ):
213
214     # header: sheet name (center)
215     # footer: date/time, page number, path/file
216
217     myheader = "&C&12&A" # fontsize 12
218     myfooter = "&L&8&T\n&8&D" + "&C &P / &N" + "&R&8&Z\n&8&F" # fontsize 8
219
220     this_worksheet.set_header( myheader )
221     this_worksheet.set_footer( myfooter )
222
223     return
224
225 # === ===== #
226
227 def xl_sheet_master( this_workbook, testObject ):
228
229     number_format = this_workbook.add_format({'num_format': '#,##0.000'})
230
231     masterRow = 0
232     masterCol = 0
233     xl_set_label_column( this_workbook, testObject, masterRow, masterCol )
234
235     dataRow = 8
236     dataCol = 0
237     s = this_workbook.get_worksheet_by_name( testObject.masterSheet )
238     for index in range(1, 29):
239         dataCol += 1
240         nu = index + 2
241         xl_computation( s, dataRow, dataCol, nu, number_format )
242
243     return
244
245 # === ===== #
246
247 # https://xlsxwriter.readthedocs.io/working_with_cell_notation.html
248 def xl_computation( wsheet, row, col, nu, number_format ):
249
250     # frequency
251     wsheet.write_number( row, col, nu )
252
253     # wavelength = c_ / ( B11 * 1000000 )
254     cell = xl_rowcol_to_cell( row, col )
255     wsheet.write( row + 1, col, '= c_ / ( ' + cell + ' * 1000000 )', number_format ); row += 1
256
257     # = radius / wavelength
258     cell = xl_rowcol_to_cell( row, col )
259     wsheet.write( row + 1, col, '= radius / ' + cell, number_format ); row += 3
260
261     # MoM average dBsm = '30 MHz'!$H4
262     wsheet.write_formula( row, col, "= " + str( nu ) + " MHz'!$H$4", number_format ); row += 1
263     # relative error dBsm
264     cell = xl_rowcol_to_cell( row - 1, col )
265     wsheet.write_formula( row, col, '= 1 - size_optical_dbsm / ' + cell, number_format ); row += 2
266
267     # rcs, sq m = 10^( B15 / 10 )
268     cell = xl_rowcol_to_cell( row - 3, col )
269     wsheet.write_formula( row, col, '= 10^( ' + cell + ' / 10 )', number_format ); row += 1

```

```

270     # rel_error (sq_m) = 1 - size_optical_sq_m / B18
271     cell = xl_rowcol_to_cell( row - 1, col )
272     wsheet.write_formula( row, col, '= 1 - size_optical_sq_m / ' + cell, number_format )
273
274     return
275
276 # == == == == == == == == == == #
277
278 def xl_set_label_column( wbook, testObject, row, col ):
279
280     # method 1
281     # setting the property as a dictionary of key/value pairs in the constructor
282     format_title = wbook.add_format()
283     format_title.set_bold()
284     format_title.set_font_color("blue")
285
286     format_label = wbook.add_format()
287     format_label.set_bold()
288
289     # https://xlsxwriter.readthedocs.io/example_defined_name.html
290     # https://docs.python.org/2.0/ref/strings.html
291     wbook.define_name('c_1', '#299792458')
292     #string = '\=' + str( testObject.sizeValue / 2 ) + '\'
293     #print( 'string = %s' % string )
294     wbook.define_name('radius1', '=5')
295     wbook.define_name('size_optical_sq_m', '=\\' + testObject.masterSheet + '\\$B$6')
296     wbook.define_name('size_optical_dbsm', '=\\' + testObject.masterSheet + '\\$B$7')
297
298     # sheet operations
299     s = xl_sheet_generate( wbook, testObject.masterSheet )
300     s.set_first_sheet()
301
302     # widen first columns
303     s.set_column( "A:A", 17 )
304     s.set_column( "B:B", 10 )
305
306     # column of labels
307     s.write_string( row, col, 'INPUT', format_title ); row += 2
308
309     s.write( row, col, 'MoM output:', format_label )
310     s.write( row, col + 1, testObject.sourceFile ); row += 2
311
312     s.write( row, col, testObject.sizeName, format_label );
313     s.write( row, col + 1, testObject.sizeValue )
314     s.write( row, col + 2, 'm' ); row += 1
315
316     s.write( row, col, 'optical size', format_label )
317     s.write( row, col + 1, 'pi() * radius^2' )
318     s.write_string( row, col + 2, testObject.areaUnits ); row += 1
319     s.write_formula( row, col + 1, '= 10 * LOG10( size_optical_sq_m )' );
320     s.write( row, col + 2, 'dB area' ); row += 2
321
322     s.write( row, col, 'frequency (MHz)', format_label ); row += 1
323     s.write( row, col, 'wavelength (m)', format_label ); row += 1
324     s.write( row, col, 'radius / lambda', format_label ); row += 2
325
326     s.write( row, col, 'MoM average (dbSm)', format_label ); row += 1
327     s.write( row, col, 'rel error (dbSm)', format_label ); row += 2
328
329     s.write( row, col, 'rcs, sq m', format_label ); row += 1
330     s.write( row, col, 'rel error (sq m)', format_label )
331
332     xl_sheet_header_footer( s )
333
334
335     return
336
337 # root@f21d93a5a2e9:sphere $ python tools_xl.py
338 #
339 # root@f21d93a5a2e9:sphere $ date
340 # Wed Jun 24 01:19:38 MDT 2020
341 #
342 # root@f21d93a5a2e9:sphere $ pwd
343 # /Tlaloc/python/sphere
344

```

D.1.2 Class Test Object

Radar return data.

