SSSU 50
ISSN 0140 3818

UNIVERSITY OF SOUTHAMPTON

DEPARTMENT OF SHIP SCIENCE

FACULTY OF ENGINEERING AND APPLIED SCIENCE

LIFTING SURFACE METHOD FOR MODELLING SHIP RUDDERS AND PROPELLERS

by S.R. Turnock

Ship Science Report No. 50

April 1992

LIFTING SURFACE METHOD FOR MODELLING SHIP RUDDERS AND PROPELLERS

by

S.R.Turnock

Ship Science Report No 50

University of Southampton

March 1992

SUMMARY

The theoretical basis for a lifting surface model using Morino's formulation is given in this report. The various choices made previously for modelling rudder and propeller interactions are described and the reasons for using, in this investigation, a lifting surface perturbation potential method. An explicit trailing edge pressure Kutta-Joukowsky condition is used to ensure that there is no pressure loading at the trailing edge. A frozen wake or an adaptive wake model can be chosen for both the rudder and propeller simulations.

A flexible scheme for geometry definition was developed to allow flow over a wide variety of geometries and multiple body lifting-surface problems. This surface definition scheme uses parametric cubic splines which require a minimal amount of data to accurately define quadrilateral panels on a three-dimensional surface.

The proposed Interaction Velocity Field method separately models the lifting surfaces. In this case, a rudder and propeller. The flow interaction between them is accounted for by modifying their respective inflow velocity fields. Expressions were derived to allow the velocity at any point within the flow domain to be calculated using the solution to the perturbation potential method. This process is used to generate the respective inflow velocity field. It can also be used to produce flow visualisation information important for design purposes.

Verification of the lifting-surface method was carried out to compare the results obtained with previously published numerical and experimental results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Ai Area of panel i

A Angular rotation vector (ordered x, y and then z)

c Chord

C_D Non dimensional drag coefficient

C_f Coefficient of skin friction

C_l Local lift coefficient

C_L Non dimensional lift coefficient

Cp Pressure coefficient

Dij Body dipole panel j's influence at point i

F Body force vector

L Ratio of the distance from panel centroid to a point to that of the length of the largest panel diagonal

M Body moment vector

n unit surface normal vector

O Offset vector

P Pressure

P	Pivot vector
r	Radial distance between two points
Rn	Reynolds number
S	Scale vector
S_B	Bounding body surface
Sij	Body source panel j's influence at point i
S_W	Wake surface
s	semi-span
U_∞	free-stream speed
U	velocity vector
v	Disturbance velocity vector
Wik	wake dipole panel k's influence at point i
R,aθ	Cylindrical coordinates
t,s	parametric arc-lengths along three-dimensional space curves
x,y,z	Cartesian coordinates
ρ	Density
Δ t	Time step
ϕ	Disturbance potential
Φ	Total potential
σ	Source strength
μ	Dipole strength
ω	angular velocity vector

1 INTRODUCTION

Only a limited amount of theoretical work has been carried out with regard to the modelling of ship's rudder and propeller in Naval Architecture. However, the similar problem on aircraft, of wing mounted propellers and the resulting interaction between the wing and propeller has received considerable attention.

In a recent paper by Cho & Williams[1] the propeller-wing interaction was investigated using a frequency domain panel method. A linearised compressible potential theory was used to determine the unsteady aerodynamic coupling between the wing and propeller. An iterative approach was used whereby the propeller and wing are solved in isolation and the effect of the interaction is obtained using a fourier transform of their respective induced velocity fields. The lifting-surface method used gave good agreement with measured mean loads on a wing/propeller system. A principal conclusion was that useful performance predictions could be made where circumferentially averaged (isolated) propwash is imposed on the wing and where a circumferentially averaged (isolated) wingwash is imposed on the propeller to get the modified steady blade/rudder loading.

Previously Kleinstein [2] had looked at the effect of a propeller on the aerodynamic characteristics of a high aspect-ratio wing. Using a lifting-line approach with the propeller stream represented by an equivalent circular jet of uniform velocity, it was found that a better approximation to propeller-wing interaction was obtained than that by Koening [3]. Koening's classical theory represents the propeller as a constant velocity but of infinite height and therefore a uniform velocity field. A drawback to the work was the lack of experimental studies to verify the results.

Kroo [4] looked at the problem of minimising induced loss for the integrated system of propeller and wings. In a simplified model using a fourier series wing lift distribution the stagger theorem of Munk [5] was used. In its generalised form this states that "the net force in the stream-wise direction is independent of the stream-wise position of surfaces with a given circulation distribution". This implies that the wing optimum circulation distribution (minimum drag) can be computed with the propeller far upstream

and therefore removing the need to calculate the effect of the rudder on the propeller and that of the propeller's bound vorticity of the wing. This simple analysis showed that the wing's optimal lift distribution significantly differed from that of the elliptic load distribution of an isolated wing. The wing acts to restore the swirl energy otherwise lost from the propeller's wake. Again, no experimental comparison with theory was made.

An experimental investigation and computational analysis was carried out by Witkowski et al [6] at Purdue University. These tests were carried out with a configuration for a tractor mounted 12" diameter propeller upstream of a rectangular wing with geometric aspect ratio of (33/4). In the computer analysis two methods were used. A semi-empirical method involved the superposition of experimentally correlated propeller wake velocity onto a uniform flow across a wing modelled using a vortex lattice method. This gave a fast means of achieving results which predicted ΔC_D to between 10% and 30%. The other method used a vortex-lattice to simultaneously model the wing and propeller. In this, the propeller is assumed to have a wake on a rigid helix of constant diameter and the wing wake is assumed flat and in the stream-wise direction. The calculation was repeated for a range of propeller blade phase angles and the quasi-steady forces obtained by averaging over a complete revolution. This method gave a better comparison for ΔC_D of between 6-12%.

Although the work of Witkowski [6] and also that of Lorber [7] in looking at helicopter rotor-fuselage interaction have been moderately successful in modelling the complete interaction geometry, a fundamental problem remains in the specification of the propeller wake geometry. At best, only an educated guess is possible in stating the wake geometry, and vortex-lattice methods can be sensitive to the location of collocation points. As most experimental results are of time-averaged quantities it is difficult to justify the use of combined geometry models where a non-physical description of the wake structure has to be used to obtain results. The approach used by Cho and Williams [1] is the most promising with the separation of the problem into the modelling of a propeller with imposed velocity field and a rudder with imposed velocity field. This allows the considerable expertise of modelling isolated rudders and propellers to be fully used and the modelling of the interaction can be concentrated on developing adapted

velocity inflow fields.

No mention has been made of more sophisticated modelling of the flow using a Euler or Navier-Stokes formulation. This is due to the large computational requirements of such methods combined with the difficulties of modelling a propeller helical wake. At the present time the author has not come across any practical methods for isolated propellers and therefore the available methods for modelling the propeller are restricted to some form of lifting surface, lifting-line, or blade-element momentum approach. The balancing of the need for physical realism with the complexity of numerical formulation and hence computational requirement is dependent on the final use of the computed solution.

In modelling the interaction of a ship rudder and propeller to develop a parametric relationship for manoeuvring studies and design purposes, only quasi-steady total forces and surface pressure information is required for both rudder and propeller. This information can be obtained from an inviscid, incompressible and irrotational potential model such as the panel method first used by Hess and Smith [8].

The separation of the modelling of the rudder and propeller allows the most appropriate method for modelling each to be chosen. Molland [9] obtained a good comparison for a lifting-line model of free-stream semi-balanced skeg rudders. However, a propeller inflow will result in considerable span-wise cross-flow which cannot be adequately modelled using a lifting-line except thorough considerable empiricism. It is also not possible to determine the local chordwise position of centre of pressure for a given span position which is required for an understanding of the rudder stock torque. Some form of lifting-surface model of the rudder should provide adequate information for the manoeuvring characteristics.

A lifting-surface approach to modelling a propeller has been developed at M.I.T. by Kerwin et al [10,11,12]. A variety of formulations and wake models have been used over the years. Blade-element momentum theory and Goldstein corrections can be used to calculate the propeller race information necessary for an inflow field to the rudder.

However, this approach uses considerable empiricism and it is difficult to take account of the upstream influence of the rudder on the propellers performance. In this work it is proposed to use the lifting surface formulation for modelling both the rudder and the propeller. This approach will give the necessary information at a reasonable computational cost for calculating the manoeuvring characteristics of a ship rudder and propeller.

An implicit method for the solution of the inviscid, incompressible, irrotational steady-state scheme such as a lifting-surface panel method provides a rigorous test for the performance of transputer networks in solving general fluid dynamic problems.

2 THEORY

Based on the review of literature, a reasonable compromise between computational effort and physical accuracy in modelling the flow interaction was concluded to be the use of a lifting surface panel method. The same panel method would model in isolation the individual rudder and propeller. The interaction between the two lifting surfaces is accounted for by the modification of their respective inflow velocity fields.

In a lifting surface formulation the approximation of the full Navier-Stokes equation assumes that the flow is inviscid, incompressible and irrotational and satisfies Laplace's potential equation:

$$\nabla^2\phi = 0 \qquad [1]$$

A detailed description of the method and a review of its historical development is given by Hess[13]. Lamb[14] showed that for a quantity satisfying Laplace's equation may be written as an integral over the bounding surface S as a source distribution per unit area and a normal dipole distribution per unit area distributed over the surface. If v represents the disturbance velocity field due to the bounding surface (or body) and is

defined as the difference between the local velocity at a point and the free-stream velocity then:

$$\underline{v} = \nabla\phi \qquad [2]$$

where ϕ is defined as the disturbance potential. This can be expressed in terms of a surface integral as:

$$\phi = \int\int_{S_B} [\frac{1}{r}\sigma + \frac{\partial}{\partial n}(\frac{1}{r})\mu]dS + \int\int_{S_W} \frac{\partial}{\partial n}(\frac{1}{r})\mu\, dS \qquad [3]$$

Where S_B is the surface of the body and S_W a trailing wake sheet, as illustrated in figure 1 for a two-dimensional flow. In the expression r is the distance from the point for which the potential is being determined to the integration point on the surface and $\partial/\partial n$ is a partial derivative in the direction normal to the local surface. A dipole distribution is used to represent the wake sheet. Hess [15] showed this can be directly related to the vorticity distribution used in vortex lattice methods (VLM).

The conditions imposed on the disturbance potential are that the (from Hess[13]):

1) velocity potential satisfies Laplaces equation everywhere outside of the body and wake;

2) the disturbance potential due to the body vanishes at infinity;

3) normal component of velocity is zero on the body surface;

4) kutta-joukowsky condition of a finite velocity at the body trailing edge is satisfied.

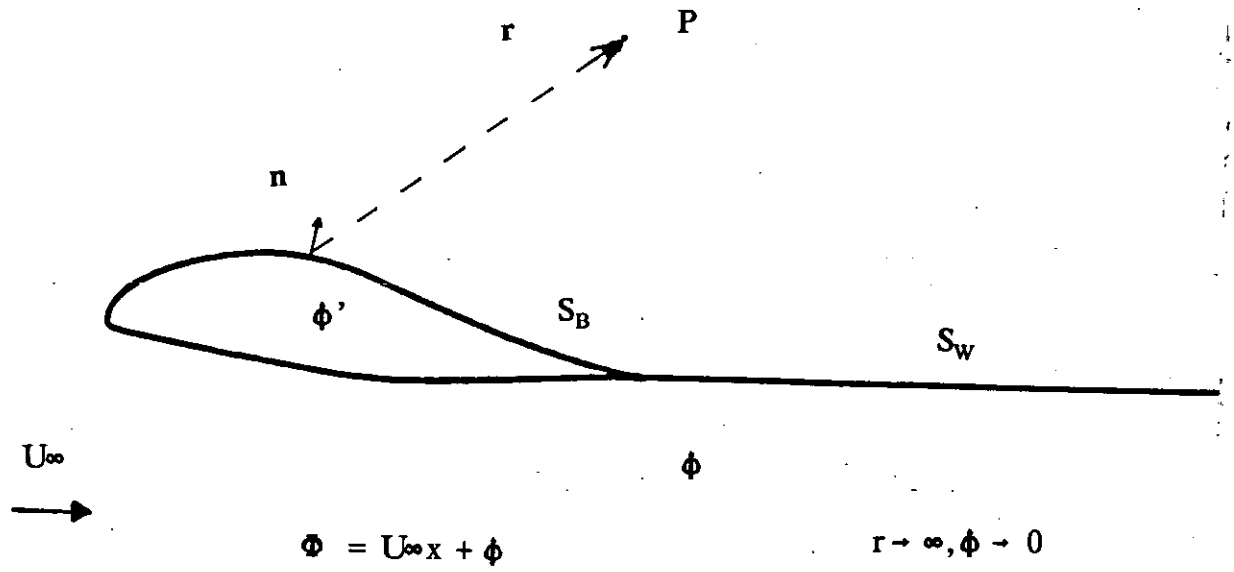


Figure 1 Two-dimensional lifting surface schematic

5) the trailing wake sheet is a stream surface with equal pressure either side.

For a steady-state solution the wake dipole strength distribution is uniquely determined by the application of the Kutta condition at the body trailing edge. As conditions (1) and (2) are satisfied as functions of μ and σ, conditions (3) and (4) are used to determine μ and σ on the body. The Kutta condition only applies at the trailing edge and some other relationship has to be used to uniquely determine the distribution of μ and σ over the body. The numerical resolution of this non-uniqueness is referred to as the singularity mix of the lifting-surface method.

The choice of numerical representation of the body integral singularity mix and boundary conditions are described in the following sections.

3 MORINO'S METHOD

Lee[16] carried out a two-dimensional investigation into four possible schemes for the solution of Lamb's equation. In this the body source strength per unit area σ was defined as:

$$\sigma = \frac{\partial \phi}{\partial n} - \frac{\partial \phi'}{\partial n} \qquad [4]$$

where $d\phi/dn$ is the normal derivative of the velocity potential on the outer surface of the body and $d\phi'/dn$ is the normal derivative on the interior of the body surface. Similarly the body dipole distribution per unit area μ was defined as:

$$\mu = \phi - \phi' \qquad [5]$$

where ϕ and ϕ' are respectively the disturbance velocity potential on the exterior and interior of the body surface. Equation [3] can then be written as:

$$4\pi\phi(p) = \iint_{S_B} [(\phi(q) - \phi'(q))\frac{\partial}{\partial n}\frac{1}{R} - (\frac{\partial \phi}{\partial n} - \frac{\partial \phi'}{\partial n})\frac{1}{R}\ ]\, dS + \iint_{S_W} \Delta\phi \frac{\partial}{\partial n}(\frac{1}{R})dS \qquad [6]$$

As the flow within the body surface S is non-physical the choice of different internal velocity potential ϕ' will result in different singularity mixes. These were classified, by Lee, as follows.

1) Total velocity: If the normal derivative with respect to the field point is taken of equation [6] a velocity formulation results. An internal potential such that:

$$\frac{\partial \phi}{\partial n} = \frac{\partial \phi'}{\partial n} \qquad [7]$$

the body source strength becomes identically zero and [6] becomes:

$$2\pi \frac{\partial \phi}{\partial n} = \int\int_{S_B} \mu \frac{\partial^2}{\partial n_p \partial n_q} \frac{1}{R} dS + \int\int_{S_W} \Delta\phi \frac{\partial^2}{\partial n_p \partial n_q} \frac{1}{R} dS \qquad [8]$$

which is an integral equation of the first kind for the unknown dipole strength μ. The dipole strength is:

$$\mu = \phi + \phi_\infty \qquad [9]$$

where ϕ_∞ is the inflow or free-stream potential

$$U_\infty = \nabla\phi_\infty \qquad [10]$$

Hess and Smith's original velocity method was based on a vortical formulation rather than a dipole distribution.

2) Perturbation velocity: For the velocity formulation, where the equation [6] is differentiated normal to the field point, if the internal potential ϕ' is set to zero it can be shown that $d\phi'/dn = 0$ on S_B. Therefore [6] becomes:

$$4\pi \frac{\partial \phi}{\partial n} = \int\int_{S_B} \sigma \frac{\partial}{\partial n}\left(\frac{-1}{R}\right) dS + \int\int_{S_B} \mu \frac{\partial^2}{\partial n_p \partial n_q} \frac{1}{R} dS + \int\int_{S_W} \Delta\phi \frac{\partial^2}{\partial n_p \partial n_q} \frac{1}{R} dS \qquad [11]$$

where $\sigma = d\phi/dn$ and $\mu = \phi$. Given the source strength from the normal boundary condition this again is an integral equation of the first kind for the unknown dipole strength μ.

3) Total Potential: In equation[6], if the inflow velocity potential can be expressed as:

$$\nabla\phi_\infty = U_\infty \qquad [12]$$

and then the internal velocity potential is set constant to:

$$\phi' = -\phi_\infty \qquad [13]$$

it becomes:

$$2\pi\Phi_p = 4\pi\phi_\infty + \iint_{S_B}\phi(q)\frac{\partial}{\partial n}\frac{1}{R}dS + \iint_{S_W}\Delta\Phi\frac{\partial}{\partial n}\frac{1}{R}dS \qquad [14]$$

Where Φ is the total potential. This represents the body as a normal dipole distribution and is a Fredholm integral equation of the second kind for the total potential Φ.
(4) Perturbation Potential: If the internal potential ϕ' is set to zero and from that $d\phi'/dn$ is zero equation [6] becomes:

$$2\pi\phi = \iint_{S_B}(\phi\frac{\partial}{\partial n}\frac{1}{R} - \frac{\partial\phi}{\partial n}\frac{1}{R})dS + \iint_{S_W}\Delta\phi\frac{\partial}{\partial n}\frac{1}{R}dS \qquad [15]$$

As $d\phi/dn$ is known as it has to satisfy the normal surface boundary condition, this expression is also a Fredholm integral equation of the second kind for the dipole strength μ. This dipole strength is equal to the external value of the potential on the body surface.

The conclusion of Lee's study was, that for lifting surfaces which have both thin and thick sections (e.g. propeller blades), the perturbation potential method(4) taken from the work by Morino and Kuo[17] was the most suitable. The principal advantages of this method are that because panel potential (scalar) rather than velocity (vector) influence

coefficients are calculated only a third of the memory requirement for the method is needed. Also, the perturbation potential influence coefficient is an order less singular. Kerwin and Lee [11] used this method and found it robust in their investigation of ducted propellers.

Maskew[18] used Morino's method which is a low-order panel method as opposed to higher-order formulations where the source/dipole strengths are not assumed piece-wise constant over a panel. The simplicity of low-order panel methods and their low computing cost allows great flexibility in their applications. In conclusion Maskew stated that a low-order method using Morino's formulation gave comparable accuracy to higher-order formulations while avoiding many problems associated with other low-order methods.

Margesson et al.[19] compared five production surface panel methods, three of which used low-order singularities. All but one used as a basis the formulation due to Morino and the other a higher-order formulation due to Hess. The two higher-order Morino formulation provided the same accuracy with lower panel numbers but at greater computational cost.

Morino's numerical procedure is based on representing the body surface by a series of N quadrilateral panels each with an unknown but constant dipole strength per unit area. The vertices of these panels are located on the actual surface of the body. The wake sheet is represented by M panels placed on the stream-surface from the trailing edge of the body surface. Its dipole strength per unit area is related to the difference in dipole potential at the trailing edge. In Morino's work the wake strength μ_w was equated to the difference in potential between the upper and lower surface at the trailing edge.

That is:

$$\mu_w = \phi_u - \phi_l \qquad [16]$$

On the body surface the source strength per unit area is prescribed by satisfying the condition for zero normal velocity at the panel centroid:

$$\sigma_s = \bar{U}\bar{n} \qquad [17]$$

where **n** is the unit normal outward from the panel surface and U the specified inflow velocity at the panel centroid..

The numerical discretisation of equation[15] gives for the potential at the centroid of panel i as:

$$\phi_i = \frac{1}{2\pi}\sum_{j=1}^{N} \left((U_\infty . n_j) S_{ij} - \phi_j D_{ij} \right) + \sum_{k=1}^{M} \Delta\phi_k W_{ik} \qquad [18]$$

where for panel j: S_{ij} is the source influence coefficient of a unit strength panel; D_j the dipole influence coefficient; and W_{ik} the influence of the constant strength wake strip extending to infinity. As there are N independent equations corresponding to the N body surface panel centroids, equation[18] can be evaluated. Expressed in matrix form it becomes:

$$[D_{ij}]\underline{\phi} + [W_{ik}]\underline{\Delta\phi} = [S_{ij}](\underline{U_\infty}.\underline{n}) \qquad [19]$$

For Morino's original trailing edge Kutta condition, which directly relates $\Delta\phi$ to the difference in trailing edge panel potential, the matrix expression [19] can then be directly solved to give the vector of dipole potentials ϕ. Numerical differentiation of dipole potential along the body surface allows the surface velocity and hence pressures on the surface to be evaluated.

The method used for the evaluation of the individual influence coefficient elements of the matrices S_{ij}, D_{ij} and W_{ik} are described in the next section.

4 NEWMAN PANEL

At the heart of a lifting surface panel method is the efficient calculation of the potential (or velocity) influence coefficients at a field point due to a particular panel's source or dipole distribution. Newman[20] derived expressions for calculating the exact influence coefficients of a constant strength distribution of sources and normal dipoles over a quadrilateral panel. The method of calculation of the dipole influence coefficient avoids the use of numerical integration. The approach of the paper was different from that used originally by Hess and Smith[8] although the form of the exact source influence coefficient is algebraically similar. Following Hess & Smith arbitrary order multi-pole expansions are derived so that at greater distances from the panel the accuracy of the source and dipole influence coefficient is maintained while at the same time the computational time is reduced.

The scheme used in this work, to reduce computational effort, was to choose the scheme to calculate the influence coefficients based on L. Parameter L is the ratio of the distance between the point of interest and panel centroid divided by the length of the panels main diagonal. Newman's exact formulation is used when L is between 0 and 2, a 4^{th} order multi-pole approximation between 2L and 2.45L, a 2^{nd} order multi-pole approximation between 2.45L and 4L, and a point source/normal dipole for distances greater than 4L. Figure 2 shows the influence coefficients for this composite scheme compared to that for the exact, 4^{th}, 2^{nd}, and point expressions for a field point at different heights above the centroid of the panel. Appendix 1 provides the mathematical derivation of the exact, multipole and point expressions and their OCCAM2 interpretation.

