Chapter 16

Boundary Element Method

We will proceed to learn about Object Oriented Programming and Design by looking at a specific example that revolves around the solution of the Laplace equation by the Boundary Element Method (BEM). The benefits of this approach is that it takes a 2D or 3D PDE and reduces it to an algorithm that involves 1D or 2D computations only, respectively. For simplicity we will focus on the 2D case only.

16.1 The equations and the boundary conditions

The Laplace equation takes the form

\[ \nabla^2 \phi = \nabla \cdot (\nabla \phi) = 0 \quad (16.1) \]

subject to boundary conditions of the type

1. **Dirichlet** where the unknown function is imposed on a portion of the boundary, i.e. \( \phi(x) = a(s) \), \( s \) being a coordinate system along the boundary.

2. **Neumann** where the normal derivative of the function is known \( \nabla \phi \cdot n = q(s) \) where \( n \) is the outward unit normal.

3. **Robin** where a relationship of the form \( \alpha \phi + \beta \nabla \phi \cdot n = r(s) \). A well-posed problem requires the specification of only one type of boundary conditions on any portion of the boundary, that is either the function value is specified, or the normal derivative but not both.

The data imposed on the boundary form are part of the problem’s specification and can be quite arbitrary except for some minor constraints. For example, if Neumann conditions are applied over the entire boundary, then the solution can be determined only up to an additive constant, that is if \( \phi \) is a solution of \( \nabla^2 \phi = 0 \) with \( \nabla \phi \cdot n = q \), then so is \( \phi + C \) where \( C \) is an arbitrary constant.
arbitrary constant. In that case the flux have to satisfy a compatibility condition that can be derived by applying the divergence theorem to equation 16.1:

\[ \int_{\Omega} \nabla^2 \phi \, dA = \int_{\partial \Omega} \nabla \phi \cdot \mathbf{n} \, ds = \int_{\partial \Omega} q \, ds = 0 \quad (16.2) \]

### 16.2 From PDE to Integral Equation

Consider a function \( G \), solution to a Poisson equation of the form:

\[ \nabla^2 G = Q(x - x_i) \quad (16.3) \]

where \( Q(x - x_i) \) is a function that depends on the space variable \( x \) and on the source point \( x_i \) where an impulse forcing is applied. I will keep these terms vague for the time being, and will delay specifying them until later. Multiplying equation 16.3 by \( \phi \) and 16.1 by \( G \) and substracting the resulting expressions we get:

\[ \phi \nabla^2 G - G \nabla^2 \phi = \phi G(x - x_i) \quad (16.4) \]

Integrating the above equation over the area of the domain \( \Omega \) we get:

\[ \int_{\Omega} (\phi \nabla^2 G - G \nabla^2 \phi) \, dA = \int_{\Omega} \phi G(x - x_i) \, dA \quad (16.5) \]

The chain rule of differentiation and the Gauss divergence theorem can be used to turn the left hand side into a boundary integral only. Indeed since for any differentiable functions \( a \) and \( b \): \( \nabla \cdot (a \nabla b) = \nabla a \cdot \nabla b + a \nabla^2 b \), we can then write:

\[ \phi \nabla^2 G - G \nabla^2 \phi = \nabla \cdot (\phi \nabla G) - \nabla \phi \cdot \nabla G - \nabla \cdot (G \nabla \phi) + \nabla \phi \cdot \nabla G = \nabla \cdot (\phi \nabla G - G \nabla \phi) \]

Further progress can be made to simplify equation 16.11 by imposing constraints on the choices of \( Q \) and \( G \). Requiring that the function \( Q \) be the equivalent of the two-dimensional Dirac delta function, and \( G \) to be its associated free space
Green function allows us to reduce the integral on the right hand side of 16.11, since by definition:

\[ \int_{\Omega} \phi Q(x - x_i) \, dA = \phi(x_i) \tag{16.12} \]

Equation 16.11 becomes

\[ \int_{\partial\Omega} \left( \phi \frac{\partial G}{\partial n} - G \frac{\partial \phi}{\partial n} \right) \, dS = \phi(x_i) \tag{16.13} \]

Equation 16.13 is the starting point of the boundary element method. It essentially implies that if the unknown function \( \phi \) and its normal derivative are known at the boundaries, then the solution at any point in the interior can be calculated. The rub is that for any well-posed problem the function \( \phi \) or its normal derivative is known at the boundary but not both. The next step is then to use the above formulation to determine the missing data on the boundary; once the boundary data is known fully the solution in the interior can be calculated simply by moving the source point \( x_i \) to any point where the solution is desired.

