AIAA'86

AIAA-86-0554

POLYNOMIAL ELIMINATION THEORY AND NON-LINEAR STABILITY ANALYSIS FOR THE EULER EQUATIONS

S. R. Kennon, University of Texas at Austin,
D. C. Jespersen, NASA Ames Research Center, and
G. S. Dulikravich, University of Texas at Austin.

AIAA 24th Aerospace Sciences Meeting
January 6-9, 1986/Reno, Nevada

POLYNOMIAL ELIMINATION THEORY AND NON-LINEAR STABILITY ANALYSIS FOR THE EULER EQUATIONS

S. R. Kennon
Graduate Research Asst.
Texas Institute for Computational Mechanics (TICOM)
University of Texas at Austin

D. C. Jespersen
Research Scientist
Computational Fluid Dynamics Branch
NASA Ames Research Center
Moffett Field, CA

G. S. Dulikravich
Asst. Professor
Texas Institute for Computational Mechanics (TICOM)
University of Texas at Austin

ABSTRACT

Numerical methods are presented that exploit the polynomial properties of discretizations of the Euler equations. It is noted that most finite difference or finite volume discretizations of the steady-state Euler equations produce a polynomial system of equations to be solved. These equations are solved using classical polynomial elimination theory, with some innovative modifications. This paper also presents some preliminary results of a new non-linear stability analysis technique. This technique is applicable to determining the stability of polynomial iterative schemes. Results are presented for applying the elimination technique to a one-dimensional test case. For this test case, the exact solution is computed in three iterations. The non-linear stability analysis is applied to determine the optimal time step for solving Burgers' equation using the MacCormack scheme. The estimated optimal time step is very close to the time step that arises from a linear stability anaysis.

INTRODUCTION

In ref. 1 a polynomial-based technique was used to accelerate the convergence of a finite-volume Euler solver. In this paper, we extend these results by describing additional techniques for exploiting the polynomial properties of the Euler equations. In particular, we use classical polynomial elimination theory to solve the discretized Euler equations and introduce a new non-linear stability analysis technique.

Polynomial elimination theory was developed along with modern abstract algebra in the late nineteenth century[2]. This elimination theory gives constructive methods for determining whether or not a system of polynomial equations (a system of equations consisting of sums of terms that are products of the variables raised to non-negative integer powers) has a solution. As a by-product of applying the theory to determine the existence of a solution, one gets the solution itself (if it exists).

In this paper, we present a practical method for applying the elimination theory to solve large systems of polynomial equations. In particular, we show that the algebraic equations resulting from standard discretizations of the equations of gasdynamics (Euler equations) and their associated boundary conditions form a system of polynomial equations. With suitable assumptions, it can also be shown that discretizations of the Navier-Stokes equations form polynomial systems of equations. Thus the elimination method has potential applicability to solving both inviscid and viscous steady compressible flows. It must be noted that if one is trying to solve a system of algebraic equations that is only partially polynomial, the non-polynomial terms could be approximated by polynomials using truncated Taylor series. Thus this solution method may be applied to many equations in the engineering sciences.

The basic idea of polynomial elimination theory is to sequentially eliminate dependent variables from the system, in close analogy with Gaussian elimination. This elimination procedure gives a modified system of equations that, if no approximations have been made, has exactly the same roots as the original system. Moreover, the modified system can be solved in the same manner as the 'back solve' is done in Gaussian elimination for linear systems. That is, the modified system is in a form such that the system can be solved by solving a sequence of one variable equations.

In general, the procedure becomes impractical for large systems; thus, we introduce some innovative modifications to the basic technique to make the method practical. The main modification is the use of approximation formulas to both reduce the amount of work needed to solve the polynomial system and increase the accuracy of the results. The use of the approximation formulas makes the results of the elimination process inaccurate. Therefore, the system of polynomial equations is recast into a form that involves corrections to the variables. The corrections to the variables then become the primary unknowns. This defines a global iterative procedure in which an initial guess for the solution is given and then each iteration consists of solving for the corrections using polynomial elimination and adding the corrections to the present iterate.

This paper also presents a non-linear stability analysis technique that can be applied to determine the stability of polynomial iteration schemes. The technique gives a bound on the error of the solution at the new iteration level. This bound depends on the error at the old iteration level and the norm of a polynomial function. The error at the new iteration level can be minimized by minimizing the norm of this function with respect to a free parameter such as the time step. In addition, for a given time step, the stability of the iteration scheme can be checked by computing this norm.

The following two sections of this paper present the mechanics of the elimination procedure and the

details of the non-linear stability analysis. These sections are followed by some results of applying these techniques to two test cases.

THE ELIMINATION ALGORITHM

This section presents: a) the theory of resultants, b) the application of this theory to the solution of a system of polynomial equations, and c) the approximation scheme used to make the solution procedure practical.

Resultants Assume that we are given two polynomials $p_1(x_1,x_2,...x_n)$ and $p_2(x_1,x_2,...x_n)$ in the n variables $x_1,x_2,...x_n$. Assume that we wish to eliminate variable x_1 from these equations and obtain a new polynomial that is both independent of x_1 and has the same roots as the original two polynomials. The so-called resultant polynomial of p_1 and p_2 with respect to x_1 has this property (cf. van der Waerden[2]). To form the resultant of two polynomials with respect to a given variable, say x_1, we first rewrite the two polynomials p_1 and p_2 as polynomials in x_1 whose coefficients depend on the other variables:

$$p_1(x_1, x_2,..., x_n) = a_i\ (x_2, x_3,..., x_n)\ x_1^{\,i}$$
$$\text{for } i = 0, 1,..., d_1$$

$$p_2(x_1, x_2,..., x_n) = b_i\ (x_2, x_3,..., x_n)\ x_1^{\,i}$$
$$\text{for } i = 0, 1,..., d_2$$

where we use the Einstein summation convention, and d_1 and d_2 are the highest powers of x_1 in p_1 and p_2 respectively. Next, we form the (d_1+d_2) by (d_1+d_2) Sylvester matrix:

$$\begin{vmatrix} a_{d1} & a_{d1-1} & \cdots & a_1 & a_0 & & \\ & a_{d1} & a_{d1-1} & \cdots & a_1 & a_0 & \\ & & \cdots\cdots & [\,d_2 \text{ rows of } a\text{'s}] & \cdots & & \\ & & & a_{d1} & a_{d1-1} & \cdots & a_1 & a_0 \\ b_{d2} & b_{d2-1} & \cdots & b_1 & b_0 & & & \\ & b_{d2} & b_{d2-1} & \cdots & b_1 & b_0 & & \\ & & \cdots\cdots & [\,d_1 \text{ rows of } b\text{'s}] & \cdots & & \\ & & & b_{d2} & b_{d2-1} & \cdots & b_1 & b_0 \end{vmatrix}$$

where all blank spaces are filled with zeros. The resultant of p_1 and p_2 with respect to x_1 is just the determinant of the Sylvester matrix and is therefore a polynomial in $x_2,x_3,...x_n$. Moreover, the resultant has the same roots as the original system, in terms of the variables $x_2,x_3,...x_n$ (see ref. 2 for more details and a proof of these claims).

Solving a System of Polynomial Equations Using the elimination properties of the resultant, we can construct a procedure for solving a system of polynomial equations that closely parallels Gaussian elimination for linear systems. In fact, if the polynomial system is indeed linear, then this procedure reduces to Gaussian elimination. The idea is to sequentially eliminate the variables from the equations using the properties of the resultant, and then perform a back solve that solves for all the variables. The procedure in algorithmic form is as follows:

(forward elimination)

For i = 1 thru n do

[Choose each variable $x_1,x_2...x_n$ in turn to be eliminated]

begin;

1. Find all equations that contain x_i

2. Choose one of these equations (say the first) as the 'eliminator'

3. Form the resultant of the eliminator equation with all the others that contain x_i

4. Set each original equation to be this resultant (now each equation except the eliminator does not depend on x_i)

end;

(the original system is now in a form amenable to back substitution)

for i = 1 thru n do

begin;

5. Find all equations that contain only one unknown variable and solve for all the roots of these equations

6. Make an intelligent guess as to which root to choose as the actual solution (e.g. check all real roots using some criteria).

7. If all variables $x_1,x_2...x_n$ have been solved for then stop.

end;

This algorithm will produce the exact solution (using exact arithmetic) in a finite number of steps. However, each time a variable is eliminated, the order of the modified equation (i.e. the resultant) increases. Consider the system

$$x_1 x_2 + 3x_2^2 - 4 = 0$$
$$x_1 (2x_2 - x_2^2) - 1 = 0 \qquad (1)$$

in x_1 and x_2. If we want to eliminate x_1 from the system, we must evaluate the determinant of a 2 x 2 Sylvester matrix (since the system is linear in x1). The

result is a fourth order polynomial in x_2

$$(3x_2^2 - 4)(2x_2 - x_2^2) + x_2 = 0 \qquad (2)$$

If we are to use this equation later on in the solution procedure to eliminate x_2 we would have to evaluate the determinant of a much larger matrix. In fact, the amount of work needed to successively eliminate each variable increases faster than exponentially. Therefore, for polynomial elimination theory to be practical, certain approximations have to be included.