The modelling of the interaction requires the evaluation of total velocity in the field domain away from the lifting surface. The expressions given by Newman are for the potential influence only. For potential flow, the velocity is the local gradient of the potential. Therefore, the velocity influence due to a panel has been derived by applying the Grad operator ∇ to the source and dipole influence expressions. These relationships are also given in Appendix 1. Figure 3 shows, for two planes through a square panel, an

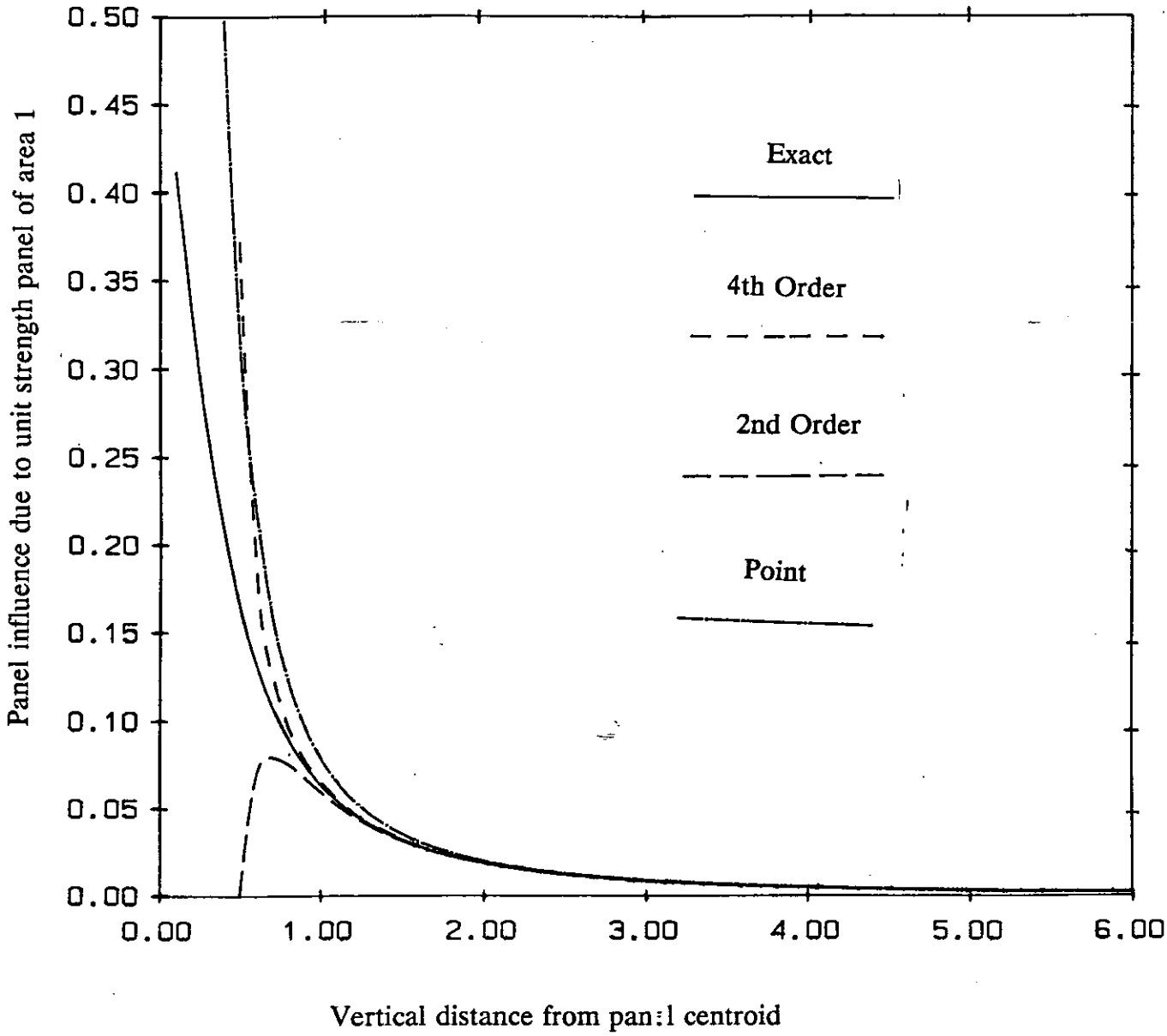


Figure 2 Comparison of scheme for calculating the dipole influence coefficient against distance above the centre of a square panel

isometric contour plot showing lines of constant total velocity for a uniform dipole distribution. The correspondence between a uniform dipole distribution and an element of four vortex lines can be directly seen.

The multipole expressions require the calculation of 13 coefficients based on the

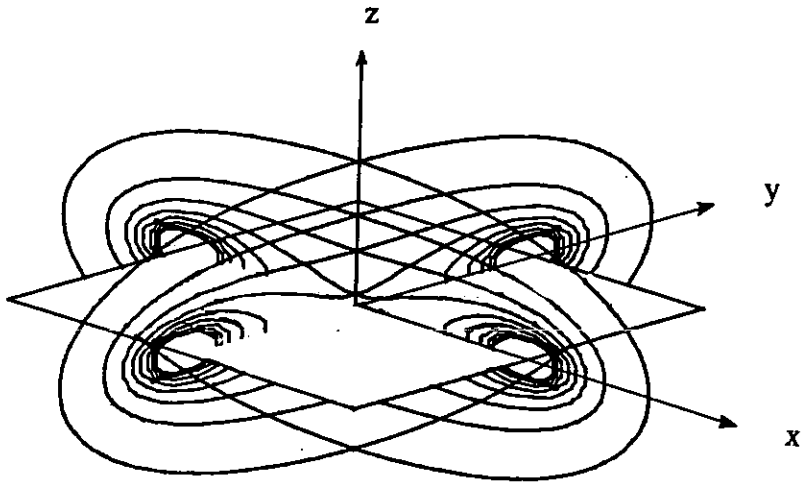


Figure 3 Isometric contour plot showing total velocity lines for two planes through a square dipole panel.

geometrical properties of the panel. To minimise computational time it would be sensible to only ever calculate these values once for each panel. However, the memory requirement to do this severely limits the maximum number of panels for a problem and it was decided to recalculate the coefficients for the four panel vertices every time a panel was used. Table 1 provides timings for calculating these geometrical coefficients for a single panel and also for the four different methods for calculating the source and normal dipole influence coefficients.

Table 1 Comparison of time taken to calculate dipole influence coefficents using different schemes between single T800 transputer and VAX 11/750.

	Clock Ticks T800 (low)	Time (mSec)	Vax 11/750 (mSec)
Geometry	119	7.62	-
Exact	27	1.73	1.5
4th Order	17	1.09	0.4
2nd Order	12	0.77	0.2
Point	9	0.58	-

These timings were obtained using a single T800 transputer running at low priority and are averaged over a large number identical calculations. The figures are given in both units of the T800 internal clock and milliseconds. Also shown are the approximate values given in Newman[20] for calculating the source and normal dipole influence coefficients in single precision on a VAX 11/750.

It can be seen that:

1) Calculation time of the geometrical coefficients is only 4.4 times that of calculating the exact influence coefficient. Therefore, if at any one time a number of field point influence coefficients can be calculated using information for a single panel there will not be that large a computational penalty for recalculation.

2) As expected an individual T800 micro-processor is slower than the VAX 11/750 but not by much!

5 KUTTA CONDITION

For a steady-state solution, the dipole strength of the trailing wake sheet has a constant strength in the stream-wise direction. This strength is directly related to the circulation around the lifting surface. The original Kutta condition, implemented by Morino, involved setting the trailing wake sheet dipole strength equal to that of the difference in perturbation potential at the trailing edge. This implies that the pressure difference at the trailing edge would be close to zero. Lee showed that a source term should also be included to ensure that there was zero difference in <u>total</u> potential caused by the difference in source strength of the two trailing edge panels and gave the expression for this as

$$\Phi_s = \bar{U}_o . \bar{r} \qquad [20]$$

For three-dimensional flow, **r** is the vector between the centroids of the two trailing edge panels. With this additional term, when significant cross-flow occurs at the trailing edge, the upper and lower panels will not necessarily be at the same pressure and a non-physical trailing edge pressure loading occurs. This was seen by Lee as the need to explicitly equate the upper and lower panel pressure using an iterative scheme to correct the dipole wakestrength based on a factor K multiplying the pressure loading at the trailing edge from the previous iteration. Only sketchy details are given by Lee into the actual evaluation of K. However, the scheme is described as a Newton-Raphson method and from this hint an expression for K has been derived.

The method developed in this work uses the description given in [11,16] as a basis. An explicit condition of no pressure loading is enforced across the upper and lower trailing edge panels. That is, the pressures are equal:

$$\Delta P_{te} = P_u - P_l = 0 \qquad [21]$$

If it is assumed that Δ Cp is primarily a function of the local trailing edge wake sheet strength $\Delta\phi$ an iterative Newton-Raphson approach is suggested to determine the wake strength for the point of zero pressure difference at the trailing edge. That is:

$$\Delta\phi^k = \Delta\phi^{k-1} - \frac{\Delta P(\Delta\phi^{k-1})}{\frac{d\Delta P}{d\Delta\phi}} \qquad [22]$$

where the trailing edge pressure loading Δ P is the difference in pressure between the upper and lower panels at the trailing edge.

$$\Delta P = P_U - P_L \qquad [23]$$

Substituting for pressure in terms of surface velocity **V** gives

$$\Delta P = (V_L \cdot V_L) - (V_U \cdot V_U) \qquad [24]$$

Differentiating with respect to a change in wake strength

$$\frac{d\Delta P}{d\Delta\phi} = 2\left(V_L \cdot \frac{dV_L}{d\Delta\phi} - V_U \cdot \frac{dV_U}{d\Delta\phi}\right) \qquad [25]$$

By deriving an expression for surface velocity in terms of the velocity influence sum of all the source and dipole panels, and wake strips (see next section 6, equation[30]) and then differentiating with respect to the wake sheet strength and assuming that the principal influence on a pair of trailing edge panel's is due to the attached wake strip then $dV_L/d\Delta\phi$ can be expressed as:

$$\frac{dV_L}{d\Delta\phi} = V_{WL} \quad and \quad \frac{dV_U}{d\Delta\phi} = V_{WU} \qquad [26]$$

where V_{WL} and V_{WU} are the velocity influence coefficients of the wake strip attached to the panels at their respective centroids. All the components can then be numerically evaluated and hence the wake strength updated.

The zero'th order (k=0) approximation for the wake strength is taken to be the original Morino kutta condition:

$$\Delta\phi = \phi_u - \phi_l \qquad [27]$$

as ϕ_u and ϕ_l are unknown then the numerical equation[19] is arranged with the unknowns on the left hand side:

$$[D_{ij} + W_{ik}]\underline{\phi} = [S_{ij}]\underline{U_\infty \cdot n_j} \qquad [28]$$

Once the solution vector ϕ is obtained this is used to calculate ΔCp at the trailing edge. Using equation[22] the correction to the wake strength is found. This correction vector of known strength is multiplied by the wake strip influence coefficient matrix W_{ik} and applied to the right hand side of the equation. This modifies Morino's original matrix expression to:

$$[D_{ij} + W_{ik}] \Phi = [S_{ij}] U_{\infty} \cdot n_j - [W_{ik}] \left(\frac{d\Delta\Phi}{d\Delta P} \Delta P \right)^k \qquad [29]$$

The process is repeated until the pressure loading has been removed to any significant degree.

6 ADAPTIVE WAKE MODEL

The accurate modelling of a wake sheet convected downstream from a lifting surface is difficult and the subject of much research. The zero thickness wake sheet should be aligned tangential to the local flow to ensure that there is no pressure jump across the sheet. The specification of the wake geometry cannot be known a priori as its location will depend on the eventual flow solution which is dependent on the actual wake geometry. There are schemes which iteratively adapt the wake shape until a converged shape is obtained with the wake everywhere aligned with the local flow. These schemes greatly increase the computational cost of solution and are prone to problems of sensitivity to the initial geometry especially where considerable wake roll-up occurs. Many of these problems originate from the non-physical representation of the wake as a zero thickness constant strength sheet extending to infinity. However, a correct geometrical representation of the wake sheet is necessary if good numerical results are to be obtained.

In the Morino formulation the wake sheet is represented as a series of trailing dipole panels. The influence coefficient of a wake strip W_k extending from the body trailing edge to infinity is the sum of the influence coefficients of all the panels making up the strip. For practical calculations the panelled wake strip will only extend a finite distance downstream. This distance can be determined by carrying out sensitivity studies where the length of wake strip is increased until no appreciable change in body dipole strengths occurs. Another, more physical approach, is to use a far-field approximation for a wake sheet extending to infinity from a given distance downstream of the body. This method was used by Lee.

For non-rotating lifting surfaces where the principal aim is to obtain surface pressure and hence body forces the wake sheet evolution downstream is not critical to the final solution. For these cases a fixed (or frozen) geometry wake can be prescribed. Usually downstream of the body the wake is aligned in the free-stream direction and close to the body the wake is aligned with the direction of the lifting surface chord.

The accurate representation of the wake sheet generated by a rotating lifting surface, such as a propeller blade, presents major difficulties. Considerable work has been done in developing propeller wake models. These usually involve some amount of empiricism and wake-adaption techniques. In Lee's work the prescribed propeller wake geometry was generated using a previously developed lifting-surface code[12] and experimental data. In a recent paper by Maitre and Rowe[21] a numerical method for iteratively relaxing an initial wake following the propeller geometric pitch was found to converge. It gave good results, although was not completely free to adapt near the hub where the wake radius was held constant.

In this work, to allow for possible variation in propeller design, it was decided to develop a wake adaption method based on that by Maitre and Rowe. The wake is divided into a near and far region. In the far region the movement of each wake panel is based on that of the last panel in the near region. For the near region the velocity at each panel centroid is calculated by summing the velocity influence coefficients of the body source and dipole panels and the wake dipole panels. The total velocity at the panel centroid is the sum of the body rotational velocity, inflow velocity and the disturbance velocity:

$$V_T = V_o + r \times \omega + \sum_{j=1}^{N} \left(\phi_j V_{D_j} + (V_o \cdot n_j) V_{S_j} \right) + \sum_{k=1}^{M} \left(\Delta\phi_k V_{W_k} \right) \qquad [30]$$

The total velocity $\mathbf{V}_t$ is translated into cylindrical coordinates where **a** is a unit vector in the direction of rotation, **r** a unit vector in the direction of the panel centroid perpendicular to **a**, and **t** forms the orthonormal set **t** = **a** x **r**. This gives

$$
\begin{aligned}
V_a &= V_T \cdot a \\
V_r &= V_T \cdot r \\
\omega &= \frac{(V_T \cdot t)}{R}
\end{aligned}
\qquad [31]
$$

where R is the radial position of the panel centroid. As the velocity of the panel nodes is required this is taken to be the sum of the cylindrical components of velocity of the surrounding panels divided by the number of panels surrounding the node. That is

$$
V_{node} = \frac{1}{N_p} \sum_{1}^{Np} V_{centroid} \qquad [32]
$$

The positions of the nodes of a wake strip are altered recursively starting from just aft of the trailing edge using the following relationships which give, in cylindrical coordinates (r,a,θ), for node k+1(Maitre).

$$
\begin{aligned}
r^i_{k+1} &= r^i_k + v_r \Delta t_k \\
\theta^i_{k+1} &= \theta^i_k + \omega R \Delta t_k \\
a^i_{k+1} &= a^i_k + v_a \Delta t_k
\end{aligned}
\qquad [33]
$$

The time step in this work is taken to be

$$
\Delta t_k = \frac{L}{|V_T|} \qquad [34]
$$

where L is the straight-line distance between two successive nodes. Using this method ensures that the stream-wise length of the panel sides remains constant.

This method was also used for non-rotating bodies by working in cartesian coordinates for the recursive relationships.

7 GEOMETRICAL DEFINITION

7.1 Introduction

Morino's method locates quadrilateral panels on the actual body surface. For lower-order formulations these panels consist of a number of connected straight lines which form a closed surface. Generally, only quadrilateral panels are used and where possible one of the principal directions of the panels is aligned in the flow dierction.

An accurate geometrical definition of a three-dimensional body as a closed surface constructed from quadrilateral elements is a crucial component of a lifting-surface analysis. How easily arbitrary bodies can be defined will determine the usefulness of a numerical analysis code. Lee [16] tailored the flow solver for a particular geometry e.g. semi-span wing or propeller blade. This approach is of limited use and a better approach is that used in the commercial code Quadpan[18] where an input file is used which contains the four vertices for each panel. The preparation of this pre-processing file is especially time consuming as a new file has to be created for every change in panelling density. However, such a input file format allows arbitrary bodies to be tested without recourse to creating individual executable code for every geometry.

In this work a principal feature is the investigation of the performance of a lifting-surface code on a transputer network. Therefore, it is necessary to have a simple means of scaling the overall problem size by altering the number of panels used to define a lifting-surface. Therefore, the decision was made to combine the two approaches described and generate the actual panel vertex coordinates within the program but use a pre-processing file to define the number of bodies and their individual geometry. This allows a problem to be scaled by using the internal panel generator to produce a different number of panels for the same overall body geometry. In addition the pre-processing input file can be used to define the interaction velocity field.

7.2 Parametric Cubic Spline

A variety of means are available for defining a three-dimensional surface (or body). In Naval Architecture a ship hull form is conventionally defined using a series of lines which lie in parallel planes. These lines, whether waterlines, buttocklines or transverse sections, are themselves defined in terms of an ordered set of coordinates. A mathematical relationship can then be used to generate the curved lines between the coordinates and hence specify a three-dimensional surface.

An extremely useful and straightforward means of relating the line coordinates to the curve passing through them is that of a parametric cubic spline procedure. A spline approximation is defined (Kreyszig [32]) as a piece-wise polynomial approximation to a curve. Each segment of a line is represented as a polynomial . For a cubic spline at the end of each segment the gradient and curvature of the polynomial expression are matched to the adjoining polynomial expressions. This results in a curve made up of a

[35]

series of cubic lines i.e.

$$y = k_1 t^3 + k_2 t^2 + k_3 t + k_4 \qquad [36]$$

where the values of the constants k_1,k_2,k_3 and k_4 are solved using the end conditions of gradient and curvature continuity. Defining a value of the parameter t will uniquely define the value of the y coordinate. Similar relationships for the x and z coordinates allow a three-dimensional curve to be uniquely determined by the single parameter t. This parameter is the distance along the original curve and this is usually approximated as the straight-line distance between points. That is:

$$\Delta t = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2} \qquad [37]$$

For the purposes of this work a surface definition using parametric cubic spline provides an accurate approximation to a three-dimensional surface. The end conditions used in this work is that of zero curvature.

7.3 Body Definition

The facility to define a number of bodies or separate parts of the same body independently allows complex geometries and flows to be investigated. If each body is defined relative to its own body coordinate system there has to be a means of relating the coordinate systems of all the bodies to the overall cartesian coordinate system in which the panels are defined. A set of four vectors (**S, P, O, A**) were used to carry out this transformation to be carried out. The definition of these vectors allows great flexibility for parametric studies of complex geometries.

Analytically the flow solution for incompressible potential flow can be made non-dimensional and independent of physical scale. However, to allow direct comparison with experiment it was decided to use the actual physical dimensions for the numerical analysis. Often geometries, for example rudder sections, are given in terms of non-dimensional percentage chord. To allow the checking of data and also alteration of such parameters as rudder aspect ratio it was decided to define body coordinates in terms of a scale vector **S**:

$$\boldsymbol{S} = \begin{bmatrix} X_{sca} \\ Y_{sca} \\ Z_{sca} \end{bmatrix} \qquad [38]$$

where X_{sca}, Y_{sca}, Z_{sca} are a scale factors for each of the body cartesian coordinates.

A body pivot vector **P** is defined which, in the body coordinate system, locates the position about which the body can be rotated. An ordered rotation vector **A** defines these angular rotations about the pivot in radians. These rotations are in order about

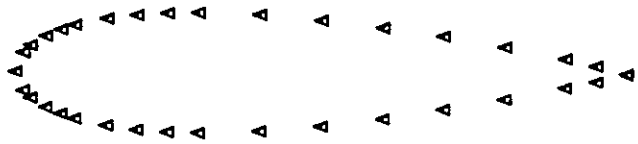
the body X,Y and Z axis. For example, for a rudder with a pivot at 25% of its chord, it can be set at any angle of incidence by changing the value of Z-axis rotation angle component of vector **A**. Effectively, this allows the individual body coordinate systems to be rotated with respect to the overall coordinate system.An absolute offset vector **O** relates the origin of each body coordinate system to that of the overall coordinate system. Changing the offset vector **O** allows the parametric study of lifting surface body separation to be investigated.

Each individual body (or part body) is defined in the same manner as that of a ship hull form; as an ordered series of lines with each line containing an ordered set of three-dimensional points. For a closed lifting body such as a rudder or wing, a wake sheet will be connected to the trailing edge and it is therefore sensible to start and finish each body definition line at the trailing edge. For non-lifting bodies this is not essential. Figure 4 illustrates the process of defining a rudder as a series of parametric cubic splines. The order of the series of lines and points is important in determining the direction of the panel surface normal used in calculating panel influence coefficients. The lines are ordered so that the normal vector to a panel always faces out into the exterior flow field.

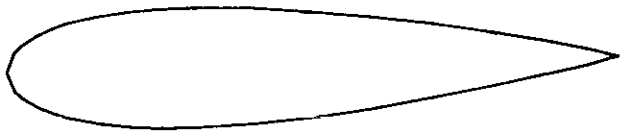
The numerical discretisation of the body geometry into a number of quadrilateral panels requires the number of panels in the section data direction (parameter t) Nt and in the line direction (parameter s) Ns. The process of generating the Nt by Ns panels is carried out by:

1) For each line of section data producing (Nt + 1) coordinates using a parametric cubic spline through the section data. The distribution of points within a section can either be spaced at intervals of Δ t, where

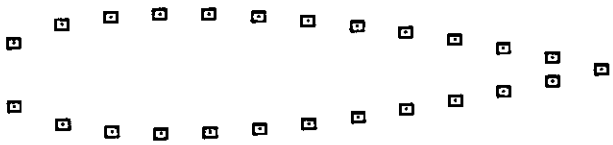
$$\Delta t = \frac{|T|}{N_t} \qquad [39]$$



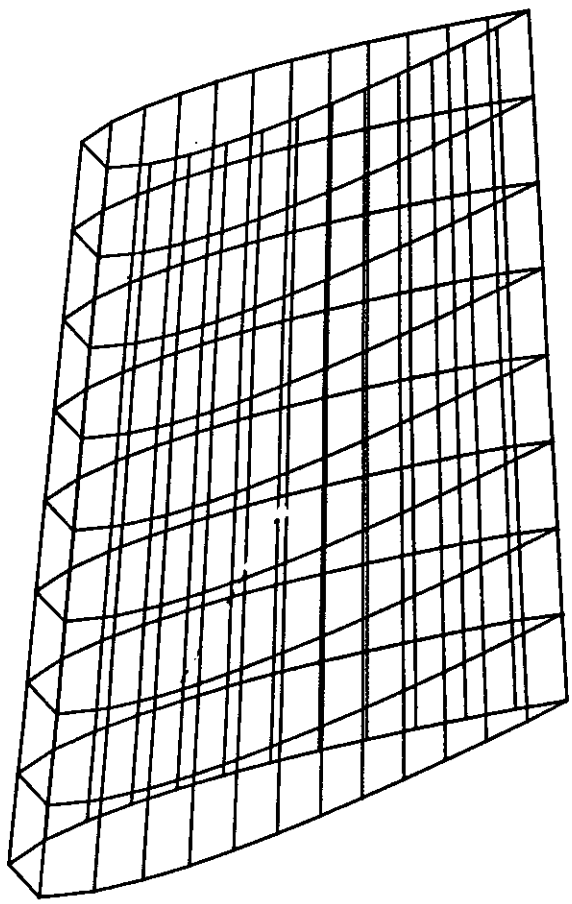
(1) Definition Points



(2) Spline through points



(3) Panel nodes



Isometric view of panelled rudder

Figure 4 Process for defining a rudder surface using parametric splines

and T is the total parametric length of the line or where Δt is some function of t. For closed bodies the start and finish point of each line are made identical.

