Simulation of Radar Profiles for Satellites Using Mercury Method of Moments

Daniel Topa
daniel.topa@hii-tsd.com

November 15, 2024

Abstract

A brief survey of characterizing the three dimensional radar cross section of satellites.

Contents

1 Precís

1.1 Models of Radar Cross Section

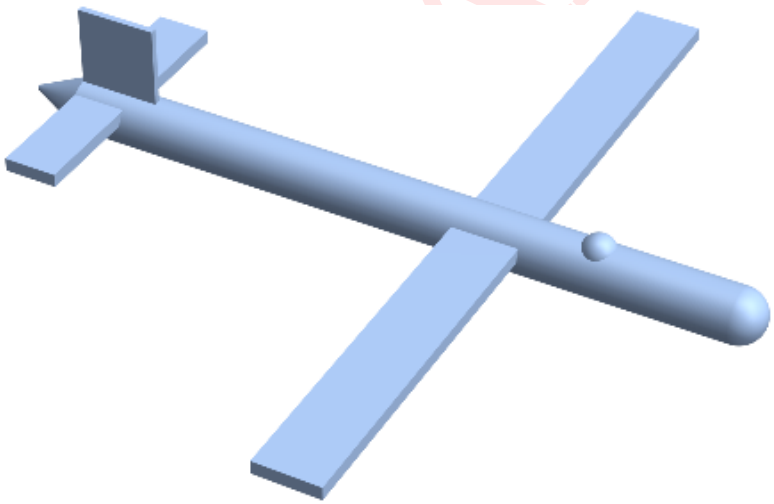


Figure 1: Toy model

Look angle

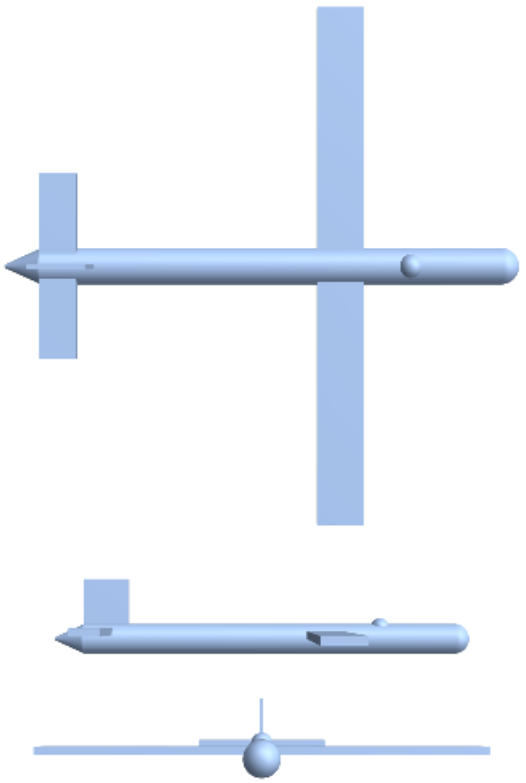
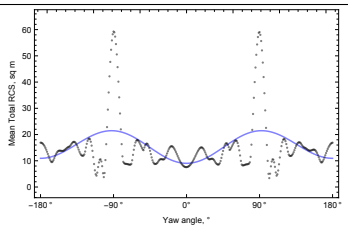
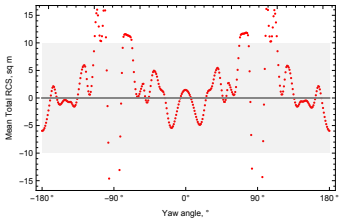
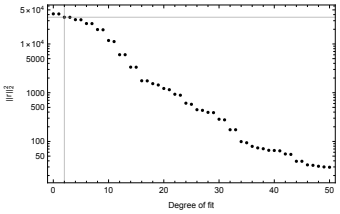
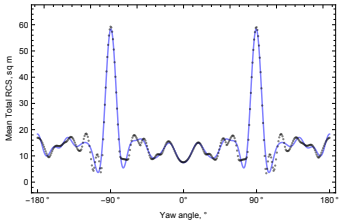
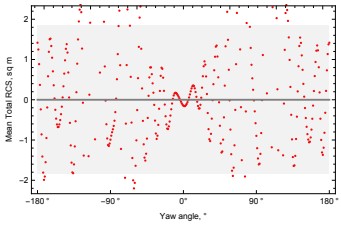
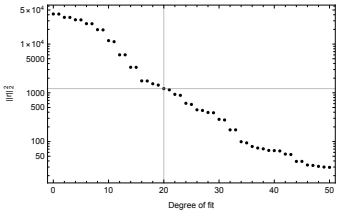
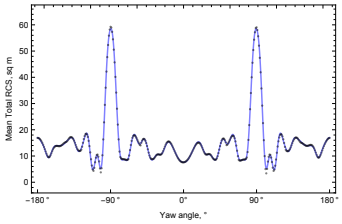
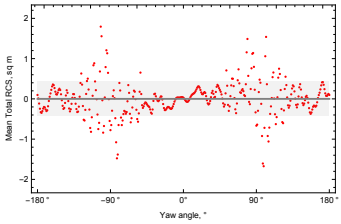
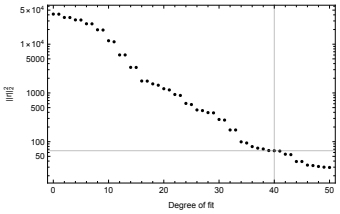


Table 1: Different views present very different areas.

d	Data and Fourier Representation	Residual Fit Error	Total Error for Fit Sequence
$d = 2$			
$d = 20$			
$d = 40$			

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

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

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

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

1.2 Models of Increasing Fidelity

1.3 Running the Code

```
./MMoM_4.1.12 sample.geo
```

1.4 Radar

[**topa20200303**] [**topa20200303**] Working with CAF files, producing output, compressing data. **topa-4-20-2024 topa-4-20-2024**

1.5 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 `materials.lib`	`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 2: Start with a CAD model and construct a Radar Cross Section model

2 Overview: Modeling Radar Cross Section

2.1 Radar

Wave speed equation

$$\lambda\nu = c \tag{2}$$

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 3: IEEE Standard Designations for Radar Bands (**bruder2003ieee**).

(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

2.2 About

(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 File Types

Standard file types

1. `*.obj`
2. `*.txt`

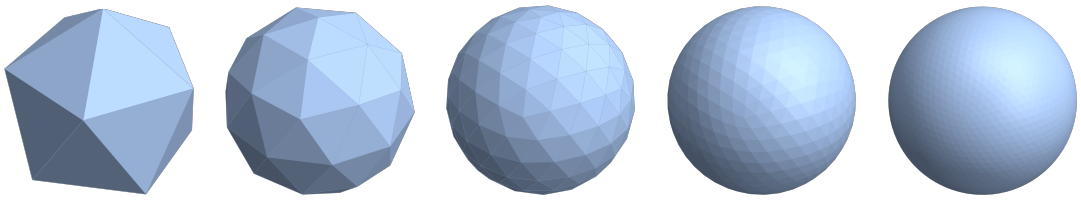
Intrinsic file types

1. `*.geo`
2. `*.facet`

3.1 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



20 80 320 1280 5120

Table 4: A sphere resolved into facets in an `*.obj` file. The application irradiates each facet and aggregates the output.

