NUMERICAL ANALYSIS OF HYDRAULIC FRACTURING AND RELATED CRACK PROBLEMS

by

DONALD RALPH PETERSEN

B.S.E.(M.E.), B.S.E.(Eng. Math.), University of Michigan-Dearborn (1977)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 1980

Signature of Author _____
Department of Mechanical Engineering
January 18, 1980

Certified by _____
Michael P. Cleary
Thesis Supervisor

Accepted by _____
Warren Rohsenow
Chairman, Department Committee

NUMERICAL ANALYSIS OF HYDRAULIC FRACTURING AND RELATED CRACK PROBLEMS

by

DONALD RALPH PETERSEN

Submitted to the Department of Mechanical Engineering
on January 18, 1980 in partial fulfillment of the
requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

The formulation for numerical analysis (by surface integral equation techniques) of crack problems related to hydraulic fracturing is presented along with solutions of several representative plane static and quasi-static problems. A general formulation for static problems involving plane cracks of arbitrary number and orientation in non-homogeneous media is given. Separate formulations for quasi-static problems are included, although, due to their developmental nature, they are restircted in scope to a single stationary plane crack. Results are presented for a static crack approaching and crossing an interface; for the effects of microcracks in adjacent strata and for simple models of crack branching and blunting. Results are also shown for the quasi-static stationary crack problems of pressure evolution in fluid filled cracks and fluid front advancement in partially filled cracks. In addition, the development and current status of a general purpose computer program for the simulation of hydraulic fracturing is discussed.

Thesis Supervisor: Dr. Michael P. Cleary
Title: Associate Professor of Mechanical Engineering

ACKNOWLEDGEMENTS

I wish to thank my advisor, Professor Michael P. Cleary, for the many hours of help which he so willingly gave throughout the course of this project. His advice and suggestions have been extremely helpful and are sincerely appreciated.

I gratefully acknowledge I. Buttlar, Mary Toscano and Nancy Toscano for their patience and high quality of work in the typing of this thesis.

My parents, Mr. and Mrs. Ralph C. Petersen, cannot be thanked deeply enough for their unfailing support and encouragement. They have always provided for me unselfishly, and I am very grateful.

Special thanks are in order for my good friend Mark Proulx, whose willingness to listen to my problems and frustrations, even during the most difficult times of his own work are very much appreciated. Without the companionship of this fellow Detroiter, it would have been much harder to learn some of the more obscure customs of New England life (e.g., "cash and carry").

I also thank Mr. Jon Doyle for his friendship and encouragement.

My many friends and neighbors at Ashdown House have made life here much easier and much more enjoyable.

I would like to express my thanks to the Lawrence Livermore Laboratory and to the National Science Foundation for the generous financial support given to me.

DONALD R. PETERSEN, Cambridge, MA. January, 1980

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

LIST OF FIGURES (CONTINUED)

LIST OF FIGURES (CONTINUED)

LIST OF FIGURES (CONTINUED)

INTRODUCTION

The work presented in this thesis was done as part of an on-going project whose objective is to develop a general purpose computer program capable of full three-dimensional simulation of physically realistic hydraulic fracturing operations in brittle (including porous) media. Attainment of this goal will require the simultaneous capabilities of computing the various structural responses to arbitrarily loaded and oriented sets of cracks (even in highly irregular material regions) and of computing the time dependent loading of those cracks-coupled to the material response-caused by the flow of a viscous (possibly non-Newtonian) fluid within them, and possibly affected by flow of fluid in the pores of surrounding strata. The problems treated herein are mainly simplified versions of the most general problems, and were chosen for their ability to provide various preliminary insights into hydraulic fracturing problems and confidence in our approaches to these problems. Another important aspect of the project, namely program development, is also discussed.

Hydraulic fracturing (see review in [1]), while useful in other applications, is usually thought of as a technique for stimulating oil or gas wells to enhance production. Essentially, it is a means of producing a large crack which serves as a highly permeable passage-way with a large surface area into which gas or oil

can escape from a relatively impermeable rock formation; it can then flow back to the well-bore, even from very large distances. The crack is produced (see Figure (1)) by sealing off a part of the borehole with packers, then pumping in a highly viscous fluid until the pressure between packers is great enough to fracture the rock; pumping is then continued for some time until it is judged (by whatever means of prediction or measurement is available) that the crack has grown to the desired size. The high viscosity of the fluid serves three purposes: it reduces the loss rate through the pores in the rock, it allows much wider cracks (than those corresponding to natural rock toughness), and it enables the fluid to carry along in suspension some form of large particles (e.g., coarse sand or bauxite) which serve to prop the crack open after the fluid pressure is reduced and the well is put into production.

Hydraulic fracturing has been in use for some thirty years, but a disturbing percentage of the jobs attempted still are less than successful. An hydraulic fracturing job would theoretically be deemed a success if the resulting crack has the proper shape: usually this means that the crack extends a great distance away from the borehole without spreading upwards to a comparable extent. Above all else, the fracture should, if possible, be confined to the "pay zone" or region containing the resource being extracted. This last consider-

ation is especially important if the pay zone consists of a narrow stratum and surrounding strata are non-productive or can produce deleterious effects (e.g., unwanted fluids, blow outs or leak-off). Hydraulic fracturing operations can fail for any of a number of reasons, but in the present context we are especially interested in the question of containment. For instance, sometimes fractures may actually propagate primarily upward along the borehole, without ever extending very far away from it. That such occurrences go unpredicted (and often unnoticed) is primarily due to inadequate mechanical analysis of the hydraulic fracturing process.

Most hydraulic fracturing analyses focus upon estimating the surface area (and hence deducing effective length based on an assumed height), rather than trying to trace the detailed geometry of a prospective hydraulic fracture. All of these analyses involve somewhat unreal assumptions about the crack geometry and fluid pressure distribution. Upon reducing the geometry to a function of a single variable, a crack shape is calculated to satisfy mass conservation: the crack volume must make up the difference between the total amount of fluid pumped in and that supposed to have leaked out into the formation (e.g., [1-4]). Some of the more recent work (e.g., [5,6]) has taken into account some of the relevant solid mechanics considerations, but the resulting analyses seem to have

numerous shortcomings and the formulations have little potential for coping with more complex geometries: specifically, no proper solution has yet been obtained (even for the simplest geometries) for the coupled crack-opening and frac-fluid flow process.

Since the problem does not lend itself to closed-form solutions, except for various very approximate formulae, we must employ an appropriate numerical method such as a Surface Integral Equation (SIE) technique or Finite Element (FE) analysis. We have chosen to work with a particularly attractive SIE scheme [8], which will be discussed in detail in the chapters that follow. This SIE scheme has the advantage over others (e.g. [7]) of giving displacement type solutions based on known tractions and requiring only fundamental solutions which are well known [8]. In general, SIE schemes are more facile than FE analysis for these types of problems: they are based on surface (rather than volume) discretization, and so are not only more economical in modeling crack surfaces, but better suited for problems involving infinite or semi-infinite regions. However, there may be a need in some cases to use either FE analysis or a suitable "hybrid" SIE/FE scheme [9] for problems in highly irregular or nonlinear regions, owing to the SIE scheme's requirement of a fundamental solution for each particular region. We emphasize, though, that enough such fundamental solutions do exist [8]

to give the SIE scheme enromous potential, and we can, indeed, compute the required numerical values for some influence functions that have not been worked out analytically. Thus, we regard this approach -- although limited to plane problems in this thesis -- as having the ability for realistic fully 3-D modelling of hydraulic fracturing processes in the future.