Approximations to the Basic Method Two modifications were made to the basic elimination procedure to make it applicable for solution of large polynomial systems. The first modification is the introduction of an iterative procedure for solving the polynomial system. This iterative procedure is based on solving for a correction to an iterative guess at the exact solution instead of solving for the variables themselves. Therefore, we make the substitution

$$x_i \leftarrow x_i + \delta x_i \quad \text{for } i = 1, 2, \ldots, n \qquad (3)$$

and now treat $\delta x_1, \delta x_2, \ldots \delta x_n$ as the unknowns in the elimination process. Clearly, the original system now becomes a polynomial system of the same degree in the new unknowns, namely the δx_i. Once the corrections are solved for, the iterates are updated according to

$$x_i^{k+1} = x_i^k + \delta x_i^k \quad \text{for } i=1,2,\ldots n \qquad (4)$$

where k is the global iterative counter. The procedure is halted when some norm of the corrections δx_i is deemed to be within a specified small tolerance. The motivation for using this correction scheme is threefold. First of all, we will introduce some approximations, as discussed below, that will not make the solution exact. Secondly, correction schemes are, in general, better conditioned than direct schemes. A final justification for the use of a correction scheme is that one usually knows with a great degree of certainty the order of magnitude of the corrections, especially if the governing system has been non-dimensionalized. This knowledge will be applied in the following discussion on the approximation scheme.

As noted above, the amount of work needed to solve a general polynomial system using elimination theory grows faster than exponentially, or for that matter faster than factorially, as each variable is eliminated. Thus, we were led to the following modification of the elimination proceure. After step 4 of the above algorithm, we add step 4.1:

> 4.1 If the degree of the resultant is too high, approximate it with a lower order polynomial.

The approximation that we use in step 4.1 is to find the polynomial $p^*(x)$ of given degree d^* that is closest to the resultant polynomial $p(x)$ of degree d in the L_2-norm sense:

$$\min \int [p(x) - p^*(x)]^2 \, dx \qquad (5)$$

where $p(x)$ and $p^*(x)$ are given by

$$p(x) = a_i x^i \quad \text{for } i = 0, 1, 2, \ldots, d$$
$$p^*(x) = b_i x^i \quad \text{for } i = 0, 1, 2, \ldots, d^*$$

The solution to this linear least squares problem can be easily found, especially with the use of a symbolic manipulator such as Macsyma[3]. In the results to be presented below, we use a quadratic approximant. The quadratic approximation is quite accurate, especially when combined with the correction scheme. The limits of the integration in eq. 5 are chosen to encompass the entire expected range of possible values for the corrections δx_i. For example, if the original system of polynomial equations has been normalized (i.e. non-dimensionalized) such that the dependent variables are of O(1), then it would be reasonable to choose the integration limits as -c to +c where c is some number less than 1. Figures 1a-c show typical results of approximating the polynomials encountered in solving the one-dimensional Euler equations (see the results section below). These plots clearly show how accurate the approximation scheme is.

NON-LINEAR STABILITY ANALYSIS METHOD

Assume that we are trying to solve a system of equations

$$f(x)=y \qquad (6)$$

where

$$f: \Re^N \rightarrow \Re^N; \quad x,y \in \Re^N$$

$\Re^N$ is N-dimensional real Euclidean space, and y is a given constant vector. Further assume that the system is to be solved using an iterative procedure defined by the iteration function $g: \Re^N \rightarrow \Re^N$

$$x^{n+1} = g(x^n) \qquad (7)$$

where g is a polynomial function of x. Many iteration schemes applied to the Euler equations give iteration functions g which are polynomial. Let x^* be the exact solution of (6) such that

$$f(x^*) = y \qquad (8)$$

and

$$x^* = g(x^*) \qquad (9)$$

that is, x^* is a fixed point of g. Now define the error at iteration level n by

$$e^n \equiv x^* - x^n \qquad (10)$$

$$\Rightarrow e^{n+1} = x^* - x^{n+1} \qquad (11)$$

Solving (10) and (11) for x^* and substituting into (9) gives

$$x^{n+1} + e^{n+1} = g(x^n + e^n) \qquad (12)$$

Since g(x) is polynomial we have from a Taylor series expansion

$$g(a+b) = g(a) + q(a,b) \qquad (13)$$

for any $a,b \in \Re^N$. The function q(a,b) is polynomial in both a and b, of degree M in b and degree M−1 in a (here M is the total degree of g). This function can be thought of as just the cross-product terms arising from the expansion of the LHS of (13). Using the identity (13) we find from (12) that

$$x^{n+1} + e^{n+1} = g(x^n) + q(x^n, e^n) \qquad (14)$$

but from (7) we have that $x^{n+1} = g(x^n)$. Thus we get the basic result of the analysis

$$e^{n+1} = q(x^n, e^n) \qquad (15)$$

This result gives an explicit formula for the error at the iteration level n+1 as a function of the solution and the error at iteration level n. Now we define a norm for the operator q:

$$\| q_a \| \equiv \max_{b \in \Re^N} \| q(a,b) \| / W(\| b \|) \qquad (16)$$

where

$$W(\| b \|) = \| b \| + \| b \|^2 + \ldots + \| b \|^M$$

and $\|\cdot\|$ is the standard Euclidean (L_2) norm. The subscript a on q in (16) is a reminder that the norm is dependent on the fixed vector a. The philosophy behind this definition for the norm of q is that we want to emulate as closely as possible the norm of a linear operator. The norm of a linear operator has the effect of "picking-off" the coefficients of the linear function, namely the elements of the matrix of the linear function. Thus we wish to define a norm for a polynomial function that also "picks-off" the coefficients of each term comprising the polynomial function. One can see that the function defining the norm is clearly bounded since the numerator is divided by a function that always has the same order of magnitude as the numerator. Using this definition of the norm, we see that for any particular value of $b \in \Re^N$ we have that

$$\| q(a,b) \| \leq \| q_a \| \; W(\| b \|) \qquad (17)$$

Thus from (15) we get

$$\| e^{n+1} \| = \| q(x^n, e^n) \|$$

$$\leq \| q(x^n) \| \; W(\| e^n \|) \qquad (18)$$

Since W(·) is a non-negative monotone increasing function of its argument (with the additional property that W(0) = 0) we see that to minimize the norm of the error at the new iteration level n+1 we must minimize the norm of the operator q, namely $\| q(x^n) \|$. Since the iteration function g(x) is usually dependent on some scalar parameters (for instance the time step), we can attempt to choose these parameters to minimize $\| q(x^n) \|$ at each iteration. The calculation of the norm of q at each iteration is possibly as computationally expensive as solving the original problem. Thus we must use an approximation to $\| q(x^n) \|$ in order to make the minimization scheme practical. For the example problem shown below, we estimated the norm as the sum of the squares of the coefficients of all terms of q of degree one. The reason for this is that if these terms dominate q, then as $\| b \| \rightarrow 0$ the higher degree terms will approach zero faster than the linear terms. Thus only the linear terms are left in the numerator of the RHS of (16). Future research will address the problem of finding better estimates for $\| q(x^n) \|$.

RESULTS

To test the elimination procedure, we solved the steady state, constant enthalpy, one-dimensional Euler equations with forcing terms given by

$$(\rho u)_x - a(x) = 0$$

$$(a_1 \rho + a_2 \rho u^2)_x - b(x) = 0 \qquad (20)$$

on the interval x = [0,1] with prescribed Dirichlet boundary conditions at x = 0 and x = 1. Here a(x) and b(x) are prescribed functions, ρ is the density, u is the fluid velocity and a_1 and a_2 are constants that depend on the ratio of specific heats and on the total enthalpy of the fluid. The continuous system (20) was discretized using simple centered differences to form the system of difference equations to be solved:

$$\rho_{i+1} u_{i+1} - \rho_{i-1} u_{i-1} - a_i = 0$$

$$(a_1 \rho_{i+1} + a_2 \rho_{i+1} u_{i+1}^2)$$
$$- (a_1 \rho_{i-1} + a_2 \rho_{i-1} u_{i-1}^2) - b_i = 0$$

for i = 2,..., imax − 1

where ρ_1, ρ_{imax}, u_1, and u_{imax} are specified Dirichlet boundary conditions and a(x) and b(x) have been multiplied by 2Δx. One can clearly see that this system of finite difference equations is polynomial of degree three, with the maximal degree of any variable being two. This is also true for discretizations of the two-dimensional and three-dimensional steady state Euler equations.