```

1  #!/usr/bin/python3
2
3  # # Daniel Topa
4
5  # imports
6  import math          # pi
7  import uuid          # Universal Unique IDentifier
8  #from pathlib import Path  # rename file
9

```

```

10  class TestObject( object ):
11      def __init__( self ):
12
13          self._descriptor     = None    # sphere
14          self._sizeName       = None    # diameter
15          self._sizeValue      = None    # 10
16          self._sizeUnits      = None    # m
17          self._areaValue      = None    # pi r^2
18          self._areaUnits      = None    # m^2
19          self._resolution     = None    # 04
20          self._mastersheet    = None    # sphere, d = 10 m
21          self._sourceFile      = None    # *.dat
22          self._sourcePath      = None    # absolute path to *.dat
23          self._sourcePathFile  = None    # path + source file name
24          self._outputfile      = None    # *.xlsx
25          self._outputPath      = None    # absolute path to *.xlsx
26          self._outputPathFile  = None    # path + *.xlsx
27          self._uuid            = uuid.uuid4( ) # de facto time stamp
28
29 #  P R O P E R T I E S   #
30
31     @property
32     def descriptor( self ):
33         """Descriptor (sphere, cube, etc.)"""
34         return self._descriptor
35
36     @property
37     def sizeName( self ):
38         """Name of size parameter (edge, radius, etc.)"""
39         return self._sizeName
40
41     @property
42     def sizeValue( self ):
43         """Length parameter"""
44         return self._sizeValue
45
46     @property
47     def sizeUnits( self ):
48         """Units (m, mm, etc.)"""
49         return self._sizeUnits
50
51     @property
52     def areaValue( self ):
53         """Area"""
54         return self._areaValue
55
56     @property
57     def areaUnits( self ):
58         """Area units (m^2, mm, etc.)"""
59         return self._areaUnits
60
61     @property
62     def masterSheet( self ):
63         """Name of master sheet"""
64         return self._masterSheet
65
66     @property
67     def sourcePath( self ):
68         """Path (absolute) to source file"""
69         return self._sourcePath
70
71     @property
72     def sourceFile( self ):
73         """Path + Name for input file"""
74         return self._sourceFile
75
76     @property
77     def outputFile( self ):
78         """Name of output file"""
79         return self._outputFile
80
81     @property
82     def outputPath( self ):
83         """Path (absolute) to output file"""
84         return self._outputPath
85
86     @property
87     def outputPathFile( self ):
88         """Path + Name for output file"""
89         return self._outputPathFile
90
91     @property
92     def uuid( self ):
93         """Universal unique identifier: connects requirements to source document"""
94         return self._uuid
95
96 #  S E T T E R S   #
97
98     @descriptor.setter
99     def descriptor( self, value ):
100        self._descriptor = value
101

```

