(VERY) SHORT MANUAL FOR **GREEN BALLS L (GBL)** PACKAGE

by D. S. Grebenkov (released on 6 February 2018)

PURPOSE

Green Balls L (GBL) Matlab package was developed by D. S. Grebenkov for numerical calculation of the Green function of the Laplace operator for exterior or interior three-dimensional domains with disconnected spherical boundaries. This package allows one to solve Dirichlet, Neumann and Robin boundary value problems in these domains. The package relies on the generalized method of separation of variables (GMSV) and its realization with solid harmonics for three-dimensional domains with disconnected spherical boundaries developed by D. S. Grebenkov and S. D. Traytak:

D. S. Grebenkov and S. D. Traytak, *Semi-analytical computation of Laplacian Green functions in three-dimensional domains with disconnected spherical boundaries* (submitted to J. Comput. Phys. 2018); available online at https://pmc.polytechnique.fr/pagesperso/dg/GBL/grebenkov.pdf

COPYRIGHT

RESPONSABILITY

This is a test version of the package released on 6 February 2018. The package has been tested on a limited set of examples but might show abnormal behavior in more complicated cases. The author declines any responsibility for inaccurate or erroneous results and eventual consequent damages. Your feedback will help to correct eventual errors and improve the package. Please report any problems, errors or encountered difficulties via email: denis.grebenkov@polytechnique.edu

INSTALLATION GUIDE

1) Create a new folder (e.g., 'gbl'), decompress all files from gbl.zip and copy them to this folder.
2) Add this folder to the list of folders accessible by Matlab (e.g., in the version 6.5, one can go to items: File, Set Path, Add Folder,...).
3) The package is ready to use. Nonetheless, it is recommended first to read this manual.

MATHEMATICAL STATEMENT OF THE PROBLEM

Exterior problem: let $\Omega_1, \Omega_2, \dots, \Omega_N$ are non-overlapping balls of radii $R_1, R_2, \dots, R_N$, centered at points $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$. We search for the Green function $G(\boldsymbol{x}, \boldsymbol{y})$ of the Laplace operator $\Delta_{\boldsymbol{x}}$ in the three-dimensional domain outside these balls, $\Omega = R^3 \setminus \cup_{i=1}^N \overline{\Omega_i}$ satisfying the following equation for any fixed $\boldsymbol{y} \in \Omega$:

$$-\Delta_x G(\boldsymbol{x}, \boldsymbol{y}) = \delta(\boldsymbol{x} - \boldsymbol{y}) \qquad \boldsymbol{x} \in \Omega$$

$$\left(a_i G + b_i R_i \, {\partial G}/{\partial n}\right)_{|\partial\Omega_i} = 0 \qquad i = 1, \dots, N$$

$$G(\boldsymbol{x},\boldsymbol{y})_{|\boldsymbol{x}|\to\infty} = 0$$

where a_i and b_i are nonnegative parameters characterizing the boundary $\partial\Omega_i$ of the i-th ball, and $\partial/\partial n$ is the normal derivative directed outwards the domain. This Robin boundary condition is reduced to Dirichlet condition by setting $b_i = 0, a_i = 1$, and to Neumann condition by setting $b_i = 1, a_i = 0$. Note that $a_i + b_i > 0$ for each i, i.e., both parameters cannot be equal to 0.

Interior problem: let $\Omega_1, \Omega_2, \dots, \Omega_N$ are non-overlapping balls of radii $R_1, R_2, \dots, R_N$, centered at points $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$, and Ω_0 is a larger ball of radius R_0 and centered at $\boldsymbol{x}_0$ that fully includes all other balls: $\cup_{i=1}^{N} \overline{\Omega_i} \subset \Omega_0$. We search for the Green function $G(\boldsymbol{x},\boldsymbol{y})$ of the Laplace operator $\Delta_{\boldsymbol{x}}$ in the three-dimensional domain $\Omega = \Omega_0 \backslash \cup_{i=1}^{N} \overline{\Omega_i}$ satisfying the following equation for any fixed $\boldsymbol{y} \in \Omega$:

$$-\Delta_{\boldsymbol{x}} G(\boldsymbol{x},\boldsymbol{y}) = \delta(\boldsymbol{x}-\boldsymbol{y}) \qquad \boldsymbol{x} \in \Omega$$
$$\left(a_i G + b_i R_i \,{}^{\partial G}\!/_{\partial n}\right)_{|\partial\Omega_i} = 0 \qquad i = 0, \dots, N$$

where a_i and b_i are nonnegative parameters characterizing the boundary $\partial\Omega_i$ of the i-th ball, and $\partial/\partial n$ is the normal derivative directed outwards the domain. This Robin boundary condition is reduced to Dirichlet condition by setting $b_i = 0, a_i = 1$, and to Neumann condition by setting $b_i = 1, a_i = 0$. Note that $a_i + b_i > 0$, i.e., both parameters cannot be equal to 0. Moreover, $a_0 + a_1 + \dots + a_N > 0$, i.e., one should exclude the Neumann boundary condition at all balls (the Green function does not exist in this case).

MATLAB DESCRIPTION OF THE PROBLEM

The geometric structure of the domain and the properties of its boundaries are represented by a structure S with the following fields:

S.x is a matrix of size N x 3, with i-th row setting the position of the i-th ball;
S.R is a vector of size N x 1 setting the radii of the balls;
S.a is a vector of size N x 1 of coefficients a_i in the boundary condition (note that $a_i = 0$ describes Neumann condition);
S.b is a vector of size N x 1 of coefficients b_i in the boundary condition (note that $b_i = 0$ describes Dirichlet condition);
S.int is a boolean variable which determines whether the problem is exterior (S.int = 0, default) or interior (S.int = 1).

Note that if S.int = 1, then the last (N-th) ball is treated as an englobing ball Ω_0.

FIRST STEPS

The complete list of matlab functions in the package is provided below. The main function for computing the Green function is **GBL_G**. To familiarize with its format, one can look at the demo function **GBL_demo**, which presents three examples for interior and exterior problems. One can launch this function in the matlab command line, indicating as argument the example index (1, 2 or 3).
To visualize the computational domain, one can use the function **GBL_viewS**.
To visualize the computed values of the Green function, two options are available: **GBL_viewcut** shows the Green function in a planar cut perpendicular to the z axis via a contour plot, whereas **GBL_viewiso** shows a prescribed isosurface of the Green function. There are also interactive versions of these functions: **GBL_viewcut_int** and **GBL_viewiso_int**.

LIST OF FUNCTIONS

[A,W] = **GBL_A**(S, y, W);
This auxiliary function evaluates the coefficients A^i_{mn}
Input:

'S' is the structure describing the problem;
'y' is a matrix of size m x 3 of m points y at which G(x,y) will be evaluated;
'W' is the (optional) matrix with the matrix elements W of the Green function; if omitted, W will be computed. Since W does not depend on x and y points, it is recommended to compute the matrix W only once and then supply it as an input parameter for further computations of the Green function in the same domain.

Output:

'A' is the matrix of size M x m of the coefficients A^i_{mn} evaluated at m points y; this matrix can be used for next evaluations of the Green function in the same domain;
'W' is the matrix with the matrix elements W of the Green function; this matrix can be used for next evaluations of the Green function in the same domain.

[r,theta,phi] = **GBL_cart2sph**(xx);
This auxiliary function transforms Cartesian coordinates to spherical coordinates
Input:

'xx' is a vector 1x3.

Output:

'r' is the radial coordinate;
'theta' is the azimuthal coordinate;
'phi' is the polar coordinate.

[ferr] = **GBL_checkS**(S);
This auxiliary function makes some basic checks for the structure 'S'
Input:

'S' is the structure describing the problem.

Output:

'ferr' is the error index; returns 0 for no detected errors.

[Ed] = **GBL_dist**(x, y);
This auxiliary function calculates the Euclidean distance
Input:

'x' is a matrix of size n x 3 of n points;
'y' is a matrix of size m x 3 of m points.

Output:

'Ed' is the matrix of size n x m of Euclidean distances between points x and y.

[d] = **GBL_distS**(x, S);
This auxiliary function calculates the Euclidean distance from points x to the domain
Input:

‘x’ is a matrix of size n x 3 of n points;
‘S’ is the structure describing the problem.