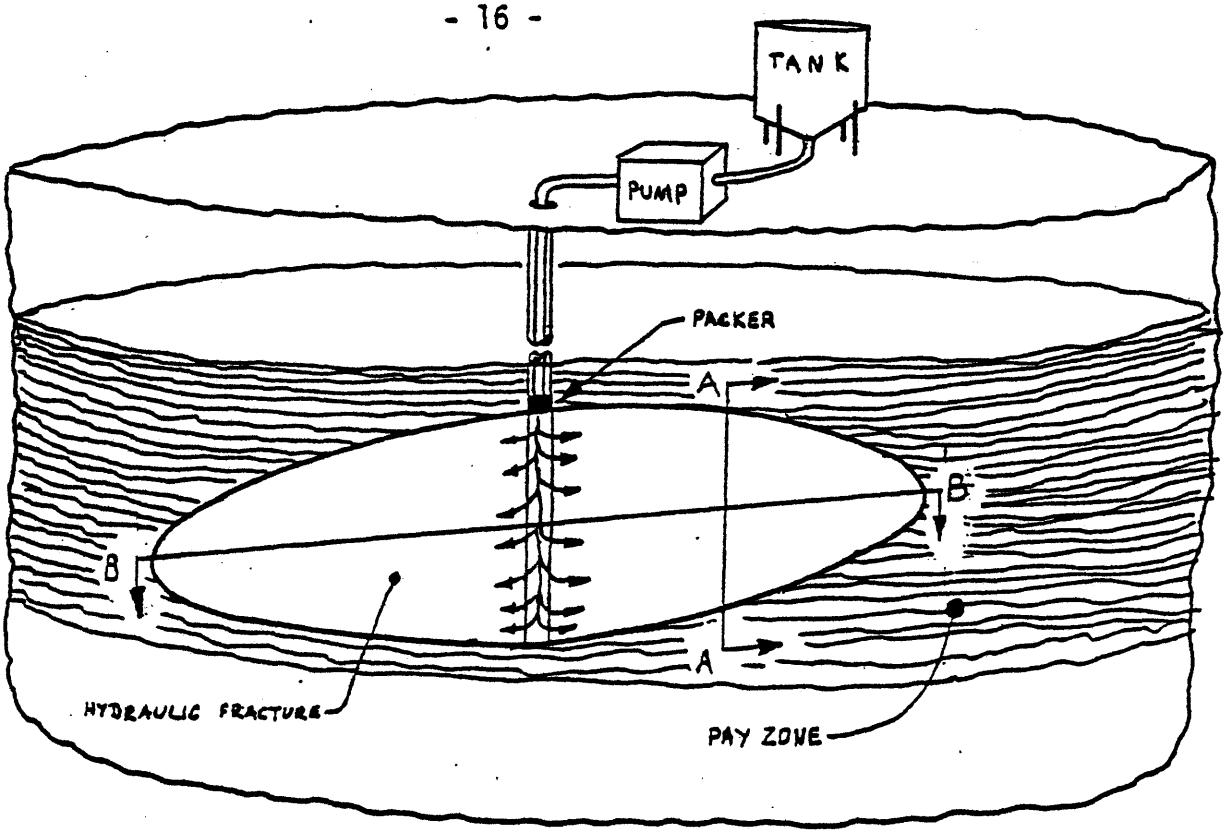


FIG. 1. Diagram of a typical hydraulic fracturing operation. Here the pay zone consists of a fairly narrow stratum in which the crack must be contained. our plane crack models represent cross sections, such as A-A or B-B, of such an hydraulic fracture.

CHAPTER 1: FORMULATION OF PLANE STATIC CRACK PROBLEMS FOR NUMERICAL ANALYSIS

We perform our analyses of crack problems with a special form of Surface Integral Equation, solution of which yields the density of dislocations or dipoles (distributed over the boundary of the region of interest) required to produce a known traction distribution on the same boundary. The method employs the fundamental solution of the governing equation pertaining to the region. It requires that the boundary be broken up into a number of discrete elements. The traction at any point on the boundary is then expressed as the sum of the integrals over each element of the product of the dislocation or dipole density and the fundamental solution. The result is an integral equation in terms of the unknown dislocation or dipole density. This particular version of classical SIE schemes [e.g., Ref. 9] has been applied to a variety of solid mechanics problems; in particular, Cleary [Ref. 10] has used it in investigations of a number of phenomena germane to the present topic. In our work, we model a crack as a distribution of dislocations (or dislocation dipoles) and determine the dislocation density required to produce the known traction on the crack surface. The region of interest, then, is the body of material containing the cracks under study. Thus for static problems in the plane (quasi-static problems will be treated separately in Chapter 3) we obtained [15]

$$\sigma^{(\alpha i)}(\underset{\sim}{x}) = \sum_{j} \int_{S_j} \Gamma^{(\alpha\beta i j)}(\underset{\sim}{x},\underset{\sim}{t})\, \mu^{(\beta j)}(\underset{\sim}{t})\, dS_j \tag{1.1}$$

where $\sigma^{(\alpha,i)}(\underset{\sim}{x})$ is the α traction component at point $\underset{\sim}{x}$ on element i, $\mu^{(\beta j)}(\underset{\sim}{t})$ is the β component of the dislocation density at point $\underset{\sim}{t}$ on element j. $\Gamma^{(\alpha\beta ij)}(\underset{\sim}{x},\underset{\sim}{t})$ is the fundamental solution (or influence function) which gives the stress $\sigma^{(\alpha i)}$ at $\underset{\sim}{x}$ due to a dislocation at $\underset{\sim}{t}$. The particular influence function used in this work is that for a dislocation near an interface, and its most important feature is its inverse distance singularity (Γ is given in complete form in Ref. [8].)

For plane problems, we represent S_j by the function $\underset{\sim}{t}(\xi)$ and S_i by $\underset{\sim}{x}(\eta)$, where $\xi,\eta \in [-1,1]$ with respect to the global origin. Equation (1.1) then becomes

$$\sigma^{(\alpha i)}(\underset{\sim}{x}(\eta)) = \sum_{j,\beta} \int_{-1}^{1} \Gamma^{(\alpha\beta i j)}(\underset{\sim}{x}(\eta),\underset{\sim}{t}(\xi))\, \mu(\underset{\sim}{t}(\xi))\, E_j\, d\xi \tag{1.2a}$$

where

$$E_j = \left[\frac{d\underset{\sim}{t}}{d\xi} \cdot \frac{d\underset{\sim}{t}}{d\xi}\right]_j^{1/2} \tag{1.2b}$$

It is most convenient to write Eq. (1.2) in terms of traction components that are normal and tangent to each element S_i, and dis-

location density components that correspond to the global $\underset{\sim}{i}_1$ and $\underset{\sim}{i}_2$ (x_1 and x_2) directions. Thus, the directions α and β refer to different coordinate systems, and in order to compute Γ, the stress in the global system (σ_{11}, σ_{12}, σ_{22}) must be transformed to coordinates normal and tangent to S_i:

$$\sigma^{(1i)} = -\tfrac{1}{2}(\sigma_{11}-\sigma_{22})\sin(2\varphi_i+\pi) + \sigma_{12}\cos(2\varphi_i+\pi)$$

$$\sigma^{(2i)} = \tfrac{1}{2}(\sigma_{11}+\sigma_{22}) + \tfrac{1}{2}(\sigma_{11}-\sigma_{22})\cos(2\varphi_i+\pi)$$

$$+\sigma_{12}\sin(2\varphi_i+\pi) \qquad (1.3)$$

where ϕ_i is the angle of inclination of S_i with respect to the global x_1 axis.

