USER'S GUIDE TO A BOUNDARY ELEMENT PACKAGE FOR SOLVING INTEGRAL EQUATIONS ON PIECEWISE SMOOTH SURFACES

Release #2

Kendall E. Atkinson
Department of Mathematics
University of Iowa
Iowa City, Iowa 52242, USA

January 5, 1998

1 OVERVIEW

This guide describes a collection of programs for (1) creating and refining triangulations on surfaces, and (2) solving integral equations using collocation methods over these triangulations. The main purpose of this package is to allow for experimentation with numerical methods for solving boundary integral equations that are defined on piecewise smooth surfaces in $\mathbf{R}^3$. General surveys of the numerical solution of boundary integral equation (BIE) reformulations of Laplace's equation are given in [5] and [9], and an introduction to the numerical solution of such BIE is given in [8, Chap. 9].

Our package is restricted to triangulations which are "uniform", as there are additional difficulties in dealing with meshes that are suitably graded. We have included a few routines to create graded meshes, to allow for numerical integration of functions which have poor behaviour near edges and corners of the surface. These can also be used for setting up graded meshes for solving boundary integral equations in which the unknown solution function has poor behaviour near edges and corners of the surface.

An additional feature of the second release of the package is the treatment of the "radiosity equation" from computer graphics and radiative transfer. It is defined and discussed in §7, and we give some programs for its numerical solution.

1.1 Framework for Triangulation

The method of triangulation used in this package is described and used in [8, Chap. 5]. It works well with a wide variety of surfaces, including polyhedra, smooth surfaces such as ellipsoids and tori, cones, paraboloids, and many others. The intended application is to the solution of integral equations over the boundary surfaces of connected regions in $\mathbf{R}^3$, although the triangulation package can also be used with open surfaces.

All triangulations $\mathcal{T}_N = \{\Delta_k \mid k = 1, \cdots, N\}$ of a surface S are given by means of three arrays, named IFACE, VERTEX, and INDEXV. The integer array IFACE contains the triangles of the triangulation, the double precision array VERTEX contains the nodes of the triangulations, and the integer array INDEXV contains additional information about the nodes in VERTEX.

Each triangular element Δ_k has six nodes associated with it: three vertices, called $v_{k,1},\ v_{k,2},\ v_{k,3}$, and three "midpoints", called $v_{k,4},\ v_{k,5},\ v_{k,6}$, which are approximate midpoints of the 3 sides of the triangular element. These are arranged as shown in Figure 1. The array IFACE contains in column k the information which defines Δ_k, $k = 1, \cdots, N$. The entries IFACE(1,k),$\cdots$,IFACE(6,k) contain the position indices of $v_{k,1}, \cdots, v_{k,6}$, relative to the array VERTEX. The entry IFACE(7,k) contains additional information about the location of Δ_k relative to the original surface. It is often convenient to decompose piecewise smooth surfaces into sub-surfaces. If the latter are indexed in some way, then IFACE(7,k) can be used to carry the information as to which sub-surface contains Δ_k. For example, a cube would ordinarily be divided into six sub-surfaces, and the location of each Δ_k relative to this decomposition is given in IFACE(7,k). The array IFACE is dimensioned as

```
DIMENSION IFACE (7, MXFACE)
```

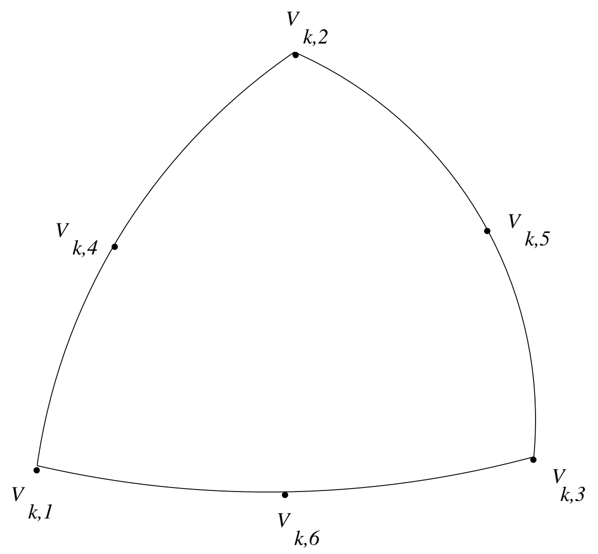

Figure 1: The element Δ_k and its nodes $v_{k,j}$

with MXFACE being the maximum number of faces to be allowed. Also, the order of the vertices which are given in column k of IFACE will also give the direction of a normal to Δ_k. If the vertices $v_{k,1},\ v_{k,2},\ v_{k,3}$ are used with the right-hand rule, then this will be taken as indicating the direction of the interior normal to Δ_k when it is considered as a portion of a closed surface S.

The array VERTEX contains the nodes of the triangulation (the vertices and midpoints of the triangular elements in the triangulation). The dimension statement is

DIMENSION VERTEX (3, MXVERT)

with MXVERT the maximum number of vertices to be allowed. For closed surfaces with conforming triangulations of the type created in this package, the number of faces N and the number of nodes N_v are related by.

$$N_v = 2(N + 1)$$

The nodes are referred to in two ways: as $v_{k,j}$ for node j in the triangular element Δ_k; and as v_i with $1 \le i \le N_v$, when we are referring to the nodes collectively and not with reference to any triangular element. The three components of the node $v_i = (x_i, y_i, z_i)$ are stored in the array entries VERTEX(1,i), VERTEX(2,i), and VERTEX(3,i), respectively. We let $\mathcal{V}_N = \{v_i \mid i = 1, \cdots, N_v\}$ denote the nodes collectively.

The one-dimensional array INDEXV, of length MXVERT, is used to describe the nature of the node points in VERTEX. Consider S as being a piecewise smooth surface, and suppose it can be decomposed into sub-surfaces $S_1, \cdots . S_L$ with each S_l a smooth sub-surface. Each node point v_i can be classified as follows: (1) v_i belongs to the interior some S_l; or (2) v_i in contained in an edge of such a S_l, but is not a vertex of S; or (3) v_i is a vertex of S. These three cases lead to INDEXV(i) = 0, 1, 2, respectively. On a smooth surface, INDEXV(i) = 0 for $i = 1, \cdots, N_v$.

1.1.1 The structure of the triangulation process

The triangulation is initiated with a call to a subroutine INIT which is to be supplied by the user. The routine INIT gives an initial triangulation of S, supplying the initial values for IFACE, VERTEX, and INDEXV. More detailed information on the requirements of writing INIT are given later in §3, along with examples for many common surfaces. To refine the triangulation, we proceed as follows. Connect the midpoints of the sides of each Δ_k. This will form four new smaller triangular elements, as is illustrated in Figure 2. The midpoints of their sides can then be constructed by calling on a second user-supplied subroutine, called MIDPT. It too is described further in §3. The programs actually used to carry out the refinement process, including the definition of the arrays IFACE, VERTEX, and INDEXV for the new refinement, are given in the triangulation package which we will refer to as TRIPACK. Its contents are described in §2.1.The most important routine in the refinement process is the subroutine REFINE (and REFIN2 when doing two-grid iteration methods).

In addition to the refinement of the surface, other routines are given to aid with other needs of working with a triangulation of the surface. For example, the subroutine CIRCLE calculates for each triangle Δ_k the center C_k and radius r_k of the circumscribing circle for the planar triangle that joins the three vertices of Δ_k. This can be used to check whether or not a given

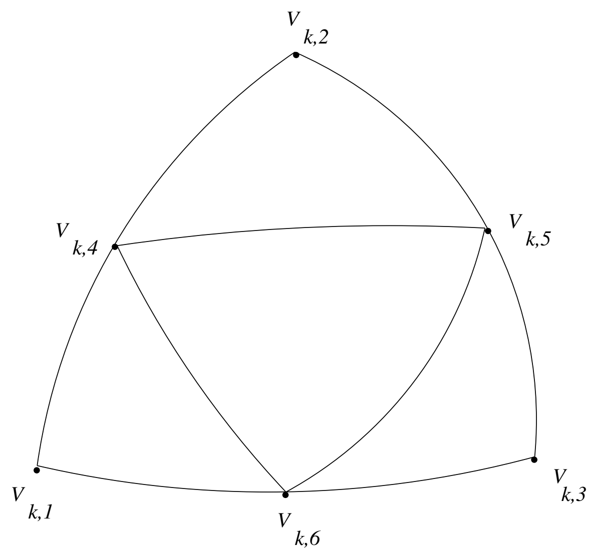

Figure 2: The refinement of Δ_k

node or point P belongs to a given element Δ_k. Merely compare $|P - C_k|$ and r_k : if $|P - C_k| > r_k$, then $P \notin \Delta_k$. In addition to CIRCLE, also consider the routines ORIENT, DIAMTR, NORMAL, FNDNRM, FNDNRM2, NRM_AV, POLYPREP, LOCATE, F_TO_V, and PRNTRI, given in the collection named TRIPACK.

1.1.2 Interpolation over the surface

The methods we use for solving boundary integral equations are based on the collocation method with approximating functions which are piecewise polynomial, in a certain sense, over the triangulation $\mathcal{T}_N$. We use approximating functions that are piecewise polynomial in the parameterization variables, of degrees 0, 1, and 2. These functions are referred to as 'piecewise constant', 'piecewise linear', and 'piecewise quadratic', respectively.

Introduce

$$\sigma = \{(s, t) \mid 0 \leq s, t, s + t \leq 1\}$$

Each triangular element Δ_k is considered to be the image of a mapping $m_k : \sigma \underset{onto}{\overset{1-1}{\rightarrow}} \Delta_k$, making σ the parametrization domain for Δ_k. For piecewise constant interpolation over $\mathcal{T}_N$ to a function $f \in C(S)$, define

$$\mathcal{P}_N f(P) = f\left(m_k\left(\tfrac{1}{3}, \tfrac{1}{3}\right)\right) \qquad (s,t) \in \sigma, \qquad P = m_k(s,t)$$

for $k = 1, ..., N$. The function $\mathcal{P}_N f$ is piecewise constant over the surface S. There is a problem of multiple definition of $\mathcal{P}_N f(P)$ at points P on the boundary of the various elements Δ_k, and we ignore this by generally not evaluating the interpolation function at such points. As an alternative, we could use an average of the values obtained from all of the triangles containing the given P.

For piecewise linear interpolation over S with the triangulation $\mathcal{T}_N$, we first introduce the following interpolation node points in σ:

$$q_1(\alpha) \;=\; (\alpha, \alpha)\,, \qquad q_2(\alpha) \;=\; (\alpha, 1-2\alpha)\,, \qquad q_3(\alpha) \;=\; (1-2\alpha, \alpha)$$

with $0 \le \alpha < \frac{1}{3}$. For $\alpha = 0$, the three nodes are the vertices of σ, and then

$$m_k\left(q_i(0)\right) = v_{k,i}, \qquad i = 1,2,3$$

For $0 < \alpha < \frac{1}{3}$, the node points are interior to σ. Introduce the linear basis functions

$$l_1(s,t) \;=\; \frac{u-\alpha}{1-3\alpha}, \quad l_2(s,t) \;=\; \frac{t-\alpha}{1-3\alpha}, \quad l_3(s,t) \;=\; \frac{s-\alpha}{1-3\alpha}$$

with $u = 1-s-t$ for $(s,t) \in \sigma$. It is straightforward to show $l_i\left(q_j(\alpha)\right) = \delta_{i,j}$, for $i, j = 1, 2, 3$.

To interpolate a function $f \in C(\Delta_k)$, define

$$\mathcal{P}_N f(P) = \sum_{j=1}^{3} f(m_k(q_j(\alpha))\ell_j(s,t), \qquad (s,t) \in \sigma, \qquad P = m_k(s,t) \qquad (1)$$

With the above properties, we have $\mathcal{P}_N f(m_k(q_i(\alpha)) = f(m_k(q_i(\alpha)),\ i = 1,2,3$. The function $\mathcal{P}_N f(m_k(s,t))$ is linear in the parametrization variables (s,t) for Δ_k. Extend this definition to all of the elements Δ_k of the triangulation $\mathcal{T}_N$, and refer again to the resulting function as $\mathcal{P}_N f$. We call $\mathcal{P}_N f$ a piecewise linear interpolation function (somewhat incorrectly). If $\alpha = 0$,

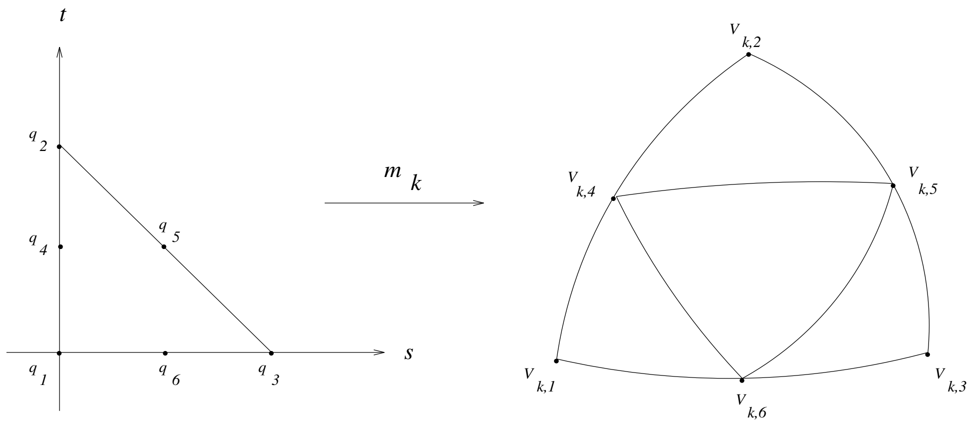

Figure 3: The parametrization of Δ_k for quadratic interpolation

then $\mathcal{P}_N f$ is continuous over S, and $\mathcal{P}_N f(v_{k,i}) = f(v_{k,i}),\ i = 1, 2, 3$. For $0 < \alpha < \frac{1}{3}$, $\mathcal{P}_N f(P)$ is generally not continuous at points P on the boundary of the elements Δ_k.

To define piecewise quadratic interpolation, introduce the quadratic basis functions $\{\ell_j\}$:

$$\begin{array}{rclrclrcl} l_1(s,t) &=& u(2u-1), & l_2(s,t) &=& t(2t-1), & l_3(s,t) &=& s(2s-1) \\ l_4(s,t) &=& 4tu, & l_5(s,t) &=& 4st, & l_6(s,t) &=& 4su \end{array} \tag{2}$$

Introduce the points

$$\begin{array}{rclrclrcl} q_1 &=& (0,0) & q_2 &=& (0,1) & q_3 &=& (1,0) \\ q_4 &=& (0,.5) & q_5 &=& (.5,.5) & q_6 &=& (.5,0) \end{array} \tag{3}$$

Then

$$\ell_j(q_i) = \delta_{ij}, \quad i,j = 1, \cdots, 6$$

These are quadratic basis functions, for quadratic interpolation over σ at the interpolation points $\{q_{1,\cdots,}q_6\}$. For a pictorial representation of the relationship of σ, Δ_k, $\{v_{k,j}\}$, and $\{q_j\}$, see Figure 3. The values of $\{\ell_j(s,t)\}$ at a given (s,t) are given by the subroutine BASIS.