Output:

‘d’ is vector of size 1 x n of Euclidean distances between points x and the domain.

[S,G] = **GBL_demo**(Nexample);
This is a demo function that illustrates some features of the GBL package
Input:

‘Nexample’ is the example index (integer from 1 to 3), the default value being 1.

Output:

‘S’ is the structure describing the problem;
‘G’ is the vector of evaluated values of the Green function.

[G, W, A] = **GBL_G**(x, y, S, nmax, W, A);
This main function evaluates the Green function $G(\boldsymbol{x},\boldsymbol{y})$
Input:

‘x’ is a matrix of size n x 3 of 'n' points x at which the Green function is evaluated;
‘y’ is a matrix of size m x 3 of 'm' source points y of the Green function;
‘S’ is the structure describing the problem;
‘nmax’ is the truncation order (nmax >= 0); if omitted, the default value nmax = 2 is used;
‘W’ is the (optional) matrix with the matrix elements W of the Green function; if omitted, W will be computed. Since W does not depend on points x and y, it is recommended to compute the matrix W only once and then supply it as an input parameter for further computations of the Green function in the same domain;
‘A’ is the (optional) matrix of size M x m of the coefficients A_{mn}^{i} formerly evaluated at m source points y, with M = N*(nmax+1)^2; if omitted, A will be computed.

Output:

‘G’ is the matrix of size n x m of values of the Green function $G(\boldsymbol{x},\boldsymbol{y})$ evaluated at n points x and m points y;
‘W’ is the matrix with the matrix elements W of the Green function; this matrix can be used for next evaluations of the Green function in the same domain;
‘A’ is the matrix of size M x m of the coefficients A_{mn}^{i} evaluated at m points y; this matrix can be used for next evaluations of the Green function in the same domain.

[Gf] = **GBL_Gfund**(x, y);
This auxiliary function evaluates the fundamental solution $G_f(\boldsymbol{x},\boldsymbol{y}) = 1/(4\pi|\boldsymbol{x}-\boldsymbol{y}|)$
Input:

‘x’ is a matrix of size n x 3 of n points x at which Gf is evaluated;
‘y’ is a matrix of size m x 3 of m points y at which Gf is evaluated.

Output:

'Gf' is the matrix of size n x m of values of the fundamental solution evaluated at n points x and m points y.

[gi] = **GBL_gi**(x, i, A, S);
This auxiliary function evaluates the i-th partial solution
Input:

'x' is a matrix of size n x 3 of n points at which G is evaluated;
'i' is an index of the sphere (integer from 1 to N = length(S.R));
'A' is a matrix of size M x m of the coefficients A^{i}_{mn} formerly evaluated at m source points y (with M = N*(nmax+1)^2);
'S' is the structure describing the problem.

Output:

'gi' is the matrix of size n x m of values of the partial solution evaluated at n points x and m points y

[flag] = **GBL_isinside**(x, S);
This auxiliary function checks whether x is inside the domain Ω
Input:

'x' is a vector of size 1 x 3;
'S' is the structure describing the problem.

Output:

'flag' is the boolean variable: 1 if x does NOT belong to the domain Ω.

[psi] = **GBL_psi_minus**(n, r,theta,phi);
This auxiliary function computes the irregular solid harmonic $\psi^{-}_{mn}(r,\theta,\phi) = Y_{mn}(\theta,\phi)/r^{n+1}$
Input:

'n' is an integer degree of the harmonic (n=0,1,2,...);
'r' is a vector of size 1 x p of radial coordinates;
'theta' is a vector of size 1 x p of azimuthal coordinates;
'phi' is a vector of size 1 x p of polar coordinates.

Output:

'psi' is the matrix of size (2*n+1) x p of the values of the irregular sold harmonic ψ^{-}_{mn} with m = -n,-n+1,...,n.

