EMBEDDED MESH MULTIGRID TREATMENT OF TWO-DIMENSIONAL TRANSONIC FLOWS

D. R. McCarthy*

Purdue University
West Lafayette, Indiana
and
Indiana University - Purdue University
Fort Wayne, Indiana

and

R. C. Swanson

NASA - Langley Research Center
Hampton, Virginia

Presented at the
International Multigrid Conference
Copper Mountain, CO
April 1983

* Work of this author supported by a grant from NASA-Langley Research Center, Hampton, VA.

Abstract

Embedded Mesh Multigrid Treatment of Two-Dimensional Transonic Flows

D. R. McCarthy, Purdue University
R. C. Swanson, NASA - Langley Research Center

We report the development of a highly flexible computer code which uses a hierarchy of embedded meshes, communication via multigrid, to solve the axisymmetric or two-dimensional viscous transonic flow problem. The purpose of the code is to demonstrate the practicality of techniques sufficiently general to be applicable to complex three-dimensional geometries, and to serve as a test vehicle for mesh communication and relaxation procedures. The code is also of interest in its own right for the fast solution of planer and axisymmetric problems in a very large domain. It consists of a multigrid solver for the full-potential equation (which incorporates the embedded meshes), with a provision for later coupling velocity injections to a separate boundary layer program.

The principal feature of the code is its treatment of the geometry by means of several levels of locally defined body-fitted meshes embedded in a system of coarser global grids which do not conform to the body. The user may define meshes as large and coarse as desired, in order to solve the problem in a large domain, and also as fine as desired, in order to resolve body geometry and flow field details. The grids need not be coextensive, so that mesh lines placed close to detail features need not extend throughout the field. The intent throughout has been to do nothing whose extension to three- dimensions is not straightforward; in particular, global mappings are avoided. We have also emphasized flexible data structures and modular code, with a view toward the development of a highly vectorized three-dimensional version.

I. INTRODUCTION

Among the phenomena which currently impede progress in computational fluid dynamics, the following are frequently cited:

(1) the slow rate of convergence of iterative schemes,

(2) the varying length scales, throughout the flow field, of the features of interest, and

(3) the difficulties of describing and resolving complex three-dimensional geometries.

Over the past decade or more a good deal of effort has been lavished, quite successfully, on (1), and many effective methods have been developed. Of these, the multi-grid method has now become widely recognized as a fast iterative solver for elliptic problems, and to some extent for mixed elliptic-hyperbolic problems as well. It has, however, received rather less attention as an approach to (2) and (3). Nevertheless, as faster algorithms and faster computation times become the rule, the demand for solutions to three-dimensional flow problems involving complex geometries must inevitably become the primary focus.

The purpose of the current undertaking was to develop a vehicle for the testing of grid manipulation procedures sufficiently general to treat arbitrary three-dimensional geometries. To this end, we have produced a computer program to solve the two-dimensional (including axisymmetric) transonic potential flow equation in conservation form. The code includes a provision for coupling to a separate boundary layer program via mass injection.

Our hypothesis is that the multigrid method can furnish the underlying philosophy for communication within a hierarchy of embedded (i.e., non-coextensive) grids. While certain assumptions have been made to simplify coding, we place as few restrictions as possible on the definition and placement of grids. As the objective is experiment, the code emphasizes flexibility and modularity rather than speed. In particular, it has been designed so that such features as smoothing algorithms and interpolation types are easily changed. It is also equipped to provide a variety of diagnostic information such as static residuals and truncation error estimates.

The code may be viewed as the precursor of a fully three-dimensional version to be developed for the CYBER 205. Thus we have put primary emphasis on procedures whose generalization to three dimensions is straight forward, and on vectorizable algorithms.

II. EMBEDDED MESH

In the numerical calculation of fluid flows over bodies of complex shape, the selection of a computational mesh poses difficult problems. The mesh must provide high density in areas of interest near the body and near the probable locations of other features, such as shocks, and resolve the body itself accurately. Conversely, it must cover a computational domain large enough to justify an assumption of far field conditions at the outer boundary, and yet remain economically feasible for calculations.

A common approach has been to introduce a curvilinear grid which conforms to the body surface, so that the entire domain is mapped to a simple shape, such as a rectangle or a circle. This approach simplifies the treatment of boundary conditions. However, the design of such a transformation is itself a difficult problem, even in two dimensions, and must be repeated for each new geometric configuration. For complicated three-dimensional objects, no transformation may be satisfactory.

An alternative is to employ a more regular, (e.g., Cartesian) grid without attempting to fit the body. However, one must then be prepared to treat a plethora of ugly cases as the mesh lines intersect the body in a haphazard fashion. In three dimensions the number of types of such intersections could be very large. [8]

Whether or not the mesh is fitted to the body, it is often desirable to allow unequal mesh spacings so as to permit greater mesh densities in locations where greater resolution is desired. Unfortunately, if all grid lines begin and end on boundaries, such density naturally projects itself into regions where it is not only unnecessary, but may be undesirable by virtue of the formation of mesh cells of large aspect ratio, which can interfere with smoothing and relaxation rates. The current work is based on the observation that the difficulties mentioned above result from the attempt to define globally a single grid under several contradictory restrictions. Instead we define a hierarchy of grids. The coarsest of these is global, covering the whole domain with minimal regard for the presence of the body, and the finest are local and body-fitted, i.e., of small spatial extent and resolving the body and the flow near it by conforming to local coordinates established on the body surface. Since the number and spatial extent of such grids at any level of the hierarchy is in principal unlimited, the system should have the flexibility to resolve arbitrary geometries.

While the notion of local mesh refinements is now new, the problem has always been that fine grids covering different spatial regions are in essence decoupled. In ordinary schemes, fine grids receive information from coarse ones, but not conversely. Thus fine grids at the same level cannot communicate, even if they overlap. Clearly, such a scheme cannot model physics properly. The introduction of the multigrid method provides fine-to-coarse information transfer, insuring that the problem is consistently formulated on all levels and grids.

As a test case we treat the axisymmetric configuration known as Reubush Configuration #3, depicted in Figure 1, for which extensive experimental data are available. Left to right, the configuration consists of a conical nose, followed by a cylindrical body, and terminating in a circular-arc "boattail" afterbody. Behind the body on the extreme right the jet plume is simulated by a solid cylinder. The region of particular interest is on the afterbody which greatly influences the drag. At the juncture between the afterbody and the solid plume simulator, there is a discontinuity in the tangent to the body surface, where inviscid theory predicts an infinite pressure spike. In practice, viscous effects tend to smooth this discontinuity, as well as that at the nose; therefore such discontinuities have been eliminated by inserting circular fairings of small radius. All units in the diagram are normalized so that the maximum body diameter is 1.0.

The test code provides for the definition of two types of grids: Cartesian grids, which are necessarily rather coarse, and are not body-fitted, and body-fitted grids, which may be as fine as desired, and are created by a simple vertical shearing of coordinates. Figure 2 depicts a typical hierarchy of Cartesian

grids on 3 levels; the code would designate these as levels 1, 2, and 3. The mesh are square; if h_k is the mesh size at level k, then $h_{k+1} = 2h_k$. Each h_k is large compared to the body diameter in order to limit the possible types of intersection of coordinate lines with the body: we require that the minimum mesh size of an unfitted grid be at least twice the body radius. As the axis is a horizontal coordinate line, only vertical coordinate lines can intersect the body surface. Figure 3 depicts a hierarchy of fitted grids, the coarsest of which has the same mesh size as the finest unfitted grid. The code would designate these as levels 4, 5, and 6, so that $h_4 = h_3$, and otherwise $h_{k+1} = 2h_k$.

While this situation is not completely general it simplifies both input and grid manipulation and permits testing of most questions of interest. In a general environment, it would devolve upon the user to specify the shapes and extent of the desired grids, say via mappings or data, as well as their levels and mesh sizes. Here one need only specify the number of levels of each type, the mesh size of the finest Cartesian level, the level of each grid and its left, right, and outer boundaries. Several restrictions are imposed:

(1) The mesh intersections within any grid must constitute a rectangular array;

(2) Each grid of level k, $k>1$, must be contained in a grid of level $k-1$, called its "parent";

(3) If any part of a cell of any level k grid is refined on level $k+1$, then the whole cell must be refined;

(4) No two grids on the same level may overlap.

If conditions (2) and (3) are not met by the requested grid boundaries, the code will attempt to satisfy them by expanding grids as necessary. All grid numbering and storage allocation functions are automatic.

It may be noted that, owing to the simplicity of the geometry, it is feasible here to allow the entire hierarchy of grids to be body fitted. In fact, as the mesh size grows large, the difference between fitted and unfitted grids diminishes, so that the coarsest grids may as well be fitted to the body. This is no doubt advantageous where it is possible. However, as our purpose was to create a test vehicle for situations where interpolations between grids of different types might be necessary, we have provided for both possibilities. In any event, fully fitted grids at all levels are included as a special case.

III. PROBLEM FORMULATION

We solve the transonic potential equation in conservation form

$$(\rho u r)_x + (\rho v r)_y = 0 \tag{1}$$

where

$$u = \frac{\partial \varphi}{\partial x}, \quad v = \frac{\partial \varphi}{\partial y} \tag{2}$$

are the velocities in the (Cartesian) x and y directions, φ is the velocity potential, ρ is the density, and r is either equal to one, in the case of plane flow, or

equal to y, the distance from the axis, in the axisymmetric case. The free-stream velocity is denoted by (u_∞, v_∞); we assume that $v_\infty = 0$ and scale φ so that $u_\infty = 1$.

Via the assumption of isentropic flow the density ρ may be calculated as a function of u and v. If a_∞ is the free stream speed of sound, and $M_\infty = 1/a_\infty$, then

$$\rho^{\gamma-1} = 1 - \frac{\gamma-1}{2} \cdot M_\infty^2 \,(u^2 + v^2 - 1), \tag{3}$$

where γ is the ratio of specific heats.