In order to solve Eq. (1.2) numerically, we must re-express it as a system of linear algebraic equations. This task may be accomplished by either of two means, namely local or global interpolation of the dislocation densities. The local interpolation approach consists of dividing each crack surface into a number of elements, then representing the dislocation density in terms of interpolation or "shape" functions defined locally on each element (Cleary [16] has performed extensive numerical computations using a "triangular" interpolating function):

$$\mu^{(\beta j)} = \sum_{k=1}^{N_j} a_{kj} m_k^{(\beta j)}(\underset{\sim}{\xi}) \quad ; \underset{\sim}{\xi} \in S_j \tag{1.4}$$

where the N_j interpolation functions on each element j are decided upon a priori in accordance with the problem being solved. Equation (1.2a) thus becomes

$$\sigma^{(\alpha i)}(\underset{\sim}{x}(\eta)) = \sum_{j,\beta} \sum_{k} a_{kj} \left[\int_{-1}^{1} \Gamma^{(\alpha\beta i j)}(\underset{\sim}{x}(\eta), \underset{\sim}{\xi}(\xi)) m_k^{(\beta j)}(\underset{\sim}{\xi}(\xi)) E_j \, d\xi \right] \tag{1.5}$$

If each element is sub-divided into discrete nodes by choosing generic sets of points (or nodes) η_r, $r = 1, \ldots, N_i$ and ξ_i, $i = 1, \ldots, M_j$ at which to evaluate σ and the m_k, the desired system of linear algebraic equations is obtained. The usefulness of the local interpolation method in fracture problems has been investigated by Wong [11], who has used it with some success in dislocation dipole formulations.

In the global interpolation method, each crack is treated as a single boundary element on which we may conveniently express the dislocation densities in terms of interpolation functions, now defined over an entire crack surface (hence the name "global"):

$$\mu^{(\beta j)}(\underset{\sim}{\xi}(\xi)) = \frac{F^{(\beta j)}(\underset{\sim}{\xi}(\xi))}{(1+\xi)^{\alpha}(1-\xi)^{\beta}} \tag{1.6}$$

The parameters α and β are chosen to reflect the anticipated singularity in density of dislocation μ (which is actually just the derivative $d\delta/dx$ of the crack width δ). The choice $\alpha = \beta = 0.5$ is exact for cracks in homogeneous media and, for reasons that will be discussed below, is an advantageous approximation even for modelling of cracks in non-homogeneous media.

Erdogan and Gupta [12] have developed a method of solving singular integral equations of the form

$$S(x) = \frac{1}{\pi}\int_{-1}^{1} \frac{F(t)\,dt}{(t-x)\sqrt{1-t^2}} \tag{1.7}$$

based on the Gauss-Chebyshev integration formula

$$\frac{1}{\pi}\int_{-1}^{1} \frac{F(t)\,dt}{\sqrt{1-t^2}} = \frac{1}{M}\sum_{k=1}^{M} F(t_k)\ ;\ t_k = \cos\left(\frac{\pi(2k-1)}{2M}\right)\quad k = 1, \ldots, M \tag{1.8}$$

Their formula is

$$S(x_r) = \frac{1}{\pi}\int_{-1}^{1} \frac{F(t)\,dt}{(t-x)\sqrt{1-t^2}} = \frac{1}{N}\sum_{k=1}^{N} \frac{F(t_k)}{(t_k - x_r)} \tag{1.9a}$$

$$t_k = \cos\left(\frac{\pi(2k-1)}{2N}\right)\quad k = 1, \ldots, N \tag{1.9b}$$

$$x_r = \cos\left(\frac{\pi r}{N}\right)\quad r = 1, \ldots, N-1 \tag{1.9c}$$

where the t_k are the zeroes of the Chebyshev polynomials of the first kind, $T_N(t)$, and x_r are the zeroes of $U_{N-1}(x)$, the Chebyshev polynomials of the second kind. Since the singular part of Γ is $(x-t)^{-1}$, in general, this formula is very well suited to use in our work, and provides a very simple and economical means of solving Eq. (1.2). This formula is based upon the observation that $(1-t^2)^{-1/2}$ is the weighting factor for the Chebyshev polynomials. A similar formula [13] has been developed for other, arbitrary choices of α and β, based on the Gauss-Jacobi integration formula; because the required computation of the zeroes of the Jacobi polynomials is relatively difficult and time consuming, we have used the Gauss-Chebyshev method in all of our work, without any apparent loss in accuracy for the answers that we have been interested in extracting.

If we now define discrete points η_r and ξ_k:

$$\eta_\pi = \cos\left(\pi\pi/(N_i-1)\right) \quad \pi = 1,\dots,N_i-1 \tag{1.10a}$$

$$\xi_k = \cos\left(\pi(2k-1)/2N_i\right) \quad k = 1,\dots,N_i \tag{1.10b}$$

and substitute Eq. (1.6) with $\alpha = \beta = 0.5$ into Eq. (1.2), then apply Eq. (1.8) we get

$$\sigma^{(\alpha i)}(\underset{\sim}{x}(\eta_\pi)) = \sum_{j,\beta} \frac{\pi}{N_j} \sum_{k=1}^{N_j} \left[\Gamma^{(\alpha\beta ij)}(\underset{\sim}{x}(\eta_\pi), \underset{\sim}{t}(\xi_k)) E_j \right] F^{(\beta j)}(\underset{\sim}{t}(\xi_k)) \tag{1.11}$$

which is the final form of our system of algebraic equations. Note that since on each element there is one less η_r than ξ_k, the system (1.11) will require several additional equations for completion, the number depending upon the number of crack surfaces and the range of α and β. Such additional conditions may be either contraints on the net entrapped dislocation (called closure conditions) or matching of dislocation densities (matching conditions) if two or more of the cracks intersect, depending upon the problem under investigation. The closure conditions may be stated for any plane crack problem as integrals of the appropriate dislocation densities:

$$\sum_{j} \int_{-1}^{1} \mu^{(\beta j)}(\pm(\xi)) E_j \, d\xi = 0 \approx \sum_{j} \frac{\pi}{N} \sum_{k=1}^{N_j} F^{(\beta j)}(\pm(\xi_k)) E_j \tag{1.12}$$

for one or more crack surfaces S_j, where the sum is taken over intersecting cracks. Since there is a variety of matching conditions, each generally applicable only to a particular problem, they are discussed separately in appropriate sections of Chapter 2.

