User Guide for SMARTIES
Supplementary Information – list of functions

W. R. C. Somerville, B. Auguié, E. C. Le Ru*

The MacDiarmid Institute for Advanced Materials and Nanotechnology, School of Chemical and Physical Sciences, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand

1. Scripts

```
ScriptSolveForFixed
% Example script showing how to obtain the field expansion coefficients and
% far-field cross-sections for a spheroid at a single wavelength, and fixed orientation.
% Outputs the far-field cross-sections with accuracy estimates.

ScriptSolveForFixedSpectrum
% Example script showing how to obtain the field expansion coefficients
% and far-field cross-sections for a spheroid in a fixed orientation, as a
% function of wavelength.
% Plots the wavelength-dependent spectra for extinction, scattering, and
% absorption cross-sections for fixed orientation, as well as
% orientation-averaged.

ScriptSolveForNearField
% Example script showing how to obtain the field expansion coefficients,
% far-field cross-sections and surface field properties for a spheroid in a
% fixed orientation and at a single wavelength.
% Outputs the cross-sections and surface-averaged properties with accuracy
% estimates.
% Plots the field enhancement factors as a function of theta for three values of phi.
% Also produces a 3D plot of the surface field intensity on the particle.

ScriptSolveForNearFieldSpectrum
% Example script showing how to obtain the field expansion coefficients,
% far-field cross-sections and surface field properties for a spheroid in
% fixed orientation, as a function of wavelength.
% Plots the wavelength-dependent spectra for extinction, scattering, and
% absorption cross-sections (fixed orientation as well as orientation-averaged),
% along with lambda-dependent surface-averaged surface field properties.

ScriptSolveForT
% Example script showing how to obtain the T-matrix and
% the scattering matrix for random orientation for a spheroid
```

*Corresponding author

Email addresses: waltersom@gmail.com (W. R. C. Somerville), baptiste.auguie@gmail.com (B. Auguié), eric.leru@vuw.ac.nz (E. C. Le Ru)

```
% at a single wavelength. Prints the cross-sections with accuracy estimates,
% saves the T-matrix elements to an external text file, and plots the
% theta-dependent scattering matrix elements.

 ScriptSolveForTSpectrum
% Example script showing how to obtain the orientation-averaged optical
% properties for a spheroid as a function of wavelength.
% Plots the wavelength-dependent spectra for orientation-averaged
% extinction, scattering, and absorption cross-sections.

 ScriptTutorial
% Example script providing a step-by-step tutorial to solve the scattering
% problem at a single wavelength, with explicit calls to the low-level
% functions used in the calculations of intermediate quantities.

 ScriptTutorialSpectrum
% Example script providing a step-by-step tutorial to solve the scattering
% problem as a function of wavelength, with explicit calls to the low-level
% functions used in the calculations of intermediate quantities.
```

2. Solve functions

```
function [stC, stAbcdnm] = slvForFixed(stParams, stOptions, stGeometry)
% Calculates the expansion coefficients and cross-sections for fixed orientation
Dependencies: pstGetCrossSections, rvhGetFieldCoefficients, slvForT,
    vshMakeIncidentParameters

function [stC, stAbcdnm] = slvForFixedSpectrum(stParams, stOptions)
% Calculates the cross-sections and expansion coefficients
% for fixed orientation and multiple wavelengths
Dependencies: pstGetCrossSections, rvhGetAverageCrossSections, rvhGetFieldCoefficients,
    rvhGetSymmetricMat, rvhGetTRfromPQ, rvhTruncateMatrices,slvGetOptionsFromStruct,
    sphCalculatePQ, sphEstimateDelta,sphEstimateNB, sphMakeGeometry,
    vshGetIncidentCoefficients,vshMakeIncidentParameters

function [stC, stAbcdnm, stEsurf] = slvForNearField(stParams, stOptions)
% Calculates the cross-sections, expansion coefficients, and surface field for fixed
    orientation
Dependencies: pstMakeStructForField, pstSurfaceField, slvForFixed,
    vshMakeIncidentParameters

function [stC, stAbcdnm, stEsurf] = slvForNearFieldSpectrum(stParams, stOptions)
% Calculates the X-sections, expansion coeffs, and surface fields for many wavelengths
Dependencies: pstMakeStructForField, pstSurfaceField, slvForFixedSpectrum,
    vshMakeIncidentParameters

function [stCoa, CstTRa] = slvForT(stParams, stOptions, stGeometry)
% Calculates the T-matrix and orientation-averaged properties
Dependencies: rvhGetAverageCrossSections, rvhGetSymmetricMat, rvhGetTRfromPQ,
    rvhTruncateMatrices, slvGetOptionsFromStruct, sphCalculatePQ,sphEstimateDelta,
    sphEstimateNB, sphMakeGeometry
```

```
function stCoa = slvForTSpectrum(stParams, stOptions)
% Calculates the T-matrix and orientation-averaged properties for multiple wavelengths
Dependencies: rvhGetAverageCrossSections, rvhGetSymmetricMat, rvhGetTRfromPQ,
    rvhTruncateMatrices, slvGetOptionsFromStruct, sphCalculatePQ,sphEstimateDelta,
    sphEstimateNB, sphMakeGeometry

function [bGetR,Delta,NB,absmvec,bGetSymmetricT,bOutput] = slvGetOptionsFromStruct(stParams
    , stOptions)
% Reads optional parameters from struct, else set to default values
% (used in the "solve" functions)
```