On sheared grids equation (1) is transformed as follows. We assume the body surface to be given by a curve of the form $y = f(x)$; sheared coordinates ξ and η are defined by

$$\begin{aligned} \xi &= x \\ \eta &= y - f(x) \end{aligned} \tag{4}$$

whereby (1) becomes

$$(\rho u\, r)_\xi + (\rho(v - mu)r)_\eta = 0, \tag{5}$$

where $m = f'(\xi)$, in terms of the (Cartesian) velocities u and v. These are obtained in turn from

$$\begin{aligned} u &= \frac{\partial\varphi}{\partial x} = \frac{\partial\varphi}{\partial\xi}\frac{\partial\xi}{\partial x} + \frac{\partial\varphi}{\partial\eta}\frac{\partial\eta}{\partial x} = \frac{\partial\varphi}{\partial\xi} - m\,\frac{\partial\varphi}{\partial\eta} \\ v &= \frac{\partial\varphi}{\partial y} = \frac{\partial\varphi}{\partial\xi}\frac{\partial\xi}{\partial y} + \frac{\partial\varphi}{\partial\eta}\frac{\partial\eta}{\partial y} = \frac{\partial\varphi}{\partial\eta} \end{aligned} \tag{6}$$

so that (5) is equivalent to

$$\left[\rho\left(\frac{\partial\varphi}{\partial\xi} - m\,\frac{\partial\varphi}{\partial\eta}\right)r\right]_\xi + \left[\rho\left(-m\frac{\partial\varphi}{\partial\xi} + (1+m^2)\,\frac{\partial\varphi}{\partial\eta}\right)r\right]_\eta = 0\,.$$

Again, $r = 1$ in the planar case, and $r = y$ in the axisymmetric case.

Boundary Conditions

On the axis and on the body we impose the free-slip condition $\partial\varphi/\partial n = 0$, where n is the normal to the axis or body surface. On the upper boundary of the largest grid we impose the free-stream condition $u = u_\infty$, so that $\varphi = u_\infty x + c$; we arbitrarily set $c = 0$. On the left (inflow) and right (outflow) boundaries we take $v = 0$, so that $\varphi = constant$; the constant is known from the value of φ on the upper boundary. Since the boundary is placed very far from the body, we expect that the exact form of the boundary conditions will have little influence on the solution. This formulation has the advantage of posing Dirichlet conditions on three sides of the domain.

Each grid except for the coarsest inherits Dirichlet boundary conditions on the left, right, and upper boundaries from its parent, while retaining the condition $\partial\varphi/\partial n = 0$ on the axis and body surface. Inherited boundary conditions are subject to change during multigrid cycling.

There is further a provision to incorporate an injection velocity from a separate boundary layer program, so that $\partial\varphi/\partial n$ becomes a specified function of position. This provision has so far not been exercised.

Finite Difference Formulas

The actual differencing formulas closely follow the scheme used by Green [4]. Let i and j represent discrete variables corresponding to ξ and η, respectively, and consider at the point (i,j) the formulation of a mass balance on the cell about i,j depicted in figure 4. The potential φ is initially specified at the mesh points $(i+\alpha, j+\beta)$, where $\alpha,\beta = -1, 0, 1$. One may calculate $\partial\varphi/\partial\xi$ at the mid-points of the cell sides via second order central differences of the type

$$(\partial\varphi/\partial\xi)_{i+\frac{1}{2}} = (\varphi_{i+1,j} - \varphi_{i,j})/h \ . \tag{8}$$

One obtains $\partial\varphi/\partial\eta$ at the same points by averaging the values of $\partial\varphi/\partial\eta$ calculated from second order central differences at the points $(i-1,j)$, (i,j), and $(i+1,j)$, i.e., via formulas of the type

$$(\partial\varphi/\partial\eta)_{i+\alpha,j} = (\varphi_{i+\alpha,j+1} - \varphi_{i+\alpha,j-1})/2h, \quad \alpha = 0,1 \tag{9}$$

followed by

$$(\partial\varphi/\partial\eta)_{i+\frac{1}{2},j} = ((\partial\varphi/\partial\eta)_{i+1,j} + (\partial\varphi/\partial\eta)_{i,j})/2 \ . \tag{10}$$

The derivatives of φ are available at $(i,j \pm \frac{1}{2})$ by the same procedure with the roles of ξ and η reversed. From these, one obtains $\partial\varphi/\partial\xi$ and $\partial\varphi/\partial\eta$ at $(i \pm \frac{1}{2}, j \pm \frac{1}{2})$ by averaging, i.e.,

$$\begin{aligned}(\partial\varphi/\partial\xi)_{i+\frac{1}{2},j+\frac{1}{2}} &= ((\partial\varphi/\partial\xi)_{i+\frac{1}{2},j} + (\partial\varphi/\partial\xi)_{i+\frac{1}{2},j+1})/2 \\ (\partial\varphi/\partial\eta)_{i+\frac{1}{2},j+\frac{1}{2}} &= ((\partial\varphi/\partial\eta)_{i,j+\frac{1}{2}} + (\partial\varphi/\partial\eta)_{i+1,j+\frac{1}{2}})/2 \ .\end{aligned} \tag{11}$$

Note that all quantities on the right side of (11) are available from within the $3x3$ computational molecule centered at (i,j).

With the derivatives of φ available at the cell corners, the velocities u and v may be obtained from (6) and the density ρ from (3). In order to estimate the mass flux out of the cell, one requires the densities ρ at each cell side; these are obtained by averaging the densities at the corners, as in

$$\rho_{i+\frac{1}{2},j} = (\rho_{i+\frac{1}{2},j-\frac{1}{2}} + \rho_{i+\frac{1}{2},j+\frac{1}{2}})/2 \ . \tag{12}$$

The efflux through the right side is estimated as

$$\rho_{i+\frac{1}{2},j} \ u_{i+\frac{1}{2},j} \ r_{i+\frac{1}{2},j} \tag{13}$$

and that through top of the cell as

$$\rho_{i,j+\frac{1}{2}}\,(v_{i,j+\frac{1}{2}} - m_i u_{i,j+\frac{1}{2}})r_{i,j+\frac{1}{2}}. \tag{14}$$

With similar quantities subtracted as the influx through the left side and bottom, the scheme amounts to a conservative second-order differencing of (5).

Some modification is necessary for points on the body or axis. The situation for a point $(i,0)$ on the body in a sheared grid is shown in Figure 5. The vertical velocities $\partial\varphi/\partial\eta$ at points $(i\pm\frac{1}{2},0)$ are obtained from the boundary condition

$$v_{i+\frac{1}{2},0} = m_{i+\frac{1}{2}}\ u_{i+\frac{1}{2},0} \tag{15}$$

rather than from (11). The densities $\rho_{i\pm\frac{1}{2},0}$ are found directly from u and v, and the fluxes through the sides are estimated in the form $\rho_{i\pm\frac{1}{2},0}\ u_{i\pm\frac{1}{2},0}\ r_{i\pm\frac{1}{2},0}$. The density $\rho_{i,0}$ is given by

$$\rho_{i,0} = (\rho_{i-\frac{1}{2},0} + \rho_{i+\frac{1}{2},0})/2 \tag{16}$$

and the flux through the body surface is $\rho_{i,0}\ v_{inj}\ r_{i,0}$, where v_{inj} is an injection velocity supplied from a separate boundary layer program. For all results reported here, $v_{inj} = 0$. For a point on the body in a Cartesian grid, depicted in Figure 6, the situation is more complex, and second order accuracy cannot be achieved without increasing the size of the molecule. However, such grids are ordinarily quite coarse, and formal truncation error estimates are not relevant since h is large. What is required is a reasonable estimate of mass flux in the large. The scheme is the same as in the sheared case, except that

$$(\partial\varphi/\partial\eta)_{i,\frac{1}{2}} = (\varphi_{i,1} - \varphi_{i,0})/(h-l) \tag{17}$$

which is not second order. Likewise, the averages (11) must be weighted to account for the unequal distances involved, and flux estimates multiplied by appropriate lengths.

While providing second order accuracy with a $3x3$ molecule, this scheme has the advantage of calculating the density only at cell corners; note that there are half as many cell corners as cell sides.

It should be noted that the slope $m = f'(\xi)$ can be found either analytically, since f is presumed known, or by differencing relevant values of f. We employ the latter, reasoning as follows. Consider, for instance, the value of m at $x=0$, at the nose of the body. If this value is obtained analytically, it will be estimated from the short circular fairing provided in that region and may thus vary rather arbitrarily between values of 0.0 and 0.25. Nevertheless, for coarse grids, its function is to describe the shape of a large cell which has no relationship to the locally defined analytic derivative $f'(0)$.

Alternatively, consider the cell depicted in Figure 4. If the boundaries follow the coordinate lines $\eta\pm d\eta = constant$, $\xi\pm d\xi = constant$, then the (continuous) estimate of the flux through the top boundary is (for the planar case)

$$\int_{\xi-d\xi}^{\xi+d\xi} \rho\cdot(v - f'(s)u)\,ds\ , \tag{18}$$

where ρ, v, and u are evaluated at $(\xi,\eta+d\eta)$, as in the estimates (14). However, (18) is merely

$$\rho \cdot (v - m\,u) \cdot 2\,d\,\xi \tag{19}$$

where

$$m = \frac{1}{2d\xi}\int_{\xi-d\xi}^{\xi+d\xi} f\,'(s)ds = \frac{f(\xi+d\xi) - f(\xi - d\xi)}{2d\xi}. \tag{20}$$

Thus utilization of (20) to calculate m is indicated for cells with curvilinear boundaries.

Shifted Density

At supersonic points the calculations are stabilized by upwind evaluation of the density, as suggested in [5]. If the local Mach number is M, we define a switching factor

$$\mu = \max\,(1 - \frac{1}{M^2}, 0) \tag{21}$$

and replace $\rho_{i,j}$ with

$$\hat{\rho}_{i,j} = \mu\rho_{i-1,j} + (1-\mu)\rho_{i,j}\ . \tag{22}$$

This corresponds to a shift in the $-\xi$ direction, which, for the flows considered here, corresponds roughly with the upwind direction.

A difficulty arises here because of the embedded mesh. In such problems the inflow boundary is normally subsonic; however, where grids of small spatial extent are introduced the probability is that certain inflow boundaries will contain supersonic points. Since such grids are generally finer than their parent grids, upwind densities are not directly available. Presumably they could be interpolated from coarser grids, but this has not yet been attempted here. Thus we must temporarily insist that no grid may have its inflow (left) boundary inside a supersonic zone. This restriction is rather limiting, since in supercritical cases the flow at the beginning of the boattail is supersonic. This forces fine grids to extend far forward on the body and limits the resolution obtainable with the present code.

