LBL-27616
Preprint

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Physics Division

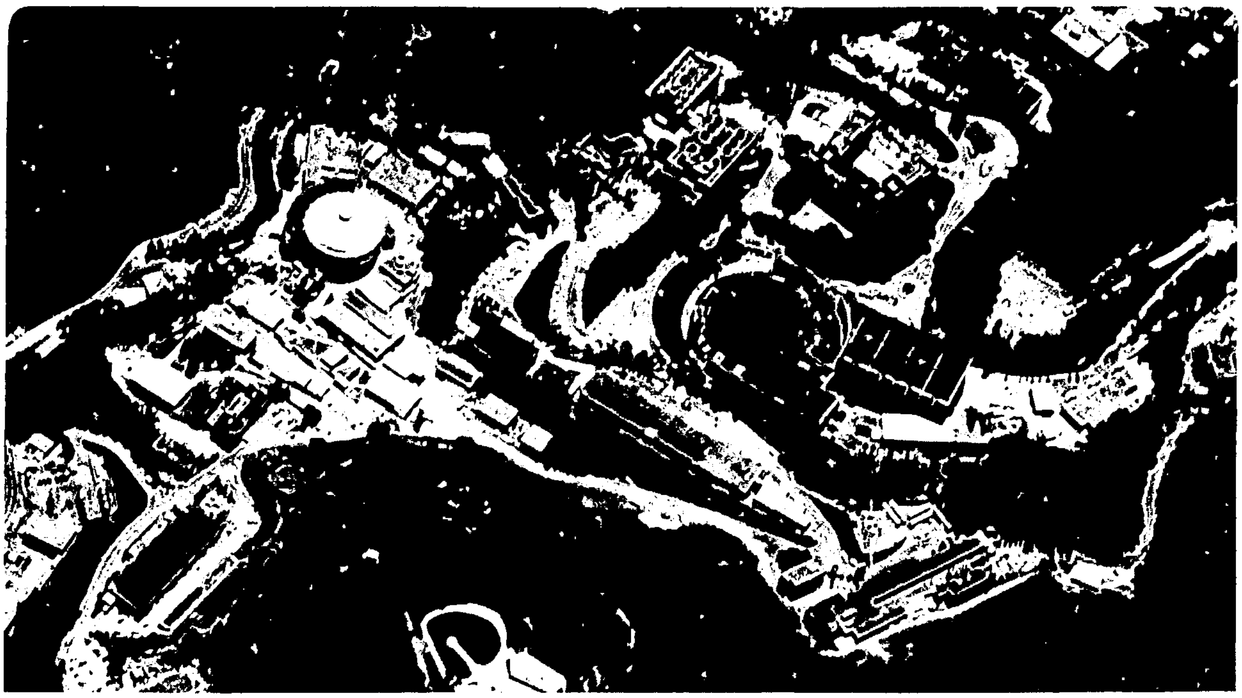
Mathematics Department

To be submitted for publication

An Application of Panel Methods to Random Vortex Simulations

D. Summers

August 1989



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

LBL-27616

AN APPLICATION OF PANEL METHODS TO RANDOM VORTEX SIMULATIONS[1]

David Summers[2]

Department of Mathematics
University of California, Berkeley
and
Department of Mathematics, Physics Division
Lawrence Berkeley Laboratory
Berkeley, California 94720

August 1989

[1]Supported by the U.K. Science and Engineering Research Council under grant number GR/F/31779 and by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy under contract DE-AC03-76SF00098.

[2]Visiting from Napier Polytechnic of Edinburgh, 219 Colinton Road, Edinburgh EH14 1DJ, Scotland.

Abstract

The flow of a slightly viscous fluid over a three-dimensional non-lifting bluff body of arbitrary geometry can, in principle, be modelled using the random vortex method. In this method the impermeability condition at the solid surface of the obstacle must be re-established at each time-step. Integral equation methods can be used to construct a potential flow which, when added to the flow evolved at the previous time-step, effects impermeability in the present time-step. The relevant equations are described and their numerical solution using a low-order discretisation (Nyström) is discussed. The method is illustrated for the cases of flow over a two-dimensional prism of rectangular cross-section and for flow over a cubic obstacle. Fortran listings are provided of these illustrations.

An Application of Panel Methods to Random Vortex Simulations

1. Introduction

The random vortex method can be applied to the problem of simulating high Reynolds number Navier-Stokes flow around bluff obstacles. The application of the method to the two-dimensional problem of flow around a cylinder was studied by Chorin [1]. In this method at each time-step the impermeability condition must be imposed, and Chorin achieved this by using an integral equation technique to construct a curl-free potential flow field which, when added to the existing flow field effected zero normal component of flow at the cylinder surface. Summers et al [5] applied an integral equation method to a random vortex simulation of flow over a two-dimensional surface-mounted obstacle of arbitrary cross-section. Cheer [2,3] used image techniques and analytical conformal mapping in a random vortex simulation to achieve the impermeability condition, for flow around a cylinder and around Joukowski aerofoils. Numerical conformal mapping has been used in the context of various applications of the random vortex method to interior problems (e.g. Ghoniem and Cagnon [4]), and in principle such a technique can be applied to exterior two-dimensional problems, although the method does not generalise to three-dimensions.

The impermeability condition can also be imposed by using grid-based methods. For exterior flow problems involving infinite or semi-infinite domains, integral equation methods have an advantage over Eulerian grid methods in the sense that in the former approach, computational effort does not need to be applied to the problem of imposing conditions at infinity. This is especially important for flow at high Reynolds number in which extensive wake structure may develop and vortical motion advects downstream. In any case, an important feature of the random vortex method is its grid-free character, and an integral equation technique (or panel method) formulation of the impermeability condition at a solid boundary is compatible with such a grid-free strategy.

In this report a statement of the problem of imposing the impermeability condition in random vortex methods is presented. The use of integral equations for two- and three-dimensional problems is discussed. A low-order discretisation (Nyström discretisation) of the relevant equation is described as well as a corresponding solution algorithm developed by Hess and Smith

[6]. A descriptive discussion of the accuracy and stability of this solution procedure is provided. By way of illustrating the basic method, details of two particular applications are included (with Fortran listings): inviscid flow over a two-dimensional prism with rectangular cross-section, and inviscid flow over a three-dimensional cuboid.

Detailed background analysis of the numerical solution of integral equations arising in potential theory can be found in various sources: Hess and Smith [6]; Jaswon and Symm [7]; Bose [8]; Rokhlin [9]; Kellogg [10]; Delves and Walsh [11]; Goldberg [12]; Atkinson [13].

2. Imposing the impermeability condition in Vortex Methods

To model numerically the flow of a viscous fluid over a three-dimensional body B of arbitrary geometry requires that the velocity field satisfy two conditions everywhere on the boundary of the body, ∂B: the component of the velocity field $\mathbf{u}$ normal to ∂B must vanish, i.e. the impermeability condition

$$\mathbf{u} \cdot \mathbf{n} = 0 \text{ on } \partial \mathrm{B} \quad ; \tag{1}$$

and the condition that the tangential component of the velocity field at the boundary must vanish, i.e. the no-slip condition,

$$\mathbf{u} \cdot \mathbf{s} = 0 \text{ on } \partial \mathrm{B} \quad . \tag{2}$$

This is to say the total velocity, $\mathbf{u}$, at the body's surface must be zero.

To solve the flow evolution problem associated with flow over a bluff body using a random vortex strategy, is to invoke a time-splitting. A time interval $dt = t_{k+1} - t_k$ is formally partitioned into two half-intervals. During the first half-interval $(t_k, t_{k+\frac{1}{2}})$ an Euler flow problem is addressed; the vorticity field of the interior flow is discretised into a collection of vortex core elements whose interaction with each other and whose translation in the collectively induced flow field represents the Lagrangian dynamics of the flow. (The vorticity field in the neighbourhood of ∂B is discretised into a collection of vortex sheet elements whose dynamics are described by the corresponding boundary layer approximation.) During the subsequent half-interval $(t_{k+\frac{1}{2}}, t_{k+1})$ a stochastic model of viscous diffusion is invoked; a gaussian random-walk displacement is imparted to each vortex element.

The task of solving the Euler problem during $(t_k, t_{k+\frac{1}{2}})$ requires a flow field $\mathbf{u}^{\mathrm{k+1}}$ to be determined from a field $\mathbf{u}^{\mathrm{k}}$ which is consistent with the inviscid vorticity transport equation and consistent with boundary conditions

(1) and (2) on ∂B. At the beginning of each time-step neither (1) nor (2) are in general satisfied. The strategy is first to construct an irrotational flow field $\mathbf{u}^{\phi}$ which, when added to $\mathbf{u}^{\mathrm{k}}$ will effect condition (1) on ∂B. The curl-free character of $\mathbf{u}^{\phi}$ will ensure that this field will not directly contribute vorticity to the existing vorticity field. Having constructed a field which has zero normal component at the surface of the obstacle,

$$(\mathbf{u}^{\mathrm{k}} + \mathbf{u}^{\phi}) \cdot \mathbf{n} = 0 \text{ on } \partial \mathrm{B} \tag{3}$$

the tangential component of this field

$$(\mathbf{u}^{\mathrm{k}} + \mathbf{u}^{\phi}) \cdot \mathbf{s} \text{ on } \partial \mathrm{B}$$

can be used to generate a vortex sheet precisely of strength sufficient to effect no-slip at each point of ∂B. The vortex elements created in this way are hence made to advect in the flow field induced by the current vortex elements and in the current irrotational field $\mathbf{u}^{\phi}$. To each element is imparted a random walk displacement during the second half interval $(t_{k+\frac{1}{2}}, t_{k+1})$. The velocity field which results after the completion of advection and diffusion is the field $\mathbf{u}^{\mathrm{k}+1}$.

The present discussion concerns the construction of an irrotational velocity field $\mathbf{u}^{\phi}$ which, when added to $\mathbf{u}^{\mathrm{k}}$ will effect (1) on ∂B. The problem of determining such a field — one exterior to a closed convex body with continuous curvature and with imposed gradient boundary conditions to be satisfied on its surface, is a familiar problem from potential theory.

3. General Statement of the Potential Problem

We consider a solid obstcle B whose interior is denoted by B^{i}, whose boundary is ∂B, and whose exterior is B^{e}.

We denote the interior Green's function by $G^{in}(\mathbf{r}|\mathbf{r}')$, $\mathbf{r}, \mathbf{r}' \in \mathrm{B}^{\mathrm{i}} \cup \partial \mathrm{B}$ and seek to determine a harmonic field $\phi(\mathbf{r}), \mathbf{r} \in \partial\mathrm{B}$ such that this field satisfies $\nabla^2 \phi(\mathbf{r}) = \mathbf{0}$ in the interior of B, and prescribed normal derivatives $\frac{\partial \phi}{\partial n} = g(\mathbf{r}), \mathbf{r} \in \partial \mathrm{B}$, where n represents the variable outward normal to the surface.

For two-dimensional problems the appropriate Green's function is

$$G^{in}(\mathbf{r}_{\mathrm{i}}|\mathbf{r}_{\mathrm{j}}) = \frac{1}{2\pi} \ln|\mathbf{r}_{\mathrm{i}} - \mathbf{r}_{\mathrm{j}}| \tag{4}$$

and for three dimensional problems,

$$G^{in}(\mathbf{r}_{\mathrm{i}}|\mathbf{r}_{\mathrm{j}}) = -\frac{1}{4\pi} \frac{1}{|\mathbf{r}_{\mathrm{i}} - \mathbf{r}_{\mathrm{j}}|} \tag{5}$$

The scalar velocity potential $\phi(\mathbf{r}_\mathrm{i})$ associated with conditions to be imposed at ∂B can be expressed in terms of an integration over a source (or doublet) distribution $\sigma(\mathbf{r}_\mathrm{j})$. This is to say

$$\phi(\mathbf{r}_\mathrm{i}) = \int_{\partial \mathrm{B}} \sigma(\mathbf{r}_\mathrm{j})\ \mathrm{G}^{\mathrm{in}}(\mathbf{r}_\mathrm{i}|\mathbf{r}_\mathrm{j})\mathrm{dS}_\mathrm{j} \tag{6}$$

where dS_j represents a linear increment of a boundary centred at $\mathbf{r}_\mathrm{j}$ for a two-dimensional problem, and represents a surface increment of a boundary with centroid $\mathbf{r}_\mathrm{j}$ for a three-dimensional problem. The terminology "body point" is used to indicate $\mathbf{r}_\mathrm{j}$; the point $\mathbf{r}_\mathrm{i}$ is called the "control point". Derivation of the formalism can be found, for example, in chapter 13 of reference [7], or in appendix 1 of reference [16].