An illustrative example of the type of plane crack problems that we are equipped to solve is depicted in Figure (1.1); note, however, that we can include more than two surfaces, some or all

of which may intersect, and we can also solve problems in which these cracks are near an interface. In this case, we have two straight crack surfaces, so that for surface S_1

$$\begin{Bmatrix} x(\eta) \\ y(\xi) \end{Bmatrix} = \frac{1}{2}\left[t_1^B\left(1-\begin{Bmatrix}\eta\\ \xi\end{Bmatrix}\right)+t_1^A\left(1-\begin{Bmatrix}\eta\\ \xi\end{Bmatrix}\right)\right] i_1 + \frac{1}{2}\left[t_2^B\left(1-\begin{Bmatrix}\eta\\ \xi\end{Bmatrix}\right)+t_2^A\left(1+\begin{Bmatrix}\eta\\ \xi\end{Bmatrix}\right)\right] i_2 \tag{1.13a}$$

$$E_1 = \frac{1}{2}\left[(t_1^B - t_1^A)^2 + (t_2^B - t_2^A)^2\right]^{1/2} \tag{1.13b}$$

and, for S_2

$$\begin{Bmatrix} x(\eta) \\ y(\xi) \end{Bmatrix} = \frac{1}{2}\left[t_1^D\left(1-\begin{Bmatrix}\eta\\ \xi\end{Bmatrix}\right)+t_1^C\left(1+\begin{Bmatrix}\eta\\ \xi\end{Bmatrix}\right)\right] i_1 + \frac{1}{2}\left[t_2^D\left(1-\begin{Bmatrix}\eta\\ \xi\end{Bmatrix}\right)+t_2^C\left(1+\begin{Bmatrix}\eta\\ \xi\end{Bmatrix}\right)\right] i_2 \tag{1.13c}$$

$$E_2 = \frac{1}{2}\left[(t_1^D - t_1^C)^2 + (t_2^D - t_2^C)^2\right]^{1/2} \tag{1.13d}$$

Solution of Eqns. (1.11) now produces the strength $F^{(\beta j)}$ of dislocation density in the β-direction on each surface j. The stress intensity factors may be computed from the relations (similar to those given by Cleary [14]) after transforming $F^{(\beta j)}$ into local coordinates, namely

$$\begin{aligned} F^{(Tj)} &= F^{(1j)}\cos(\varphi_j) + F^{(2j)}\sin(\varphi_j) \\ F^{(Nj)} &= -F^{(1j)}\sin(\varphi_j) + F^{(2j)}\cos(\varphi_j) \\ K^{(\beta j)} &= \frac{2G}{1-\nu}\sqrt{\frac{\pi l_j}{2}}\,F^{(\beta j)} \end{aligned} \tag{1.14}$$

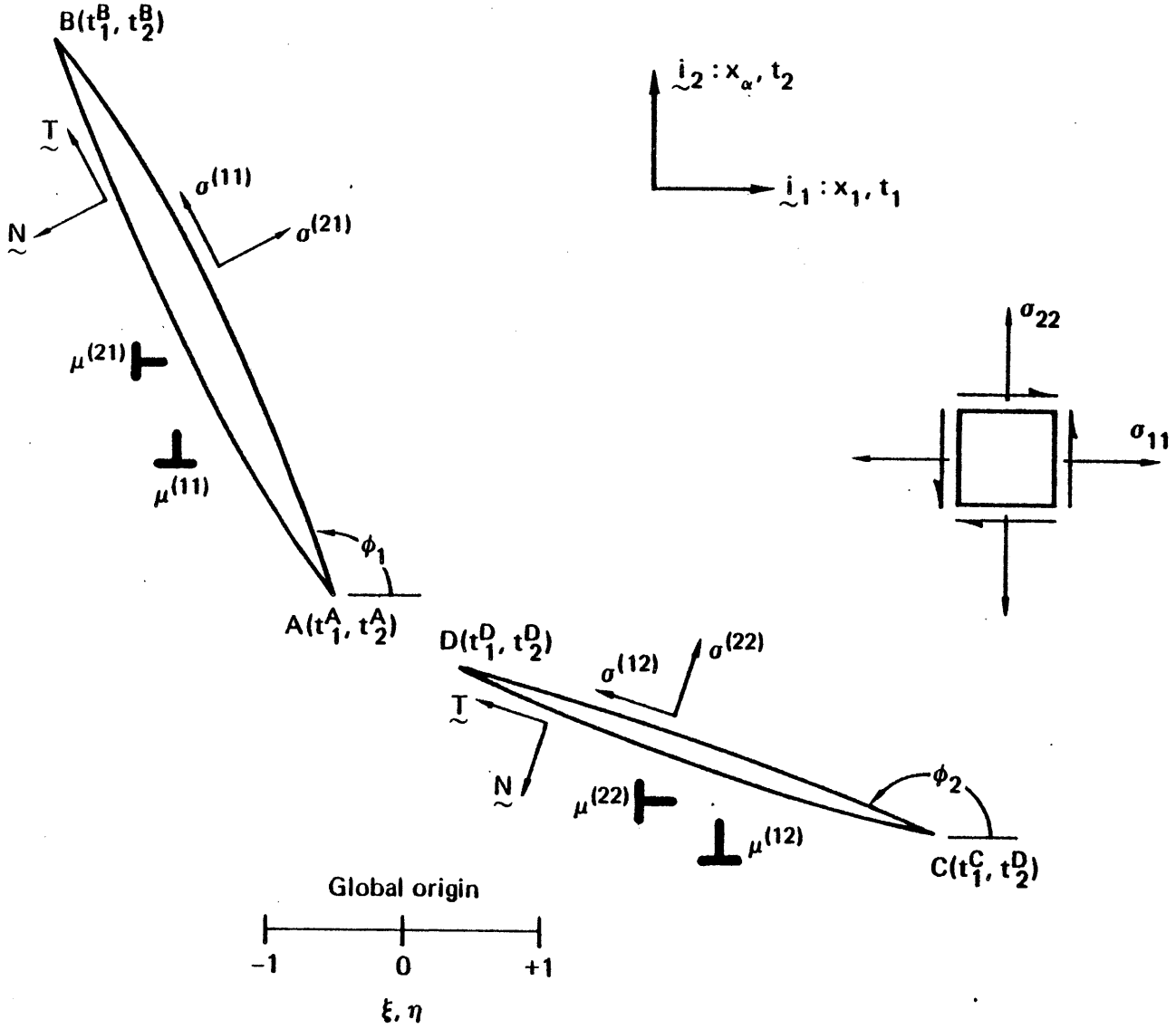


Fig. 1.1. Diagram showing coordinates and angles needed to formulate the general two-dimensional crack problem for numerical solution by the Gauss-Chebyshev scheme.

CHAPTER 2: ANALYSIS OF STATIC-CRACK PROBLEMS

2.1 Introduction

In this Chapter the results of our investigations of several static plane crack problems are presented. These problems (the numerical formulations of which were presented in Chapter 1) were chosen for the dual purposes of gaining preliminary insight into some relevant hydraulic fracturing situations and of perfecting the models to be used for studying similar crack geometries in quasi-static simulations (which account for the non-dynamic time-dependent loading due to frac. fluid flow, as discussed in Chapter 3). Thus we have progressed toward the capability of simulating the changing of course (branching), blunting, containment in or breaking out of a stratum, and the effect of zones of damage or microscopic flaws on propagating hydraulic fractures.

2.2 Straight Crack Near an Interface

Perhaps the most important goal in the design of an hydraulic fracture is containment of the fracture in the "pay zone", or resource-bearing stratum. Thus, the first problem that we undertook to study was to determine the behavior of the opening mode stress intensity factors at the tips of a crack approaching and eventually penetrating an interface with a material having different elastic moduli (Figure 2.1).