IV. SOLUTION PROCEDURE

The Multigrid Cycle

We have implemented the multigrid procedure described by Brandt [1,2]; we describe the situation here in general terms, referring the reader to [2] for details.

The principal difference between the multigrid method for embedded mesh and that for coextensive grids lies in the dual role of the coarse grids. On that portion of any grid which underlies a finer one, the former plays the role of a correction grid for the latter. On the remainder, it plays the role of the finest grid. Since on this portion it must serve to define the finite difference solution to the original problem, so on that portion which serves as a correction grid it

must also describe the full solution, rather than a correction, to the original problem. Thus the full approximation storage (*FAS*) algorithm must be used. The use of *FAS* is also indicated for non-linear problems [1].

For embedded mesh problems, the full multigrid (*FMG*) algorithm provides convenient initialization on all levels. This algorithm begins by solving the problem on the coarsest level, level 1; having solved on level k, it interpolates an initial approximation to all grids at level $k+1$, and then executes solution at level $k+1$ via *FAS*. We initialize level 1 to uniform flow.

Coarse grids communicate information to fine ones by establishing (Dirichlet) boundary conditions for fine grid problems. Communication from fine grids to coarse ones takes place during grid coarsening, when the right hand sides of the coarse grid problems are adjusted by *FAS* to accommodate the relative truncation error between the two and thus enable the coarse grids to simulate the fine grid problems. Subsequent processing on the coarse grids permits adjustment of fine grid boundary values.

We provide several ways to control the multigrid cycling. The simplest of these provides for a fixed number of multigrid cycles at each level, with a fixed number of relaxation sweeps on each grid; these numbers may vary from level to level and grid to grid, and according to whether the relaxation follows grid coarsening or grid refinement. If the relaxation scheme smooths the error properly, and if the theoretical smoothing rate is known, then it should be possible to specify in advance the desired number of sweeps and cycles. Unfortunately, for non-linear problems and less than ideal smoothing procedures, it may not be possible to make such a prediction, since such rates depend on the evolving solution and vary from point to point throughout the field.

We also implement the "accommodative" procedure described by Brandt. This procedure attempts to monitor actual convergence rates and direct the calculations accordingly. When the convergence rate on any grid deteriorates past a user-defined value, calculations are transferred to the next coarser level. When convergence, as determined by a user-defined tolerance for the residual, is achieved on all grids at any level, calculations are transferred to the next finer level. This scheme, of course, requires the user to supply reasonable estimates of convergence rates. If the rate requested is too low (i.e., too fast), then grid coarsening will occur before high frequency error has been eliminated. This error will either not be seen by the coarse grid, or will be identified (i.e., "aliased") with low frequency error components. The coarse grid problem will then appear to be converged, possibly after some unnecessary work, and calculations will return to finer level with no improvement; the cycle then repeats. If, on the other hand, the requested rate is too high (i.e., slow), then unnecessary work is done.

In the context of embedded mesh, this scheme presents some theoretical difficulties as well with respect to the definition of convergence. In the case of coextensive grids, the problem to be solved is the fine grid problem; when the residual there is small, the solution is considered converged. (The multigrid procedure can, by estimating truncation error, supply some indication of what is "small", but the objective remains the same.) Thus the exact cycling procedure, assuming convergence is achieved, affects only the work done and not the solution itself. One may easily see, however, that in the case of embedded mesh, convergence cannot be defined in terms of residuals alone, whether on the fine grid or even the entire hierarchy. One must also be certain that the correct problem has been posed on each grid, that is, that each grid has communicated sufficient information to its parent that its inherited boundary values are

correct. Thus one should also insist that, at convergence, each coarse grid problem should be the correct one, that is, that a recalculation of its right hand side would not result in significant change. The natural criterion is that such change should be small compared to the relative truncation error, which itself may be estimated during grid coarsening. An extrapolation of this estimate also provides a natural convergence criterion for the unrefined portion of each grid [1]. The test code contains a rudimentary capability to monitor convergence in this way. However, it should be noted that the validity of the truncation error estimate depends on the assumption that the error on the fine grid is smooth when the estimate is made. Thus it is precisely in the circumstance in which convergence is difficult to achieve that such estimates may be much too large; indeed, that has been our experience so far.

Iterative Procedures

The solution by iterative methods of non-linear equations in general, and (1) in particular, may be regarded as consisting of two logical parts. First, some linearization is formed by presuming certain quantities to be known. Then some iteration is performed on the resulting linear system: This we term the inner iteration. Subsequently, the quantities previously presumed known, which occur as coefficients of the linear system, are updated to form a new linear system. This iteration, necessitated by the non-linearity, we call the outer iteration. For the equation (1), a natural linearization is obtained by calculating and freezing the density ρ. The resulting system of equations in the components of φ is linear because u and v are linear in φ and (1) is linear in u and v for fixed ρ. Other linearizations are of course possible.

In practice, the inner and outer iterations are often intermingled. Firstly, it is seldom regarded as useful to fully solve the linearized equations when the coefficients are only approximate. Usually the coefficients are updated as quickly as possible, or at least after each single full iterative step for the linearized problem. For line *SOR*, for example, updating of coefficients for each line normally preceeds the updating of unknowns for that line. Such procedures obscure the separate convergence and smoothing properties of the inner and outer iterations.

The distinction between inner and outer iterations also bears heavily on the vectorizability of the algorithm. If the coefficients for a full grid are updated simultaneously, for example, then full vectorization of the outer iteration is possible. For line *SOR*, as described above, it is not. In addition, the inner iteration for the unknown φ generally takes the form

$$N \cdot \Delta\varphi = R \tag{23}$$

where N is a matrix, $\Delta\varphi$ the change in the solution φ, and R the vector of residuals. If R and N can also be precalculated for the entire grid, the calculation of $\Delta\varphi$ can be simplified (and possibly vectorized, depending on the nature of N). Therefore, prior to each relaxation sweep, the test code provides precalculation over the entire grid of all densities, followed by residuals and molecular coefficients. These are then available to any relaxation scheme which can utilize them.

We have currently implemented four standard relaxation schemes, to which we refer as follows:

(1) Fully Implicit (*FI*). Here the frozen coefficient equations are solved directly by a LINPACK banded matrix solver.

(2) Gauss-Seidel Vertical Line Relaxation (*GSVLR*). For each vertical line, the densities and coefficients are updated and the resulting tridiagonal system is solved by LINPACK. The precalculated coefficients and residuals are not used.

(3) "Jacobi" Vertical Line Relaxation (*JVLR*). This is the same as GSVLR except that the precalculated coefficients and residuals are used.

(4) Point Jacobi (*PJ*). Each point is updated using the precalculated residual.

The *FI* method is supplied in order to completely eliminate the inner iteration. It is not particularly vectorizable. This type of solution is available at modest cost provided the sizes of grids, and their bandwidths, remain small. This would obviously be more difficult to achieve in three dimensions. We include *GSVLR* because it is standard in many codes of this kind; the updating of coefficients is vectorizable with vector lengths equal to the relatively small number of vertical mesh lines; the tridiagonal solution is not vectorizable. The *JVLR* method is a variant of *GSVLR* for which coefficient updating is vectorizable with vector lengths equal to the number of points in the entire grid. The *PJ* method is eminently vectorizable and exceedingly cheap, and lies at the opposite end of the relaxation method spectrum from *FI*.

Operationally, the user may specify the relaxation type to be used for each grid; they need not be all the same. The modularity of code and the availability of precalculated coefficients and residuals make additional relaxation types easy to implement.

V. Numerical Experiments

The code was of course designed with extensive numerical testing in mind. We report here only some preliminary experiments done for the purpose of validating the programming and obtaining some initial experience with embedded mesh. We compare the results with experiment and the performance with that of *CONRAX*, a conservative version of *RAXBOD* [9] developed by Green. It calculates inviscid potential flow on a single stretched grid by *SLOR*.

The experiments for our program, which we call *MG4*, were performed at Purdue University on a system composed of *CDC* 6500 and 6600 computers. For the purposes of reporting timing information, we report *CPU* seconds for the 6500, using Purdue's rough rule of thumb that 1 second for the 6600 equals 2 seconds for the 6500 to convert where necessary. As a rough measure of convergence we refer to r_{avg} and r_{max}, the average and maximum residuals on the finest grid on the afterbody. These represent the defect in (1), and are in divided form, so that they represent a rate of mass production per unit area per unit time.

We consider two cases: one in which the free-stream Mach number $M_\infty = 0.80$, and one with $M_\infty = 0.96$. The former is a subcritical case presenting no unusual difficulties; the latter is supercritical with shocks on the cylindrical body and the afterbody and is difficult to resolve.

Timing

Table 1 shows *CPU* seconds and residuals achieved for $M_\infty = 0.80$ for a variety of solution procedures. Except for the last case, each *MG4* run uses 5 levels of grids, all body fitted. The sizes of these are listed in table 2. The 6th level is present only in the last table 1 entry.

Procedure	CPU sec.	r_{avg}	r_{max}	Figure
FI, 1 cycle	47.6	3.0×10^{-4}	1.2×10^{-3}	7
FI, 5 cycles	143.6	2.9×10^{-6}	4.1×10^{-5}	7
GSVLR, 1 cycle	30.3	4.5×10^{-4}	4.4×10^{-3}	7
GSVLR, 5 cycles	86.8	3.9×10^{-4}	4.4×10^{-3}	7
JCVLR, 1 cycle	19.8	6.0×10^{-3}	2.9×10^{-2}	7
JCVLR, 5 cycles	55.2	2.4×10^{-4}	8.6×10^{-4}	7
PJ, 1 cycle	18.6	8.4×10^{-3}	4.0×10^{-2}	7
PJ, 5 cycles	51.7	9.2×10^{-4}	3.0×10^{-3}	7
FI, 6 levels,1 cycle	81.9	5.1×10^{-4}	2.3×10^{-3}	9

Table 1
Times and Residuals for $M_\infty = 0.80$

The test code restricts each grid to 231 points with a maximum of 11 mesh widths vertically, with the total number of points not to exceed 1024. The total number of points for the *MG4* results in table 1 is 649, except for the last entry where it is 874. The multigrid cycle is fixed at 5 relaxation sweeps on each grid before coarsening, 2 sweeps after refinement. The number of cycles (in *FMG*) before reaching the finest level is always 1; the number after reaching the finest level is given in the table. The program was compiled using the *FTN4* compiler at optimization level 2; the compilation requires about 40 seconds on the 6500. Times in the table are execution times only.