2) A parametric cubic spline is formed in parameter s by using the i^{th} point form each of the section data splines. Each of these new cubic splines is used to define (Ns+1) points. Repeating this for (Nt+1) points generates the coordinates vertices for all the required panels.

A wake sheet can be panelled in a similar manner. To ensure accurate matching of a body trailing edge and its wake sheet the number of section lines must be the same for body and wake. Also, the section lines should have identical start coordinates. This ensures that an identical s parameter spline will be produced and sub-divided so that wake strip coordinates correspond to those of the trailing edge panels.

7.4 Flow Definition

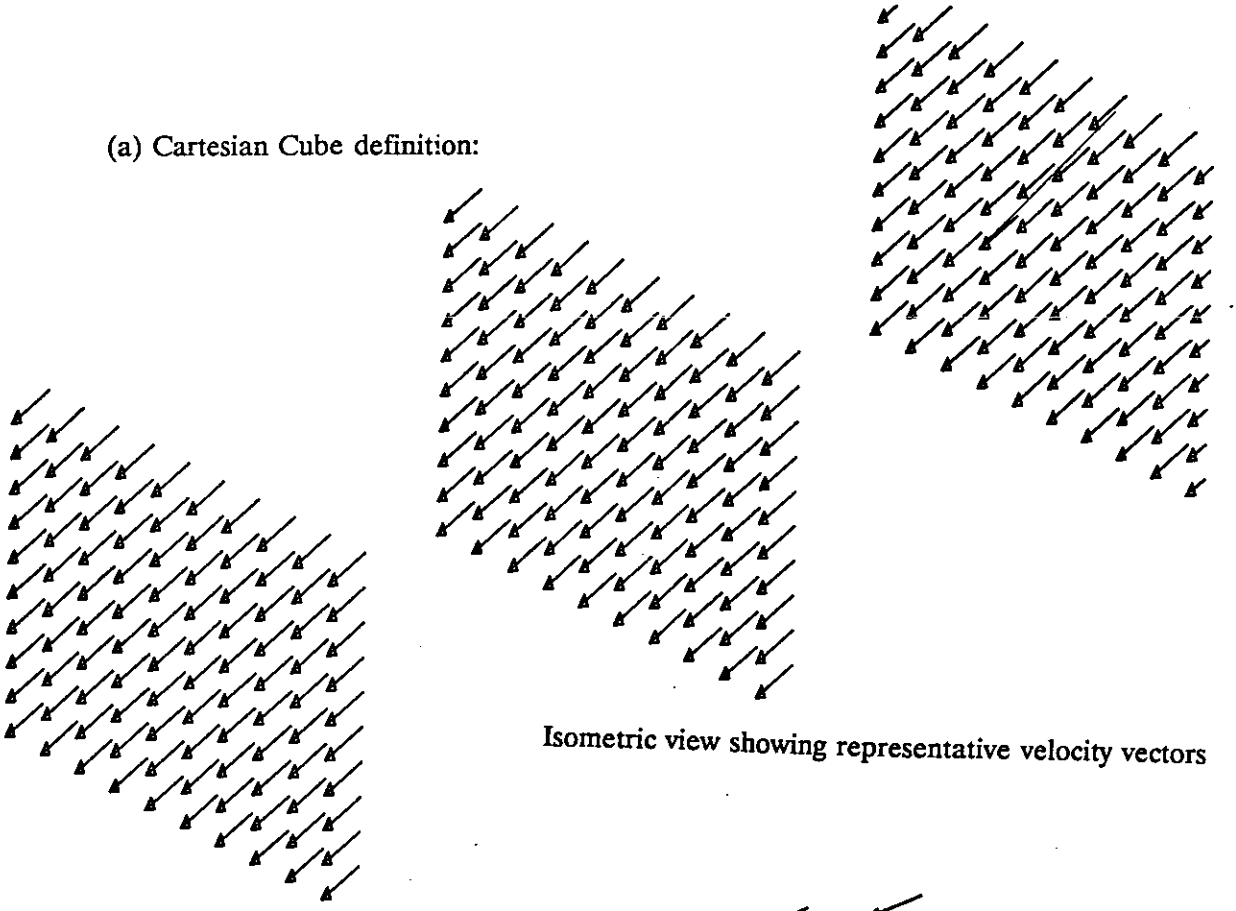
The disturbance velocity field generated by a body is superimposed on the velocity field existing in the absence of the body. For many problems this inflow velocity field is a constant velocity in the free-stream direction throughout the domain and can be directly specified. In the case of a rotating body the velocity on its surface will be the vector sum of the free-stream velocity and the body's rotational speed. That is:

where $\mathbf{r}$ is the position vector of the panel centroid from a point at the origin of the axis of rotation. The angular velocity $\boldsymbol{\omega}$ vector is the scalar speed of rotation in the direction of the axis of rotation.

The modelling of the interaction between a ship rudder and propeller requires a spatially varying flow definition in the absence of a lifting-surface. So in addition to the uniform free-stream velocity and where necessary rotational velocity a spatially varying velocity distribution is needed. To facilitate modelling, a velocity field definition is used whereby

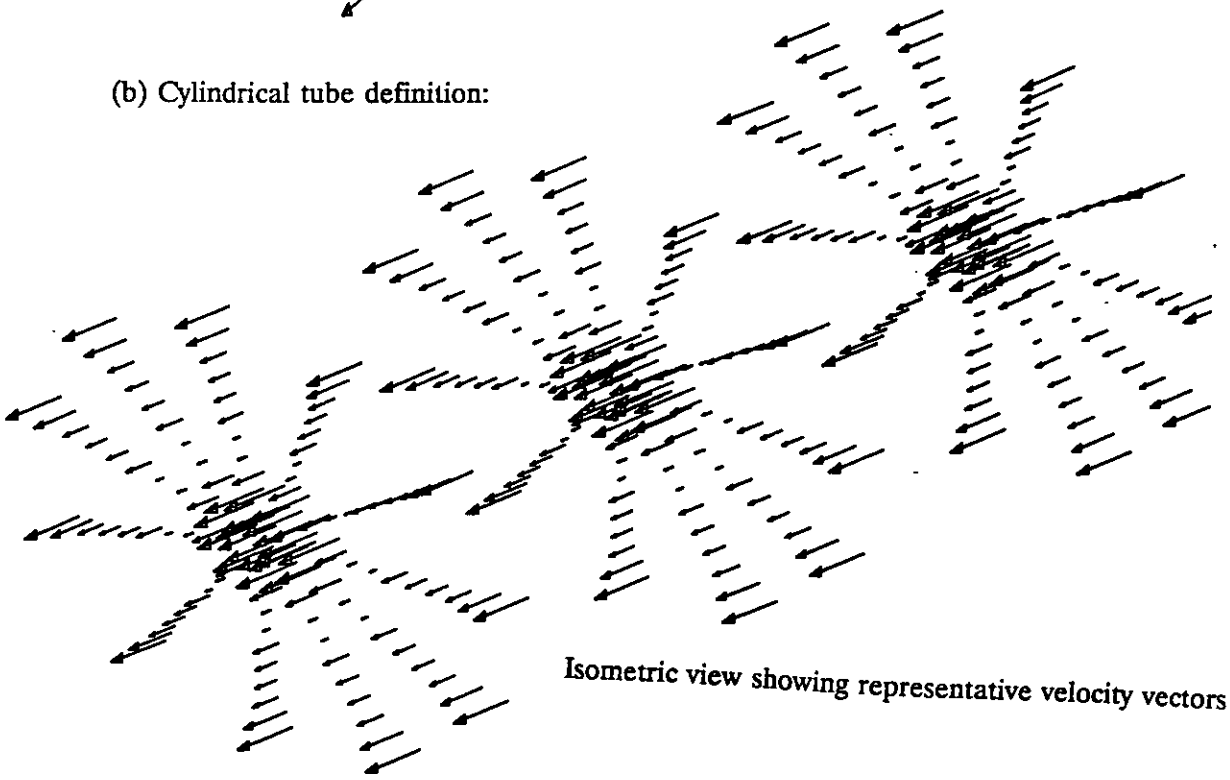
the u,v and w components of velocity corresponding to velocities in the x,y and z

(a) Cartesian Cube definition:



Isometric view showing representative velocity vectors

(b) Cylindrical tube definition:



Isometric view showing representative velocity vectors

Figure 5 Velocity field definitions: Cube and Tube

directions are specified at a uniform spacing within a three-dimensional block. This is illustrated in figure 5. A cylindrical and cubic block were respectively used for the propeller and rudder inflow fields.

Theoretically, the interaction velocity field could be replaced by the location of suitably placed source/dipole panels of known strength and used directly in the calculation of influence coefficients. However, it is more convenient to specify a velocity field and if the velocity field satisfies Laplace's equation identical. That is if:

$$\nabla^2\phi = 0 \qquad [41]$$

then as

$$u = -\frac{\partial\phi}{\partial x},\ v = -\frac{\partial\phi}{\partial y},\ and\ w = -\frac{\partial\phi}{\partial z} \qquad [42]$$

to satisfy Laplace's equation the interaction velocity filed **Ui** must satisfy:

$$\underline{\nabla} \cdot \underline{U}_i = 0 \qquad [43]$$

or expressed as the sum of velocity differentials should everywhere equal zero:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \qquad [44]$$

It is probably acceptable (and more practical) to allow a certain amount of deviation from this condition. Possible difficulties will arise from the use of circumferentially averaged quantities in deriving interaction velocity fields.

8 Calculation of Aerodynamic Coefficients

The numerical solution of Morino's method gives a result vector which specifies a dipole strength at the centre of each panel. As explained previously, this corresponds to the potential ϕ on the surface of the body. To obtain practical engineering information from

this surface potential distribution a numerical differentiation has to be carried out. The differentiation gives the disturbance velocity tangential to the panel surface. The total velocity at the panel centroid is the vector sum of the tangential disturbance velocity U_d, and the normal component of the body surface $r \times \omega$ and interaction field velocity U_i.

$$U_t = U_d + \left(U_i - (U_i \cdot n)\, n \right) + \left(r \times \omega - \left((r \times \omega) \cdot n \right) n \right) \qquad [45]$$

where **n** is a unit vector normal to the panel surface.

There are two methods of obtaining the disturbance tangential surface velocity[16]. That is either by fitting a parametric cubic spline through the panel centroids and using the cubic polynomial constants to obtain the gradient and hence velocity or by a finite difference approach. The spline approach requires the assemblage of information from all the panels in a particular parametric direction. On the other hand, the finite difference method only requires information about from its four neighbouring

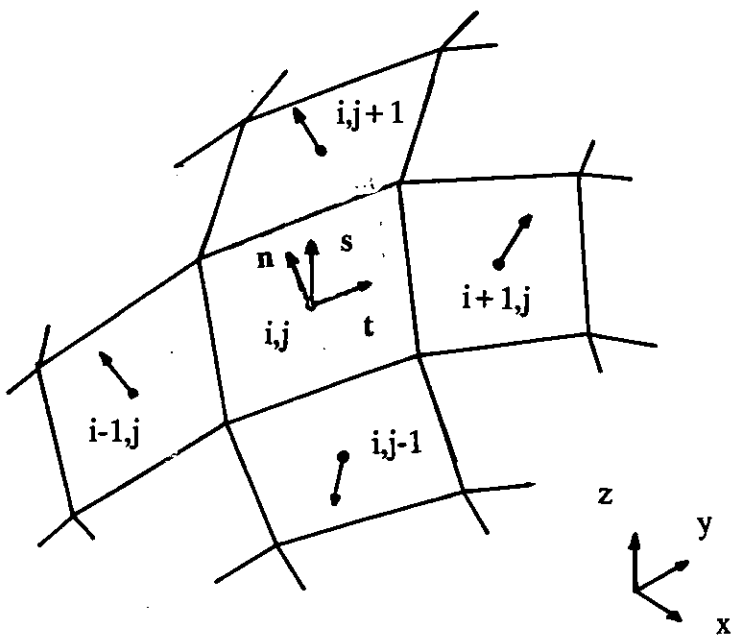


Figure 6 Detail of surface panel vectors for calculating surface velocity

panels. This is shown in figure 6, with the two unit vectors s and t in the local parametric directions. The velocity in the t and s directions are then obtained using a central difference:

$$u_t = \frac{(t_{i,j} - t_{i-1,j})(\Phi_{i+1,j} - \Phi_{i,j})}{(t_{i+1,j} - t_{i-1,j})(t_{i+1,j} - t_{i,j})} - \frac{(t_{i+1,j} - t_{i,j})(\Phi_{i,j} - \Phi_{i-1,j})}{(t_{i+1,j} - t_{i-1,j})(t_{i,j} - t_{i-1,j})} \qquad [46]$$

and

$$v_s = \frac{(s_{i,j} - s_{i,j-1})(\Phi_{i,j+1} - \Phi_{i,j})}{(s_{i,j+1} - s_{i,j-1})(s_{i,j+1} - s_{i,j})} - \frac{(s_{i,j+1} - s_{i,j})(\Phi_{i,j} - \Phi_{i,j-1})}{(s_{i,j+1} - s_{i,j-1})(s_{i,j} - s_{i,j-1})} \qquad [47]$$

As the finite difference is easier to implement and for panels which closely follow the curved surface of comparable accuracy it was chosen.

Having determined the surface velocity in the parametric coordinate system a transformation has to be carried out to give the surface velocity components in the overall coordinate system. Unit vector s and t are not necessarily orthogonal and therefore the velocities are first transformed into an orthogonal system with one direction normal to the panel. The u,v and w components can then be found. the combined expression as given by Lee[16] becomes:

$$\underline{U}_d = \frac{\frac{d\Phi}{dt}(t - (s.t)\,s) + \frac{d\Phi}{ds}(s - (s.t)\,t)}{||\,sxt\,||^2} \qquad [48]$$

Knowing the disturbance velocity U_d and hence total velocity U_t allows the local non-dimensional pressure coefficient Cp to be found.

$$Cp = 1 - \left|\frac{U_t}{U_\infty}\right|^2 \qquad [49]$$

The integration of the pressure distribution over the N panels defining the body surface allows the total body force **F** to be evaluated as a vector sum.

$$\boldsymbol{F} = \tfrac{1}{2}\rho U_\infty^2 \sum_{i=1}^{N} C_{Pi} A_i \boldsymbol{n} \qquad [50]$$

where Ai is the area of the ith panel and **ni** the direction of its unit surface normal.

The calculation of the pressure components of the non-dimensional body force and moment coefficients requires a further transformation into the correct body coordinate system. For example, as shown in figure 7 for a ship rudder at incidence with the x direction in the free-stream direction and z vertical the lift is the **j** component of **F**. That is:

$$C_L = \frac{\boldsymbol{F} \cdot \boldsymbol{j}}{\tfrac{1}{2}\rho U_\infty^2 s\bar{c}} \qquad [51]$$

where s and c are respectively the rudder span and meanchord. The pressure component of Drag is correspondingly:

$$C_D = \frac{\boldsymbol{F} \cdot \boldsymbol{i}}{\tfrac{1}{2}\rho U_\infty^2 s\bar{c}} \qquad [52]$$

An estimate of the viscous skin friction force acting on a lifting-surface can be found by using the panel surface velocity and distance from the leading edge to estimate the skin friction coefficient C_f. This gives a viscous force contribution equal to:

$$F_{visc} = \tfrac{1}{2}\rho \sum_{i=1}^{N} C_f A_i \left(V_T \cdot V_T \right) \boldsymbol{v} \qquad [53]$$

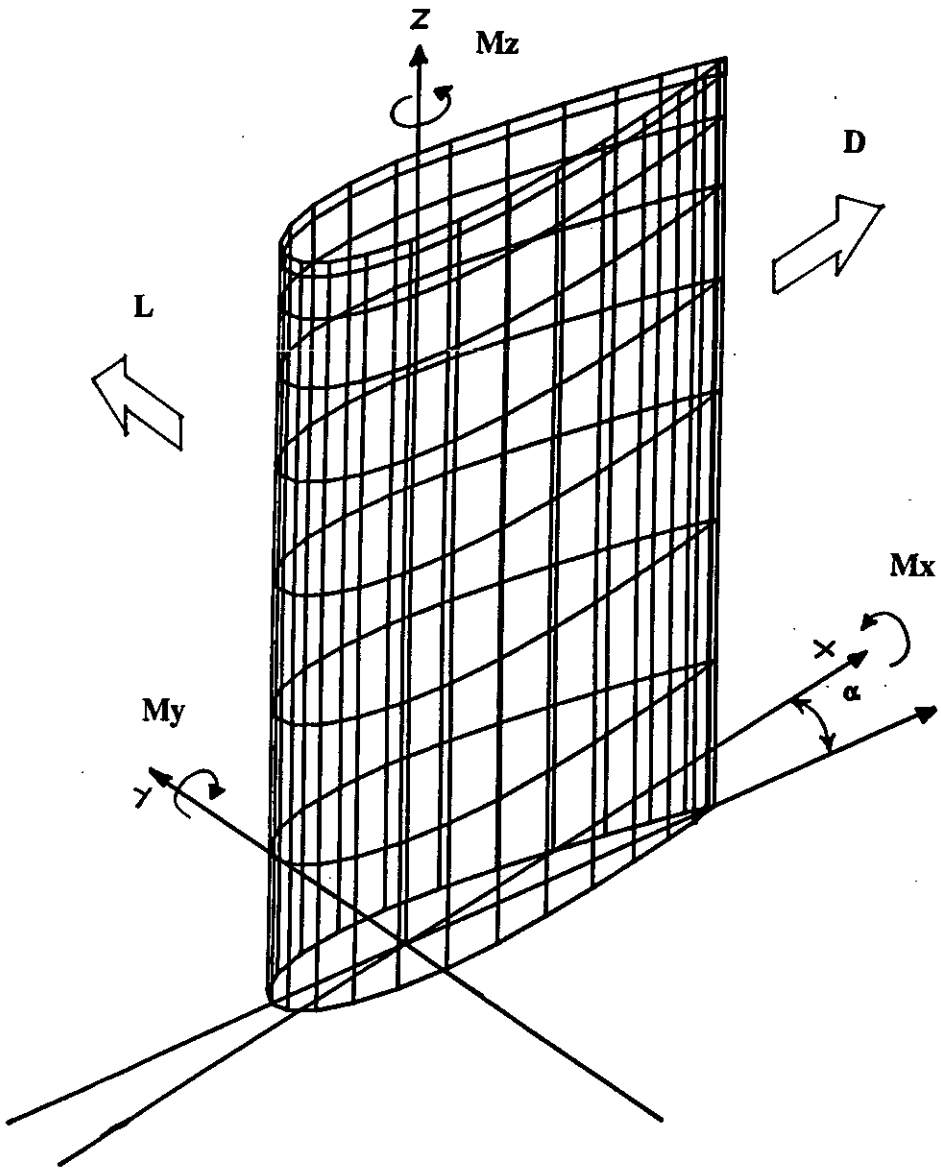


Figure 7 Direction of forces and coordinate system on a rudder at incidence

where **v** is a unit vector in the local flow direction. The skin friction coefficient is calculated in terms of local Reynolds number:

$$Rn = \frac{|U_\infty| s}{\nu} \qquad [54]$$

where s is the distance to the leading edge. The expressions used for C_f are from Schlicting[25]:

$$\begin{aligned} Rn &< 3x10^5 \quad C_f = 0.664Rn^{-0.5} \\ 3x10^5 \leq Rn &< 1x10^7 \quad C_f = 0.074Rn^{-0.2} - 1050Rn^{-1} \end{aligned} \qquad [55]$$

Combining the viscous and pressure contributions gives the total force $\mathbf{F_T}$ acting on the body as:

$$F_T = F + F_{visc} \qquad [56]$$

Similar expressions are derived to give the total moment acting about the body pivot **P**, where $\mathbf{L_i}$ is the position of the panel relative to the origin of the body axes.

$$M = \tfrac{1}{2}\rho U_\infty^2 \sum_{i=1}^{N} C_{Pi} A_i (L_i - P) xn \qquad [57]$$

$$M_{visc} = \tfrac{1}{2}\rho \sum_{i=1}^{N} C_f A_i (V_T \cdot V_T)(L_i - P) \times v \qquad [58]$$

$$M_T = M + M_{visc} \qquad [59]$$

9 VERIFICATION

9.1 Introduction

To verify the numerical implementation of Morino's method a number of trial geometries were tested. The geometries used are the same as those by Lee and allowed comparison both with Lee's work and the original source. The test cases presented are to verify the numerical analysis procedures and not the wake adaption method which will be considered in more detail in a later report.

It was not possible to carry out extensive sensitivity studies on these test cases, for instance on panelling density, due to a maximum limit of 400 on the number of panels during development of the code. However, it was considered that if the results

replicated those by Lee then such tests were unnecessary.

9.2 Ellipsoid

Lamb[14] gives an analytical solution for the surface potential of an ellipsoid geometry. for a non-lifting configuration (symmetrical flow) a direct comparison of the potential found using the numerical method and the analytical solution can be made.

The ellipsoid used for this test has a surface defined by:

$$x^2 + y^2 + c^2z^2 = 1 \qquad [60]$$

where the value of c was taken as 0.1. The perturbation potential on ellipsoid surface given by Lamb is:

$$\phi(x,y,z) = U_\infty xK \qquad [61]$$

where

$$K = \frac{c}{2-\alpha_o} D \qquad [62]$$

$$D = \int_0^\infty \frac{d\lambda}{(1+\lambda)^{\frac{3}{2}}(1+\lambda)^{\frac{1}{2}}(c^2+\lambda)^{\frac{1}{2}}} \qquad [63]$$

$$\alpha_o = cD \qquad [64]$$

Figure 8 shows the geometric representation of the ellipsoid which had 32 chordwise and 12 spanwise panels with a cosine and sine panel distribution respectively. The sinusoidal panel distributions concentrates the panels in the areas of large variation of surface curvature and potential. In figure 9 the numerical potential for four spanwise strips is compared with the corresponding theoretical value. They are plotted to a base

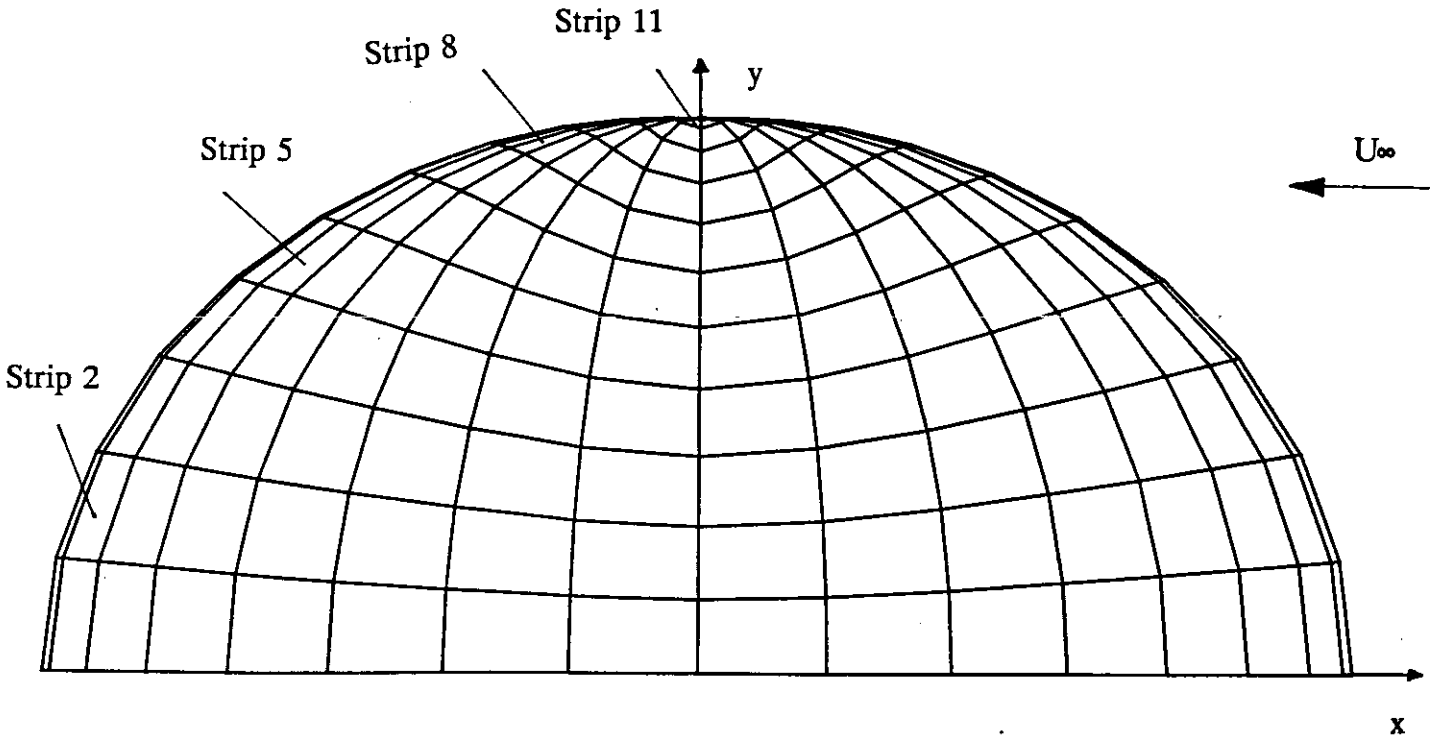


Figure 8 Panelling arrangement fo an ellipsoid

of normalised parametric length of each strip. The potential is normalised with respect to the free-stream velocity and ellipsoid radius. It can be seen that the comparison is very good although in the region of the tip there are small discrepancies. This was found be Lee who noted that the discrepancies were magnified when the surface velocity is calculated but the effect was localised at the tip.