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

```
# Created with the Wolfram Language : www.wolfram.com
mtllib sphere-d050-01.mtl

# 12 vertex positions
v  13.81966018676758 42.53253936767578 22.36067962646484
v  -13.81966018676758 42.53253936767578 -22.36067962646484
v  -36.18033981323242 26.28655624389648 22.36067962646484
v  -36.18033981323242 -26.28655624389648 22.36067962646484
v  -13.81966018676758 -42.53253936767578 -22.36067962646484
v  13.81966018676758 -42.53253936767578 22.36067962646484
v  36.18033981323242 26.28655624389648 -22.36067962646484
v  -44.72135925292969 0 -22.36067962646484
v  44.72135925292969 0 22.36067962646484
v  36.18033981323242 -26.28655624389648 -22.36067962646484
```

```
v  0 0 50
v  0 0 -50

# 0 UV coordinates

# 0 vertex normals

# Mesh '' with 20 faces
usemtl DefaultMaterial
f  1/ 2/ 3/
f  4/ 5/ 6/
f  2/ 1/ 7/
f  4/ 3/ 8/
f  9/ 6/ 10/
f  3/ 4/ 11/
f  2/ 7/ 12/
f  3/ 2/ 8/
f  2/ 12/ 8/
f  12/ 5/ 8/
f  7/ 1/ 9/
f  5/ 4/ 8/
f  6/ 5/ 10/
f  5/ 12/ 10/
f  12/ 7/ 10/
f  7/ 9/ 10/
f  4/ 6/ 11/
f  6/ 9/ 11/
f  9/ 1/ 11/
f  1/ 3/ 11/
```

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

```
facimusFacet.f08  2020-06-25 11:34:36
    1
<partName>
0
     12
     13.819660      42.532539      22.360680
    -13.819660      42.532539     -22.360680
    -36.180340      26.286556      22.360680
    -36.180340     -26.286556      22.360680
    -13.819660     -42.532539     -22.360680
     13.819660     -42.532539      22.360680
     36.180340      26.286556     -22.360680
    -44.721359       0.000000     -22.360680
     44.721359       0.000000      22.360680
     36.180340     -26.286556     -22.360680
      0.000000       0.000000      50.000000
      0.000000       0.000000     -50.000000
    1
<partName>
      3     20      0      0      0      0      0
      1      2      3      0
      4      5      6      0
      2      1      7      0
      4      3      8      0
      9      6     10      0
      3      4     11      0
      2      7     12      0
      3      2      8      0
      2     12      8      0
     12      5      8      0
      7      1      9      0
      5      4      8      0
      6      5     10      0
      5     12     10      0
     12      7     10      0
      7      9     10      0
      4      6     11      0
```

```
      6     9     11     0
      9     1     11     0
      1     3     11     0
```

3.2 New Efforts

4 Running Mercury Method of Moments

4.1 Inputs

Consider an example with the sphere.

Listing 3: "tabula-rasa.geo"

```
ego

!Mercury MoM input file, VIE/SIE Version 4.x compatible (VIE/Dual Sided SIE)

&MM_MOM
  bUseACA = .TRUE.,
  bSolve_ACA = .TRUE.,
  bOutOfCore = .TRUE.,
  bNormalizeToWaveLength = .FALSE.,
  bNormalize             = .FALSE.,
  dCloseLambda  = 0.100000,
  ACA_Factor_Tol = 0.000010,
  ACA_RHS_Tol = 0.000100,
  Point_Tolerance = 0.001000,
  nLargestBlockSize = 400,
  MemorySize_GB = -1.000000,
  stackSize_GB = -1.000000,
  nFillThreads = -1,
  nFillMKLThreads = 1,
  nLUThreads = -1,
  nLUMKLThreads = 1,
  nRHSThreads = 1,
  nRHSMKLThreads = 1,
  bOutputACAGrouping     = .FALSE.,
  bOutputRankFraction    = .FALSE.,
  bLimitLUColumns        = .FALSE.,
  Lop_Admissibility = WEAK,
  Kop_Admissibility = CLOSE
/

&Scratch_Memory
  Scratch_RankFraction_Z     = 0.300000,
  Scratch_RankFraction_LU    = 0.600000,
  Scratch_RankFraction_RHS   = 0.500000,
  Scratch_RankFraction_Solve = 1.000000,
  MemoryFraction_Z           = 0.950000,
  MemoryFraction_Scratch_LU  = 0.500000,
  MemoryFraction_LU          = 1.000000,
  MemoryFraction_RHS         = 0.500000,
  MemoryFraction_Solve       = 0.900000,
/

&QUADRATURE
  NTRISELF    = 7,
  NTRINEAR    = 3,
  NTRIFAR     = 3,
  NTETSELF    = 11,
  NTETNEAR    = 4,
  NTETFAR     = 4,
  NQGAUSS  = 4,
/
```

```

FREQUENCY
  mhz
  nu-mhz  nu-mhz  1   !Freq Start, Freq Stop, Number of Frequencies

Excitation
  MONOSTATIC

Angle Cut
  1
  0.000000  359.000000  360
  AZIMUTH
  90.000000

Boundary Conditions
PEC-Materials.lib
4
V_FREE_SPACE => Free_Space
V_PEC => PEC
V_PMC => PMC
V_NULL => NULL
1
0 BC_PEC V_FREE_SPACE

SIE
myFacet.facet
m

Geometry_End

! Fiducial run
```

Listing 4: A simple `*.facet` file

```
facimusFacet.f08  2020-06-25 11:34:36
    1
<partName>
0
     12
     13.819660      42.532539      22.360680
    -13.819660      42.532539     -22.360680
    -36.180340      26.286556      22.360680
    -36.180340     -26.286556      22.360680
    -13.819660     -42.532539     -22.360680
     13.819660     -42.532539      22.360680
     36.180340      26.286556     -22.360680
    -44.721359       0.000000     -22.360680
     44.721359       0.000000      22.360680
     36.180340     -26.286556     -22.360680
      0.000000       0.000000      50.000000
      0.000000       0.000000     -50.000000
    1
<partName>
      3     20      0      0      0      0      0
      1      2      3      0
      4      5      6      0
      2      1      7      0
      4      3      8      0
      9      6     10      0
      3      4     11      0
      2      7     12      0
      3      2      8      0
      2     12      8      0
     12      5      8      0
      7      1      9      0
      5      4      8      0
      6      5     10      0
      5     12     10      0
     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
```

5 Additional Information

5.1 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

5.2 Further Reading

Radar rudiments

1. **Handbook**
2. **kolosov1987**
3. **peebles2007radar**

Radar cross section

1. **crispin2013methods**
2. **fuhs1982radar**
3. **knott2004radar**
4. **madheswaran2012estimation**

Method of Moments

1. **gibson2021method**
2. **harrington1987method**
3. **lu2003comparison**
4. **yuan2009efficient**

Using Mercury MoM and post-processing

1. **topa20200303**
2. **topa-4-14-2024**
3. **topa-4-20-2024**
4. **Topa-2020-07-07**

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 $c = alpha + 1$

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 straightaway. 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
```

```
    ! phi = elevation (North Pole = 0, equator = 90)
    type :: electricFields
        real     ( rp ) :: meanTotalRCS = 0.0_rp
        real     ( rp ) :: dBsm  = 0.0_rp
        real     ( rp ) :: theta = 0.0_rp, phi = 0.0_rp
        complex  ( rp ) :: thetaTheta = cZero, thetaPhi = cZero, phiTheta = cZero, phiPhi = cZero
    contains
        procedure, public :: gather_mean_total_rcs => gather_mean_total_rcs_sub
    end type electricFields

    private :: gather_mean_total_rcs_sub
    private :: compute_mean_total_rcs_sub, compute_dbsm_sub, extract_electric_fields_sub
    private :: gather_complex_field_sub, gather_real_field_sub

    ! parameters
    integer ( ip ), parameter :: mll = MomlineLength
    ! finger print of data line: start and stop positions for each numerical field
    ! load matrix as columns
    ! sample data line:
    !      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)
    integer, parameter :: endpoints ( 1 : 10, 1 : 2 ) = &
        reshape ( [ [  1, 14, 28, 44, 62, 78,  96, 112, 130, 146 ], &
    [ 12, 25, 42, 58, 76, 92, 110, 126, 144, 160 ] ], [ 10, 2 ] )
    ! constructor
    type ( electricFields ), parameter :: electricFields0 = &
        electricFields ( meanTotalRCS = 0.0, theta = 0.0, phi = 0.0, &
        thetaTheta = cZero, thetaPhi = cZero, phiTheta = cZero, phiPhi = cZero )
contains