```

102     @sizeName.setter
103     def sizeName( self, value ):
104         self._sizeName = value
105
106     @sizeValue.setter
107     def sizeValue( self, value ):
108         self._sizeValue = value
109
110     @sizeUnits.setter
111     def sizeUnits( self, value ):
112         self._sizeUnits = value
113
114     @areaValue.setter
115     def areaValue( self, value ):
116         self._areaValue = value
117
118     @areaUnits.setter
119     def areaUnits( self, value ):
120         self._areaUnits = value
121
122     @masterSheet.setter
123     def masterSheet( self, value ):
124         self._masterSheet = value
125
126     @sourcePath.setter
127     def sourcePath( self, value ):
128         self._sourcePath = value
129
130     @sourceFile.setter
131     def sourcefile( self, value ):
132         self._sourceFile = value
133
134     @outputFile.setter
135     def outputFile( self, value ):
136         self._outputFile = value
137
138     @outputPath.setter
139     def outputPath( self, value ):
140         self._outputPath = value
141
142     @outputPathFile.setter
143     def outputPathFile( self, value ):
144         self._outputPathFile = value
145
146 #  D E L E T E R S  #
147
148     @descriptor.deleter
149     def descriptor( self ):
150         del self._descriptor
151
152     @sizeName.deleter
153     def sizeName( self ):
154         del self._sizeName
155
156     @sizeValue.deleter
157     def sizeValue( self ):
158         del self._sizeValue
159
160     @sizeUnits.deleter
161     def sizeUnits( self ):
162         del self._sizeUnits
163
164     @areaValue.deleter
165     def areaValue( self ):
166         del self._areaValue
167
168     @areaUnits.deleter
169     def areaUnits( self ):
170         del self._areaUnits
171
172     @masterSheet.deleter
173     def masterSheet( self ):
174         del self._masterSheet
175
176     @sourcePath.deleter
177     def sourcePath( self ):
178         del self._sourcePath
179
180     @sourceFile.deleter
181     def sourcefile( self ):
182         del self._sourceFile
183
184     @outputFile.deleter
185     def outputFile( self ):
186         del self._outputFile
187
188     @outputPath.deleter
189     def outputPath( self ):
190         del self._outputPath
191
192     @outputPathFile.deleter
193     def outputPathFile( self ):

```

```

194     del self._outputPathFile
195
196     @uuid.deleter
197     def uuid( self ):
198         del self._uuid
199
200 # M E T H O D S #
201
202     def print_attributes( self ):
203         print( '\nSource attributes:' )
204         print( 'descriptor = %s' % self.descriptor )
205         print( 'sizeName = %s' % self.sizeName )
206         print( 'sizeValue = %s' % self.sizeValue )
207         print( 'sizeUnits = %s' % self.sizeUnits )
208         print( 'sourcePath = %s' % self.sourcePath )
209         print( 'sourceFile = %s' % self.sourceFile )
210         print( 'sourcePathFile = %s' % self.sourcePathFile )
211         print( 'outputFile = %s' % self.outputFile )
212         print( 'outputPath = %s' % self.outputPath )
213         print( 'outputPathFile = %s' % self.outputPathFile )
214         print( 'uuid = %s' % self.uuid )
215
216     return
217 # === #
218
219     def scenario( self ):
220         self.setup_io( ) # establish output file
221         #self.read_MoM_file( )
222         self.area_circular( ) # compute area for given geometry
223         return
224
225 # === #
226
227     def read_MoM_file( self ):
228         ## read source file
229         print( "reading source file %s" % self.sourceFile )
230         # https://stackoverflow.com/questions/3277503/in-python-how-do-i-read-a-file-line-by-line-into-a-list
231         with open( self.sourceFile ) as f:
232             self.col_lines = f.read( ).splitlines( )
233             self.numLines = len( self.col_lines )
234         return
235
236 # === #
237
238     def setup_io( self ):
239         # combine path and file name
240         self.sourcePathFile = self.sourcePath + self.sourceFile
241         self.outputPathFile = self.outputPath + self.outputFile
242         self.masterSheet = self.descriptor + ', ' + self.sizeName[0] + ' = ' + str( self.sizeValue ) + ' ' + self.sizeUnits
243         return
244
245 # === #
246
247     def area_circular( self ):
248         # combine path and file name
249         self.areaValue = math.pi * ( self.sizeValue / 2 )**2
250         return
251
252 # === #
253

```

D.1.3 Excel Details

Toolkit for writing to spreadsheets.