9.3 Circular wing

Jordan [24] gives an analytic solution to the flow over a circular planform wing of zero thickness. By comparing numerical solutions at 1% and 5% thickness/chord ratio scetions for a NACA 4 digit series section with that of the analytic solution an assessment can be made on the prediction of lift for an aerofoil at incidence. As noted by Lee, this geometry represents the flow in the region of a propeller tip. The panelling arrangement used was similar to that for the ellipsoid. The limit on panel numbers restricted the accuracy obtainable (a higher number of spanwise strips is required to

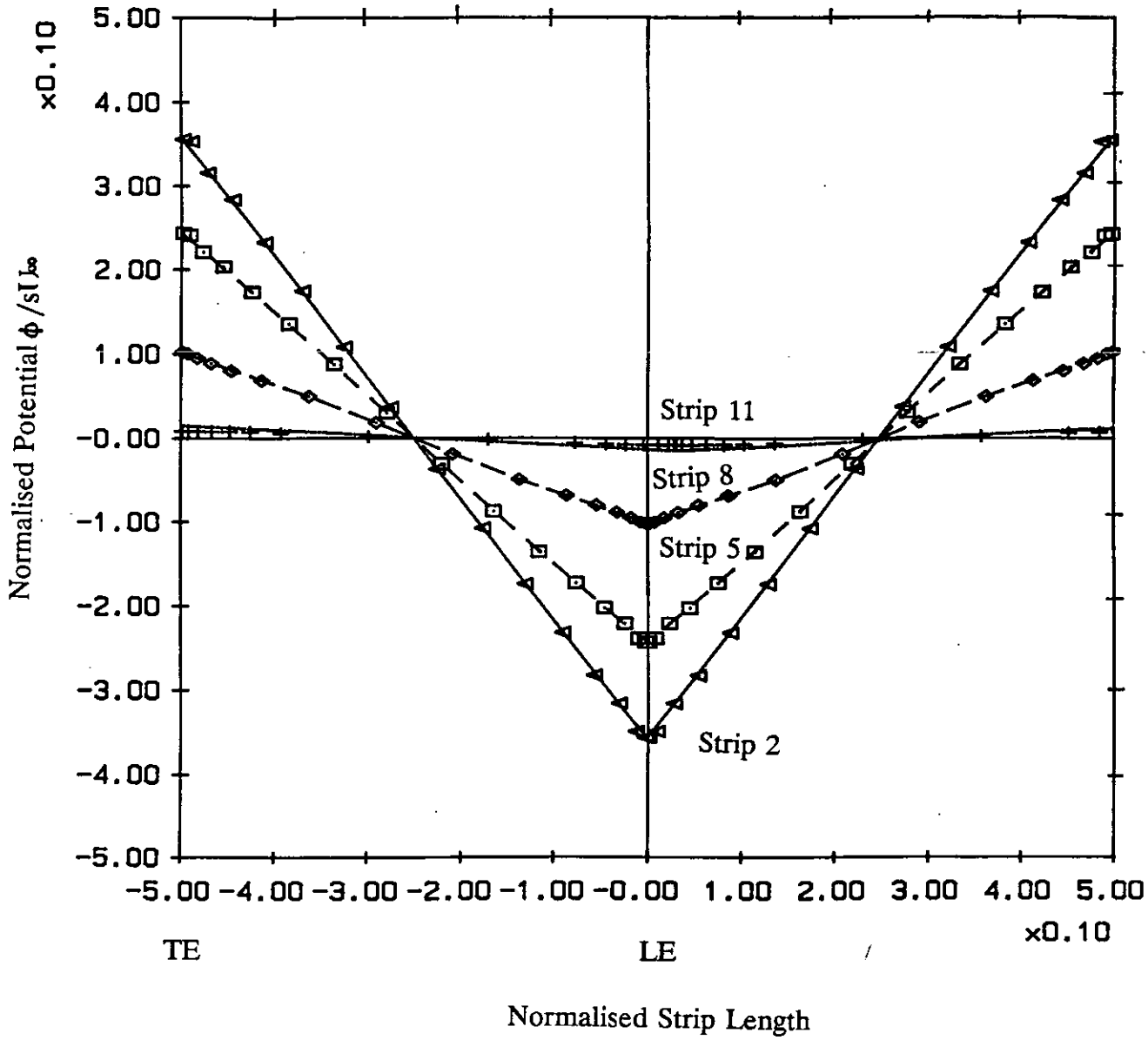


Figure 9 Comparison of theoretical and numerical potential for an ellipsoid with a=1, b=1, and c=0.1

achieve the converged panel density of Lee). However, as shown in figure 10, there is a good comparison of chordwise pressure distribution for spanwise strip 9 for the 1% aerofoil between the analytic solution and the numerical result run with and without the iterative Kutta condition. The importance of the application of the explicit Kutta condition can be seen in removing the spurious trailingedge pressure loading.

The effect of wing thickness in comparing the spanwise circulation distribution between

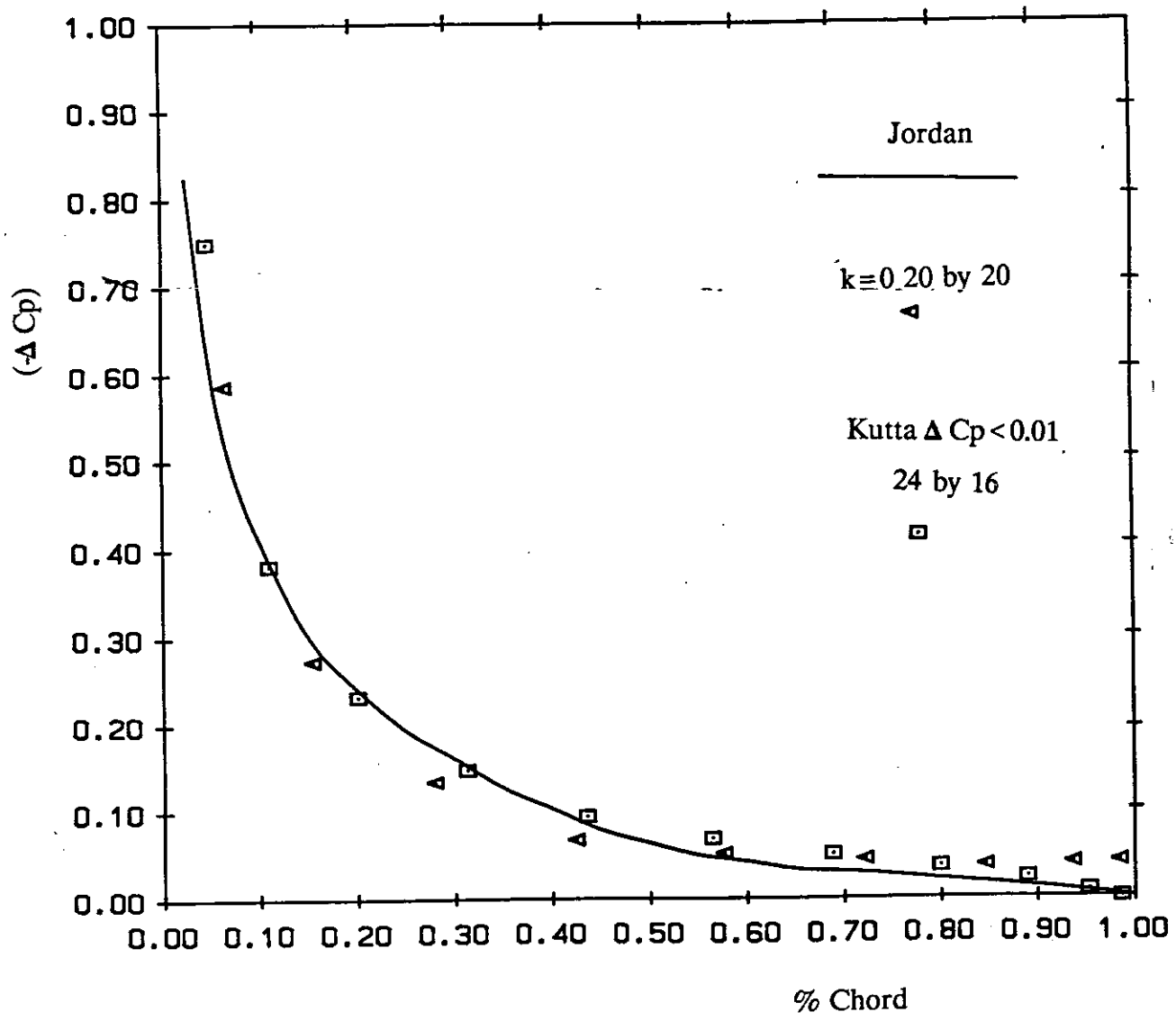


Figure 10 Chordwise pressure distribution for cicular planform wing at 91% semi-span at 0.1 radian incidence

the analytic (0% t/c) with the numeric solution (1% t/c, and 5% t/c) is shown in figure 11. The result for the 5% and 1% distributions are close to the analytic solution with the 5% distribution slightly further away. Overall, the results for the circular wing show good correspondence with the analytic solution even with the limited number of panels. The necessity of applying the explicit kutta condition is demonstrated for geometries with

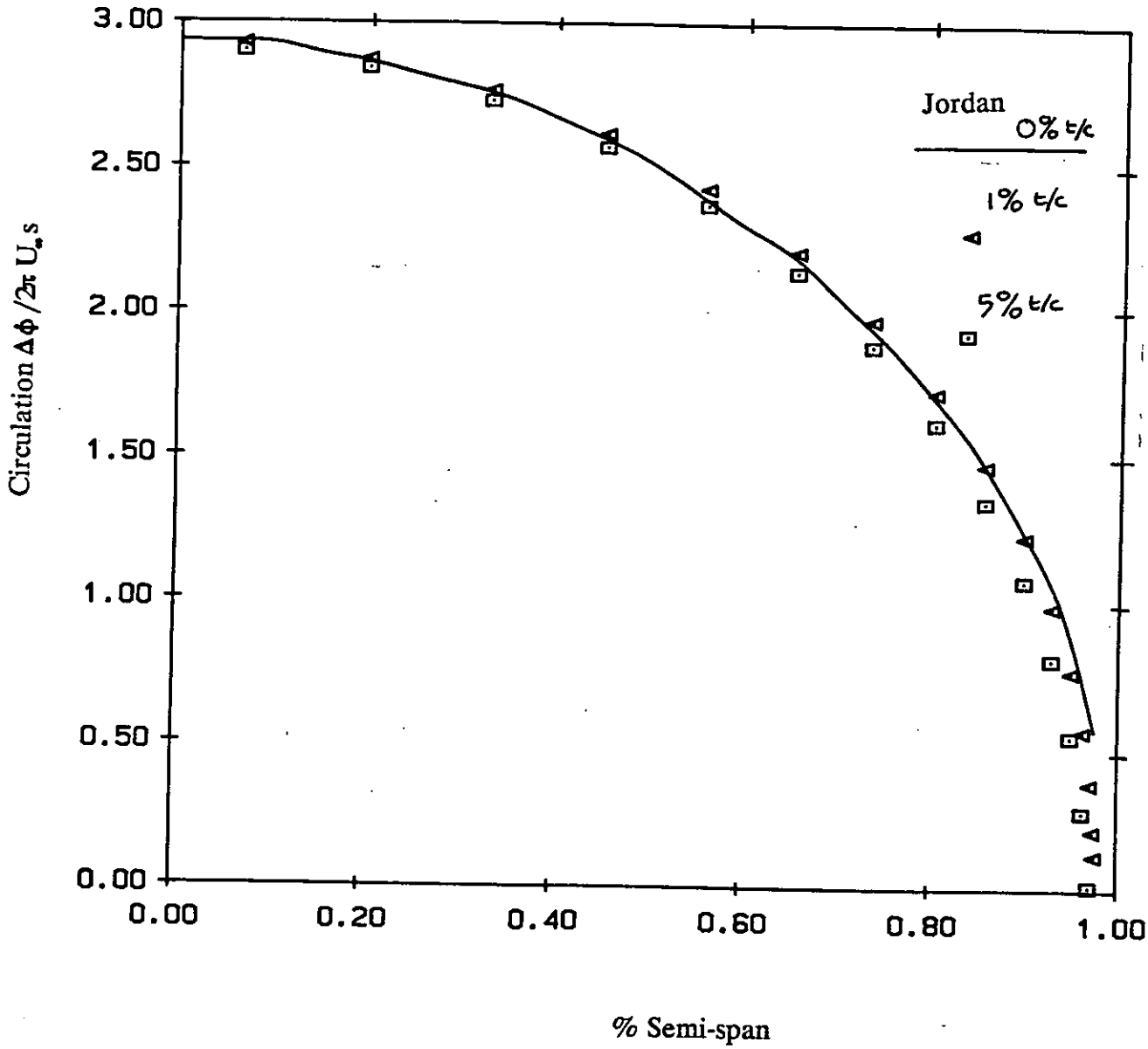


Figure 11 Spanwise circulation distribution for circular planform wing at 0.1 radian incidence

significant cross-flow components.

9.4 NACA 0012 unswept wing

To demonstrate the behaviour of the method for high aspect-ratio lifting surfaces a case

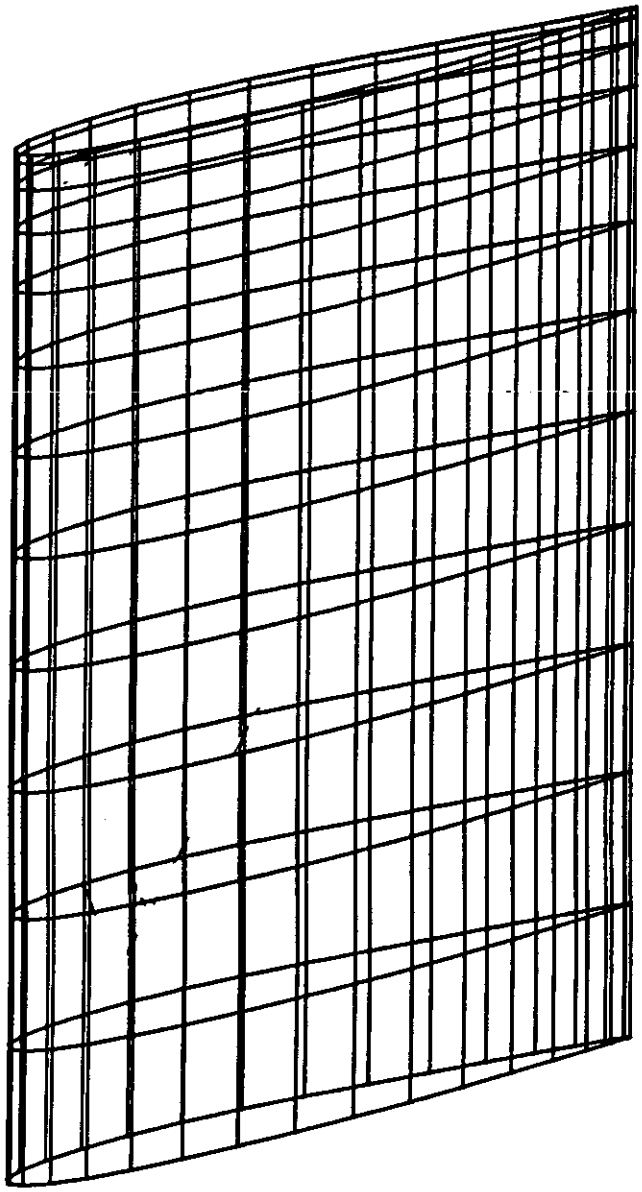


Figure 12 Isometric wire-frame plot of half span of NACA0012 unswept wing with AR = 5.95

was run for a NAC0012 wing with an effective aspect ratio of 5.9 at an incidence of 8°. Two panel arrangements of 50 by 8 and 33 by 12 chordwise by spanwise panels were tried. The panelling geometry of the later scheme is shown as an isometric wireframe plot in figure 12. Again sinusoidal clustering of panels was used.

In figure 13 a comparison is made of the spanwise circulation distribution obtained by Lee and the two panelling arrangements tested. There is good agreement over the whole

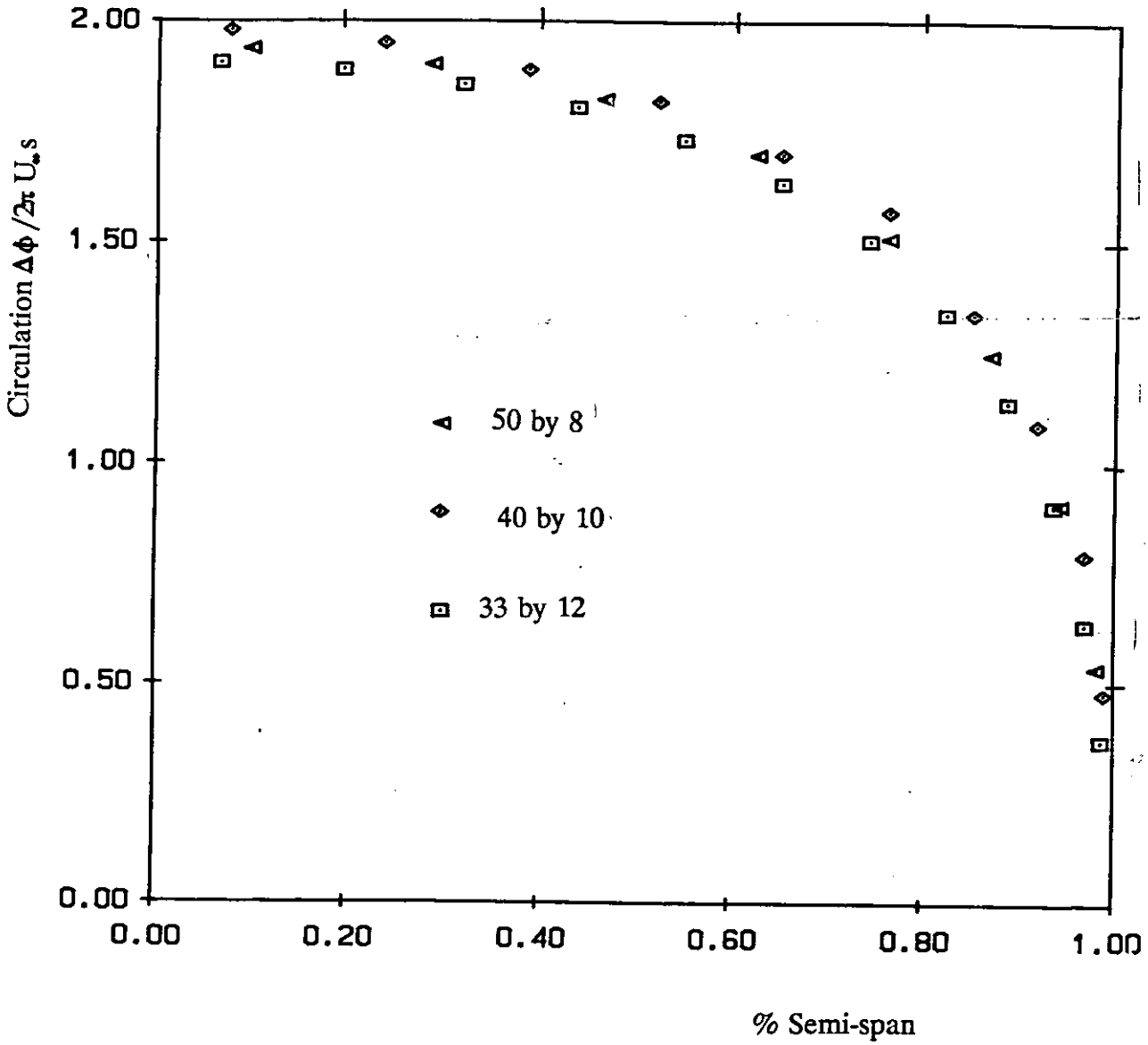


Figure 13 Spanwise circulation distribution for NACA0012 unswept wing at 8° incidence

span. This agreement is also illustrated in figure 14 which gives the chordwise presure distribution at x% semi-span. The area under the three sets of data is almost identical and close correspondence in both the leading and trailing edge regions.