    ! master routine: only exposed procedure
    subroutine gather_mean_total_rcs_sub ( me, textLine )
        class ( electricFields ),       target :: me
        character ( len = mll ), intent ( in ) :: textLine
            call extract_electric_fields_sub ( me, textLine )
            call compute_mean_total_rcs_sub  ( me )
            call compute_dbsm_sub            ( me )
        return
    end subroutine gather_mean_total_rcs_sub

    ! Sciacca prescription
    subroutine compute_dbsm_sub ( me )
        class ( electricFields ), target :: me
            me % dBsm = 10.0_rp * log10 ( me % meanTotalRCS )
        return
    end subroutine compute_dbsm_sub

    ! Sciacca prescription
    subroutine compute_mean_total_rcs_sub ( me )
        class ( electricFields ), target :: me
            me % meanTotalRCS = abs ( me % thetaTheta ) + abs ( me % thetaPhi ) &
                              + abs ( me % phiTheta ) + abs ( me % phiPhi )
            me % meanTotalRCS = me % meanTotalRCS / real ( 2, kind = rp )
        return
    end subroutine compute_mean_total_rcs_sub

    subroutine extract_electric_fields_sub ( me, textLine )
        class ( electricFields ),       target :: me
        character ( len = mll ), intent ( in ) :: textLine
        integer ( ip ) :: position = 0
            ! move across text line gathering numeric values
            position = 1
            call gather_real_field_sub  &
                ( position = position, real_value = me % theta, textLine = textLine, fmt = "( f12.4 )" )
            call gather_real_field_sub  &
                ( position = position, real_value = me % phi,   textLine = textLine, fmt = "( f12.4 )" )
            call gather_complex_field_sub  &
                ( position = position, complex_value = me % thetaTheta, textLine = textLine )
            call gather_complex_field_sub  &
                ( position = position, complex_value = me % thetaPhi,   textLine = textLine )
            call gather_complex_field_sub  &
                ( position = position, complex_value = me % phiTheta,   textLine = textLine )
            call gather_complex_field_sub  &
                ( position = position, complex_value = me % phiPhi,     textLine = textLine )
        return
    end subroutine extract_electric_fields_sub

    subroutine gather_real_field_sub ( position, real_value, textLine, fmt )
        real ( rp ),             intent ( out )   :: real_value
        integer ( ip ),          intent ( inout ) :: position
        character ( len = mll ), intent ( in )    :: textLine
        character ( len =   9 ), intent ( in )    :: fmt
            left  = endpoints ( position, 1 )
            right = endpoints ( position, 2 )
            write ( number, fmt = 100 ) textLine ( left : right )
                if ( io_stat /= 0 ) then
                    write ( * , fmt = '( 3g0 )' ) "Failure to WRITE string value '", trim ( textLine ( left : right ) ) , "'."
                    write ( * , fmt = fmt_stat  ) io_stat
                    write ( * , fmt = fmt_iomsg ) trim ( io_msg )
                    stop "Error occured in module 'mClassElectricFields', subroutine 'gather_real_field_sub'."
                end if
            read  ( number, fmt = fmt ) real_value
                if ( io_stat /= 0 ) then
```

```
                    write ( * , fmt = '( 5g0 )' ) "Failure to READ string value '", trim ( textLine ( left : right ) ) , &
                        "' as a REAL number using format descriptor ", fmt, "."
                    write ( * , fmt = fmt_stat  ) io_stat
                    write ( * , fmt = fmt_iomsg ) trim ( io_msg )
                    stop "Error occured in module 'mClassElectricFields', subroutine 'gather_real_field_sub'."
                end if
            position = position + 1
        return
        100 format ( g0 )
    end subroutine gather_real_field_sub

    subroutine gather_complex_field_sub ( position, complex_value, textLine )
        complex ( rp ),          intent ( out )   :: complex_value
        integer ( ip ),          intent ( inout ) :: position
        character ( len = mll ), intent ( in )    :: textLine
        real ( rp ) :: x = 0.0_rp, y = 0.0_rp
            call gather_real_field_sub ( position = position, real_value = x, textLine = textLine, fmt = "( e15.7 )" )
            call gather_real_field_sub ( position = position, real_value = y, textLine = textLine, fmt = "( e15.7 )" )
            complex_value = cmplx ( x, y )
        return
    end subroutine gather_complex_field_sub

end module mClassElectricFields
```

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

```
module mClassDataFile

    use, intrinsic :: iso_fortran_env,  only : iostat_end
    ! classes
    use mClassAverages,                 only : average, average0
    use mClassElectricFields,           only : electricFields, electricFields0
    use mClassMesh,                     only : meshReal
    use mAllocations,                   only : allocationToolKit, allocationToolKit0
    use mAllocationsSpecial,            only : allocate_rank_one_averages_sub
    ! utilities
    use mLibraryOfConstants,            only : fileNameLength, messageLength, MoMlineLength
    ! use mBulkRCS,                       only : BulkRCS, BulkRCS0
    use mFileHandling,                  only : safeopen_readonly, safeopen_writereplace
    use mFormatDescriptors,             only : fmt_one, fmt_stat, fmt_iomsg, fmt_shape2
    use mPrecisionDefinitions,          only : ip, rp
    use mTextFileUtilities,             only : count_lines_sub, mark_frequencies_sub,read_text_lines_sub, file_closer_sub, &
                                               iostat_check_sub
    ! use mTextFileUtilities,             only : count_lines_sub, file_closer_sub, iostat_check_sub, mark_frequencies_sub, &
    !                                            parse_name_sub, read_text_lines_sub
    implicit none

    ! parameters
    integer ( ip ), parameter :: fnl = fileNameLength, msgl = messageLength, mll = MoMlineLength
    character ( len = 9 ), parameter :: strAzimuth   = "azimuth  "
    character ( len = 9 ), parameter :: strElevation = "elevation"
    character ( len = * ), parameter :: moduleCrash  = "Program crashed in module 'mClassDataFile', "

    integer :: io_stat = 0
    character ( len = msgl ) :: io_msg = ""

    type :: dataFile4112
        ! rank 2
        real ( rp ),             allocatable :: rcs_table  ( : , : ) ! angle mesh length x nu mesh length
        real ( rp ),             allocatable :: dbsm_table ( : , : ) ! angle mesh length x nu mesh length
        ! ! rank 1
        integer ( ip ),          allocatable :: lineNumbersFrequency ( : )!, &
!                                               lineNumbersFinished  ( : )
        type ( average ),        allocatable :: perFrequencyAverage  ( : )  ! nu mesh length
        type ( average )                     :: globalAverage
        character ( len = mll ), allocatable :: lines4112Text ( : )        ! length numlines4112Text
        ! rank 0
        type ( electricFields ) :: eFields = electricFields0
        type ( meshReal ) :: meshFrequency, &
                             meshFreeAngle
        integer ( ip ) :: numFrequencies   = 0, &
                          numFixedAngles   = 0, &
                          numFreeAngles    = 0, &
                          numMeasurements  = 0, &
                          numLines4112Text = 0
        integer ( ip ) :: io_unit = 0
        character ( len =   9 ) :: angleFixedType  = "", angleFreeType = ""
        character ( len = fnl ) :: file4112Name = "", fileRCStxtName = "", fileRCSbinaryName  = "", &
                                                      filedBsmTxtName = "", filedBsmBinaryName = ""
        ! allocation tools
        type ( allocationToolKit ) :: myKit = allocationToolKit0
    contains
        procedure, public :: allocate_rcs_tables              => allocate_rcs_tables_sub, &
                             allocate_rcsAverages             => allocate_rcsAverages_sub, &
                             characterize_rcs_by_frequency    => characterize_rcs_by_frequency_sub, &
                             check_rcs_table_structure        => check_rcs_table_structure_sub, &
                             establish_free_angle_mesh        => establish_free_angle_mesh_sub, &
                             establish_frequency_mesh         => establish_frequency_mesh_sub, &
```