To interpolate a function $f \in C(\Delta_k)$, define

$$\mathcal{P}_N f(P) = \sum_{j=1}^{6} f(v_{k,j})\ell_j(s,t), \qquad (s,t) \in \sigma, \qquad P = m_k(s,t) \tag{4}$$

With the above properties, we have $\mathcal{P}_N f(v_{k,i}) = f(v_{k,i}),\ i = 1, \cdots, 6$. The function $\mathcal{P}_N f(m_k(s,t))$ is quadratic in the parametrization variables (s,t) for Δ_k. Extend this definition to all of the elements Δ_k of the triangulation $\mathcal{T}_N$, and again refer to the resulting function as $\mathcal{P}_N f$. We call $\mathcal{P}_N f$ a piecewise quadratic interpolation function, somewhat incorrectly. The routine BASIS calculates the basis functions $\{\ell_j\}$; and the function INTERP carries out the interpolation over a given Δ_k. Piecewise quadratic interpolation is the principal form of approximation used in our package, although some routines are also included for piecewise constant and piecewise linear interpolation.

For $f \in C(S)$, and for all the above forms of piecewise polynomial interpolation, it is easy to show that $\|f - \mathcal{P}_N f\|_\infty \to 0$ as $N \to \infty$. Let

$$h \equiv h_N = \max_{1 \le k \le N} \ \mathit{diameter}(\Delta_k)$$

If f is sufficiently smooth, and if the sub-surfaces of S are sufficiently smooth, then $\|f - \mathcal{P}_N f\|_\infty = O(h^{d+1})$ with d the degree of the interpolation being used, $d = 0, 1, 2$. For the piecewise linear case, the choice of $\alpha = \frac{1}{6}$ is often an optimal choice when solving integral equations and when creating a numerical integration formula from the interpolation formula. For additional detail, see [12]. For general degrees $d \ge 0$, more information about the convergence of $\mathcal{P}_N f$ is given in [8, Chap. 5]. The above interpolation operators $\mathcal{P}_N$ are used below in defining the collocation method for solving boundary integral equations. It should be clear from the context as to which degree of interpolation is being used and therefore which definition of $\mathcal{P}_N$ to use.

1.1.3 Approximation of the surface by interpolation

When doing numerical integration over the surface, as is needed in implementing the collocation method for solving integral equations, we need to calculate integrals

$$\int_{\Delta_k} f(Q)\, dS_Q = \int_\sigma f(m_k(s,t))\ |D_s m_k \times D_t m_k|\, d\sigma \tag{5}$$

In this, $D_s m_k(s,t) = \partial m_k(s,t)/\partial s$, and similarly for $D_t m_k(s,t)$. If the surface S is polyhedral, then calculating the Jacobian $|D_s m_k \times D_t m_k|$ is straightforward. But when the surface S is curved, calculating the Jacobian requires an explicitly differentiable parametrization of S. To avoid this inconvenience, we approximate $m_k(s,t)$ by quadratic interpolation.

Define

$$\widetilde{m}_k(s,t) = \sum_{j=1}^{6} v_{k,j}\, \ell_j(s,t), \qquad (s,t) \in \sigma \tag{6}$$

use the quadratic basis functions $\{\ell_j\}$ of (2). This mapping satisfies $\widetilde{m}_k(q_j) = v_{k,j},\ j = 1, \cdots, 6$. We let $\widetilde{\Delta}_k = \widetilde{m}_k(\sigma),\ k = 1, \cdots, N$; and

$$\widetilde{S}_N = \bigcup_{k=1}^{N} \widetilde{\Delta}_k$$

The triangular elements Δ_k and $\widetilde{\Delta}_k$ agree at the nodes $\{\, v_{k,1}, \cdots, v_{k,6}\}$. The routine QUADSF calculates $\widetilde{m}_k(s,t)$.

Approximate the integral (5) by

$$\int_{\Delta_k} f(Q)\, dS_Q \approx \int_{\sigma} f\left(\widetilde{m}_k(s,t)\right)\, |D_s\widetilde{m}_k \times D_t\widetilde{m}_k|\, d\sigma \tag{7}$$

This approximation is analyzed in considerable detail in Chien [13]-[14] and is discussed further in [8, §5.3]. With the refinement procedure which we use, it can be shown that the error in (7) is of order h^4, rather than the order h^2 which might be expected on the basis of using the derivative of a quadratic interpolant. Using $\widetilde{m}_k(s,t)$ considerably simplifies integrations over triangular elements Δ_k.

In the case the surface S is polyhedral, the functions m_k are affine mappings, $\widetilde{m}_k = m_k$, and the Jacobian satisfies

$$|D_s\widetilde{m}_k \times D_t\widetilde{m}_k| = 2 \cdot \text{Area}\left(\Delta_k\right), \qquad k = 1, ..., N$$

An option for such polyhedral surfaces is given with most of the numerical integration and integral equation programs in the package. Also, there is a subroutine POLYPREP which is used to pre-process the surface triangulation $\mathcal{T}_N$ for later use by other programs in the package.

1.1.4 Numerical integration over the surface

Many numerical integration routines are included in TRIPACK, to be used for the wide variety of integrals which must be calculated. We list the four main forms of integration used over the standard parametrization region σ. The first three formulas refer to the nodes $\{q_i\}$ of (3).

$$\int_\sigma g(s,t)\,d\sigma \approx \frac{1}{6}\left[g(q_1)+g(q_2)+g(q_3)\right] \tag{8}$$

$$\int_\sigma g(s,t)\,d\sigma \approx \frac{1}{6}\left[g(q_4)+g(q_5)+g(q_6)\right] \tag{9}$$

$$\begin{aligned}\int_\sigma g(s,t)\,d\sigma \approx \tfrac{9}{40}g\left(\tfrac{1}{3},\tfrac{1}{3}\right)+\tfrac{1}{40}\left[g(q_1)+g(q_2)+g(q_3)\right]\\ +\tfrac{1}{15}\left[g(q_4)+g(q_5)+g(q_6)\right]\end{aligned} \tag{10}$$

$$\int_\sigma g(s,t)\,d\sigma \approx \sum_{j=1}^{7} w_j g(\rho_j) \tag{11}$$

with the weights and nodes for (11) taken from the formula T2:5-1 of Stroud [21, p. 314]. For the latter, the weights $\{w_j\}$ and nodes $\{\rho_j\}$ are given by the routine SMPLXR; and the integration over σ is carried out in the routines SMPLX, SMPLX3, SMPLX6, and TRI_INT. The degrees of precision over σ of these four formulas are respectively 1, 2, 3, and 5. We apply these formulas to the calculation of integrals over Δ_k by means of (7). An alternative to formulas (10) and (11) is given in SMPLXL, giving the weights and nodes over σ of a 6-point formula with degree of precision 4.

For routines which are used just for numerical integration over the surface S, see the programs INTEGRATE and INTEGRATE_PL. The first is based on the method (9), and the second is based on (8). Both use the piecewise quadratic approximation of the surface as defined in (6).

When applying (8) or (9) to the right side of (7), the weights of the integration are given by the routines WGT1 and WGT, respectively, incorporating the quadratic surface approximation of (6). When applying (11) to the right side of (7), we use the routines SMPLX, SMPLX3, SMPLX6, and TRI_INT, depending on the intended application. We seldom use (10), but it is a good formula to use when the refinement method for the triangulation does not have the symmetry of that described in §1.1.1.

When the integrand in (7) is considered to be badly behaved, we use composite forms of (11). This occurs principally with integrands formed from $1/|P-Q|$, with P near to but not in the integration triangle Δ_k. In such cases, we subdivide the parametrization region σ, and then (11) is used over each subdivision. To decide on the number of subdivisions needed, one of the routines SETLEV and SETLV2 is used.

Some of the integrals that arise in solving integral equations will have a singular integrand, and we handle such integrals with a change of variables. For integrands of the type occurring with boundary integral equations, the new integrand will generally be quite smooth. The transformed integral is then approximated using Gauss-Legendre quadrature with respect to both variables. The needed integrations are carried out with the aid of the routines INTEGR, INTGR2, INTEGR3, INTEGR3G, TRI_SNG, and RECT_SNG, with the choice varying with the particular application.

There are many types of integral equations which we wish to consider, and within the same equation, there may be more than one type of collocation integral which must be evaluated. The many quadrature routines in §2.5 are designed for these various types of integration. They use the above quadrature schemes, together with the routines we have mentioned already. In addition, all the numerical integration programs incorporate the approximation of the surface given in §1.1.3. These programs also incorporate an option for polyhedral boundaries, to avoid the calculations associated with the approximation of the surface, since the approximation would be exact for such boundaries.

1.2 The Solution of Integral Equations

The routines in this package can be used to set up the numerical solution of a wide variety of linear and nonlinear integral equations defined on piecewise smooth surfaces S in $\mathbf{R}^3$. We have developed general programs for a number of these equations, although not all programs we have developed are yet in the package. Most of our programs are for integral equations of the second kind, with most of them relating to the solution of boundary integral equations and the radiosity equation.

1.2.1 Integral equations with smooth kernel functions

Consider the integral equation

$$\lambda\,\rho(P) - \int_S K(P,Q)\,\rho(Q)\,dS_Q = g(P), \quad P \in S \tag{12}$$

with $\lambda \neq 0$. Symbolically, we will write this equation as

$$(\lambda - \mathcal{K})\rho = g \tag{13}$$

The smoothness of solutions to this equation depends on that of S, K, and g, and we give more precise assumptions in §5.1. We have programs to solve this integral equation in several ways. We use the collocation method with piecewise quadratic approximation, as defined earlier in §1.1.2. We also use the Nyström method with the numerical integration rule based on (9). Variants of the latter are given for both direct solution of the associated linear system and the iterative solution of the linear system using a two-grid iteration method.

1.2.2 Integral equations from potential theory for Laplace's equation

We include programs for solving the interior Dirichlet problem and the exterior Neumann problem for Laplace's equation, defined as follows.

P1. The interior Dirichlet problem. Let D be a bounded, open, simply connected region in $\mathbf{R}^3$, and let its boundary S be piecewise smooth, which is defined more precisely in Section 5. The problem is to find $u \in C(\bar{D}) \cap C^2(D)$ such that

$$\Delta u(P) = 0, \quad P \in D$$
$$u(P) = f(P), \quad P \in S$$

We assume u can be represented as a double layer potential:

$$u(P) = \int_S \rho(Q)\,\frac{\partial}{\partial \boldsymbol{\nu}_Q}\left[\frac{1}{|P-Q|}\right] dS_Q, \quad P \in D \tag{14}$$

The density function ρ is determined from the integral equation

$$2\pi\rho(P) + \int_S \rho(Q)\,\frac{\partial}{\partial \boldsymbol{\nu}_Q}\left[\frac{1}{|P-Q|}\right] dS_Q + [2\pi - \Omega(P)]\rho(P) = f(P), \quad P \in S \tag{15}$$

For notation, $\boldsymbol{\nu}_Q$ denotes the unit normal to S at Q (if it exists), pointing into D. The quantity $\Omega(P)$ is the inner solid angle of S at $P \in S$; and we assume

$$0 < \Omega(P) < 4\pi.$$

Symbolically, we write the integral equation (15) as

$$(2\pi + \mathcal{K})\rho = f$$

Under suitable assumptions on S,

$$\mathcal{K} : C(S) \to C(S)$$

is a bounded linear operator. This is discussed at length in Wendland [22].
P2. The exterior Neumann problem. Let D and S be as above, and let $D_e = \mathbf{R}^3 \backslash \bar{D}$, the region exterior to D and S. The problem is to find $u \in C(\bar{D}_e) \cap C^2(D_e)$ such that

$$\Delta u(P) = 0, \quad P \in D_e$$

$$\frac{\partial u(P)}{\partial \boldsymbol{\nu}_P} = f(P), \quad P \in S$$

$$u(P) = O(|P|^{-1}), \quad |\nabla u(P)| = O(|P|^{-2}) \quad \text{as} \quad |P| \to \infty$$

It can be shown that such a function u exists (under suitable assumptions on S and f) and that Green's third identity can be applied to u:

$$4\pi u(P) = \int_S f(Q) \frac{1}{|P-Q|} dS_Q - \int_S u(Q) \frac{\partial}{\partial \boldsymbol{\nu}_Q} \left[\frac{1}{|P-Q|} \right] dS_Q, \quad P \in D_e \tag{16}$$

To find u on S, we solve the integral equation

$$2\pi u(P) + \int_S u(Q) \frac{\partial}{\partial \boldsymbol{\nu}_Q} \left[\frac{1}{|P-Q|} \right] dS_Q + [2\pi - \Omega(P)]u(P) \tag{17}$$

$$= \int_S f(Q) \frac{1}{|P-Q|} dS_Q, \quad P \in S$$

Then (16) gives u on D_e. Symbolically, we write (17) as

$$(2\pi + \mathcal{K})u = \mathcal{S} f \tag{18}$$

with $\mathcal{K}$ as before and $\mathcal{S}$ the single layer potential integral operator.

An alternative approach to solving the exterior Neumann problem is obtained by representing the solution as a single layer potential:

$$u(P) = \int_S \frac{\rho(Q)}{|P-Q|}\, dS_Q, \quad P \in D \tag{19}$$

Then the single layer density ρ must satisfy the second kind equation

$$(2\pi + \mathcal{K})\rho = f \tag{20}$$

This approach does not require the numerical integration needed on the right side of (18), but the solution ρ must then be processed further using (19).

Programs are included for both of these problems. In the case of the interior Dirichlet problem, we approximate the density function ρ, and then we provide for evaluating the resulting solution $u(P)$ at user provided points P in D. For the exterior Neumann problem, the program solves only for $u(P)$ at points $P \in S$. The needed numerical integration programs to evaluate the various integrals at points P both on and off of the surface S are provided in INTPACK. Variants of some of the integral equation programs are provided to either solve the discretized linear systems directly by Gaussian elimination or by two-grid iteration.

1.2.3 The radiosity equation

The *radiosity equation* is a mathematical model for the brightness of a collection of one or more surfaces when their reflectivity and emissivity are given, and it occurs in both computer graphics and heat transfer. The equation is

$$u(P) - \frac{\rho(P)}{\pi} \int_S u(Q) G(P,Q) V(P,Q)\, dS_Q = E(P), \qquad P \in S \tag{21}$$

with $u(P)$ the "brightness" or *radiosity* at P and $E(P)$ the *emissivity* at $P \in S$. The function $\rho(P)$ gives the *reflectivity* at $P \in S$, with $0 \le \rho(P) < 1$. In deriving this equation, the reflectivity is assumed to be independent of the angle at which the reflection takes place, that is the surface is a *Lambertian diffuse reflector*.