The strategy for determining the missing data on the boundary starts by moving the source point to the boundary, and setting the equation purely in terms of the boundary unknowns. The equation is then termed an integral equation. Prior to presenting the discretization of this integral equation, we take a detour to determine the Green function and its properties.

**16.3 The Free Space Green Function**

The free space Green function is by definition the solution to a forced problem with homogeneous forcing, except for a single impulse of infinite magnitude in an unbounded domain. The derivation proceeds by setting the Laplace equation in polar coordinates:

\[ \nabla^2 G = \frac{\partial^2 \phi}{\partial r^2} + \frac{1}{r} \frac{\partial \phi}{\partial r} + \frac{1}{r^2} \frac{\partial^2 \phi}{\partial \theta^2} \tag{16.14} \]

The function \( G \) is defined on an unbounded domain with zero forcing except at a single point, we thus expect \( G \) to depend only on the distance to the point \( x_i \) but not on the angle \( \theta \), so that \( \frac{\partial^2 \phi}{\partial \theta^2} = 0 \).

The two-dimensional form of the Dirac delta function can be built from its one-dimensional counterpart: \( \delta(x) \); the latter is defined as

\[ \delta(x) = \begin{cases} 0 & x \neq 0 \\ \infty & x = 0 \end{cases} \quad \text{with} \quad \int_{-\infty}^{\infty} \delta(x) \, dx = 1. \tag{16.15} \]

In particular we have for any function \( f(x) \) that

\[ \int_{-\infty}^{\infty} f(x)\delta(x - x_0) \, dx = f(x_0) \tag{16.16} \]
One may wonder how to define a function which vanishes everywhere, spikes to infinity at a single point, and yet has unit area under the curve. A simple example is given by the function
\[ \delta_a = \max(0, (a - |x|)/a^2), \]
and taking its limit as \( a \to 0 \). In 2D the definition of the function differs a bit from its one-dimensional counterpart. In particular setting
\[ Q(x, x_i) = \frac{\delta(r)}{2\pi r}, \]
where \( r = |x - x_i| \) is the distance of the point \( x \) from the source point \( x_i \). It is easy to verify that the area under \( Q \) is unity, indeed we have
\[
\int_A Q(x) dA = \int_0^{2\pi} \int_0^\infty \frac{\delta(r)}{2\pi r} r \, dr \, d\theta = 1
\]  
We are now ready to determine our free space Green function. Dropping the azimuthal dependency from the Laplacian in polar coordinates we have
\[
\frac{\partial^2 \phi}{\partial r^2} + \frac{1}{r} \frac{\partial G}{\partial r} = \frac{\delta(r)}{2\pi r} \quad (16.19)
\]
\[
\frac{1}{r} \frac{\partial}{\partial r} \left( \frac{r}{r} \frac{\partial G}{\partial r} \right) = \frac{\delta(r)}{2\pi r} \quad (16.20)
\]
\[
r \frac{\partial G}{\partial r} = \int \frac{\delta(r)}{2\pi} \, dr \quad (16.21)
\]
\[
\frac{\partial G}{\partial r} = \frac{1}{2\pi} + A \quad (16.22)
\]
\[
\frac{\partial G}{\partial r} = \frac{1}{2\pi r} + A \quad (16.23)
\]
\[
G = \frac{\ln |r|}{2\pi} + A \ln |r| + B \quad (16.24)
\]
where \( A \) and \( B \) are integration constants. The constant \( B \) is just an offset to the function, and we can arbitrarily set it to 0. The constant \( A \) can be determined by an application of the divergence theorem on a circle of radius \( R \) we then have
\[
\int_A Q \, dA = \int_A \nabla^2 G dA = \int_{\partial A} \frac{\partial G}{\partial n} ds = \int_0^{2\pi} \frac{\partial G}{\partial r} Rd\theta = 1 + 2\pi A \quad (16.25)
\]
Since the left hand side of the equation is equal to 1, the constant \( A = 0 \).

