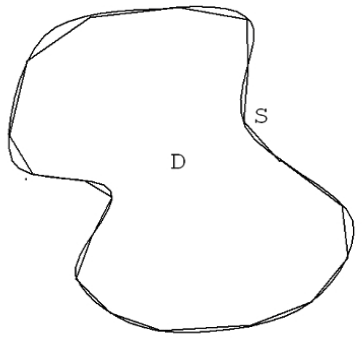
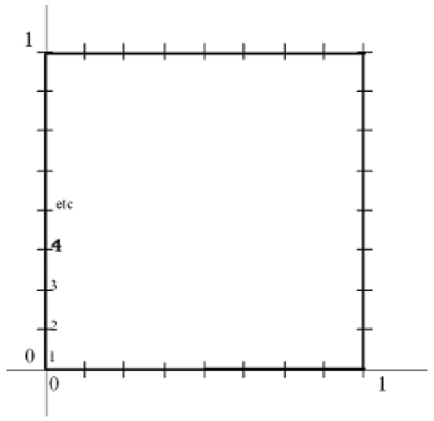
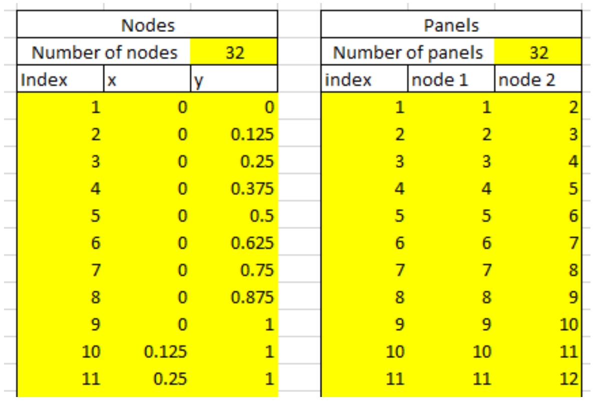
Boundary Element Method Open Source Software in Excel VBA

File / Module(s)	LIBEM2.xlsm/ LIBEM2.xlsm
Title	A spreadsheet that solves Laplace's equation in an interior two-dimensional domain.
Version(Date) and History	**Version 1.1** (April 2016). Minor changes and 'VGeom2' renamed 'VBNDRY2' (Version 1 March 2016)
Description	This Excel spreadsheet solves the interior two-dimensional Laplace equation[1] $$\nabla^2 \varphi(\mathbf{p}) = 0 \quad (\mathbf{p} \in D)\,. \qquad (1)$$ by the boundary element method (BEM)[2], where D is the interior domain. This software can form the basis for solving a range of physical or engineering problems that can be resolved to Laplace's equation in two dimensions, such as steady state heat conduction, ideal fluid, steady state electrical potential and groundwater flow. The solution by the BEM is based on the direct and indirect integral equation reformulation[3] of the Laplace equation. The integral equations that arise are solved by collocation[4].As part of the process of describing and approximating the boundary, it is represented by a set of straight line panels[5]. The data is input and output via the sheets, this in particular utilises the main feature of the spreadsheet in order to allow us to visualise the data that we are setting and creating as we proceed with the stages of the method.  The diagram to the left illustrates the boundary S, and its approximation and the interior domain D. The initial Laplace problem is set up on the sheet *Set Problem*. In order to set up the problem, the boundary must be defined, and this is carried out by approximating the boundary by a set of *n* flat panels[5]. The file LIBEM2.xlsm sets up the test of a square, which is also the sample boundary in reference 5, with 32 points or nodes defined on the boundary and $n_S = 32$ panels linking these points, as illustrated in the diagram on the right.  Perhaps the unique selling point in the boundary element method is that only the boundary requires a mesh; most methods – such as the finite difference method and finite element method are *domain methods* that require a mesh through the domain.

Generally, on the spreadsheet, the areas coloured yellow are to be completed by the user, the blue areas are not to be altered and the computed solutions generally has a green background. Intermediate computations are left with a white background. The first two sets of columns for the square test problem are shown below. The nodes with coordinates (0,0), (0,0.125), (0,0.25)... are stated in the set of columns with the title *Nodes*. This also indexes the points as they progress around the boundary, as shown in the diagram above, on the right. The number of nodes is stated at the to (in this example there are 32).

