Boundary Element Method Open Source Software in Matlab/ Octave/Freemat/Scilab

<table>
<tr><td>File / Module(s)</td><td>interiorSquareTestRobin.m/interiorSquareTestRobin.m</td></tr>
<tr><td>Title</td><td>Practical test of the gls.m and regls.m routines for solving a general linear system of equations in the context of the direct boundary element method.</td></tr>
<tr><td>Version(Date) and History</td><td>1. (July 2015).</td></tr>
<tr><td>Description</td><td>This is a Matlab /Octave/Freemat/Scilab code for testing the gls.m routine for solving the linear system

$$A\underline{x} = B\underline{y} + \underline{c}, \quad (1a)$$
where A and B are known $n \times n$ matrices and $\underline{c}$ is a known n-vector with

$$\alpha_i\, x_i + \beta_i\, y_i = f_i \text{ for } i = 1 \dots n \quad (1b)$$
where the α_i, β_i and f_i are constants with α_i and β_i are never both zero for each i.

The outline test problsm

n_D=2;
p_D=[0.25, 0.75; 0.75, 0.25];.

The matrices $\mathrm{L_{SS}}, \mathrm{M_{SS}}$ and $\mathrm{M_{SS}^t}$ are generally required for the implementation of the method in order to find the surface solutions and the following command generates these matrices:
[L_SS,M_SS,Mt_SS,N_SS] =lbem2_on(n_S,vertpts,elemvert,true,true,true,false); .

In the command above the ‘on’ in ‘lbem2_on’ indicates that the points are on the surface. The ‘true,true,true,false’ indicates that the $\mathrm{L_{SS}}, \mathrm{M_{SS}}$ and $\mathrm{M_{SS}^t}$ are required, but the final matrix is not required. Following this command the matricses are stored in L_SS,M_SS, and Mt_SS. Similarly, for the domain points in the methods above, only the $\mathrm{L_{DS}}$ and $\mathrm{M_{DS}}$ matrices are required, and these values are stored in L_DS and M_DS following the command:

[L_DS,M_DS,Mt_DS,N_DS]=lbem2(n_D,p_D,vecp_D,n_S,vertpts,elemvert,false,true,true,false,false);

Since the $\mathrm{M_{SS}}$ and $\mathrm{M_{SS}^t}$ matrices are always used with $\frac{1}{2}\mathrm{I}$ added, it is convenient to form the following matrices:</td></tr>
</table>

```
M_SSplus=M_SS+eye(n_S)/2;
Mt_SSplus=Mt_SS+eye(n_S)/2;
```

The exact solutions on the boundary for the test problem have been stated and are illustrated on Figure 1. In the Matlab code they are set as follows:

```
for k=1:n_S
        phi_S_exact(k)=2*(colpoints(k,1)^2-colpoints(k,2)^2);
end
for k=1:n_S/4
        v_S_exact(k)=0;
        v_S_exact(n_S/4+k)=-4;
        v_S_exact(n_S/2+k)=4;
        v_S_exact(3*n_S/4+k)=0;
end
```

Test 1: Half Dirichlet and half Neumann boundary condition.

As a first test the direct method involving the *gls* algorithm is compared with the alternative methods. In this case the direct boundary element method is still easily applicable in the usual, if rather clumsy fashion. In this test the Dirichlet boundary condition is applied on the left and top sides and the Nemann condition is applied on the right and bottom sides. In matlab this is coded as follows:

```
alpha(1:n_S/2)=1.0;
alpha(n_S/2+1:n_S)=0.0;
beta(1:n_S/2)=0.0;
beta(n_S/2+1:n_S)=1.0;
for (k=1:n_S)
        f(k)=alpha(k)*phi_S_exact(k)+beta(k)*v_S_exact(k);
end
```

Direct Methods

In the direct method, with variation in the method of solving the system of equations, the purpuse of the first part of the method is to find vectors $\underline{\hat{\varphi}_S}$ and $\underline{\hat{v}_S}$ by solving equation (9). Once these have been determined, $\underline{\hat{\varphi}_D}$ can be found using equation (11) and in matlab this is implemented with the following code:

```
phi_D=L_DS*v_S-M_DS*phi_S .
```