Comparison with Experiment

Figure 7 presents the coefficient of pressure (*CP*) on the afterbody computed with *MG4* as compared with experimental data. Characteristically, one expects a drop in the *CP* during the expansion phase at the beginning of the afterbody, followed by a precipitous rise as the flow compresses approaching the juncture with the exhaust plume (indicated by a tick mark on the axis in the figure. Because of the discontinuity in the derivative at the juncture, the inviscid flow experiences an infinite pressure spline at the juncture, which is mollified in the experimental data by the presence of the viscous boundary layer which in effect smooths the discontinuity. Thus the computed pressure rise naturally

Level	Mesh ($\xi x \eta$)	Step Size	Boundaries Left	Right	Outer
1	11x5	4.0	-16.0	24.0	16.0
2	13x5	2.0	-8.0	16.0	8.0
3	21x5	1.0	-4.0	16.0	4.0
4	13x5	0.5	-2.0	4.0	2.0
	13x5	0.5	6.0	12.0	2.0
5	21x5	0.25	-1.0	4.0	1.0
	21x9	0.25	6.0	11.0	2.0
6*	25x9	0.125	8.0	11.0	1.0

* Present in last case of table 1 only.

Table 2
Mesh Sizes

exceeds, and to some extent must precede, that of the experimental data.

We present only one such graph because of the rather remarkable result that *the computed CP for all 8 cases differs by only a few percent, so that the graphs are indistinguishable.* We elaborate on this situation in the next section.

Figure 8 shows a similar comparison with pressures computed by *CONRAX* on a $77x39$ grid. It may be observed that the *CONRAX* data follows the experimental data farther along the afterbody before feeling the effect of the pressure spike. However, the density of points on the boattail for *CONRAX* is about twice that at level 5 of *MG*4. We thus augmented the *MG*4 mesh by adding an additional level, with results as shown on Figure 9. (Note the expanded horizontal scale.) The effect is in fact to draw the computed data to the right, and to increase the pressure spike because of better resolution of the discontinuity. An attempt to refine the mesh yet one more level placed a point extremely near the juncture with the result of an enormous pressure rise past which the finite difference scheme could not track the solution.

Clearly additional investigation of the effects of mesh refinement are required.

A few selected results for $M_\infty = 0.96$ are shown in table 3. This case is far more critical with respect to the dependence of the solution on the degree of convergence and placement of grids. We show no attempt to obtain *MG*4 results with 6 levels because, in order to avoid supersonic inflow, the fine grid would need to extend too far to the left and exceed dimension limits in the test code. An attempt was made with supersonic inflow; this procedure ultimately diverged. Because of previous experience, we limit *GSVLR* to 1 cycle. *CONRAX* results for this case are shown in Figure 14.

Procedure	*CPU* sec.	r_{avg}	r_{max}	Figure
GSVLR, 1 cycle	30.6	$1.6\,x\,10^{-3}$	$1.1\,x\,10^{-2}$	10
FI, 1 cycle	47.8	$1.7\,x\,10^{-3}$	$7.6\,x\,10^{-3}$	11
FI, 5 cycles	212.4	$4.6\,x\,10^{-4}$	$5.7\,x\,10^{-3}$	12
FI, 10 cycles*	542.0	$3.0\,x\,10^{-5}$	$2.4\,x\,10^{-4}$	13

* Number of smoothing iterations after refinement was increased from 2 to 5.

Table 3
Times and Residuals for $M_\infty = 0.96$

VI. Concluding Remarks

First, it is clear that the procedure can be made to work, and that the cost is low compared to other standard methods. However, it is also obvious that it is quite sensitive to certain factors; it is the identification and explanation of these dependencies which is of fundamental importance.

The most striking feature here, as with all such schemes, is the deterioration of its effectiveness with increasing Mach numbers. Even the fully implicit procedure, which commonly exhibits a smoothing rate of 0.4 to 0.6 in the case $M_\infty = 0.80$, deteriorates to a rate of 0.85 to 0.95 when $M_\infty = 0.96$. The important observation here is that, since *FI* eliminates completely the inner iteration by solving the linearized equations exactly, it is the outer iteration, i.e., variation in the coefficients, which impedes the smoothing process. Thus, success will necessarily depend on either a clever scheme for updating the coefficients or on a different linearization, such as a Newton-like method, designed to take the dependence of ρ on φ into closer account. The extremely poor performance of *GSVLR* seems to indicate that the simple approach of updating the coefficients more frequently is not the solution.

It is also obvious that it is necessary to clearly define what is meant by a solution. For $M_\infty = 0.80$, the prediction of the *CP* apparently is very little effected after convergence of a certain minimal amount has been achieved. This indicates that variation in computed results is due to variation in the computational model, including the degree of refinement and the placement of grids as well as the finite difference equations. Such effects account for most of the difference between *MG4* and *CONRAX* results. Unfortunately, it is impossible to compare computed results with a "true inviscid solution". Comparison with experiment should be more reasonable after coupling to a separate boundary layer program. However, the only hope of monitoring convergence internally to the computation is by estimating truncation error as suggested by Brandt [1]. We have made some preliminary attempts to do so, with the result that the estimates are quite large. This may be correct, but needs further study. It is suspect largely because of the uncertainty, especially for $M_\infty = 0.96$, whether the error has been sufficiently smoothed to permit such estimates.

Some rather more specific observations may be made from the computed results. Referring to table 1, we find that good results can be achieved with relatively cheap calculations. Some of these, such as *PJ*, are highly vectorizable as well. Note that one can afford 5 cycles of *PJ* for the cost of 1 cycle of *FI*, with residuals larger by only a factor of 3. For vector calculations, *PJ* might well be superior. It should be recalled, too, that since the grids considered here are not large in the vertical direction, the size of the tridiagonal systems encountered in the *VLR* methods, and the bandwidth of the systems in *FI*, is not large. In three-dimensional applications the solution of these systems will occupy a larger proportion of the computing time.

The poor performance of some of the schemes, such as *GSVLR*, may be due to a failure to take sufficiently many relaxation sweeps after grid refinement. Apparently the (linear) interpolation used introduces some high frequency error which is eliminated only by several sweeps. It is essential to note that we have not attempted to optimize the performance of any of the solvers.

A definite feature of the computed data is the horizontal smearing of features, which may well be an artifact of excessive reliance on coarse grids. This is especially evident in the $M_\infty = 0.96$ case, where the pressure rise is anticipated well in advance, resulting in an underprediction of the minimum pressure. This phenomenon is indicative of excess artificial viscosity, induced here in the form of upstream shifting of density evaluations. Since these shifts are proportional to the mesh size, they can be quite large on the coarsest grids. It is apparent that some balance must be struck which will limit the artificial viscosity and yet maintain stability. We have attempted to relate all shifts to the mesh size of the finest grid, but calculations on coarse grids then become unstable. The matter requires more attention, as does the question of shifting for supersonic inflow on fine grids.

We have little experience so far with the unfitted grids. We have done calculations with 3 levels of fitted grids and 3 levels of Cartesian grids, with good results. There seems to be no difficulty in achieving the appropriate mass balances near the body, and, in fact, in some cases the computed *CP*'s appear closer to experimental data than those shown here. Linear interpolation between the finest Cartesian and coarsest sheared grids, however, did not appear to be effective, nor did cubic interpolation exhibit particularly superior characteristics. We anticipate further work in this area.

VII. Caveat

It must be emphasized that the program *MG4* which we describe here remains in a state of development. The results which we show were obtained partially for the purpose of testing the method, but mainly for the purpose of testing the code. This code has not been extensively exercised. In view of its complexity, it is quite possible that coding errors remain to be uncovered. We caution against placing premature reliance on negative results. Considering the embryonic nature of the software, we must report that, at this stage, the method and program both work remarkably well.

BIBLIOGRAPHY

[1] **Brandt, A.**, "Multi-Level Solutions to Boundary Value Problems," Math. Comp. 31 (1977), pp. 333-390.

[2] **Brandt, A.**, "Multi-Level Adaptive Techniques (MLAT) for Partial Differential Equations: Ideas and Software," in Mathematical Software III, Academic Press, 1977.

[3] **Brown, J.J.**, "A Multigrid Mesh Embedding Technique for Three-Dimensional Transonic Potential Flow Analysis," in Multigrid Methods, proceedings of the NASA-Ames conference of October, 1981, NASA Conference Publication CP-2202.

[4] **Green, L.L.**, "Conservative Full-Potential Calculations for Axisymmetric, Transonic Flow," AIAA paper 81-1204, June, 1981.

[5] **Hafez, M.M.**, **Murman, E.M.**, and **South, J.C.**, "Artificial Compressibity Methods for Numerical Solution of Transonic Full Potential Equation," AIAA paper 78-1148, July, 1978.

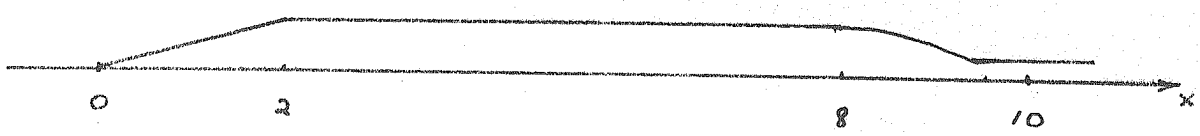
[6] **McCarthy, D.R.**, and **Reyhner, T.A.**, "Multigrid Code for Transonic Potential Flow about Inlets," AIAA Journal 20 (1982), pp. 45-50.

[7] **Reubush, D.E.**, "Experimental Study of the Effectiveness of Cylindrical Plume Simulators for Predicting Jet-on Boattail Drag at Mach Numers up to 1.30," NASA Technical Note TN D-7795, November, 1974.

[8] **Reyhner, R.A.**, "Transonic Potential Flow Computation about Three-Dimensional Inlets, Ducts, and Bodies," AIAA Journal 19 (1981), pp. 1112-1121.

[9] **South, J.C.**, and **Jamson, A.**, "Relaxation Solution for Inviscid Axisymmetric Transonic Flow over Blunt or Pointed Bodies," proceedings AIAA Computational Fluid Dynamics Conference, July 1973, pp. 81-17.