The grad operator applied to (6) will determine an expression for the velocity field on ∂B,

$$\mathbf{u}^\phi = \nabla\phi = \int_{\partial \mathrm{B}} \sigma(\mathbf{r}_\mathrm{j})\ \nabla\mathrm{G}^{\mathrm{in}}(\mathbf{r}_\mathrm{i}|\mathbf{r}_\mathrm{j})\mathrm{dS}_\mathrm{j} \tag{7}$$

The normal component to ∂B of $\mathbf{u}^\phi$ at $\mathbf{r}_\mathrm{i}$ is hence

$$\mathbf{u}^\phi \cdot \mathbf{n}_\mathrm{i} = \int_{\partial \mathrm{B}} \sigma(\mathbf{r}_\mathrm{j}) \frac{\partial}{\partial \mathrm{n}_\mathrm{i}}\ \mathrm{G}^{\mathrm{in}}(\mathbf{r}_\mathrm{i}|\mathbf{r}_\mathrm{j})\ \mathrm{dS}_\mathrm{j} \tag{8}$$

where $\mathbf{n}_\mathrm{i}$ is the unit normal outward to ∂B at $\mathbf{r}_\mathrm{i} \in \partial\mathrm{B}$.

The kernel of this line integration is singular at $\mathbf{r}_\mathrm{i} = \mathbf{r}_\mathrm{j}$; analysis of the singularity shows that (for continuous ∂B and continuous $\sigma(\mathbf{r}_\mathrm{j})$) this can be formally removed from the integral so that

$$\mathbf{u}^\phi \cdot \mathbf{n}_\mathrm{i} = \frac{1}{2}\,\sigma(\mathbf{r}_\mathrm{i})\ +\ \int_{\partial \mathrm{B}} \sigma(\mathbf{r}_\mathrm{j})\ \frac{\partial}{\partial \mathrm{n}_\mathrm{i}}\ \mathrm{G}^{\mathrm{in}}(\mathbf{r}_\mathrm{i}|\mathbf{r}_\mathrm{j})\ \mathrm{dS}_\mathrm{j}\ , \tag{9}$$

Our object is to construct an irrotational velocity field $\mathbf{u}^\phi$ which satisfies the prescribed conditions (3) on ∂B. Since $\mathbf{u}^\mathrm{k}$ is known from the vorticity distribution resulting from the advection and random walk processes of the previous time-step, equation (9) can be viewed as an integral equation with known inhomogeneity vector,

$$\frac{1}{2}\,\sigma(\mathbf{r}_\mathrm{i})\ +\ \int_{\partial \mathrm{B}} \sigma(\mathbf{r}_\mathrm{j})\ \frac{\partial}{\partial \mathrm{n}_\mathrm{i}}\mathrm{G}^{\mathrm{in}}(\mathbf{r}_\mathrm{i}|\mathbf{r}_\mathrm{j})\ \mathrm{dS}_\mathrm{j}\ =\ -\mathbf{u}^\mathrm{k} \cdot \mathbf{n}_\mathrm{i} \tag{10}$$

This equation can be solved numerically for the distribution $\sigma(\mathbf{r}_\mathrm{j})$. Hence $\mathbf{u}^\phi$ can be determined by numerical integration of (7).

4. The two-dimensional problem

The kernel of equation (10) can be expressed in two-dimensional cartesian form from (4) as

$$\frac{\partial}{\partial n_i} \ln |\mathrm{r_i} - \mathrm{r_j}| = \frac{(x_i - x_j)n_{xi} + (y_i - y_j)n_{yi}}{|\mathrm{r_i} - \mathrm{r_j}|^2}$$

where the unit vector (n_{xi}, n_{yi}) is normal to ∂B at $\mathrm{r_i}$. If β_i is the slope angle of ∂B with respect to the x-axis at $\mathrm{r_i}$ then $n_{xi} = \sin \beta_i$ and $n_{yi} = \cos \beta_i$.

The boundary can be partitioned into a discrete set of N line-segment panels, each centred at a body-point (x_i, y_i). Over each panel there will be assumed to be a uniform source distribution σ_i. Such a representation, called Nyström discretisation, replaces the integral in (10) with a trapezoidal rule quadrature. This is to say we will content ourselves here with the error $O(h^3)$ for this problem, rather than resort to higher-order quadratures, since trapezoidal rule has an obvious and simple generalisation to the multiple integration required in three-dimensional problems.

If the length of the panel S_j at (x_j, y_j) is h_j, we can write equation (10) explicitly as the matrix equation

$$\sum_{j=1}^{N} C_{ij}\, \sigma_j = b_i \tag{11}$$

where

$$b_i = -\,\mathrm{u^k} \cdot \mathrm{n_i}, \tag{12a}$$

$$C_{ij} = \frac{1}{2\pi} \int_{S_j} \frac{(x_i - \xi)n_{xi} + (y_i - \eta)n_{yi}}{(x_i - \xi)^2 + (y_i - \eta)^2}\, d\ell, \quad i \neq j \tag{12b}$$

$$C_{ii} = \frac{1}{2} \tag{12c}$$

with dummy variables of integration $(\xi, \eta) \in S_j$; $d\ell$ is a line differential lying in S_j. The coefficient C_{ij} can be expressed in alternative notation

$$C_{ij} = A_x n_{xi} + A_y n_{yi} \tag{13a}$$

where

$$A_x = \frac{1}{2\pi} \int_{S_j} \frac{(x_i - \xi)\, d\ell}{(x_i - \xi)^2 + (y_i - \eta)^2} \tag{13b}$$

and

$$A_y = \frac{1}{2\pi}\int_{S_j} \frac{(y_i - \eta)\ d\ell}{(x_i - \xi)^2 + (y_i - \eta)^2} \tag{13c}$$

Retaining the first term in a multipole expansion of (12b) the following approximate relationship is achieved for small S_j

$$C_{ij} \simeq \frac{h_j}{2\pi}\left(\frac{(y_i - y_j)\cos\beta_i - (x_i - x_j)\sin\beta_i}{(x_i - x_j)^2 + (y_i - y_j)^2}\right) \tag{14}$$

Alternative to this approach, an exact integration of (12b) over each panel segment can be performed. This is most easily done by transforming into a panel-based coordinate system.

If the direction cosines of the jth panel are (s_x, s_y) — which can also be expressed in terms of the unit normal vector at $\mathbf{r}_j$, $(n_{yj}, -n_{xj})$ — then the position vector of $\mathbf{r}_j$ relative to $\mathbf{r}_i$ is given in a coordinate system with origin at (x_j, y_j) by

$$\begin{pmatrix} s_x & s_y \\ -s_y & s_x \end{pmatrix} \begin{pmatrix} x_i - x_j \\ y_i - y_j \end{pmatrix} = \begin{pmatrix} \tilde{x}_{ij} \\ \tilde{y}_{ij} \end{pmatrix} \tag{15}$$

where $\tilde{x}_{ij}$ is effectively a distance component from a body point to a control point taken parallel to the jth panel and $\tilde{y}_{ij}$ is a corresponding distance component taken normal to this panel. The subscripts ij will be suppressed in the following notation.

Explicit integration over a panel in (11) can now be performed in this simplifying choice of coordinates. The component $A_{\tilde{x}}$ parallel to the panel is

$$\begin{aligned} A_{\tilde{x}} &= \frac{1}{2\pi}\int_{-h_j/2}^{h_j/2} \frac{(\tilde{x} - \xi)}{(\tilde{x} - \xi)^2 + \tilde{y}^2} d\xi \\ &= \frac{1}{4\pi}\ln\left(\frac{(\tilde{x} + \frac{h_j}{2})^2 + \tilde{y}^2}{(\tilde{x} - \frac{h_j}{2})^2 + \tilde{y}^2}\right) \end{aligned} \tag{16a}$$

and the component perpendicular to the panel is

$$A_{\tilde{y}} = \frac{1}{2\pi}\int_{-h_j/2}^{h_j/2} \frac{\tilde{y}}{(\tilde{x} - \xi)^2 + \tilde{y}^2} d\xi$$

$$= \frac{1}{2\pi}\left(\tan^{-1}\left(\frac{\tilde{x} + \frac{h_j}{2}}{\tilde{y}}\right) - \tan^{-1}\left(\frac{\tilde{x} - \frac{h_j}{2}}{\tilde{y}}\right)\right) \tag{16b}$$

Having evaluated these integrals in panel-based coordinates, we can now perform the inverse rotation into global coordinates by

$$\begin{pmatrix} s_x & -s_y \\ s_y & s_x \end{pmatrix} \begin{pmatrix} A_{\tilde{x}} \\ A_{\tilde{y}} \end{pmatrix} = \begin{pmatrix} A_x \\ A_y \end{pmatrix} \tag{17}$$

and hence C_{ij} can be written in a form identical to (13a). Because equations (16) are bounded at $(\tilde{x}, \tilde{y}) = (0,0)$, there is no necessity to formally remove the self-potential term in the integral equation.

Upon solving the linear system (11) for $\{\sigma_j\}$, equation (7) can be solved for $\mathbf{u}^\phi$ on ∂B. In the context of the present formulation, this requires simply the following summations to be performed:

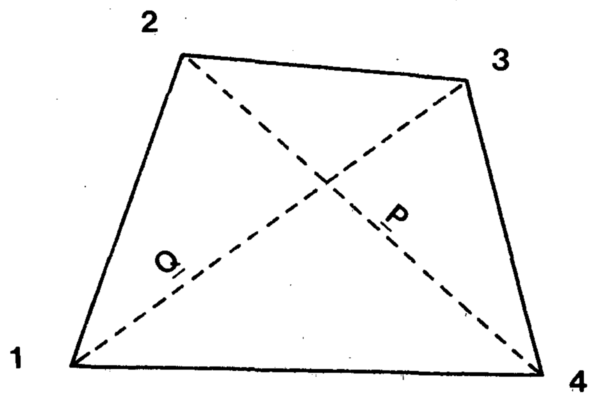
$$\begin{pmatrix} u_x^\phi \\ u_y^\phi \end{pmatrix} = \sum_{j=1}^{N} \sigma_j \begin{pmatrix} s_x & -s_y \\ s_y & s_x \end{pmatrix} \begin{pmatrix} A_{\tilde{x}} \\ A_{\tilde{y}} \end{pmatrix} \tag{18}$$

To determine the field in the interior of the fluid, $A_{\tilde{x}}$ and $A_{\tilde{y}}$ must be recomputed for control points $\mathrm{r_i} \in \mathrm{B^e}$ using a consistent sense for the panel-based coordinate system.

5. Three-dimensional problems

The solution strategy developed previously for two-dimensional problems can be extended to three-dimensions.

First, we should note the following description of body panel geometry for flat quadrilateral panels in contiguous distribution over ∂B. We consider a flat panel with four vertices, with position vectors labelled clockwise:



The position vector of the centroid of the panel is $\mathbf{R}_o = \frac{1}{4}\sum_{j=1}^{4}\mathbf{r}_j$, with components denoted (X_o, Y_o, Z_o). We define "diagonal vectors" $\mathbf{Q} = \mathbf{r}_3 - \mathbf{r}_1$, and $\mathbf{P} = \mathbf{r}_4 - \mathbf{r}_2$. A unit outward normal vector to the panel can be constructed from

$$\mathbf{n} = -\mathbf{k} = -\frac{\mathbf{Q}\times\mathbf{P}}{|\mathbf{Q}\times\mathbf{P}|} \tag{19}$$

A second unit vector, normal both to $\mathbf{n}$ and to $\mathbf{i} = \frac{\mathbf{Q}}{|\mathbf{Q}|}$ can be constructed as

$$\mathbf{j} = \frac{\mathbf{Q}\times\mathbf{P}\times\mathbf{Q}}{|\mathbf{Q}\times\mathbf{P}\times\mathbf{Q}|} \tag{20}$$

Unit vectors $(\mathbf{i},\mathbf{j},\mathbf{k})$ defined in this way constitute a right-handed system.

A position vector of a point in space (e.g. a control point) relative to the centroid of a panel is $\mathbf{r}' = \mathbf{r} - \mathbf{R}_o$. This position vector can be expressed in a panel-based coordinate system, namely one characterised by unit vectors $(\mathbf{i},\mathbf{j},\mathbf{k})$, by the rotation

$$\begin{pmatrix} i_x & i_y, & i_z \\ j_x & j_y & j_z \\ k_x & k_y & k_z \end{pmatrix} \begin{pmatrix} x - X_o \\ y - Y_o \\ z - Z_o \end{pmatrix} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} \tag{21}$$

Similarly, panel-based coordinates can be transformed into global coordinates by the inverse rotation

$$\begin{pmatrix} i_x & j_x & k_x \\ i_y & j_y & k_y \\ i_z & j_z & k_z \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \begin{pmatrix} x - X_o \\ y - Y_o \\ z - Z_o \end{pmatrix} \tag{22}$$

The normal derivative of the Green's function (5) has the form

$$\frac{\partial}{\partial n_i}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} = -\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|^3}\left\{(x_i - x_j)n_{xi} + (y_i - y_j)n_{yi} + (z_i - z_j)n_{zi}\right\} .$$

Equation (10) can now be written

$$\sum_{j=1}^{N} C_{ij}\ \sigma_j\ = b_i \tag{23}$$

with

$$b_i = -\mathbf{u}^k \cdot \mathbf{n}_i \tag{24a}$$

$$C_{ij} = \frac{1}{4\pi} \iint_{S_j} \frac{(x_i - \xi)n_{xi} + (y_i - \eta)n_{yi} + (z_i - \varsigma)n_{zi}}{((x_i - \xi)^2 + (y_i - \eta)^2 + (z_i - \varsigma)^2)^{\frac{3}{2}}} da, \quad i \neq j \tag{24b}$$

$$C_{ii} = \frac{1}{2} \tag{24c}$$

with dummy variables $(\xi, \eta, \varsigma) \in S_j$; da is a surface differential in S_j. The coefficient C_{ij} can be written in analogy with (13a) as:

$$C_{ij} = A_x n_{xi} + A_y n_{yi} + A_z n_{zi} \tag{25}$$

The coefficients can be approximated by retaining the first term in a multipole expansion,

$$C_{ij} \simeq \frac{\delta a_j}{4\pi} \left(\frac{(x_i - x_j)\cos\beta_x + (x_i - x_j)\cos\beta_y + (z_i - z_j)\cos\beta_z}{((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)^{\frac{3}{2}}} \right) \tag{26}$$

where $\cos\beta_x$ etc. represent the direction cosines of the panel, and δa_j is the area of the panel S_j.