[psi] = **GBL_psi_plus**(n, r,theta,phi);
This auxiliary function computes the regular solid harmonic $\psi^{+}_{mn}(r,\theta,\phi) = r^{n}Y_{mn}(\theta,\phi)$
Input:

'n' is an integer degree of the harmonic (n=0,1,2,...);
'r' is a vector of size 1 x p of radial coordinates;
'theta' is a vector of size 1 x p of azimuthal coordinates;
'phi' is a vector of size 1 x p of polar coordinates.

Output:

'psi' is the matrix of size (2*n+1) x p of the values of the regular sold harmonic ψ_{mn}^{+} with m = -n,-n+1,...,n.

[Uhat] = **GBL_Uhat**(S, nmax);
This auxiliary function constructs the matrix Uhat
Input:

'S' is the structure describing the problem;
'nmax' is the truncation order (nmax >= 0); if omitted, the default value nmax = 2 is used.

Output:

'Uhat' is the mixed-based matrix of size M x M, with M = N*(nmax+1)^2.

[G,W,A] = **GBL_viewcut**(S, y, z, nmax,nx,ny, W,A);
This function visualizes the Green function $G(\boldsymbol{x},\boldsymbol{y})$ in a planar cut perpendicular to z axis
Input:

'S' is the structure describing the problem;
'y' is a vector of size 1 x 3 of the source point y of the Green function $G(\boldsymbol{x},\boldsymbol{y})$;
'z' is a scalar level (z coordinate) defining the planar cut;
'nmax' is the truncation order (nmax >= 0); if omitted, the default value nmax = 2 is used;
'nx' is an (optional) scalar defining the number of discretization points along x direction; if omitted, nx = 50 is used;
'ny' is an (optional) scalar defining the number of discretization points along y direction; if omitted, ny = 50 is used;
'W' is the (optional) matrix with the matrix elements W of the Green function; if omitted, W will be computed. Since W does not depend on points x and y, it is recommended to compute the matrix W only once and then supply it as an input parameter for further computations of the Green function in the same domain;
'A' is the (optional) matrix of size M x 1 of the coefficients A_{mn}^{i} formerly evaluated at the source point y, with M = N*(nmax+1)^2; if omitted, A will be computed.

Output:

'G' is the matrix of size n x 1 of values of the Green function evaluated at regularly spaced points (x,y,z), with a given z and n = (nx+1)*(ny+1);
'W' is the matrix with the matrix elements W of the Green function; this matrix can be used for next evaluations of the Green function in the same domain;
'A' is the matrix of size M x 1 of the coefficients A_{mn}^{i} evaluated at the point y; this matrix can be used for next evaluations of the Green function in the same domain.

[] = **GBL_viewcut_int**(S, y, nmax);
This function interactively visualizes the Green function $G(\boldsymbol{x},\boldsymbol{y})$ in a planar cut perpendicular to z axis. The user can move the cut level z via a slider.
Input:

‘S’ is the structure describing the problem;
‘y’ is a vector of size 1 x 3 of the source point y of the Green function $G(\boldsymbol{x},\boldsymbol{y})$;
‘nmax’ is the truncation order (nmax >= 0); if omitted, the default value nmax = 2 is used;

Warning: At each move of the slider, the function automatically re-computes the Green function at a new planar cut. The computational time strongly depends on the complexity of the structure, the truncation size nmax, and the discretization level. It may take from few seconds to much longer times. To keep the visualization interactive, it is suggested to use the default value nmax = 2. However, this relatively small truncation size can result in inaccurate computations (e.g., negative values of the Green function). To get more accurate results, it is recommended to use the non-interactive function **GBL_view** with larger nmax.

[p,fs] = **GBL_viewiso**(S, y, g0, nmax, nx, ny, nz, W,A);
This function visualizes an isosurface of the Green function $G(\boldsymbol{x},\boldsymbol{y})$, i.e., the surface on which $G(\boldsymbol{x},\boldsymbol{y}) = g_0$.