The function G is given by

$$G(P,Q) = \frac{\cos\theta_P \cos\theta_Q}{|P-Q|^2} = \frac{[(Q-P)\cdot\mathbf{n}_P]\,[(P-Q)\cdot\mathbf{n}_Q]}{|P-Q|^4} \tag{22}$$

In this, $\mathbf{n}_P$ is the inner unit normal to S at P, and θ_P is the angle between $\mathbf{n}_P$ and $Q-P$; and $\mathbf{n}_Q$ and θ_Q are defined analogously. The function $V(P,Q)$ is a "line of sight" function. More precisely, if the points P and Q can "see each other" along a straight line segment which does not intersect S at any other point, then $V(P,Q)=1$; and otherwise, $V(P,Q)=0$. An *unoccluded* surface is one for which $V\equiv 1$ on S, and it is this case for which we provide programs. Note that S need not be connected, and it may be only piecewise smooth. The interior surface of a convex solid is unoccluded; but one can also be dealing with disconnected surfaces.

We often write (21) in the simpler form

$$(I-\mathcal{K})u = E \tag{23}$$

An introduction to the use of (21) in computer graphics is given in [15] and [20], along with methods for its numerical solution. The unoccluded case is of lessor importance in applications, and we intend to include programs for occluded surfaces in future extensions of BIEPACK.

1.2.4 Iteration methods for solving the discretized linear systems

The numerical approximation of integral equations on surfaces in $\mathbf{R}^3$ leads easily to linear systems of quite large order. With both the integral equations having smooth kernel functions and those arising from solving boundary integral equations, we provide programs which implement two-grid iteration methods for solving the associated linear systems. For the case of integral equations having smooth kernel functions, this is a straightforward application of techniques discussed earlier in [8, Chap. 6].

With discretizations of the boundary integral equations (15) and (17), the iteration methods used are discussed in [6] and [8, Chap. 9]. When the surface S is smooth, then the standard two-grid iteration methods are quite satisfactory. But with surfaces S that are only piecewise smooth, the two-grid methods must be modified, by multiplying the linear system by a 'preconditioner'. This is discussed further in §8.

2 SUMMARY OF GENERAL TOOLKITS

We provide three general "toolkit" packages named AUXPACK, INTPACK, and TRIPACK. These are used in putting together programs to solve a wide variety of programs for numerical integration, refinement and manipulation of triangulations, and numerical solution of integral equations.

In brief, AUXPACK are general routines needed for general purposes, primarily setting machine constants, solving linear systems, and the creation of Gaussian quadrature nodes and weights. INTPACK contains programs for specific integrations associated with particular types of integral equations. TRIPACK contains programs for refinement and manipulation of triangulations of S. It also contains programs to do interpolation and numerical integration over such triangulations.

2.1 Contents of TRIPACK

We give a very short description of each of the subprograms in the triangulation package TRIPACK. These programs can be used with a variety of main programs, which are discussed elsewhere in this document. We group the various subprograms by function.

2.1.1 Refinement of the Triangulation.

1. SUBROUTINE REFINE (NFACE, NVERT, IFACE, VERTEX, INDEXV)
   This program takes a triangulation, defined by the input parameters, and it creates a new triangulation by subdividing each triangle into four new triangles.

2. SUBROUTINE REFIN2 (NFACE, NVERT, IFACE, VERTEX, INDEXV, F_TO_C, STFACE)
   This variant on REFINE is used when two-grid iteration methods are being used in solving a boundary integral equation. It assigns to each

new fine mesh element its parameters relative to the coarse mesh element containing it.

3. SUBROUTINE SUBDIV (IDX, IFT, VRTX, IVX, IFACE, VERTEX, INDEXV)
   This routine subdivides a triangle into four new triangles. The portion of the work which is dependent on the particular surface is done in the user given subroutine MIDPT.

4. SUBROUTINE VRTXLC (V, IDX, NVERT, IVX, VERTEX, INDEXV)
   For the given vertex V, find whether it is already in the known nodes in VERTEX. If not, then add it to the list and assign it an index.

5. SUBROUTINE ADAPT (NFACE, NVERT, IFACE, VERTEX, INDEXV)
   This program takes a triangulation, defined by the input parameters, and it creates a new adaptive triangulation. It subdivides each triangle which contains either a node on an edge or a vertex of the original surface.

6. SUBROUTINE ADAPT2 (NFACE, NVERT, IFACE, VERTEX, INDEXV, F_TO_C, STFACE)
   This variant on ADAPT is used when two-grid iteration methods are being used in solving a boundary integral equation. It assigns to each new fine mesh element its parameters relative to the coarse mesh element containing it.

2.1.2 Additional Information on the Triangles in the Triangulation.

1. SUBROUTINE ORIENT (IFACE, NFACE, VERTEX, NVERT, P)
   This program chooses an orientation for each triangular face so that the normal to the planar triangle determined by the three vertices will point towards the given vector P.

2. SUBROUTINE CIRCLE (IFACE, NFACE, VERTEX, RADSQR, CENTER)
   This routine calculates the center and the square of the radius of the

circumscribing circle for the planar triangle determined by each triangular face.

3. SUBROUTINE DIAMTR (IFACE, NFACE, VERTEX, H_MIN, H_MAX)
   This routine calculates the maximum and minimum diameter of the triangles in the triangulation.

4. SUBROUTINE NORMAL (IFACE, VERTEX, NVERT, NFACE, VNRM)
   Calculate the normals to all faces in IFACE, at all six nodes in each face. This is intended for only the case of piecewise quadratic interpolation with the node points based on the parameterization variables (3). The normal is obtained is based on the approximate surface defined using (6).

5. SUBROUTINE FNDNRM (IFACE, V, K, J, VN)
   Calculate the "inner normal" to the surface S at the node #J in element #K. This routine is called by NORMAL. This is intended for only the case of piecewise quadratic interpolation with the node points based on the parameterization variables (3). The normal is obtained is based on the approximate surface defined using (6).

6. SUBROUTINE FNDNRM2 (S, T, IFACE, V, K, VN)
   Calculate the "inner normal" to the surface at the point in element #K with parameterization variables (S, T). The normal is obtained is based on the approximate surface defined using (6).

7. SUBROUTINE NRM_AV (NVERT, VNRM, NFACE, V_TO_F, VFLIST, VNRM_V)
   Beginning with the normals in VNRM for each element in IFACE, calculate an average of the normals at each node point. This means that for some node point, say #I, find all faces to which this node point belongs and then average the corresponding normals from VNRM. This is intended for only the case of piecewise quadratic interpolation with the node points based on the parameterization variables (3).

8. SUBROUTINE POLYPREP (NFACE, IFACE, VERTEX, AREA, V_NORMAL)
   Calculate areas and normals of elements in IFACE for polyhedral surfaces. This is to be used only if the triangular elements are all planar. It

is preparation for setting up the boundary element integrals for polyhedral surfaces, leading to faster numerical integration routines for planar triangular approximations of such surfaces.

9. SUBROUTINE LOCATE (P, IFACE, V, CENTER, RADIUS, NFACE, LOC, S, T, U)
   Given P in the surface S, find the planar triangular element within IFACE to which P belongs. This should be used only when the triangulation leads to planar triangular faces.

10. SUBROUTINE F_TO_V (IFACE, NFACE, NVERT, V_TO_F, VFLIST)
    For each node point, produce the faces containing this vertex and store these in a list. This considers only the node points associated with the parameterization variables in (3).

11. SUBROUTINE PRNTRI (IWRITE, NFACE, NVERT, IFACE, VERTEX, INDEXV, CENTER, RADSQR, H_MIN, H_MAX)
    This prints the triangles of the triangulation, including their indices, vertices, approximate area, and circumscribing circles.

12. SUBROUTINE PRNTR2 (IWRITE, NFACE, NVERT, IFACE, VERTEX, INDEXV, CENTER, RADSQR, F_TO_C, STFACE, H_MIN, H_MAX)
    This variant of PRNTRI is used to print triangulation information when a two-grid situation is being used with the triangulations. It includes additional information on the relation of the fine mesh to the coarse mesh.

2.1.3 Piecewise Quadratic Isoparametric Interpolation.

1. FUNCTION INTERP (S,T, IDX, IFACE, F)
   Evaluate the piecewise quadratic isoparametric function interpolating to F at the point with parameters (S,T).

2. SUBROUTINE QUADSF (S, T, IDX, IFACE, VERTEX, Q)
   Find the point Q on the piecewise quadratic isoparametric surface associated with triangle #IDX and parameters (S,T).

3. SUBROUTINE BASIS (S, T, P)
   Evaluate the quadratic basis functions on the unit simplex.

2.1.4 General Numerical Integration.

1. SUBROUTINE WGT (IDX, W, IFACE, V)
   This program calculates the weights for the numerical integration of F(P) over triangle #IDX within the triangulation. This integration has degree of precision 2.

2. SUBROUTINE WGT1 (IDX, W, IFACE, V)
   This program calculates the weights for the first order integration method over the quadratic isoparametric surface specified by triangle #IDX of the triangulation. This integration has degree of precision 1.

3. SUBROUTINE SETLEV (K, IFACE, VERTEX, NVERT, CENTER, RADIUS, LEVEL, MAXLEV, H)
   This program calculates integration levels over triangle #K for each of the nodes in VERTEX. If the node point lies in triangle #K, then we will want a special integration, to compensate for the singularity in the integrand. For nodes immediately next to triangle #K, we will set the level to MAXLEV. As the node moves away from triangle #K, the assigned level decreases.

4. SUBROUTINE SETLV2 (K, IFACE, VERTEX, NFACE, NVERT, CENTER, RADIUS, LEVEL, MAXLEV, H, F_TO_C, STFACE, POINT, STINFO)
   This variant of SETLEV is used for adaptive refinement of the triangulation.

5. SUBROUTINE INTEGR (VINT, F, N)
   This program calculates a double integral over the unit square [0,1]x[0,1] for the vector-valued function F(S,T) of length 6, using an N-point product Gaussian quadrature formula.

6. FUNCTION INTGR2 (F, N)
   This program calculates a double integral over the unit square [0,1]x[0,1] for the function F(S,T), using an N-point product Gaussian quadrature formula.

7. SUBROUTINE ZEROLG (N, ZZ, WW, A, B)
   This routine produces the nodes and weights for Gauss-Legendre quadrature on (A,B), of order N.

8. SUBROUTINE SMPLXR (W, SM, TM)
   This routine gives the weights and nodes that are used in the quadrature rule used in function SMPLX, SMPLX3, and SMPLX6. It is the 7-point rule T2:5-1 of Art Stroud, with degree of precision 5.

9. FUNCTION SMPLX (FN, LEVEL)
   This program calculates the integral of FN(S,T) over the unit simplex in the plane by using a composite version of the 7-point rule T2:5-1 of Art Stroud.

10. SUBROUTINE SMPLX3 (VRINT, FCNKL, LEVEL)
    This program calculates the integral of the vector-valued function FCNKL(S,T) of length 3 over the unit simplex in the plane by using a composite version of the 7-point rule T2:5-1 of Art Stroud. Intended for use with piecewise linear interpolation.

11. SUBROUTINE SMPLX6 (VRINT, FCNKL, LEVEL)
    This program calculates the integral of the vector-valued function FCNKL(S,T) of length 6 over the unit simplex in the plane by using a composite version of the 7-point rule T2:5-1 of Stroud [21]. Intended for use with peicewise quadratic interpolation.

12. SUBROUTINE SMPLXL (W, S, T)
    This routine gives the nodes and weights for the six point method of degree of precision 4, given as *Rule 41* of Table 4 from [18].

2.2 Contents of INTPACK

These are specialized routines for calculating various integrals needed for discretizing integral equations. Most of the routines are for calculating collocation integrals over triangular elements.

1. SUBROUTINE INTKL1 (IDX,P, VRINT, OPTION, LEVEL, IFACE, VERTEX)

This program evaluates the integrals over triangle #IDX of the integrands

KERNEL(P,Q(S,T))*C(I,S,T), I=1,...,6

The functions C(I,S,T) are quadratic basis functions. The numerical integration uses SMPLX6. If the kernel function KERNEL is singular when P=Q, then subroutine INTKL3 should be used in preference to INTKL1.

2. SUBROUTINE INTKL2 (IDX, P, VRINT, OPTION, LEVEL, IFACE, VERTEX, LAYER)
   This program evaluates the integrals over triangle #IDX of the integrands

   KERNEL(P,Q(S,T))*C(I,S,T), I=1,...,6

   where KERNEL(P,Q) is either a single layer or double layer kernel function. The functions C(I,S,T) are quadratic basis functions. The numerical integration uses SMPLX6. This program is intended for the case where the field point P is not in the triangle #IDX. In the case P does belong to triangle #IDX, one should use subroutine INTKL4.

3. SUBROUTINE INTKL3 (IDXF, IDXP, VRINT, OPTION, NINTEG, IFACE, VERTEX)
   This program evaluates the integrals over triangle #IDXF of the integrands

   KERNEL(P,Q(S,T))*C(I,S,T), I=1,...,6

   The numerical integration uses INTEGR. It is restricted to the case that P is a node point in triangle #IDXF.

4. SUBROUTINE INTKL4 (IDXF, IDXP, VRINT, OPTION, NINTEG, LAYER, IFACE, VERTEX)
   This program evaluates the integrals over triangle #IDXF of the integrands

   KERNEL(P,Q(S,T))*C(I,S,T), I=1,...,6

   where KERNEL(P,Q) is either a single layer or double layer kernel function. The numerical integration uses INTEGR. It is restricted to the case that P is a node point in triangle #IDXF.

5. SUBROUTINE INTKB5 (IDX, P, VRINT, OPTION, LEVEL, IFACE, VERTEX, LAYER)
   This program evaluates the integral over triangle #IDX of the integrand

   KERNEL(P,Q(S,T))*B(Q(S,T))

   where KERNEL(P,Q) is either a single layer or double layer kernel function, the function B(Q) is a given density function. The numerical integration uses SMPLX. This routine is intended for the case where P is not in the triangle specified by #IDX. In the case P does belong to triangle #IDX, one should use subroutine INTKB6.

6. SUBROUTINE INTKB6 (IDXF, IDXP, VRINT, OPTION, NINTEG, LAYER, IFACE, VERTEX)
   This program evaluates the integral over triangle #IDXF of the integrand

   KERNEL(P,Q(S,T))*B(Q(S,T))

   where KERNEL(P,Q) is either a single layer or double layer kernel function, the function B(Q) is a given density function. The numerical integration uses INTGR2. It is restricted to the case that P is a node point in triangle #IDXF.