We used the correction scheme and the modifications noted above to solve the system for three values of imax: 6, 8 and 10. The forcing terms were calculated such that the solution given by

$$\rho_i = 0.4 + 0.2 \tanh [\, 5 (-1 + 2 (i-1)/(imax-1)) \,]$$

$$u_i = 0.4 - 0.3 \tanh [\, 5 (-1 + 2 (i-1)/(imax-1)) \,]$$

satisfies the equations exactly. This solution exhibits shock-like behavior and can be seen in figs. 2a and 2b. Figure 2 also shows the initial guess for the solution, namely a linear variation between the given end-point values, and the error between the initial guess and the exact solution. For the cases imax = 6 and imax = 8, the exact solution was found (to within machine zero) in one iteration, and therefore these cases will not be presented. Figure 3 shows the convergence history for the velocity u for the case imax = 10. Figure 4 shows the convergence history for the density for the same case. Convergence of the solution was obtained after three iterations. After the first iteration, we note that 10 of the 16 unknowns have been found exactly; therefore, only six unknowns remain to be found. Once it is determined that a variable has been solved for exactly (i.e. the correction goes to zero) this variable is eliminated from the system. Thus, the second iteration is used to determine which variables have converged. The third iteration is then used to solve for the remaining six unknowns and the system is solved.

Several points concerning these results must be noted at this time: 1) After one iteration, some of the unknowns are solved for exactly and are completely eliminated from the system of equations; 2) The solution procedure involves only the manipulation of the coefficients and exponents of the polynomial system, and therefore is quite fast for small systems; 3) Some criteria must be used to choose a root in the back substitution process in the case of multiple real roots. For the test case, we used the criteria that the root should be the one that is closest to the exact solution. This criteria is only applicable in a test case where the exact solution is known. In a general problem, the root should be chosen so that either its absolute value is the smallest of the roots, or that the root produces the minimal value of the residual of the system at the new iteration level k+1; 4). This method requires large amounts of storage. The code that was used to calculate the presented results made extensive use of unformatted input/output. Nevertheless, the i/o time was less than 15% of the total execution time.

The non-linear stability analysis method was tested on the solution of Burgers' equation using the MacCormack scheme (see ref. 4, pages 154-164 for details). Burgers' equation

$$u_t + 1/2 \, (u - u^2)_x = \mu \, u_{xx} \qquad (21)$$

was solved for $\mu = 0.01$ on a 21 point grid with uniform spacing. At each iteration, the linear estimate of the norm $\| q(x^n) \|$ was minimized with respect to the time step Δt using a canned subroutine for minimization of a function of one unknown. The initial guess for the solution was a linear variation between the values at the endpoints of the grid on the interval x = [-1, 1]. First, the standard MacCormack scheme was used to find the converged exact solution. This solution was then used to generate the results of fig. 5. This figure shows a plot of the error of the standard MacCormack scheme and the modified scheme in which an optimal Δt was chosen at each time step. The time step used for the standard scheme is as given in ref. 4. The optimal time steps were fairly constant and very close to the standard time step. From fig. 5 we see that the standard time step is indeed a good one since the optimal time-stepping scheme did not improve the convergence by much.

We note that the elimination technique was only tested for a small one-dimensional problem. Larger one-dimensional problems required too much computer memory for the present version of our computer program (written in FORTRAN for a Cray-XMP computer). Therefore, it is not known if the elimination technique will be a practical alternative to existing Euler solvers. For larger problems, much use will have to be made of unformatted i/o which could severely restrict the method's efficiency. Another alternative that is being looked into is the use of other high-level languages such as Pascal or Lisp that allow dynamic memory allocation. The use of these laguages would make it possible to compute much larger problems than is now possible with FORTRAN. It is not known at this time either if the non-linear stability analysis technique can be made practical. In any event, the stability analysis is thought to be mathematically sound, and thus should find use in the theory of numerical methods.

SUMMARY

A new method was presented for the direct solution of polynomial systems of equations. The method is based on the classical theory of polynomial elimination, and the numerical implementations of the elimination scheme have been shown to produce the correct solutions in a very small number of iterations. The method is applicable to the steady-state Euler equations and to the steady-state Navier-Stokes equations (with certain assumptions).

In addition, a non-linear stability analysis method was introduced and applied to a simple test case. This stability analysis method is applicable to polynomial iteration functions and can be used to calculate optimal parameters of the iteration scheme.

ACKNOWLEDGEMENTS

Funds for the support of this study have been allocated in part by NASA-Ames Research Center, Moffett Field, California, under Interchange No. NCA2-16. The authors would like to thank Dr. Harvard Lomax for his support of this research. This research was also supported by a National Science Foundation Graduate Fellowship (supporting the lead author) and by the Air Force Office of Scientific Research, Grant No. 85-0052, under the supervision of Capt. John Thomas, Manager of Computational Mathematics.

References

1. Kennon, S. R., and G. S. Dulikravich, "Optimal Acceleration Factors for Iterative Solution of Linear and Non-Linear Differential Systems," Computer Methods in Applied Mechanics and Engineering, 47, 1984, pp. 357-367. (also AIAA paper No. 85-0162, Reno, January, 1985).

2. van der Waerden, B. L., Algebra, volumes 1 and 2, Frederick Ungar Publishing Co., New York, 1970.

3. MACSYMA Reference Manual. Mathlab Group, Laboratory for Computer Science, Massachusetts Institute of Technology, 1977.

4. Anderson, D. A., J. C. Tannehill, and R. H. Pletcher. Computational Fluid Mechanics and Heat Transfer. McGraw-Hill, 1984.

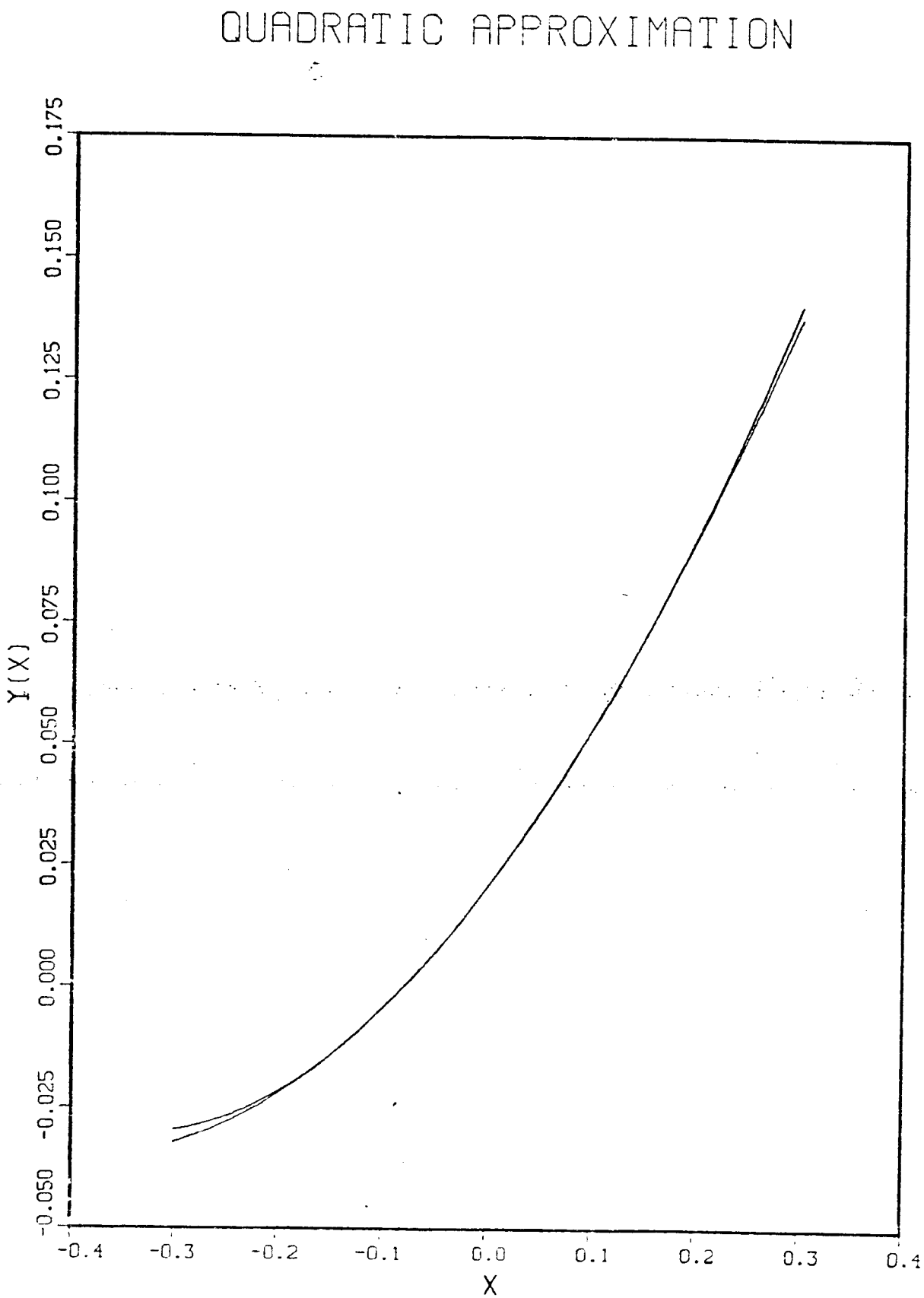


Fig. 1a Plot of approximation of high order polynomial with quadratic approximant approximation # 1