Again, the coefficients C_{ij} can be evaluated by exact integration over the surface of the panel. Formulated in terms of the panel-based coordinates defined in (21) we must evaluate

$$A_{\tilde{x}} = \frac{1}{4\pi} \iint_{S_j} (\tilde{x} - \xi)\tilde{r}^{-3} d\xi d\eta \tag{27a}$$

$$A_{\tilde{y}} = \frac{1}{4\pi} \iint_{S_j} (\tilde{y} - \eta)\tilde{r}^{-3} d\xi d\eta \tag{27b}$$

$$A_{\tilde{y}} = \frac{1}{4\pi} \iint_{S_j} (\tilde{z} - \varsigma)\tilde{r}^{-3} d\xi d\eta \tag{27c}$$

with

$$\tilde{r} = \sqrt{(\tilde{x} - \xi)^2 + (\tilde{y} - \eta)^2 + \tilde{z}^2}$$

The double integral of an arbitrary function $f(\xi, \eta)$ over a quadralateral domain can be achieved by double integration over four semi-infinite strips each bounded by vertical lines passing through each pair of successive vertices of the quadralateral. The integral at each point will be the sum of such integrals.

For example, the integration can be performed over the strip $\xi_1 \leq \xi \leq \xi_2$, $|\eta| < \infty$. If a weight factor of $-\frac{1}{2}$ is applied as a premultiplier to the integral above the segment $\eta \geq g(\xi)$, and a factor of $+\frac{1}{2}$ to the integral below the segment $\eta \leq g(\xi)$, upon summation over all bounding segments, the contribution of the integration outside the panel will sum to zero, and similarly the net weight factor for the contribution to the integral from those differentials interior to the panel will sum to 1.

Details of this integration can be found in Chapter 4 of reference [6]. The $\tilde{y}$-component of A associated with the successive vertices ξ_m and ξ_ℓ, can be expressed

$$(A_{\tilde{y}})_{m\ell} = \frac{1}{4\pi}\int_{\xi_m}^{\xi_\ell} d\xi \quad \left[\frac{1}{2}\int_{-\infty}^{\eta} - \frac{1}{2}\int_{\eta}^{\infty}\right](\tilde{y}-\eta)\tilde{r}^{-3}d\eta$$

$$= \frac{1}{4\pi}\left(\frac{\xi_m - \xi_\ell}{d_{m\ell}}\right)\ln\left|\frac{\tilde{r}_m + \tilde{r}_\ell - d_{m\ell}}{\tilde{r}_m + \tilde{r}_\ell + d_{m\ell}}\right| \tag{31a}$$

with

$$d_{m\ell} = \sqrt{(\xi_m - \xi_\ell)^2 + (\eta_m - \eta_\ell)^2}$$

and

$$\tilde{r}_i = \sqrt{(\tilde{x} - \xi_i)^2 + (\tilde{y} - \eta_i)^2 + \tilde{z}^2} \quad i \in m, \ell.$$

Similarly,

$$(A_{\tilde{x}})_{m,\ell} = \frac{1}{4\pi}\left(\frac{\eta_\ell - \eta_m}{d_{m\ell}}\right)\ln\left|\frac{\tilde{r}_m + \tilde{r}_\ell - d_{m\ell}}{\tilde{r}_m + \tilde{r}_\ell + d_{m\ell}}\right| \tag{31b}$$

and

$$(A_{\tilde{z}})_{m\ell} = \frac{1}{4\pi}\left(\tan^{-1}\left[\{(\frac{\eta_\ell - \eta_m}{\xi_\ell - \xi_m})(\tilde{z}^2 + (\tilde{x} - \xi_m)^2) - (\tilde{y} - \eta_m)(\tilde{x} - \xi_m)\}/\tilde{z}\tilde{r}_m\right]\right.$$

$$\left. - \tan^{-1}\left[\{(\frac{\eta_\ell - \eta_m}{\xi_\ell - \xi_m})(\tilde{z}^2 + (\tilde{x} - \xi_\ell)^2) - (\tilde{y} - \eta_\ell)(\tilde{x} - \xi_\ell)\}/\tilde{z}\tilde{r}_\ell\right]\right) \tag{31c}$$

Summing over these terms for four successive pairs of vertices determines the following relationships

$$A_{\tilde{x}} = \sum_{\text{paired } m,\ell} (A_{\tilde{x}})_{m\ell}$$

$$A_{\tilde{y}} = \sum_{\text{paired } m,\ell} (A_{\tilde{y}})_{m\ell}$$

$$A_{\tilde{z}} = \sum_{\text{paired } m,\ell} (A_{\tilde{z}})_{m\ell}$$

Transforming to global coordinates we obtain

$$\begin{pmatrix} i_x & j_x & k_x \\ i_y & j_y & k_y \\ i_z & j_z & k_z \end{pmatrix} \begin{pmatrix} A_{\tilde{x}} \\ A_{\tilde{y}} \\ A_{\tilde{z}} \end{pmatrix} = \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} \tag{33}$$

and the coefficient C_{ij} follows from (25). Having established the matrix C_{ij}, equation (23) can be solved for the source distribution $\{\sigma_j\}$. The velocity field $\mathbf{u}^\phi$ can then be determined on ∂B from the following summation (which represents exact integration in (7)):

$$\begin{pmatrix} u_x^\phi \\ u_y^\phi \\ u_z^\phi \end{pmatrix} = \sum_{j=1}^{N} \sigma_j \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} \tag{32}$$

The corresponding potential field in $\mathrm{r_i} \in \mathrm{B^e}$ can be subsequently evaluated. The column vector $(A_x, A_y, A_z)^T$ must be recalculated in this case, remembering to observe a consistent panel-based coordinate system. The total flow in B^e satisfying impermeability at ∂B will hence be $\mathbf{u}^k + \mathbf{u}^\phi$.

6. Numerical Resolution of Flow

The numerical error committed in the potential flow solution strategy which has been described, is related to the relaxation of two assumptions implicit to the analytical expression of the problem. Implicit to the exact theory is the condition $\partial B \in C^1$ namely the boundedness of the curvature of the boundary. Obviously the imposition of gradient boundary conditions is compromised if this condition is not satisfied. Furthermore, in exact theory the doublet distribution $\sigma(\mathbf{r})$ is formally considered to be a continuous function of position.

The representation of ∂B by a discrete set of panels, with doublet distribution considered constant on each panel, introduces error into the solution. As the number of panels, N, tends to infinity, the definition of the integral as a Riemann sum will imply convergence to an exact expression of the potential problem. In practice, convergence may be achieved for relatively modest N so long as the distribution of body points adequately represents the geometry of the body.

This latter consideration has specific implications for bodies with continuous higher-order curvature. The articulated interfaces between flat panels introduce higher-order rippling in the solution for such cases. One can seek to reduce such effects by introducing higher numbers of flat panels, or by adopting panels with body-fitted curvature. For example, for flow over a sphere, the surface integrations in (27) can be transformed to spherical co-ordinates. Also methods have been developed which admit higher-order source density distribution over each panel. For example, one can consider a linear distribution [14]

$$\sigma = \sigma_o + \sigma_1 \xi + \sigma_2 \eta$$

or a quadratic distribution [6],

$$\sigma = \sigma_o + \sigma_1 \xi + \sigma_2 \eta + \frac{1}{2}\sigma_{11}\xi^2 + \sigma_{12}\xi\eta + \frac{1}{2}\sigma_{22}\eta^2$$

For bodies with sharp edges, some effort must be made to parameterise the edge in such a way that discontinuity of curvature is mitigated. This is to effectively represent a corner as a point of very high, but bounded, surface curvature. This will typically require the introduction of panels of smaller dimension at such points. The error in problems consisting of flat walls and sharp corners tends to concentrate at corners. An upper bound to error is

$$\frac{h^3}{12} \max |f''(x)|$$

where $f(x)$ represents the integrand of (10). The kernel in (10) consists of normal gradients and these typically have large higher-order derivatives in the neighbourhood of corners, and smaller higher-order derivatives towards the mid-point of the obstacle wall. Appendix A of reference [14] offers an alternative discussion of corners.

In the context of random vortex methods it is desirable to resolve the field within a regime $O(Re^{-\frac{1}{2}})$ of the wall. Considerable effort is made to construct there an explicit model of boundary layer dynamics – by representing the motion with overlapping vortex sheets – which should be smooth in the sense that the continuity condition is explicitly satisfied at some point during each time-step. In principle, the choice of body points in the random vortex method can be made to coincide with $\{h_j\}$ from the potential flow calculation, although this expedient will have significant implications.

In the random vortex method, the lengths of the panels control the resolution of the boundary layer model. Typically, one might expect a panel length $h_j \sim O(Re^{-\frac{1}{2}})$ will ensure a desired accuracy. Of course, if common body-points are to be chosen, different considerations apply in the choice of the parameter set $\{h_j\}$ for the two aspects of the calculation, namely applying conditions (1) and (2). For example, Puckett [15] suggests for a random vortex sheet model in two dimensions, the accuracy conditions $\Delta t\ U_{\max} \leq h$ and $\omega_{\max} \leq C_o h^2/\Delta t$, with $C_o = O(\frac{1}{L})$. These conditions would have to be reconciled with the requirements of a sufficiently accurate solution to (11). In fact a choice of much smaller panel length for the potential flow part of the calculation (than that for the random vortex part) could allow a more accurate resolution of the boundary-layer.

In the present discretisation, the potential flow field is well-behaved in the neighbourhood of the body- point. Although the Green's function is singular there, it is also integrable. In the immediate neighbourhood of a body-point, we can observe in two dimensions

$$\lim_{|\mathbf{r}_i|\to|\mathbf{r}_j|} A_x = 0$$

and

$$\lim_{|\mathbf{r}_i|\to|\mathbf{r}_j|} A_y = \frac{1}{2}.$$

The latter limit reflects the fact that the angle subtended by a panel at any point $P \notin \partial B$ approaches the limit $\pm\pi$ as P approaches the body point at the panel centre. Examination of equations (31) shows that the Nyström discretisation in three-dimensions is also well-behaved in the neighbourhood of the body-point.

On the other hand, both in two- and three-dimensions, the velocity field u^ϕ experiences logarithmic singularity at the edge of the panel. This can be seen by letting $(\tilde{x},\tilde{y}) \to (\pm\frac{h_j}{2},0)$ in equation (16a) or by letting $\tilde{r}_m+\tilde{r}_\ell \to d_{m\ell}$ in equation (31a). The field solution $\mathbf{u}^\phi$ is strictly valid at the body-point alone. Since, in the context of random vortex methods, a knowledge of the velocity field in the neighbourhood of the wall is required, the influence of these singularities extending into the boundary layer can be important. This spatial extent is reduced by reducing the panel dimensions, of course expending greater computational effort in the process.

For large N, the solutions to (11) and (23) might be achieved iteratively. C_{ij} is a dense matrix and for two-dimensional problems its condition number is typically low. Some discussion of the structure of C_{ij} (for strictly smooth convex bodies) is offered in chapter 5.2 of reference [6]. Appendix III of the present report reproduces the structure of C for a simple case of flow over a cube each face of which is partitioned into four panels (see Chapter 8). In three-dimensional problems the matrix may be more poorly conditioned. The ordering for three dimensional problems is not uniquely determined and may affect the behaviour of iterative methods. Groh [16,17] has proposed a regularised iterative algorithm for three-dimensional problems in which convergence is apparently assured. However this method also requires analytic knowledge of the scalar potential associated with the velocity field $\mathbf{u}^{\mathbf{k}}$ at each time-step. Numerical investigation of this algorithm should demonstate its convergence properties.

In the context of the random vortex method, these iterative solutions must be accomplished at each time-step, as the inhomogeneity vector b_i is updated. If the inhomogeneity vector is incremented slightly during a given time-step, then the solution at the previous time-step could serve as a close approximation for the initial guess solution to the iteration in the subsequent time-step. Although direct methods of solution may be impractical for large N, $N > 500$ say, they do have the advantage that the inverse matrix $\mathbf{C}^{-1}$ need be computed only once, perhaps preliminary to the entire time-dependent calculation. Using direct methods, the evaluation of $\mathbf{u}^{\phi}$ then becomes a matter of performing the matrix product $\mathbf{C}^{-1}$ $\mathbf{b}$ at each time-step, as $\mathbf{b}$ is updated. For large N this advantage is submerged by the computer effort in matrix multiplication.