Direct Method 1: Swapping rows. The method that would typically be used to find the solution using the direct method (9) would be to swap rows from one side of the equation to the other. This is only possible because $a(\boldsymbol{p})$ or $b(\boldsymbol{p})$ is always zero on the boundary. This is carried out using the following code

```
A=[ L_SS(1:n_S/2,1:n_S/2) -M_SS(1:n_S/2,n_S/2+1:n_S);
L_SS(n_S/2+1:n_S,1:n_S/2) -M_SSplus(n_S/2+1:n_S,n_S/2+1:n_S)];
B=[ M_SSplus(1:n_S/2,1:n_S/2) -L_SS(1:n_S/2,n_S/2+1:n_S);
M_SS(n_S/2+1:n_S,1:n_S/2) -L_SS(n_S/2+1:n_S,n_S/2+1:n_S)];
sol=A\(B*f);
v_S(1:n_S/2)=sol(1:n_S/2);
v_S(n_S/2+1:n_S)=f(n_S/2+1:n_S);
phi_S(1:n_S/2)=f(1:n_S/2);
phi_S(n_S/2+1:n_S)=sol(n_S/2+1:n_S); .
```

Direct Method 2: Compound matrix. In this case the method of solving the system is based on the compoiund matrix in equation (3). In Matlab the coding is as follows

```
phi_v_S= [M_SSplus -L_SS; diag(alpha) diag(beta)]\[c;f];
phi_S=phi_v_S(1:n_S);
v_S=phi_v_S(n_S+1:2*n_S); .
```

Direct Method 3: gls. The gls algorithm in called in Matlab as follows:

```
[phi_S, v_S, xory, L, U, B_gls, P, Ifail ] = gls(M_SSplus, L_SS, c, n_S, alpha, beta, f); .
```

Indirect Method. In the indirect method the linear system (19) is in standard form and the solution in the domain can be found from equation (20). The Matlab coding for this is as follows.

```
for i=1:n_S
        C(i,1:n_S)=alpha(i)*L_SS(i,1:n_S)+beta(i)*Mt_SSplus(i,1:n_S);
end
sigma_S=C\f;
phi_D= L_DS*sigma_S
```

The various direct solutions ae carried out to illustrate the various matrix solution techniques. They also verify the *gls* algorithm that is the subject of this paper. The three methods solving the system arising in the direct all give the same solutions and these are shown in Table 1. The solutions using the indirect method are given in Table 2.

Test problem 1: Direct Solution		
elements	point (0.25,0.75)	point (0.75, 0.25)
exact solution	-1	1
32	-1.00116340579288	0.99140414916650
64	-1.00032267851548	0.99749398360249
128	-1.00009049851497	0.99926478494698
256	-1.00002558523010	0.99978267595968
512	-1.00000729360382	0.99993525789670
1024	-1.00000209812315	0.99998056513669

Table 1. The results from the direct solution on Test problem 1.

Test problem 1: Indirect Solution		
elements	point (0.25,0.75)	point (0.75, 0.25)
exact solution	-1	1
32	-1.00723281386399	0.93455909390238
64	-1.00406088516754	0.95961702269782
128	-1.00246190337801	0.97476365754309
256	-1.00153247427505	0.98415050828163
512	-1.00096190256784	0.99002659494003
1024	-1.00060530545130	0.99371954438783

Table 2. The results from the indirect solution on Test problem 1.

Test 2: $\alpha(\boldsymbol{p})$ and $\beta(\boldsymbol{p})$ are both generally non-zero.