```

1  #! /usr/bin/python3
2
3 # # Daniel Topa
4
5 # # Excel tools
6 # xl_new_workbook( workbook_title )
7 # xl_sheet_requirements( this_workbook )
8 # xl_sheet_generate( this_workbook, title_sheet )
9 # xl_s( this_workbook )
10 # xl_sheet_header_footer( this_worksheet )
11
12 # # imports
13 import os           # probe, change directories
14 import sys          # python version
15 import datetime     # https://stackoverflow.com/questions/415511/how-to-get-the-current-time-in-python
16 import numpy as np
17 import pandas as pd
18 import xlsxwriter    # API for Excel
19 from xlsxwriter.utility import xl_rowcol_to_cell
20 import numpy as np
21 import pandas as pd
22
23 import cls_TestObject
24

```

```

25  # # modules
26  def xl_new_workbook( testObject ):
27
28      MoMresults = xlswriter.Workbook( testObject.outputFile )
29      print( "output file %s" % testObject.outputFile )
30      print( "source file %s" % testObject.sourceFile )
31
32      xl_sheet_master( MoMresults, testObject ) # MoM summary
33      xl_add_data_sheets( MoMresults, testObject ) # MoM summary
34      xl_sheet_provenance( MoMresults ) # provenance sheet
35
36      return MoMresults;
37
38  # === === === === === === === === #
39
40  def xl_add_data_sheets( this_workbook, testObject ):
41
42      format_MoM_title = this_workbook.add_format( )
43      format_MoM_title.set_bold( )
44      format_MoM_title.set_font_color( "red" )
45
46      format_MoM_head = this_workbook.add_format( )
47      format_MoM_head.set_bold( )
48
49      format_MoM_polarization = this_workbook.add_format( )
50      format_MoM_polarization.set_bold( )
51
52      number_format = this_workbook.add_format({'num_format': '#,##0.000'})
53
54      # https://xlsxwriter.readthedocs.io/format.html#set_center_across
55      cell_format = this_workbook.add_format()
56      cell_format.set_center_across()
57
58      for index in range( 1, 29 ):
59          # add sheet and tag header and footer
60          title = str( index + 2 ) + ' MHz'
61          print( 'adding sheet %s' % title )
62          s = xl_sheet_generate( this_workbook, title )
63          xl_sheet_header_footer( s )
64          s.write( "A1", "MoM 4.1.12 output (*.dat)", format_MoM_title )
65          #
66          s.write( "A3", "azimuth, °", format_MoM_head )
67          s.write( "B3", "HH, dBsm", format_MoM_head )
68          s.write( "C3", "VV, dBsm", format_MoM_head )
69          s.write( "D3", "HV, dBsm", format_MoM_head )
70          s.write( "E3", "VH, dBsm", format_MoM_head )
71          #
72          s.write( "H3", "mean", format_MoM_head )
73          s.write( "J3", "standard deviation", format_MoM_head )
74          #
75          s.write( "G4", "HH", format_MoM_polarization )
76          s.write( "G5", "VV", format_MoM_polarization )
77          #
78          # AttributeError: 'str' object has no attribute '_get_xf_index'
79          #     s.write( "I4", "HH", "\u000B1" )
80          s.write( "I4", '\u000B1', cell_format )
81          s.write( "I5", '\u000B1', cell_format )
82          s.set_column( "I:I", 3 )
83
84          # = AVERAGE( B5:B364 )
85          # = STDEV( B5:B364 )
86          s.write( "H4", '=AVERAGE( B5:B364)', number_format )
87          s.write( "H5", '=AVERAGE( C5:C364)', number_format )
88          s.write( "J4", '=STDEV( B5:B364 )', number_format )
89          s.write( "J5", '=STDEV( B5:B364 )', number_format )
90
91          # read in data file
92          filename = './data/sphere-005-' + testObject.resolution + '-' + str( index + 2 ).zfill(2) + '.4112.dat.txt'
93          s.write_string( "D1", filename )
94          data = pd.read_csv( filename, delimiter=r"\s+", header = None )
95          data_np = data.values
96          row = 3
97          col = 0
98          for line in range( 0, len( data_np ) ):
99              cell = xl_rowcol_to_cell( row, col )
100             s.write( row, col, data_np[ line ][ 0 ], number_format )
101             s.write( row, col + 1, data_np[ line ][ 1 ], number_format )
102             s.write( row, col + 2, data_np[ line ][ 2 ], number_format )
103             s.write( row, col + 3, data_np[ line ][ 3 ], number_format )
104             s.write( row, col + 4, data_np[ line ][ 4 ], number_format )
105             row += 1
106
107         return
108
109  # === === === === === === === === #
110
111  def xl_sheet_generate( this_workbook, title_sheet ):
112
113      # insure every worksheet has a header and footer
114      mySheet = this_workbook.add_worksheet( title_sheet )
115      xl_sheet_header_footer( mySheet )
116

```