The problem of controlling the computational effort for large distributions of panels can be addressed by adopting fast multipole strategies, in particular the application of multipole expansions to the Green's function and its derivatives (see Rokhlin [9]). In the context of random vortex methods, this strategy can be allied to a fast particle summation method to evaluate the flow induced by a collection of vortex elements (see Greengard and Rokhlin [18,19]).

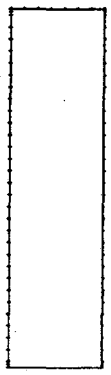
7. A two-dimensional application: flow over a prism of rectangular cross-section.

To illustrate the method in the context of a two-dimensional problem, we can consider the inviscid flow over a "prism" with rectangular cross-section defined in the $x - y$ plane. A uniform free-stream velocity in the positive x-direction, $\mathbf{u}^{\mathbf{k}} \equiv \mathbf{i}$ is assumed.

The main program listed in Appendix Ia represents the controlling section of a simple algorithm to evaluate the flow around this prism. A paramter **nn** is read from screen, and this determines the streamwise depth to vertical height aspect ratio of the obstacle, stored as variable **aspect**. The geometry of the obstacle is defined in subroutine **getgeo** (Appendix Ib).

The geometry is generated in such a way that the streamwise length of the obstacle can be adjusted by the insertion of additional panels (this is to ensure a constant h_j for different aspect ratios). An L-shaped stencil is defined in **getgeo** in data statements, namely in arrays **ax(),ay()**, to define the boundary ∂B. The boundary is generated by three-fold reflection of this stencil. The following parameters are then evaluated in **getgeo**: the coordinate positions of the ends of the panels (**xseg(),yseg()**), the panel lengths, (**h()**); the number of body-points, **nbox**; the coordinate positions of the body-points, (xbody(),ybody()); the direction cosines of the panels, (**s1(),s2()**); and the outward normal to the panels, (**an1(),an2()**). Direction cosines are calculated from the angle the panel makes with the postive x-axis. Having defined geometrical parameters $\{h_j\}$, $\{x_{cj}, y_{cj}\}$, $\{n_{xj}, n_{yj}\}$, $\{s_{xj}, s_{yj}\}$, control is returned to main program.

The following is an illustration of an obstacle generated in this way: it consists of 84 body-points and represents a body with aspect ratio 0.20 . The maximum value of h_j is 0.0333, and the minimum value (at the four corners) is 0.000416 . The input variable **nn** for this case is 2.



The program then enters subroutine **agenr** (Appendix Ic) which generates the coefficient matrix C_{ij}, and stores this in array **a(,)**. The routine enters a nested do loop to generate the terms describing the effect at control point i due to a unit source at body point j.

The inverse rotation matrix

$$\begin{pmatrix} s_x & -s_y \\ s_y & s_x \end{pmatrix}$$

is defined and stored in array **az(,)**; panel-based variables $\tilde{x}$ (**dxp**) and $\tilde{y}$ (**dyp**) are defined.

$A_{\tilde{x}}$ is calculated (**av(1)**) as well as $A_{\tilde{y}}$ (**av(2)**). Care must be taken in the case of $A_{\tilde{y}}$ to insure the intrinsic function **atan2** returns a value of angle in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$, bearing in mind that this function's default range of definition is $[-\pi, \pi]$.

The values of A_x and A_y are then calculated by inverse rotation into global coordinates.

Finally, the linear combination (13a) is calculated.

Upon visiting every pair of body-points, control is passed back to the main program.

The inverse matrix C^{-1}, stored in array **atom1(,)** is now calculated in subroutine **inver** (Appendix Ic) using standard Gauss elimination.

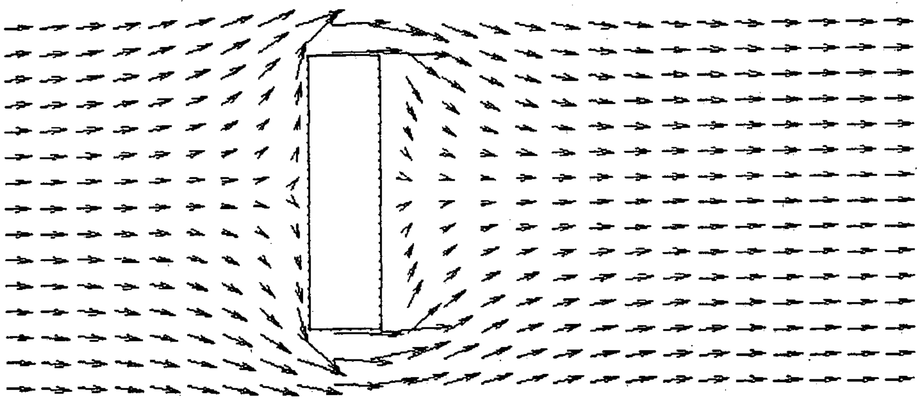
Upon return to the main program, the inhomogeneity column vector (12a) is calculated (and stored in **bcol()**). The matrix product $C^{-1}\ b$ is then performed (in **do 13** and **do 14**) to determine the source distribution $\{\sigma_j\}$; this distribution is stored in array **sigma()**.

Having evaluated the source distribution, we can evaluate (18), i.e. the velocity field $\mathbf{u}^\phi$ at each $\mathbf{r}_i \in \partial\mathbf{B}$. This integration is achieved in **do 15** of the main program. The velocity field $\left(u_x^\phi(\mathbf{r}_i), u_y^\phi(\mathbf{r}_i)\right)$ — stored in arrays **uq(i)** and **vq(i)** respectively — is calculated in subroutine **potvel** (Appendix Ie).

Subroutine **potvel** evaluates the potential flow (components stored in variables **u** and **v**) at any point $\mathbf{r}_i$ (coordinates stored in variables **x** and **y**). The integration (7) proceeds in loop **do 50** of **potvel**.

Upon return to loop **do 15** of the main program, $\mathbf{u}^\phi$ is added to $\mathbf{u}^{\mathrm{k}}$ to determine the flow at ∂B. In a random vortex computation, this velocity field is required for the vorticity creation algorithm to establish condition (2).

Finally, the velocity field over a regularly distributed array of exterior points $\mathbf{r}_i \in \mathrm{B}^{\mathrm{e}}$ is evaluated by a call to subroutine **velext** (Appendix If). This subroutine establishes the array of field points (**xfld,yfld**) and then calls **potvel** to determine u^ϕ at each such point. These exterior field values are stored in arrays **ufld,vfld**. The following diagram illustrates the vector velocity field satisfying condition (1) achieved in this way.



8. A three-dimensional example: flow over a cuboid.

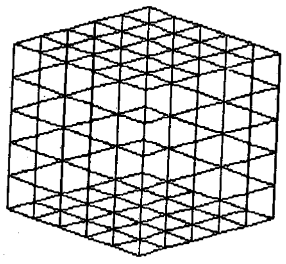
To illustrate potential flow over a three-dimensional bluff obstacle, we can consider the case of a cuboid. In particular we consider a cube with each face partitioned into 25 square panels. This is obviously an extremely coarse discretisation, which allows direct inversion. The total number of panels is 150. The main controlling program for such a calculation is found in Appendix IIa, where the total number of panels is stored in the variable **iptot.**

Subroutine **geom3** (Appendix IIb) establishes the coordinate locations of the vertices of the **iptot** panels, e.g. (ξ_m, η_m) in equations (31), and these are stored in arrays **vertx(ip,i)** , **verty(ip,i)**, **vertz(ip,i)**, with the integer $ip \in [1, iptot]$ refers to the panel label number, and integer $i \in [1,4]$ refers to the vertex label of the panel. The vertices are established in a clockwise sequence when the panel is viewed from outside the body. The partition of the surface is achieved through definition of a grid defined by arrays **ax()**, **ay()**, **az()**, with dimensions **nx**, **ny**, **nz** respectively. The six faces of the cuboid are visited in the following (arbitrary) order: $y = 0; z = 0; y = ay(ny); x = ax(nx); z = az(nz)$.

When the vertices of the panels are defined, their centroids **(xcen, ycen, zcen)** are determined, and the system of unit vectors $(\mathbf{i},\mathbf{j},\mathbf{k})$ — which are stored in arrays **ai()**, **aj()**, **ak()** — and the unit outward normal, stored in **an()** .

Finally the 3×3 rotation matrix **rot(ip,i,j)** is created for each panel, as well as its inverse **rotb(ip,i,j)** whereupon control passes back to the main program.

The following illustrates the geometry of the cuboid.

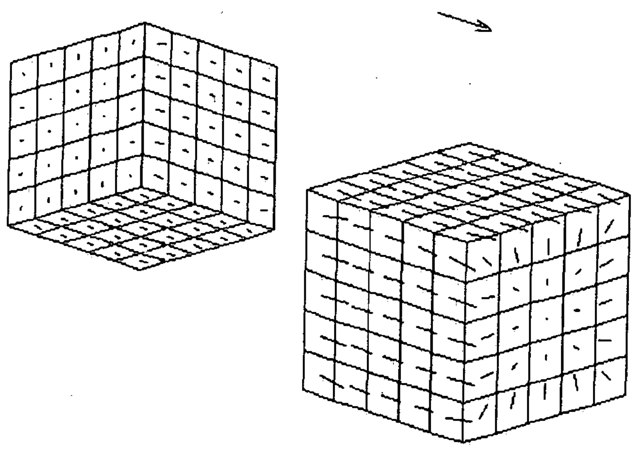


The loops **do 2** and **do 3** in the main program generate the kernel C_{ij}, which is stored in array **a(,)** . The vector displacements between control and body points are transformed into local panel-based coordinates (**do 6**) as are the coordinate locations of the four vertices for each panel (**do 9, do 10, do 11**). The vector $(A_{\bar{x}}, A_{\bar{y}}, A_{\bar{z}})$ defined in equations (31) are evaluated for each successive pair of line-segments bounding the ith panel, by four calls to subroutine **linek** (Appendix IIc). These are then added together for each component (**do 15** in main program). By inverse rotation the vector (A_x, A_y, A_z) is determined (**do 16, do 17**) and finally **a(,)** is determined.

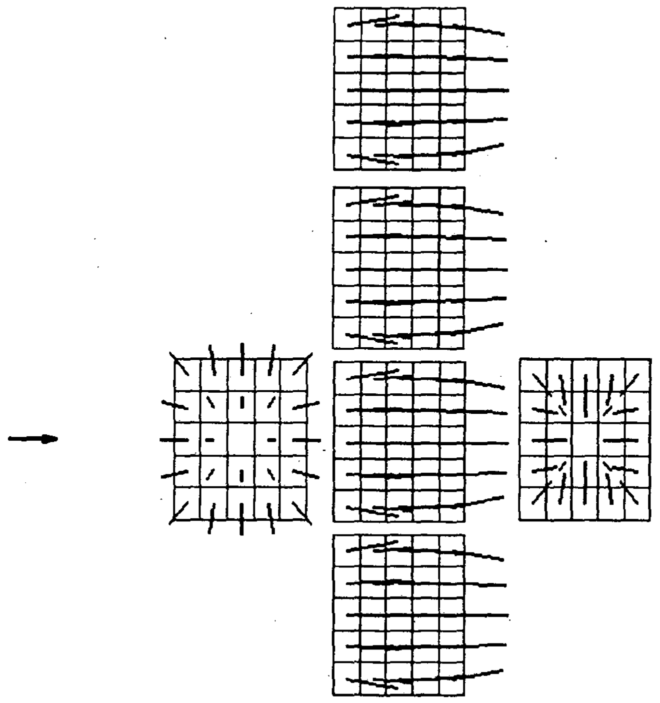
The inverse of C_{ij} (stored in **atom1(,)**) is determined directly by Gaussian elimination in a call to subroutine **inver** (Appendix Id), and the accuracy of this inverse is tested (**do 60** of main program).

The inverse of C_{ij} is then made to operate on the inhomogeneity vector $-\mathbf{u}_k \cdot \mathbf{n}$ (stored in vector array **bcol()**) to determine the source distribution σ (stored in array **alpha**). This is accomplished in subroutine **getalf** (Appendix IId).