3. High-level functions for core computations

```
function stAbcdnm = rvhGetFieldCoefficients(nNmax, CstTRa, stIncPar, stIncEabnm)
% Calculates the field coefficients from the T-(R-)matrix for a fixed orientation
Dependencies: vshGetIncidentCoefficients

function CstMaSym = rvhGetSymmetricMat(CstMa, CsMatList)
% Uses symmetry of T-matrix to get lower triangle from upper triangle part

function CstTRa = rvhGetTRfromPQ(CstPQa, bGetR)
% Calculates T (and possibly R) from P,Q for scatterers with a plane of symmetry
Dependencies: isOctave

function CstMat = rvhTruncateMatrices(CstMat, nNmax)
% Truncate matrices to a maximum number of multipoles N

function CstPQa = sphCalculatePQ(nNmax, absmvec, stRtfunc, stParams, NB)
% Calculates P,Q matrices for a spheroid using the algorithm of [JQSRT 123, 153 (2013)]
Dependencies: isOctave, sphGetModifiedBesselProducts, vshPinmTaunm

function [Delta, err] = sphEstimateDelta(stGeometry, stParams, NQmax)
% Estimates Delta from the convergence of T^{22,m=1}_{11} [see JQSRT2015]
Dependencies: rvhGetTRfromPQ, rvhTruncateMatrices, sphCalculatePQ

function [Nreturn, err] = sphEstimateN(stParams, stOptions, maxAcc)
% Estimates the required number of multipoles N for convergence of physical properties
Dependencies: rvhGetAverageCrossSections, rvhGetSymmetricMat, rvhGetTRfromPQ,
    rvhTruncateMatrices, slvGetOptionsFromStruct, sphCalculatePQ,sphEstimateNB,
    sphMakeGeometry

function [N, nNbTheta, err] = sphEstimateNandNT(stParams, stOptions, maxAcc)
% Estimates the required number of multipoles N and quadrature points nNbTheta
Dependencies: sphEstimateN, sphEstimateNbTheta

function NB = sphEstimateNB(NQ, stGeometry, stParams, acc)
% Finds the number of n required for accurate modified Bessel products
Dependencies: sphCheckBesselConvergence [private], sphGetFpovx

function [nNbTheta, err] = sphEstimateNbTheta(stParams, stOptions,maxAcc)
% Estimates nNbTheta needed for accurate calculations
Dependencies: slvForT
```

```
function stRtfunc = sphMakeGeometry(nNbTheta, a, c, theta)
% Evaluates the functions defining the geometry r(theta) for spheroids
Dependencies: auxPrepareIntegrals

function stIncPar = vshMakeIncidentParameters(sIncType, nMax, thetap, phip, alphap)
% Create struct with parameters of the incident electric field
```