```
                            extract_rcs_from_4112_file        => extract_rcs_from_4112_file_sub, &
                            harvest_frequencies               => harvest_frequencies_sub, &
                            set_file_names                    => set_file_names_sub, &
                            set_free_angle_azimuth            => set_free_angle_azimuth_sub, &
                            set_free_angle_elevation          => set_free_angle_elevation_sub, &
                            write_rcs_file_set                => write_rcs_file_set_sub, &
                            write_rcs_binary                  => write_rcs_binary_sub, &
                            write_rcs_csv                     => write_rcs_csv_sub, &
                            write_dBsm_binary                 => write_dBsm_binary_sub, &
                            write_dBsm_csv                    => write_dBsm_csv_sub, &
                            write_summary_by_frequency        => write_summary_by_frequency_sub, &
                            write_summary_for_all_frequencies => write_summary_for_all_frequencies_sub
    end type dataFile4112

    private :: allocate_rcs_tables_sub, allocate_rcsAverages_sub, &
               establish_free_angle_mesh_sub, establish_frequency_mesh_sub, extract_rcs_from_4112_file_sub, &
               harvest_frequencies_sub, &
               set_file_names_sub, set_free_angle_azimuth_sub, set_free_angle_elevation_sub, &
               write_summary_by_frequency_sub, write_summary_for_all_frequencies_sub

contains

    subroutine characterize_rcs_by_frequency_sub ( me )
        class ( dataFile4112 ), target :: me
        type ( average ), pointer :: p => null ( )
        integer ( ip ) :: kFrequency = 0

            sweep_frequencies: do kFrequency = 1, me % numFrequencies
                p => me % perFrequencyAverage ( kFrequency )
                    call p % find_max_and_min          ( vector = me % rcs_table ( 1 : me % numFreeAngles, kFrequency ) )
                    call p % compute_mean_and_variance ( vector = me % rcs_table ( 1 : me % numFreeAngles, kFrequency ), &
                                                                  one = me % meshFreeAngle % one )
                p => null ( )
            end do sweep_frequencies

        return
    end subroutine characterize_rcs_by_frequency_sub

    module subroutine write_summary_for_all_frequencies_sub ( me )
        class ( dataFile4112 ), target :: me
        integer ( ip ) :: kFrequency = 0, first = 0, last = 0, numConvolution = 0
        real ( rp ), allocatable :: global_rcs ( : ), one ( : )
            ! allocate memory for all RCS measurements
            numConvolution = me % numFrequencies * me % numFreeAngles
            call me % myKit % allocate_rank_one_reals ( real_array = global_rcs, index_min = 1, index_max = numConvolution )
            call me % myKit % allocate_rank_one_reals ( real_array = one,        index_min = 1, index_max = numConvolution )
            ! load data vector
            sweep_frequencies: do kFrequency = 1, me % meshFrequency % numMeshElements
                first = ( kFrequency - 1 ) * me % numFreeAngles + 1
                last  = first + me % numFreeAngles - 1
                global_rcs ( first : last ) = me % rcs_table ( 1 : me % numFreeAngles, kFrequency )
            end do sweep_frequencies
            ! compute extrema
            one ( : ) = global_rcs ( : ) - global_rcs ( : ) + 1.0_rp
            call me % globalAverage % find_max_and_min          ( vector = global_rcs ( 1 : numConvolution ) )
            call me % globalAverage % compute_mean_and_variance ( vector = global_rcs ( 1 : numConvolution ), one = one )
            write ( * , * )
            write ( * , fmt = 100 ) me % globalAverage % mean, &
                                    me % globalAverage % standardDeviation, &
                                    me % globalAverage % extrema % minValue, &
                                    me % globalAverage % extrema % maxValue
        return
        100 format ( "Aggregate for all RCS measurements: mean = ", g0, " +/- ", g0, ", min = ", g0, ", max = ", g0 )
    end subroutine write_summary_for_all_frequencies_sub

    module subroutine write_summary_by_frequency_sub ( me )
        class ( dataFile4112 ), target :: me
        integer ( ip ) :: kFrequency = 0
            write ( * , * )
            sweep_frequencies: do kFrequency = 1,   me % meshFrequency % numMeshElements
                write ( * , fmt = 100 ) kFrequency, me % meshFrequency % meshValues ( kFrequency ), &
                                                    me % perFrequencyAverage ( kFrequency ) % mean, &
                                                    me % perFrequencyAverage ( kFrequency ) % standardDeviation, &
                                                    me % perFrequencyAverage ( kFrequency ) % extrema % minValue, &
                                                    me % perFrequencyAverage ( kFrequency ) % extrema % maxValue
            end do sweep_frequencies
        return
        100 format ( I3.3, ". nu = ", g0, ", mean RCS = ", g0, " +/- ", g0, ", min = ", g0, ", max = ", g0 )
    end subroutine write_summary_by_frequency_sub

    module subroutine write_rcs_file_set_sub ( me )
        class ( dataFile4112 ), target :: me
            call me % write_rcs_csv     ( )
            call me % write_rcs_binary  ( )
            call me % write_dBsm_csv    ( )
            call me % write_dBsm_binary ( )
        return
    end subroutine write_rcs_file_set_sub

    module subroutine write_rcs_binary_sub ( me )
        class ( dataFile4112 ), target :: me
        integer ( ip )           :: io_rcs = 0
```

```
        character ( len = msgl ) :: crashChain = ""

            crashChain = moduleCrash // "subroutine 'write_rcs_binary_sub'."

            open ( newunit = io_rcs, file = me % fileRCSbinaryName, action = 'WRITE', status = 'REPLACE', form = 'UNFORMATTED', &
                                iostat = io_stat, iomsg = io_msg )
            call iostat_check_sub ( action = "UNFORMATTED OPENING", fileName = me % fileRCSbinaryName, crashChain = crashChain, &
                                iostat = io_stat, iomsg = io_msg )

            write ( io_rcs, iostat = io_stat,  iomsg = io_msg ) me % rcs_table ( 1 : me % meshFreeAngle  % numMeshElements, &
                                                                                1 : me % meshFrequency  % numMeshElements )
            call iostat_check_sub ( action = "UNFORMATTED WRITE to", fileName = me % fileRCSbinaryName, crashChain = crashChain, &
                                iostat = io_stat, iomsg = io_msg )
            call file_closer_sub ( io_unit = io_rcs, fileName = me % fileRCSbinaryName, crashChain = crashChain )

        return
    end subroutine write_rcs_binary_sub

    module subroutine write_rcs_csv_sub ( me )
        class ( dataFile4112 ),         target :: me
        integer ( ip ) :: kFrequency = 0, kFreeAngle = 0, &
                          io_out = 0
        character ( len = msgl ) :: crashChain = ""

            crashChain = moduleCrash // "subroutine 'write_rcs_csv_sub'."
            io_out = safeopen_writereplace ( me % fileRCStxtName )
            ! write RCS values one row (frequency) at a time
            sweep_frequencies: do kFrequency = 1, me % meshFrequency % numMeshElements
                write ( io_out, fmt = me % meshFreeAngle % valuesFormatDescriptor ) ( me % rcs_table ( kFreeAngle, kFrequency ), &
                        kFreeAngle = 1, me % meshFreeAngle % numMeshElements )
                call iostat_check_sub ( action = "WRITE to", fileName = me % fileRCStxtName, crashChain = crashChain, &
                                        iostat = io_stat, iomsg = io_msg )
            end do sweep_frequencies
            ! close io handle
            call file_closer_sub ( io_unit = io_out, fileName =  me % fileRCStxtName, crashChain = crashChain )

        return
    end subroutine write_rcs_csv_sub

    module subroutine write_dBsm_binary_sub ( me )
        class ( dataFile4112 ), target :: me
        integer ( ip )           :: io_rcs = 0
        character ( len = msgl ) :: crashChain = ""

            crashChain = moduleCrash // "subroutine write_dBsm_binary_sub'."