In the second set of columns termed *Panels*, which lists the index of each panel in the first column. The number of panels n_S is stated (in this example $n_S = 32$. The next two columns state the indices (as defined in the *Nodes* columns) that are at either end of each panel. For example the first panel (index 1) links vertex 1 (0,0) to vertex 2 (0,0.125). For an interior problem the nodes on a panel on the outer boundary must be ordered clockwise are around the boundary, as shown in the illustration of the square on the above right and in the tables defining the boundary to the right. If there are inner surfaces then each panel's vertices must be arranged in the anticlockwise order around the boundary.



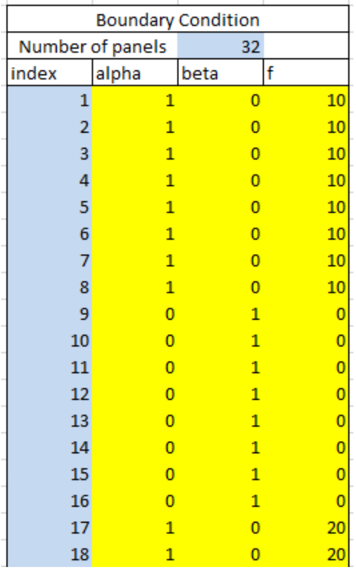
Nodes		
Number of nodes		32
Index	x	y
1	0	0
2	0	0.125
3	0	0.25
4	0	0.375
5	0	0.5
6	0	0.625
7	0	0.75
8	0	0.875
9	0	1
10	0.125	1
11	0.25	1

Panels		
Number of panels		32
index	node 1	node 2
1	1	2
2	2	3
3	3	4
4	4	5
5	5	6
6	6	7
7	7	8
8	8	9
9	9	10
10	10	11
11	11	12

The functions defined on the boundary are represented at the mid-points of each panel; the collocation points $\boldsymbol{p_i}$ for $i = 1..n$. A boundary condition must be set and this is defined in the most general Robin form at every collocation point $\boldsymbol{p_i}$,

$$\alpha(\boldsymbol{p_i})\boldsymbol{\varphi}(\boldsymbol{p_i}) + \beta(\boldsymbol{p_i})\frac{\partial\varphi}{\partial n_p}(\boldsymbol{p_i}) = f(\boldsymbol{p_i}) \quad (\boldsymbol{p_i} \in S) \qquad (2)$$

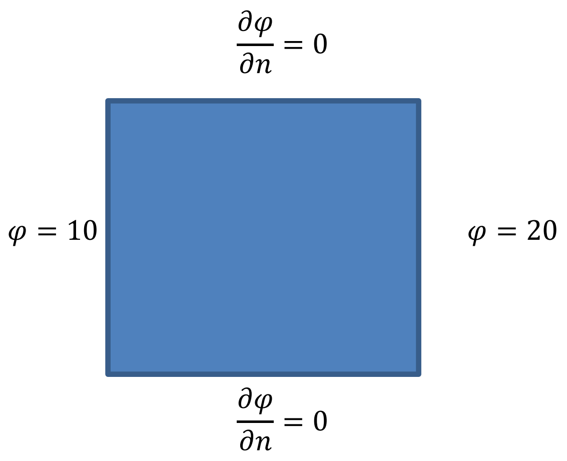
the boundary and boundary condition together forming the boundary value problem[6]. For simplicity, the notation $\varphi_{S_i} = \varphi(\boldsymbol{p_i})$, $v_{S_i} = \frac{\partial\varphi}{\partial n_p}(\boldsymbol{p_i}), \alpha_{S_i} = \alpha(\boldsymbol{p_i}), \beta_{S_i} = \beta(\boldsymbol{p_i})$ and $f_{S_i} = f(\boldsymbol{p_i})$. The boundary condition is set on the spreadsheet in the *Boundary Condition* column. The number of panels and the panel indices are already set (so they are in blue).



Boundary Condition			
Number of panels		32	
index	alpha	beta	f
1	1	0	10
2	1	0	10
3	1	0	10
4	1	0	10
5	1	0	10
6	1	0	10
7	1	0	10
8	1	0	10
9	0	1	0
10	0	1	0
11	0	1	0
12	0	1	0
13	0	1	0
14	0	1	0
15	0	1	0
16	0	1	0
17	1	0	20
18	1	0	20

The boundary condition is applied individually to each panel; the α, β and *f*- values are set for all panels.