A final note concerns the case where the source point is located on the boundary of the domain. The integral takes then a slightly different form. Figure 16.1 shows a sketch with the source point on the boundary. The area integral is now divided into a piece that excludes the point plus an infinitesimally small area near the source. The first integral is of course zero while the second leads to
\[
\int_0^\alpha \int_0^r \phi(x) \frac{\delta(r)}{2\pi r} r \, dr \, d\theta = \frac{\alpha}{2\pi} \phi(x_i) \quad (16.26)
\]
16.4. DISCRETIZATION

The discretization of the continuous problem proceeds by first dividing the boundary of the domain into line segments so that the geometry of the domain is well-represented. The choice of line type is rather arbitrary. In the following we will only consider the case where the line segments are straight lines as sketched in figure 16.1. Notice that for linear segments:

\[ r^2 = s^2 + n^2, \quad \text{and} \quad \frac{\partial r}{\partial n} = \frac{n}{r} \]  

(16.29)
The boundary integral can now be changed to a discretization over the individual segments so that equation 16.28 becomes

$$\sum_{j=1}^{N} \int_{s_j}^{s_{j+1}} \left( \frac{n}{n^2 + s^2} - \frac{1}{2} \ln(s^2 + n^2) \frac{\partial \phi}{\partial n} \right) ds = \alpha \phi(x_i) \quad (16.30)$$

The second piece of the discretization is to approximate the variations of \(\phi\) and its normal derivative \(\nabla \phi \cdot n\) on the boundary. Again, different choices lead to different formulations, here we will use a basic linear variations between the nodes so that

$$\phi(s) = \frac{s - s_j}{\Delta s} \phi_{j+1} + \frac{s_{j+1} - s_j}{\Delta s} \phi_j \quad (16.31)$$

$$\frac{\partial \phi}{\partial n}(s) = \frac{s - s_j}{\Delta s} \left. \frac{\partial \phi}{\partial n} \right|_{j+1} + \frac{s_{j+1} - s_j}{\Delta s} \left. \frac{\partial \phi}{\partial n} \right|_{j} \quad (16.32)$$

with \(\Delta s = s_{j+1} - s_j\) being the length of the boundary segment \(j\). Note that the \(\phi_j\) and \(\left. \frac{\partial \phi}{\partial n} \right|_{\cdot}\) are just coefficients that will be chosen to enforce the integral equations.

Replacing the approximations in equation 16.30 we get

$$\sum_{j=1}^{N} \left( \int_{s_j}^{s_{j+1}} \frac{n}{n^2 + s^2} \left[ \frac{s - s_j}{\Delta s} \phi_{j+1} + \frac{s_{j+1} - s_j}{\Delta s} \phi_j \right] ds \right) - \alpha \phi(x_i) =$$

$$\sum_{j=1}^{N} \left( \int_{s_j}^{s_{j+1}} \frac{1}{2} \ln(s^2 + n^2) \left[ \frac{s - s_j}{\Delta s} \left. \frac{\partial \phi}{\partial n} \right|_{j+1} + \frac{s_{j+1} - s_j}{\Delta s} \left. \frac{\partial \phi}{\partial n} \right|_{j} \right] ds \right) \quad (16.33)$$

The above equation is nothing but a row of the matrix equation governing the relationship between \(\phi\) and its normal derivative. A more compact form is:

$$\sum_{j=1}^{N} (K_{i,j} \phi_j + K_{i,j+1} \phi_{j+1}) - \alpha \phi(x_i) = \sum_{j=1}^{N} \left( R_{i,j} \frac{\partial \phi}{\partial n}_{j} + R_{i,j+1} \frac{\partial \phi}{\partial n}_{j+1} \right) \quad (16.34)$$

with

$$K_{i,j} = \frac{n}{\Delta s} \int_{s_j}^{s_{j+1}} \frac{s_{j+1} - s}{n^2 + s^2} ds \quad (16.35)$$

$$K_{i,j+1} = \frac{n}{\Delta s} \int_{s_j}^{s_{j+1}} \frac{s - s_j}{n^2 + s^2} ds \quad (16.36)$$

$$R_{i,j} = \frac{1}{2\Delta s} \int_{s_j}^{s_{j+1}} (s_{j+1} - s) \ln(s^2 + n^2) ds \quad (16.37)$$

$$R_{i,j+1} = \frac{1}{2\Delta s} \int_{s_j}^{s_{j+1}} (s - s_j) \ln(s^2 + n^2) ds \quad (16.38)$$

Notice that we have been able to pull \(n\) out of the integral because it is constant for the straight line approximation of the boundary curves.
16.4.1 Evaluation of Integrals