7. SUBROUTINE INTKL7 (IDX, P, VRINT, OPTION, LEVEL, IFACE, VERTEX, LAYER)
   This program evaluates the integrals over triangle #IDX of the integrands

   KERNEL(P,Q(S,T))*D(I,S,T),   I=1,2,3

   where KERNEL(P,Q) is either a single layer or double layer kernel function. The functions D(I,S,T) are linear basis functions. The numerical integration uses SMPLX3. This routine is intended for the case where the field point P is not in the triangle #IDX. In the case P does belong to triangle #IDX, one should use subroutine INTKL8.

8. SUBROUTINE INTKL8 (IDXF, IDXP, VRINT, OPTION, NINTEG, LAYER, IFACE,

VERTEX)
This program evaluates the integrals over triangle #IDXF of the integrands

KERNEL(P,Q(S,T))*D(I,S,T), I=1,2,3

where KERNEL(P,Q) is either a single layer or double layer kernel function. The functions D(I,S,T) are linear basis functions. The numerical integration uses INTEGR. It is restricted to the case that P is a node point in triangle #IDXF.

9. SUBROUTINE INTKB9 (IDXF, IDXP, VRINT, OPTION, NINTEG, LAYER, IFACE, VERTEX, IDXPST, STINFO)
This program evaluates the integral over triangle #IDXF of the integrand

KERNEL(P,Q(S,T))*B(Q(S,T))

where KERNEL(P,Q) is either a single layer or double layer kernel function, the function B(Q) is a given density function. The numerical integration uses INTGR2. It is restricted to the case that P is a node point in triangle #IDXF, But P is not one of the regular six nodes which define the triangle. This routine is to be used with adaptive mesh refinements, where the mesh is nonconforming.

10. SUBROUTINE INTKL10 (IDXF, IDXP, VRINT, OPTION, NINTEG, LAYER, IFACE, VERTEX, IDXPST, STINFO)
This program evaluates the integrals over triangle #IDXF of the integrands

KERNEL(P,Q(S,T))*C(I,S,T), I=1,...,6

where KERNEL(P,Q) is either a single layer or double layer kernel function. The numerical integration uses INTEGR. It is restricted to the case that P is a node point in triangle #IDXF, But P is not one of the regular six nodes which define the triangle. This routine is to be used with adaptive mesh refinements, where the mesh is nonconforming.

11. SUBROUTINE INTKL14 (K_INT, VRINT,OPTION, LEVEL, IFACE, VERTEX, P, P_NORM)
Form the integral with respect to Q over triangular element #K_INT of the radiosity kernel function (unoccluded). This is the collocation integral for using piecewise constant interpolation over S, with collocation

at the "centroid" of the element. This is for the case the collocation point P is not in the element #K_INT. The numerical integration uses SMPLX.

12. SUBROUTINE INTKL15 (K_INT, VRINT, OPTION, NINTEG, IFACE, VERTEX, P, P_NORM)
    Form the integral with respect to Q over triangular element #K_INT of the radiosity kernel function (unoccluded). This is the collocation integral for using piecewise constant interpolation over S, with collocation at the "midpoint" of the element, with P the "centroid" of element #K_INT.

13. SUBROUTINE INTKL16 (K_INT, VRINT, OPTION, LEVEL, IFACE, VERTEX, P, LAYER)
    Form the integral with respect to Q over triangular element #K_INT of the potential kernel function. This is the collocation integral for using piecewise constant interpolation over S, with collocation at the "centroid" of the element. This is for the case the collocation point P is not in the element #K_INT. The numerical integration uses SMPLX.

14. SUBROUTINE INTKL17 (K_INT, VRINT, OPTION, NINTEG, IFACE, VERTEX, P, LAYER)
    Form the integral with respect to Q over triangular element #K_INT of the potential kernel function. This is the collocation integral for using piecewise constant interpolation over S, with collocation at the "centroid" of the element, with P the "midpoint" of element #K_INT. The numerical integration uses INTGR2.

15. SUBROUTINE INTKB18 (IDXF, P, VRINT, OPTION, NINTEG, LAYER, IFACE, VERTEX)
    Form the integral over triangle #IDXF of the integrand

    $$\text{KERNEL(P,Q(S,T))*BDYFCN(Q(S,T),ISUR)}$$

    where KERNEL(P,Q) is either a single layer or double layer kernel function, depending on the value of the variable LAYER; the function BDYFCN is a given density function. This program is restricted to the case that P is the centroid in triangle #IDXF. The numerical integration uses INTGR2.

16. SUBROUTINE INTKL19 (K_INT, VRINT, OPTION, LEVEL, IFACE, VERTEX, P, P_NORM, A)
Form the integral with respect to Q over triangle #K_INT of

KERNEL(P,Q)*LINEAR_BASIS(I,Q), I=1,2,3

where KERNEL(P,Q) is the radiosity kernel function (unoccluded). The functions LINEAR_BASIS(I,Q) are the basis functions for the linear interpolation defined in (1) with the interpolation parameter A. This routine assumes the field point P is not in triangle #K_INT.

17. SUBROUTINE INTKL19G (K_INT, VRINT, OPTION, LEVEL, IFACE, VERTEX, P, P_NORM, A)
This is the same as INTKL19, except that the numerical integration uses INTEGR3G, based on a change of variables (cf. (35)) and Gaussian quadrature.

18. SUBROUTINE INTKL20 (K_INT, VRINT, OPTION, NINTEG, IFACE, VERTEX, P, P_NORM, P_ST, A)
Form the integral with respect to Q over triangle #K_INT of

KERNEL(P,Q)*LINEAR_BASIS(I,Q), I=1,2,3

where KERNEL(P,Q) is the radiosity kernel function (unoccluded). The functions LINEAR_BASIS(I,Q) are the basis functions for the linear interpolation defined in (1) with the interpolation parameter A. This routine assumes the field point P is in triangle #K_INT. The numerical integration uses INTEGR3.

19. SUBROUTINE INTKB21 (K_INT, OPTION, LEVEL, IFACE, VERTEX, NODES, P, P_NORM, A, FCN, INTEGRAL)
Form the integral with respect to Q over triangle #K_INT of

KERNEL(P,Q)*FCN_N(Q)

where KERNEL(P,Q) is the radiosity kernel function (unoccluded). The function FCN_N(Q) is the interpolation of FCN(Q) at the interpolation points in triangle #K_INT. This routine assumes P does not belong to the element #K_INT. The numerical integration uses SMPLX3.

20. SUBROUTINE INTKB22 (K_INT, OPTION, NINTEG, IFACE, VERTEX, NODES, P, P_NORM, P_ST, A, FCN, INTEGRAL)
Form the integral with respect to Q over triangle #K_INT of

$$\text{KERNEL(P,Q)*FCN\_N(Q)}$$

where KERNEL(P,Q) is the radiosity kernel function (unoccluded). The function FCN_N(Q) is the interpolation of FCN(Q) at the interpolation points in triangle #K_INT. This routine assumes P does belong to the element #K_INT. The numerical integration uses INTEGR3.

21. SUBROUTINE INTKL23 (IDX, P, VRINT, LEVEL, IFACE, VERTEX, LAYER, AREA, V_NORMAL)
This program assumes all elements in IFACE are <u>planar</u> with an associated preliminary preparation of needed facts about the surface. The routine evaluates the integrals over triangle #IDX of the integrands

$$\text{KERNEL(P,Q(S,T))*C(I,S,T)}, \qquad \text{I=1,...,6}$$

where KERNEL(P,Q) is either a single layer or double layer kernel function. The functions C(I,S,T) are quadratic basis functions. The numerical integration uses SMPLX6. This program is intended for the case where the field point P is not in the triangle #IDX. The numerical integration uses SMPLX6.

22. SUBROUTINE INTKL24 (IDXF, IDXP, VRINT, NINTEG, LAYER, IFACE, VERTEX, AREA)
This program assumes all elements in IFACE are <u>planar</u> with an associated preliminary preparation of needed facts about the surface. The routine evaluates the integrals over triangle #IDX of the integrands

$$\text{KERNEL(P,Q(S,T))*C(I,S,T)}, \qquad \text{I=1,...,6}$$

where KERNEL(P,Q) is either a single layer or double layer kernel function. The functions C(I,S,T) are quadratic basis functions. The numerical integration uses SMPLX6. This program is intended for the case where the field point P is in the triangle #IDX. The numerical integration uses INTEGR.

23. SUBROUTINE INTKB25 (IDX, P, VRINT, LEVEL, IFACE, VERTEX, LAYER, AREA, V_NORMAL)
This program assumes all elements in IFACE are planar with an associated preliminary preparation of needed facts about the surface. Form the integral over planar triangle #IDX of the integrand

KERNEL(P,Q(S,T))*BDYFCN(Q(S,T),I_SUR)

where KERNEL(P,Q) is either a single layer or double layer kernel function, depending on the value of LAYER; the function BDYFCN is a given density function. This routine assumes P does not belong to triangle #IDX. The numerical integration uses SMPLX6.

24. SUBROUTINE INTKB26 (IDXF, IDXP, VRINT, NINTEG, LAYER, IFACE, VERTEX, AREA)
This program assumes all elements in IFACE are planar with an associated preliminary preparation of needed facts about the surface. Form the integral over planar triangle #IDX of the integrand

KERNEL(P,Q(S,T))*BDYFCN(Q(S,T),I_SUR)

where KERNEL(P,Q) is either a single layer or double layer kernel function, depending on the value of LAYER; the function BDYFCN is a given density function. This routine assumes P does belong to triangle #IDX. The numerical integration uses INTEGR.

25. SUBROUTINE INTKL27 (K_INT, VRINT, OPTION, LEVEL, IFACE, VERTEX, P, P_NORM, A, LAYER)
Form the integral with respect to Q over triangle #K_INT of

KERNEL(P,Q)*LINEAR_BASIS(I,Q),   I=1,2,3

where KERNEL(P,Q) is the single layer kernel or the normal derivative of the single layer. The functions LINEAR_BASIS(I,Q) are the basis functions for the linear interpolation defined in (1) with the interpolation parameter A. This routine assumes the field point P is not in triangle #K_INT. The numerical integration uses SMPLX3.

26. SUBROUTINE INTKL27G (K_INT, VRINT, OPTION, LEVEL, IFACE, VERTEX, P, P_NORM, A)

This is the same as INTKL27, except that the numerical integration uses INTEGR3G, based on a change of variables (cf. (35)) and Gaussian quadrature.

27. SUBROUTINE INTKL28 (K_INT, VRINT, OPTION, NINTEG, IFACE, VERTEX, P, P_NORM, P_ST, A)
    Form the integral with respect to Q over triangle #K_INT of

    KERNEL(P,Q)*LINEAR_BASIS(I,Q),   I=1,2,3

    where KERNEL(P,Q) is the single layer kernel or the normal derivative of the single layer. The functions LINEAR_BASIS(I,Q) are the basis functions for the linear interpolation defined in (1) with the interpolation parameter A. This routine assumes the field point P is in triangle #K_INT. The numerical integration uses INTEGR3.

2.2.1 Evaluation of Integrands.

To carry out the integrations implicit in the above subroutines, there are accompanying functions or subroutines defining the needed integrands. For the subroutine named INTabcd, there is an accompanying program named FCNabcd, with 'abcd' of the form 'KLef' or 'KBef'. In addition, there is a routine

FUNCTION DBLAYR (P, Q, S, T, V)

This routine calculates the double layer kernel for the triangular surface specified by the vertices in V.

2.3 Contents of AUXPACK

This package contains standard routines from general numerical computing, including setting machine constants, setting Gaussian quadrature nodes and weights, solving dense linear systems, and other tasks.

1. FUNCTION D1MACH (I)
   This is the standard routine from Bell Labs for computing double precision machine constants.

2. FUNCTION DELTA (I)
   This uses D1MACH to calculate specific machine constants for use with BIEPACK. The value DELTA(2) is not used in the present release of BIEPACK.

3. SUBROUTINE ZEROLG (N, ZZ, WW, A, B)
   Produce the nodes and weights for Gauss-Legendre quadrature on the interval [A,B] of order N. This yields values for N = 1, 2, ..., 20.

4. SUBROUTINE GAUSS2 (A, B, N, TT, WW)
   Produce the nodes and weights for Gauss-Legendre quadrature on the interval [A,B] of order N. This yields values for N = 1, 2, 4, 8, 16, 32, 64, 128, 256.

5. SUBROUTINE LINSYS (A, LUFACT, N, B, SOLN, OPTION, RCOND, IPIVOT, MACHIN, Z, ERRMAX)
   This is a driver program for solving the linear system A*X=B using the LINPACK routines DGECO and DGESL. Various options are available. Other needed LINPACK routines are also given in AUXPACK.

3 THE ROUTINES INIT & MIDPT

Our framework has been so designed as to maximize the types of surfaces to which our programs can be applied. Because surfaces S can be specified in many different ways, we have chosen a way to work with S which does not require knowing an explicit parametrization of S. Instead the user must supply two subroutines named INIT and MIDPT. The routine INIT will define the initial 'coarse' mesh for the surface; and the routine MIDPT will allow us to find an approximate midpoint on S between two given nodes v_i and v_j. The coarse mesh $\mathcal{T}_N$ and associated nodes $\mathcal{V}_N$ are to be so chosen that the following assumptions are satisfied.

1. All vertices on S are in the set of vertices $\mathcal{V}_N$.

2. The edges of the surface S are to be contained in the union of the sides of the triangular elements in $\mathcal{T}_N$.

3. If two elements Δ_k and Δ_l intersect, then they do so at either a common vertex or along an entire common edge. [For adaptive mesh refinement programs, which we are now constructing, this assumption is dispensed with.]

To help explain the structure of these two routines, we include here two examples of them, for the cases of an ellipsoid and a simplex in $\mathbf{R}^3$. With the subroutine INIT for S an ellipsoid, we have omitted (for reasons of space) the subroutine POLYHD which calculates the faces and vertices for those regular polyhedra which have triangular faces.