            open ( newunit = io_rcs, file = me % filedBsmBinaryName, action = 'WRITE', status = 'REPLACE', form = 'UNFORMATTED', &
                                iostat = io_stat, iomsg = io_msg )
            call iostat_check_sub ( action = "UNFORMATTED OPENING", fileName = me % fileRCSbinaryName, crashChain = crashChain, &
                                iostat = io_stat, iomsg = io_msg )

            write ( io_rcs, iostat = io_stat,  iomsg = io_msg ) me % dBsm_table ( 1 : me % meshFreeAngle  % numMeshElements, &
                                                                                  1 : me % meshFrequency  % numMeshElements )
            call iostat_check_sub ( action = "UNFORMATTED WRITE to", fileName = me % fileRCSbinaryName, crashChain = crashChain, &
                                iostat = io_stat, iomsg = io_msg )
            call file_closer_sub ( io_unit = io_rcs, fileName = me % filedBsmBinaryName, crashChain = crashChain )

        return
    end subroutine write_dBsm_binary_sub

    module subroutine write_dBsm_csv_sub ( me )
        class ( dataFile4112 ),         target :: me
        integer ( ip ) :: kFrequency = 0, kFreeAngle = 0, &
                          io_out = 0
        character ( len = msgl ) :: crashChain = ""

            crashChain = moduleCrash // "subroutine 'write_dBsm_csv_sub'."
            io_out = safeopen_writereplace ( me % filedBsmTxtName )
            ! write RCS values one row (frequency) at a time
            sweep_frequencies: do kFrequency = 1, me % meshFrequency % numMeshElements
                write ( io_out, fmt = me % meshFreeAngle % valuesFormatDescriptor ) ( me % dBsm_table ( kFreeAngle, kFrequency ), &
                        kFreeAngle = 1, me % meshFreeAngle % numMeshElements )
                call iostat_check_sub ( action = "WRITE to", fileName = me % filedBsmTxtName, crashChain = crashChain, &
                                        iostat = io_stat, iomsg = io_msg )
            end do sweep_frequencies
            ! close io handle
            call file_closer_sub ( io_unit = io_out, fileName =  me % fileRCStxtName, crashChain = crashChain )

        return
    end subroutine write_dBsm_csv_sub

    subroutine set_file_names_sub ( me, file4112Name )
        class ( dataFile4112 ),         target :: me
        character ( len = fnl ), intent ( in ) :: file4112Name
        integer ( ip ) :: nameLength = 0

            nameLength =        len ( trim ( file4112Name ) )
            me % file4112Name       = trim ( file4112Name )
            me % fileRCStxtName     = trim ( file4112Name ( 1 : nameLength - 4 ) ) // ".rcs.txt"
            me % fileRCSbinaryName  = trim ( file4112Name ( 1 : nameLength - 4 ) ) // ".rcs.r32"
            me % filedBsmTxtName    = trim ( file4112Name ( 1 : nameLength - 4 ) ) // ".dBsm.txt"
            me % filedBsmBinaryName = trim ( file4112Name ( 1 : nameLength - 4 ) ) // ".dBsm.r32"
```

```
247
248         return
249     end subroutine set_file_names_sub
250
251     subroutine allocate_rcsAverages_sub ( me )
252         class ( dataFile4112 ), target :: me
253             call allocate_rank_one_averages_sub ( rank_1_average = me % perFrequencyAverage, &
254                                                   index_min = 1, index_max = me % numFrequencies )
255         return
256     end subroutine allocate_rcsAverages_sub
257
258     subroutine allocate_rcs_tables_sub ( me )
259         class ( dataFile4112 ), target :: me
260             call me % myKit % allocate_rank_two_reals (  rank_2_real_array = me % rcs_table,           &
261                                                          dim1_index_min = 1, dim1_index_max = me % numFreeAngles, &
262                                                          dim2_index_min = 1, dim2_index_max = me % numFrequencies )
263             call me % myKit % allocate_rank_two_reals (  rank_2_real_array = me % dBsm_table,          &
264                                                          dim1_index_min = 1, dim1_index_max = me % numFreeAngles, &
265                                                          dim2_index_min = 1, dim2_index_max = me % numFrequencies )
266         return
267     end subroutine allocate_rcs_tables_sub
268
269     subroutine establish_frequency_mesh_sub ( me )
270         class ( dataFile4112 ), target :: me
271             ! count lines in MoM file (e.q. 14844)
272             call count_lines_sub ( fullFileName = me % file4112Name, numLines = me % numLines4112Text )
273             ! allocate object to hold text of MoM file as a collection of text lines
274             call me % myKit % allocate_rank_one_characters ( character_array = me % lines4112Text, &
275                                                              index_min = 1, index_max = me % numLines4112Text )
276             ! load MoM text into memory to count frequencies and angles
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         return
281     end subroutine establish_frequency_mesh_sub
282
283     ! sweep through character array looking for "  Freq"
284     ! store these values in a temporary array until nuMesh is allocated
285     subroutine harvest_frequencies_sub ( me )
286         class ( dataFile4112 ), target :: me
287         ! pointers
288         !character ( len =  mll ),   pointer :: p => null ( )
289         type ( meshReal ),          pointer :: q => null ( )
290         type ( allocationToolKit ), pointer :: s => null ( )
291         ! temp arrays
292         real    ( ip ) :: tempFrequencyValues   ( 1 : 500 )
293         integer ( ip ) :: tempLineNumsFrequency ( 1 : 500 )
294         ! scalars
295         integer ( ip ) :: numfrequencies = 0, kFrequency = 0
296
297             ! find lines containing " Freq ="
298             call mark_frequencies_sub ( lines4112Text = me % lines4112Text,    &
299                                         numLines4112Text = me % numLines4112Text, &
300                                       tempFrequencyValues = tempFrequencyValues,   &
301                                     tempLineNumsFrequency = tempLineNumsFrequency, &
302                                           numfrequencies = numfrequencies )
303
304             ! record what we have learned about the mesh
305             q => me % meshFrequency
306                 q % numMeshElements = numfrequencies
307                 ! allocate data objects
308                 call q % allocate_mesh_real (  )
309                 s => me % myKit
310                     call s % allocate_rank_one_integers ( integer_array = me % lineNumbersFrequency, index_min = 1, &
311                                                           index_max = q  % numMeshElements )
312                 s => null ( )
313                 ! move temporary array data into data object
314                 do kFrequency = 1, q % numMeshElements
315                      q % meshValues           ( kFrequency ) = tempFrequencyValues   ( kFrequency )
316                     me % lineNumbersFrequency ( kFrequency ) = tempLineNumsFrequency ( kFrequency )
317                 end do
318                 me % numFrequencies = q % numMeshElements
319                 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 = 0.0_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 )
```