FIGURE 1



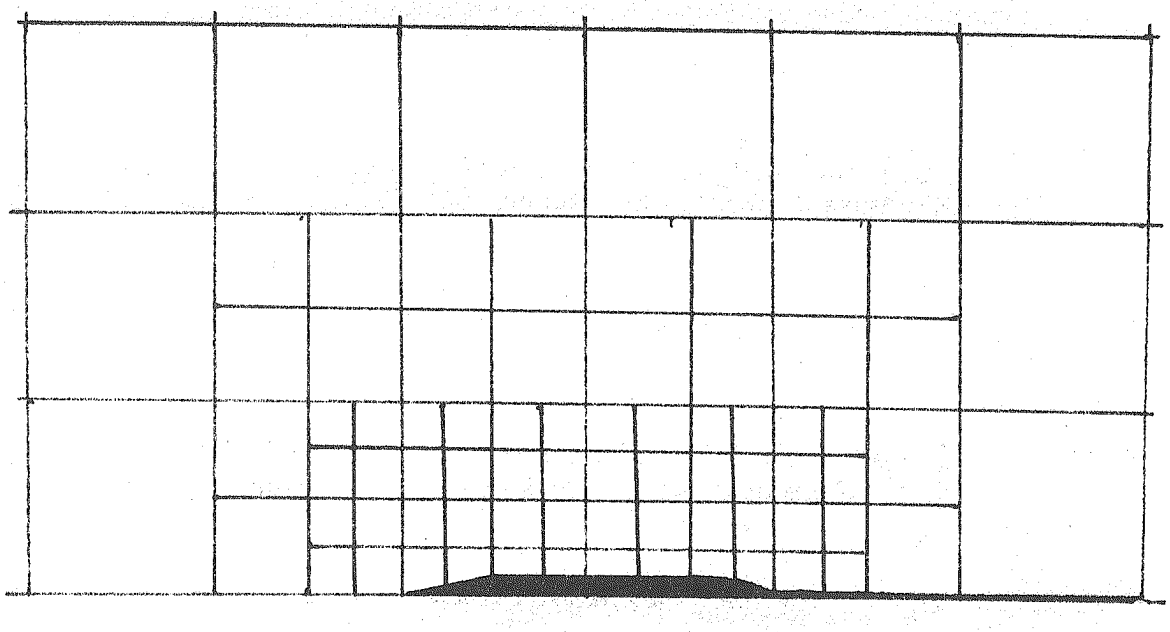


FIGURE 2

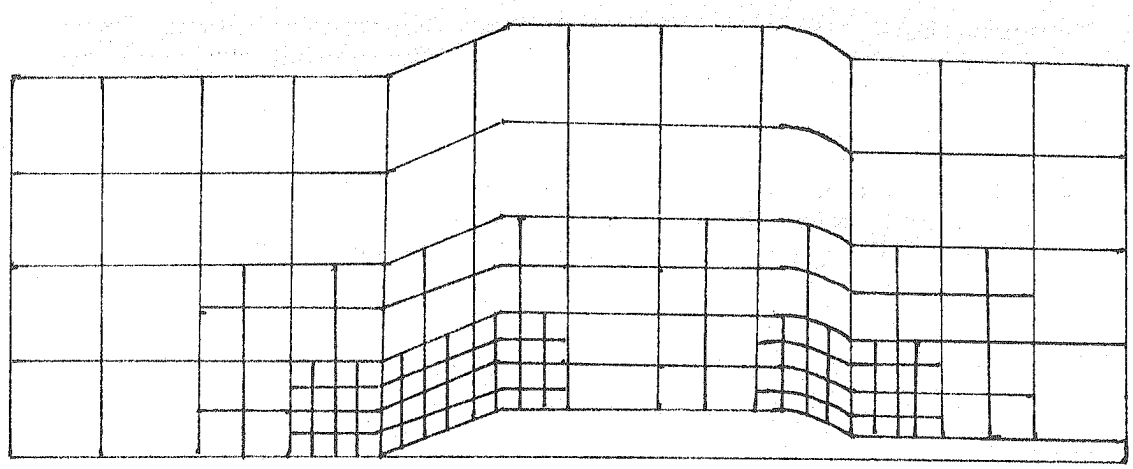


FIGURE 3

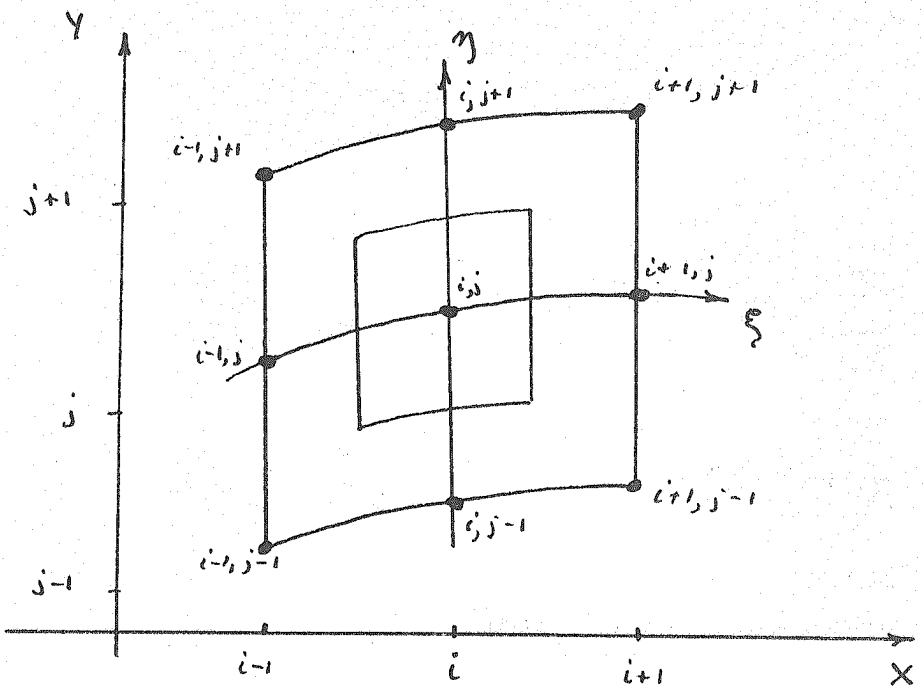


FIGURE 4

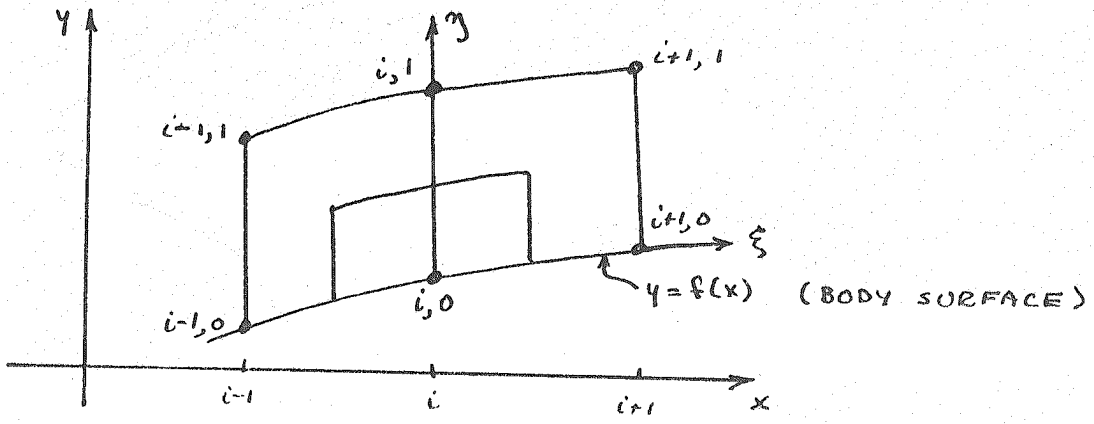


FIGURE 5

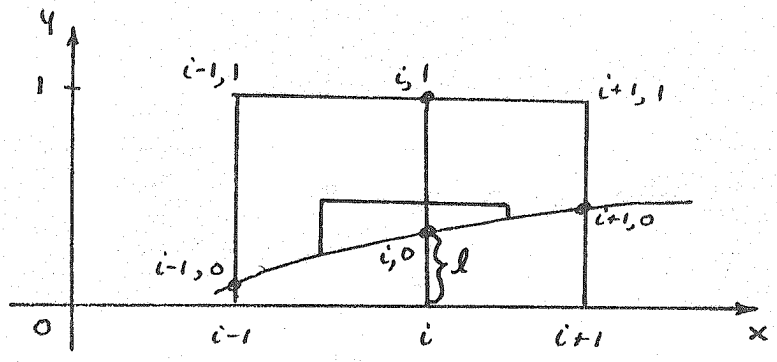


FIGURE 6

FIGURE 7

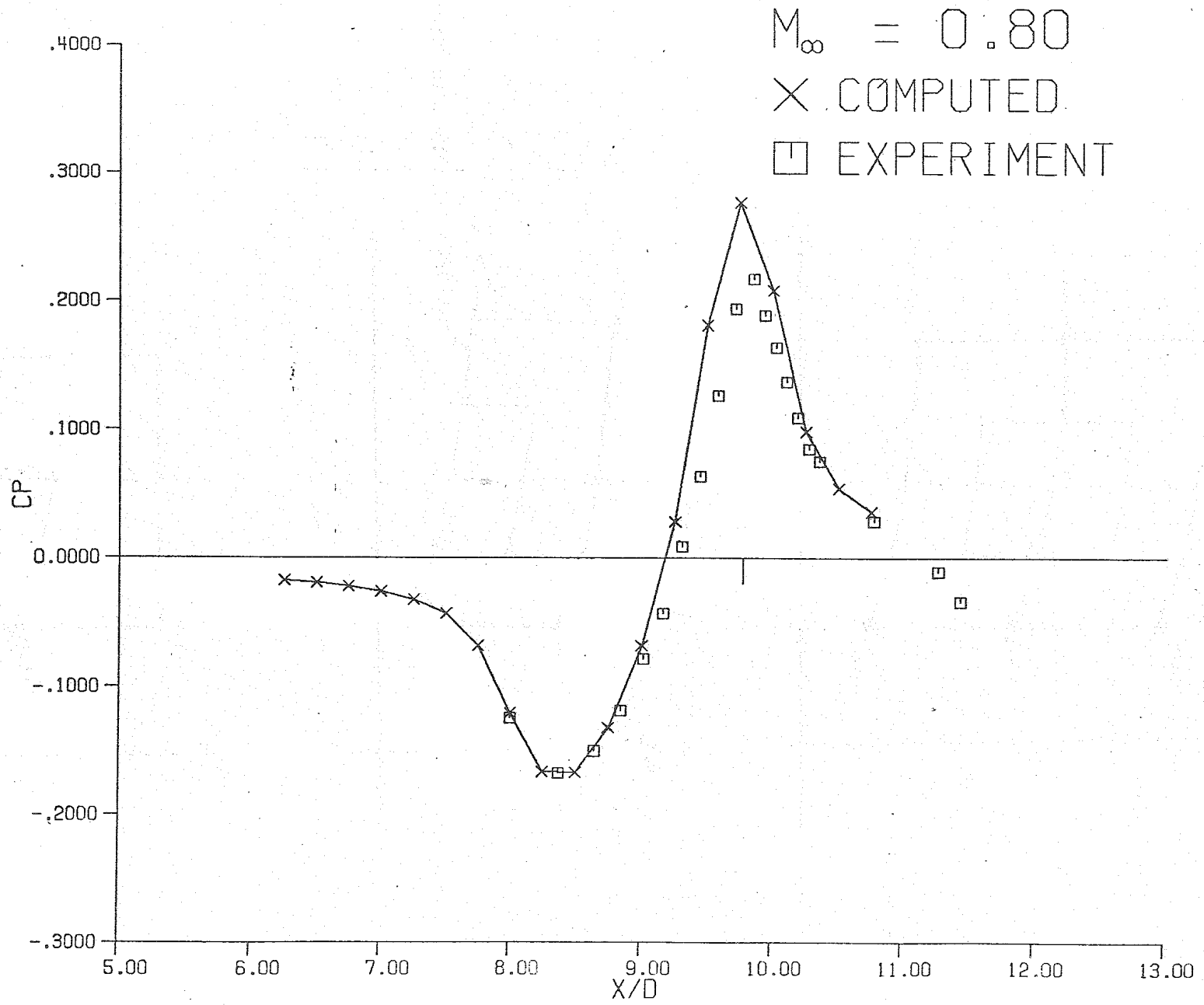

$M_\infty = 0.80$
X COMPUTED
EXPERIMENT
.4000
.3000
.2000
.1000
0.0000
-.1000
-.2000
-.3000
CP
5.00
6.00
7.00