This problem has been discussed by Cleary [17], and very similar problems have been solved numerically by Erdogan and his co-workers [13,21]. The model for a crack approaching an interface simply involves a single crack surface, consisting of a distribution of dislocations, employing the influence function for a dislocation near an interface. To model a crack which extends through the interface, however, we found it most effective to employ two crack surfaces which intersect each other tip-to-tip at the interface. The advantage of the two-crack model is that a large number of nodal points are concentrated around the interface, owing to the spacing required by the Gauss-Chebyshev scheme. With the two-crack model, we require two additional conditions to complete our system of equations. One of these is the requirement that there be no net entrapped dislocation (i.e. the crack must close at both ends), given by Equation (1.12). The other is a "matching condition" relating the value of $F^{(11)}(0)$ to that of $F^{(12)}(0)$. For this we adapt the condition used by Erdogan and Biricikoglu [13].

Their matching condition is a requirement that must be met to insure consistency between their solution and the calculated power of the stress singularity at the crack tips intersecting the interface: in its full form, it is quite a complicated relation, but for our purposes a simpler version which embodies the essential features of theirs seems to suffice. Thus, we use the following relation:

$$F^{(11)}_{(0)} = \alpha F^{(12)}_{(0)} \sqrt{b_1/a_2} \qquad (2.1)$$

This condition is imposed at the two nodal points closest to either side of the interface. Our results indicate that the choice of α does not have an important effect. It may, however, be best to choose $\alpha = 0$ (see Sec. 2.4). In fact, although Erdogan and Biricikoglu use a Gauss-Jacobi scheme which gives better account of the fact that the stress singularity for a crack tip at an interface is not inverse square-root of distance, we are able to essentially reproduce their results, especially for behavior of stress intensity factors, with our Gauss-Chebyshev method. It seems likely that this agreement is due to the difference from 0.5 of the power of the singularity at the interface having only a very local effect on the solution. By choosing enough nodal points we can smooth out any imposed perturbation in the solution at the two points nearest the interface so that the solution at the crack tips remains relatively unchanged.

The results, which illustrate the behavior of K_I for varying tip-to-interface distances and relative shear moduli, are shown in Figure 2.2 (a) (for a crack approaching the interface) and Figure 2.2(b) (for a crack having penetrated the interface). The dependence of K_I on d/A and g is as anticipated by Cleary [17] who made his deduction on the basis of simple material deformation

matching arguments. As the crack approahces an interface with a stiffer medium ($g < 1$), K_I at tip A drops sharply to zero, whereas it rises sharply toward infinity if the interface is with a softer material ($g > 1$). For a crack which has penetrated an interface, going from a stiff material to a softer one, K_I at tip A has been found to drop sharply (as shown) from infinity, reach a broad local minimum, and gradually become asymptotic to its value remote from the interface. For a crack which has broken out of a soft material into a stiffer one (having somehow overcome the apparent "elasticity barrier" noted above) K_I rises sharply from zero, levels off, and remain nearly constant until $d/A = -2$ (the point at which tip B crosses over the interface), whereupon it drops sharply toward its remote value. The behavior of K_{IB} (K_I at tip B) can be obtained from Figure 2.2 by complementing d/A and inverting g.

While the strong decrease in K_I at the tip of a crack approaching a stiff adjacent medium leads us to conclude that the crack might be contained in the softer stratum, some care is required when applying this conclusion. The crack may break through the interface if, for example, the range in which de-cohesion takes place is greater than the distance at which K_I becomes strongly influenced by the interface. Also, as will be discussed in the next section, microcracks in the stiffer medium can be induced to propagate across the interface and link up with an hydraulic fracture.

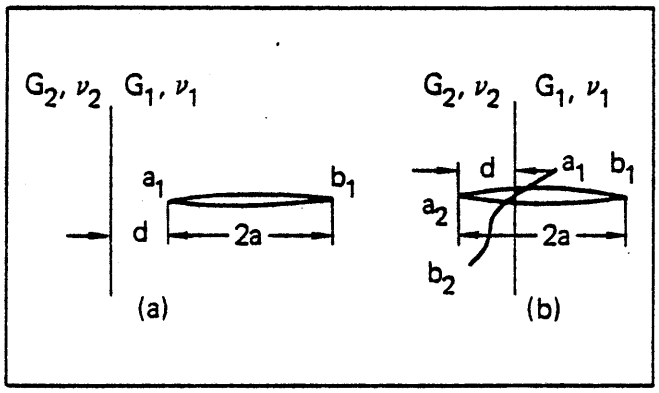


FIG. 2.1. (a) Single crack model for a crack approaching the interface; (b) two-crack model for a crack crossing the interface.

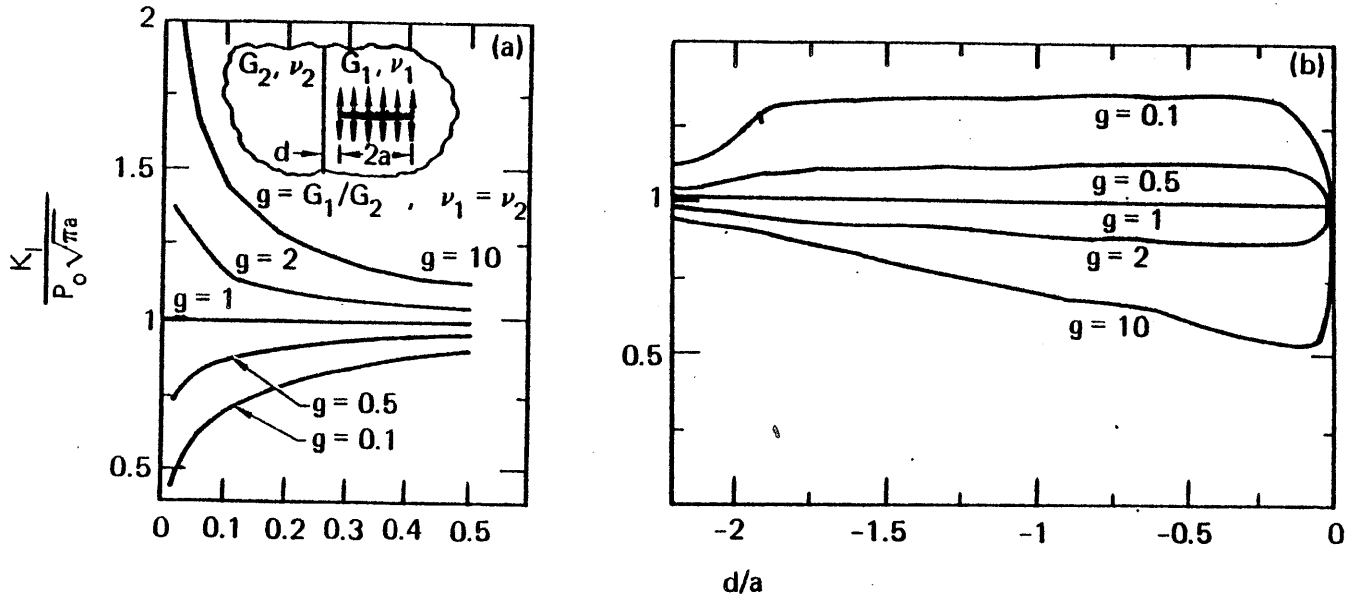


FIG. 2.a. (a) Plot of stress-intensity factor vs distance from interface for a crack approaching the interface; (b) plot of stress-intensity factor vs distance between crack tip and interface for a crack crossing the interface.