The test boundary condition for the square is illustrated in the diagram on the right. Going around the square clockwise, as is the convention in the BEM discussed earlier, starting from the bottom left corner, the right hand side of the square has the condition $\varphi = 10$, so for the first eight panels $\alpha = 1, \beta = 0$ and $f = 10$. For the top row of panels, enumerated 9-16, the boundary condition is $\alpha = 0, \beta = 1$ and $f = 0$, and so on. These values are input in the yellow areas in the table on the left.

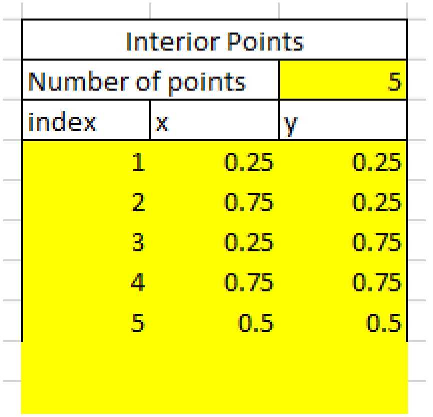

$\frac{\partial\varphi}{\partial n} = 0$
$\varphi = 10$
$\varphi = 20$
$\frac{\partial\varphi}{\partial n} = 0$

The boundary condition on each side of the square

The solution is that of a simple gradient between $\varphi = 10$ on the left side of the square and $\varphi = 20$ on the right side of the square. For simplicity, let us write ***p***=*(x,y)*, and formally the solution is

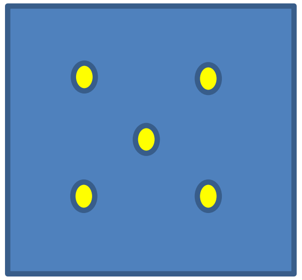
$$\varphi(\boldsymbol{p}) = 10 + 10x \qquad (3)$$

which is clearly a solution of Laplace's equation (1), that also fits the boundary condition.



Interior Points		
Number of points		5
index	x	y
1	0.25	0.25
2	0.75	0.25
3	0.25	0.75
4	0.75	0.75
5	0.5	0.5

The solution is normally required in the domain, or, more precisely, at a set of domain points. These points must be listed on the *SetProblem* sheet in the table *Interior Points*. The number of interior points n_D (=5) is stated at the top of the table. The coordinates of the chosen points are listed in the table on the left and illustrated on the square on the right. Note that the exact solution, found by inspection or by substituting the *x*-values into equation (3) is φ=12.5 when *x*=0.25, φ=15 when *x*=0.5, and φ=17.5 when x=0.75.

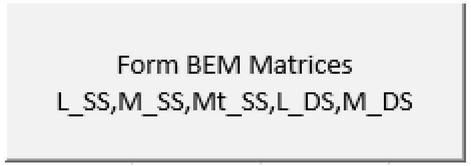


The selected domain points at which the solution is sought.

Once the mesh and boundary conditions are defined, the boundary element method computation follows several stages, which are decoupled in the spreadsheet. The spreadsheet provides a practical and versatile boundary element method for solving interior two-dimensional Laplace or *potential* problems, but the aim is also to use the spreadsheet to illutrate the method in action, with interim results at each stage. The visible results in LIBEM2.xlsm file also provides a supporting document for development of the 2D BEM in this and other programming languages.

A boundary element method is based on an integral equation reformulation of the partial differential equation. There are two fundamentally distinct approaches to deriving the integral eqation, one is historically termed the *direct method* and the other one termed the *indirect method* and both of these methods will be applied within the spreadsheet. The two methods both require the dicretisaton of the integral operators defined on the boundary, replacing the boundary integral operators by matrices[9].

The matrices that are requires by the two methods for determining the solution on the boundary are termed L_{SS}, M_{SS} and $M^t{}_{SS}$, all are $n_S \times n_S$ matices. Once a solution on the boundary is obtained the L_{DS}, M_{DS} matrices are required to find the solution in the domain and these are all $n_D \times n_S$ matrices. The matrices are generated once the button to the right on the sheet *Set Problem* is pressed. The matrices are each listed explicitly on separate sheets.



For this test problem, the sheets L_SS, M_SS and M^t_SS list the contents of the 32×32 matrices and L_DS and M_DS list the contents of the 5×32 matrices.