- **ELLIPSOID**

```
      SUBROUTINE INIT(VERTEX,IFACE,NFACE,NVERT,NMIDPT,INDXV)
C     ---------------
C
C     SET UP THE INITIAL TRIANGULATION OF THE SURFACE S.
C     'VERTEX' WILL CONTAIN THE NODES.
C     'IFACE'  WILL CONTAIN THE FACES, WITH INDICES OF THE APPROPRIATE
C              NODES AND OTHER INFORMATION GIVEN FOR EACH FACE.
C     'NFACE'  IS THE DIMENSION OF 'IFACE'.
C     'NVERT'  IS THE DIMENSION OF 'VERTEX'.
C     'NMIDPT' IS THE NUMBER OF MIDPOINT NODES FOR THE TRIANGULATION.
C     'INDXV'  WILL CONTAIN AN INDEX FOR EACH NODE, INDICATING WHETHER
C              IT IS AT A VERTEX, EDGE, OR FACE OF THE ORIGINAL SURFACE S.
C              INDXV(I)=0 MEANS VERTEX(I) IS INTERIOR TO A FACE OF S.
C              INDXV(I)=1 MEANS VERTEX(I) IS INTERIOR TO AN EDGE OF S.
C              INDXV(I)=2 MEANS VERTEX(I) IS A VERTEX OF S.
C
C     IN THIS VERSION OF INIT, WE FIRST GIVE THE FACES IN TERMS OF
C     JUST THEIR MAIN VERTICES.  THEN SUBROUTINE 'MIDPT' IS USED TO
C     OBTAIN THE MIDPOINTS OF THE SIDES.
C
C     ****************************************************************
C     IN THIS CASE S IS THE ELLIPSOID
C        (X/AP)**2 + (Y/BP)**2 + (Z/CP)**2 = 1
```

```
C     THE USER MUST SPECIFY THE PARAMETERS (AP,BP,CP) THRU THE COMMON
C     STATEMENT GIVEN BELOW.
C     *****************************************************************
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/SURPAR/AP, BP, CP, NP
      DIMENSION IFACE(7,*), VERTEX(3,*), INDXV(*),
     *  IFT(3,20), V(3), ID(4), CENTER(3)
      PARAMETER(ZERO=0.0D0)
      DATA CENTER/3*ZERO/
      DATA ID/1,2,3,1/
C
C     ***************************************************************
C
C     THIS PORTION OF THE ROUTINE DEPENDS ON THE SURFACE BEING
C     AN ELLIPSOID.
C
      NFACE = NP
      CALL POLYHD(IFT,NFACE,VERTEX,NVERT)
      DO 10 I=1,NFACE
        IFACE(7,I) = 0
        DO 10 J=1,3
10      IFACE(J,I) = IFT(J,I)
      DO 20 I=1,NVERT
20      INDXV(I) = 0
      DO 30 I=1,NVERT
        VERTEX(1,I) = AP*VERTEX(1,I)
        VERTEX(2,I) = BP*VERTEX(2,I)
30      VERTEX(3,I) = CP*VERTEX(3,I)
      NVOLD = NVERT
C
C     ***************************************************************
C
C     CALCULATE THE MIDPOINTS OF THE SIDES GIVEN IN IFACE.
      DO 100 I=1,NFACE
      DO 100 J=1,3
        CALL MIDPT(I,V,IVX,ID(J),ID(J+1),1,IFACE,VERTEX,INDXV)
```

```
        CALL VRTXLC(V,IDX,NVERT,IVX,VERTEX,INDXV)
100     IFACE(J+3,I) = IDX
C
C     RE-ORIENT THE FACES OF 'IFACE' SO THAT THEY WILL DETERMINE
C     A NORMAL WHICH POINTS TOWARDS THE ORIGIN.
      CALL ORIENT(IFACE,NFACE,VERTEX,NVERT,CENTER)
      NMIDPT = NVERT - NVOLD
      RETURN
      END

      SUBROUTINE MIDPT(IDX,V,IVX,I1,I2,LED,IFACE,VERTEX,INDXV)
C     ----------------
C
C     FIND THE MIDPOINT 'V' BETWEEN TWO GIVEN NODES, OF INDICES
C     #I1 AND #I2 WITHIN FACE #IDX OF 'IFACE'.  ALSO SET 'IVX' TO
C     INDICATE WHETHER 'V' IS AT A VERTEX, EDGE, OR FACE OF THE
C     ORIGINAL SURFACE S.  LED=0 MEANS THAT 'V' IS NOT ON AN EDGE
C     OF S NOR IS IT A VERTEX OF S; AND THUS IVX=0 IN THIS CASE.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION V(3), IFACE(7,*), VERTEX(3,*), INDXV(*)
      COMMON/SURPAR/AP, BP, CP, NP
C
      I = IFACE(I1,IDX)
      J = IFACE(I2,IDX)
C
C     ****************************************************************
C
C     THESE STATEMENTS ASSUME THE TRIANGULAR SURFACE IS ON AN ELLIPSOID.
C
      V(1) = (VERTEX(1,I) + VERTEX(1,J))/AP
      V(2) = (VERTEX(2,I) + VERTEX(2,J))/BP
      V(3) = (VERTEX(3,I) + VERTEX(3,J))/CP
      SUM = SQRT(V(1)*V(1) + V(2)*V(2) + V(3)*V(3))
      DO 20 K=1,3
20      V(K) = V(K)/SUM
      V(1) = AP*V(1)
```

```
      V(2) = BP*V(2)
      V(3) = CP*V(3)
C
C     ***************************************************************
C
      K1 = INDXV(I)
      K2 = INDXV(J)
      IF((K1*K2 .EQ. 0) .OR. (LED .EQ. 0)) THEN
          IVX = 0
      ELSE
          IVX = 1
      END IF
      RETURN
      END
```

- **SIMPLEX**

```
      SUBROUTINE INIT(VERTEX,IFACE,NFACE,NVERT,NMIDPT,INDXV)
C     ---------------
C
C     SET UP THE INITIAL TRIANGULATION OF THE SURFACE S.
C     'VERTEX' WILL CONTAIN THE NODES.
C     'IFACE'  WILL CONTAIN THE FACES, WITH INDICES OF THE APPROPRIATE
C              NODES AND OTHER INFORMATION GIVEN FOR EACH FACE.
C     'NFACE'  IS THE DIMENSION OF 'IFACE'.
C     'NVERT'  IS THE DIMENSION OF 'VERTEX'.
C     'NMIDPT' IS THE NUMBER OF MIDPOINT NODES FOR THE TRIANGULATION.
C     'INDXV'  WILL CONTAIN AN INDEX FOR EACH NODE, INDICATING WHETHER
C              IT IS AT A VERTEX, EDGE, OR FACE OF THE ORIGINAL SURFACE S.
C              INDXV(I)=0 MEANS VERTEX(I) IS INTERIOR TO A FACE OF S.
C              INDXV(I)=1 MEANS VERTEX(I) IS INTERIOR TO AN EDGE OF S.
C              INDXV(I)=2 MEANS VERTEX(I) IS A VERTEX OF S.
C
```

```
C     IN THIS VERSION OF INIT, WE FIRST GIVE THE FACES IN TERMS OF
C     JUST THEIR MAIN VERTICES.  THEN SUBROUTINE 'MIDPT' IS USED TO
C     OBTAIN THE MIDPOINTS OF THE SIDES.
C
C     ***************************************************************
C     IN THIS CASE S IS A TETRAHEDRON IN SPACE, WITH THREE OF THE
C     EDGES LYING ALONG THE COORDINATE AXES, AND WITH THE ORIGIN
C     AS ONE OF THE VERTICES.  THE LENGTHS OF THE EDGES ALONG THE
C     X, Y, AND Z AXES ARE A,B, AND C, RESPECTIVELY.  THE USER MUST
C     SPECIFY THE PARAMETERS (A,B,C) THRU THE COMMON STATEMENT GIVEN
C     BELOW.
C     ***************************************************************
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER(ZERO=0.0D0, FOUR=4.0D0)
      DIMENSION IFACE(7,*), VERTEX(3,*), INDXV(*),
     *  IFC(3,4), CENTER(3), ID(4), V(3)
      COMMON/SURPAR/A, B, C, NP
      DATA IFC/2,3,4,1,3,4,1,2,4,1,2,3/
      DATA ID/1,2,3,1/
C
C     ***************************************************************
C
C     THIS PORTION OF THE ROUTINE DEPENDS ON THE TETRAHEDRON.
C
      NFACE = 4
      NVERT = 4
      DO 5 I=1,3
      DO 5 J=1,4
5       VERTEX(I,J) = ZERO
      VERTEX(1,2) = A
      VERTEX(2,3) = B
      VERTEX(3,4) = C
      DO 15 I=1,4
15      INDXV(I) = 2
C
C     SET FACES.
```

```
      DO 20 J=1,NFACE
        DO 30 I=1,3
30        IFACE(I,J) = IFC(I,J)
20      IFACE(7,J) = J
C
C     **********************************************************
C
C     CALCULATE THE MIDPOINTS OF THE SIDES GIVEN IN IFACE.
      NV=NVERT
      DO 100 I=1,NFACE
      DO 100 J=1,3
        CALL MIDPT(I,V,IVX,ID(J),ID(J+1),1,IFACE,VERTEX,INDXV)
        CALL VRTXLC(V,IDX,NVERT,IVX,VERTEX,INDXV)
100   IFACE(J+3,I) = IDX
C
C     ORIENT THE TRIANGLES TOWARD THE CENTER OF THE SOLID.
      CENTER(1) = A/FOUR
      CENTER(2) = B/FOUR
      CENTER(3) = C/FOUR
      CALL ORIENT(IFACE,NFACE,VERTEX,NVERT,CENTER)
      NMIDPT = NVERT - NV
      RETURN
      END

      SUBROUTINE MIDPT(IDX,V,IVX,I1,I2,LED,IFACE,VERTEX,INDXV)
C     ----------------
C
C     FIND THE MIDPOINT 'V' BETWEEN TWO GIVEN NODES, OF INDICES
C     #I1 AND #I2 WITHIN FACE #IDX OF 'IFACE'.  ALSO SET 'IVX' TO
C     INDICATE WHETHER 'V' IS AT A VERTEX, EDGE, OR FACE OF THE
C     ORIGINAL SURFACE S.  LED=0 MEANS THAT 'V' IS NOT ON AN EDGE
C     OF S NOR IS IT A VERTEX OF S; AND THUS IVX=0 IN THIS CASE.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION V(3), IFACE(7,*), VERTEX(3,*), INDXV(*)
      DATA TWO/2.0D0/
C
```

Table 1: IFACE for S a simplex: i_j=IFACE(j,k)

k	i_1	i_2	i_3	i_4	i_5	i_6	i_7
1	3	2	4	5	7	6	1
2	1	3	4	8	6	9	2
3	2	1	4	10	9	7	3
4	1	2	3	10	5	8	4

```
      I = IFACE(I1,IDX)
      J = IFACE(I2,IDX)
C     *************************************************************
C
C     THESE STATEMENTS ASSUME THE TRIANGULAR ELEMENT IS PLANAR
C     WITH EDGES WHICH ARE STRAIGHT LINES.
C
      DO 10 K=1,3
10      V(K) = (VERTEX(K,I) + VERTEX(K,J))/TWO
C
C     **************************************************************
      K1 = INDXV(I)
      K2 = INDXV(J)
      IF((K1*K2 .EQ. 0) .OR. (LED .EQ. 0)) THEN
          IVX = 0
      ELSE
          IVX = 1
      END IF
      RETURN
      END
```

The case of the simplex is illustrated in Figure 4. To illustrate what is produced by INIT for this surface, we give IFACE, VERTEX, and INDEXV in Tables 1 and 2.

A careful perusal of these programs will show the common structure and the types of quantities which need to be computed or specified within the

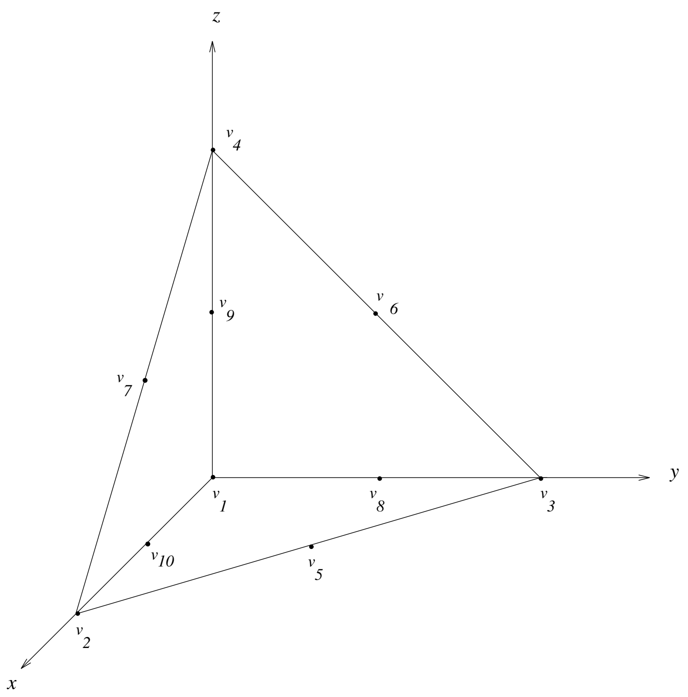

Figure 4: Initial triangulation of a simplex

programs. In INIT, we usually specify first the vertices of the triangulation and the portions of the faces which depend upon them. Following that, we use MIDPT to create the midpoints of the sides of the triangular elements. The routine ORIENT is used to so orient each element, as stored in IFACE, that the order of storage within IFACE will give the direction of an inner normal to the surface.

Suppose Δ_k is specified by the nodes $\{v_{k,i} \mid 1 \leq i \leq 6\}$ in VERTEX of indices IFACE(i,k), $i = 1, \cdots, 6$. Then the order of $v_{k,1}$, $v_{k,2}$, $v_{k,3}$, together with the 'right-hand rule' for vectors, gives the direction of the inner normal to Δ_k when considered as a portion of S. For most surfaces one encounters in practice, the routine ORIENT is a simple way to create the correct orientation for all triangular faces stored in IFACE. The user gives a point P

Table 2: VERTEX for S a simplex: $v_i = (x_i, y_i, z_i)$

i	x_i	y_i	z_i	$Index$
1	0.0	0.0	0.0	2
2	1.0	0.0	0.0	2
3	0.0	1.0	0.0	2
4	0.0	0.0	1.0	2
5	0.5	0.5	0.0	1
6	0.0	0.5	0.5	1
7	0.5	0.0	0.5	1
8	0.0	0.5	0.0	1
9	0.0	0.0	0.5	1
10	0.5	0.0	0.0	1

inside the region bounded by S, so chosen that the following is true for all points $Q \in S$: The angle between the normal vector ν and the vector from Q to P is to be less than $90°$. If the surface S is the boundary of a region which is not homeomorphic to a sphere, then the routine ORIENT cannot be used to create the correct orientation of the triangulation; and there are also regions which are homeomorphic to the sphere for which ORIENT is also not suitable. All of the surfaces we provide (ellipsoid, simplex, and the remaining ones given below) can be oriented with ORIENT; but a toroidal region could not be oriented with this routine.

The routine MIDPT can vary greatly with the form of the surface. The version for the simplex is the same as that used for all polyhedral surfaces. In general, the routine MIDPT contains within it an implicit definition of the surface S. We have used this operational manner of specifying the surface as a way of being very flexible to the user. Different types of surfaces require different types of definitions.

The routine MIDPT accepts as input the indices of two existing node points on S and a triangular element to which they belong. MIDPT is to return a third point V which is to be considered a midpoint on S between the two given nodes. The routine also returns IVX, which indicates whether V is on an edge of the original surface or is on an interior of a smooth initial sub-surface of S.