4. Functions used for post-processing

```
function stCrossSection = pstGetCrossSections(k1, stAbcdnm)
% Calculates cross sections from expansion coefficients

function stEsca = pstGetNearField(stRes, stRtfunc, stRprime, stPinmTaunm)
% Calculates scattered field from integral of surface fields
Dependencies: crossComp [private], dotComp [private], pstGetResStructOneLambda,
    pstGetTangentialFields [private],rvhGetThetasForAverage, sph2cart2 [private],
    sphMakeGeometry, vshEgenThetaAllPhi, vshEthetaForPhi, vshPinmTaunm

function stParams1 = pstGetParamsStructOneLambda(stParamsAll, lambda0)
% Extracts a single lambda from the result structure to calculate fields

function stRes1 = pstGetResStructOneLambda(stResAll, lambda0)
% Extracts a single lambda from the result structure to calculate fields

function M = pstGetThetaDepFieldIntensity(stEsurf, phi0, lambda0)
% Calculates surface field intensity M=|E|^2(theta) for some phi
Dependencies: vshEthetaForPhi

function stRes = pstMakeStructForField(stAbcdnm, nNmax, lambda, epsilonIn, epsilonM,
    stIncPar, a, c)
% Creates the structure with necessary parameters to calculate surface fields
Dependencies: vshMakeIncidentParameters

function pstPlotAllSurfaceField(nNbPts, stResE, lambda0, showMesh)
% Makes a 3D plot of surface field intensity M=|E|^2(theta,phi)
Dependencies: isOctave, pstGetResStructOneLambda, pstSurfaceField, sphMakeGeometry,
    vshEthetaForPhi, vshMakeIncidentParameters

function stSM = pstScatteringMatrixOA(CstTRa,lambda,Csca,nNbTheta)
% Calculates scattering matrix for random orientation

function stEsurf = pstSurfaceField(stRes, stRtfunc, stPinmTaunm)
% Calculates electric field on the surface
Dependencies: rvhGetThetasForAverage, sphMakeGeometry, vshEFaverages [private],
    vshEgenThetaAllPhi, vshPinmTaunm, vshSquareCEm [private]

function stCrossSection = rvhGetAverageCrossSections(k1, CstTRa)
% Calculate orientation-averaged cross sections from T-matrix

function [M, nvec] = rvhGetFullMatrix(stMa, sMatName)
% Returns full matrix from a struct stored in block-rvh form
```

```
function stRtfunc = rvhGetThetasForAverage(stRtfunc)
% Modifies geometry for postprocessing by extending theta range to [0;pi]

function stEforPhi = vshEthetaForPhi(stEsurf, phi0)
% Calculate E at a given phi for all theta from stEsurf struct
```

5. Low-level functions

```
function [x, w] = auxInitLegendreQuad(N, a, b)
% Calculates nodes and weights for Legendre Gaussian quadrature of order N

function stRtfunc = auxPrepareIntegrals(nNint, sInt)
% Calculates points and weights for integral quadrature
Dependencies: auxInitLegendreQuad

function [Fpovx, rbchi, rbpsi] = sphGetFpovx(nNmax, s, x)
% Calculate F^+/x (see Eq. 46 of JQSRT 2013)
Dependencies: sphGetFpRow, vshRBchi, vshRBpsi

function [S,lossPrecS] = sphGetFpRow(n, s, x)
% Calculates problematic Bessel products for one row
Dependencies: sphGetUforFp [private]

function [stXipsiAll, stPsipsiAll] = sphGetModifiedBesselProducts(nNmax, s, x, NB)
% Returns matrices of modified Bessel products
Dependencies: sphGetBesselProductsPrimes [private], sphGetXiPsi

function stBessel = sphGetXiPsi(nNmax, s, x, NB)
% Calculates modified Bessel function products for spheroids
Dependencies: sphGetFpovx, vshRBpsi

function stEAllPhi = vshEgenThetaAllPhi(lambda, epsilon, pnm, qnm, rt, theta, sBessel,
    stPinmTaunm)
% Calculates the field on a surface r(theta) using series expansions
Dependencies: vshGetZnAll [private], vshPinmTaunm

function stIncEabnm = vshGetIncidentCoefficients(nMax, stIncPar)
% Calculates expansion coefficients of an incident plane wave
Dependencies: vshPinmTaunm

function stPinmTaunm = vshPinmTaunm(nMax, theta)
% Calculates angular functions pi and tau as defined in Mishchenko 2002

function chix = vshRBchi(n, x)
% Calculates the Riccati–Bessel function chi_n(x) = x y_n(x)

function psix = vshRBpsi(n,x)
% Calculates the Riccati–Bessel function psi_n(x)=x j_n(x)
```

6. Material definition functions

```
function epsAg = epsAg(lambda)
% Returns the wavelength–dependent relative dielectric function of silver (Ag)

function epsAu = epsAu(lambda)
% Returns the wavelength–dependent relative dielectric function of gold (Au)
```

```
function [epsilon] = epsSi(wavelength)
% Dielectric function of silicon in the UV–vis–nearIR region from tabulated
% values (D. E. Aspnes and A. A. Studna. Phys. Rev. B 27, 985–1009 (1983))
```

7. Misc. utility functions

```
function [T] = exportTmatrix( stT, format, out )
% Reshaping to long format and exporting T–matrix entries to a text file

function [ bIsOctave ] = isOctave()
% Test if we are running octave

function [] = storeGLquadrature()
% Calculates and stores points and weights for integral quadrature.
% This calculates Nt in steps of 5 from 50 to 505, then in steps of 100 from
% 600 to 2000, as well as 5 above each of those values (605, 705 etc).
Dependencies: updateGLquadrature

function quadTable = updateGLquadrature(Nt, keepOld)
% Calculates and stores points and weights for integral quadrature. This
% calculates for Nt not already stored in Utils/quadTable.mat, and returns
% the full structure of values. If the file doesn't exist, then it
% populates it with the values Nt. It also saves the values back into
% Utils/quadTable.mat.
Dependencies: auxInitLegendreQuad
```