2.3 Effects of a Micro-Crack on Containment

It was noted in the last section that an hydraulic fracture, propagating toward an interface with a stiffer material, will at some point encounter an "elasticity barrier" which will, in the absence of moderating microstructural conditions, drive K_I at the near-interface tip to zero. Among the strongest of these counteracting conditions is the presence of a micro-crack a short distance across the interface from the main fracture (Figure (2.3)). Under these circumstances, the near tip tensile stress field of the main fracture could potentially induce a large enough K_I on the tip of the micro-crack to cause it to propagate back across the interface and link up with the main fracture.

Our approach to this problem was to determine the frac-fluid pressure (p_o) on the main crack required to produce a positive K_I at the near-interface tip of the micro crack if both cracks are in a region of compressive tectonic stress of magnitude σ_M . This solution was obtained by first solving the micro-crack problem with a unit positive normal load on the hydraulic fracture and no load on the micro-crack so as to obtain the stress intensity factor at the tip of the unloaded micro-crack, K_{Iu}. The problem was then solved for the converse crack loading to obtain the stress intensity factor at the tip of the loaded microcrack, K_I . By superposition we can write the expression for K_I at the micro-crack tip for the case where the hydraulic fracture is subjected to frac.

fluid pressure P_0 and confining stress $-\sigma_M$ and the micro-crack to $-\sigma_M$ alone

$$K_I = (P_0 - \sigma_M) K_{Iu} - \sigma_M K_{Il} \tag{2.2}$$

from which we deduce that for K_I to be positive, the ratio of frac. fluid pressure to confining stress must exceed

$$\left(\frac{P_0}{\sigma_M}\right)_c = \frac{K_{Il}}{K_{Iu}} + 1 \tag{2.3}$$

The effects of geometric and material parameters on $(P_0/\sigma_M)_c$ are shown in Figures 2.4 and 2.5. Of special interest is the fact that the capability to actually open the micro-crack is not strongly affected by the micro-crack's size. It is also apparent that the proximity of the hydraulic fracture to the interface is more important than that of the micro-crack. The ratio of shear moduli for the strata is also an important factor.

Figure 2.4 shows that it is possible to produce a positive K_I at the tip of a micro-crack without having a frac. fluid pressure excessively above the confining stress. For example, a frac-fluid pressure of $1.4\sigma_M$ in a 30 foot hydraulic fracture 1.5 feet from an interface (with a shear modulus ratio of 2) can produce a positive K_I at the tip of a 3.5 inch micro-crack 3.5 inches from the interface. Figure 4 shows that if the same hydraulic fracture were instead 4 inches from the interface (still not strongly

under the influence of the elasticity barrier), the same frac. fluid pressure would produce a positive stress intensity factor at the top of a 3.5 inch microcrack as far as 3 feet beyond the interface. Statistically,* this provides a higher probability of finding enough micro-cracks and damage to back-propagate ahead of the major fracture.

We conclude that micro-cracks are significant factors influencing the containment of hydraulic fractures in shallow, soft strata. It is in these situations -- where the lateral confining stresses are small compared to the frac. fluid pressures required for hydraulic fracture propagation -- that micro-cracks in a stiff adjacent stratum can be easily induced to break through the interface and link up with the hydraulic fracture, thus allowing it to overcome the elasticity barrier presented by the stiff stratum and thereby break out of the pay zone. At greater depths, we expect the hydraulic fracture to be more readily contained in the pay zone by the elasticity barrier because the frac. fluid pressure required for propagation is then such that $(P_0-\sigma_M)/\sigma_M$ can be too small for the mechanism above to operate.

---

*Note that our conclusions here need very little modification in discussing the fully 3-D character of the real field operation.

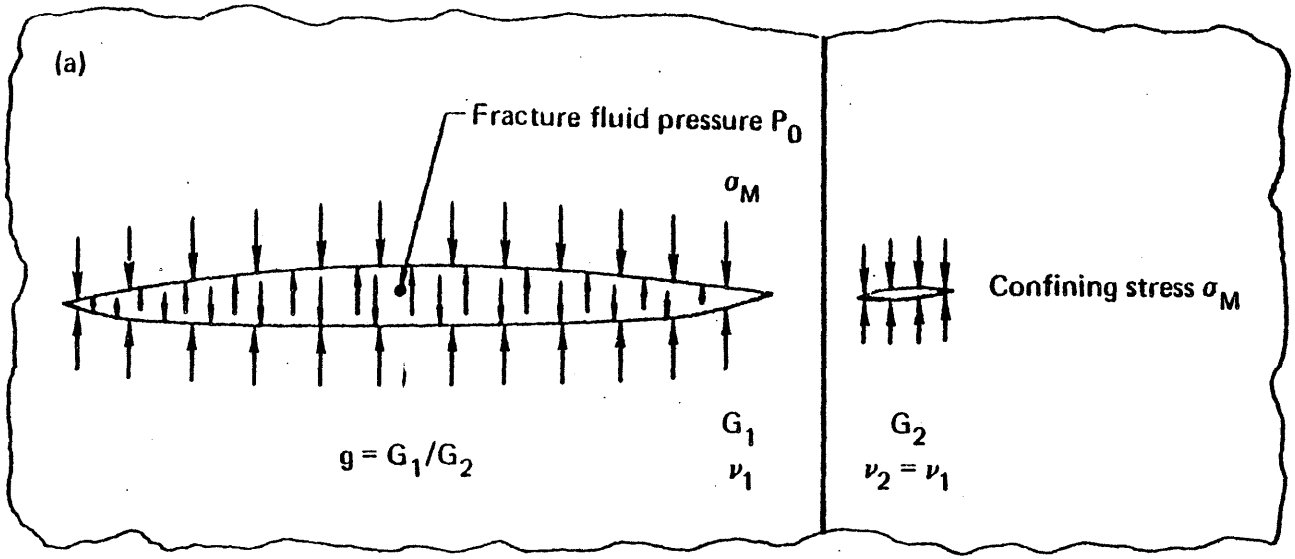


Fig. 2.3 Diagram of the microcracks problem. We must determine the fracture fluid pressure required to cause a positive stress-intensity factor at the near-interface tip of the microcracks for given microcracks and hydraulic fracture lengths, distances from the interface and relative shear moduli