The library we provide contains versions of INIT and MIDPT for the following surfaces, in addition to the above ellipsoid and simplex:

1. Elliptical paraboloid with top.

2. Rectangular parallelopiped.

3. Elliptical cone.

4. L-block. This is an L-shaped block.

5. Beanbag. This surface can be thought of as an ellipsoid which has been squeezed inward around its intersection in the xy-plane. It is a smooth boundary, but the region it encloses is neither convex nor symmetric.

6. Sector. This is a sector of a circle in the xy-plane, with the central vertex at the origin. This is useful for studying numerical integration of functions with a singularity at the origin.

7. 2-piece. The "two-piece surface" is defined as follows:

$$\begin{aligned} S_1 &= \{(x, y, 0) \mid 0 \le x, y \le 1\} \\ S_2^{(p)} &= \{(x, y, z) \mid 0 \le x, y \le 1,\ z = 2 - x^p\} \end{aligned} \tag{24}$$

   and let $S^{(p)} = S_1 \cup S_2^{(p)}$. We use $S^{(2)}$ and $S^{(3)}$, and there are separate routines for both, named INIT_TWOPIECE_2 and INIT_TWOPIECE_3, respectively. Note that the interpolatory quadratic interpolation of $S_2^{(2)}$ is exact, so problems solved on this surface with BIEPACK do not have any errors due to approximation of the surface. The interpolatory quadratic interpolation of $S_2^{(3)}$ is, of course, not exact. These surfaces were intended for use with the radiosity equation, but can be used as an example for any integral equation over surfaces in space. They can be used for comparisons of the effect on the order of convergence of using the approximation $S \approx \widetilde{S}_N$.

8. Wedge surface. This is a three dimensional wedge boundary. It is intended for use in experimenting with the behaviour of solutions of boundary integral equations and the radiosity equation on surfaces with edges. The routine is named INIT_3D-WEDGE.

9. Octant. This is the portion in the first octant of an ellipsoidal surface centered at the origin.

4 NUMERICAL INTEGRATION

Many numerical integration routines are included in TRIPACK, to be used for the wide variety of integrals which must be calculated. We list again the four main forms of integration used over the standard parametrization region σ.

$$\int_\sigma g(s,t)\,d\sigma \approx \frac{1}{6}\left[g(q_1)+g(q_2)+g(q_3)\right] \tag{25}$$

$$\int_\sigma g(s,t)\,d\sigma \approx \frac{1}{6}\left[g(q_4)+g(q_5)+g(q_6)\right] \tag{26}$$

$$\begin{aligned}\int_\sigma g(s,t)\,d\sigma \approx \tfrac{9}{40}g\left(\tfrac{1}{3},\tfrac{1}{3}\right) &+ \tfrac{1}{40}\left[g(q_1)+g(q_2)+g(q_3)\right] \\ &+\tfrac{1}{15}\left[g(q_4)+g(q_5)+g(q_6)\right]\end{aligned} \tag{27}$$

$$\int_\sigma g(s,t)\,d\sigma \approx \sum_{j=1}^{7} w_j g(\rho_j) \tag{28}$$

with the weights and nodes for (28) taken from the formula T2:5-1 of Stroud [21, p. 314]. For the latter, the weights $\{w_j\}$ and nodes $\{\rho_j\}$ are given by the routine SMPLXR; and the integration over σ is carried out in the routine SMPLX (and SMPLX3 and SMPLX6). The degrees of precision over σ of these four formulas are respectively 1, 2, 3, and 5.

We apply these formulas to the calculation of integrals over Δ_k by means of (7), which includes the approximation of the surface S by $\widetilde{S}_N$. Let the result of using these formulas to integrate $I(f) \equiv \int_S f(Q)\,dS_Q$ be denoted by $I_N^{(r)}(f)$ for $r = 1, 2, 3, 4$ for the above three quadrature formulas, respectively. If the sub-surfaces of S are sufficiently smooth, then with sufficient smoothness on f, we can show

$$I(f) - I_N^{(1)}(f) = O(h^2) \tag{29}$$

$$I(f) - I_N^{(2)}(f) = O(h^4) \tag{30}$$

$$I(f) - I_N^{(3)}(f) = O(h^4) \tag{31}$$

$$I(f) - I_N^{(4)}(f) = O(h^4) \tag{32}$$

The result (30) is better than expected might have been expected when considering the degree of precision of (26), and a detailed discussion is given in Chien [13] and Atkinson [8, Chap. 5]. Although $I_N^{(2)}$, $I_N^{(3)}$, and $I_N^{(4)}$ have equal orders of convergence, we have generally found $I_N^{(4)}$ is the more accurate formula in our applications to boundary integral equations. In the case of polyhedral boundaries S and those boundaries for which $\widetilde{S}_N = S$, the result in (32) can be improved to $O(h^6)$; and this gives an additional reason for using it in preference to $I_N^{(2)}$. We seldom use $I_N^{(3)}$, but it is less sensitive to the manner of refinement of the triangulation; and for surfaces S for which $\widetilde{S}_N = S$, the result (31) is valid regardless of the method of refinement being used.

For routines which are used just for numerical integration over the surface S, see the programs INTEGRATE and INTEGRATE_PLANAR. The first is based on the method (26), and the second is based on (25). Both use the piecewise quadratic approximation of the surface as defined in (6).

When applying (25) or (26) to the right side of (7), the weights of the integration are given by the routines WGT1 and WGT, respectively. When applying (28) to the right side of (7), we use the routines SMPLX, SMPLX3, and SMPLX6, depending on the intended application.

When the integrand in (7) is considered to be badly behaved, we use composite forms of (28). This occurs principally with integrands formed from $1/|P - Q|$, with P near to the integration triangle Δ_k but not contained in it. In such cases, we subdivide the parametrization region σ, and then (11) is used over each subdivision. To decide on the number of subdivisions needed, the routine SETLEV is used.

A user chosen parameter MAXLEV is given to SETLEV. For a given element Δ_k, each node point $P = v_i$ is classified by SETLEV as to its position relative to Δ_k, with the classification to be stored in LEVEL(i), $i = 1, \cdots, N_v$. Let r_k and C_k be the radius and center for the circle which circumscribes the planar triangle determined by IFACE(j, k), $k = 1, 2, 3$.

If $P \in \Delta_k$, then LEVEL(i) = -1.

If $P \notin \Delta_k$, define

$$R = |P - C_k|$$

$$
\begin{aligned}
\ell &= \max\{0, \text{floor}[(R - r_k)/h]\} \\
\text{LEVEL}(i) &= \max\{0, \text{MAXLEV} - \ell\}
\end{aligned}
\tag{33}
$$

The function 'floor(x)' refers to the integer part of x. For LEVEL$(i) \geq 0$, the value of LEVEL(i) will be the number of levels of subdivision to be used in applying (11) to the evaluation of our integrals for which the integrand contains a singularity of the type of $1/|P - Q|$.

As the distance between Δ_k and $P = v_i$ increases, the number of levels of subdivision of σ decreases, eventually reaching an integration over σ with no subdivision needed. This is the most efficient way we have found for handling the nearly singular integrals which arise when implementing boundary element methods. Also, it seems sufficient to begin with MAXLEV $= 0$ for the coarse mesh; and then to increase MAXLEV by 1 for each time the triangulation $\mathcal{T}_N$ is refined. In fact, this is probably a faster rate of increase for MAXLEV (considered as a function of N) than is necessary. This applies only to an accurate calculation of the unknown density function which solves the integral equation. When the density function is to be used to calculate a potential function by integration of a single or double layer potential integral formula at points P not on the boundary (as with an interior Dirichlet problem), then MAXLEV $= 0$ or 1 is probably sufficient when calculating the density function. Higher values of MAXLEV will be needed in the numerical integration of the potential function, with the size of MAXLEV increasing as the distance of P from the boundary S decreases. This schema does reduce considerably the computational cost of setting up the linear system for the collocation method and in computing potential integrals, without affecting the error in the solution to a significant extant.

Some of the integrals have a singular integrand. For example, if in (7), we have

$$f(Q) = \rho(Q)/|P - Q|$$

with $P \in \Delta_k$, then the integrand is singular. We handle such integrals with a change of variables. For simplicity, suppose we are calculating

$$\int_\sigma g(s,t)\, d\sigma \tag{34}$$

with $g(s,t)$ singular at only $(s,t) = (0,0)$. Introduce the change of variables

$$s = (1-y)x, \qquad t = yx, \qquad 0 \leq x, y \leq 1$$

With this, we have

$$\int_\sigma g(s,t)\,d\sigma = \int_0^1 \int_0^1 x\, g((1-y)x, yx)\,dx\,dy \tag{35}$$

For integrands of the type occurring with boundary integral equations, the new integrand will generally be quite smooth over the integration region $[0,1] \times [0,1]$. For a detailed error analysis of the use of this transformation, see Schwab and Wendland [19].

For calculating the right side of (35), we use Gauss-Legendre quadrature with respect to both variables. The needed integrations are carried out with the aid of the routines INTEGR and INTGR2. With all of the potential theory programs, the user is asked for an integration parameter NINTEG for the calculation of (35). Then iterated Gaussian quadrature with NINTEG nodes in each variable is used to calculate the integral. Our experiments have shown that NINTEG = 10 is more than adequate for all of the surfaces S we have used. Contrary to one's expectations, these singular integral calculations are less expensive than the nonsingular ones discussed earlier. Using a value of NINTEG which is somewhat too large will not lead to a significant increase in computation time.

4.1 Numerical integration programs

We provide two integration programs, named INTEGRATE and INTEGRATE_PLANAR. These programs do a numerical integration of

$$\int_S f(Q)\,dS_Q$$

They use a given triangulation $\mathcal{T}_N$, along with the methods in (26) and (25), respectively. These basic integration formulas are applied to the reformulation (7) for integrals over a triangle Δ_k. The user must supply the definition of the function f, and this is done in a function subprogram

FUNCTIONF (P, KFACE)

In this, P is the point at which f is to be evaluated; and as a possible aid in computing $f(P)$, KFACE is the face of the original surface S to which P belongs.

5 NONSINGULAR INTEGRAL EQUATIONS

Consider the numerical solution of the integral equation,

$$\lambda\,\rho(P) - \int_S K(P,Q)\,\rho(Q)\,dS_Q = g(P), \quad P \in S, \quad \lambda \neq 0 \tag{36}$$

In this section, we consider the case in which $K(P,Q)$ and $g(P)$ are considered "smooth" functions. For the numerical approximation of (36), we use the Nyström method with the numerical integration based on (26). This is the same numerical integration method as is implemented in INTEGRATE, which is discussed in §4.

To talk about the differentiability of functions defined over S, proceed as follows. Recall that S can be decomposed as

$$S = S_1 \cup \cdots \cup S_L$$

with each S_l a smooth surface; and for S itself smooth, $L = 1$. If $L > 1$, we assume that S_l is the image of a four times continuously differentiable function that is defined on a polygonal region in the plane. If $L = 1$, we assume that at each point $P \in S$, there is local representation of the surface which is four times continuously differentiable. [Note that this assumption disallows conical boundaries, since the differentiability assumption cannot hold at the vertex.]

5.1 The Collocation Method for Integral equations

For any $f \in C(S)$, we will say $f \in C^4(S)$ if $f \mid S_l \in C^4(S_l),\ l = 1, \cdots, L$. With this definition, we consider the differentiability of solutions $\rho \in C(S)$. It is easy to see that if (i) $g \in C^4(S)$, and (ii) with respect to both P and Q, the kernel function K is in $C^4(S)$, then the solution $\rho \in C^4(S)$.

The collocation solution of (36) is obtained by solving

$$(\lambda - \mathcal{P}_N \mathcal{K})\rho_N = \mathcal{P}_N g \tag{37}$$

in which $\mathcal{P}_N g$ denotes the piecewise quadratic interpolatory approximation of g. In more concrete form, we let

$$\rho_N(m_k(s,t)) = \sum_{j=1}^{6} \rho_N(v_{k,j})\ell_j(s,t), \qquad (s,t) \in \sigma, \quad k = 1, \cdots, N \tag{38}$$

and we solve for ρ_N at the node points by solving the linear system

$$\lambda\rho_N(v_i)-\sum_{k=1}^{N}\sum_{j=1}^{6}\rho_N(v_{k,j})\int_\sigma K(v_i,\widetilde{m}_k(s,t))\,\ell_j(s,t)\;|D_s\widetilde{m}_k\times D_t\widetilde{m}_k|\;d\sigma=g(v_i) \tag{39}$$

for $i=1,\cdots,N_v$.

The theory for this method is well-known and quite straightforward. Assuming (36) is uniquely solvable, we can show (39) is uniquely solvable for all sufficiently large N. Moreover, with the above assumptions on the smoothness of g, K, and S, it can be shown that

$$\|\rho-\rho_N\|_\infty=O(h^3)$$

Moreover, it can also be shown that

$$\max_{1\le i\le N_v}|\rho(v_i)-\rho_N(v_i)|=O(h^4) \tag{40}$$

See [13], [14], and [8, Chap. 5] for a detailed discussion of these results.

The collocation method can be implemented using the programs in TRIPACK. In particular, the quadrature routines INTKL1 and INTKL3. The former is used when the kernel function is smooth; and the latter is used when the kernel function is singular at a collocation point.

5.2 The Nyström Method

Recall the composite numerical integration formula which is based on (26) and whose convergence rate is given in (30). Write this formula as

$$\int_S \rho(Q)\,dS_Q\approx\sum_{j=1}^{N_v}w_{j,N}\;\rho(\,v_j),\quad \rho\in C(S) \tag{41}$$

We approximate (36) by applying this integration formula to obtain

$$\lambda\,\rho_N(P)-\sum_{j=1}^{N_v}w_{j,N}\,K(P,v_j)\,\rho_N(\,v_j)=g(P),\quad P\in S \tag{42}$$

As is well-known, this functional equation is equivalent to the linear system

$$\lambda\,\rho_N(v_i)-\sum_{j=1}^{N_v}w_{j,N}\,K(v_i,v_j)\,\rho_N(\,v_j)=g(P),\quad I=1,\cdots,N_v \tag{43}$$

Once a solution is obtained for this linear system, use the Nyström interpolation formula to extend it to a function $\rho_N(P)$ defined for all $P \in S$:

$$\rho_N(P) = \frac{1}{\lambda}\left[g(P) + \sum_{j=1}^{N_v} w_{j,N}\, K(P, v_j)\, \rho_N(\, v_j)\right], \quad P \in S \tag{44}$$

The equation (42) is written abstractly as

$$(\lambda - \mathcal{K}_N)\rho_N = g$$

From the well-known theory of collectively compact operator approximations (e.g. see [1]), an existence and convergence theory can be given for (42)-(44). Assuming (36) is uniquely solvable on $C(S)$, we have that the same will be true of (43) and (44), for all sufficiently large N, say $N \geq N_0$. Moreover, the inverses $(\lambda - \mathcal{K}_N)^{-1}$ will be uniformly bounded for $N \geq N_0$, and

$$\|\rho - \rho_N\|_\infty \leq \left\|(\lambda - \mathcal{K}_N)^{-1}\right\| \|\mathcal{K}\rho - \mathcal{K}_N\rho\|_\infty\,, \quad N \geq N_0 \tag{45}$$

When combined with (30), this leads to

$$\|\rho - \rho_N\|_\infty = O(h^4)$$

Two programs are given which implement the Nyström method. One is for smooth surfaces, and it is called INTEQN_SMOOTH; and the other is for piecewise smooth surfaces, and it is called INTEQN_PWSMOOTH. Both of these programs solve the linear system (43) by using Gaussian elimination, by means of LINPACK via a driver program called LINSYS. The routine INTEQN_SMOOTH is set up for the solution of problems on an ellipsoid; and the routine INTEQN_PWSMOOTH is set up for the solution of problems on an elliptical paraboloid. Both programs can be easily modified to handle other surfaces, as is described in their introductory comment statements.