A final comparison, figure 15, demonstrates how the method predicts the

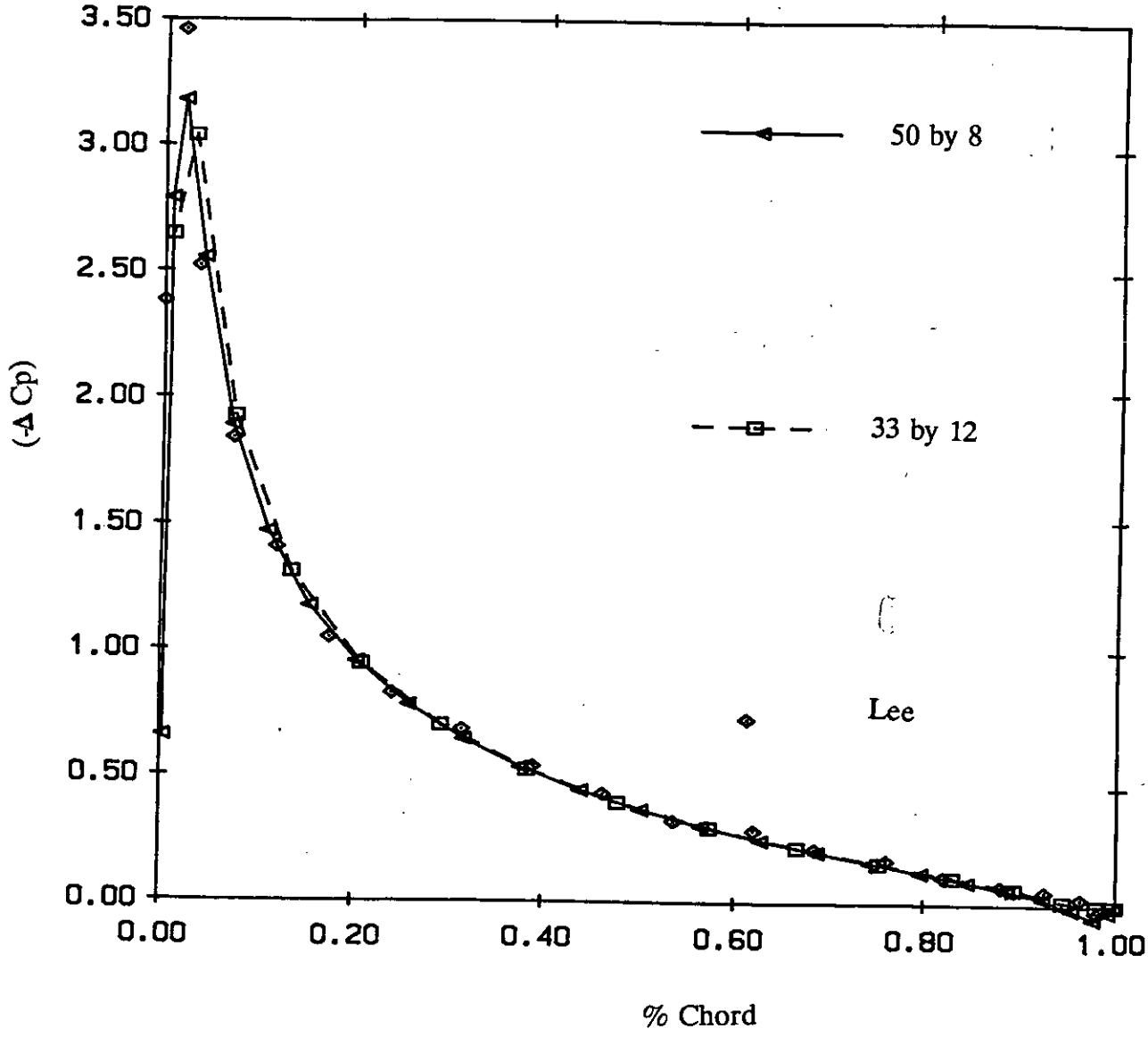


Figure 14 Chordwise pressure distribution for NACA0012 unswept wing at 8° incidence at 64% semi-span.

experimentally measured spanwise local lift distribution of the wing. The lifting surface fails to pick up the slight drop in lift coefficient towards the wing centre-line (probably due to boundary layer effects) and the holding up of lift right at the tip (tip vortex effects see Molland[9]) but overall there is a very good correlation which indicates the suitability of the method for modelling both ship rudders and propellers.

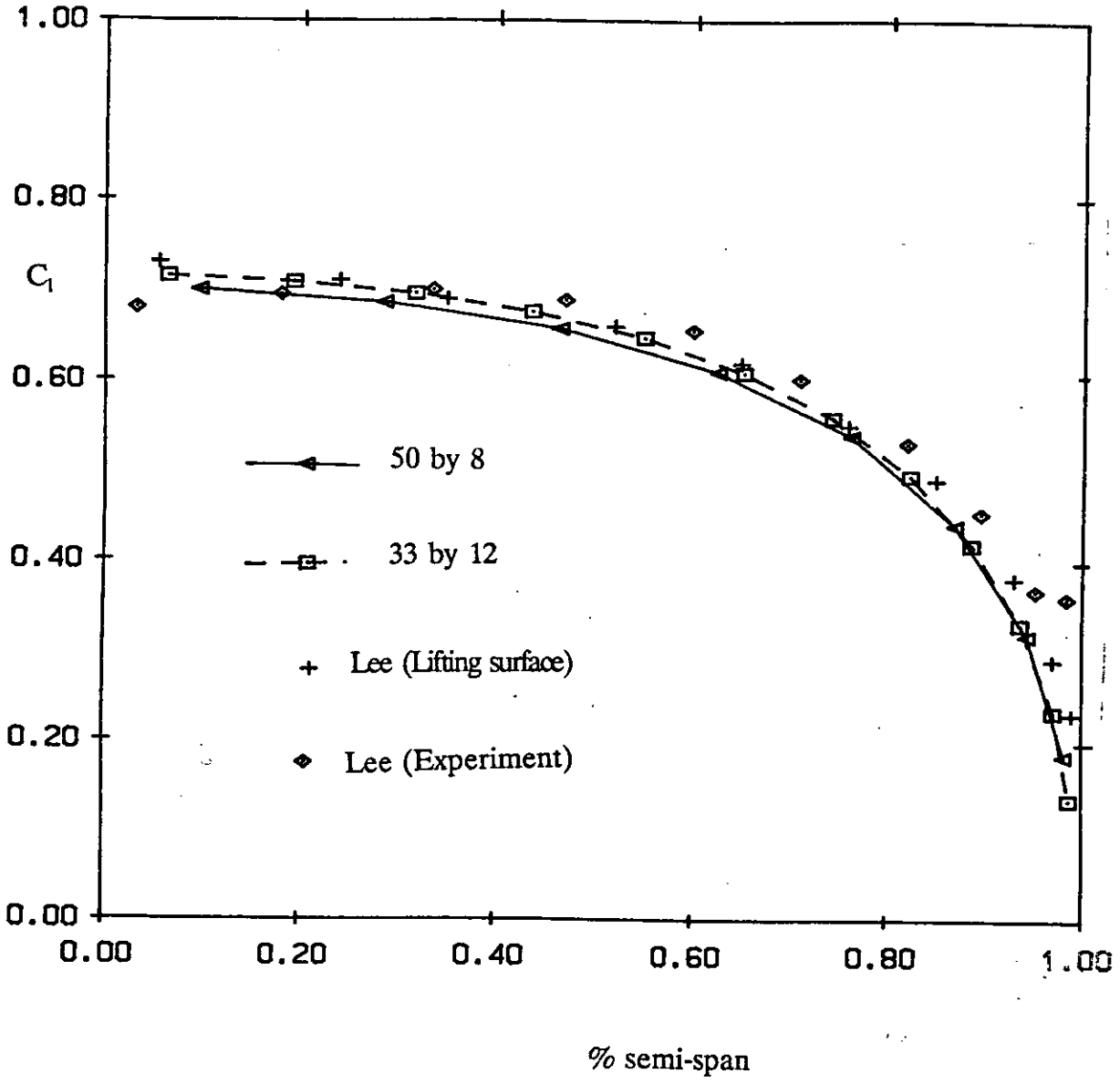


Figure 15 Local lift coefficient distribution for NACA0012 unswept wing at 8° incidence

10 CONCLUSION

A robust lifting surface scheme has been developed based on Lee's implementation of Morino's perturbation potential method. The procedure has been verified with a wide range of representative non-rotational lifting surfaces. The method is suitable for both rotational and non-rotating bodies.

The verification confirmed that the method is suitable for modelling both ship rudder geometries and ship propeller geometries.

An explicit trailing edge pressure kutta condition was found essential to remove spurious trailing edge pressure loading.

A method of surface definition using parametric cubic splines provides a flexible and rapid means of defining complex multi-body geometries.

Surface pressure/velocity distribution information can be obtained from the numerical solution and this allows total body force and moment coefficients to be evaluated.

The formulation implemented allows three-dimensional velocity field information to be obtained. This can be used in defining the inflow velocity field necessary to implement the interaction velocity field necessary for modelling rudder-propeller interaction.

ACKNOWLEDGEMENTS

The work described in this report covers part of a research project funded by the S.E.R.C./M.o.D. through the Marine Technology Directorate Ltd. under research grant Ref No GR/E/65289.

REFERENCES

[1] Cho. J, & Williams, M., "Propeller-Wing Interaction using a Frequency Domain Panel Method", Journal of Aircraft, Vol. 27, No. 3, March 1990.

[2] Kleinstein, G, & Liu, C.H., " Application of airfoil theory for non-uniform streams to wing propeller interaction", Journal of Aircraft, Vol. 9, No. 2, Feb. 1972.

[3] Koening, C., " Influence of the propeller on other parts of the airplane structure", Aerodynamic Theory, Vol IV, ed. Durand, W.F., Dover, 1963.

[4] Kroo, I., "Propeller-wing integration for minimum induced loss", Journal of Aircraft, Vol. 23, No. 7, July 1986, pp. 561-565.

[5] Munk, M., "Minimum induced drag of airfoils", NACA Rept. 121, 1921.

[6] Witkowski, D.P, Lee, A.K.H., & Sullivan, J.P, "Aerodynamic interaction between propellers and wings", Journal of Aircraft, Vol. 26, No. 9, Sept. 1989, pp. 829-836.

[7] Lorber, P.F., and Egolf, T.A., " An unsteady helicopter rotor-fuselage aerodynamic interaction analysis", pp. 32-42, Journal of the American Helicopter Society, July 1990.

[8] Hess, J.L., & Smith, A.M.O.," Calculation of non-lifting potential flow around arbitrary three-dimensional bodies", Journal of Ship Research, Vol. 8, No. 2, Sept. 1964.

[9] Molland, A.F., "A method for determining the free-stream characteristics of ship skeg-rudders", International Shipbuilding Progress, Vol 32, June 1985.

[10] Kerwin, J.E., & Lee, C-S, "Prediction of steady and un-steady marine propeller performance by numerical lifting surface theory", SNAME Transactions, Vol. 86, 1978, pp. 218-253.

[11] Kerwin, J.E., Kinnas, S.A., Lee, J-T, Shih, W-Z, " A surface panel method for the hydrodynamic analysis of ducted propellers", SNAME Transactions, Vol. 95, 1987.

[12] Greeley, D.S., & Kerwin, J.E., "Numerical methods for propeller design and analysis in steady flow", SNAME Transactions, Vol. 90, 1982, pp. 415-453.

[13] Hess. J.L., "Panel Methods in Computational Fluid Dynamics", Annual review of fluid mechanics. 1990. Vol 22. pp.255-274.

[14] Lamb, H.,"Hydrodynamics", Cambridge University Press, sixth edition, 1932.

[15] Hess, J.L., "The problem of three-dimensional lifting flow and its solution by means of a surface singularity distribution", Computational Methods Applied Mechanical Engineering, 4:283-319, als0 1972, Rep. MDC-J5679, McDonnell Douglas Aircraft Co. Long Beach, Calif.

[16] Lee, J-T, "A potential based method for the analysis of marine propellers in steady flow", Ph.D. thesis, M.I.T. Dept. of Ocean Engineering, Aug. 1987.

[17] Morino, L., & Kuo, C-C, " Subsonic Potential aerodynamics for Complex Configurations: A general theory", A.I.A.A. Journal, Vol 12., No. 2 Feb. '74.

[18] Maskew, B., "Prediction of subsonic aerodynamic characteristics: A case for low-order panel methods", Journal of Aircraft, Vol 19, No. 2, Feb. 1982, pp. 157-163.

[19] Margesson, R.J., Kjelgaard, S.O., Sellers, W.L., Morris, C.E.K., Walkley, K.B., & Shields, E.W.," Subsonic panel methods - a comparison of several production codes", Proceedings of AIAA 23rd Aerospace Science Meeting, Reno, Nevada, Jan. 1985.

[20] Newman, J.N.,"Distribution of sources and normal dipoles over a quadrilateral panel". Journal of Engineering Mathematics,Vol 20., pp113-126, 1986.

[21] Maitre, T.A., and Rowe, A.R., "Modelling of flow around a propeller using a potential based method", Journal of Ship Research, Vol 35., No. 2, June 1991, pp.114-126.

[22] Kreyszig, E., "Advanced Engineering Mathematics". pp. 781, 5th edition, John Wiley & Sons, 1983.

[23] Schlichting, H., " Boundary Layer Theory", McGraw-Hill, New York 1968.

[24] Jordan, P.F., "Exact solutions for lifting surfaces", AIAA Journal, Vol 11., No. 8, Aug. 1973, pp. 1123-1129.

APPENDIX Calculation of panel potential/velocity influence coefficients

(i) Introduction

Newman[20] gave the potential due to a distribution of sources or normal dipoles on a flat quadrilateral panel. The derivation given for the normal dipole potential is considered to be superior to that of Hess and Smith[8] and is valid for panel's whose panel density is of an arbitrary polynomial form. However, in this work only the constant strength panel case is used.

An efficient scheme for the calculation of the influence coefficient of a panel for an arbitrary field point uses a graded series of far-field approximations to reduce computational time. Newman also gave an arbitrary order multipole expressions for source and normal dipole potential. The scheme used in this work for choosing the relevant expression is that of Newman and is based on the ratio L of the distance between the panel centre R to the size of the largest diagonal of the panel. The value of L for the various approximations is:

(1)		L < 2.0	Exact expression.
(2)	2.00 ≤	L < 2.45	4^{th} Order Multipole.
(3)	2.45 ≤	L < 4.0	2^{nd} Order Multipole.
(4)	4.00 ≤	L	Point

In this work expressions are also needed for the source and normal dipole velocity influence of of a flat quadrilateral panels. These are used in the wake adaption scheme and the calculation of the interaction velocity field. The velocity influence were obtained by differentiation of the expressions for the source and dipole potential given by Newman.

(ii) Panel Geometry

Figure A-1 illustrates an arbitrary quadrilateral panel located in an overall cartesian coordinate system (x',y',z').

Following the method of Hess and Smith[8], the coordinate centre **C** of the arbitrary panel is defined as the average position of the panel nodes P_i:

$$C = \frac{1}{N}\sum_{i=1}^{N} P_i \qquad [A.1]$$

A panel centroid coordinate system is defined using the panel diagonals $\mathbf{a} = \mathbf{P}_3 - \mathbf{P}_1$ and $\mathbf{d} = \mathbf{P}_4 - \mathbf{P}_2$. The unit normal **n** is then:

$$n = \frac{a \times d}{|a \times d|} \qquad [A.2]$$

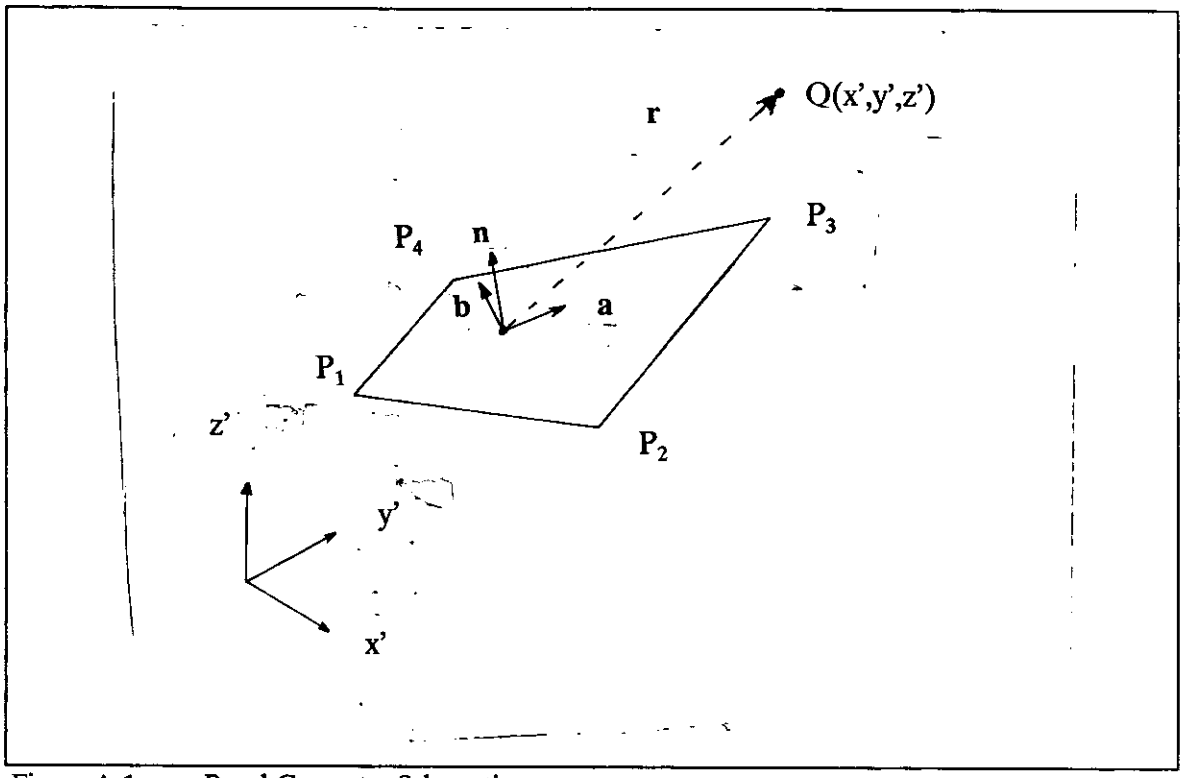


Figure A-1 Panel Geometry Schematic

and the panel area A = ½ | **a** x **d** |. Once **n** is defined the third vector of the panel orthonormal system is found using the unit vector in the direction of a.

$$b = \frac{n \times a}{|a|} \qquad [A.3]$$

To obtain a flat panel, which is necessary for the analysis, the position of each node is adjusted so that they are all coplanar with the panel centre C using:

$$For\ i = 1\ to\ 4, \qquad P'_i = P_i + \left(\left(C - P_i \right) \cdot n \right) n \qquad [A.4]$$

The orthonormal system matrix [B] = (**a**,**b**,**n**) is then used to transform the panel nodes to transform the panel nodes into the panel centred coordinates on the plane z=0.

$$P''_i = [B]\left(P_i - C \right) \qquad [A.5]$$

A further translation is applied to locate the centre of the panel coordinate system at the centroid of the panel.

Figure A-2 illustrates the planar representation after the transformation to the panel coordinate system, with ξ, η define the position of the panel nodes and s the length between nodes.

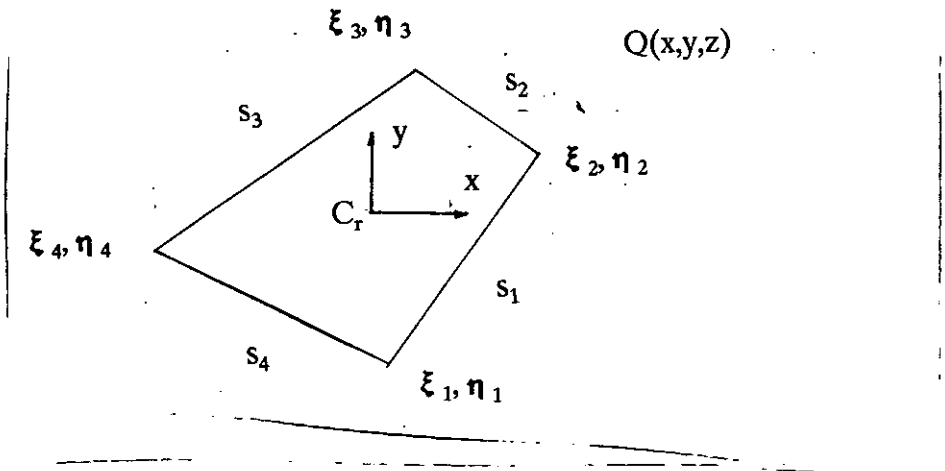


Figure A-2 Schematic of panel in panel coordinate system with origin at centroid.

To find the influence of the at a particular field point Q(x,y,z) the first step is to translate the point into the individual panel coordinate system:

$$Q' = [B]\left(Q - P_{cr}\right) \qquad [A.6]$$

where P_{cr} is the panel centroid in the overall coordinate system. The radial distance from the centroid to Q is then simply the magnitude of Q'.

As the expression for velocity influence coefficient V is calculated within the panel coordinate system a final transformation using the transpose of [B] has to be applied to find the velocity influence in the overall system:

$$V^{o} = [B]^{T} V^{p} \qquad [A.7]$$

(iii) Exact

Dipole potential

The potential at Q for a normal dipole distribution of constant strength -4π is an integral over the panel surface:

Newman's derived expression is simply the sum of four arctangents for a quadrilateral panel, with coefficients

$$\Phi = z \iint [(x-\xi)^2 + (y-\eta)^2 + z^2]^{\frac{-3}{2}} d\xi \, d\eta \qquad [A.8]$$

based on the geometric properties of the sides and does not require any numerical integration and is:

$$\Phi = \sum_{n=1}^{4} \tan^{-1}\left(\frac{S_{3n}}{C_{3n}} \right) \qquad [A.9]$$

where $C_3 = S_1C_2 - S_2C_1$ and $C_3 = C_1C_2 + S_1S_2$.

The four geometric coefficients are:

$$S_1 = \delta\eta_n [(x-\xi_n)^2 + z^2] - \delta\xi_n (x-\xi_n) (y-\eta_n) \qquad [A.10]$$

$$S_2 = \delta\eta_n [(x-\xi_{n+1})^2 + z^2] - \delta\xi_n (x-\xi_{n+1}) (y-\eta_{n+1}) \qquad [A.11]$$

$$C_1 = R_n z \delta\xi_n \qquad [A.12]$$

$$C_2 = R_{n+1} z \delta\xi_n \qquad [A.13]$$

where R_n is the radial distance between Q and node n, $\delta\eta_n = \eta_{n+1} - \eta_n$, and $\delta\xi_n = \xi_{n+1} - \xi_n$.