All the integrals in 16.35-16.38 can be evaluated analytically; indeed it is easy to verify that:

\[
A(s_{j+1}, s_j, n) = \frac{n}{2\Delta s} \int_{s_j}^{s_{j+1}} \frac{s}{n^2 + s^2} \, ds = \left[ \frac{n}{2\Delta s} \ln(n^2 + s^2) \right]_{s_j}^{s_{j+1}}
\]

\[
= \frac{n}{2\Delta s} \ln \frac{r_j^{2+1}}{r_j} \tag{16.39}
\]

\[
B(s_{j+1}, s_j, n) = \frac{n}{2\Delta s} \int_{s_j}^{s_{j+1}} \frac{1}{n^2 + s^2} \, ds = \frac{1}{\Delta s} \left[ \tan^{-1} \frac{s}{n} \right]_{s_j}^{s_{j+1}}
\]

\[
= \frac{1}{\Delta s} \tan^{-1} \frac{s_{j+1}}{n} - \frac{1}{\Delta s} \tan^{-1} \frac{s_j}{n} \tag{16.40}
\]

\[
C(s_{j+1}, s_j, n) = \frac{1}{2\Delta s} \int_{s_j}^{s_{j+1}} s \ln(s^2 + n^2) \, ds
\]

\[
= \frac{1}{4\Delta s} \left[ (s^2 + n^2) \ln(n^2 + s^2) - (s^2 + n^2) \right]_{s_j}^{s_{j+1}}
\]

\[
= \frac{1}{4\Delta s} \left[ (r_{j+1}^2 \ln r_{j+1}^2 - r_j^2) - (r_j^2 \ln r_j^2 - r_j^2) \right] \tag{16.42}
\]

\[
D(s_{j+1}, s_j, n) = \frac{1}{2\Delta s} \int_{s_j}^{s_{j+1}} \ln(s^2 + n^2) \, ds
\]

\[
= \frac{1}{4\Delta s} \left[ \ln(n^2 + s^2) - 2s + 2n \tan^{-1} \frac{s}{n} \right]_{s_j}^{s_{j+1}}
\]

\[
= \frac{1}{2\Delta s} \left\{ \left[ s_{j+1} \ln r_{j+1}^2 - 2s_{j+1} + 2n \tan^{-1} \frac{s_{j+1}}{n} \right] - \left[ s_j \ln r_j^2 - 2s_j + 2n \tan^{-1} \frac{s_j}{n} \right] \right\} \tag{16.43}
\]

\[
= \frac{1}{2\Delta s} \left\{ \left[ s_{j+1} \ln r_{j+1}^2 + 2n \tan^{-1} \frac{s_{j+1}}{n} \right] - \left[ s_j \ln r_j^2 + 2n \tan^{-1} \frac{s_j}{n} \right] \right\} - 1 \tag{16.44}
\]

\[
= \frac{1}{2\Delta s} \left[ s_{j+1} \ln r_{j+1}^2 - s_j \ln r_j^2 + 2n \tan^{-1} \frac{n\Delta s}{1 + s_j s_{j+1}} \right] - 1
\]

where we have used the fact that \(\tan(x-y) = \frac{\tan x - \tan y}{1 + \tan x \tan y}\) to simplify the expressions.

Replacing these expressions into the definition of the matrix entries we have

\[
K_{i,j} = s_{j+1}B(s_{j+1}, s_j, n) - A(s_{j+1}, s_j, n) \tag{16.45}
\]

\[
K_{i,j+1} = -s_jB(s_{j+1}, s_j, n) + A(s_{j+1}, s_j, n) \tag{16.46}
\]

\[
R_{i,j} = s_{j+1}D(s_{j+1}, s_j, n) - C(s_{j+1}, s_j, n) \tag{16.47}
\]

\[
R_{i,j+1} = -s_jD(s_{j+1}, s_j, n) + C(s_{j+1}, s_j, n) \tag{16.48}
\]
Notice that special care must be taken when integrating along an edge that contains the source point, since in that case either \( r_j \) or \( r_{j+1} \) is zero. To account for this case we notice first that when \( n = 0 \) we have

\[
A(s_{j+1}, s_j, 0) = 0
\]

\[
B(s_{j+1}, s_j, 0) = 0
\]

\[
C(s_{j+1}, s_j, 0) = \frac{1}{4\Delta s} \left[ (s_{j+1}^2 \ln s_{j+1}^2 - s_j^2) - \left( s_j^2 \ln s_j^2 - s_{j+1}^2 \right) \right]
\]

\[
D(s_{j+1}, s_j, 0) = \frac{1}{2\Delta s} \left[ s_{j+1} \ln s_{j+1}^2 - s_j \ln s_j^2 \right] - 1
\]

In a second step it is now easy to take the limit when \( s_j \) or \( s_{j+1} \) are zero, since \( x \ln x \to 0 \) when \( x \to 0 \), and so we have:

\[
C(0, s_j, 0, 0) = \frac{s_j(2 \ln s_j - 1)}{4}
\]

\[
C(s_{j+1}, 0, 0) = \frac{s_{j+1}(2 \ln s_{j+1} - 1)}{4}
\]

\[
D(0, s_j, 0) = \ln s_j - 1
\]

\[
D(s_{j+1}, 0, 0) = \ln s_{j+1} - 1
\]

16.5 Algorithm

The development of a code for solving the Laplace equation on arbitrary domain using BEM requires several steps:

1. Description of the geometry
2. Description of the boundary conditions
3. Building the matrices mostly by evaluating integrals
4. Constructing of the linear system by assembly
5. Solution of the linear system
6. Evaluating the solution at interior points
7. Plotting the solution and displaying it.