On pressing the button on the left the direct solution is generated and placed on the sheet *Direct Solution.* The approximations to φ and $\frac{\partial \varphi}{\partial n}$ on the boundary is found by solving the linear system of equations

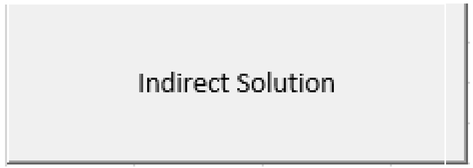
$$(\mathrm{M}_{SS} + \tfrac{1}{2}\mathrm{I})\underline{\hat{\varphi}}_S = \mathrm{L}_{SS}\underline{\hat{v}}_S \tag{3}$$

alongside the equations representing the boundary condition (2).

The solution at the domain points is then found using the following matrix multiplications

$$\underline{\hat{\varphi}}_D = \mathrm{L}_{DS}\underline{\hat{v}}_S - \mathrm{M}_{DS}\underline{\hat{\varphi}}_S \ . \tag{4}$$

On pressing the button on the right the indirect solution is generated and placed on the sheet *Indirect Solution.* In the indirect method a layer potential σ_{0_S} is introduced, a function defined on the surface, but having no physical meaning.



Its value can be obtained through solving the following system of equations

$$(D_\alpha \mathrm{L}_{SS} + D_\beta(\mathrm{M}^t_{SS} + \tfrac{1}{2}\mathrm{I}))\underline{\widehat{\sigma_0}}_S = \underline{f}_S . \tag{5}$$

where D_α and D_β are diagonal matrices with $[D_\alpha]_{ii} = \alpha_{S_i}$ and $[D_\beta]_{ii} = \beta_{S_i}$. Once $\underline{\widehat{\sigma_0}}_S$ is found the solution on the boundary and the solution at the domain points can be found using the matrix-vector multiplications

$$\underline{\hat{\varphi}}_S = L_{SS}\underline{\widehat{\sigma_0}}_S, \quad \underline{\hat{v}}_S = (M^t_{SS} + ½I)\underline{\widehat{\sigma_0}}_S, \qquad \underline{\hat{\varphi}}_D = L_{DS}\underline{\widehat{\sigma_0}}_S. \qquad (6)$$

The results from the test problem are placed on the sheets *Direct Solution* and *Indirect Solution*. The solution at the collocation points on the boundary is given and the solution at the domain points. The latter results for the two methods applied to the test problem are given below and these may be compared with the exact solution (3).

Solution at the interior points by the direct BEM

Solution in D	
Number of points	5
index	phi_D
1	12.49568
2	17.50432
3	12.49568
4	17.50432
5	15

Solution at the interior points by the indirect BEM

Solution in D	
Number of points	5
index	phi_D
1	12.4837916
2	17.4779762
3	12.4837916
4	17.4779762
5	14.9836094

Although the solution for the 32 element boundary element method seem instantaneous on the spradsheet, it is important in general to have an overview of computational costs within the boundary element method, especially for the cases in which a much larger number of elements are required. Clearly the formation of the $*_{SS}$ matrices is $O(n_S{}^2)$ and the $*_{DS}$ matrices is $O(n_D n_S)$, using O notation[10]. The solution of the linear systems are carried out by the most obvious method of LU factorisation and back substitution[11], using the VBA functions LUfac and LUfbsub, although the direct BEM also requires the wrapper of the *gls* method[11] that carries out swapping of the columns of the matrices in preparation for LU factorisation. The solution of the matrices by this sort of method is generally characteried as $O(n_S{}^3)$. The matrix multipilcations involved are $O(n_S{}^2)$ or $O(n_D n_S)$, and they are of lower order and so they can usually be subsumed into the $O(n_S{}^3)$.

Although the computational cost of the matrix-vector solutions is apparently an order greater than the cost of forming the matrices, the matrix elements usually require a numerical integration[13] to obtain their value and hence the computational cost of the matrix solution will