Input:

‘S’ is the structure describing the problem;
‘y’ is a vector of size 1 x 3 of the source point y of the Green function $G(\boldsymbol{x},\boldsymbol{y})$;
‘g0’ is a scalar level of the Green function defining the isosurface;
‘nmax’ is the truncation order (nmax >= 0); if omitted, the default value nmax = 2 is used;
‘nx’ is an (optional) scalar defining the number of discretization points along x direction; if omitted, nx = 30 is used;
‘ny’ is an (optional) scalar defining the number of discretization points along y direction; if omitted, ny = 30 is used;
‘nz’ is an (optional) scalar defining the number of discretization points along z direction; if omitted, nz = 30 is used;
‘W’ is the (optional) matrix with the matrix elements W of the Green function; if omitted, W will be computed. Since W does not depend on points x and y, it is recommended to compute the matrix W only once and then supply it as an input parameter for further computations of the Green function in the same domain;
‘A’ is the (optional) matrix of size M x 1 of the coefficients A_{mn}^{i} formerly evaluated at the source point y, with M = N*(nmax+1)^2; if omitted, A will be computed.

Output:

‘p’ is the pointer to the patch showing the isosurface;
‘fs’ is the isosurface.

Warning: This function evaluates the Green function $G(\boldsymbol{x},\boldsymbol{y})$ on a regular three-dimensional grid of points covering the domain, and then uses the matlab function ‘isosurface’ to compute the isosurface. The related computational time strongly depends on the complexity of the domain, the truncation size nmax, and the discretization level. It may take from few seconds to much longer times.

[] = **GBL_viewiso_int**(S, y, nmax);
This function interactively visualizes an isosurface of the Green function $G(\boldsymbol{x},\boldsymbol{y})$, i.e., the surface on which $G(\boldsymbol{x},\boldsymbol{y}) = g_0$. The user can change the level g_0 interactively via a slider.
Input:

'S' is the structure describing the problem;
'y' is a vector of size 1 x 3 of the source point y of the Green function $G(\boldsymbol{x},\boldsymbol{y})$;
'nmax' is the truncation order (nmax >= 0); if omitted, the default value nmax = 2 is used;

Warning: This function evaluates the Green function $G(\boldsymbol{x},\boldsymbol{y})$ on a regular three-dimensional grid of points covering the domain, and then uses the matlab function 'isosurface' to compute the isosurface. The related computational time strongly depends on the complexity of the domain, the truncation size nmax, and the discretization level. It may take from few seconds to much longer times.
At each move of the slider, the Green function is NOT re-evaluated but the isosurface is re-computed automatically. To keep the visualization interactive, it is suggested to use relatively low discretization level. However, the coarse discretization may result in inaccurate form of isosurfaces. To get more accurate results, it is recommended to use the non-interactive function **GBL_viewiso** with a finer discretization level.

[] = **GBL_viewS**(S, x0,r0,c0);
This function visualizes the computational domain. The color of each ball indicates the reactivity of its surface, ranging from dark blue (strongly absorbing) to dark red (strongly reflecting).
Input:

'S' is the structure describing the problem;
'x0' is an (optional) vector of size p x 3 of points to visualize;
'r0' is an (optional) vector of radii of balls showing the points;
'c0' is an (optional) matrix of size p x 3 of colors of balls showing these points (default color: black).

[W, Uhat] = **GBL_W**(S, nmax, Uhat);
This auxiliary function computes the matrix W.
Input:

'S' is the structure describing the problem;
'nmax' is the truncation order (nmax >= 0); if omitted, the default value nmax = 2 is used;
'Uhat' is the (optional) mixed-based matrix of size M x M, with M = N*(nmax+1)^2; if omitted, it will be computed.

Output:

'W' is the matrix with the matrix elements W of the Green function; this matrix is used for evaluations of the Green function;
'Uhat' is the mixed-based matrix of size M x M, with M = N*(nmax+1)^2.

[Ymn] = **GBL_Ymn**(n,theta,phi);
This auxiliary function computes the (non-normalized) spherical harmonics $Y_{mn}(\theta, \phi) = P_n^m(\cos\theta)\exp(im\phi)$.
Input:

‘n’ is an integer degree of the harmonic (n=0,1,2,...);
‘theta’ is a vector of size 1 x p of azimuthal coordinates;
‘phi’ is a vector of size 1 x p of polar coordinates.

Output:

‘Ymn’ is the matrix of size (2*n+1) x p of the values of the spherical harmonic Y_{mn} with m = -n,-n+1,...,n.