We will take up each of these tasks in the following subsections.

16.5.1 Geometry description

The geometry is essentially formed by a collection of nodes connected by edges. We can already envision a derived type called bemnode and a derived type called polygon made-up of a collection of bemnodes. The location of the nodes of each polygon must be read from files that contain the list of nodes and their attributes.
bemnode data type and functionality

The bemnode data type must provide the following information:

1. The coordinates of the node list in an array of dimension $\text{sdim}$; here $\text{sdim}$ is parameter equal to 2 for 2D problems.

2. The type of boundary condition applied at the node. We will code that with an integer data type called $\text{bctype}$.

3. The value of the function and its normal derivative at the node. If these are not known, they should be set to 0.

4. Finally we need a code to tell whether the node is a corner that is if the boundary condition type switches from Neumann to Dirichlet, or from Dirichlet to Neumann, or if the type does not switch at all.

I would like to stress that the actual implementation of bemnode must be all encapsulated in a module, and that the private data should only be accessible via the functionality provided. This functionality must be able to do the following:

1. Count the number of nodes in a file

2. Allocate the list of nodes to be read

3. Read the nodes contained in the file and their attributes

4. Print the list of nodes.

5. Count the number of nodes in a file

6. Be able to query and modify the attributes, e.g.
   
   (a) return the coordinate of a node in an array
   
   (b) return the boundary condition code
   
   (c) return the corner condition code
   
   (d) return the values of the function and/or its derivative

bempolygon data type and functionality

A polygon must contain the number of nodes it is made-up of, and a list of the nodes that compose it. This list consists of an allocatable array (since we don’t know a priori the number of nodes a polygon contains) of data type bemnode. It may be useful for polygon to include also a list of edges that connect the nodes.
16.5.2 Geometry module

It is clear that the development of the code will require a module that include some amount of analytical geometry manipulation. These include the calculation of the angle between 2 vectors, the inner product and the vector product of two vectors. The use of the following equations will be useful:

\[
\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 \quad (16.57)
\]

\[
\mathbf{u} \times \mathbf{v} = u_1 v_2 - u_2 v_1 \quad (16.58)
\]

\[
\|\mathbf{u}\| = \sqrt{u_1^2 + u_2^2} \quad (16.59)
\]

\[
\sin \theta = \frac{\mathbf{u} \times \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} \quad (16.60)
\]

\[
\cos \theta = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} \quad (16.61)
\]

The normal distance of a point \( S \) to the edge spanned by the points \( A \) and \( B \) is given by

\[
n = \frac{\mathbf{AS} \times \mathbf{AB}}{\|\mathbf{AB}\|} = (x_s - x_a) \sin \beta - (y_a - y_s) \cos \beta \quad (16.63)
\]

where

\[
\cos \beta = \frac{x_b - x_a}{\Delta s}, \quad \sin \beta = \frac{y_b - y_a}{\Delta s}, \quad \Delta s = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2} \quad (16.64)
\]

The point of intersection of the edge and the normal is given by

\[
x_h = (y_s - y_a) \cos \beta \sin \beta - (x_s - x_a) \sin^2 \beta + x_s \quad (16.65)
\]

\[
y_h = (x_s - x_a) \cos \beta \sin \beta - (y_s - y_a) \cos^2 \beta + y_s \quad (16.66)
\]

The distance \( s_a \) from \( A \) to \( H \) and \( s_b \) from \( B \) to \( H \) are given by

\[
s_A = \frac{\mathbf{SA} \cdot \mathbf{AB}}{\|\mathbf{AB}\|} = (x_a - x_s) \cos \beta + (y_a - y_s) \sin \beta \quad (16.67)
\]

\[
s_B = \frac{\mathbf{SB} \cdot \mathbf{AB}}{\|\mathbf{AB}\|} = (x_b - x_s) \cos \beta + (y_b - y_s) \sin \beta \quad (16.68)
\]