5.3 Two-grid Iteration Methods

The linear system (43) increases rapidly in its order as N is increased. The order of (43) is N_v, and when N increases to $4N$, the size of the matrix for (43) increases by a factor of approximately 16. In Table 3, we give a typical set of values of the triangulation N, along with the order N_v and the storage

Table 3: Storage requirements

N	N_v	$8N_v^2$
8	18	2,592
32	66	34,848
128	258	532,512
512	1026	8,421,408
2048	4098	134,348,832

requirement (in bytes) $8N_v^2$. Clearly, the case with $N = 2048$ can be solved on only the largest of computers in terms of computer memory requirements; or some kind of virtual memory with efficient buffering will be needed. We do not include such programs in the general package being made available, but we have done some work on developing such programs.

With values of N that are such that the matrix for (43) can be stored in the computer's main memory, the use of direct methods (Gaussian elimination) will still be too expensive if N is at all large. For example, with most current workstations, $N = 512$ (and $N_v = 1026$) would be considered too large for a direct solution. In our package, the more efficient of the two-grid iteration methods described in [1, p. 142] is implemented for the problem with a smooth boundary. This program is called ITERATION, and it is the iterative analogue of INTEQN_SMOOTH.

6 POTENTIAL THEORY PROBLEMS

We provide tools for the numerical solution of Laplace's equation

$$\Delta u(P) = 0, \qquad P \in D \tag{46}$$

for a variety of regions D and boundary conditions on u. We give means for numerically evaluating potential integrals over the surface $S = \partial D$, and we use these to solve the following problems. We use piecewise quadratic collocation in solving the boundary integral equation reformulations of Laplace's equation, except where stated explicitly to the contrary.

1. The exterior Neumann problem: Let D be an unbounded region with a complement which is a simply connected region. Find $u \in C^2(\overline{D})$ that satisfies the equation (46), the Neumann boundary condition

$$\frac{\partial u(P)}{\partial \boldsymbol{\nu}_P} = f(P), \quad P \in S \tag{47}$$

   and the growth condition

$$|u(P)| = O(|P|^{-1}), \quad |\nabla u(P)| = O(|P|^{-2}) \quad \text{as} \quad |P| \to \infty$$

   In this, $\boldsymbol{\nu}_Q$ is the inner unit normal to S at Q, directed into the bounded region associated with the surface S.

2. The interior Dirichlet problem: Let D be a bounded simply connected region. Find $u \in C^1(\overline{D}) \cap C^2(D)$ that satisfies the equation (46) and the Dirichlet boundary condition

$$u(P) = f(P), \quad P \in S \tag{48}$$

The numerical methods we will use to solve these problems are based on reformulating them as boundary integral equations of the second kind. We do not investigate the use of boundary integral equations of the first kind; but the tools to do such are included in our package.

6.1 Single Layer Integral Approximations

Single layer potentials are integrals of the form

$$u(P) = \int_S \rho(Q) \frac{1}{|P-Q|} \, dS(Q), \quad P \in \mathbf{R}^3 \tag{49}$$

Our codes are directed towards the evaluation of these integrals for the case that $P \in S$. The numerical integration methods given earlier in §4 can be used to evaluate such integrals for $P \notin S$, and we do not discuss it further in this section.

To numerically evaluate (49) when ρ is known, we proceed much as in §4. Using piecewise quadratic interpolation of the surface, we write

$$u(P) \approx \sum_{k=1}^{N} \int_\sigma \rho(\widetilde{m}_k(s,t)) \frac{|D_s\widetilde{m}_k \times D_t\widetilde{m}_k|}{|P - \widetilde{m}_k(s,t)|} d\sigma \tag{50}$$

The sub-integrals are considered according to whether $P \in \Delta_k$ or $P \notin \Delta_k$. For the first case, use the subroutine INTKB6, which is based on the change of variables of (35). For the second case, we proceed in the manner described in §4 for handling singular and near singular integrands, given both before and after (33). A composite rule based on (28) is used, with the number of levels of subdivision based on the distance of P from Δ_k, with a maximum MAXLEV on the number of levels of subdivision specified by the user. The needed number of subdivisions is calculated in SETLEV, as was described earlier preceding (33). The numerical integration is implemented in the subroutine INTKB5. The routines INTKB5 and INTKB6 are restricted to having P be a node point in the triangulation.

For approximating integrals (49) when ρ is unknown, so as to discretize boundary integral equations containing it, we proceed using a variation of the above. Write

$$u(P) \approx \sum_{k=1}^{N} \sum_{j=1}^{6} \rho(v_{k,j}) \int_{\sigma} \frac{\ell_j(s,t)\,|D_s \widetilde{m}_k \times D_t \widetilde{m}_k|}{|P - \widetilde{m}_k(s,t)|} d\sigma \tag{51}$$

The integrals are again grouped according to whether $P \in \Delta_k$ or $P \notin \Delta_k$. For the first case, the subroutine INTKL4 is used, and it evaluates simultaneously the integrals over Δ_k for $j = 1, \cdots, 6$. For the second case, use INTKL3; and again the subroutine evaluates the integrals over Δ_k for $j = 1, \cdots, 6$. The methods used for the evaluation are the same as those described in the preceding paragraph. The routines INTKL3 and INTKL4 are restricted to having P be a node point in the triangulation. The results of these two routines are used to define collocation matrices associated with the single layer integral operator.

6.1.1 Normal derivative of single layer potentials

The Neumann problem (both interior and exterior) can be solved by representing the solution as a single layer potential (49). The density function ρ is obtained by solving a boundary integral equation involving the boundary integral operator

$$\mathcal{K}^* \rho(P) = \int_S \rho(Q) \frac{\partial}{\partial \boldsymbol{\nu}_P} \left[\frac{1}{|P - Q|} \right] dS(Q), \qquad P \in S^* \tag{52}$$

where $S^* = \{P \in S \mid \Omega(P) = 2\pi\}$. The quantity $\Omega(P)$ is the solid interior angle at $P \in S$; and when P is a point at which S is smooth, we have $\Omega(P) = 2\pi$. Since the normal derivative in (52) requires knowledge of the normal $\boldsymbol{\nu}_P$, the quantity $\mathcal{K}^*\rho(P)$ can be defined only at $P \in S^*$. It follows that $\mathcal{K}^*\rho(P)$ can be approximated only at $P \in S^*$; and therefore, collocation schemes that involve approximating $\mathcal{K}^*\rho(P)$ must restrict collocation nodes to belong to S^*. See the following subsection for a discussion of the approximation of $\boldsymbol{\nu}_P$.

6.2 Double Layer Integral Approximations

Double layer potentials are integrals of the form

$$u(P) = \int_S \rho(Q) \frac{\partial}{\partial \boldsymbol{\nu}_Q} \left[\frac{1}{|P - Q|} \right] dS(Q), \quad P \in \mathbf{R}^3 \tag{53}$$

Our codes are directed towards the evaluation of these integrals for the case that $P \in S$. This integral is discontinuous as P crosses the boundary S. The numerical integration methods given earlier in §4 can be used to evaluate such integrals for $P \notin S$, and we do not discuss this case further in this section.

When $P \in S$, we consider a modified version of (53), namely

$$\mathcal{K}\rho(P) = \int_S \rho(Q) \frac{\partial}{\partial \boldsymbol{\nu}_Q} \left[\frac{1}{|P - Q|} \right] dS(Q) + [2\pi - \Omega(P)]\rho(P), \quad P \in S. \tag{54}$$

The quantity $\Omega(P)$ is the solid interior angle at $P \in S$. When P is a point at which S is smooth, we have $\Omega(P) = 2\pi$. With this definition, $\mathcal{K} : C(S) \to C(S)$ is a bounded linear operator; but it is not compact unless S is sufficiently smooth, having no edges or corners. This has major implications for the error analysis of approximations of $\mathcal{K}$.

For $P \in S$, the integration of $\mathcal{K}\rho(P)$ proceeds almost exactly as for single layer potentials. The use of the approximate surface leads to the integral term in (54) being approximated by

$$\sum_{k=1}^{N} \sum_{j=1}^{6} \rho(v_{k,j}) \int_\sigma \ell_j(s,t) \frac{[P - \widetilde{m}_k(s,t)] \cdot [D_s\widetilde{m}_k \times D_t\widetilde{m}_k]}{|P - \widetilde{m}_k(s,t)|^3} d\sigma \tag{55}$$

For the calculation of the collocation integral approximations, we use the subroutines INTKL3 and INTKL4 just as is described above for single layer potentials. For the case of evaluating $\mathcal{K}\rho(P)$ when ρ is known, we also proceed in analogy with the single layer case, using INTKB5 and INTKB6. The only cautionary note is that one must be careful in setting up the triangulation so that $D_s\widetilde{m}_k \times D_t\widetilde{m}_k$ will be oriented in the correct direction, namely into the bounded portion of the region associated with the surface S. If one follows the examples given earlier in §3 for constructing subroutine INIT, then the correct direction for normals to the approximate surface will be calculated, namely

$$\widetilde{\nu}_Q = \frac{D_s\widetilde{m}_k \times D_t\widetilde{m}_k}{|D_s\widetilde{m}_k \times D_t\widetilde{m}_k|}, \quad Q = \widetilde{m}_k(s,t) \tag{56}$$

The only major difference with the case of single layer potentials is in the need to evaluate $\Omega(P)$, which is unknown in general at edges and corners of S. To evaluate $\Omega(P)$, we use the following well-known identity, which implicitly defines $\Omega(P)$:

$$\int_S \frac{\partial}{\partial \boldsymbol{\nu}_Q}\left[\frac{1}{|P-Q|}\right] dS(Q) + 2\pi - \Omega(P) = 2\pi, \qquad P \in S \tag{57}$$

Let K_N denote the collocation matrix associated with the double layer potential operator (54). Then we use

$$K_{i,l} = \sum_{\substack{k,j \\ v_{k,j}=v_l}} \int_\sigma \ell_j(s,t)\frac{[v_i - \widetilde{m}_k(s,t)]\cdot[D_s\widetilde{m}_k \times D_t\widetilde{m}_k]}{|v_i - \widetilde{m}_k(s,t)|^3} d\sigma + [2\pi - \Omega_N(v_i)]\delta_{i,l} \tag{58}$$

The value of $\Omega_N(v_i)$ is so determined that

$$\sum_{l=1}^{N_v} K_{i,l} = 2\pi, \quad i = 1,\cdots,N_v \tag{59}$$

This correction has been found to increase the order of convergence in all calculations involving the double layer potential; and this includes surfaces S which are smooth. With the definition of $\Omega_N(v_i)$ which this implies, we have found empirically that

$$\Omega(v_i) - \Omega_N(v_i) = O(h^3)$$

although we have been able to prove only an order of $O(h^2)$.

6.3 The Exterior Neumann Problem

We consider two approaches to solving the exterior Neumann problem (46)-(48).

6.3.1 Using Green's representation formula:

Using Green's representation formula, the solution $u(P)$ of (46)-(48) satisfies the following:

$$4\pi u(P) = \int_S f(Q) \frac{1}{|P-Q|} dS_Q - \int_S u(Q) \frac{\partial}{\partial \boldsymbol{\nu}_Q} \left[\frac{1}{|P-Q|} \right] dS_Q, \quad P \in D \tag{60}$$

Letting $P \to S$, we obtain the integral equation

$$\begin{aligned} 2\pi u(P) + \int_S u(Q) \frac{\partial}{\partial \boldsymbol{\nu}_Q} \left[\frac{1}{|P-Q|} \right] dS_Q + [2\pi - \Omega(P)] u(P) \\ = \int_S f(Q) \frac{1}{|P-Q|} dS_Q, \quad P \in S \end{aligned} \tag{61}$$

This is well-studied, for both the cases of smooth boundaries and piecewise smooth boundaries; see [8, Chap. 9]. For piecewise smooth boundaries, the seminal work on this equation, including its numerical solution by boundary integral equations, is that of Wendland [22]; and many of our error analyses have been based on generalizing and applying his ideas.

For a presentation and error analysis of the collocation methods we use, see [4], [10], and [8, Chap. 9]. We replace the integral operators in (61) by using the approximations described in (55) and (58). The resulting approximation is then collocated at the node points, leading to a linear system of equations of order N_v. The linear system is solved directly using standard routines from LINPACK.

Programs implementing the collocation method for given for several smooth and piecewise smooth surfaces. The names given are NEUMANN_###, where ### is an abbreviation for the name of the surface S. The surfaces included are

- ellipsoid

- "beanbag". This is a squashed and unsymmetric ellipsoid.
- elliptical paraboloid
- simplex
- L-block

By studying the programs for these cases, it should be clearer as to how to proceed with a problem and surface of your own design. With all of these programs, you will need to link with the following other programs: AUXPACK, INTPACK, and TRIPACK. You will also need to link to the appropriate version of INIT_###.

Polyhedral boundaries When the surface S is polyhedral, there are several desirable properties for the surface. First, the normals are constant over each face of the surface, and they are quite simple to calculate. Second, the surface S need not be approximated by $\widetilde{S}_N$, and again this leads to simpler programming, including the calculation of the Jacobians needed in the various collocation integral calculations. We have introduced special tools into TRIPACK for this case, to make it easier to deal with such surfaces. Their use is illustrated in the program NEUMN_EXT_POLYHED_S, in which S is a simplex.

6.3.2 Representation as a single layer potential:

Assuming the representation (49), we have that the single layer density ρ satisfies

$$(2\pi + \mathcal{K}^*)\,\rho = f$$

with $\mathcal{K}^*$ defined as in (52). To solve this with a collocation method with nodes chosen from S^*, we use the piecewise linear interpolation of (1) with $0 < \alpha < \frac{1}{3}$. The routine implementing this is named NEUMN_EXT_INDIR_ELLIP.