	normally exceed the computational cost of the determination of the matrices for higher values of n_S. Once the matrices have been found and factorised, the computational cost of the back substitution is $O(n_S{}^2)$. Hence it makes sense to save the factorised matrix and the other necessary information that is required to find a new solution. At this stage the boundary and for of the boundary condition are set. However the *f*-values can be changed and new solutions found from the saved values.
Interface	The sheet *Set Problem* is for setting up the boundary and boundary condition for the initial problem to be solved. The points in the domain at which the solution is sought is also stated. Pressing the button *Form BEM Matrices* generates the matrices required to implement the method and lists the contents of the matrices on the appropriate sheet. Two boundary element methods are supported; the *direct method* and the *indirect method*. By pressing the button *Direct Solution*, the direct solution is found and this I placed on the sheet *Direct Solution.* By pressing the button *Indirect Solution*, the direct solution is found and this is placed on the sheet *Indirect Solution.* If the direct method is used then two matrices are saved on the sheets *A_gls* and *B_gls* and further information on the sheet *perm xory*, that result from gls method and the embedded LU factorisation. If the indirect method is used then the LU factorisation of the matrix and further information are stored on the sheets *Indirect_LU* and *perm.* If another run of the method (with the same boundary definition and the same α- and β-values) then this (the *f*-values) can be placed on the sheet *New Condition*. On that sheet, the buttons *New direct solution* and *New indirect solution*, completes the new solutions and places them in the sheets *New Direct Solution* and *New Indirect Solution*. In order to test the method for finding the secondary solutions, the boundary condidtion was changed so that φ=20 on the left hand side of the square and φ=10 on the right hand side. The altered boundary condition is placed on the sheet *New Condition.*

Solution at the interior points by the direct BEM		Solution at the interior points by the indirect BEM	
Solution in D		Solution in D	
Number of points	5	Number of points	5
index	phi_D	index	phi_D
1	17.50432	1	17.4779762
2	12.49568	2	12.4837916
3	17.50432	3	17.4779762
4	12.49568	4	12.4837916
5	15	5	14.9836094

Web source of code.	www.boundary-element-method.com/Excel_VBA/LIBEM2.xlsm
Web source of this guide	www.boundary-element-method.com/Excel_VBA/LIBEM2_xls.pdf
Web source of the algorithm	http://www.boundary-element-method.com/laplace
Dependent routines	Internal Routines Module for computing the boundary element matrices L_{SS}, M_{SS}, M^t_{SS}, L_{DS}, and M_{DS}: lbem2mat.bas Module for solving the interior Laplace problem by the direct BEM: libem2SolveDirect.bas Module for computing additional solutions from the direct BEM: reSolveDirect.bas Module for solving the interior Laplace problem by the indirect BEM: libem2SolveIndirect.bas Module for computing additional solutions from the direct BEM: reSolveDirect.bas Module for verifying the geometry of the boundary and solution points: VBNDRY.bas External Routines Module for computing each element of the matrices: l2lc.bas from l2lc.xlsm gls algorithm, carrying out column swaps in the matrices in order to prepare for LU factoriation: gls.bas from GLS.xlsm regls algorithm, carrying out additional solutions , following the application of gls : regls.bas from GLS.xlsm Module for carrying out LU factorisation: LUfac.bas from LUfac.xlsm Module for caryingou the forward and back substitution to find the solution following Lufbsub from LUfac.xlsm

	Utility routines for 2D geometry: GEOM2D.bas module from GEOM.xlsm Utility module for setting the quadrature rule; gl8.bas in GL8.xlsm Verification module for checking the input geometry; VG2lc.bas in VG2lc.xlsm Verification module for checking the input quadrature rule; VQuad.bas in VQuad.xlsm
Test problems or modules tested	Testing lbem2mat.bas, libem2SolveDirec, libem2SolveIndirect.bas, reSolveDirect.bas, reSolveIndirect.bas contained in this file.
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	Not applicable.
Similar codes that may be of interest	The initial concept of the core code are based on the Acoustics, Laplace and Helmholtz libraries in Fortran. (www.boundary-element-method.com). This also builds on the concepts in the Matlab/Freemat/Octave Scilab Laplace library.
Applications	Potential problems: eg steady state heat conduction, steady state electric fields.
Author	Stephen Kirkup
References	1. Laplace's Equation 2. www.boundary-element-method.com 3. Integral Equation Formulation of the Interior Laplace Problem 4. Solution of Fredholm Integral Equations by Collocation 5. Representation of a line by flat panels 6. Boundary Value Problems and Boundary Conditions 7. Finite Difference Method 8. Finite Element Method 9. Boundary Element Method for the Interior Laplace Problem 10. Big O Notation in Computing 11. LU Factorisation and the solution of linear systems of equations 12. http://www.boundary-element-method.com/gls.htm 13. Numerical Integration