```
                        me % rcs_table  ( kFreeAngle, kFrequency ) = me % eFields % meanTotalRCS
                        me % dBsm_table ( kFreeAngle, kFrequency ) = me % eFields % dBsm
                    linePosition = linePosition + 1
                end do sweep_free_angles
            end do sweep_frequencies

        return
    end subroutine extract_rcs_from_4112_file_sub

    subroutine set_free_angle_elevation_sub ( me )
        class ( dataFile4112 ), target :: me
            me % angleFreeType  = strElevation
            me % angleFixedType = strAzimuth
        return
    end subroutine set_free_angle_elevation_sub

    subroutine set_free_angle_azimuth_sub ( me )
        class ( dataFile4112 ), target :: me
            me % angleFreeType  = strAzimuth
            me % angleFixedType = strElevation
        return
    end subroutine set_free_angle_azimuth_sub

    subroutine establish_free_angle_mesh_sub ( me, angle_min, angle_max, angle_count )
        class ( dataFile4112 ), target :: me
        real    ( rp ), intent ( in ) :: angle_min, angle_max
        integer ( ip ), intent ( in ) :: angle_count

            me % meshFreeAngle % meshAverage % extrema % minValue = angle_min
            me % meshFreeAngle % meshAverage % extrema % maxValue = angle_max
            me % meshFreeAngle % numMeshElements = angle_count
            me % numFreeAngles                   = angle_count

            call me % meshFreeAngle % allocate_mesh_real (  )
            call me % meshFreeAngle % compute_real_mesh_length ( )
            call me % meshFreeAngle % compute_real_mesh_interval ( )
            call me % meshFreeAngle % populate_real_mesh ( )
            call me % meshFreeAngle % populate_integer_mesh ( )

        return
    end subroutine establish_free_angle_mesh_sub

    subroutine check_rcs_table_structure_sub ( me )
        class ( dataFile4112 ), target :: me
            write ( * , * )
            write ( * , fmt = '(  g0 )' ) "#  #  Dimensions for RCS data container  #  #"
            write ( * , * )
            write ( * , fmt = '(  g0 )' ) "# Expected dimensions:"
            write ( * , fmt = '( 2g0 )' ) "# Number of radar frequencies scanned by MoM:   ",               me % numFrequencies
            write ( * , fmt = '( 4g0 )' ) "# Number of ", me % angleFreeType, " angles scanned by MoM:  ",   me % numFreeAngles
            write ( * , * )
            write ( * , fmt = '( 2g0 )' ) "# Container MoM 4112.txt file: rcs_table"
            write ( * , fmt = '( 6g0 )' ) "# Free angle dimension = ",  size   ( me % rcs_table, 1 ), &
                                                  " indices run from ", lbound ( me % rcs_table, 1 ), &
                                                  " to ",               ubound ( me % rcs_table, 1 )
            write ( * , fmt = '( 6g0 )' ) "# Frequency dimension  = ",  size   ( me % rcs_table, 2 ), &
                                                  " indices run from ", lbound ( me % rcs_table, 2 ), &
                                                  " to ",               ubound ( me % rcs_table, 2 )
            write ( * , * )
        return
    end subroutine check_rcs_table_structure_sub

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. Petyak
NASA Software Release Authority
(202) 358-4387

C.2 Copyright Statement by the Author

C.3 Legal Statement

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

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

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

==========

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 2: 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

D.1 Inputs

D.1.1 Main

Main module:

```
1 #! /usr/bin/python3
2
3 # # Daniel Topa
4
5 # # Excel tools
```

```
# xl_new_workbook( workbook_title )
# xl_sheet_requirements( this_workbook )
# xl_sheet_generate( this_workbook, title_sheet )
# xl_s( this_workbook )
# xl_sheet_header_footer( this_worksheet )

# # imports
import os               # probe, change directories
import sys              # python version
import datetime         # https://stackoverflow.com/questions/415511/how-to-get-the-current-time-in-python
import numpy as np
import pandas as pd
import xlsxwriter       # API for Excel
from xlsxwriter.utility import xl_rowcol_to_cell
import numpy as np
import pandas as pd

import cls_TestObject

# # modules
def xl_new_workbook( testObject ):

    MoMresults = xlsxwriter.Workbook( testObject.outputFile )
    print( "output file %s" % testObject.outputFile )
    print( "source file %s" % testObject.sourceFile )

    xl_sheet_master( MoMresults, testObject ) # MoM summary
    xl_add_data_sheets( MoMresults, testObject ) # MoM summary
    xl_sheet_provenance( MoMresults ) # provenance sheet

    return MoMresults;

#  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #

def xl_add_data_sheets( this_workbook, testObject ):

    format_MoM_title = this_workbook.add_format( )
    format_MoM_title.set_bold( )
    format_MoM_title.set_font_color( "red" )

    format_MoM_head = this_workbook.add_format( )
    format_MoM_head.set_bold( )

    format_MoM_polarization = this_workbook.add_format( )
    format_MoM_polarization.set_bold( )

    number_format = this_workbook.add_format({'num_format': '#,##0.000'})

    # https://xlsxwriter.readthedocs.io/format.html#set_center_across
    cell_format = this_workbook.add_format()
    cell_format.set_center_across()

    for index in range( 1, 29 ):
        # add sheet and tag header and footer
        title = str( index + 2 ) + ' MHz'
        print ( 'adding sheet %s' % title )
        s = xl_sheet_generate( this_workbook, title )
        xl_sheet_header_footer( s )
        s.write( "A1", "MoM 4.1.12 output (*.dat)", format_MoM_title )
        #
        s.write( "A3", "azimuth, ", format_MoM_head )
        s.write( "B3", "HH, dBsm",   format_MoM_head )
        s.write( "C3", "VV, dBsm",   format_MoM_head )
        s.write( "D3", "HV, dBsm",   format_MoM_head )
        s.write( "E3", "VH, dBsm",   format_MoM_head )
        #
        s.write( "H3", "mean", format_MoM_head )
        s.write( "J3", "standard deviation", format_MoM_head )
        #
        s.write( "G4", "HH", format_MoM_polarization )
        s.write( "G5", "VV", format_MoM_polarization )
        #
    #    AttributeError: 'str' object has no attribute '_get_xf_index'
    #    s.write( "I4", "HH", u"\u00B1" )
        s.write( "I4", '', cell_format )
        s.write( "I5", '', cell_format )
        s.set_column( "I:I", 3 )

        # = AVERAGE( B5:B364)
        # = STDEV( B5:B364 )
        s.write( "H4", '= AVERAGE( B5:B364)', number_format )
        s.write( "H5", '= AVERAGE( C5:C364)', number_format )
        s.write( "J4", '= STDEV( B5:B364 )',  number_format )
        s.write( "J5", '= STDEV( B5:B364 )',  number_format )

        # read in data file
        filename = './data/sphere-005-' + testObject.resolution + '-' + str( index + 2 ).zfill(2) + '.4112.dat.txt'
        s.write_string( "D1", filename )
        data = pd.read_csv( filename, delimiter=r"\s+", header = None )
        data_np = data.values
        row = 3
        col = 0
```

```
        for line in range( 0, len ( data_np ) ):
            cell = xl_rowcol_to_cell ( row, col )
            s.write( row, col, data_np[ line ][ 0 ], number_format )
            s.write( row, col + 1, data_np[ line ][ 1 ], number_format )
            s.write( row, col + 2, data_np[ line ][ 2 ], number_format )
            s.write( row, col + 3, data_np[ line ][ 3 ], number_format )
            s.write( row, col + 4, data_np[ line ][ 4 ], number_format )
            row += 1

    return

#  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #

def xl_sheet_generate( this_workbook, title_sheet ):

    # insure every worksheet has a header and footer
    mySheet = this_workbook.add_worksheet( title_sheet )
    xl_sheet_header_footer( mySheet )

    return mySheet;

#  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #

def xl_sheet_provenance( this_workbook ):

    # Define some global names.
    this_workbook.define_name( 'c_', '=299792458' )
    # forensic info
    s = xl_sheet_generate( this_workbook, "provenance" )
    #  # special formats
    # https://xlsxwriter.readthedocs.io/format.html?highlight=bold

    # method 1
    # setting the property as a dictionary of key/value pairs in the constructor
    format_title = this_workbook.add_format( )
    format_title.set_bold( )
    format_title.set_font_color( "blue" )

    # method 2
    # passing a dictionary of properties to the add_format() constructor
    format_time = this_workbook.add_format( {'num_format': 'yy/mm/dd hh:mm'} ) # https://xlsxwriter.readthedocs.io/working_with_dates_and_time.html

    # widen first columns
    s.set_column( "A:A", 15 )
    s.set_column( "B:B", 13 )

    # https://xlsxwriter.readthedocs.io/worksheet.html
    s.write_url( "A1", "https://en.wikipedia.org/wiki/Computational_electromagnetics", string = "Radar Cross Section Measurements" )

    #  #  provencance
    s.write( "A3", "Workbook created by", format_title )
    #s.write( "A1", tip, "boo" )

    # python notebook which creates workbook
    s.write( "A4", "python source" )
    s.write( "B4", os.path.basename( __file__ ) ) # charlie.py

    # current working directory
    s.write( "A5", "directory" )
    s.write( "B5", os.getcwd( ) ) # /Volumes/Tlaltecuhtli/repos/GitHub/topa-development/python/xlsx

    # python version
    s.write( "A6", "python version" )
    s.write( "B6", sys.version ) # "3.7.0 (default, Jun 28 2018, 07:39:16) [Clang 4.0.1 (tags/RELEASE_401/final)]"

    #  #  environment variables
    # practise row, col notation
    col = 0 # starting column
    row = 7 # starting row
    s.write( row, col, "Environment variables", format_title ); row += 1

    s.write( row, col, "$USER" ) # l127914
    s.write( row, col + 1, os.environ[ "USER" ] ); row += 1

    s.write( row, col, "$HOSTNAME" ) # Cauchy.Schwarz
    s.write( row, col + 1, os.environ[ "HOSTNAME" ] ); row += 1

    s.write( row, col, "$HOME" ) # /Users/l127914
    s.write( row, col + 1, os.environ[ "HOME" ] ); row += 1

    s.write( row, col, "timestamp" ) # 11/21/18 16:18
    s.write( row, col + 1, datetime.datetime.now( ), format_time ); row += 1

    #  #  Excel info routines
    # https://xlsxwriter.readthedocs.io/working_with_formulas.html

    row += 1 # jump
    s.write( row, col, "XL info function", format_title ); row += 1

    s.write( row, col, "platform" ) # mac
    s.write_formula( row, col + 1, '= INFO( "system" )' ); row += 1