```

117     return mySheet;
118 # --- --- --- --- --- --- --- --- #
120
121 def xl_sheet_provenance( this_workbook ):
122
123     # Define some global names.
124     this_workbook.define_name( 'c_', '=299792458' )
125     # forensic info
126     s = xl_sheet_generate( this_workbook, "provenance" )
127     # # special formats
128     # https://xlsxwriter.readthedocs.io/format.html?highlight=bold
129
130     # method 1
131     # setting the property as a dictionary of key/value pairs in the constructor
132     format_title = this_workbook.add_format( )
133     format_title.set_bold( )
134     format_title.set_font_color( "blue" )
135
136     # method 2
137     # passing a dictionary of properties to the add_format() constructor
138     format_time = this_workbook.add_format( { 'num_format': 'yy/mm/dd hh:mm' } ) # https://xlsxwriter.readthedocs.io/working_with_dates_and_time.html
139
140     # widen first columns
141     s.set_column( "A:A", 15 )
142     s.set_column( "B:B", 13 )
143
144     # https://xlsxwriter.readthedocs.io/worksheet.html
145     s.write_url( "A1", "https://en.wikipedia.org/wiki/Computational_electromagnetics", string = "Radar Cross Section Measurements" )
146
147     # # provenance
148     s.write( "A3", "Workbook created by", format_title )
149     s.write( "A1", tip, "boo" )
150
151     # python notebook which creates workbook
152     s.write( "A4", "python source" )
153     s.write( "B4", os.path.basename( __file__ ) ) # charlie.py
154
155     # current working directory
156     s.write( "A5", "directory" )
157     s.write( "B5", os.getcwd() ) # /Volumes/Tlaltecuhli/repos/GitHub/topa-development/python/xlsx
158
159     # python version
160     s.write( "A6", "python version" )
161     s.write( "B6", sys.version ) # "3.7.0 (default, Jun 28 2018, 07:39:16) [Clang 4.0.1 (tags/RELEASE_401/final)]"
162
163     # # environment variables
164     # practise row, col notation
165     col = 0 # starting column
166     row = 7 # starting row
167     s.write( row, col, "Environment variables", format_title ); row += 1
168
169     s.write( row, col, "$USER" ) # 1127914
170     s.write( row, col + 1, os.environ[ "USER" ] ); row += 1
171
172     s.write( row, col, "HOSTNAME" ) # Cauchy.Schwarz
173     s.write( row, col + 1, os.environ[ "HOSTNAME" ] ); row += 1
174
175     s.write( row, col, "$HOME" ) # /Users/1127914
176     s.write( row, col + 1, os.environ[ "HOME" ] ); row += 1
177
178     s.write( row, col, "timestamp" ) # 11/21/18 16:18
179     s.write( row, col + 1, datetime.datetime.now( ), format_time ); row += 1
180
181     # # Excel info routines
182     # https://xlsxwriter.readthedocs.io/working_with_formulas.html
183
184     row += 1 # jump
185     s.write( row, col, "XL info function", format_title ); row += 1
186
187     s.write( row, col, "platform" ) # mac
188     s.write_formula( row, col + 1, '=INFO("system")' ); row += 1
189
190     s.write( row, col, "recalculation mode" ) # Automatic
191     s.write_formula( row, col + 1, '=INFO("recalc")' ); row += 1
192
193     s.write( row, col, "active sheets" ) # 1
194     s.write_formula( row, col + 1, '=INFO("numfile")' ); row += 1
195
196     s.write( row, col, "cursor" ) # $A:$A$1
197     s.write_formula( row, col + 1, '=INFO("origin")' ); row += 1
198
199     s.write( row, col, "XL release" ) # 16.16
200     s.write_formula( row, col + 1, '=INFO("release")' ); row += 1
201
202     s.write( row, col, "application directory" ) # /Users/dantopa/Library/Containers/com.microsoft.Excel/Data/Documents/
203     s.write_formula( row, col + 1, '=INFO("directory")' ); row += 1
204
205     s.write( row, col, "operating systems" ) # Macintosh (Intel) Version 10.13.3 (Build 17D47)
206     s.write_formula( row, col + 1, '=INFO("osversion")' ); row += 1
207
208     return

```