Dipole velocity

The velocity influence coefficient of a constant normal dipole distribution of strength -4π at field point Q is:

$$V = -\nabla\Phi \qquad [A.14]$$

Differentiating the expression for dipole potential influence gives:

$$V_d = -\sum_{n=1}^{4} \frac{1}{S_3^2 + C_3^2 + 1} ((C_1C_2^2 + S_2^2C_1) \nabla S_1 + (S_1^2S_2 + S_2C_1^2) \nabla C_2 - (C_1^2C_2 + S_1^2C_2) \nabla S_2 - (S_1C_2^2 + S_1S_2^2) \nabla C_1) \qquad [A.15]$$

where

$$\nabla S_1 = \begin{bmatrix} \delta\eta_n (x-\xi_n) - \delta\xi_n (y-\eta_n) \\ -\delta\xi_n (x-\xi_n) \\ \delta\eta_n z \end{bmatrix} \qquad [A.16]$$

$$\nabla S_2 = \begin{bmatrix} \delta\eta_{n+1} (x-\xi_{n+1}) - \delta\xi_n (y-\eta_{n+1}) \\ -\delta\xi_n (x-\xi_{n+1}) \\ \delta\eta_n z \end{bmatrix} \qquad [A.17]$$

$$\nabla C_1 = \begin{bmatrix} \dfrac{z\delta\eta_n}{R_n} \\ \dfrac{z\delta\xi_n}{R_n} \\ R_n\delta\xi_n + \dfrac{z^2\delta\xi_n}{R_n} \end{bmatrix} \qquad [A.18]$$

$$\nabla C_2 = \begin{bmatrix} \dfrac{z\delta\eta_n}{R_{n+1}} \\ \dfrac{z\delta\xi_n}{R_{n+1}} \\ R_{n+1}\delta\xi_n + \dfrac{z^2\delta\xi_n}{R_{n+1}} \end{bmatrix} \qquad [A.19]$$

In the plane z = 0 the expression for V_d reduces to:

$$V_d = -\sum_{n=1}^{4} \begin{bmatrix} 0 \\ 0 \\ \dfrac{\delta\xi_n \left(S_1 R_{n+1} - S_2 R_n \right)}{S_1 S_2} \end{bmatrix} \qquad [A.20]$$

Source potential

The potential at field point Q of a constant source distribution of strength -4π is a surface integral:

$$\Psi = \iint \frac{\delta\xi\, \delta\eta}{r} \qquad [A.21]$$

as $\Phi = -d\Psi/dz$ and since Φ and Ψ vanish at infinity partial integration gives that:

$$\Psi = \int_z^\infty \Phi \, dz = -\int_z^\infty z \, d\Phi - z\Phi \qquad [A.22]$$

Newman evaluated the integral terms to find the expression for Ψ as:

$$\Psi = \sum_{n=1}^{4} v_n Q_n - z\Phi \qquad [A.23]$$

where

$$v_n = (x - \xi_n)\sin(\theta_n) - (y - \eta_n)\cos(\theta_n) \qquad [A.24]$$

$$Q_n = \log\left|\frac{R_n + R_{n+1} + s_n}{R_n + R_{n+1} - s_n}\right| \qquad [A.25]$$

and for the angle θ_n

$$\sin\theta_n = \frac{\delta\eta_n}{s_n}$$
$$\cos\theta_n = \frac{\delta\xi_n}{s_n} \qquad [A.26]$$

Source velocity

The velocity influence of a constant source distribution of strength -4π at field point Q is:

$$V_s = -\nabla\Psi \qquad [A.27]$$

but

$$\nabla\Psi = \nabla\left(\sum_{n=1}^{4} v_n Q_n - z\Phi\right)$$
$$= \sum_{i=1}^{4} \nabla v_n Q_n - \Phi n - z V_d \qquad [A.28]$$

The terms of Φ_n and zV_d are already known, after manipulation $\nabla v_n Q_n$ can be expressed as:

$$\nabla v_n Q_n = Q_n \begin{bmatrix} \sin\theta_n \\ \cos\theta_n \\ 0 \end{bmatrix} - \nabla\left(R_n + R_{n+1}\right) \frac{2s_n}{\left(R_n + R_{n+1}\right)^2 - s_n^2}\left[(x-\xi_n)\sin\theta_n - (y-\eta_n)\cos\theta_n\right] \quad [A.29]$$

but

$$\nabla\left(R_n + R_{n+1}\right) = \left(\frac{1}{R_n} + \frac{1}{R_{n+1}}\right)\left[\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} \frac{\xi_n}{R_n} + \frac{\xi_{n+1}}{R_{n+1}} \\ \frac{\eta_n}{R_n} + \frac{\eta_{n+1}}{R_{n+1}} \\ 0 \end{bmatrix}\right] \quad [A.30]$$

(iv) Multipole

An appropriate far-field expression for the source Ψ can be found by expanding the integral expression for Ψ as a Taylor series:

$$\Psi = \iint \frac{d\xi\, d\eta}{r} = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \frac{(-)^{n+m}}{m!n!} I_{mn} \frac{\partial^{m+n}}{\partial x^m \partial y^n} \frac{1}{\sqrt{x^2 + y^2 + z^2}} \quad [A.31]$$

where I_{mn} is the moment of the panel about the origin:

$$I_{mn} = \iint \xi^m \eta^n d\xi\, d\eta \quad [A.32]$$

Similarly for a normal dipole distribution:

$$\Phi = \iint \frac{z d\xi\, d\eta}{r^3} = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \frac{(-)^{n+m}}{m!n!} I_{mn} \frac{\partial^{m+n+1}}{\partial z \partial x^m \partial y^n} \frac{1}{\sqrt{x^2 + y^2 + z^2}} \quad [A.33]$$

The recursive relationships defined in Newman were used to calculate up to the 4th order moments and these 13 values were calculated during the process of setting up a panel.

The various order partial differentials of the radius vector **r** were obtained as were the expressions for calculating the multipole expressions for V_s and V_d:

$$\nabla \frac{\partial^{m+n}}{\partial x^m y^n}\left(\frac{1}{r}\right) \qquad [A.34]$$

and

$$\nabla \frac{\partial^{m+n+1}}{\partial z \partial x^m y^n}\left(\frac{1}{r}\right) \qquad [A.35]$$

These expressions are not given here as their evaluation is straightforward if tedious.

(v) Point

The zero order multipole expression gives the influence coefficient of the panel as a point with the panel area (I_{00}). That is:

$$\Psi = \frac{A}{r}$$
$$\Phi = \frac{zA}{r^3} \qquad [A.36]$$

and

$$V_s = -\nabla\Psi = \frac{A}{r^3} r \qquad [A.37]$$

$$V_d = -\nabla\Phi = \frac{3Az}{r^5} r - \frac{A}{r^3}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad [A.38]$$

(vi) Implementation

Two procedures were written in occam2 which:

(1) SetUpSourcePanel, calculates all the geometric parameters and moments for a quadrilateral panel;

(2) NewmanPanel, evaluates the dipole potential and source potential, or the dipole velocity and source

velocity influence coefficients.

These procedures were written using 3-component vector processes so that their function is transparent.

(vii) Code Listing

SetUpSourcePanel

```
PROC SetUpSourcePanel(VAL [3]REAL32 P1,P2,P3,P4,
                      [3][3]REAL32 A,At,
                      [4]REAL32 E,N,
                      [3]REAL32 Pcr,n,
                      [4]REAL32 s,dE,dN,
                      [5][5]REAL32 Imn,
                      REAL32 BigDiag,ScaleUp)
  [3]REAL32 a,b,Po,E1,Pc,E2,E3,E4,C1,C2,C3,C4,Cr:
  #USE snglmath
  REAL32 sze,dk,diff,sz2,area:
  SEQ

    VectorSub(P3,P1,a)
    VectorSub(P4,P2,b)

    CrossProduct(a,b,n)
    Magnitude(n,area)
    ScalarDiv(area,n)

    Magnitude(a,sze)
    Magnitude(b,sz2)

    IF
      sz2>sze
        BigDiag:=sz2 -- *ScaleUp
      TRUE
        BigDiag:=sze -- *ScaleUp


    ScalarDiv(sze,a)
    CrossProduct(n,a,b)


    FindAverage(P1,P2,P3,P4,Po)
    MoveToSamePlane(P1,P2,P3,P4,Po,n,E1,E2,E3,E4)
    DefineAMatrix(a,b,n,A,At)
    ChangeCoordinates(E1,E2,E3,E4,Po,A,C1,C2,C3,C4)
    FindCentroid(C1,C2,C3,C4,Cr)
    MoveToCentroid(Cr,C1,C2,C3,C4,E,N)
    FindDifferences(E,N,s,dE,dN)
    FindMoments(E,N,s,dE,dN,Imn)
    TranslateCoord(Cr,At,Pc)
    VectorAdd(Pc,Po,Pcr)
:
```