```

```
    s.write( row, col, "recalculation mode" ) # Automatic
    s.write_formula( row, col + 1, '= INFO( "recalc" )' ); row += 1

    s.write( row, col, "active sheets" ) # 1
    s.write_formula( row, col + 1, '= INFO( "numfile" )' ); row += 1

    s.write( row, col, "cursor" ) # $A:$A$1
    s.write_formula( row, col + 1, '= INFO( "origin" )' ); row += 1

    s.write( row, col, "XL release" ) # 16.16
    s.write_formula( row, col + 1, '= INFO( "release" )' ); row += 1

    s.write( row, col, "application directory" ) # /Users/dantopa/Library/Containers/com.microsoft.Excel/Data/Documents/
    s.write_formula( row, col + 1, '= INFO( "directory" )' ); row += 1

    s.write( row, col, "operating systems" ) # Macintosh (Intel) Version 10.13.3 (Build 17D47)
    s.write_formula( row, col + 1, '= INFO( "osversion" )' ); row += 1

    return

#  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #

def xl_sheet_header_footer( this_worksheet ):

    # header: sheet name (center)
    # footer: date/time, page number, path/file

    myheader = "&C&12&A" # fontsize 12
    myfooter = "&L&8&T\n&8&D" + "&C &P / &N" + "&R&8&Z\n&8&F" # fontsize 8

    this_worksheet.set_header( myheader )
    this_worksheet.set_footer( myfooter )

    return

#  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #

def xl_sheet_master( this_workbook, testObject ):

    number_format = this_workbook.add_format({'num_format': '#,##0.000'})

    masterRow = 0
    masterCol = 0
    xl_set_label_column ( this_workbook, testObject, masterRow, masterCol )

    dataRow = 8
    dataCol = 0
    s = this_workbook.get_worksheet_by_name( testObject.masterSheet )
    for index in range(1, 29):
        dataCol += 1
        nu = index + 2
        xl_computation ( s, dataRow, dataCol, nu, number_format )

    return

#  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #

# https://xlsxwriter.readthedocs.io/working_with_cell_notation.html
def xl_computation ( wsheet, row, col, nu, number_format ):

    # frequency
    wsheet.write_number ( row, col, nu )

    # wavelength =  c_ / ( B11 * 1000000 )
    cell = xl_rowcol_to_cell ( row, col )
    wsheet.write ( row + 1, col, '= c_ / ( ' + cell + ' * 1000000 )', number_format ); row += 1

    # = radius / wavelength
    cell = xl_rowcol_to_cell ( row, col )
    wsheet.write ( row + 1, col, '= radius / ' + cell, number_format ); row += 3

    # MoM average dBsm  = '30 MHz'!$H4
    wsheet.write_formula(row, col, "= '" + str( nu ) + " MHz'!$H$4", number_format ); row += 1
    # relative error dBsm
    cell = xl_rowcol_to_cell ( row - 1, col )
    wsheet.write_formula ( row, col, '= 1 - size_optical_dbsm / ' + cell, number_format ); row += 2

    # rcs, sq m  = 10^( B15 / 10 )
    cell = xl_rowcol_to_cell ( row - 3, col )
    wsheet.write_formula ( row, col, '= 10^( ' + cell + ' / 10 )', number_format ); row += 1
    # rel error (sq m)  =  1 - size_optical_sq_m / B18
    cell = xl_rowcol_to_cell ( row - 1, col )
    wsheet.write_formula ( row, col, '= 1 - size_optical_sq_m / ' + cell, number_format )

    return

#  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #

def xl_set_label_column( wbook, testObject, row, col ):

    # method 1
    # setting the property as a dictionary of key/value pairs in the constructor
```

```
    format_title = wbook.add_format( )
    format_title.set_bold( )
    format_title.set_font_color( "blue" )

    format_label = wbook.add_format( )
    format_label.set_bold( )

    # https://xlsxwriter.readthedocs.io/example_defined_name.html
    # https://docs.python.org/2.0/ref/strings.html
    wbook.define_name( 'c_', '=299792458' )
    #string = '\'=' + str( testObject.sizeValue / 2 ) + '\''
    #print( 'string = %s' % string )
    wbook.define_name( 'radius', '=5' )
    wbook.define_name( 'size_optical_sq_m', '=\'' + testObject.masterSheet + '\'!$B$6' )
    wbook.define_name( 'size_optical_dbsm', '=\'' + testObject.masterSheet + '\'!$B$7' )

    # sheet operations
    s = xl_sheet_generate( wbook, testObject.masterSheet )
    s.set_first_sheet( )

    # widen first columns
    s.set_column( "A:A", 17 )
    s.set_column( "B:B", 10  )

    # column of labels
    s.write_string( row, col, 'INPUT', format_title ); row += 2

    s.write( row, col, 'MoM output:', format_label )
    s.write( row, col + 1, testObject.sourceFile ); row += 2

    s.write( row, col, testObject.sizeName, format_label );
    s.write( row, col + 1, testObject.sizeValue )
    s.write( row, col + 2, 'm' );  row += 1

    s.write( row, col, 'optical size', format_label )
    s.write( row, col + 1, '= pi() * radius^2' )
    s.write_string( row, col + 2, testObject.areaUnits ); row += 1
    s.write_formula( row, col + 1, '= 10 * LOG10( size_optical_sq_m )' );
    s.write( row, col + 2, 'dB area' ); row += 2

    s.write( row, col, 'frequency (MHz)', format_label ); row += 1
    s.write( row, col, 'wavelength (m)', format_label ); row += 1
    s.write( row, col, 'radius / lambda', format_label ); row += 2

    s.write( row, col, 'MoM average (dbSm)', format_label ); row += 1
    s.write( row, col, 'rel error (dbSm)', format_label ); row += 2

    s.write( row, col, 'rcs, sq m', format_label ); row += 1
    s.write( row, col, 'rel error (sq m)', format_label )

    xl_sheet_header_footer( s )


    return

# root@f21d93a5a2e9:sphere $ python tools_xl.py
#
# root@f21d93a5a2e9:sphere $ date
# Wed Jun 24 01:19:38 MDT 2020
#
# root@f21d93a5a2e9:sphere $ pwd
# /Tlaloc/python/sphere
```