8.00
9.00
10.00
11.00
12.00
13.00
X/D

FIGURE 8

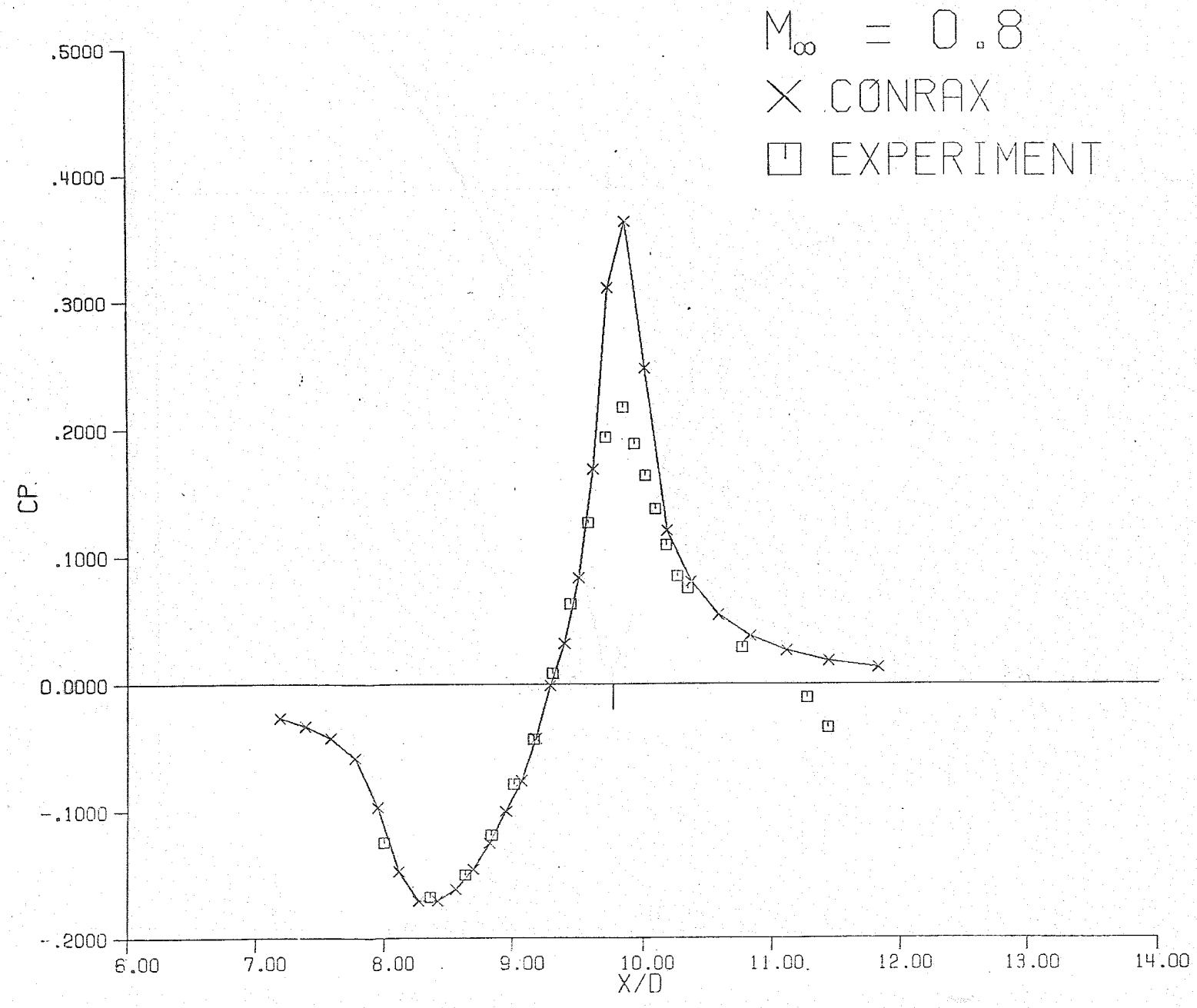

$M_\infty = 0.8$
CONRAX
EXPERIMENT
.5000
.4000
.3000
.2000
.1000
0.0000
-.1000
-.2000
CP
6.00
7.00
8.00
9.00
10.00
11.00
12.00
13.00
14.00
X/D

FIGURE 9

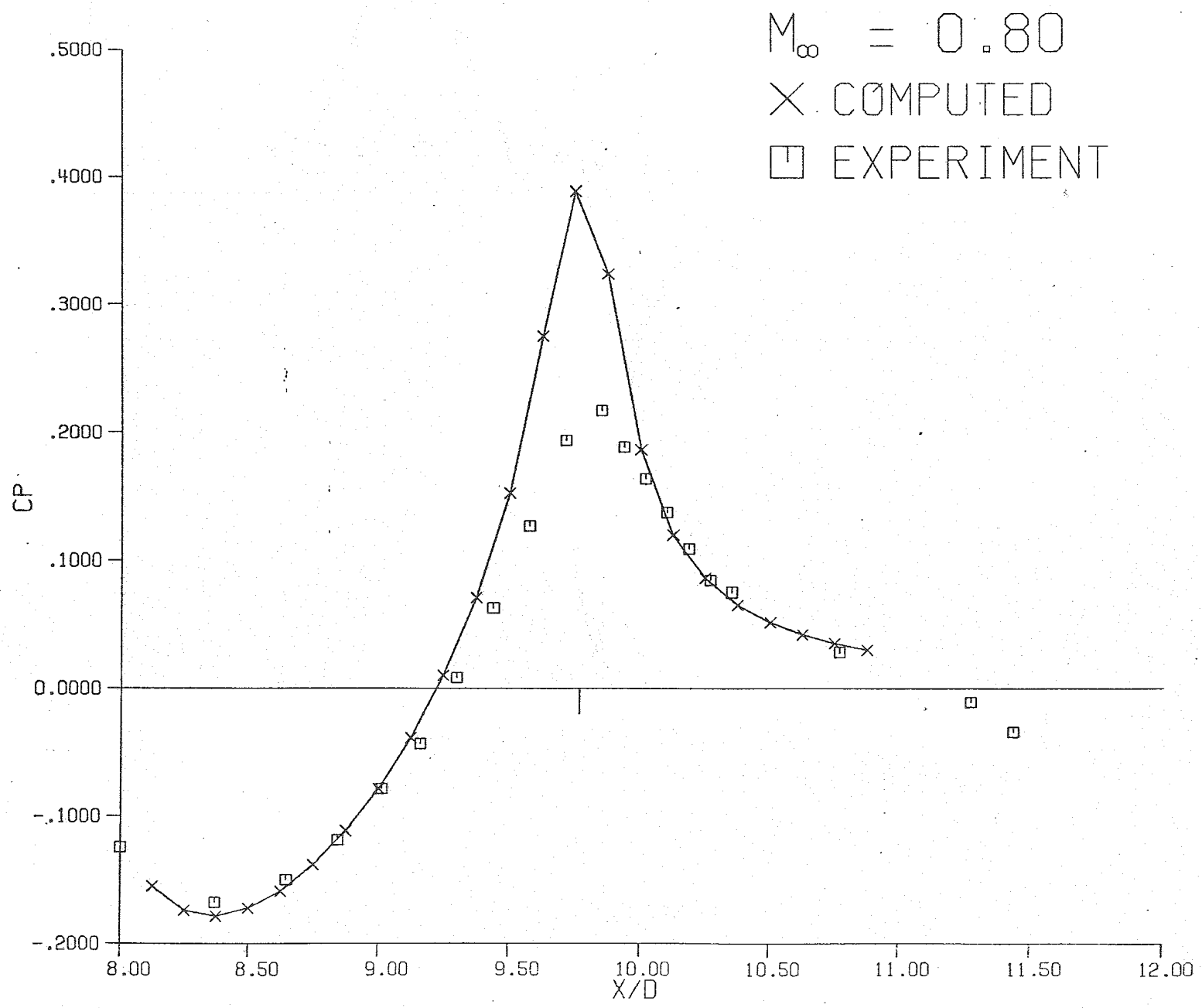

$M_\infty = 0.80$
X COMPUTED
EXPERIMENT
.5000
.4000
.3000
.2000
.1000
0.0000
-.1000
-.2000
CP
8.00
8.50
9.00
9.50
10.00
10.50
11.00
11.50
12.00
X/D