NewmanPanel

```
--{{{ Newman panel

PROC NewmanPanel(VAL BOOL Both,IsVel,
                 VAL [4]REAL32 E,N,s,dE,dN,
```

```
            VAL [3][3]REAL32 A,At,
            VAL [3]REAL32 Pcr,
            VAL [5][5]REAL32 Imn,
            VAL REAL32 BigDiag,ScaleUp,
            VAL [3]REAL32 P,
            [3]REAL32 Qs,Qd)
[3]REAL32 Pen,Pt,Qsen,Qden:
REAL32 choiceparameter,radius,areais:
SEQ
  VectorSub(P,Pcr,Pt)
  TranslateCoord(Pt,A,Pen)
  Magnitude(Pen,radius)
  choiceparameter:=radius/BigDiag
  IF
    choiceparameter>4.0(REAL32)
      --{{{  point source
      IF
        IsVel=TRUE
          SEQ
            --{{{  point source
            REAL32 mags,magsS:
            SEQ
              magsS:=(-Imn[0][0])/((radius*radius)*radius)
              IF
                Both=TRUE
                  SEQ
                    ScalarMultiply(magsS,Pen,Qsen)
                    ScalarMul((0.0(REAL32)-0.079577471(REAL32)),Qsen)
                TRUE
                  SEQ w=0 FOR 3
                    Qs[w]:=0.0(REAL32)
              -- dipole potential
              mags:=((3.0(REAL32)*Imn[0][0])*Pen[2])/
                    ((radius)*((radius*radius)*(radius*radius)))
              ScalarMultiply(mags,Pen,Qden)
              Qden[2]:=Qden[2]+(magsS)
              ScalarMul((0.0(REAL32)-0.079577471(REAL32)),Qden)
            --}}}
        TRUE
          SEQ
            --{{{  point source
            SEQ
              IF
                Both=TRUE
                  SEQ
                    -- source potential
                    Qs[0]:=(0.0(REAL32)-0.079577471(REAL32))*(Imn[0][0]/radius)
                    -- dipole potential
                    Qd[0]:=(Pen[2]*Qs[0])/(radius*radius)
                TRUE
                  SEQ
                    Qs[0]:=0.0(REAL32)
                    -- dipole potential
                    Qd[0]:=((Pen[2]*Imn[0][0])/(radius*(radius*radius)))*
                          (0.0(REAL32)-0.079577471(REAL32))
            --}}}
    --}}}
    TRUE
      IF
        choiceparameter>2.45(REAL32)
          --{{{  evaluate 2nd order multipole
```

```
        SEQ
          IF
            IsVel=TRUE
              IF
                Both=TRUE
                  VectMultipole(2,radius,Pen,Imn,Qsen,Qden)
                TRUE
                  VectDipoleMultipole(2,radius,Pen,Imn,Qden)
            TRUE
              IF
                Both=TRUE
                  EvaluateMultipole(2,radius,Pen,Imn,Qs[0],Qd[0])
                TRUE
                  DipoleMultipole(2,radius,Pen,Imn,Qd[0])
        -}}}
      TRUE
        IF
          choiceparameter>2.0(REAL32)
          -{{{  evaluate 4th order multipole
          SEQ
            IF
              IsVel=TRUE
                IF
                  Both=TRUE
                    VectMultipole(4,radius,Pen,Imn,Qsen,Qden)
                  TRUE
                    VectDipoleMultipole(4,radius,Pen,Imn,Qden)
              TRUE
                IF
                  Both=TRUE
                    EvaluateMultipole(4,radius,Pen,Imn,Qs[0],Qd[0])
                  TRUE
                    DipoleMultipole(4,radius,Pen,Imn,Qd[0])
          -}}}
        TRUE
          -{{{  exact
          IF
            IsVel=TRUE
              -{{{  velocity
              VectExactPanel(Both,Imn[0][0],radius,choiceparameter,Pen,E,N,s,dE,dN,Qsen,Qden)
              -}}}
            TRUE
              -{{{  exact
              ExactPanel(Both,Imn[0][0],radius,choiceparameter,Pen,E,N,s,dE,dN,Qs[0],Qd[0])
              -}}}
          -}}}
  -{{{  translate
  IF
    IsVel=TRUE
      SEQ
        IF
          Both=TRUE
            TranslateCoord(Qsen,At,Qs)
          TRUE
            SKIP
        TranslateCoord(Qden,At,Qd)
    TRUE
      SKIP
  -}}}
:
-}}}
```

```
--{{{  Dipole
--{{{  vectdipole
--{{{  Vectdimultipole
PROC VectDipoleMultipole(VAL INT ordur,VAL REAL32 radius,
                         VAL [3]REAL32 Pen,
                         VAL [5][5]REAL32 lmn,
                         [3]REAL32 Qden)
  --{{{  calculatedipole
  PROC VCalculateDipole(VAL REAL32 R2,x,y,z,
                            RM1,RM3,RM5,
                            xz,RM7,yz,
                            RM9,x2,x2z,xy,xyz,y2,y2z,
                            RM11,x3,x3z,x2y,x2yz,y2x,y2xz,y3,y3z,
              x4,x4z,x3y,x3yz,x2y2z,y4,y4z,y3x,y3xz,x2y2,RM13,
                          VAL INT m,n,VAL [3]REAL32 Pen,
                          [3]REAL32 Qden)
    [3]REAL32 VaddD,temp:
    REAL32 ScaleitD:
    SEQ
      VectorEqual(Pen,temp)
      --{{{  set up
      ScaleitD:=0.0(REAL32)
      SEQ i=0 FOR 3
        VaddD[i]:=0.0(REAL32)
      --}}}
      IF
        m=0
          --{{{
          IF
            n=0
              --{{{  0 0
              SEQ

                -- dipole
                VaddD[2]:=RM3

                ScaleitD:=((-3.0(REAL32))*z)*RM5

              --}}}
            n=1
              --{{{  0 1
              SEQ

                -- dipole
                VaddD[1]:=(3.0(REAL32)*z)*(-RM5)
                VaddD[2]:=(3.0(REAL32)*y)*(-RM5)

                ScaleitD:=(15.0(REAL32)*y)*(z*RM7)
              --}}}
            n=2
              --{{{  0 2
              SEQ

                -- dipole

                VaddD[1]:=(30.0(REAL32)*y)*(z*RM7)
                VaddD[2]:=((15.0(REAL32)*y2)*RM7)-(3.0(REAL32)*RM5)

                ScaleitD:=((15.0(REAL32)*z)*RM7)-((105.0(REAL32)*y2z)*RM9)
              --}}}
            n=3
```

```
    --{{{  0 3
    SEQ

      -- dipole
      VaddD[1]: = ((45.0(REAL32)*z)*RM7)-((210.0(REAL32)*y2z)*RM9)
      VaddD[2]: = ((45.0(REAL32)*y)*RM7)-((105.0(REAL32)*y3)*RM9)

      ScaleitD: = ((945.0(REAL32)*y3z)*RM11)-((315.0(REAL32)*yz)*RM9)
    --}}}
  n=4
    --{{{  0 4
    SEQ

      -- dipole

      VaddD[1]: = ((3780.0(REAL32)*y3z)*RM11)-((1260.0(REAL32)*yz)*RM9)
      VaddD[2]: = 0.0(REAL32)-((((630.0(REAL32)*y2)*RM9)-
            ((945.0(REAL32)*y4)*RM11))-
            (45.0(REAL32)*RM7))
      ScaleitD: = (((5670.0(REAL32)*y2z)*RM11)-((10395.0(REAL32)*y4z)*RM13))-
            ((315.0(REAL32)*z)*RM9)
    --}}}
  TRUE
    SKIP
--}}}
m=1
--{{{
IF
  n=0
    --{{{  1 0
    SEQ
      -- dipole
      VaddD[0]: = (3.0(REAL32)*z)*(-RM5)
      VaddD[2]: = (3.0(REAL32)*x)*(-RM5)
      ScaleitD: = ((15.0(REAL32)*xz)*RM7)
    --}}}
  n=1
    --{{{  1 1
    SEQ

      -- dipole

      VaddD[0]: = ((15.0(REAL32)*yz)*RM7)
      VaddD[1]: = ((15.0(REAL32)*xz)*RM7)
      VaddD[2]: = 0.0(REAL32)-(((15.0(REAL32)*xy)*(-RM7)))
      ScaleitD: = ((105.0(REAL32)*xyz)*(-RM9))
    --}}}
  n=2
    --{{{  1 2
    SEQ

      -- dipole

      VaddD[0]: = ((15.0(REAL32)*z)*RM7)-((105.0(REAL32)*y2z)*RM9)
      VaddD[1]: = ((210.0(REAL32)*y2x)*(-RM9))
      VaddD[2]: = 0.0(REAL32)-( ((105.0(REAL32)*y2x)*RM9)-((15.0(REAL32)*x)*RM7)  )

      ScaleitD: = ((945.0(REAL32)*y2xz)*RM11)-((105.0(REAL32)*xz)*RM9)
    --}}}
  n=3
    --{{{  1 3
```

```
      SEQ

        -- dipole
        VaddD[0]:=((945.0(REAL32)*y3z)*RM11)-((315.0(REAL32)*yz)*RM9)
        VaddD[1]:=((2835.0(REAL32)*y3xz)*RM11)-((315.0(REAL32)*xz)*RM9)
        VaddD[2]:=0.0(REAL32)-( ((315.0(REAL32)*xy)*RM9)-((945.0(REAL32)*y3x)*RM11) )

        ScaleitD:=((2835.0(REAL32)*xyz)*RM11)-((10395.0(REAL32)*y3x)*RM13)

      -}}}
    TRUE
      SKIP
  -}}}
 m=2
  --{{{
  IF
    n=0
      --{{{  2 0
      SEQ

        -- dipole

        VaddD[0]:=((30.0(REAL32)*xz)*RM7)
        VaddD[2]:=0.0(REAL32)-( (3.0(REAL32)*RM5)-((15.0(REAL32)*x2)*RM7) )
        ScaleitD:=((15.0(REAL32)*z)*RM7)-((105.0(REAL32)*x2z)*RM9)
      -}}}
    n=1
      --{{{  2 1
      SEQ

        -- dipole

        VaddD[0]:=(210.0(REAL32)*xyz)*(-RM9)
        VaddD[1]:=((15.0(REAL32)*z)*RM7)-((105.0(REAL32)*x2z)*RM9)
        VaddD[2]:=0.0(REAL32)-( ((105.0(REAL32)*xy)*RM9)-((15.0(REAL32)*y)*RM7)  )

        ScaleitD:=((945.0(REAL32)*x2yz)*RM11)-((105.0(REAL32)*yz)*RM9)
      -}}}
    n=2
      --{{{  2 2
      SEQ

        -- dipole
        VaddD[0]:=((1890.0(REAL32)*y2xz)*RM11)-((210.0(REAL32)*xz)*RM9)
        VaddD[1]:=((1890.0(REAL32)*x2yz)*RM11)-((210.0(REAL32)*yz)*RM9)
        VaddD[2]:=0.0(REAL32)-( (((105.0(REAL32)*RM9)*(x2+y2))-
                ((945.0(REAL32)*x2y2)*RM11))-(15.0(REAL32)*RM7)  )

        ScaleitD:=(((945.0(REAL32)*z)*(RM11*(x2+y2)))-((210.0(REAL32)*z)*RM9))-
                ((945.0(REAL32)*x2y2z)*RM13)

      -}}}
    TRUE
      SKIP
  -}}}
 m=3
  --{{{
  IF
    n=0
      --{{{  3 0
      SEQ
```

```
          -- dipole

          VaddD[0]:=((45.0(REAL32)*z)*RM7)-((210.0(REAL32)*x2z)*RM9)
          VaddD[2]:=0.0(REAL32)-( ((105.0(REAL32)*x3)*RM9)-((45.0(REAL32)*x)*RM7)  )
          ScaleitD:=((945.0(REAL32)*x3z)*RM11)-((315.0(REAL32)*xz)*RM9)
        -}}}
       n=1
        -{{{  3 1
        SEQ

          -- dipole

          VaddD[0]:=((2835.0(REAL32)*x2yz)*RM11)-((315.0(REAL32)*yz)*RM9)
          VaddD[1]:=((945.0(REAL32)*x3z)*RM11)-((315.0(REAL32)*xz)*RM9)
          VaddD[2]:=0.0(REAL32)-( ((315.0(REAL32)*xy)*RM9)-((945.0(REAL32)*x3y)*RM11) )

          ScaleitD:=((2835.0(REAL32)*xyz)*RM11)-((10395.0(REAL32)*x3yz)*RM13)

        -}}}
       TRUE
        SKIP
      -}}}
     m=4
      -{{{
      IF
       n=0
        -{{{  4 0
        SEQ

          -- dipole
          VaddD[0]:=((3780.0(REAL32)*x3z)*RM11)-((1260.0(REAL32)*xz)*RM9)
          VaddD[2]:=0.0(REAL32)-( (((630.0(REAL32)*x2)*RM9)-((945.0(REAL32)*x4)*
                RM11))-(45.0(REAL32)*RM7)  )

          ScaleitD:=(((5670.0(REAL32)*x2z)*RM11)-((10395.0(REAL32)*x4z)*RM13))-
                ((315.0(REAL32)*z)*RM9)


        -}}}
       TRUE
        SKIP
      -}}}
     TRUE
      SKIP
   ScalarMul(ScaleitD,temp)
   VectorAdd(temp,VaddD,Qden)
:
-}}}
-{{{  variables
REAL32 R2,x,y,z:
REAL32 RM1,RM3,RM5:
REAL32 xz,RM7,yz:
REAL32 RM9,x2,x2z,xy,xyz,y2,y2z:
REAL32 RM11,x3,x3z,x2y,x2yz,y2x,y2z,y3,y3z:
REAL32 RM13,x4,x4z,x3y,x3yz,x2y2z,y4,y4z,y3x,y3xz,x2y,y2x,x2y2:
-}}}
SEQ
  -{{{  calculate parameters
  R2:=radius*radius
  x:=Pen[0]
```

```
y:=Pen[1]
z:=Pen[2]
--{{{  ordur=0
IF
  ordur>=0
    SEQ
      RM1:=1.0(REAL32)/radius
      RM3:=RM1/R2
      RM5:=RM3/R2
  TRUE
    SKIP
--}}}
--{{{  ordur=1
IF
  ordur>=1
    SEQ
      xz:=x*z
      RM7:=RM5/R2
      yz:=y*z
  TRUE
    SKIP
--}}}
--{{{  ordur=2
IF
  ordur>=2
    SEQ
      RM9:=RM7/R2
      x2:=x*x
      x2z:=x2*z
      xy:=x*y
      xyz:=xy*z
      y2:=y*y
      y2z:=y2*z
  TRUE
    SKIP
--}}}
--{{{  ordur=3
IF
  ordur>=3
    SEQ
      RM11:=RM9/R2
      x3:=x2*x
      x3z:=x3*z
      x2y:=x2*y
      x2yz:=x2y*z
      y2x:=y2*x
      y2xz:=y2x*z
      y3:=y2*y
      y3z:=y3*z
  TRUE
    SKIP
--}}}
--{{{  ordur=4
IF
  ordur>=4
    SEQ
      RM13:=RM11/R2
      x4:=x3*x
      x4z:=x4*z
      x3y:=x3*y
      x3yz:=x3y*z
```

```
        x2y2: = x2*y2
        x2y2z: = x2y2*z
        y4: = y3*y
        y4z: = y4*z
        y3x: = y3*x
        y3xz: = y3x*z
      TRUE
        SKIP
    -}}}
    -}}}
    -{{{  initialise Dipole
    SEQ i=0 FOR 3
      Qden[i]: = 0.0(REAL32)
    -}}}
    SEQ m=0 FOR  (ordur+1)
      SEQ n=0 FOR (ordur+1)
        SEQ
          IF
            (m+n)< =ordur
              SEQ
                IF
                  Imn[m][n]< >0.0(REAL32)
                    -{{{  calculate coefficients
                    [3]REAL32 Qd:
                    SEQ
                      VCalculateDipole(R2,x,y,z,
                                       RM1,RM3,RM5,
                                       xz,RM7,yz,
                                       RM9,x2,x2z,xy,xyz,y2,y2z,
                                       RM11,x3,x3z,x2y,x2yz,y2x,y2xz,y3,y3z,
                                  x4,x4z,x3y,x3yz,x2y2z,y4,y4z,y3x,y3xz,x2y2,RM13,
                                  m,n,Pen,Qd)
                      ScalarMul(Imn[m][n],Qd)
                      VectorAdd(Qd,Qden,Qden)
                    -}}}
                  TRUE
                    SKIP
            TRUE
              SKIP
    ScalarMul((0.0(REAL32)-0.079577471(REAL32)),Qden)
:
-}}}
-}}}
-{{{  multipole
PROC EvaluateMultipole(VAL INT ordur,VAL REAL32 radius,
                       VAL [3]REAL32 Pen,
                       VAL [5][5]REAL32 Imn,
                       REAL32 Source,Dipole)
  -{{{  calculatesourcedipole
  PROC CalculateSourceDipole(VAL REAL32 R2,x,y,z,
                             RM1,RM3,RM5,
                             xz,RM7,yz,
                             RM9,x2,x2z,xy,xyz,y2,y2z,
                             RM11,x3,x3z,x2y,x2yz,y2x,y2xz,y3,y3z,
                    x4,x4z,x3y,x3yz,x2y2z,y4,y4z,y3x,y3xz,x2y2,
                             VAL INT m,n,VAL [3]REAL32 Pen,
                             REAL32 Qs,Qd)
    SEQ
      -{{{  set up
      Qs: = 0.0(REAL32)
      Qd: = 0.0(REAL32)
```

```
-}}}
IF
 m=0
  --{{{
  IF
   n=0
    --{{{  0 0
    SEQ
     -- source
     Qs:=RM1

     -- dipole

     Qd:=z*RM3

    -}}}
   n=1
    --{{{  0 1
    SEQ
     -- source

     Qs:=0.0(REAL32)-(y*RM3)

     -- dipole

     Qd:=0.0(REAL32)-(3.0(REAL32)*(yz*RM5))

    -}}}
   n=2
    --{{{  0 2
    SEQ
     -- source

     Qs:=((3.0(REAL32)*y2)*RM5)-RM3

     -- dipole

     Qd:= ((15.0(REAL32)*y2z)*RM7)  - (3.0(REAL32)*(z*RM5))

    -}}}
   n=3
    --{{{  0 3
    SEQ
     -- source

     Qs:= (9.0(REAL32)*(y*RM5)) - ((15.0(REAL32)*y3)*RM7)

     -- dipole

     Qd:= ((45.0(REAL32)*yz)*RM7) - ((105.0(REAL32)*y3z)*RM9)

    -}}}
   n=4
    --{{{  0 4
    SEQ
     -- source

     Qs:= (9.0(REAL32)*RM5) +
         ((105.0(REAL32)*y4)*RM9)

     -- dipole
```

```
      Qd:= ((45.0(REAL32)*z)*RM7) +
         ((945.0(REAL32)*y4z)*RM11)

    -}}}
   TRUE
    SKIP
  -}}}
m=1
  -{{{
  IF
   n=0
    -{{{  1 0
    SEQ
     -- source

     Qs:= 0.0(REAL32)-(x*RM3)

     -- dipole

     Qd:= 0.0(REAL32)-((3.0(REAL32)*xz)*RM5)

    -}}}
   n=1
    -{{{  1 1
    SEQ
     -- source

     Qs:=(3.0(REAL32)*xy)*RM5

     -- dipole

     Qd:=(15.0(REAL32)*xyz)*RM7

    -}}}
   n=2
    -{{{  1 2
    SEQ
     -- source

     Qs:=((3.0(REAL32)*x)*RM5) - ((15.0(REAL32)*y2x)*RM7)

     -- dipole

     Qd:=((15.0(REAL32)*xz)*RM7) - ((105.0(REAL32)*y2xz)*RM9)

    -}}}
   n=3
    -{{{  1 3
    SEQ
     -- source

     Qs:= ((105.0(REAL32)*y3x)*RM9) - ((45.0(REAL32)*xy)*RM7)

     -- dipole

     Qd:= ((945.0(REAL32)*y3xz)*RM11) - ((315.0(REAL32)*xyz)*RM9)

    -}}}
   TRUE
    SKIP
```

```
  --}}}
m=2
  --{{{
  IF
    n=0
      --{{{  2 0
      SEQ
        -- source

        Qs: =((3.0(REAL32)*x2)*RM5)-RM3

        -- dipole

        Qd: =  ((15.0(REAL32)*x2z)*RM7)-(3.0(REAL32)*(z*RM5))

      --}}}
    n=1
      --{{{  2 1
      SEQ
        -- source

        Qs: =  ((3.0(REAL32)*y)*RM5) - ((15.0(REAL32)*x2y)*RM7)

        -- dipole

        Qd: =   ((15.0(REAL32)*yz)*RM7) - ((105.0(REAL32)*x2yz)*RM9)

      --}}}
    n=2
      --{{{  2 2
      SEQ
        -- source

        Qs: =  (((105.0(REAL32)*x2y2)*RM9)+(3.0(REAL32)*RM5))-
            ((15.0(REAL32)*(x2+y2))*RM7)

        -- dipole

        Qd: =  (((945.0(REAL32)*x2y2z)*RM11)  -
             ((105.0(REAL32)*(x2z+y2z))*RM9)) +
            ((15.0(REAL32)*z)*RM7)

      --}}}
    TRUE
      SKIP
  --}}}
m=3
  --{{{
  IF
    n=0
      --{{{  3 0
      SEQ
        -- source

        Qs: =  ((9.0(REAL32)*x)*RM5) - (( 15.0(REAL32)*x3)*RM7)

        -- dipole

        Qd: =  ((45.0(REAL32)*xz)*RM7) - ((105.0(REAL32)*x3z)*RM9)

      --}}}
```

```
      n=1
       -{{{  3 1
       SEQ
        -- source

        Qs:= ((105.0(REAL32)*x3y)*RM9) - ((45.0(REAL32)*xy)*RM7)

        -- dipole

        Qd:= ((945.0(REAL32)*x3yz)*RM11) - ((315.0(REAL32)*xyz)*RM9)

       -}}}
      TRUE
       SKIP
    -}}}
   m=4
    -{{{
    IF
     n=0
      -{{{  4 0
      SEQ
       -- source

       Qs:=  (9.0(REAL32)*RM5)   +
          ((105.0(REAL32)*x4)*RM9)

       -- dipole

       Qd:= ((45.0(REAL32)*z)*RM7)  +
          ((945.0(REAL32)*x4z)*RM11)

      -}}}
     TRUE
      SKIP
   -}}}
  TRUE
   SKIP
:
-}}}
-{{{  variables
REAL32 R2,x,y,z:
REAL32 RM1,RM3,RM5:
REAL32 xz,RM7,yz:
REAL32 RM9,x2,x2z,xy,xyz,y2,y2z:
REAL32 RM11,x3,x3z,x2y,x2yz,y2x,y2xz,y3,y3z:
REAL32 x4,x4z,x3y,x3yz,x2y2z,y4,y4z,y3x,y3xz,x2y,y2x,x2y2:
-}}}
SEQ
 -{{{  calculate parameters
 R2:=radius*radius
 x:=Pen[0]
 y:=Pen[1]
 z:=Pen[2]
 -{{{  ordur=0
 IF
  ordur>=0
   SEQ
    RM1:=1.0(REAL32)/radius
    RM3:=RM1/R2
  TRUE
   SKIP
```

```
--}}}
--{{{  ordur=1
IF
  ordur>=1
    SEQ
      xz:=x*z
      RM5:=RM3/R2
      yz:=y*z
  TRUE
    SKIP
--}}}
--{{{  ordur=2
IF
  ordur>=2
    SEQ
      RM7:=RM5/R2
      x2:=x*x
      x2z:=x2*z
      xy:=x*y
      xyz:=xy*z
      y2:=y*y
      y2z:=y2*z
  TRUE
    SKIP
--}}}
--{{{  ordur=3
IF
  ordur>=3
    SEQ
      RM9:=RM7/R2
      x3:=x2*x
      x3z:=x3*z
      x2y:=x2*y
      x2yz:=x2y*z
      y2x:=y2*x
      y2xz:=y2x*z
      y3:=y2*y
      y3z:=y3*z
  TRUE
    SKIP
--}}}
--{{{  ordur=4
IF
  ordur>=4
    SEQ
      RM11:=RM9/R2
      x4:=x3*x
      x4z:=x4*z
      x3y:=x3*y
      x3yz:=x3y*z
      x2y2:=x2*y2
      x2y2z:=x2y2*z
      y4:=y3*y
      y4z:=y4*z
      y3x:=y3*x
      y3xz:=y3x*z
  TRUE
    SKIP
--}}}
--}}}
--{{{  initialise Source,Dipole
```

```
  Source:=0.0(REAL32)
  Dipole:=0.0(REAL32)
  --}}}
  SEQ m=0 FOR  (ordur+1)
   SEQ n=0 FOR (ordur+1)
    SEQ
     IF
      (m+n)<=ordur
       SEQ
        IF
         Imn[m][n]<>0.0(REAL32),
          --{{{  calculate coefficients
          REAL32 Qs,Qd:
          SEQ
           CalculateSourceDipole(R2,x,y,z,
                                 RM1,RM3,RM5,
                                 xz,RM7,yz,
                                 RM9,x2,x2z,xy,xyz,y2,y2z,
                                 RM11,x3,x3z,x2y,x2yz,y2x,y2xz,y3,y3z,
                           x4,x4z,x3y,x3yz,x2y2z,y4,y4z,y3x,y3xz,x2y2,
                           m,n,Pen,Qs,Qd)
           Qs:=Imn[m][n]*Qs
           Qd:=Imn[m][n]*Qd
           Source:=Source+Qs
           Dipole:=Dipole+Qd
          --}}}
         TRUE
          SKIP
       TRUE
        SKIP
  Source:=(0.0(REAL32)-0.079577471(REAL32))*Source
  Dipole:=(0.0(REAL32)-0.079577471(REAL32))*Dipole
:
--}}}
--{{{  dimultipole
PROC DipoleMultipole(VAL INT ordur,VAL REAL32 radius,
                     VAL [3]REAL32 Pen,
                     VAL [5][5]REAL32 Imn,
                     REAL32 Dipole)
 --{{{  calculatedipole
 PROC CalculateDipole(VAL REAL32 R2,x,y,z,
                      RM1,RM3,RM5,
                      xz,RM7,yz,
                      RM9,x2,x2z,xy,xyz,y2,y2z,
                      RM11,x3,x3z,x2y,x2yz,y2x,y2xz,y3,y3z,
           x4,x4z,x3y,x3yz,x2y2z,y4,y4z,y3x,y3xz,x2y2,
                      VAL INT m,n,VAL [3]REAL32 Pen,
                      REAL32 Qd)
  SEQ
   --{{{  set up
   Qd:=0.0(REAL32)
   --}}}
   IF
    m=0
     --{{{
     IF
      n=0
       --{{{  0 0
       SEQ

        -- dipole
```

```
      Qd:=z*RM3

    --}}}
   n=1
    --{{{  0 1
    SEQ

      -- dipole

      Qd:=0.0(REAL32)-(3.0(REAL32)*(yz*RM5))

    --}}}
   n=2
    --{{{  0 2
    SEQ

      -- dipole

      Qd:= ((15.0(REAL32)*y2z)*RM7)  - (3.0(REAL32)*(z*RM5))

    --}}}
   n=3
    --{{{  0 3
    SEQ

      -- dipole

      Qd:= ((45.0(REAL32)*yz)*RM7) - ((105.0(REAL32)*y3z)*RM9)

    --}}}
   n=4
    --{{{  0 4
    SEQ

      -- dipole

      Qd:= ((45.0(REAL32)*z)*RM7) +
          ((945.0(REAL32)*y4z)*RM11)

    --}}}
   TRUE
    SKIP
  --}}}
m=1
  --{{{
  IF
    n=0
      --{{{  1 0
      SEQ

        -- dipole

        Qd:= 0.0(REAL32)-((3.0(REAL32)*xz)*RM5)

      --}}}
    n=1
      --{{{  1 1
      SEQ

        -- dipole
```

```
      Qd:=(15.0(REAL32)*xyz)*RM7

    -}}}
   n=2
    -{{{  1 2
    SEQ

      -- dipole

      Qd:=((15.0(REAL32)*xz)*RM7) - ((105.0(REAL32)*y2xz)*RM9)

    -}}}
   n=3
    -{{{  1 3
    SEQ

      -- dipole

      Qd:= ((945.0(REAL32)*y3xz)*RM11) - ((315.0(REAL32)*xyz)*RM9)

    -}}}
   TRUE
    SKIP
 -}}}
m=2
 -{{{
 IF
   n=0
    -{{{  2 0
    SEQ

      -- dipole

      Qd:= ((15.0(REAL32)*x2z)*RM7)-(3.0(REAL32)*(z*RM5))

    -}}}
   n=1
    -{{{  2 1
    SEQ

      -- dipole

      Qd:=  ((15.0(REAL32)*yz)*RM7) - ((105.0(REAL32)*x2yz)*RM9)

    -}}}
   n=2
    -{{{  2 2
    SEQ

      -- dipole

      Qd:= (((945.0(REAL32)*x2y2z)*RM11)  -
           ((105.0(REAL32)*(x2z+y2z))*RM9)) +
          ((15.0(REAL32)*z)*RM7)

    -}}}
   TRUE
    SKIP
 -}}}
m=3
```

```
      --{{{
      IF
        n=0
          --{{{  3 0
          SEQ

            -- dipole

            Qd:=  ((45.0(REAL32)*xz)*RM7) - ((105.0(REAL32)*x3z)*RM9)

          --}}}
        n=1
          --{{{  3 1
          SEQ

            -- dipole

            Qd:= ((945.0(REAL32)*x3yz)*RM11) - ((315.0(REAL32)*xyz)*RM9)

          --}}}
        TRUE
          SKIP
      --}}}
    m=4
      --{{{
      IF
        n=0
          --{{{  4 0
          SEQ

            -- dipole

            Qd:= ((45.0(REAL32)*z)*RM7)  +
                 ((945.0(REAL32)*x4z)*RM11)

          --}}}
        TRUE