```

209 # === === === === === === === === #
210 # def xl_sheet_header_footer( this_worksheet ):
211 #     # header: sheet name (center)
212 #     # footer: date/time, page number, path/file
213 #
214 #     myheader = "&C&12&A" # fontsize 12
215 #     myfooter = "&L&&T\n&R&D" + "&C &P / &N" + "&R&8&Z\n&8&F" # fontsize 8
216 #
217 #     this_worksheet.set_header( myheader )
218 #     this_worksheet.set_footer( myfooter )
219 #
220 #     return
221 #
222 # === === === === === === === === #
223 # def xl_sheet_master( this_workbook, testObject ):
224 #
225 #     number_format = this_workbook.add_format({'num_format': '#,##0.000'})
226 #
227 #     masterRow = 0
228 #     masterCol = 0
229 #     xl_set_label_column ( this_workbook, testObject, masterRow, masterCol )
230 #
231 #     DataRow = 8
232 #     DataCol = 0
233 #     s = this_workbook.get_worksheet_by_name( testObject.masterSheet )
234 #     for index in range(1, 29):
235 #         DataCol += 1
236 #         nu = index + 2
237 #         xl_computation ( s, DataRow, DataCol, nu, number_format )
238 #
239 #     return
240 #
241 # === === === === === === === === #
242 # https://xlsxwriter.readthedocs.io/working_with_cell_notation.html
243 # def xl_computation ( wsheet, row, col, nu, number_format ):
244 #
245 #     # frequency
246 #     wsheet.write_number ( row, col, nu )
247 #
248 #     # wavelength = c_ / ( B11 * 1000000 )
249 #     cell = xl_rowcol_to_cell ( row, col )
250 #     wsheet.write ( row + 1, col, '=c_ / (' + cell + ' * 1000000 )', number_format ); row += 1
251 #
252 #     # = radius / wavelength
253 #     cell = xl_rowcol_to_cell ( row, col )
254 #     wsheet.write ( row + 1, col, '=radius / ' + cell, number_format ); row += 3
255 #
256 #     # MoM average dBsm = 30 MHz!$H4
257 #     wsheet.write_formula(row, col, "= " + str( nu ) + " MHz!$H$4", number_format ); row += 1
258 #
259 #     # relative error dBsm
260 #     cell = xl_rowcol_to_cell ( row - 1, col )
261 #     wsheet.write_formula ( row, col, '= 1 - size_optical_dbsm / ' + cell, number_format ); row += 2
262 #
263 #     # rcs, sq_m = 10^( B15 / 10 )
264 #     cell = xl_rowcol_to_cell ( row - 3, col )
265 #     wsheet.write_formula ( row, col, '= 10^( ' + cell + ' / 10 )', number_format ); row += 1
266 #
267 #     # rel error (sq_m) = 1 - size_optical_sq_m / B18
268 #     cell = xl_rowcol_to_cell ( row - 1, col )
269 #     wsheet.write_formula ( row, col, '= 1 - size_optical_sq_m / ' + cell, number_format )
270 #
271 #     return
272 #
273 # === === === === === === === === #
274 # def xl_set_label_column( wbook, testObject, row, col ):
275 #
276 #     # method 1
277 #     # setting the property as a dictionary of key/value pairs in the constructor
278 #     format_title = wbook.add_format()
279 #     format_title.set_bold()
280 #     format_title.set_font_color( "blue" )
281 #
282 #     format_label = wbook.add_format()
283 #     format_label.set_bold()
284 #
285 #     format_title = wbook.add_format()
286 #     format_label = wbook.add_format()
287 #
288 #     # https://xlsxwriter.readthedocs.io/example_defined_name.html
289 #     # https://docs.python.org/2.0/ref/strings.html
290 #     wbook.define_name( 'c_', '299792458' )
291 #     string = '\'= ' + str( testObject.sizeValue / 2 ) + '\'
292 #     #print( 'string = %s' % string )
293 #     wbook.define_name( 'radius', '5' )
294 #     wbook.define_name( 'size_optical_sq_m', '=\\' + testObject.masterSheet + '\\$B$6' )
295 #     wbook.define_name( 'size_optical_dbsm', '=\\' + testObject.masterSheet + '\\$B$7' )
296 #
297 #     # sheet operations
298 #     s = xl_sheet_generate( wbook, testObject.masterSheet )
299 #     s.set_first_sheet()
300

```