D.1.2 Class Test Object

Radar return data.

```
#!/usr/bin/python3

# # Daniel Topa

# imports
import math                 # pi
import uuid                 # Universal Unique IDentifier
#from pathlib import Path   # rename file

class TestObject( object ):
    def __init__( self ):

        self._descriptor      = None    # sphere
        self._sizeName        = None    # diameter
        self._sizeValue       = None    # 10
        self._sizeUnits       = None    # m
        self._areaValue       = None    # pi r^2
        self._areaUnits       = None    # m^2
        self._resolution      = None    # 04
        self._mastersheet     = None    # sphere, d = 10 m
        self._sourceFile      = None    # *.dat
```

```
        self._sourcePath      = None    # absolute path to *.dat
        self._sourcePathFile  = None    # path + source file name
        self._outputFile      = None    # *.xlsx
        self._outputPath      = None    # absolute path to *.xlsx
        self._outputPathFile  = None    # path + *.xlsx
        self._uuid            = uuid.uuid4( ) # de facto time stamp

#   P R O P E R T I E S   #

    @property
    def descriptor( self ):
        """Descriptor (sphere, cube, etc.)"""
        return self._descriptor

    @property
    def sizeName( self ):
        """Name of size parameter (edge, radius, etc.)"""
        return self._sizeName

    @property
    def sizeValue( self ):
        """Length parameter"""
        return self._sizeValue

    @property
    def sizeUnits( self ):
        """Units (m, mm, etc.)"""
        return self._sizeUnits

    @property
    def areaValue( self ):
        """Area"""
        return self._areaValue

    @property
    def areaUnits( self ):
        """Area units (m^2, mm, etc.)"""
        return self._areaUnits

    @property
    def masterSheet( self ):
        """Name of master sheet"""
        return self._masterSheet

    @property
    def sourcePath( self ):
        """Path (absolute) to source file"""
        return self._sourcePath

    @property
    def sourceFile( self ):
        """Path + Name for input file"""
        return self._sourceFile

    @property
    def outputFile( self ):
        """Name of output file"""
        return self._outputFile

    @property
    def outputPath( self ):
        """Path (absolute) to output file"""
        return self._outputPath

    @property
    def outputPathFile( self ):
        """Path + Name for output file"""
        return self._outputPathFile

    @property
    def uuid( self ):
        """Universal unique identifier: connects requirements to source document"""
        return self._uuid

#   S E T T E R S   #

    @descriptor.setter
    def descriptor( self, value ):
        self._descriptor = value

    @sizeName.setter
    def sizeName( self, value ):
        self._sizeName = value

    @sizeValue.setter
    def sizeValue( self, value ):
        self._sizeValue = value

    @sizeUnits.setter
    def sizeUnits( self, value ):
        self._sizeUnits = value

```

```
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 )
```

```
        print( 'sizeValue      = %s' % self.sizeValue )
        print( 'sizeUnits      = %s' % self.sizeUnits )
        print( 'sourcePath     = %s' % self.sourcePath )
        print( 'sourceFile     = %s' % self.sourceFile )
        print( 'sourcePathFile = %s' % self.sourcePathFile )
        print( 'outputFile     = %s' % self.outputFile )
        print( 'outputPath     = %s' % self.outputPath )
        print( 'outputPathFile = %s' % self.outputPathFile )
        print( 'uuid           = %s' % self.uuid )
        return

#  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #

    def scenario( self ):
        self.setup_io( ) # establish outut file
        #self.read_MoM_file( )
        self.area_circular( ) # compute area for given geometry
        return

#  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #

    def read_MoM_file( self ):
        ## ## read source file
        print ( "reading source file %s" % self.sourceFile )
        # https://stackoverflow.com/questions/3277503/in-python-how-do-i-read-a-file-line-by-line-into-a-list
        with open( self.sourceFile ) as f:
            self.col_lines = f.read( ).splitlines( )
            self.numLines = len( self.col_lines )
        return

#  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #

    def setup_io( self ):
        # combine path and file name
        self.sourcePathFile  = self.sourcePath + self.sourceFile
        self.outputPathFile  = self.outputPath + self.outputFile
        self.masterSheet     = self.descriptor + ', ' + self.sizeName[0] + ' = ' + str( self.sizeValue ) + ' ' + self.sizeUnits
        return

#  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #

    def area_circular( self ):
        # combine path and file name
        self.areaValue = math.pi * ( self.sizeValue / 2 )**2
        return

#  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #

```

D.1.3 Excel Details

Toolkit for writing to spreadsheets.

```
#! /usr/bin/python3

# # Daniel Topa

# # Excel tools
# xl_new_workbook( workbook_title )
# xl_sheet_requirements( this_workbook )
# xl_sheet_generate( this_workbook, title_sheet )
# xl_s( this_workbook )
# xl_sheet_header_footer( this_worksheet )

# # imports
import os               # probe, change directories
import sys              # python version
import datetime         # https://stackoverflow.com/questions/415511/how-to-get-the-current-time-in-python
import numpy as np
import pandas as pd
import xlsxwriter       # API for Excel
from xlsxwriter.utility import xl_rowcol_to_cell
import numpy as np
import pandas as pd

import cls_TestObject

# # modules
def xl_new_workbook( testObject ):

    MoMresults = xlsxwriter.Workbook( testObject.outputFile )
    print( "output file %s" % testObject.outputFile )
    print( "source file %s" % testObject.sourceFile )

    xl_sheet_master( MoMresults, testObject ) # MoM summary
    xl_add_data_sheets( MoMresults, testObject ) # MoM summary
    xl_sheet_provenance( MoMresults ) # provenance sheet

    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", u"\u00B1" )
80         s.write( "I4", '', cell_format )
81         s.write( "I5", '', 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     #  #  provencance
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, "$USER" ) # l127914
170     s.write( row, col + 1, os.environ[ "USER" ] ); row += 1
171
172     s.write( row, col, "$HOSTNAME" ) # Cauchy.Schwarz
173     s.write( row, col + 1, os.environ[ "HOSTNAME" ] ); row += 1
174
175     s.write( row, col, "$HOME" ) # /Users/l127914
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$4", number_format ); row += 1
263     # relative error dBsm
264     cell = xl_rowcol_to_cell ( row - 1, col )
265     wsheet.write_formula ( row, col, '= 1 - size_optical_dbsm / ' + cell, number_format ); row += 2
266 
267     # rcs, sq m  = 10^( B15 / 10 )
268     cell = xl_rowcol_to_cell ( row - 3, col )
269     wsheet.write_formula ( row, col, '= 10^( ' + cell + ' / 10 )', number_format ); row += 1
270     # rel error (sq m)  =  1 - size_optical_sq_m / B18
271     cell = xl_rowcol_to_cell ( row - 1, col )
272     wsheet.write_formula ( row, col, '= 1 - size_optical_sq_m / ' + cell, number_format )
273 
274     return
275 
276 #  ==   ==   == ==   ==   == ==   ==   == ==   ==   ==  #
277 
278 def xl_set_label_column( wbook, testObject, row, col ):
279 
280     # method 1
281     # setting the property as a dictionary of key/value pairs in the constructor
282     format_title = wbook.add_format( )
283     format_title.set_bold( )
284     format_title.set_font_color( "blue" )
285 
286     format_label = wbook.add_format( )
287     format_label.set_bold( )
288 
289     # https://xlsxwriter.readthedocs.io/example_defined_name.html
290     # https://docs.python.org/2.0/ref/strings.html
291     wbook.define_name( 'c_', '=299792458' )
292     #string = '\'=' + str( testObject.sizeValue / 2 ) + '\''
293     #print( 'string = %s' % string )
294     wbook.define_name( 'radius', '=5' )
295     wbook.define_name( 'size_optical_sq_m', '=\'' + testObject.masterSheet + '\'!$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
345
```