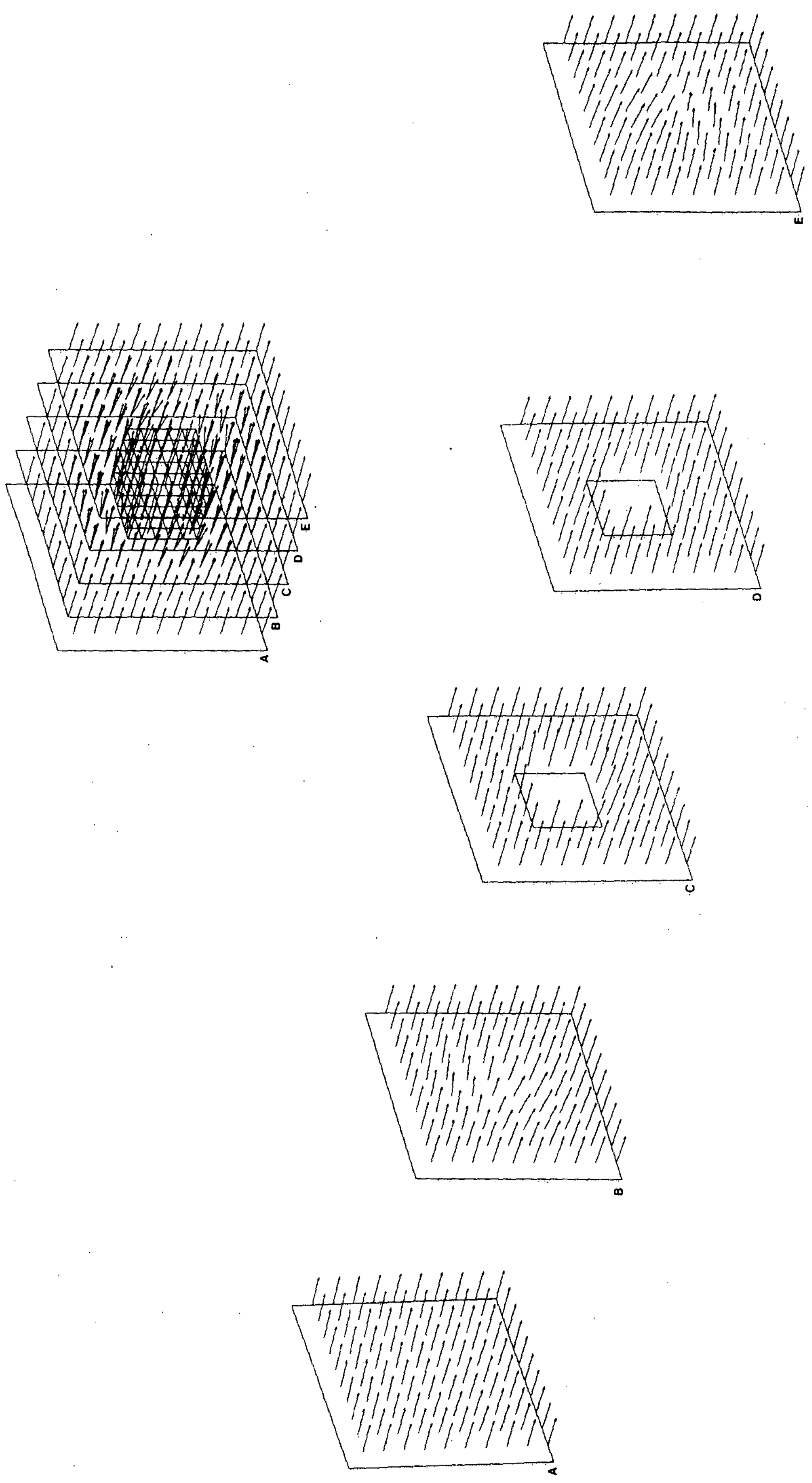
Through a call to subroutine **potvel** (Appendix IIe), the perturbation velocity field at each body-point is evaluated, and this is added to the free-stream velocity to determine the total field — stored in **vx(), vy(), vz()**. The parallel and normal components to the boundary are also calculated. The following diagram illustates the distribution of velocity over the six faces of the cuboid for this coarsely paramaterised problem.



The flow on the surface of the obstacle is more clearly seen in a fold-out of the cube:



Finally, the velocity field over a regularly distributed array of exterior points is evaluated by a call to **velav** (Appendix IIf) which itself calls subroutine **potmeas** (Appendix IIg). The following diagram illustrates this external field.



A
B
C
D
E
A
B
C
D
E

9. References

1. A. J. Chorin, Numerical study of slightly viscous flow, J. Fluid Mech., **57** , 785 (1973).

2. A. Y. Cheer, Numerical study of incompressible slightly viscous flow past blunt bodies and airfoils, SIAM J. Sci. Stat. Comput., **4** , 685 (1983).

3. A. Y. Cheer, Unsteady separated wake behind an impulsively started cylinder in slightly viscous fluid, J. Fluid Mech., **201** , 485 (1989).

4. A. Ghoniem and Y. Cagnon, Vortex simulation of laminar recirculating flow, J. Comput. Phys., **68** , 346 (1987).

5. D. M. Summers, T. Hanson and C. B. Wilson, A random vortex simulation of wind-flow over a building, Int. J. for Num. methods Fluids, **5** , 849 (1985).

6. J. L. Hess and A. M. O. Smith, "Calculation of potential flow about arbitary bodies", in *Prog. in Aeronautical Sciences* , **8** , 1 (1967).

7. M. A. Jaswon and G. T. Symm, *Integral Equation Methods in Potential Theory and Elastostatics* , Academic Press, London 1977.

8. T. K. Bose, *Computational Fluid Dynamics* , Chapter 5, Wiley, N.Y., 1988.

9. V. Rokhlin, Rapid solution of integral equations of classical potential theory, J. Comput. Phys., **60** , 187 (1985).

10. O. D. Kellogg, *Foundations of Potential Theory* , Dover, N. Y., 1929.

11. L. Delves and J. Walsh, *Numerical Solution of Integral Equations* , Oxford University Press, Oxford, 1974.

12. M. Goldberg (ed.), *Solution Methods for Integral Equations* , Plenum, N.Y., 1979.

13. K. Atkinson, *A Survey of Numerical Methods for the Solution of Fredholm Integral equations of the Second Kind* , SIAM, Philadelphia, 1976.

14. D. Lindholm, Three dimensional magnetostatic fields from point- matched integral equations with linearly varying scalar sources, IEEE Trans. Mag. (1984).

15. G. Puckett, A study of the vortex sheet method and its rate of convergence, SIAM J. Sci. Stat. Comput., **10** , 298 (1989).

16. G. Groh, "On panel methods for potential aerodynamics and for random vortex methods", LBL Report 16537 Berkeley, California, 1983.

17. G. Groh, "An iterative method for solving a class of integral equations in potential aerodynamics", LBL Report 18345, Berkeley California, 1984.

18. L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations", Technical Report 459, Yale Computer Science Department, Yale Connecticut, 1986.

19. L. Greengard and V. Rokhlin, "The rapid evaluation of potential fields in three dimensions", in *Vortex Methods* , C. Anderson and C. Greengard (eds.), Lecture Notes in Mathematics, Springer Verlag, N.Y., 1988.

Appendix I

```
c       ----------------------------------------------
c
c       A Program to compute the potential flow over a prism with
c       rectangular cross-section.
c
c       ----------------------------------------------
c
        common /invis /sigma(120),bcol(120),uq(120),vq(120)
        common /arays /a(120,120),aa(120,240),atom1(120,120)
        common /geom /xbody(120),ybody(120),an1(120),an2(120),s1(120),
       & s2(120),h(120),nbox,nbody
        common /field /ufld(6000),vfld(6000),xfld(6000),yfld(6000)
        common /walls /x0,y0,xb,yb
        real h
        real*8 aa
c
        data pi /3.14159265 /
c       ----------------------------------------------
c       The incident flow is in the positive x-direction and has
c       unit magnitude.
c       ----------------------------------------------
c
        v=0.
        u=1.
        write(6,9020)
 9020  format('enter below the input parameter nn')
        read(6,9010) nn
 9010  format(i5)
        aspect=0.2+2.*float(nn)*0.03333333
        twopi=2.*pi
        div2pi=1.0 /twopi
c       ----------------------------------------------
c       The impermeable body is defined by points (xseg,yseg) which are
c       generated by a call to getgeo which also computes geometrical
c       parameters. nn controls the aspect ratio of the body and is an
c       integer read from screen < 6 for current dimensioning
c       ----------------------------------------------
        call getgeo(nn)
c       ----------------------------------------------
c       A call to agenr generates the kernel to the boundary integral
c       equation.
c       ----------------------------------------------
        call agenr
c       ----------------------------------------------
c       The linear system is solved directly by a call to
c       subroutine inver which evaluates the matrix inverse
c       atoml of the descretised kernel.
c       ----------------------------------------------
        do 10 i=1,nbox
        do 10 j=1,nbox
        aa(i,j)=a(i,j)
  10   continue
        call inver(aa,det,nbox,nbox,nb2)
        call testinv
        do 11 i=1,nbox
        do 11 j=1,nbox
        atom1(i,j)=aa(i,j)
  11   continue
        do 12 i=1,nbox
        bcol(i)=-(u*an1(i)+v*an2(i))
  12   continue
```

```
c     ----------------------------------------------------
c     ...this then operates on the inhomogeneity vector bcol
c     to determine the source distribution sigma.
c     ----------------------------------------------------
      do 13 i=1,nbox
      dummy=0.
      do 14 j=1,nbox
      dummy=dummy+atom1(i,j)*bcol(j)
 14   continue
      sigma(i)=dummy
 13   continue
c     ----------------------------------------------------
c     The velocity field at the surface of the obstacle
c     is evaluated by a call to potvel at each body point.
c     ----------------------------------------------------
      do 15 i=1,nbox
      x=xbody(i)
      y=ybody(i)
      call potvel(0,x,y,uq(i),vq(i))
      uq(i)=u+uq(i)
      vq(i)=v+vq(i)
 15   continue
c     ----------------------------------------------------
c     ...and the velocity field in the interior of the fluid
c     is evaluated by a call to velext.
c     ----------------------------------------------------
      call velext
      end
```

```
c       ------------------------------------------------------------
c       Starting from an L-shaped stencil whose
c       coordinates are (ax,ay) a rectangular prism (with bevelled corners)
c       is generated.  This is done by various reflections of the stencil,
c       together with the insertion of NN added panel elements in the stream
c       direction;  the number of body points on the obstacle is NBOX.
c       Coordinates of a sheet layer and image sheet layer, of thickness
c       DELTA are generated.  The panel lengths (H), midpoints (XBODY,YBODY),
c       direction cosines (S1,S2) and normal directions (AN1,AN2) are
c       calculated.
c       ------------------------------------------------------------
        subroutine getgeo(nn)                                          getgeo
        real h
        dimension ax(19),ay(19),asx(120),asy(120)
        common /geom /xbody(120),ybody(120),an1(120),an2(120),s1(120),
       1 s2(120),h(120),nbox,nbody
        common /segs /xseg(120),yseg(120)
        data n /4 /,x0 /1. /
        data pi /3.14159265 /
        data ay /.48333333,.45,.416666,.38333,.35,.316666,.28333,.25,
       1 .216666,.183333,.150000,.116666,.083333,.05,.016666,
       1 .0083333,.0004166,0.,0. /
        data ax /1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
       1 1.,1.00041666,1.03375 /
        aah=0.0333333
        nbss=19
c       ----------------------------------------
c       process of generating body
c       ----------------------------------------
        do 10 i=1,nbss
        yseg(i)=ay(i)
        xseg(i)=ax(i)
  10    continue
        do 11 i=1,nn
        yseg(nbss+i)=yseg(nbss)
        xseg(nbss+i)=xseg(nbss)+i*aah
  11    continue
        nt=nbss+nn
        a0=xseg(nt)+0.5*aah
        do 12 i=1,nt
        yseg(nt+i)=yseg(nt-i+1)
        xseg(nt+i)=a0+a0-xseg(nt-i+1)
  12    continue
        nt2=2*nt
        a1=yseg(nt2)+0.5*aah
        do 13 i=1,nt2
        yseg(nt2+i)=a1+a1-yseg(nt2-i+1)
        xseg(nt2+i)=xseg(nt2-i+1)
  13    continue
        nt4=2*nt2
        nbox=nt4
        x0=xseg(1)
        xb=xseg(nt2)
        y0=yseg(nt)
        yb=yseg(nt2+nt)
c       ------------------------------------
c       invert order of points
c       ------------------------------------
        do 500 i=1,nbox
        asx(i)=xseg(i)
        asy(i)=yseg(i)
 500    continue
        do 501 i=1,nbox
        xseg(i)=asx(nbox+1-i)
       yseg(i)=asy(nbox+1-i)
 501  continue
c     ----------------------------------------------
c     determine midpoints and body-point parameters
c     ----------------------------------------------
       nboxm1=nbox-1
       do 100 i=1,nboxm1
       ah1=xseg(i+1)-xseg(i)
       ah2=yseg(i+1)-yseg(i)
       xbody(i)=xseg(i)+0.5*ah1
       ybody(i)=yseg(i)+0.5*ah2
       h(i)=sqrt(ah1*ah1+ah2*ah2)
       phi=atan2(ah2,ah1)
       s1(i)=cos(phi)
       s2(i)=sin(phi)
       an1(i)=cos(phi+0.5*pi)
       an2(i)=sin(phi+0.5*pi)
 100  continue
       ah1=xseg(1)-xseg(nbox)
       ah2=yseg(1)-yseg(nbox)
       xbody(nbox)=xseg(nbox)+0.5*ah1
       ybody(nbox)=yseg(nbox)+0.5*ah2
       h(nbox)=sqrt(ah1*ah1+ah2*ah2)
       phi=atan2(ah2,ah1)
       s1(nbox)=cos(phi)
       s2(nbox)=sin(phi)
       an1(nbox)=cos(phi+0.5*pi)
       an2(nbox)=sin(phi+0.5*pi)
       return
       end
```