```

301     # widen first columns
302     s.set_column( "A:A", 17 )
303     s.set_column( "B:B", 10 )
305
306     # column of labels
307     s.write_string( row, col, 'INPUT', format_title ); row += 2
308
309     s.write( row, col, 'MoM output:', format_label )
310     s.write( row, col + 1, testObject.sourceFile ); row += 2
311
312     s.write( row, col, testObject.sizeName, format_label );
313     s.write( row, col + 1, testObject.sizeValue )
314     s.write( row, col + 2, 'm' ); row += 1
315
316     s.write( row, col, 'optical size', format_label )
317     s.write( row, col + 1, '= pi() * radius^2' )
318     s.write_string( row, col + 2, testObject.areaUnits ); row += 1
319     s.write_formula( row, col + 1, '= 10 * LOG10( size_optical_sq_m )' );
320     s.write( row, col + 2, 'dB area' ); row += 2
321
322     s.write( row, col, 'frequency (MHz)', format_label ); row += 1
323     s.write( row, col, 'wavelength (m)', format_label ); row += 1
324     s.write( row, col, 'radius / lambda', format_label ); row += 2
325
326     s.write( row, col, 'MoM average (dBsm)', format_label ); row += 1
327     s.write( row, col, 'rel error (dBsm)', format_label ); row += 2
328
329     s.write( row, col, 'rcs, sq m', format_label ); row += 1
330     s.write( row, col, 'rel error (sq m)', format_label )
331
332     xl_sheet_header_footer( s )
333
334
335     return
336
337 # root@f21d93a5a2e9:sphere $ python tools_xl.py
338 #
339 # root@f21d93a5a2e9:sphere $ date
340 # Wed Jun 24 01:19:38 MDT 2020
341 #
342 # root@f21d93a5a2e9:sphere $ pwd
343 # /Tlaloc/python/sphere
344
345

```

D.2 Outputs

The main module arranges the output data by creating a tab for each separate radar frequency in the range 3 – 30 MHz. Other tabs contain aggregate information and diagnostics.

```

1  #! /usr/bin/python3
2
3  # # Daniel Topa
4
5  # # imports
6  import datetime          # timestamps
7  import os                 # operating system
8  import sys                # python version
9  from pathlib import Path  # rename file
10 import xlsxwriter         # API for Excel
11 import tools_xl           # spreadsheet authoring tools
12 # home brew
13 # classes
14 import cls_TestObject
15
16 # == == == == == == == == == == == == == == ==
17
18 if __name__ == "__main__":
19
20     series = '050'
21     object = cls_TestObject.TestObject()      # instantiate TestObject
22
23     # populate object properties
24     object.descriptor = "sphere"
25     object.sizeName = "diameter"
26     object.sizeValue = 10
27     object.sizeUnits = "m"
28     object.areaUnits = "m^2"
29     object.resolution = "03"
30     object.sourceFile = "sphere-" + series + "-" + object.resolution
31     object.sourcePath = "/Tlaloc/python/sphere/"
32     object.outputFile = 'sphere-d' + series + '-res' + object.resolution + '.xlsx'
33     object.outputPath = "/Tlaloc/python/sphere/"
34     object.setup_io()
35     object.area_circular()
36
37     # container for MoM data and results
38     MoMresults = tools_xl.xls_new_workbook( object )

```