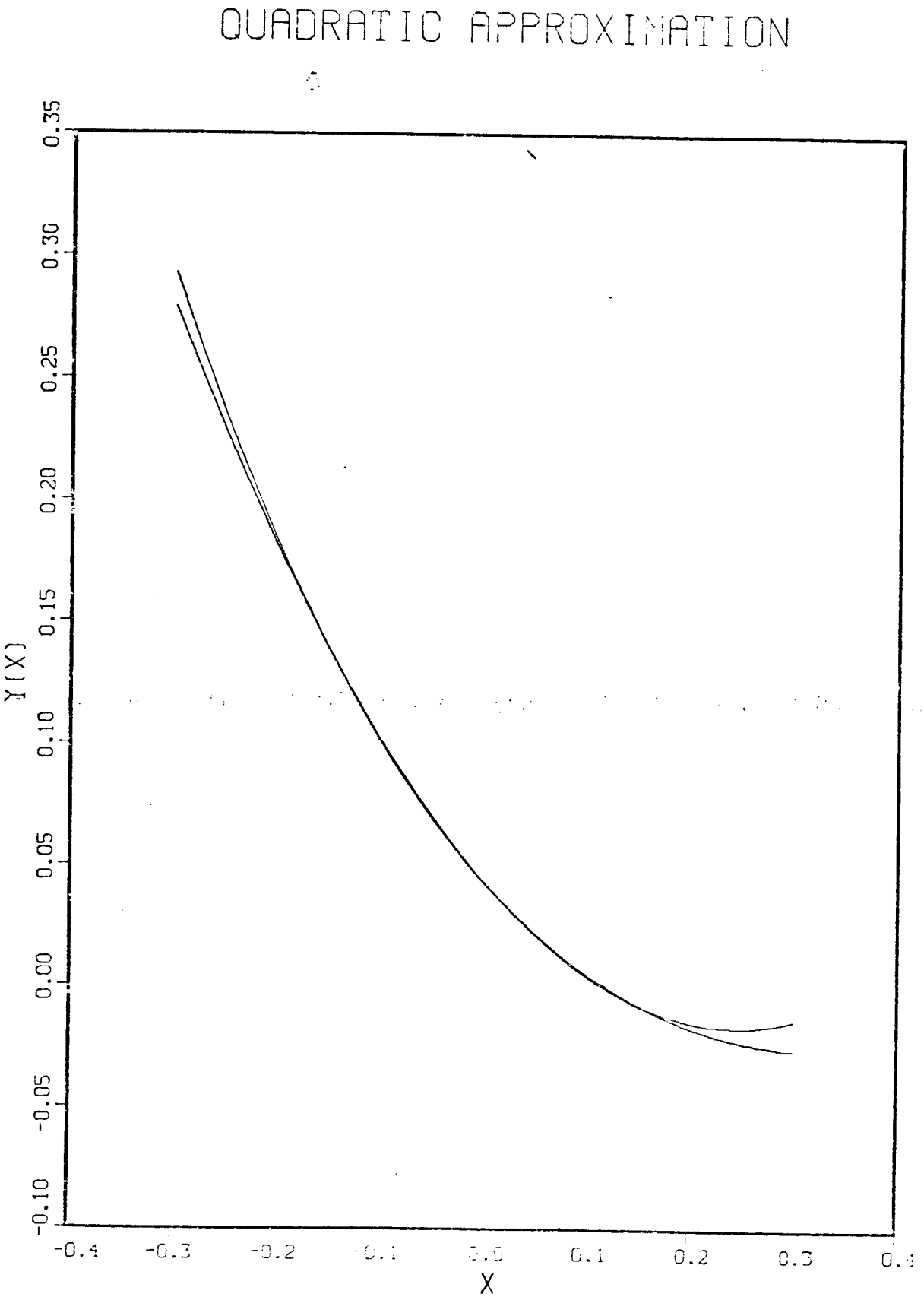


Fig. 1b approximation # 2

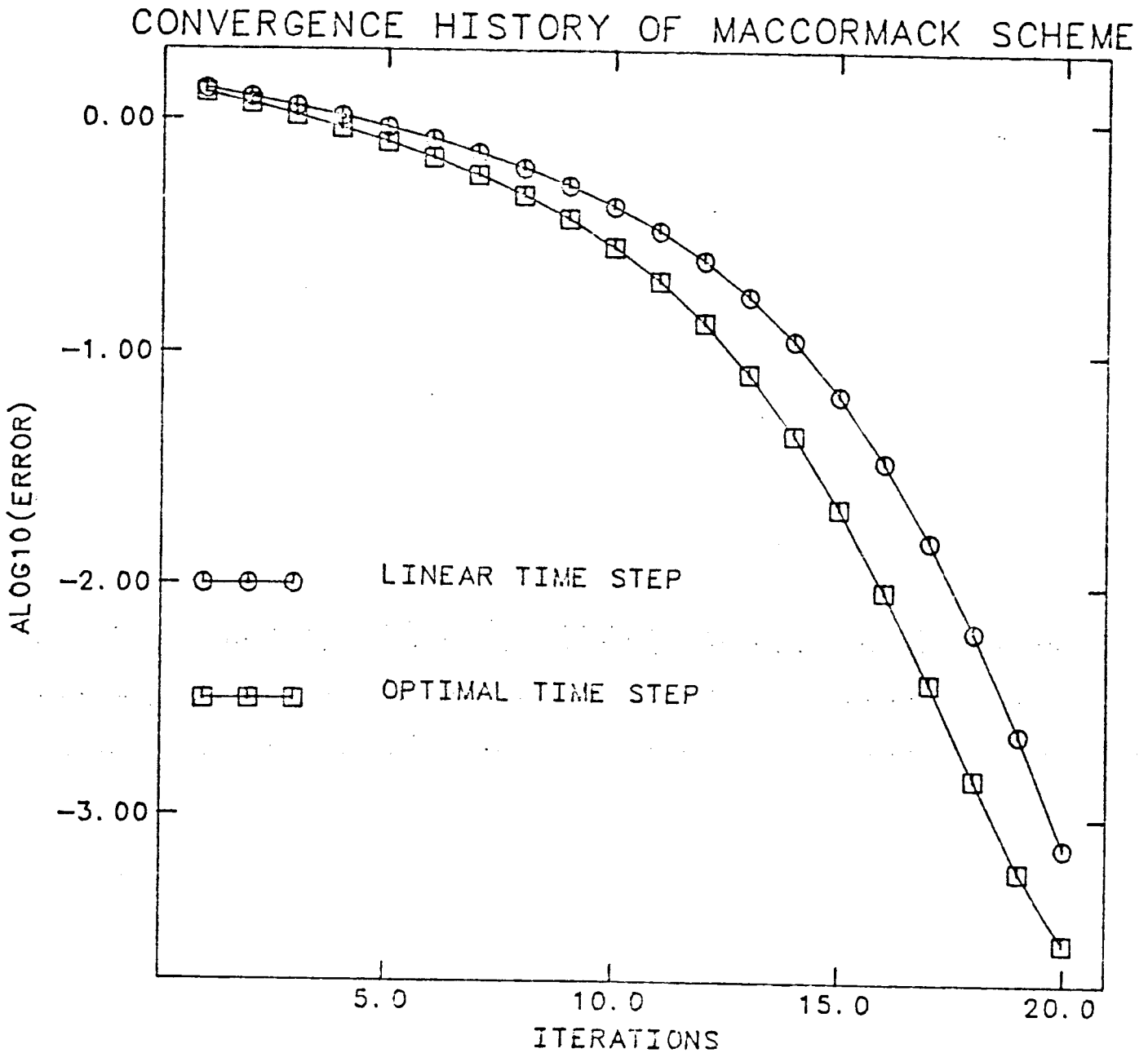


Fig. 5 Convergence history for solution of Burgers' equation using linear and optimal time steps