agenr

```
c       -------------------------------------------
c       Subroutine to generate coefficient matrix associated
c       with kernel to boundary integral equation.
c       -------------------------------------------
        subroutine agenr
        real hhlf,h
        common /geom /xbody(120),ybody(120),an1(120),an2(120),s1(120),
      1 s2(120),h(120),nbox,nbody
        dimension az(2,2),av(2),bv(2)
        data pi /3.14159265 /
        div2pi=1. /(2.*pi)
c       -------------------------------------------
c       loop to generate terms relating to the i th. control point
c       -------------------------------------------
        do 2 i=1,nbox
        x=xbody(i)
        y=ybody(i)
c       -------------------------------------------
c       loop to generate terms describing effect at a control point i of
c       unit source at body  point j
c       -------------------------------------------
        do 3 j=1,nbox
        if(i .ne. j) go to 4
c       -------------------------------------------
c       ...self potential term
c       -------------------------------------------
        a(i,i)=0.5
        go to 3
  4     dx=x-xbody(j)
        dy=y-ybody(j)
c       -------------------------------------------
c       ...establish rotation matrix az
c       -------------------------------------------
        az(1,1)=s1(j)
        az(1,2)=-s2(j)
        az(2,1)=s2(j)
        az(2,2)=s1(j)
        dxp=az(1,1)*dx-az(1,2)*dy
        dyp=-az(2,1)*dx+az(2,2)*dy
        hhlf=0.5*h(j)
        av(1)=0.5*div2pi*alog(((dxp+hhlf)**2+dyp**2) /((dxp-hhlf)**2
      & +dyp**2))
        at1=dxp+hhlf
        at2=dyp
        at3=dxp-hhlf
        dp1=atan2(at1,at2)
        dp2=atan2(at3,at2)
        dpp=dp1-dp2
        qk=0.
        if(dpp.gt.pi) qk=1.
        if(dpp.lt.-pi) qk=-1.
        av(2)=div2pi*(dp1-dp2-qk*2.*pi)
        do 40 mm=1,2
        add=0.
        do 50 nn=1,2
        add=az(mm,nn)*av(nn)+add
 50     continue
        bv(mm)=add
 40     continue
        a(i,j)=-s2(i)*bv(1)+s1(i)*bv(2)
  3     continue
  2     continue
        return
        end
```

inver

```
c       -----------------------------------------------
c       Standard Gaussian elimination is used  to invert
c       the nbox x nbox matrix generated in subroutine AGENR.
c       -----------------------------------------------
        subroutine inver(aa,d,n,nx,mx)
        real*8 aa(120,240)
        n1=n-1
        n2=2*n
c
        do 2 i=1,n
        do 1 j=1,n
        j1=j+n
 1      aa(i,j1)=0.
        j1=i+n
 2      aa(i,j1)=1.
c
        do 10 k=1,n1
        c=aa(k,k)
        if(abs(c)-0.001) 3,3,5
 3      write(1,4) k
 4      format(' **** singularity in row ',i5)
        d=0.
        go to 300
 5      k1=k+1
        do 6 j=k1,n2
 6      aa(k,j)=aa(k,j) /c
        do 10 i=k1,n
        c=aa(i,k)
        do 10 j=k1,n2
 10     aa(i,j)=aa(i,j)-c*aa(k,j)
        np1=n+1
        if(abs(aa(n,n))-0.000001) 30,30,19
 19     do 20 j=np1,n2
 20     aa(n,j)=aa(n,j) /aa(n,n)
        go to 33
 30     write(1,31)
 31     format('***** singular matrix *****')
        go to 300
c
 33     do 200 l=1,n1
        k=n-l
        k1=k+1
        do 200 i=np1,n2
        do 200 j=k1,n
 200    aa(k,i)=aa(k,i)-aa(k,j)*aa(j,i)
c
        do 250 i=1,n
        do 250 j=1,n
        j1=j+n
 250    aa(i,j)=aa(i,j1)
        d=1.
        do 220 i=1,n
 220    d=d*aa(i,i)
 300    return
        end
```

```
c       ------------------------------------------
c       Evaluation of the potential field (u,v) at a point
c       (x,y) associated with source distribution sigma.
c       ------------------------------------------
        subroutine potvel(x,y,u,v)
        real hhlf,h
        common /invis /sigma(120),bcol(120),uq(120),vq(120)
        common /geom /xbody(120),ybody(120),an1(120),an2(120),s1(120),
       1 s2(120),h(120),nbox,nbody
        data pi /3.14159265 /
        div2pi=1. /(2.*pi)
        dum1=0.
        dum2=0.
        do 50 i=1,nbox
        hhlf=0.5*h(i)
        dx=x-xbody(i)
        dy=y-ybody(i)
        dxp=s1(i)*dx+s2(i)*dy
        dyp=-s2(i)*dx+s1(i)*dy
        if(abs(dyp).le.1.0e-05) dyp=0.
        at1=dxp+hhlf
        at2=dxp-hhlf
        at3=dyp
        dummy1=0.5*sigma(i)*div2pi*alog((at1**2+at3**2) /(at2**2+at3**2))
        dp1=atan2(at1,at3)
        dp2=atan2(at2,at3)
        dpp=dp1-dp2
        qk=0.
        if(dpp.gt.pi) qk=1.
        if(dpp.lt.-pi) qk=-1.
        dummy2=sigma(i)*div2pi*(dp1-dp2-qk*2.*pi)
        dum1=dum1+s1(i)*dummy1-s2(i)*dummy2
        dum2=dum2+s2(i)*dummy1+s1(i)*dummy2
 50     continue
        u=dum1
        v=dum2
        return
        end
```

velext

```
c       ————————————————————————————————
c       Evaluation of flow at regular spacing around
c       obstacle.
c       ————————————————————————————————
        subroutine velext
        common /geom / xbody(120),ybody(120),an1(120),an2(120),
      & s1(120),s2(120),,h(120),nbox,nbody
        common /field /ufld(6000),vfld(6000),xfld(6000),yfld(6000)
        common /walls /x0,y0,xb,yb
        dimension x(40),y(20)
        data xrmin /0. /,xrmax /3. /,yrmin /-.2 /,yrmax /1.2 /,nx /25 /,ny /15 /
        nprobe=nx*ny
        x(1)=xrmin
        y(1)= yrmin
        do 10 i=2,nx
        x(i)=x(i-1)+(xrmax-xrmin) /float(nx)
 10      continue
        do 11 i=2,ny
        y(i)=y(i-1)+(yrmax-yrmin) /float(ny)
 11      continue
        do 20 i=1,nx
        do 20 j=1,ny
        np=ny*(i-1)+j
c       ————————————————————————————————
c       jump over interior of obstacle
c       ————————————————————————————————
        if(x(i).lt.x0) go to 4
        if(x(i).gt.xb) go to 4
        if(y(j).gt.yb) go to 4
        if(y(j).lt.y0) go to 4
 9      u1=.0
        v1=0.
        go to 30
 4      call potvel(x(i),y(j),u1,v1)
 30     vfld(np)=v1
        ufld(np)=u1+1.
        xfld(np)=x(i)
        yfld(np)=y(j)
 20     continue
        return
        end
```