          SKIP
      --}}}
    TRUE
      SKIP
:
--}}}
--{{{  variables
REAL32 R2,x,y,z:
REAL32 RM1,RM3,RM5:
REAL32 xz,RM7,yz:
REAL32 RM9,x2,x2z,xy,xyz,y2,y2z:
REAL32 RM11,x3,x3z,x2y,x2yz,y2x,y2xz,y3,y3z:
REAL32 x4,x4z,x3y,x3yz,x2y2z,y4,y4z,y3x,y3xz,x2y,y2x,x2y2:
--}}}
SEQ
  --{{{  calculate parameters
  R2:=radius*radius
  x:=Pen[0]
  y:=Pen[1]
  z:=Pen[2]
  --{{{  ordur=0
  IF
    ordur>=0
      SEQ
```

```
      RM1:=1.0(REAL32)/radius
      RM3:=RM1/R2
  TRUE
    SKIP
-}}}
-{{{  ordur=1
IF
  ordur>=1
    SEQ
      xz:=x*z
      RM5:=RM3/R2
      yz:=y*z
  TRUE
    SKIP
-}}}
-{{{  ordur=2
IF
  ordur>=2
    SEQ
      RM7:=RM5/R2
      x2:=x*x
      x2z:=x2*z
      xy:=x*y
      xyz:=xy*z
      y2:=y*y
      y2z:=y2*z
  TRUE
    SKIP
-}}}
-{{{  ordur=3
IF
  ordur>=3
    SEQ
      RM9:=RM7/R2
      x3:=x2*x
      x3z:=x3*z
      x2y:=x2*y
      x2yz:=x2y*z
      y2x:=y2*x
      y2xz:=y2x*z
      y3:=y2*y
      y3z:=y3*z
  TRUE
    SKIP
-}}}
-{{{  ordur=4
IF
  ordur>=4
    SEQ
      RM11:=RM9/R2
      x4:=x3*x
      x4z:=x4*z
      x3y:=x3*y
      x3yz:=x3y*z
      x2y2:=x2*y2
      x2y2z:=x2y2*z
      y4:=y3*y
      y4z:=y4*z
      y3x:=y3*x
      y3xz:=y3x*z
  TRUE
```

```
      SKIP
  --}}}
  --}}}
  --{{{  initialise Dipole
  Dipole:=0.0(REAL32)
  --}}}
  SEQ m=0 FOR  (ordur+1)
    SEQ n=0 FOR (ordur+1)
      SEQ
        IF
          (m+n)<=ordur
            SEQ
              IF
                Imn[m][n]<>0.0(REAL32)
                  --{{{  calculate coefficients
                  REAL32 Qs,Qd:
                  SEQ
                    CalculateDipole(R2,x,y,z,
                                    RM1,RM3,RM5,
                                    xz,RM7,yz,
                                    RM9,x2,x2z,xy,xyz,y2,y2z,
                                    RM11,x3,x3z,x2y,x2yz,y2x,y2xz,y3,y3z,
                                x4,x4z,x3y,x3yz,x2y2z,y4,y4z,y3x,y3xz,x2y2,
                                m,n,Pen,Qd)
                    Qd:=Imn[m][n]*Qd
                    Dipole:=Dipole+Qd
                  --}}}
                TRUE
                  SKIP
          TRUE
            SKIP
    Dipole:=(0.0(REAL32)-0.079577471(REAL32))*Dipole
:
--}}}
--{{{  Vectmultipole
PROC VectMultipole(VAL INT ordur,VAL REAL32 radius,
                   VAL [3]REAL32 Pen,
                   VAL [5][5]REAL32 Imn,
                   [3]REAL32 Qsen,Qden)
  --{{{  calculatedipole
  PROC VCalculateDipole(VAL REAL32 R2,x,y,z,
                        RM1,RM3,RM5,
                        xz,RM7,yz,
                        RM9,x2,x2z,xy,xyz.y2,y2z,
                        RM11,x3,x3z,x2y,x2yz,y2x,y2xz,y3,y3z,
            x4,x4z,x3y,x3yz,x2y2z,y4,y4z,y3x,y3xz,x2y2,RM13,
                      VAL INT m,n,VAL [3]REAL32 Pen,
                      [3]REAL32 Qsen,Qden)
    [3]REAL32 VaddD,VaddS,temp:
    REAL32 ScaleitD,ScaleitS:
    SEQ
      VectorEqual(Pen,temp)
      --{{{  set up
      ScaleitD:=0.0(REAL32)
      ScaleitS:=0.0(REAL32)
      SEQ i=0 FOR 3
        SEQ
          VaddD[i]:=0.0(REAL32)
          VaddS[i]:=0.0(REAL32)
      --}}}
      IF
```

```
m=0
 -{{{
 IF
  n=0
   -{{{  0 0
   SEQ
    ScaleitS: = -RM3
    -- dipole
    VaddD[2]: = RM3

    ScaleitD: = ((-3.0(REAL32))*z)*RM5

   -}}}
  n=1
   -{{{  0 1
   SEQ
    VaddS[1]: = -RM3
    ScaleitS: = (3.0(REAL32)*y)*RM5

    -- dipole
    VaddD[1]: = (3.0(REAL32)*z)*(-RM5)
    VaddD[2]: = (3.0(REAL32)*y)*(-RM5)

    ScaleitD: = (15.0(REAL32)*y)*(z*RM7)
   -}}}
  n=2
   -{{{  0 2
   SEQ
    VaddS[1]: = (6.0(REAL32)*y)*RM5
    ScaleitS: = (3.0(REAL32)*RM5)-((15.0(REAL32)*y2)*RM7)

    -- dipole

    VaddD[1]: = (30.0(REAL32)*y)*(z*RM7)
    VaddD[2]: = ((15.0(REAL32)*y2)*RM7)-(3.0(REAL32)*RM5)

    ScaleitD: = ((15.0(REAL32)*z)*RM7)-((105.0(REAL32)*y2z)*RM9)
   -}}}
  n=3
   -{{{  0 3
   SEQ
    VaddS[1]: = (9.0(REAL32)*RM5)-((30.0(REAL32)*y2)*RM7)
    ScaleitS: = ((105.0(REAL32)*y3)*RM9)-((45.0(REAL32)*y)*RM7)
    -- dipole
    VaddD[1]: = ((45.0(REAL32)*z)*RM7)-((210.0(REAL32)*y2z)*RM9)
    VaddD[2]: = ((45.0(REAL32)*y)*RM7)-((105.0(REAL32)*y3)*RM9)

    ScaleitD: = ((945.0(REAL32)*y3z)*RM11)-((315.0(REAL32)*yz)*RM9)
   -}}}
  n=4
   -{{{  0 4
   SEQ
    VaddS[1]: = ((420.0(REAL32)*y3)*RM9)-((180.0(REAL32)*y)*RM7)
    ScaleitS: = (((630.0(REAL32)*y2)*RM9)-((945.0(REAL32)*y4)*RM11))-
              (45.0(REAL32)*RM7)
    -- dipole

    VaddD[1]: = ((3780.0(REAL32)*y3z)*RM11)-((1260.0(REAL32)*yz)*RM9)
    VaddD[2]: = 0.0(REAL32)-((((630.0(REAL32)*y2)*RM9)-
              ((945.0(REAL32)*y4)*RM11))-
              (45.0(REAL32)*RM7))
```

```
      ScaleitD: = (((5670.0(REAL32)*y2z)*RM11)-((10395.0(REAL32)*y4z)*RM13))-
            ((315.0(REAL32)*z)*RM9)
    --}}}
   TRUE
    SKIP
 --}}}
m=1
 --{{{
 IF
  n=0
   --{{{  1 0
   SEQ
    VaddS[0]: = -RM3
    ScaleitS: = (3.0(REAL32)*RM5)*x
    -- dipole
    VaddD[0]: = (3.0(REAL32)*z)*(-RM5)
    VaddD[2]: = (3.0(REAL32)*x)*(-RM5)
    ScaleitD: = ((15.0(REAL32)*xz)*RM7)
   --}}}
  n=1
   --{{{  1 1
   SEQ
    VaddS[0]: = (3.0(REAL32)*y)*RM5
    VaddS[1]: = (3.0(REAL32)*x)*RM5

    -- dipole

    VaddD[0]: = ((15.0(REAL32)*yz)*RM7)
    VaddD[1]: = ((15.0(REAL32)*xz)*RM7)
    VaddD[2]: = 0.0(REAL32)-(((15.0(REAL32)*xy)*(-RM7)))
    ScaleitS: = -VaddD[2]
    ScaleitD: = ((105.0(REAL32)*xyz)*(-RM9))
   --}}}
  n=2
   --{{{  1 2
   SEQ
    VaddS[0]: = (3.0(REAL32)*RM5)-((15.0(REAL32)*y2)*RM7)
    VaddS[1]: = ((30.0(REAL32)*xy)*RM7)
    -- dipole

    VaddD[0]: = ((15.0(REAL32)*z)*RM7)-((105.0(REAL32)*y2z)*RM9)
    VaddD[1]: = ((210.0(REAL32)*y2x)*(-RM9))
    VaddD[2]: = 0.0(REAL32)-( ((105.0(REAL32)*y2x)*RM9)-((15.0(REAL32)*x)*RM7) )
    ScaleitS: = -VaddD[2]
    ScaleitD: = ((945.0(REAL32)*y2xz)*RM11)-((105.0(REAL32)*xz)*RM9)
   --}}}
  n=3
   --{{{  1 3
   SEQ
    VaddS[0]: = ((105.0(REAL32)*y3)*RM9)-((45.0(REAL32)*y)*RM7)
    VaddS[1]: = ((315.0(REAL32)*y2x)*RM9)-((45.0(REAL32)*x)*RM7)

    -- dipole
    VaddD[0]: = ((945.0(REAL32)*y3z)*RM11)-((315.0(REAL32)*yz)*RM9)
    VaddD[1]: = ((2835.0(REAL32)*y3xz)*RM11)-((315.0(REAL32)*xz)*RM9)
    VaddD[2]: = 0.0(REAL32)-( ((315.0(REAL32)*xy)*RM9)-((945.0(REAL32)*y3x)*RM11) )

    ScaleitD: = ((2835.0(REAL32)*xyz)*RM11)-((10395.0(REAL32)*y3x)*RM13)
    ScaleitS: = -VaddD[2]
   --}}}
  TRUE
```

```
      SKIP
    -}}}
  m=2
    -{{{
    IF
      n=0
        -{{{  2 0
        SEQ
          VaddS[0]:=(6.0(REAL32)*y)*RM5

          -- dipole

          VaddD[0]:=((30.0(REAL32)*xz)*RM7)
          VaddD[2]:=0.0(REAL32)-( (3.0(REAL32)*RM5)-((15.0(REAL32)*x2)*RM7) )
          ScaleitD:=((15.0(REAL32)*z)*RM7)-((105.0(REAL32)*x2z)*RM9)
          ScaleitS:=-VaddD[2]
        -}}}
      n=1
        -{{{  2 1
        SEQ
          VaddS[0]:=(30.0(REAL32)*xy)*(-RM7)
          VaddS[1]:=(3.0(REAL32)*RM5)-((15.0(REAL32)*x2)*RM7)
          -- dipole

          VaddD[0]:=(210.0(REAL32)*xyz)*(-RM9)
          VaddD[1]:=((15.0(REAL32)*z)*RM7)-((105.0(REAL32)*x2z)*RM9)
          VaddD[2]:=0.0(REAL32)-( ((105.0(REAL32)*xy)*RM9)-((15.0(REAL32)*y)*RM7)  )

          ScaleitD:=((945.0(REAL32)*x2yz)*RM11)-((105.0(REAL32)*yz)*RM9)
          ScaleitS:=-VaddD[2]
        -}}}
      n=2
        -{{{  2 2
        SEQ
          VaddS[0]:=((210.0(REAL32)*y2x)*RM9)-((30.0(REAL32)*x)*RM7)
          VaddS[1]:=((210.0(REAL32)*x2y)*RM9)-((30.0(REAL32)*y)*RM7)
          -- dipole
          VaddD[0]:=((1890.0(REAL32)*y2xz)*RM11)-((210.0(REAL32)*xz)*RM9)
          VaddD[1]:=((1890.0(REAL32)*x2yz)*RM11)-((210.0(REAL32)*yz)*RM9)
          VaddD[2]:=0.0(REAL32)-( (((105.0(REAL32)*RM9)*(x2+y2))-
                  ((945.0(REAL32)*x2y2)*RM11))-(15.0(REAL32)*RM7)  )

          ScaleitD:=(((945.0(REAL32)*z)*(RM11*(x2+y2)))-((210.0(REAL32)*z)*RM9))-
                  ((945.0(REAL32)*x2y2z)*RM13)
          ScaleitS:=-VaddD[2]

        -}}}
      TRUE
        SKIP
    -}}}
  m=3
    -{{{
    IF
      n=0
        -{{{  3 0
        SEQ
          VaddS[0]:=(9.0(REAL32)*RM5)-((30.0(REAL32)*x2)*RM7)

          -- dipole

          VaddD[0]:=((45.0(REAL32)*z)*RM7)-((210.0(REAL32)*x2z)*RM9)
```

```
          VaddD[2]:=0.0(REAL32)-( ((105.0(REAL32)*x3)*RM9)-((45.0(REAL32)*x)*RM7)   )
          ScaleitD:=((945.0(REAL32)*x3z)*RM11)-((315.0(REAL32)*xz)*RM9)
          ScaleitS:=-VaddD[2]
        -}}}
      n=1
        -{{{  3 1
        SEQ
          VaddS[0]:=((315.0(REAL32)*x2y)*RM9)-((45.0(REAL32)*y)*RM7)
          VaddS[1]:=((105.0(REAL32)*x3)*RM9)-((45.0(REAL32)*x)*RM7)
          -- dipole

          VaddD[0]:=((2835.0(REAL32)*x2yz)*RM11)-((315.0(REAL32)*yz)*RM9)
          VaddD[1]:=((945.0(REAL32)*x3z)*RM11)-((315.0(REAL32)*xz)*RM9)
          VaddD[2]:=0.0(REAL32)-( ((315.0(REAL32)*xy)*RM9)-((945.0(REAL32)*x3y)*RM11) )

          ScaleitD:=((2835.0(REAL32)*xyz)*RM11)-((10395.0(REAL32)*x3yz)*RM13)
          ScaleitS:=-VaddD[2]

        -}}}
      TRUE
        SKIP
    -}}}
  m=4
    -{{{
    IF
      n=0
        -{{{  4 0
        SEQ
          VaddS[0]:=((420.0(REAL32)*x3)*RM9)-((180.0(REAL32)*x)*RM7)

          -- dipole
          VaddD[0]:=((3780.0(REAL32)*x3z)*RM11)-((1260.0(REAL32)*xz)*RM9)
          VaddD[2]:=0.0(REAL32)-( (((630.0(REAL32)*x2)*RM9)-((945.0(REAL32)*x4)*
                RM11))-(45.0(REAL32)*RM7)  )

          ScaleitD:=(((5670.0(REAL32)*x2z)*RM11)-((10395.0(REAL32)*x4z)*RM13))-
                ((315.0(REAL32)*z)*RM9)

          ScaleitS:=-VaddD[2]
        -}}}
      TRUE
        SKIP
    -}}}
  TRUE
    SKIP
  ScalarMul(ScaleitD,temp)
  VectorAdd(temp,VaddD,Qden)
  ScalarMultiply(ScaleitS,Pen,temp)
  VectorAdd(temp,VaddS,Qsen)
:
-}}}
-{{{  variables
REAL32 R2,x,y,z:
REAL32 RM1,RM3,RM5:
REAL32 xz,RM7,yz:
REAL32 RM9,x2,x2z,xy,xyz,y2,y2z:
REAL32 RM11,x3,x3z,x2y,x2yz,y2x,y2xz,y3,y3z:
REAL32 RM13,x4,x4z,x3y,x3yz,x2y2z,y4,y4z,y3x,y3xz,x2y,y2x,x2y2:
-}}}
SEQ
  -{{{  calculate parameters
```

```
R2:=radius*radius
x:=Pen[0]
y:=Pen[1]
z:=Pen[2]
--{{{  ordur=0
IF
  ordur>=0
    SEQ
      RM1:=1.0(REAL32)/radius
      RM3:=RM1/R2
      RM5:=RM3/R2
  TRUE
    SKIP
--}}}
--{{{  ordur=1
IF
  ordur>=1
    SEQ
      xz:=x*z
      RM7:=RM5/R2
      yz:=y*z
  TRUE
    SKIP
--}}}
--{{{  ordur=2
IF
  ordur>=2
    SEQ
      RM9:=RM7/R2
      x2:=x*x
      x2z:=x2*z
      xy:=x*y
      xyz:=xy*z
      y2:=y*y
      y2z:=y2*z
  TRUE
    SKIP
--}}}
--{{{  ordur=3
IF
  ordur>=3
    SEQ
      RM11:=RM9/R2
      x3:=x2*x
      x3z:=x3*z
      x2y:=x2*y
      x2yz:=x2y*z
      y2x:=y2*x
      y2xz:=y2x*z
      y3:=y2*y
      y3z:=y3*z
  TRUE
    SKIP
--}}}
--{{{  ordur=4
IF
  ordur>=4
    SEQ
      RM13:=RM11/R2
      x4:=x3*x
      x4z:=x4*z
```

```
        x3y:=x3*y
        x3yz:=x3y*z
        x2y2:=x2*y2
        x2y2z:=x2y2*z
        y4:=y3*y
        y4z:=y4*z
        y3x:=y3*x
        y3xz:=y3x*z
      TRUE
        SKIP
    -}}}
    -}}}
    -{{{  initialise Dipole
    SEQ i=0 FOR 3
      SEQ
        Qden[i]:=0.0(REAL32)
        Qsen[i]:=0.0(REAL32)
    -}}}
    SEQ m=0 FOR  (ordur+1)
      SEQ n=0 FOR (ordur+1)
        SEQ
          IF
            (m+n)<=ordur
              SEQ
                IF
                  Imn[m][n]<>0.0(REAL32)
                    -{{{  calculate coefficients
                    [3]REAL32 Qs,Qd:
                    SEQ
                      VCalculateDipole(R2,x,y,z,
                                       RM1,RM3,RM5,
                                       xz,RM7,yz,
                                       RM9,x2,x2z,xy,xyz,y2,y2z,
                                       RM11,x3,x3z,x2y,x2yz,y2x,y2xz,y3,y3z,
                                  x4,x4z,x3y,x3yz,x2y2z,y4,y4z,y3x,y3xz,x2y2,RM13,
                                  m,n,Pen,Qs,Qd)
                      ScalarMul(Imn[m][n],Qs)
                      ScalarMul(Imn[m][n],Qd)
                      VectorAdd(Qs,Qsen,Qsen)
                      VectorAdd(Qd,Qden,Qden)
                    -}}}
                  TRUE
                    SKIP
            TRUE
              SKIP
    ScalarMul((0.0(REAL32)-0.079577471(REAL32)),Qden)
    ScalarMul((0.0(REAL32)-0.079577471(REAL32)),Qsen)
:
-}}}
-}}}

-{{{  Exact
-{{{  Vectexact
PROC VectExactPancl(VAL BOOL Both,
                    VAL REAL32 area,radius,choice,VAL [3]REAL32 P,
                    VAL [4]REAL32 E,N,s,dE,dN,
                    [3]REAL32 Qsen,Qden)
  #USE snglmath
  -{{{  variables
  INT n:
  REAL32 Vt,thet,z2,x,y,z,dipole,ScaleIt:
```

```
[4]REAL32 L,dL,U,R,s1,c1,s2,c2,s3,c3,Fact:
[3]REAL32 Q1,A1,A2,A3,A4,Q,GradC1,GradC2,SD1,SD2,GradS1,GradS2:
BOOL corner,toosmall:
-}}}
SEQ
  z:=P[2]
  x:=P[0]
  y:=P[1]
  z2:=z*z
  SEQ m=0 FOR 4
    SEQ
      -{{{
      R[m]:=POWER(ABS( (((x-E[m])*(x-E[m]))+((y-N[m])*(y-N[m])))+z2),
              0.5(REAL32))
      L[m]:=POWER(ABS((dE[m]*dE[m])+(dN[m]*dN[m])),0.5(REAL32))
      -}}}
  SEQ m=0 FOR 4
    SEQ
      IF
        m=3
          n:=0
        TRUE
          n:=m+1
      s1[m]:=(dN[m]*(((x-E[m])*(x-E[m]))+z2))-
              (dE[m]*((x-E[m])*(y-N[m])))
      s2[m]:=(dN[m]*(((x-E[n])*(x-E[n]))+z2))-
              (dE[m]*((x-E[n])*(y-N[n])))
      dL[m]:=(dN[m]*(x-E[m]))-(dE[m]*(y-N[m]))
      c1[m]:=R[m]*(z*dE[m])
      c2[m]:=R[n]*(z*dE[m])
  dipole:=0.0(REAL32)
  -{{{  dipole potential Qd
  SEQ m=0 FOR 4
    SEQ
      c3[m]:=(c1[m]*c2[m])+(s1[m]*s2[m])
      s3[m]:=(s1[m]*c2[m])-(s2[m]*c1[m])
      IF
        (ABS(c3[m])<=1.0E-09(REAL32)) AND (ABS(s3[m])<=1.0E-09(REAL32))
          SKIP
        TRUE
          dipole:=dipole+ATAN2(c3[m],s3[m])
  SEQ i=0 FOR 3
    Qden[i]:=0.0(REAL32)
  -{{{  dipole potential derivative  Qden
  -  Calculate dipole potential derivative
  SEQ m=0 FOR 4
    REAL32 denom1,denom2,ScaleIt1,ScaleIt2,ScaleIt3,ScaleIt4:
    [3]REAL32 GC1,GC2,GS1,GS2:
    SEQ
      IF
        m=3
          n:=0
        TRUE
          n:=m+1
      -{{{  new scaleit  13/11/91
      -{{{  (s1s) and (s2s)
      Q1[0]:=0.0(REAL32)
      Q1[1]:=0.0(REAL32)
      IF
        (R[m]<(0.05(REAL32)*L[m])) OR (R[n]<(0.05(REAL32)*L[m]))
          SEQ
```

```
    corner:=TRUE
    Q1[2]:=0.0(REAL32)
  TRUE
   SEQ
    denom1:=((s1[m]*s1[m])+(c1[m]*c1[m]))
    denom2:=((s2[m]*s2[m])+(c2[m]*c2[m]))
    --{{{  calculate GradC1  mod
    IF
     ABS(z*dE[m])<ABS(R[m])
       ScaleIt:=(z*dE[m])/R[m]
     TRUE
       IF
        ABS(R[m])<1.0E-06(REAL32)
          ScaleIt:=0.0(REAL32)
        TRUE
          ScaleIt:=(z*dE[m])/R[m]

    ScalarMultiply(ScaleIt,P,GradC1)

    GradC1[0]:=GradC1[0]-(ScaleIt*E[m])

    GradC1[1]:=GradC1[1]-(ScaleIt*N[m])

    GradC1[2]:=GradC1[2]+(R[m]*dE[m])

    ScalarMultiply(s1[m],GradC1,GC1)
    --}}}
    IF
     (ABS(GC1[2])<denom1) OR (ABS(z)>1.0E-04(REAL32))
       SEQ
        --{{{
        --{{{  calculate GradC2  mod
        IF
          ABS(z*dE[m])<ABS(R[n])
            ScaleIt:=(z*dE[m])/R[n]
          TRUE
            IF
              ABS(R[n])<1.0E-06(REAL32)
                ScaleIt:=0.0(REAL32)
              TRUE
                ScaleIt:=(z*dE[m])/R[n]

        ScalarMultiply(ScaleIt,P,GradC2)


        GradC2[0]:=GradC2[0]-(ScaleIt*E[n])

        GradC2[1]:=GradC2[1]-(ScaleIt*N[n])


        GradC2[2]:=GradC2[2]+(R[n]*dE[m])
        ScalarMultiply(s2[m],GradC2,GC2)
        --}}}
        --{{{  calculate GradS1  mod
        GradS1[0]:=((2.0(REAL32)*(x-E[m]))*dN[m])-
                  ((dE[m]*(y-N[m])))

        GradS1[1]:=dE[m]*(E[m]-x)

        GradS1[2]:=dN[m]*(z*2.0(REAL32))
        ScalarMultiply(c1[m],GradS1,GS1)
```

```
            -}}}
            -{{{  calculate GradS2  mod
            GradS2[0]:=((2.0(REAL32)*(x-E[n]))*dN[m])-
                    ((dE[m]*(y-N[n])))

            GradS2[1]:=dE[m]*(E[n]-x)

            GradS2[2]:=dN[m]*(z*2.0(REAL32))
            ScalarMultiply(c2[m],GradS2,GS2)
            -}}}
            -}}}
            -{{{  normal
            ScaleIt1:=1.0(REAL32)/denom1
            ScalarMultiply(ScaleIt1,GC1,A1)
            ScaleIt3:=1.0(REAL32)/denom1
            ScalarMultiply(ScaleIt3,GS1,A3)
            ScaleIt2:=1.0(REAL32)/denom2
            ScalarMultiply(ScaleIt2,GC2,A2)
            ScaleIt4:=1.0(REAL32)/denom2
            ScalarMultiply(ScaleIt4,GS2,A4)
            -{{{  adds
            VectorSub(A2,A1,SD1)
            VectorSub(A3,A4,SD2)
            VectorAdd(SD1,SD2,Q1)
            -}}}
            -}}}
          TRUE
            SEQ
              IF
                ABS(dL[m])<(0.0001(REAL32)*L[m])
                  Q1[2]:=0.0(REAL32)
                TRUE
                  IF
                    (s1[m]*s2[m])=0.0(REAL32)
                      Q1[2]:=0.0(REAL32)
                    TRUE
                      Q1[2]:=(dE[m]*((s1[m]*R[n])-(s2[m]*R[m])))/(s1[m]*s2[m])
    -}}}
    -}}}
    SEQ i=0 FOR 3
      Qden[i]:=Qden[i]+Q1[i]
-}}}
ScalarMul((0.0(REAL32)-0.079577471(REAL32)),Qden)
-}}}
IF
  Both=TRUE
    -{{{  source potential Qs
    IF
      ABS(choice)<0.00001(REAL32)
        -{{{
        SEQ
          SEQ i=0 FOR 2
            Qsen[i]:=0.0(REAL32)
          IF
            z>0.0(REAL32)
              Qsen[2]:=-0.5(REAL32)
            TRUE
              Qsen[2]:=0.5(REAL32)
        -}}}
      TRUE
        SEQ
```

```
--{{{  source potential derivative  Qsen
SEQ m=0 FOR 3
  SEQ
    A1[m]:=0.0(REAL32)

SEQ m=0 FOR 4
  REAL32 Logit,CON,SON,numerator,denominator:
  [3]REAL32 unitR1,unitR2,RT:
  SEQ
    IF
      m=3
        n:=0
      TRUE
        n:=m+1

    SON:=dN[m]/s[m]
    CON:=dE[m]/s[m]
    RT[0]:=x-E[m]
    RT[1]:=y-N[m]
    RT[2]:=z
    numerator:=(2.0(REAL32)*s[m])*((RT[1]*CON)-(RT[0]*SON))
    denominator:=((R[m]+R[n])*(R[m]+R[n]))-(s[m]*s[m])
    IF
      (ABS(s[m])<1.0E-09(REAL32)) OR (ABS(denominator)<=1.0E-09(REAL32))
        ScaleIt:=0.0(REAL32)
      TRUE
        ScaleIt:=numerator/denominator
    IF
      ((R[m]+R[n])-s[m])<1.0E-10(REAL32)
        Logit:=0.0(REAL32)
      TRUE
        Logit:=ALOG(((R[m]+R[n])+s[m])/((R[m]+R[n])-s[m]))


    A1[0]:=A1[0]+(Logit*SON)
    A1[1]:=A1[1]-(Logit*CON)
    IF
      ScaleIt=0.0(REAL32)
        SKIP
      TRUE
        SEQ
          ScalarDivide(R[m],P,unitR1)
          ScalarDivide(R[n],P,unitR2)

          VectorAdd(unitR1,unitR2,A2)

          ScalarMul(ScaleIt,A2)

          A3[0]:=ScaleIt*((E[m]/R[m])+(E[n]/R[n]))
          A3[1]:=ScaleIt*((N[m]/R[m])+(N[n]/R[n]))
          A3[2]:=0.0(REAL32)

          SEQ i=0 FOR 3
            A1[i]:=A1[i]+(A2[i]-A3[i])


A1[2]:=A1[2]-dipole

ScalarMultiply(z,Qden,Q)

VectorSub(A1,Q,Qsen)
```

```
          IF
            ABS(z)<1.0E-09(REAL32)
              Qsen[2]:=0.0(REAL32)
            TRUE
              SKIP

          -}}}
          ScalarMul((0.0(REAL32)-0.07/9577471(REAL32)),Qsen)
      -}}}
    TRUE
      SEQ l=0 FOR 3
        Qsen[l]:=0.0(REAL32)
:
-}}}

-{{{  exact
PROC ExactPanel(VAL BOOL Both,VAL REAL32 area,radius,choice,VAL [3]REAL32 P,
            VAL [4]REAL32 E,N,s,dE,dN,
            REAL32 Qs,Qd)
  #USE snglmath
  -{{{  variables
  INT n:
  REAL32 thet,z2,x,y,z:
  [4]REAL32 R,s1,c1,s2,c2,s3,c3,Fact:
  BOOL toosmall:
  -}}}
  SEQ
    z:=P[2]
    x:=P[0]
    y:=P[1]
    z2:=z*z
    SEQ m=0 FOR 4
      SEQ
        -{{{
        R[m]:=POWER(ABS( (((x-E[m])*(x-E[m]))+((y-N[m])*(y-N[m])))+z2),
                0.5(REAL32))
        -}}}
    SEQ m=0 FOR 4
      SEQ
        IF
          m=3
            n:=0
          TRUE
            n:=m+1
        s1[m]:=(dN[m]*(((x-E[m])*(x-E[m]))+z2))-
                (dE[m]*((x-E[m])*(y-N[m])))
        s2[m]:=(dN[m]*(((x-E[n])*(x-E[n]))+z2))-
                (dE[m]*((x-E[n])*(y-N[n])))
        c1[m]:=R[m]*(z*dE[m])
        c2[m]:=R[n]*(z*dE[m])
    IF
      ABS(choice)<0.000001(REAL32)
        SEQ
          IF
            choice<0.0000001(REAL32)
              Qd:=-0.5(REAL32)
            TRUE
              Qd:=0.5(REAL32)
      TRUE
        SEQ
          IF
```

```
        z=0.0(REAL32)  -- in plane of dipole
          Qd:=0.0(REAL32)
        TRUE
          SEQ
            -{{{  dipole potential Qd
            Qd:=0.0(REAL32)
            SEQ m=0 FOR 4
              SEQ
                c3[m]:=(c1[m]*c2[m])+(s1[m]*s2[m])
                s3[m]:=(s1[m]*c2[m])-(s2[m]*c1[m])
                Qd:=Qd+ATAN2(c3[m],s3[m])
            Qd:=(0.0(REAL32)-0.079577471(REAL32))*Qd
            -}}}
  Qs:=0.0(REAL32)
  IF
    Both=TRUE
      SEQ
        -{{{  source potential Qs
        SEQ m=0 FOR 4
          REAL32 Logit,CON,SON,numerator:
          [3]REAL32 RT:
          REAL32 fact:
          SEQ
            IF
              m=3
                n:=0
              TRUE
                n:=m+1
            SON:=(dN[m]/s[m])
            CON:=(dE[m]/s[m])
            RT[0]:=x-E[m]
            RT[1]:=y-N[m]
            numerator:=((RT[0]*SON)-(RT[1]*CON))
            fact:=(R[m]+R[n])/s[m]
            fact:=(ABS(fact-1.0(REAL32)))+1.0E-15(REAL32)
            fact:=1.0(REAL32)+(2.0(REAL32)/fact)
            Logit:=ALOG(fact)
            Qs:=Qs+(numerator*Logit)
        Qs:=(Qs*(0.0(REAL32)-0.079577471(REAL32))) - (z * Qd)
        -}}}
    TRUE
      SKIP
:
-}}}
--}}}
```