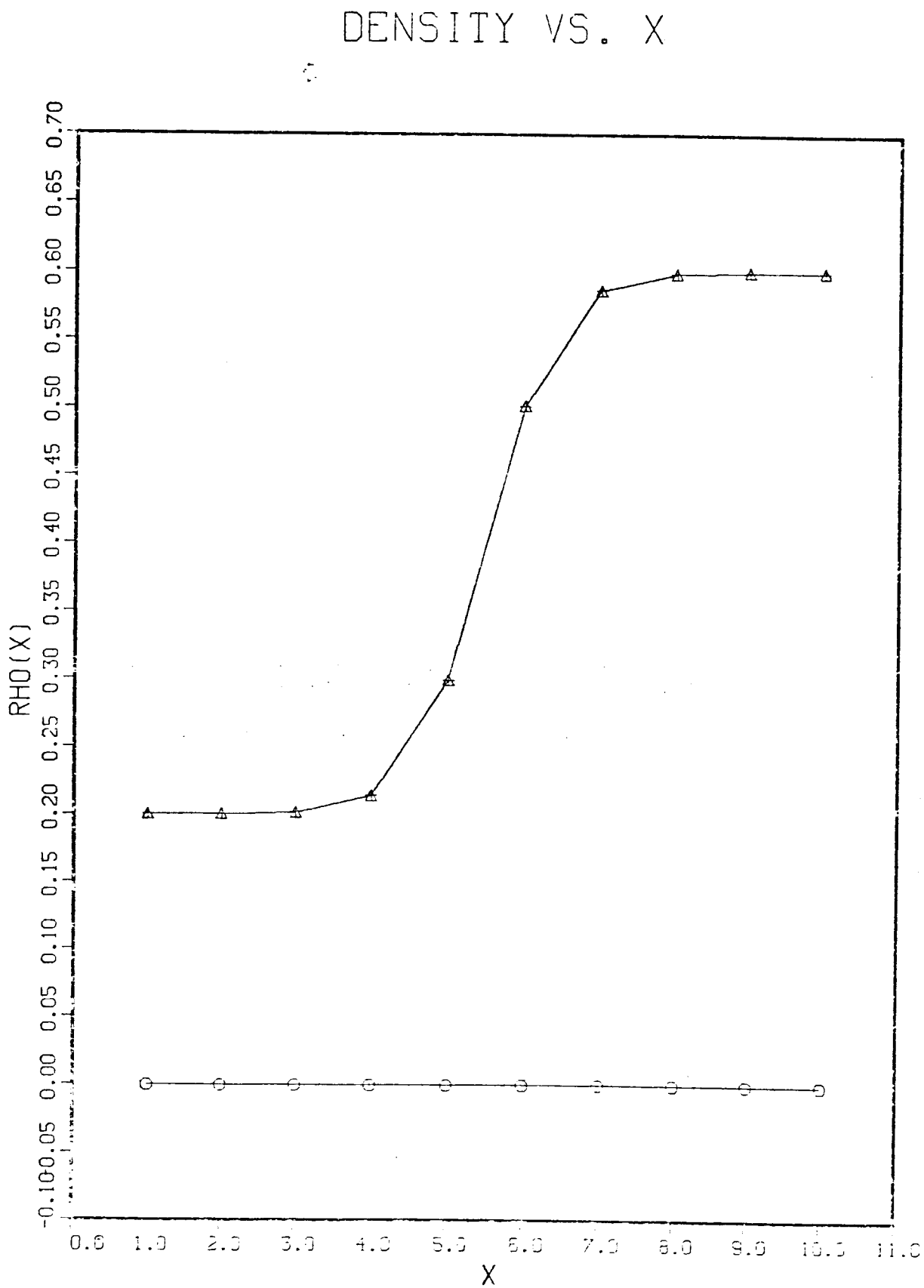
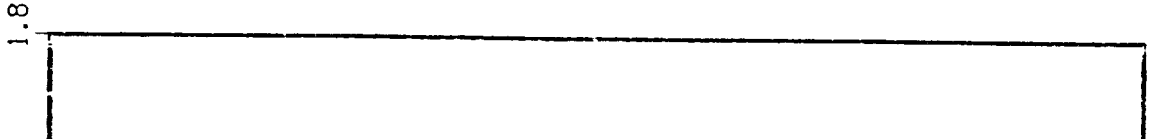


Fig. 4c Iteration # 3 (converged)