FIGURE 10

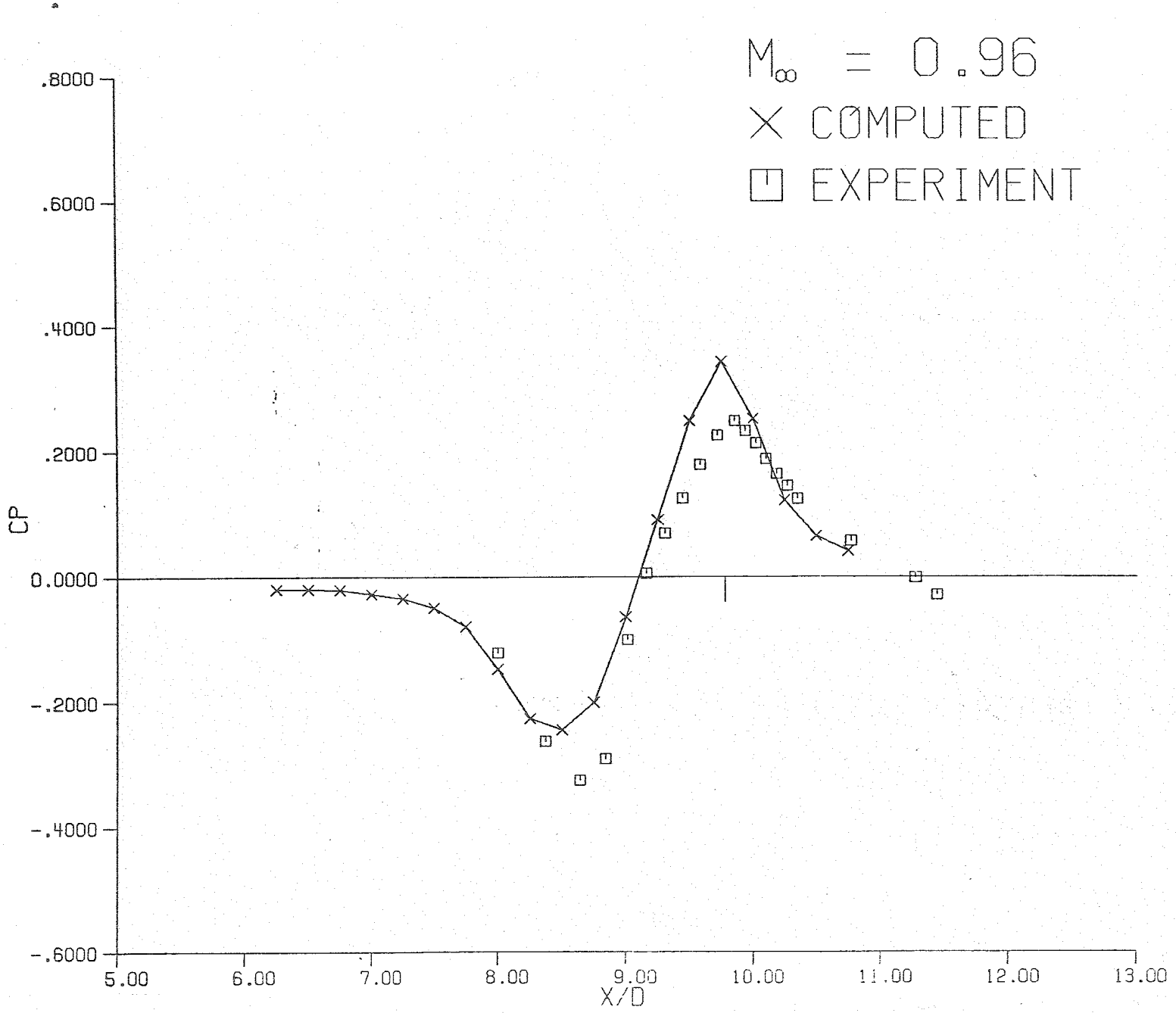

M_∞ = 0.96
X COMPUTED
EXPERIMENT
.8000
.6000
.4000
.2000
0.0000
-.2000
-.4000
-.6000
CP
5.00
6.00
7.00
8.00
9.00
10.00
11.00
12.00
13.00
X/D

FIGURE 11

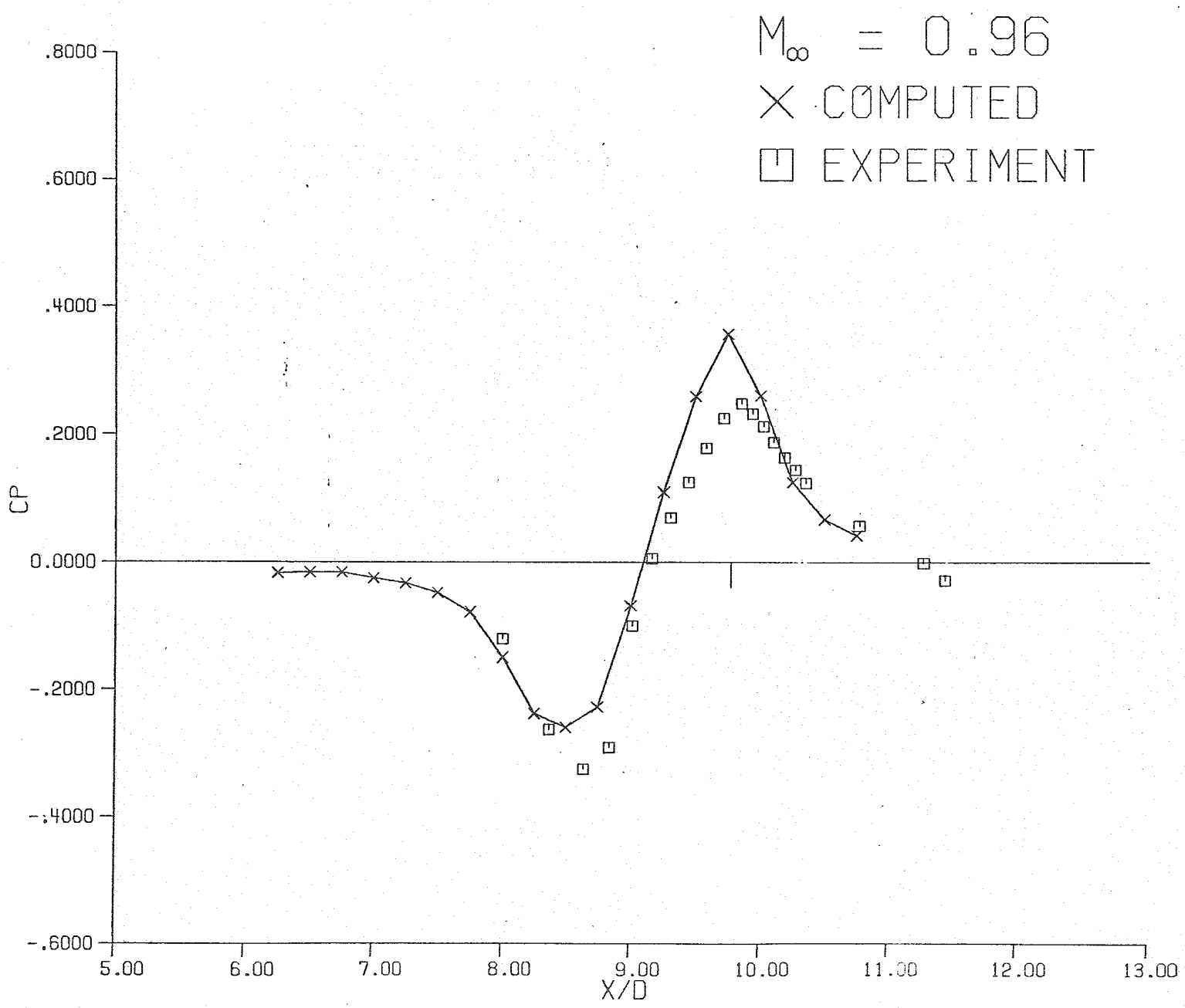

M_∞ = 0.96
X COMPUTED
EXPERIMENT
.8000
.6000
.4000
.2000
0.0000
-.2000
-.4000
-.6000
CP
5.00
6.00
7.00
8.00
9.00
10.00
11.00
12.00
13.00
X/D