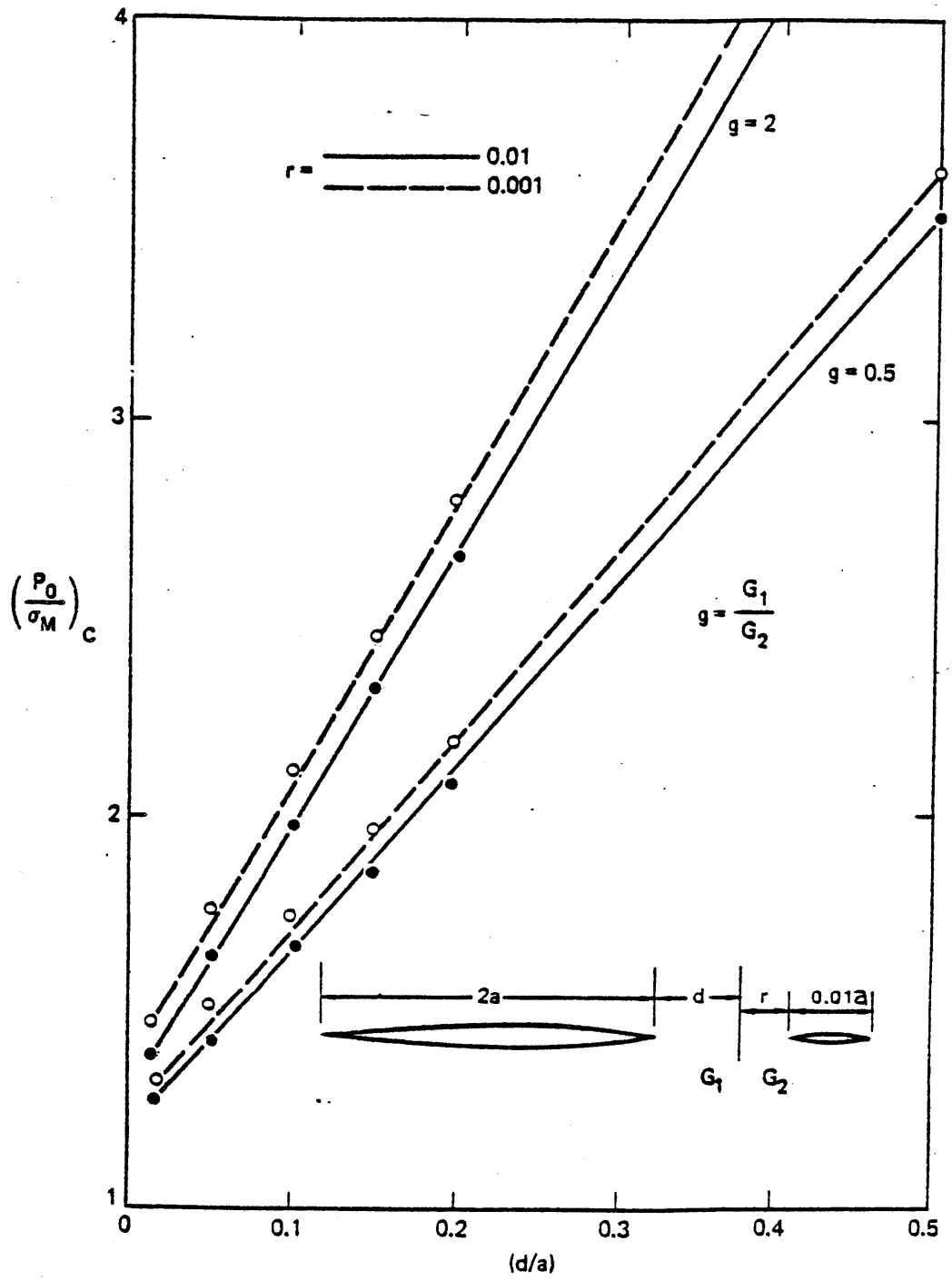


Fig. 2.4 Plot of $(P_0/\sigma_M)_c$ as a function of the distance from the tip of the hydraulic fracture to the interface for the microcrack problem.

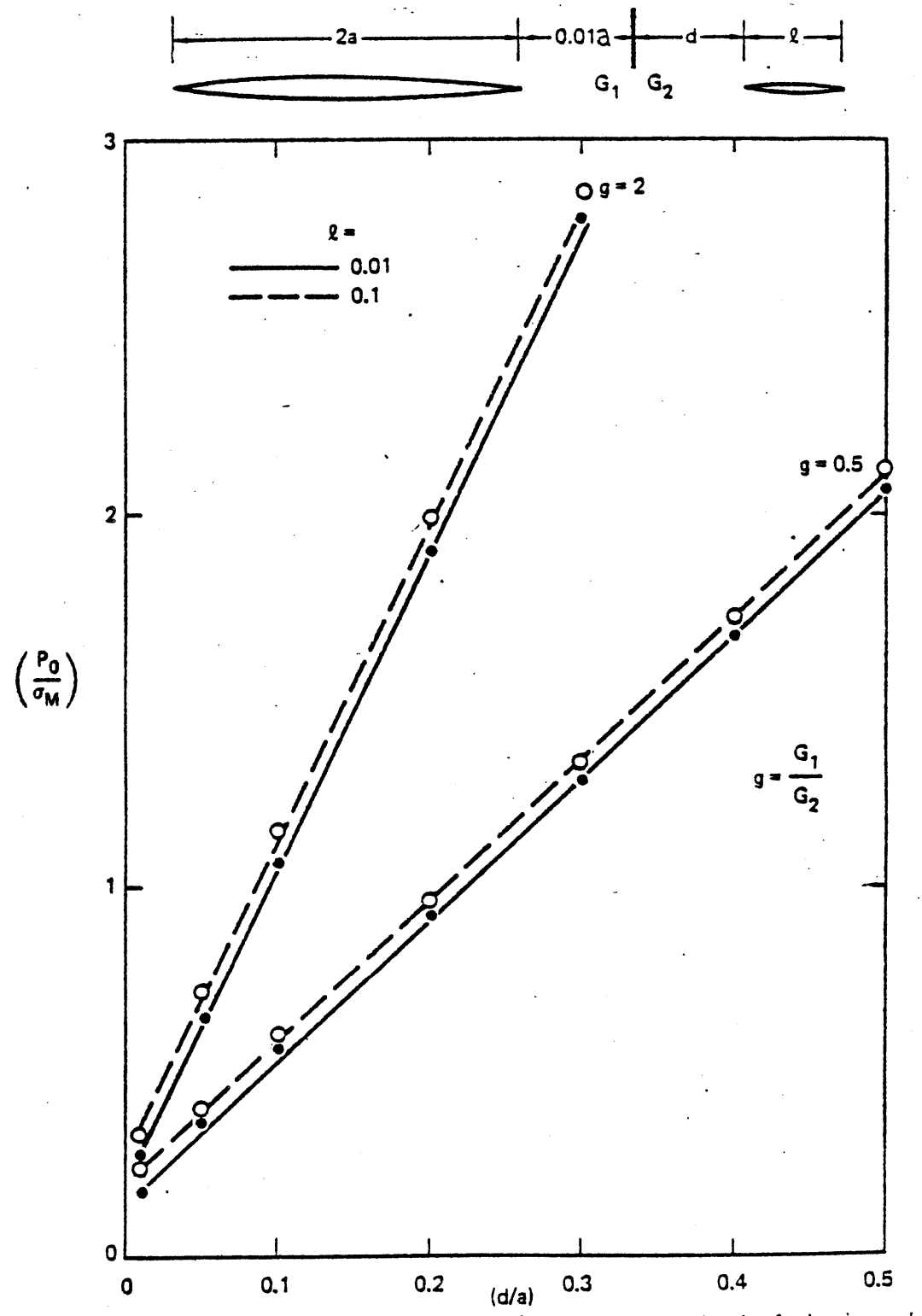


Fig. 2.5 Plot of $(P_0/\sigma_M)_c$ as a function of distance from the tip of the microcrack to the interface for the microcrack problem.