The second test is based on the same solution as Test 1, but the boundary condidion is such that $\alpha(\boldsymbol{p})$ and $\beta(\boldsymbol{p})$ are finite over the boundary. In order to activate the gls method fully, the $\alpha(\boldsymbol{p})$ and $\beta(\boldsymbol{p})$ are set over a wide range of values, $\alpha(\boldsymbol{p})$ from small (almost zero) to large (10^6) and $\beta(\boldsymbol{p})$ similarly from large to small. The code for setting the boundary condidtion is as follows:

```
for k=1:n_S
        alpha(k)=10^(6*k/n_S);
        beta(k)=10^(6*(n_S-k)/n_S);
        f(k)=alpha(k)*phi_S_exact(k)+beta(k)*v_S_exact(k);
end .
```

If the boundary condition is truly mixed, that is $\alpha(\boldsymbol{p})$ and $\beta(\boldsymbol{p})$ are generally non-zero, then the row swapping method (as in Test 1, *Direct Method 1*) cannot be used. The method based on the compound matrix (as in Test 1, *Direct Method 2*) is used to verify the results from the *gls* method (as in Test 1, *Direct Method 2*). The indirect method is also applied in the same way as in Test1. The results from these tests are given Tables 3 and 4.

Test problem 2: Direct Solution		
elements	point (0.25,0.75)	point (0.75, 0.25)
exact solution	-1	1
32	-0.99313784783564	1.00082851722167
64	-0.99803407600987	1.00022336648713
128	-0.99941881322429	1.00006375224490
256	-0.99982435277907	1.00001885053645
512	-0.99994612588802	1.00000570093421
1024	-0.99998331140774	1.00000174981983

Table 3. The results from the direct solution on Test problem 2.

Test problem 2: Indirect Solution		
elements	point (0.25,0.75)	point (0.75, 0.25)
exact solution	-1	1
32	-0.93310167620744	1.00787906346035
64	-0.95833487012629	1.00457866374391
128	-0.97382320310768	1.00282928391449
256	-0.98351131440556	1.00177838256915
512	-0.98960793716359	1.00112171388078
1024	-0.99345041461506	1.00070760841826

Table 4. The results from the indirect solution on Test problem 4.

Interface	function [phi_D,phi_S,v_S]= interiorSquareTestRobin(n_S) *Input Parameters* integer n_S *The number of elements on the square, must be a multiple of 4.* *Output Parameters* phi_D *The computed solution φ at the interior points (0.25,0.75) and (0.75,0.25)* phi_S *The computed solution φ at the collocation points, the centres of the elements* v_S *The computed solution $\frac{\partial\varphi}{\partial n}$ at the collocation points, the centres of the elements*
Web source of code.	www.boundary-element-method.com/mfiles/interiorSquareTestRobin.m
Web source of this guide	www.boundary-element-method.com/mfiles / interiorSquareTestRobin _m.pdf
Web source of the algorithm	www.boundary-element-method.com/tutorials/Integral Equation Formulations of the Interior Laplace Problem.pdf
Dependent routines	lbem2.m http://www.boundary-element-method.com/mfiles/lbem2.m lbem2_on.m http://www.boundary-element-method.com/mfiles/lbem2_on.m gls.m http://www.boundary-element-method.com/mfiles/gls.m regls.m http://www.boundary-element-method.com/mfiles/regls.m square_general.m http://www.boundary-element-method.com/mfiles/square_general.m
Test problems or modules tested	gls.m http://www.boundary-element-method.com/mfiles/gls.m regls.m http://www.boundary-element-method.com/mfiles/regls.m lbem2.m http://www.boundary-element-method.com/mfiles/lbem2.m lbem2_on.m http://www.boundary-element-method.com/mfiles/lbem2_on.m
Licence	This is 'open source'; the software may be used and applied within other systems as long as its provenance is appropriately acknowledged. See the GNU Licence for more information or contact webmaster@boundary-element-method.com .
Codes that this may be used alongside this one	gls.m will normally be run before regls: http://www.boundary-element-method.com/mfiles/gls.m
Similar codes that may be of interest	A similar m-file code is available in Excel-VBA on www.boundary-element-method.com/Excel_VBA/GLS.xlsm and a similar code is available in Fortran on http://www.boundary-element-method.com/fortran/REGLS.FOR
Applications	

Author	Stephen Kirkup
References	1. Numerical Solution of General Linear Systems of Equations 2. The Boundary Element Method in Acoustics 3. www.boundary-element-method.com 4. www.freemat.info