FIGURE 12

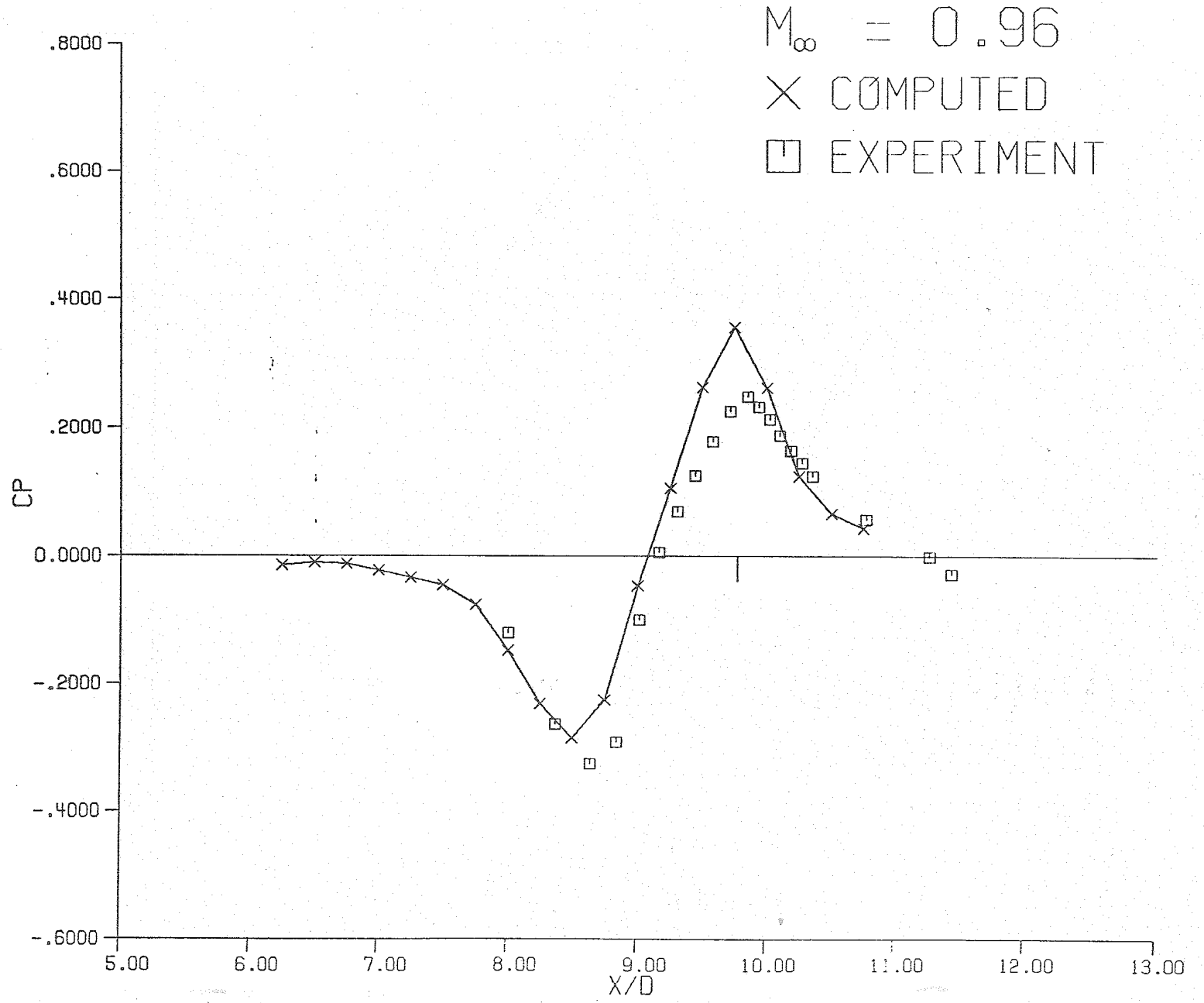

M_∞ = 0.96
X COMPUTED
EXPERIMENT
.8000
.6000
.4000
.2000
0.0000
-.2000
-.4000
-.6000
CP
5.00
6.00
7.00
8.00
9.00
10.00
11.00
12.00
13.00
X/D

FIGURE 13

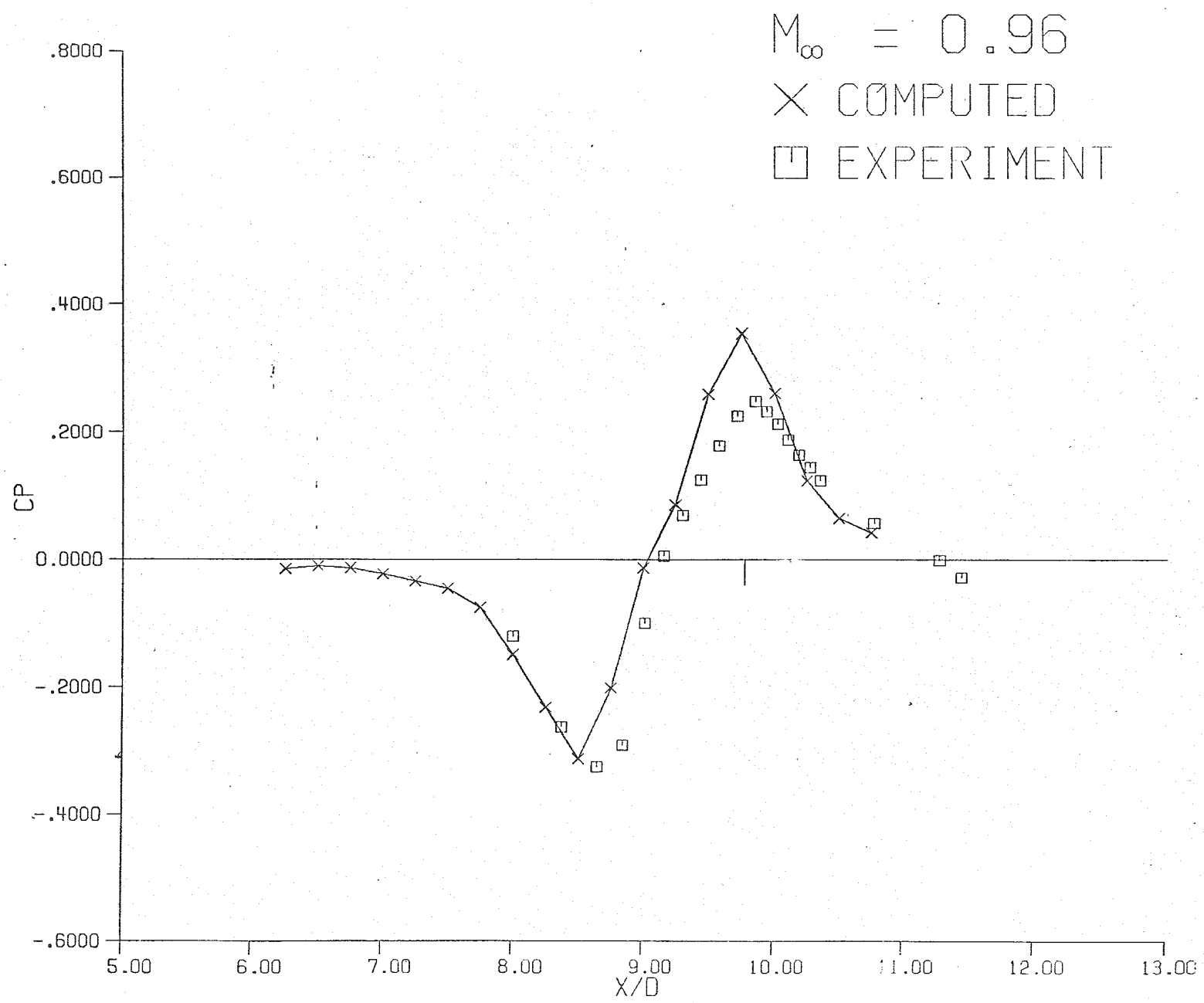
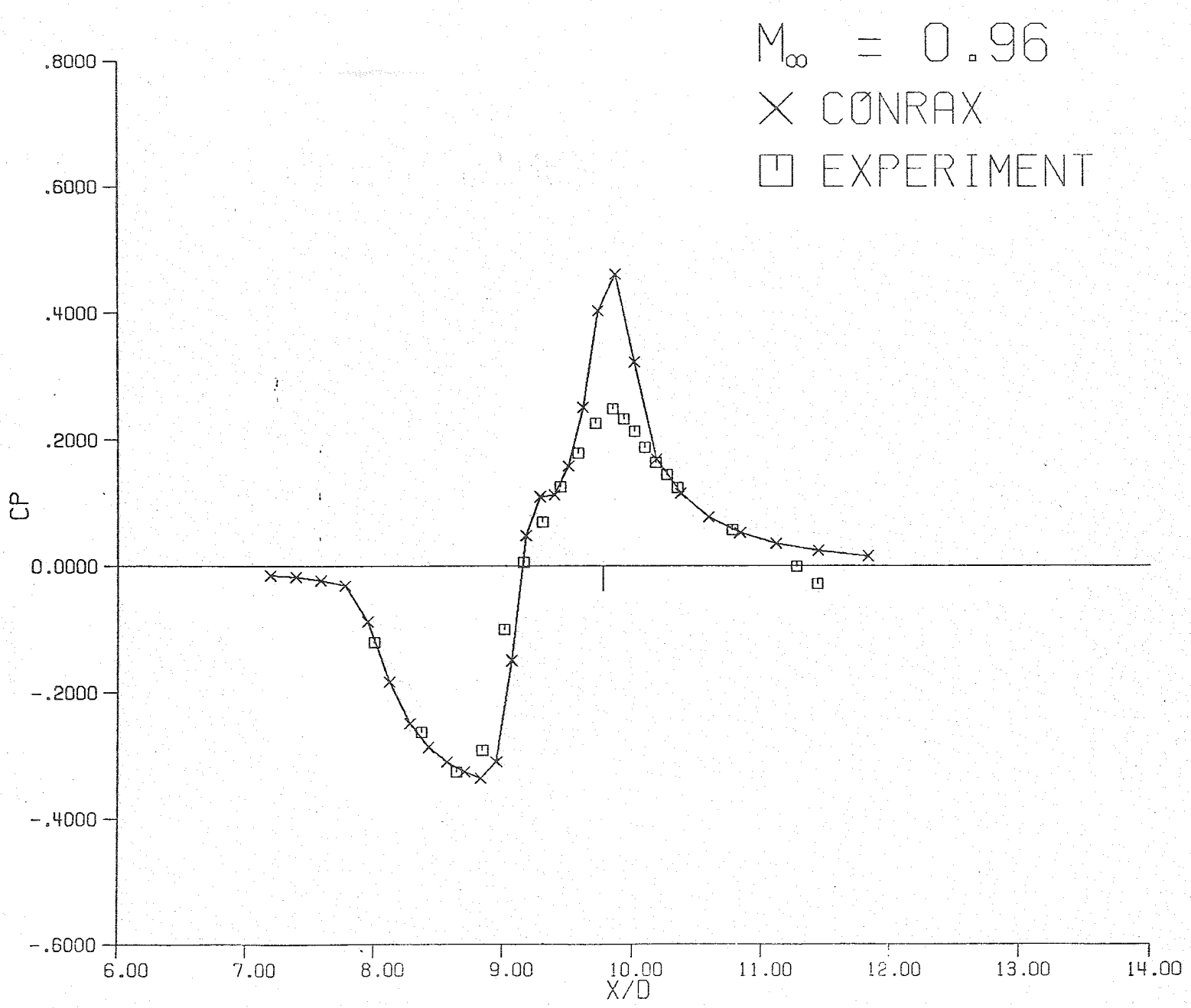

$M_\infty = 0.96$
X COMPUTED
EXPERIMENT
.8000
.6000
.4000
.2000
0.0000
-.2000
-.4000
-.6000
CP
5.00
6.00
7.00
8.00
9.00
10.00
11.00
12.00
13.00
X/D

FIGURE 14


M_∞ = 0.96
CONRAX
EXPERIMENT
.8000
.6000
.4000
.2000
0.0000
-.2000
-.4000
-.6000
CP
6.00
7.00
8.00
9.00
10.00
11.00
12.00
13.00
14.00
X/D