6.4 The Interior Dirichlet Problem

The ideas involved here are very much the same as for the exterior Neumann problem, and again we give to approaches to the problem.

6.4.1 Representation as a double layer potential:

We represent the solution $u(P)$ as a double layer potential:

$$u(P) = \int_S \rho(Q) \frac{\partial}{\partial \boldsymbol{\nu}_Q} \left[\frac{1}{|P - Q|} \right] dS_Q, \quad P \in D \tag{62}$$

The unknown density function ρ is determined by solving the integral equation

$$2\pi\rho(P) + \int_S \rho(Q) \frac{\partial}{\partial \boldsymbol{\nu}_Q} \left[\frac{1}{|P - Q|} \right] dS_Q + [2\pi - \Omega(P)]\rho(P) = f(P), \quad P \in S \tag{63}$$

This is essentially the integral equation (61), with a different right side to the equation. It can be solved numerically by the same methods as are used for the boundary integral equation (61) for the exterior Neumann problem. After finding ρ, the solution $u(P)$ is found with numerical integration, of the type described in §4, in and about (33). The needed number of levels of integration in calculating the composite numerical integration will vary with the distance of P from the boundary S, and it is calculated in the subroutine LEVDIR which is included as a part of all the programs listed below. The needed value of MAXLEV will be requested for each given point P, and it should increase as P approaches S more closely. The user should experiment to determine reasonable values of MAXLEV for the points P at which $u(P)$ is being evaluated.

The included programs for solving the interior Dirichlet problem are named DRCHLT, DRCHLT_SMOOTH_ITERT, and DRCHLT_PWSMOOTH_ITERT. The first program solves the discretized collocation system directly, using LINPACK programs; and the second and third programs solve this system with two-grid iteration, for smooth and piecewise smooth surfaces, respectively.

6.4.2 Representation as a single layer potential:

Let the solution be represented as single layer potential, as in (49). To determine the single layer density function ρ, we solve the first kind equation

$$\int_S \rho(Q) \frac{1}{|P - Q|} dS(Q) = f(P), \quad P \in S \tag{64}$$

We give a collocation method based on the use of piecewise constant interpolation at the centroids of the triangular elements. It is relatively straightforward to generalize this to higher order interpolation, both linear and quadratic, with the routines we have provided. The illustrative example of this is named 1ST_KIND_CENTROID, and it is for the simple case that S is an ellipsoid.

The linear system being solved in this case is

$$\sum_{j=1}^{N} \rho_N(P_j) \int_{\Delta_j} \frac{dS_Q}{|P_i - Q|} = f(P_i), \qquad i = 1, 2, ..., N$$

with $P_j = m_j\left(\frac{1}{3}, \frac{1}{3}\right)$ the "centroid" of triangular element Δ_j. There is no convergence or stability theory for this or any other collocation method for solving (64). Some numerical examples are given in [8, pp. 467-468].

7 THE RADIOSITY EQUATION

The radiosity equation of §1.2.3,

$$u(P) - \frac{\rho(P)}{\pi} \int_S u(Q) G(P, Q) V(P, Q)\, dS_Q = E(P), \qquad P \in S \tag{65}$$

occurs in both computer graphics and radiative transfer. In our programs, we assume $V(P, Q) \equiv 1$, which is equivalent to assuming there are no shadows generated on the surface; and this is known as an "unoccluded" surface. In using collocation to approximate (65), note that the kernel function $G(P, Q)$ (cf. (22)) is undefined when P is on an edge of S. Moreover, our interpolatory approximation $\widetilde{S}_N$ is such that the normal produced at points P on the common edge of two triangular elements Δ_i and Δ_j will be different (although only slightly if $\Omega(P) = 2\pi$) when produced from the two functions $\widetilde{m}_i(s, t)$ and $\widetilde{m}_j(s, t)$. Consequently, collocation nodes must be chosen interior to the triangular elements, and our piecewise quadratic interpolatory approximation of (4) cannot be used in solving (65).

The collocation solution of (65) is quite similar to what is done for the collocation solution of the exterior Neumann problem following (61) above. We include programs for both piecewise constant and piecewise linear collocation. The resulting systems are solved directly by Gaussian elimination, although iteration is a straightforward variant, as is discussed in [12] and [11].

For general discussions of the radiosity equation and its numerical solution, see [15] and [20]. For discussions of our contributions and methods, see [12] and [11]. The problems we include in the package are as follows.

1. RDSTY-E-CEN - Solve using the centroid method over an ellipsoidal surface.

2. RDSTY-E-LIN - Solve using piecewise linear collocation over an ellipsoidal surface.

3. RDSTY-R-CEN - Solve using the centroid method over a rectangular parallelopiped.

4. RDSTY-R-LIN - Solve using piecewise linear collocation over a rectangular parallelopiped.

5. RDSTY-RG-LIN - Solve using piecewise linear collocation over a rectangular parallelopiped. This uses a variant for calculating the collocation integrals, using the change of variables described following (34).

6. RDSTY-T2-LIN - Solve using piecewise linear collocation over the disconnected surface $S^{(2)}$ of (24).

7. RDSTY-T3-LIN - Solve using piecewise linear collocation over the disconnected surface $S^{(3)}$ of (24).

As test programs, these programs take a given radiosity $u(P)$ and produce the emissivity $E(P)$ of (65) by numerical integration. Then the program proceeds to solve for the radiosity, obtaining $u_N(P)$, which can then be compared with the true solution $u(P)$. Of necessity, this numerical integration to calculate $E(P)$ varies with the form of the surface S, and the user is asked to supply the needed numerical integration parameter INTEG_EM. In RDSTY-E-LIN, the integration for $E(P)$ involves spherical integration; whereas with RDSTY-R-LIN and the corresponding programs over the two piece surfaces $S^{(p)}$, the integration is reduced to product Gaussian quadrature over the unit square.

7.1 A Package for Very Large N.

We are developing a separate package for piecewise constant collocation with very large values of N. It includes both an iterative method of solution and a "clustering" method due to Hackbusch and Nowak [17]. We will also allow for occluded surfaces, i.e. $V(P,Q)$ is not equivalent to 1 in (65). The method being used is presented in [11]. When completed, it will be available from the same source as the present package.

8 ITERATION METHODS FOR POTENTIAL THEORY

As can be seen in table 3, the size of the linear systems being solved can become very large with only a few refinements of the initial triangulation. For this reason, iteration methods are needed for solving these systems. The methods we use are examples of two-grid iteration; and a general theory for this is given in [8, Chap. 6]. The actual methods we use are described and analyzed in [6] and [8, Chap. 9]. These methods are quite effective, although there has been little comparison of them with methods based on generalizing the conjugate gradient method, such as GMRES.

8.1 Smooth Surfaces

Our methods are a combination of the two-grid iteration method described in [1, p. 142] and ideas of Hackbusch [16]. In [1, p. 142], Nyström interpolation is used to extend functions defined on the coarse grid and to restrict functions defined on the fine grid. In our situtation, this is too expensive computationally; and instead we use prolongation [based on the quadratic interpolation of (4)] and restriction operators. For the case of smooth surfaces, a proof of the convergence of the two-grid iteration method is given in [6].

The programs which implement this method are given in NEUMN_###_ITERT. The two surfaces included are the ellipsoid and the "beanbag". As with the earlier case in which the system is solved directly, you will need to link with the following other programs: AUXPACK, INTPACK, TRIPACK, and the appropriate INIT_### subroutine.

In addition, a third program called NEUMN_ELL_ITERT_T, based on S an ellipsoid, contains computations of elapsed time. The program requires a machine dependent timing routine (called SECNDS in our program) to give the machine clock. It is quite interesting to compare the computational costs of the different parts of the solution process. In spite of great attention having been given to efficient numerical integration of collocation integrals, such integrations are still, by far, the most costly part of the solution process.

8.2 Piecewise Smooth Surfaces

For surfaces which are only piecewise smooth, the standard two-grid iteration method will often not converge. See [4] and [6] for a more extensive discussion of this. To retain the use of two-grid iteration, we first precondition the system (and its coarse grid counterpart). In essence, we partition the node points into two subsets, called "singular" and "nonsingular". The portion of the linear system that corresponds to the singular nodes is solved exactly. Then two-grid iteration is applied to the modified linear system. A more complete description of this method is given in [6], and defer to that paper for a discussion of the method. Generally, the method converges, but it is not as well-behaved an iteration method as is the original two-grid method for the smooth surface case.

The programs implementing this method are named NEUMN_###_ITERT. The surfaces included are the L-block, the simplex, and the elliptical paraboloid. The programs print out the norms of the individual corrections in the iteration, along with the ratios by which they are decreasing.

9 NONLINEAR PROBLEMS

Work has been done for solving the nonlinear boundary value problem

$$\Delta u(P) = 0, \quad P \in D \tag{66}$$

$$\frac{\partial u(P)}{\partial \boldsymbol{\nu}_P} = g(P, u(p)) - f(P), \quad P \in S = \partial D \tag{67}$$

The region D is an open connected region in $\mathbf{R}^3$ with a smooth connected boundary S, and $\boldsymbol{\nu}_P$ is the interior unit normal at $P \in S$. We seek a solution

$u \in C^2(D) \cap C^1(\overline{D})$. The function $g(P, v)$ is assumed to be continuous for $(P, v) \in S \times \mathbf{R}$, although this can be relaxed somewhat.

Using Green's representation formula for harmonic functions, the function u satisfies

$$u(P) = \frac{1}{4\pi} \int_S \frac{\partial u(Q)}{\partial \boldsymbol{\nu}_Q} \frac{dS(Q)}{|P-Q|} - \frac{1}{4\pi} \int_S u(Q) \frac{\partial}{\partial \boldsymbol{\nu}_Q} \left[\frac{1}{|P-Q|} \right] dS(Q), \quad P \in D \tag{68}$$

Letting P tend to a point on S, and using the boundary condition (67), we obtain the nonlinear boundary integral equation

$$2\pi\, u(P) - \int_S u(Q) \frac{\partial}{\partial \boldsymbol{\nu}_Q} \left[\frac{1}{|P-Q|} \right] dS(Q) = \int_S [g(Q, u(Q)) - f(Q)] \frac{dS(Q)}{|P-Q|} \tag{69}$$

for $P \in S$. This can be solved for $u(P)$ on S. The normal derivative of u can be obtained from (67), and (68) then yields $u(P)$ at all $P \in D$.

Modifications of the numerical methods of sections 6 and 7 are used to solve (69). This work is discussed in [7], and we include an associated program with the present package, named NONLIN-ELL. This is for an ellipsoidal surface, but it is straightforward to use it for other surfaces.

References

[1] K. Atkinson, *A Survey of Numerical Methods for the Solution of Fredholm Integral Equations of the Second Kind,* SIAM Pub., Philadelphia, PA, 1976.

[2] K. Atkinson, Piecewise polynomial collocation for integral equations on surfaces in three dimensions, *Journal of Integral Equations* **9** (1985, supplementary issue), pp. 25-48.

[3] K. Atkinson, Solving integral equations on surfaces in space, in *Constructive Methods for the Practical Treatment of Integral Equations,* ed. by G. Hämmerlin and K. Hoffman, Birkhäuser, Basel, 1985, pp.20-43.

[4] K. Atkinson, An empirical study of the numerical solution of integral equations on surfaces in $\mathbf{R}^3$, *Reports in Computational Mathematics* **1** (1989), University of Iowa, Iowa City, Iowa.

[5] K. Atkinson, A survey of boundary integral equation methods for the numerical solution of Laplace's equation in three dimensions, in *Numerical Solution of Integral Equations*, ed. by M. Golberg, Plenum Press, New York, 1990, pp. 1-34.

[6] K. Atkinson, Two-grid iteration methods for linear integral equations of the second kind on piecewise smooth surfaces in $\mathbf{R}^3$, *SIAM Journal of Scientific and Statistical Computing* **15** (1994), pp. 1083-1104.

[7] K. Atkinson, The numerical solution of a nonlinear boundary integral equation on smooth surfaces, *IMA Journal on Numerical Analysis* **14** (1994), pp. 461-483.

[8] K. Atkinson, *The Numerical Solution of Fredholm Integral Equations of the Second Kind*, Cambridge Univ. Press, 1997.

[9] K. Atkinson, The numerical solution of boundary integral equations, in *The State of the Art in Numerical Analysis*, ed. by I. Duff and G. Watson, Clarendon Press, Oxford, 1997, pp. 223-259.

[10] K. Atkinson and D. Chien, Piecewise polynomial collocation for boundary integral equations, *SIAM J. Sci. Stat. Comp.* **16** (1995), pp. 651-681.

[11] K. Atkinson and D. Chien, A fast matrix-vector multiplication method for solving the radiosity equation, submitted to *SIAM J. on Scientific Computing*.

[12] K. Atkinson and G. Chandler, The collocation method for solving the radiosity equation for unoccluded surfaces, *Journal of Integral Eqns & Applics*, 1998, to appear.

[13] D. Chien, *Piecewise Polynomial Collocation for Integral Equations on Surfaces in Three Dimensions*, PhD thesis, Univ. of Iowa, Iowa City, Iowa, 1991.

[14] D. Chien, Piecewise polynomial collocation for integral equations with a smooth kernel on surfaces in three dimensions, *Journal of Integral Equations and Applications* **5** (1992), pp. 315-344.

[15] M. Cohen and J. Wallace, *Radiosity and Realistic Image Synthesis*, Academic Press, New York, 1993.

[16] W. Hackbusch, Die schnelle Auflösung der Fredholmschen Integralgleichungen zweiter Art, *Beiträge Numerische Math.* **9** (1981), pp. 47-62.

[17] W. Hackbusch and Z. Nowak, On the fast matrix multiplication in the boundary element method by panel clustering, *Numerische Math.* **54** (1989), pp. 463-491.

[18] J. Lyness and D. Jesperson, Moderate degree symmetric quadrature rules for the triangle, *Institute of Mathematics and Its Applications* **15** (1975), pp. 19-32.

[19] C. Schwab and W. Wendland, On numerical cubatures of singular surface integrals in boundary element methods, *Numerische Mathematik* **62** (1992), pp. 343-369.

[20] F. Sillion and C. Puech, *Radiosity and Global Illumination*, Morgan Kaufmann Pub., San Francisco, 1994.

[21] A. Stroud, *Approximate Calculation of Multiple Integrals*, Prentice-Hall, New Jersey, 1971.

[22] W. Wendland, Die Behandlung von Randwertaufgaben im $\mathbf{R}^3$ mit Hilfe von Einfach- und Doppelschichtpotentialen, *Numerische Mathematik* **11** (1968), pp. 380-404.

[23] Yajun Yang and K. Atkinson, Numerical integration for multivariable functions with point singularities, *SIAM J. Numerical Analysis* **32** (1995), pp. 969-983.