Appendix II

```
c       3D Pot Flow over cuboid
c       ------------------------
c
        real*8 aa
        common /geom /ax(6),ay(6),az(6),vertx(150,4),
       & verty(150,4),vertz(150,4),xcen(150),ycen(150),zcen(150),
       & ai(150,3),aj(150,3),ak(150,3),an(150,3),dd(3),v(3),
       & rot(150,3,3),rotb(150,3,3)
        common /kprime /dp(3),vp(4,3)
        common /stream /uinfin(3)
        common /vels /avx(4),avy(4),avz(4),avp(3),av(3)
        common /kernal /a(150,150)
        common /integs /nx,ny,nz,iptot
        common /dims /boxdimx,boxdimy,boxdimz
        common /inv /aa(150,300),atom1(150,150),alpha(150)
        common /vs /vax(150,150),vay(150,150),vaz(150,150),vx(150),vy(150),
       & vz(150)
        common /parm /div4pi,pi
        data boxdimx /1. /,boxdimy /1.0 /,boxdimz /1. /,nx /6 /,ny /6 /,nz /6 /
        data pi /3.14159265 /
        data uinfin /1.,0.,0. /
        open(9,file='box3d.dat')
        open(10,file='field3')
        open(8,file='boxarray')
        div4pi=1. /(4.*pi)
        iptot=(nx-1)*(ny-1)*(nz-1)
c       ------------------------------------
c       call geometry generator
c       ------------------------------------
        call geom3d
c       ------------------------------------
c       write geometrical data to file
c       ------------------------------------
c
        write(9,102) iptot
        do 100 i=1,iptot
        write(9,101) (vertx(i,k),k=1,4)
        write(9,101) (verty(i,k),k=1,4)
        write(9,101) (vertz(i,k),k=1,4)
        write(9,103) xcen(i)
        write(9,103) ycen(i)
        write(9,103) zcen(i)
        write(9,104) (an(i,k),k=1,3)
 100    continue
 101    format(4e11.4)
 102    format(i5)
 103    format(e11.4)
 104    format(3e11.4)
c
c       ----------------------------------------
c       enter nested do loop to generate matrix
c       a(i,j).  i = j is self potential term.
c       the distance between centroid (xcen,ycen,
c       zcen) and control point is stored in
c       array dd().  This is then expressed
c       in panel-based coordinates by rotation.
c       ----------------------------------------
c
        do 2 i=1,iptot
        x=xcen(i)
        y=ycen(i)
        z=zcen(i)
        do 3 j=1,iptot
        if(i.ne.j) go to 4
```

```
      a(i,j)=0.5
      go to 3
 4    dd(1)=x-xcen(j)
      dd(2)=y-ycen(j)
      dd(3)=z-zcen(j)
      do 5 k=1,3
      dum=0.
      do 6 l=1,3
 6    dum=dum+rot(j,k,l)*dd(l)
 5    dp(k)=dum
c     ------------------------------------------------
c     determine the coordinates of the vertices of the
c     jth panel in global coordinates, then by
c     rotation express this in terms of panel-based
c     coordinates.
c     ------------------------------------------------
      do 9 n=1,4
      v(1)=vertx(j,n)-xcen(j)
      v(2)=verty(j,n)-ycen(j)
      v(3)=vertz(j,n)-zcen(j)
      do 10 k=1,3
      dum=0.
      do 11 l=1,3
 11   dum=dum+rot(j,k,l)*v(l)
 10   vp(n,k)=dum
 9    continue
c     ------------------------------------------------
c     Perform integration over the quadrilateral domain
c     by four calls to subroutine linek
c     ------------------------------------------------

      avp(1)=0.
      avp(2)=0.
      avp(3)=0.
      call linek(1,2,avx(1),avy(1),avz(1))
      call linek(2,3,avx(2),avy(2),avz(2))
      call linek(3,4,avx(3),avy(3),avz(3))
      call linek(4,1,avx(4),avy(4),avz(4))
c
      do 15 k=1,4
      avp(1)=avp(1)+avx(k)
      avp(2)=avp(2)+avy(k)
      avp(3)=avp(3)+avz(k)
 15   continue
c
c     ------------------------------------------
c     rotate back into global coordinates, and divide
c     by normalising constant
c     ------------------------------------------
c
      do 16 k=1,3
      dum=0.
      do 17 l=1,3
 17   dum=dum+rotb(j,k,l)*avp(l)
      av(k)=dum*div4pi
 16   continue
c     ----------------------------------------
c     express component of av  normal to surface
c     ----------------------------------------
c
      a(i,j)=an(i,1)*av(1)+an(i,2)*av(2)+an(i,3)*av(3)
```

```
c       ----------------------------------
c       store invariant part of kernel for later use
c       in external computation.
c       ---------------------------------------------
        vax(i,j)=av(1)
        vay(i,j)=av(2)
        vaz(i,j)=av(3)
 3      continue
 2      continue

c       ---------------------------------------------
c       use gaussian elimination to determine direct
c       inverse to a(i,j), using subroutine inver
c       (see Appendix Id).  Perform accuracy check.
c       ---------------------------------------------

        do 40 i=1,iptot
        do 40 j=1,iptot
        aa(i,j)=a(i,j)
  40    continue
        call inver(aa,det,iptot,iptot,nb2)
        if(nfatal.gt.0) go to 999
        ncheck=0

        do 60 i=1,iptot
        do 60 j=1,iptot
        dummy=0.0
        do 61 k=1,iptot
        dummy=dummy+a(i,k)*aa(k,j)
 61     continue
        if(i.eq.j) go to 62

        if(abs(dummy).lt.1.0e-03) go to 60
        if(ncheck.eq.0) write(6,2005)
*
 2005  format( /,1x,'as an accuracy check matrix a has been',
       11x,'multiplied by its inverse', /,1x,'the answer should be a',
       21x,'unit matrix')
        ncheck=1
        write(6,2004) i,j,dummy
 2004  format(1x,'element (',i3,',',i3,') of matrix'
       1,1x,'product =',1pe11.4)
        go to 60

 62     if(abs(dummy-1.0).lt.1.0e-05) go to 60
        if(ncheck.eq.0) write(6,2005)
        ncheck=1
        write(6,2004) i,i,dummy
 60     continue
c
 999   continue
        do 65 i=1,iptot
        do 66 j=1,iptot
        atom1(i,j)=aa(i,j)
 66     continue
 65     continue
c       ---------------------------------------------
c       determine the source distribution alpha()
c       by operating the inverse on the inhomogeneity
c       vector, by a call to subroutine getalf.
c       ---------------------------------------------
        call getalf
```

```
c       ------------------------------------
c       determine the velocity field along the
c       boundary by a call to subroutine potvel
c      ---------------------------------------
c
        do 70 i=1,iptot
        write(6,2006) alpha(i),i
 70     continue
 2006 format('alpha, i = ', e11.4,i5)
        call potvel
c       --------------------------
c       add free-stream field
c       --------------------------
        do 80 i=1,iptot
        vx(i)=uinfin(1)+vx(i)
        vy(i)=vy(i)+uinfin(2)
        vz(i)=vz(i)+uinfin(3)
c       -----------------------------------------
c       calculate and write components of field
c       normal (anz) and parallel (apar) to boundary.
c       -----------------------------------------

        anz=vx(i)*an(i,1)+vy(i)*an(i,2)+vz(i)*an(i,3)
        ap1=vx(i)*ai(i,1)+vy(i)*ai(i,2)+vz(i)*ai(i,3)
        ap2=vx(i)*aj(i,1)+vy(i)*aj(i,2)+vz(i)*aj(i,3)
        apar=sqrt(ap1**2+ap2**2)
        write(10,85) vx(i),vy(i),vz(i),anz,apar,i
 80     continue
 85     format('vx,vy,vz=,vperp,vpar=,i=',1x,5e11.4,i5)
 940    format(7e11.4)
        write(9,940) (vx(i),i=1,iptot)
        write(9,940) (vy(i),i=1,iptot)
        write(9,940) (vz(i),i=1,iptot)
c       ------------------------------------
c       determine flow field in a region surrounding
c       the obstacle, by a call to subroutine velav,
c       which generates the three-dimensional dis-
c       tribution of measurement points (or probe locations).
c       ----------------------------------------------------
        call velav(nprobe)
        close(9)
        close(10)
        close(8)
        end
c
c
```

```
c       generator of 3d geometrics for cuboid
c
        subroutine geom3d
        common /geom /ax(6),ay(6),az(6),vertx(150,4),verty(150,4),
       & vertz(150,4),xcen(150),ycen(150),zcen(150),ai(150,3),
       & aj(150,3),ak(150,3),an(150,3),dd(3),v(3),rot(150,3,3),
       & rotb(150,3,3)
        common /integs / nx,ny,nz,iptot
        common /dims /boxdimx,boxdimy,boxdimz
c       ------------------------------------
c       based on length of each side, set up
c       partition increment.
c       ------------------------------------
c
        ahx=boxdimx /float(nx-1)
        ahy=boxdimy /float(ny-1)
        ahz=boxdimz /float(nz-1)
c       ------------------------------------
c       determine x,y,z coordinates of
c       panel vertices...
c       ------------------------------------
        ax(1)=0.
        ay(1)=0.
        az(1)=0.
        do 10 i=2,nx
        ax(i)=ax(i-1)+ahx
 10     continue
        do 11 i=2,ny
        ay(i)=ay(i-1)+ahy
 11     continue
        do 12 i=2,nz
        az(i)=az(i-1)+ahz
 12     continue
c       ------------------------------------
c       ... and store these in arrays
c       identified with each labelled panel.
c       ------------------------------------
c
c       Face 1: plane y = 0
c       --------------------
c
        ip=0
c
        do 13 i=1,nx-1
        do 13 j=1,nz-1
        ip=ip+1
        do 9 k=1,4
        verty(ip,k)=0.
 9      continue
        vertx(ip,1)=ax(i)
        vertx(ip,2)=ax(i)
        vertx(ip,3)=ax(i+1)
        vertx(ip,4)=ax(i+1)
        vertz(ip,1)=az(j)
        vertz(ip,2)=az(j+1)
        vertz(ip,3)=az(j+1)
        vertz(ip,4)=az(j)
 13     continue
c
c       Face 2: plane x = 0
c       --------------------
```

```
c
       do 14 i=1,ny-1
       do 14 j=1,nz-1
       ip=ip+1
       do 15 k=1,4
       vertx(ip,k)=0.
 15    continue
       verty(ip,1)=ay(i)
       verty(ip,2)=ay(i+1)
       verty(ip,3)=ay(i+1)
       verty(ip,4)=ay(i)
       vertz(ip,1)=az(j)
       vertz(ip,2)=az(j)
       vertz(ip,3)=az(j+1)
       vertz(ip,4)=az(j+1)
 14    continue
c
c      Face 3:  plane z = 0
c      ____________________
c
       do 16 i=1,nx-1
       do 16 j=1,ny-1
       ip=ip+1
       do 17 k=1,4
       vertz(ip,k)=0.
 17    continue
       vertx(ip,1)=ax(i)
       vertx(ip,2)=ax(i+1)
       vertx(ip,3)=ax(i+1)
       vertx(ip,4)=ax(i)
       verty(ip,1)=ay(j)
       verty(ip,2)=ay(j)
       verty(ip,3)=ay(j+1)
       verty(ip,4)=ay(j+1)
 16    continue
c
c      Face 4:   y = ay(NY)
c      ____________________
c
       do 18 i=1,nx-1
       do 18 j=1,nz-1
       ip=ip+1
       do 19 k=1,4
 19    verty(ip,k)=ay(ny)
       vertx(ip,1)=ax(i+1)
       vertx(ip,2)=ax(i+1)
       vertx(ip,3)=ax(i)
       vertx(ip,4)=ax(i)
       vertz(ip,1)=az(j)
       vertz(ip,2)=az(j+1)
       vertz(ip,3)=az(j+1)
       vertz(ip,4)=az(j)
 18    continue
c
c      Face 5:   x = ax(NX)
c      ____________________
c
       do 20 i=1,ny-1
       do 20 j=1,nz-1
       ip=ip+1
       do 21 k=1,4
       vertx(ip,k)=ax(nx)
 21    continue
       verty(ip,1)=ay(i+1)
```

```
      verty(ip,2)=ay(i)
      verty(ip,3)=ay(i)
      verty(ip,4)=ay(i+1)
      vertz(ip,1)=az(j)
      vertz(ip,2)=az(j)
      vertz(ip,3)=az(j+1)
      vertz(ip,4)=az(j+1)
  20  continue
c
c     Face 6: z=az(NZ)
c     ----------------
c
      do 22 i=1,nx-1
      do 22 j=1,ny-1
      ip=ip+1
      do 23 k=1,4
      vertz(ip,k)=az(nz)
  23  continue
      vertx(ip,1)=ax(i+1)
      vertx(ip,2)=ax(i)
      vertx(ip,3)=ax(i)
      vertx(ip,4)=ax(i+1)
      verty(ip,1)=ay(j)
      verty(ip,2)=ay(j)
      verty(ip,3)=ay(j+1)
      verty(ip,4)=ay(j+1)
  22  continue
      iptot=ip
c     ------------------------------
c     Establish centroid of each panel
c     ------------------------------
      do 29 ip=1,iptot
      zcen(ip)=0.
      xcen(ip)=0.
      ycen(ip)=0.
      do 30 k=1,4
      xcen(ip)=0.25*vertx(ip,k)+xcen(ip)
      ycen(ip)=0.25*verty(ip,k)+ycen(ip)
      zcen(ip)=0.25*vertz(ip,k)+zcen(ip)
  30  continue
c     ------------------------------
c     determine geometrical parameters,
c     and the unit vectors i,j,k
c     ------------------------------
      qx=vertx(ip,3)-vertx(ip,1)
      qy=verty(ip,3)-verty(ip,1)
      qz=vertz(ip,3)-vertz(ip,1)
      rx=vertx(ip,4)-vertx(ip,2)
      ry=verty(ip,4)-verty(ip,2)
      rz=vertz(ip,4)-vertz(ip,2)
      amq=sqrt(qx*qx+qy*qy+qz*qz)
      amr=sqrt(rx*rx+ry*ry+rz*rz)
      dimax=amq
      if(amr.gt.amq) dimax=amr
      akx=qy*rz-qz*ry
      aky=qz*rx-qx*rz
      akz=qx*ry-qy*rx
      tx=qz*aky-qy*akz
      ty=qx*akz-akx*qz
      tz=akx*qy-aky*qx
      amt=sqrt(tx*tx+ty*ty+tz*tz)
      amk=sqrt(akx*akx+aky*aky+akz*akz)
```

```
c
        ai(ip,1)=-qx /amq
        ai(ip,2)=-qy /amq
        ai(ip,3)=-qz /amq
c
        aj(ip,1)=-tx /amt
        aj(ip,2)=-ty /amt
        aj(ip,3)=-tz /amt
c
        ak(ip,1)=-akx /amk
        ak(ip,2)=-aky /amk
        ak(ip,3)=-akz /amk
c     ------------------------------
c     .... and outward normal
c     ------------------------------
        do 32 i=1,3
        an(ip,i)=-ak(ip,i)
 32   continue
c     ------------------------------
c     establish rotation (rot) and
c     inverse rotation (rotb) matrices
c     ------------------------------
        do 33 i=1,3
        rotb(ip,i,1)=ai(ip,i)
 33   continue
        do 34 i=1,3
        rotb(ip,i,2)=aj(ip,i)
 34   continue
        do 35 i=1,3
        rotb(ip,i,3)=ak(ip,i)
 35   continue
        do 36 i=1,3
        rot(ip,1,i)=ai(ip,i)
 36   continue
        do 37 i=1,3
        rot(ip,2,i)=aj(ip,i)
 37   continue
        do 38 i=1,3
        rot(ip,3,i)=ak(ip,i)
 38   continue
 29   continue
        return
        end
```

```
c       --------------------------------
c       subroutine to evaluate the
c       decomposed surface integral over
c       a quadralateral:  the integral
c       over a semi-infinite strip
c       bounded by vertices labelled
c       n and m.
c       --------------------------------
        subroutine linek(n,m,a1,a2,a3)
        common /kprime /dp(3),vp(4,3)
        common /parm /div4pi,pi
        dimension dr(2)
c
c
        dkl2=(vp(n,1)-vp(m,1))**2+(vp(n,2)-vp(m,2))**2
        dkl=sqrt(dkl2)
        if(dkl.le.1.0e-05) go to 14
        dr2=(dp(1)-vp(n,1))**2+(dp(2)-vp(n,2))**2+dp(3)**2
        dr(1)=sqrt(dr2)
        dr2=(dp(1)-vp(m,1))**2+(dp(2)-vp(m,2))**2+dp(3)**2
        dr(2)=sqrt(dr2)
        b=(dr(1)+dr(2)-dkl) /(dr(1)+dr(2)+dkl)
        bb=alog(abs(b))
        a1=(vp(m,2)-vp(n,2))*bb /dkl
        a2=(vp(n,1)-vp(m,1))*bb /dkl
        go to 15
 14     a1=0.
        a2=0.
c
 15     cc=abs(vp(m,1)-vp(n,1))
        if(cc.le.1.e-05) go to 10
        cc=abs(dp(3))
        if(cc.le.1.e-05) dp(3)=0.
        b1=(vp(m,2)-vp(n,2)) /(vp(m,1)-vp(n,1))
        b2=dp(3)**2+(dp(1)-vp(n,1))**2
        b3=dp(3)**2+(dp(1)-vp(m,1))**2
        b4=dp(3)*dr(1)
        b5=dp(3)*dr(2)
        at1=b1*b2-(dp(2)-vp(n,2))*(dp(1)-vp(n,1))
        at2=b1*b3-(dp(2)-vp(m,2))*(dp(1)-vp(m,1))
        a3=(atan2(at1,b4)-atan2(at2,b5))
        qk=0.
        if(a3.gt.pi) qk=1.
        if(a3.lt.-pi) qk=-1.
        a3=a3-qk*2.*pi
        return
 10     a3=0.
        return
        end
```

```
c       ------------------------------------
c       Subroutine to evaluate the linear
c       system for the source distribution
c       alpha().
c       ------------------------------------
        subroutine getalf                                          getalf
        real*8 aa
        common /inv /aa(150,300),atom1(150,150),alpha(150)
        common /stream /uinfin(3)
        common /integs /nx,ny,nz,iptot
        common /geom /ax(6),ay(6),az(6),vertx(150,4),
     &  verty(150,4),vertz(150,4),xcen(150),ycen(150),zcen(150),
     &  ai(150,3),aj(150,3),ak(150,3),an(150,3),dd(3),v(3),
     &  rot(150,3,3),rotb(150,3,3)
        dimension bcol(150)
        data pi /3.14159265 /
        do 10 i=1,iptot
        bcol(i)=-(uinfin(1)*an(i,1)+uinfin(2)*an(i,2)+uinfin(3)*an(i,3))
        bcol(i)=bcol(i)
 10     continue
        do 11 i=1,iptot
        dummy=0.
        do 12 j=1,iptot
        dummy=dummy+atom1(i,j)*bcol(j)
 12     continue
        alpha(i)=dummy
 11     continue
        return
        end
```