```

38
39     # close MoMresults
40     MoMresults.close( )
41     print( "\n", datetime.datetime.now( ) )
42     print( "source: %s/%s" % ( os.getcwd( ), os.path.basename( __file__ ) ) )
43     print( "python version %s" % sys.version )
44
45 # root@f21d93a5a2e9:sphere $ date
46 # Wed Jun 24 01:20:41 MDT 2020
47 #
48 # root@f21d93a5a2e9:sphere $ pwd
49 # /Tialoc/python/sphere
50 #
51 # root@f21d93a5a2e9:sphere $ python MoM.py
52 # output file RCS-sphere-10.xlsx
53 # source file sphere-050-01
54 # adding sheet 3 MHz
55 # adding sheet 4 MHz
56 # adding sheet 5 MHz
57 # adding sheet 6 MHz
58 # adding sheet 7 MHz
59 # adding sheet 8 MHz
60 # adding sheet 9 MHz
61 # adding sheet 10 MHz
62 # adding sheet 11 MHz
63 # adding sheet 12 MHz
64 # adding sheet 13 MHz
65 # adding sheet 14 MHz
66 # adding sheet 15 MHz
67 # adding sheet 16 MHz
68 # adding sheet 17 MHz
69 # adding sheet 18 MHz
70 # adding sheet 19 MHz
71 # adding sheet 20 MHz
72 # adding sheet 21 MHz
73 # adding sheet 22 MHz
74 # adding sheet 23 MHz
75 # adding sheet 24 MHz
76 # adding sheet 25 MHz
77 # adding sheet 26 MHz
78 # adding sheet 27 MHz
79 # adding sheet 28 MHz
80 # adding sheet 29 MHz
81 # adding sheet 30 MHz
82 #
83 # 2020-06-24 01:20:46.349142
84 # source: /Tialoc/python/sphere/MoM.py
85 # python version 3.7.7 (default, Jun 22 2020, 22:42:46)
86 # [GCC 10.1.0]
87
88 # $ lsb_release -a
89 # LSB Version: :core-4.1-amd64:core-4.1-noarch
90 # Distributor ID: CentOS
91 # Description: CentOS Linux release 7.8.2003 (Core)
92 # Release: 7.8.2003
93 # Codename: Core

```

References

- D. K. Barton and H.R. Ward. *Handbook of Radar Measurement*. New York, NY: Penguin Random House, 1969.
- J Bruder et al. “IEEE standard for letter designations for radar-frequency bands”. In: *IEEE Aerospace & Electronic Systems Society* (2003), pp. 1–3.
- JW Jr Crispin. *Methods of radar cross-section analysis*. Elsevier, 2013.
- Allen E Fuhs. *Radar cross section lectures*. Monterey, California, Naval Postgraduate School, 1982. URL: <https://calhoun.nps.edu/server/api/core/bitstreams/9e69ec48-4628-4243-9f9b-7e879521f7f8/content>.
- Walton C Gibson. *The method of moments in electromagnetics*. Chapman and Hall/CRC, 2021.

- Roger F Harrington. “The method of moments in electromagnetics”. In: *Journal of Electromagnetic waves and Applications* 1.3 (1987), pp. 181–200.
- Eugene F Knott, John F Schaeffer, and Michael T Tulley. *Radar cross section*. SciTech Publishing, 2004.
- Andrei A. Kolosov. *Over the Horizon Radar*. Artech House, 1987. ISBN: 9780890062333. URL: <https://us.artechhouse.com/Over-the-Horizon-Radar-P254.aspx>.
- Cai-Cheng Lu and Chong Luo. “Comparison of iteration convergences of SIE and VSIE for solving electromagnetic scattering problems for coated objects”. In: *Radio Science* 38.2 (2003), pp. 11–1.
- M Madheswaran and P Suresh Kumar. “Estimation of wide band radar cross section (RCS) of regular shaped objects using method of moments (MOM)”. In: *Ictact Journal on Communication Tech-nology* 3.2 (2012), pp. 536–541.
- Peyton Z Peebles. *Radar principles*. John Wiley & Sons, 2007.
- Daniel Topa. *Mercury Method of Moments Adjunct Visualization Tool: Trials and Tribulations*. Tech. rep. ARFL/RVB, Apr. 2020.
- Daniel Topa. *Mercury Method of Moments: AFRL Quick Start Guide*. Tech. rep. AFRL, 2020.
- Daniel Topa. *Radar Cross Section Models for AFCAP Dashboard: Rapid Report 2020-02: Corrected*. Briefing. Mar. 2020.
- Daniel Topa. *Radar Cross Section: Phase 1 Summary Report*. Tech. rep. ARFL/RVB, Apr. 2020.
- Jiade Yuan, Changqing Gu, and Guodong Han. “Efficient generation of method of moments matrices using equivalent dipole-moment method”. In: *IEEE Antennas and Wireless Propagation Letters* 8 (2009), pp. 716–719.