DENSITY VS. X



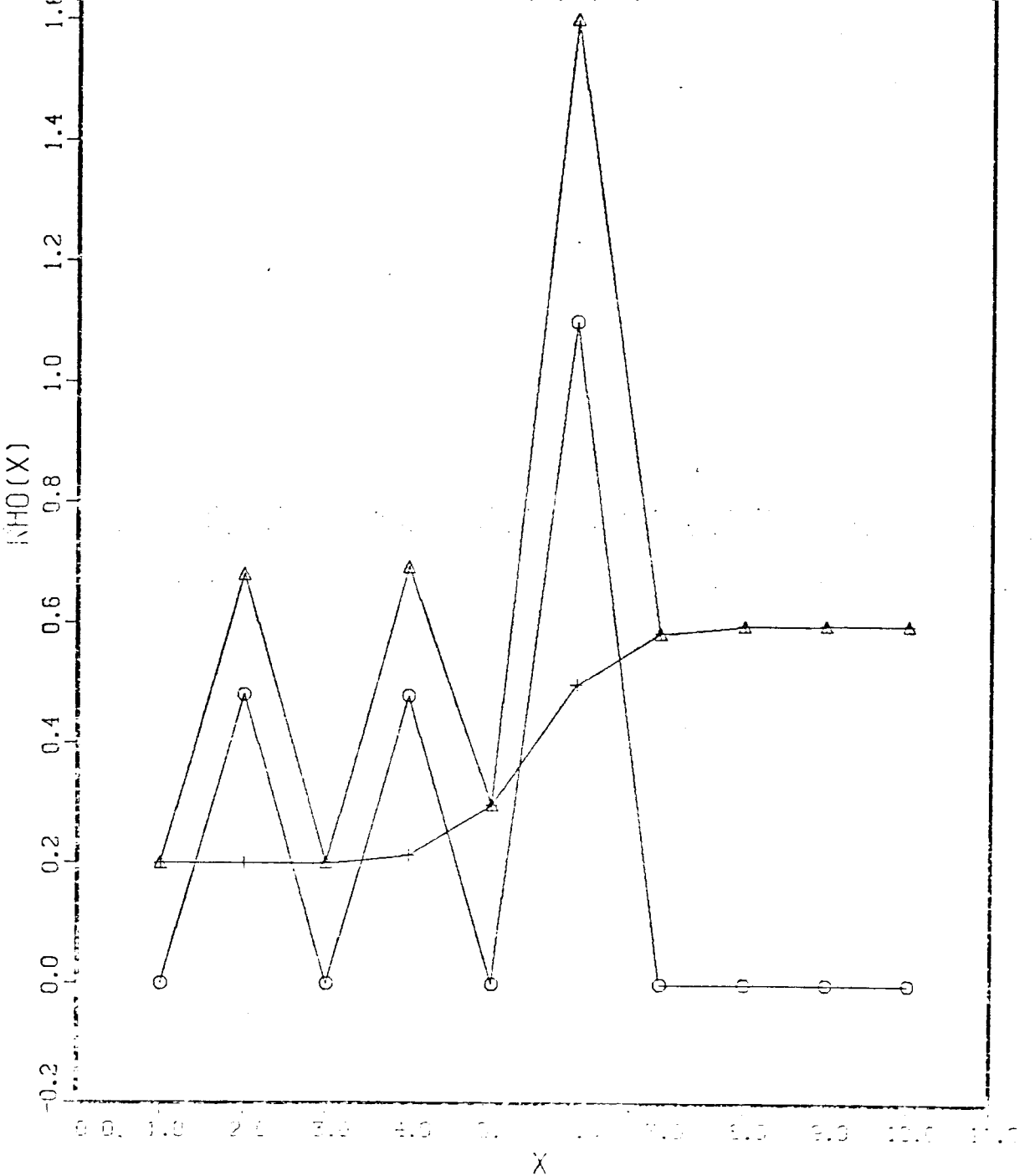


Fig. 4b Iteration # 2

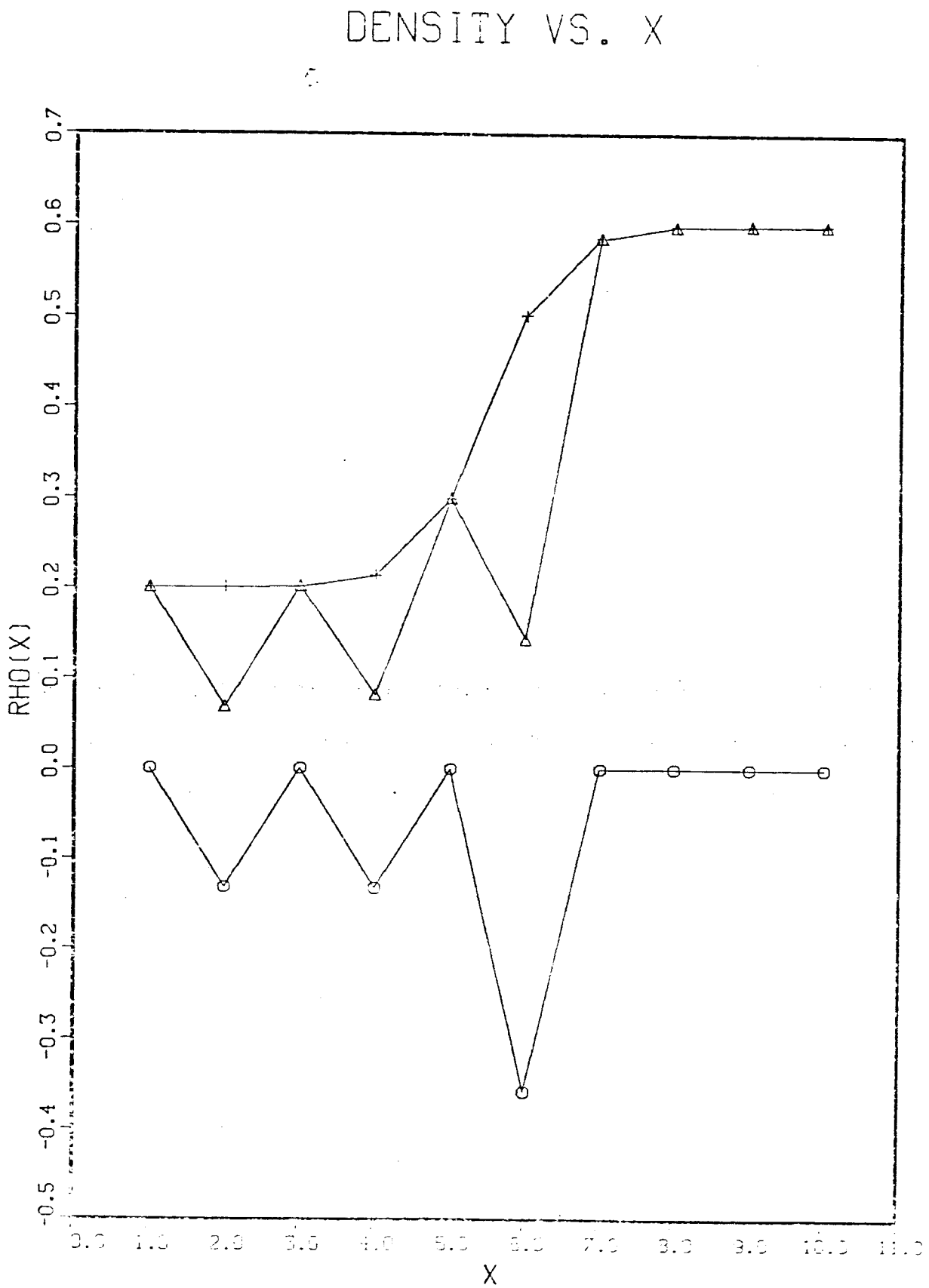


Fig. 4a Convergence history for density: + exact solution, Δ calculated results, o error
iteration # 1

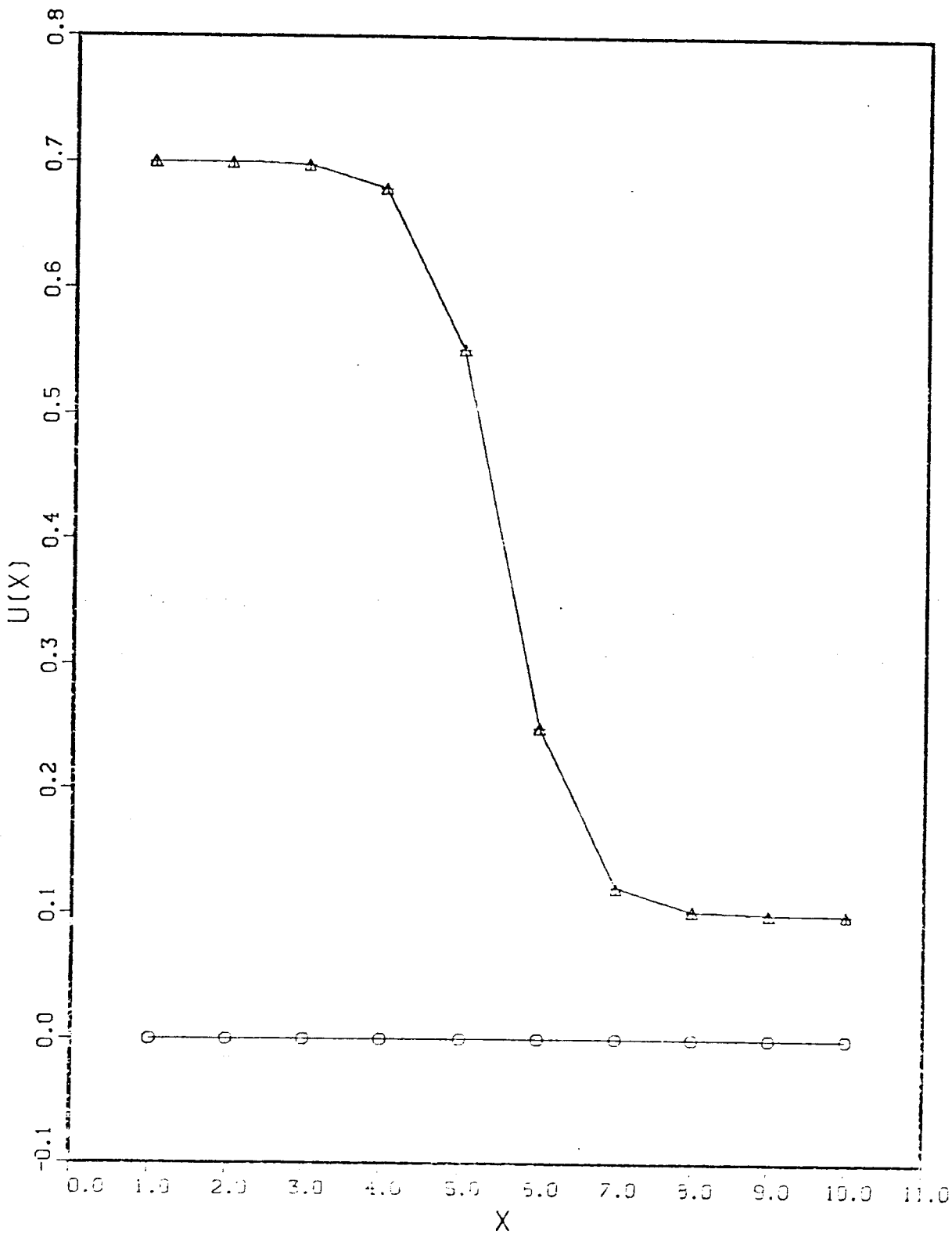


Fig. 3c Iteration # 3 (converged)