potvel

```
c       ------------------------------------------------
c       Subroutine to determine the velocity
c       field at the surface of the obstacle.
c       ------------------------------------------------
        subroutine potvel
        real*8 aa
        common /geom /ax(6),ay(6),az(6),vertx(150,4),verty(150,4),
      & vertz(150,4),xcen(150),ycen(150),zcen(150),
      & ai(150,3),aj(150,3),ak(150,3),an(150,3),dd(3),v(3),
      & rot(150,3,3),rotb(150,3,3)
        common /inv /aa(150,300),atom1(150,150),alpha(150)
        common /vs /vax(150,150),vay(150,150),vaz(150,150),vx(150),vy(150),
      & vz(150)
        common /integs /nx,ny,nz,iptot
        common /pis /div2pi
        do 50 i=1,iptot
        dum1=0.
        dum2=0.
        dum3=0.
        do 60 j=1,iptot
        dum1=dum1+vax(i,j)*alpha(j)
        dum2=dum2+vay(i,j)*alpha(j)
        dum3=dum3+vaz(i,j)*alpha(j)
  60    continue
        vx(i)=dum1
        vy(i)=dum2
        vz(i)=dum3
  50    continue
        return
        end

c
```

velav

```
c        ______________________________
c        Subroutine to establish a regular
c        distribution of probe points around
c        obstacle at which to measure potential
c        field, which is achieved in a call
c        to potmeas.
c        __________________________________________
c
         subroutine velav(nprobe)
         dimension x(25),y(15),z(15),xpr(6000),ypr(6000),zpr(6000),
       & uf(6000),vf(6000),wf(6000)
         data xrmin /-1. /,xrmax /3. /,yrmin /-1. /,yrmax /2. /,nfx /25 /,nfy /15 /,
       & nfz /15 /,zrmin /-1. /,zrmax /2. /
         nprobe=nfx*nfy*nfz
         x(1)=xrmin
         y(1)= yrmin
         z(1)=zrmin
         do 10 i=2,nfx
         x(i)=x(i-1)+(xrmax-xrmin) /float(nfx)
  10    continue
         do 11 i=2,nfy
         y(i)=y(i-1)+(yrmax-yrmin) /float(nfy)
  11    continue
         do 12 i=2,nfz
         z(i)=z(i-1)+(zrmax-zrmin) /float(nfy)
  12    continue
         do 20 i=1,nfx
         do 20 j=1,nfy
         do 20 k=1,nfz
         np=nfz*nfy*(i-1)+nfz*(j-1)+k
         if(x(i).gt.1.and.x(i).lt.0.) go to 4
         if(z(k).gt.1.and.z(k).lt.0.) go to 4
         if(y(j).gt.1..and.y(j).lt.0.) go to 4
         uf(np)=0.
         vf(np)=0.
         wf(np)=0.
         gc to 30
  4     call potmeas(x(i),y(j),z(k),uf(np),vf(np),wf(np))
  30    xpr(np)=x(i)
         ypr(np)=y(j)
         zpr(np)=z(k)
  20    continue
c        ______________________________
c        measured external field is written to file
c        ________________________________________________
         write(9,901) nprobe
         write(9,900) (xpr(np),np=1,nprobe)
         write(9,900) (ypr(np),np=1,nprobe)
         write(9,900) (zpr(np),np=1,nprobe)
         write(9,900) (uf(np),np=1,nprobe)
         write(9,900) (vf(np),np=1,nprobe)
         write(9,900) (wf(np),np=1,nprobe)
  900   format(7e11.4)
  901   format(i5)
         return
         end
```

potmeas

```
c      -----------------------------------
c      Subroutine to evaluate the potential
c      field at a probe position (xx,yy,zz).
c      This is done using the source distribution
c      alpha previously evaluated
c      -----------------------------------
       subroutine potmeas(xx,yy,zz,ufx,ufy,ufz)
       real*8 aa
       common /geom /ax(6),ay(6),az(6),vertx(150,4),
      & verty(150,4),vertz(150,4),xcen(150),ycen(150),zcen(150),
      & ai(150,3),aj(150,3),ak(150,3),an(150,3),dd(3),v(3),
      & rot(150,3,3),rotb(150,3,3)
       common /kprime /dp(3),vp(4,3)
       common /stream /uinfin(3)
       common /vels /avx(4),avy(4),avz(4),avp(3),av(3)
       common /inv /aa(150,300),atom1(150,150),alpha(150)
       common /integs /nx,ny,nz,iptot
       common /parm /div4pi,pi
c
       dum1=0.
       dum2=0.
       dum3=0.
       do 2 i=1,iptot
       dd(1)=xx-xcen(i)
       dd(2)=yy-ycen(i)
       dd(3)=zz-zcen(i)
       do 3 k=1,3
       dum=0.
       do 4 l=1,3
 4     dum=dum+rot(i,k,l)*dd(l)
 3     dp(k)=dum
c
       do 7 n=1,4
       v(1)=vertx(i,n)-xcen(i)
       v(2)=verty(i,n)-ycen(i)
       v(3)=vertz(i,n)-zcen(i)
       do 8 k=1,3
       dum=0.
       do 9 l=1,3
 9     dum=dum+rot(i,k,l)*v(l)
 8     vp(n,k)=dum
 7     continue
c
c
       avp(1)=0.
       avp(2)=0.
       avp(3)=0.
       call linek(1,2,avx(1),avy(1),avz(1))
       call linek(2,3,avx(2),avy(2),avz(2))
       call linek(3,4,avx(3),avy(3),avz(3))
       call linek(4,1,avx(4),avy(4),avz(4))
c
       do 10 k=1,4
       avp(1)=avp(1)+avx(k)
       avp(2)=avp(2)+avy(k)
       avp(3)=avp(3)+avz(k)
 10    continue
c
       do 11 k=1,3
       dum=0.
       do 12 l=1,3
 12    dum=dum+rotb(i,k,l)*avp(l)
 11    av(k)=dum*div4pi
```

```
c
      dum1=dum1+av(1)*alpha(i)
      dum2=dum2+av(2)*alpha(i)
      dum3=dum3+av(3)*alpha(i)
c
 2    continue
c
      ufx=uinfin(1)+dum1
      ufy=uinfin(2)+dum2
      ufz=uinfin(3)+dum3
c
      return
      end
```

Appendix III

0.5	0	0	0	0.0713	0.0212	0.0308	0.0187	0.0713	0.0308	0.0212	0.0187	0.0187	0.0139	0.0139	0.0107	0.00852	0.0056	0.012	0.00903	0.00852	0.012	0.0056	0.00903
0	0.5	0	0	0.0212	0.0713	0.0187	0.0308	0.00852	0.012	0.0056	0.00903	0.0139	0.0187	0.0107	0.0139	0.0056	0.00852	0.00903	0.012	0.0713	0.0308	0.0212	0.0187
0	0	0.5	0	0.00852	0.0056	0.012	0.00903	0.0212	0.0187	0.0713	0.0308	0.0139	0.0107	0.0187	0.0139	0.0713	0.0212	0.0308	0.0187	0.0056	0.00903	0.00852	0.012
0	0	0	0.5	0.0056	0.00852	0.00903	0.012	0.0056	0.00903	0.00852	0.012	0.0107	0.0139	0.0139	0.0187	0.0212	0.0713	0.0187	0.0308	0.0212	0.0187	0.0713	0.0308
0.0713	0.0212	0.0308	0.0187	0.5	0	0	0	0.0713	0.0212	0.0308	0.0187	0.00852	0.0056	0.012	0.00903	0.0187	0.0139	0.0139	0.0107	0.00852	0.0056	0.012	0.00903
0.0212	0.0713	0.0187	0.0308	0	0.5	0	0	0.00852	0.0056	0.012	0.00903	0.0056	0.00852	0.00903	0.012	0.0139	0.0187	0.0107	0.0139	0.0713	0.0212	0.0308	0.0187
0.00852	0.0056	0.012	0.00903	0	0	0.5	0	0.0212	0.0713	0.0187	0.0308	0.0713	0.0212	0.0308	0.0187	0.0139	0.0107	0.0187	0.0139	0.0056	0.00852	0.00903	0.012
0.0056	0.00852	0.00903	0.012	0	0	0	0.5	0.0056	0.00852	0.00903	0.012	0.0212	0.0713	0.0187	0.0308	0.0107	0.0139	0.0139	0.0187	0.0212	0.0713	0.0187	0.0308
0.0713	0.0308	0.0212	0.0187	0.0713	0.0308	0.0212	0.0187	0.5	0	0	0	0.00852	0.012	0.0056	0.00903	0.00852	0.012	0.0056	0.00903	0.0187	0.0139	0.0139	0.0107
0.00852	0.012	0.0056	0.00903	0.0212	0.0187	0.0713	0.0308	0	0.5	0	0	0.0713	0.0308	0.0212	0.0187	0.0056	0.00903	0.00852	0.012	0.0139	0.0187	0.0107	0.0139
0.0212	0.0187	0.0713	0.0308	0.00852	0.012	0.0056	0.00903	0	0	0.5	0	0.0056	0.00903	0.00852	0.012	0.0713	0.0308	0.0212	0.0187	0.0139	0.0107	0.0187	0.0139
0.0056	0.00903	0.00852	0.012	0.0056	0.00903	0.00852	0.012	0	0	0	0.5	0.0212	0.0187	0.0713	0.0308	0.0212	0.0187	0.0713	0.0308	0.0107	0.0139	0.0139	0.0187
0.0187	0.0139	0.0139	0.0107	0.0308	0.0187	0.0713	0.0212	0.0308	0.0713	0.0187	0.0212	0.5	0	0	0	0.012	0.00903	0.00852	0.0056	0.012	0.00852	0.00903	0.0056
0.0139	0.0187	0.0107	0.0139	0.0187	0.0308	0.0212	0.0713	0.012	0.00852	0.00903	0.0056	0	0.5	0	0	0.00903	0.012	0.0056	0.00852	0.0308	0.0713	0.0187	0.0212
0.0139	0.0107	0.0187	0.0139	0.012	0.00903	0.00852	0.0056	0.0187	0.0212	0.0308	0.0715	0	0	0.5	0	0.0308	0.0187	0.0713	0.0212	0.00903	0.0056	0.012	0.00852
0.0107	0.0139	0.0139	0.0187	0.00903	0.012	0.0056	0.00852	0.00903	0.0056	0.012	0.00852	0	0	0	0.5	0.0187	0.0308	0.0212	0.0713	0.0187	0.0212	0.0308	0.0713
0.0308	0.0187	0.0713	0.0212	0.0187	0.0139	0.0139	0.0107	0.0308	0.0187	0.0713	0.0212	0.012	0.00903	0.00852	0.0056	0.5	0	0	0	0.012	0.00903	0.00852	0.0056
0.0187	0.0308	0.0212	0.0713	0.0139	0.0187	0.0107	0.0139	0.012	0.00903	0.00852	0.0056	0.00903	0.012	0.0056	0.00852	0	0.5	0	0	0.0308	0.0187	0.0713	0.0212
0.012	0.00903	0.00852	0.0056	0.0139	0.0107	0.0187	0.0139	0.0187	0.0308	0.0212	0.0713	0.0308	0.0187	0.0713	0.0212	0	0	0.5	0	0.00903	0.012	0.0056	0.00852
0.00903	0.012	0.0056	0.00852	0.0107	0.0139	0.0139	0.0187	0.00903	0.012	0.0056	0.00852	0.0187	0.0308	0.0212	0.0713	0	0	0	0.5	0.0187	0.0308	0.0212	0.0713
0.0308	0.0713	0.0187	0.0212	0.0308	0.0713	0.0187	0.0212	0.0187	0.0139	0.0139	0.0107	0.012	0.00852	0.00903	0.0056	0.012	0.00852	0.00903	0.0056	0.5	0	0	0
0.012	0.00852	0.00903	0.0056	0.0187	0.0212	0.0308	0.0713	0.0139	0.0187	0.0107	0.0139	0.0308	0.0713	0.0187	0.0212	0.00903	0.0056	0.012	0.00852	0	0.5	0	0
0.0187	0.0212	0.0308	0.0713	0.012	0.00852	0.00903	0.0056	0.0139	0.0107	0.0187	0.0139	0.00903	0.0056	0.012	0.00852	0.0308	0.0713	0.0187	0.0212	0	0	0.5	0
0.00903	0.0056	0.012	0.00852	0.00903	0.0056	0.012	0.00852	0.0107	0.0139	0.0139	0.0187	0.0187	0.0212	0.0308	0.0713	0.0187	0.0212	0.0308	0.0713	0	0	0	0.5