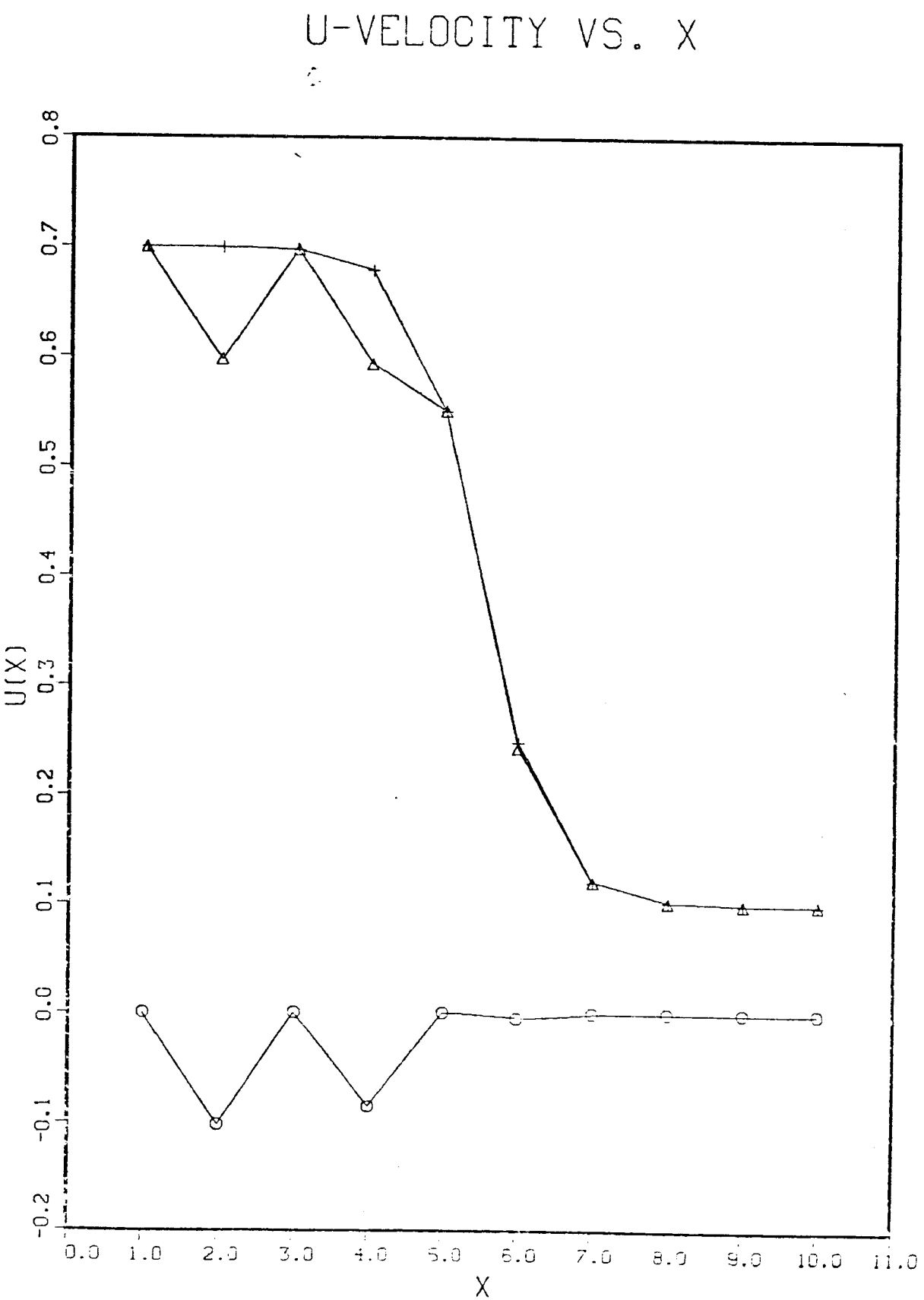


Fig [illegible] Iteration # 2

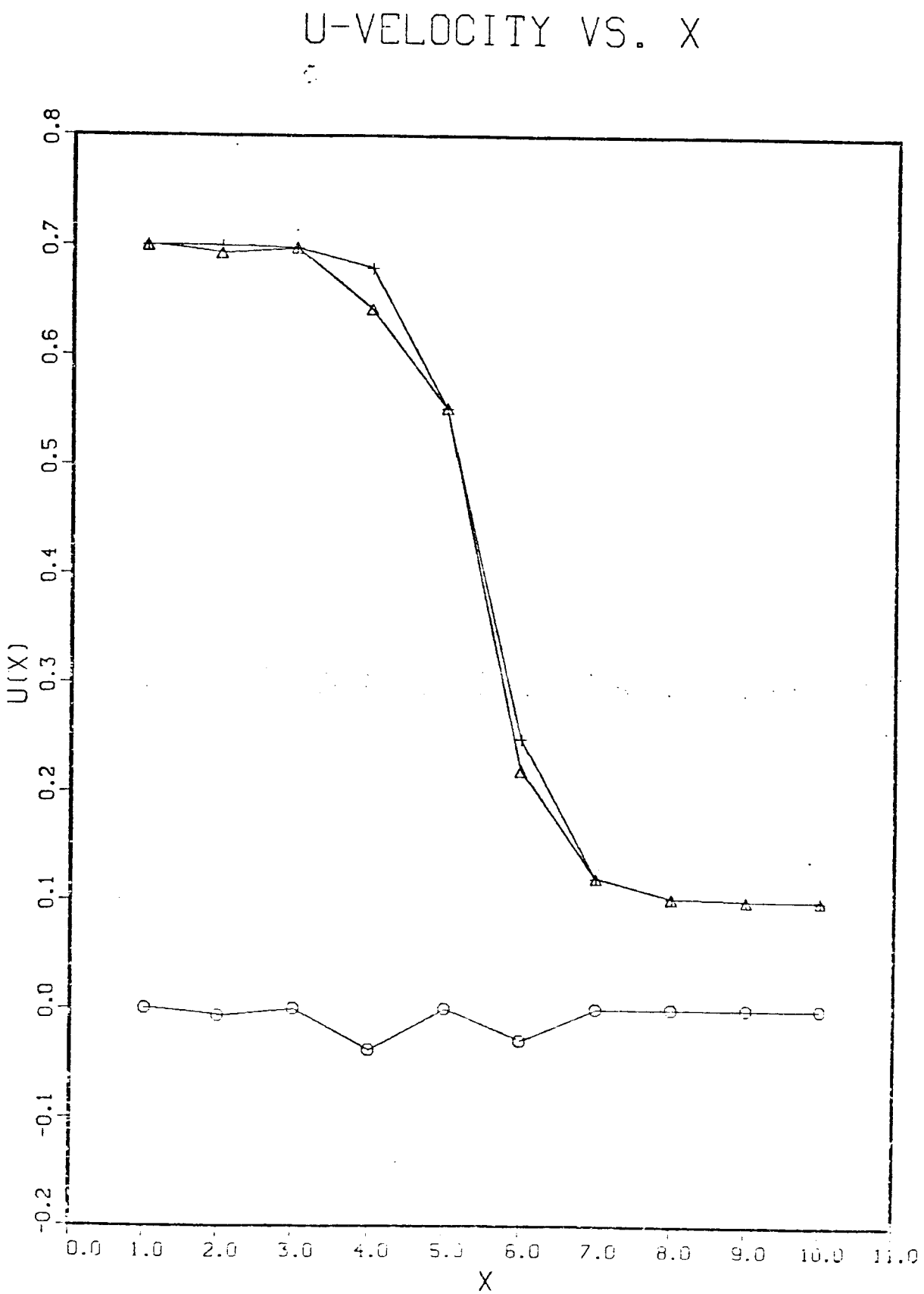


Fig. 3a Convergence history for velocity: + exact solution, Δ calculated results, o error: iteration # 1

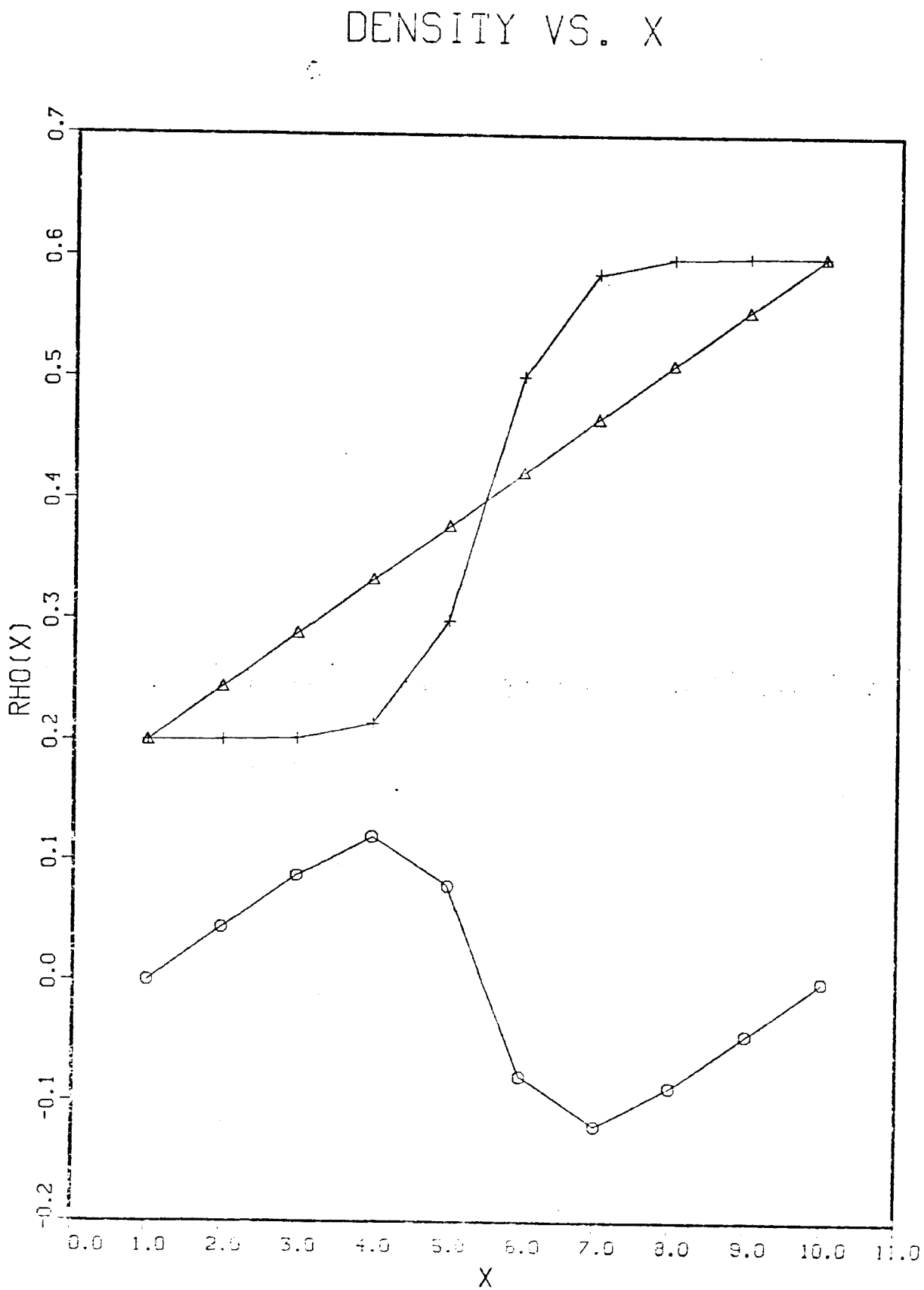


Fig. 2b Exact solution, initial guess and error for the density ρ

+ exact solution, Δ initial guess, o error

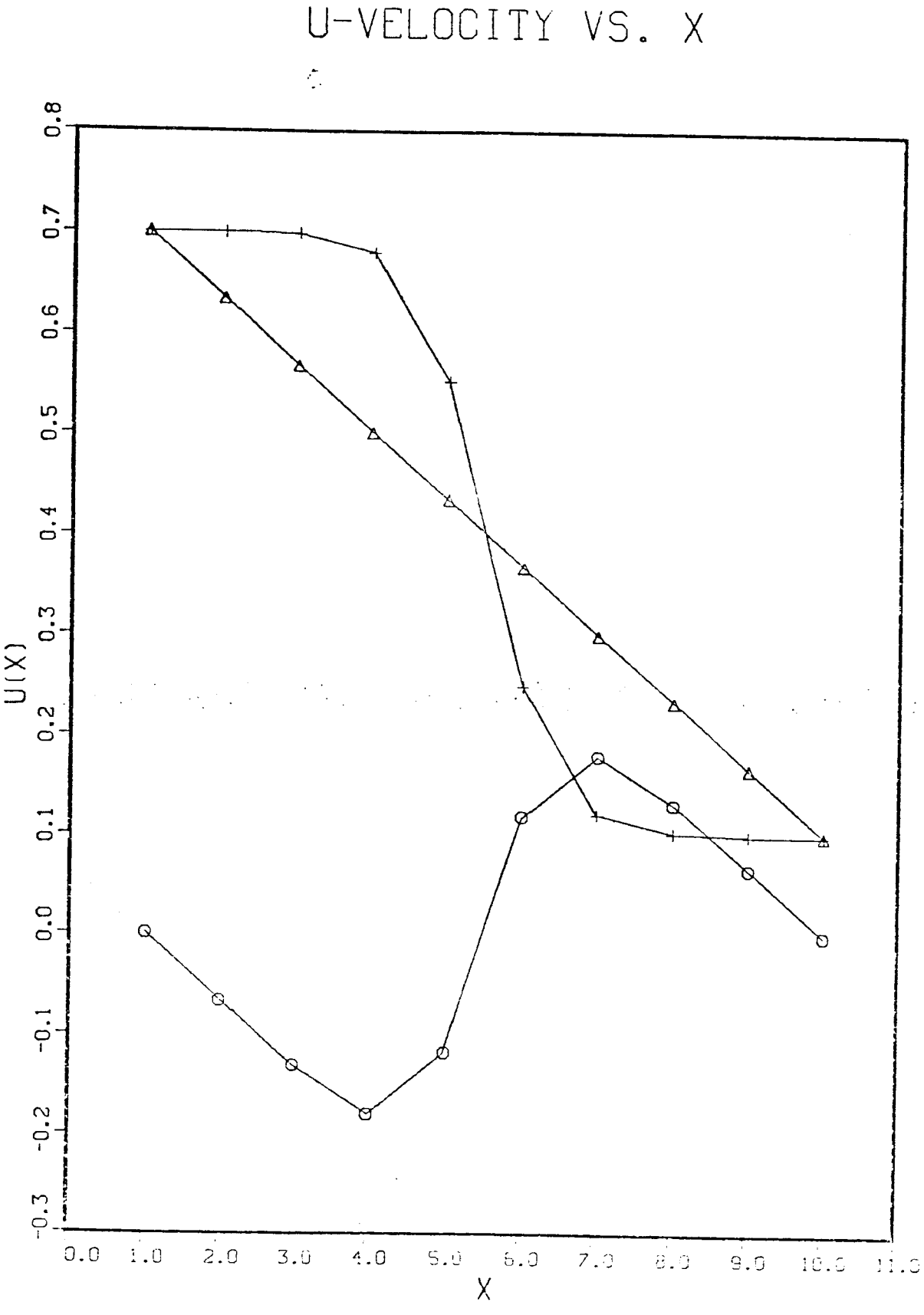


Fig. 2a Exact solution, initial guess and error for the velocity u

+ exact solution, Δ initial guess, o error

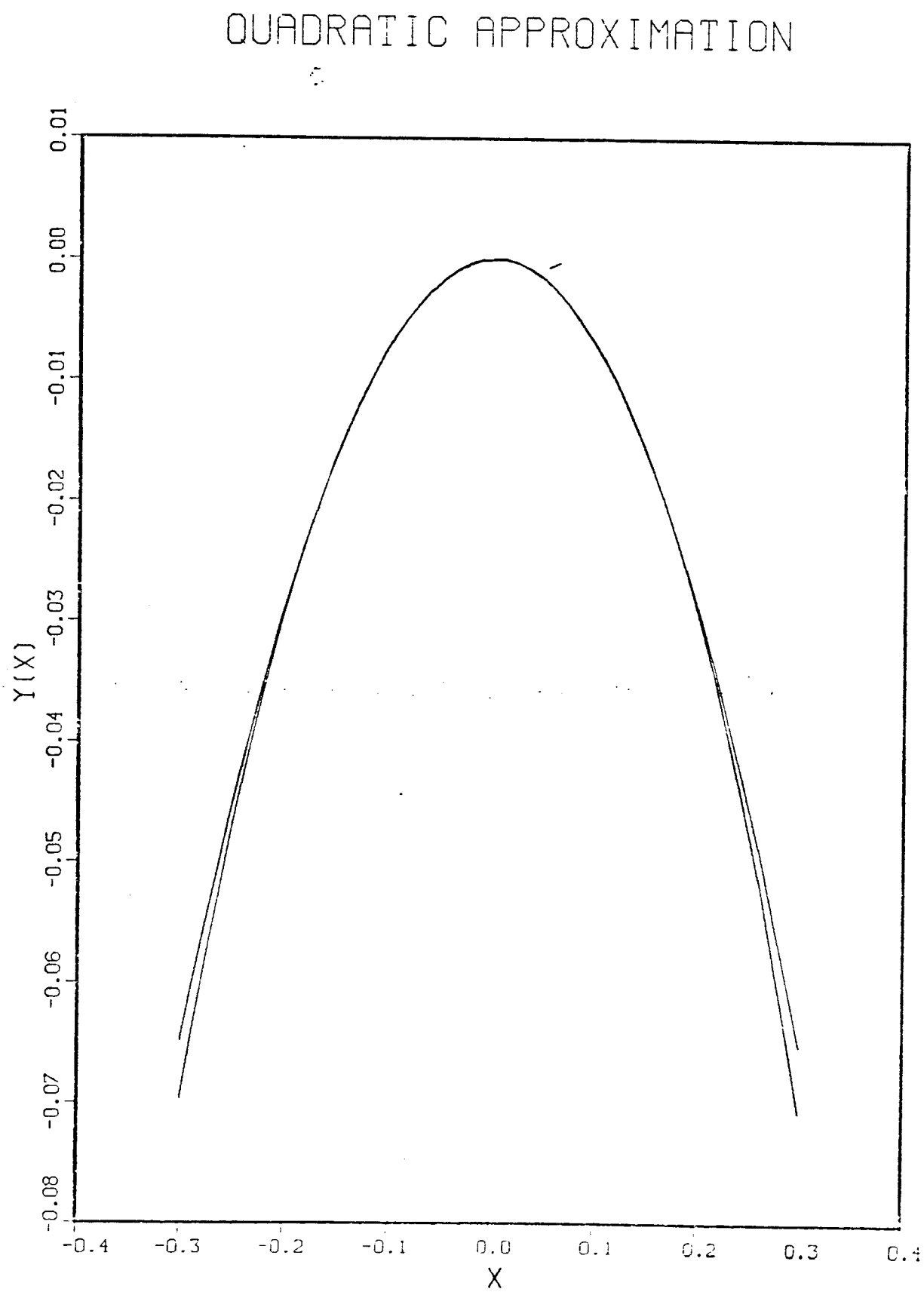


Fig. 1c approximation # 3