2.4 Behavior of Stress Intensity Factors at the Tips of Singly Branched Cracks

Under certain conditions we might expect a propagating hydraulic fracture to branch (i.e., to change course) as it encounters unsymmetric stress fields, changes in material composition or structural defects. Branching would be expected, for instance, in an hydraulic fracture obliquely approaching an inclusion or interface. The type of event that occurs may range from formation of a single branch (the subject of this section) to generation of multiple branches (of which more than two usually are observed only under dynamic propagation conditions). The results of an investigation concerning the appropriate model for crack blunting -- an interesting and very important example of multiple branching -- will be presented in the next section, along with the results of a study of some simple blunting problems.

We model the singly branched crack as two separate, intersecting crack surfaces. In this respect the problem is similar to that of a crack crossing an interface. Now, however, the two cracks are not collinear, and additional complications are thus introduced: specifically, since we must now solve for dislocation densities in two directions on two surfaces, we require not two but four extra equations to complete the resulting system. Two of these, naturally, are simply the closure conditions (Eq. (1.12)), namely, that there be no net opening or sliding dislocation over the entire branched crack. The question of what matching conditions

are appropriate for the branched crack problem is not so easily answered. While equations do exist in the literature [18], neither their physical motivation nor the extent of their applicability is clear, and our attempts at resolving these issues have not yet produced conclusive answers. However, we developed the notion that, in the immediate vicinity of the intersection, the opening and sliding dislocation densities may be adequately represented by an assumption of antisymmetry, which is certainly valid in one particular case of two cracks with identical loading and length. The adequacy of this assumption was verified by comparison with the results of Gupta [18] and Lo [19] (see Table 1), and is further vindicated by our observation that any reasonable relationship between the dislocation densities at the intersection produces equally satisfactory results, at least whenever crack lengths are of comparable order. Recently, however, Barr [22] has found that the agreement with Lo's results deteriorates somewhat for very short branches when using this specification of antisymmetry as a matching condition. He has obtained good agreement with Lo for a very wide range of branch lengths ($2 \leq a/d \leq 200$) by requiring the much stronger condition that all dislocation densities to vanish at the intersection, thereby excluding stress singularities at that point. He has implemented this requirement in two ways, with equally good results: by explicitly requiring $\bar{\mu}^{\beta 1}$ and $\mu^{\beta 2}$ to vanish (which necessitates removing the integral equations at one of the points x_r near the intersection), or by requiring the dislocation

densities on only one of the cracks to vanish. While the latter method may appear to be insufficient, it seems to result in essentially vanishing μ on both surfaces and offers the advantage of allowing the governing integral equation to be written at all of the x_r's.

It seems likely thatsimilar requirements will prove to be more acceptable than the ones currently used for other intersecting crack problems such as a crack penetrating an interface (Section 2.2) and the blunted crack problem (Section 2.5). We are currently evaluating its performance in such problems.

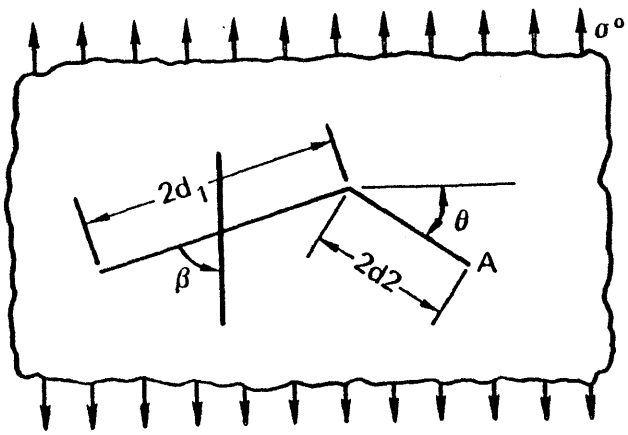
Along with the results of Gupta and Lo cited above, we interpret as further validation of our branched crack model the results shown in Figure 2.6, where K_I and K_{II} are plotted as functions of branching angle θ for a symmetric branched crack (viz. a crack whose legs are of equal length). We attribute the decrease of K_I with increasing θ to the decreasing portion of the total crack length subject to loading in one of the two normal directions. Likewise, the increase in K_{II} is related to the increasing shear component on S_i of the frac. fluid pressure acting on S_j. If effective length were the only factor affecting K_I, we would expect the decrease to be very roughly described by

$$K_I = \sigma_0 \sqrt{\pi a (1+\cos(\theta))} \qquad (2.4)$$

Equation (2.4) is plotted as a dashed line in Figure 2.6. For small values of θ , the agreement between the computed K_I and that predicted by Equation (2.4) is quite good, but at greater angles we see that K_I does not drop as far as we would expect. It is likely that with increasing θ , the decrease of K_I is moderated by the tendency of one surface to partly influence the other, as if it were a free surface.

The behavior of the stress intensity factors with increasing extent of branching is shown in Figures 2.7-2.10. Figures 2.7 and 2.9 show the expected increase of K_I with branch length, and K_I curves for various branch angles are compared with the well-known elevation of K_I at the tips of a propagating straight crack (dashed line).

The behavior of K_I with increasing branch angle is as expected based on a crude "effective length" argument mentioned above. The behavior of K_{II} at the tip of the branch (Figure 2.10) is reasonable, since we would expect the contribution to the effective shear loading of the branch from the opening of the main crack to be greatest when the branch is very small; we would, therefore, expect an initial increase in K_{II}, followed by a fall-off when the branch length (and thus the normal load due to frac fluid pressure) becomes significant. K_{II} at the tip of the main crack predictably increases from zero as the length of the branch surpasses that of the main crack and their roles reverse.



Comparison of results obtained by antisymmetric matching to results of K. K. Lo.

Results from antisymmetric matching[a]

θ, deg	$K_{IA}/\sigma^\circ\sqrt{\pi d_2}$	$K_{IIA}/\sigma^\circ\sqrt{\pi d_2}$
15	1.1518	0.3214
45	0.6630	0.7238
75	0.6549	0.6489

Lo's results

θ, deg	$K_{IA}/\sigma^\circ\sqrt{\pi d_2}$	$K_{IIB}/\sigma^\circ\sqrt{\pi d_2}$
15	1.15	0.32
45	0.65	0.72
75	∼0.70	∼0.63

[a] $d_2/d_1 = 0.5$, $\beta = 90$ deg.

Comparison of results obtained with antisymmetric matching to those reported by Gupta.

Gupta's results for $\beta = 90$ deg, $d_2 = d_1$

θ, deg	$K_{IA}/\sigma^\circ\sqrt{\pi d_2}$	$K_{IIA}/\sigma^\circ\sqrt{\pi d_2}$
30	1.0873	0.6833
45	0.7463	0.8405
60	0.3900	0.8319

Results by antisymmetric matching for $\beta = 90$ deg, $d_2 = d_1$

θ, deg	$K_{IA}/\sigma^\circ\sqrt{\pi d_2}$	$K_{IIA}/\sigma^\circ\sqrt{\pi d_2}$
30	1.0852	0.6818
45	0.7450	0.8388
60	0.3880	0.8302

Comparison of antisymmetric matching with Gupta's results for $\beta = 60°$, $\theta = 40°$

d_2/d_1	Gupta's results $K_{IA}/\sigma^\circ\sqrt{\pi d_2}$	Antisymmetric matching, $K_{IA}/\sigma^\circ\sqrt{\pi d_2}$
0.1	0.9942	0.9948
0.05	0.9950	0.9889
0.025	0.9983	0.9902

TABLE I

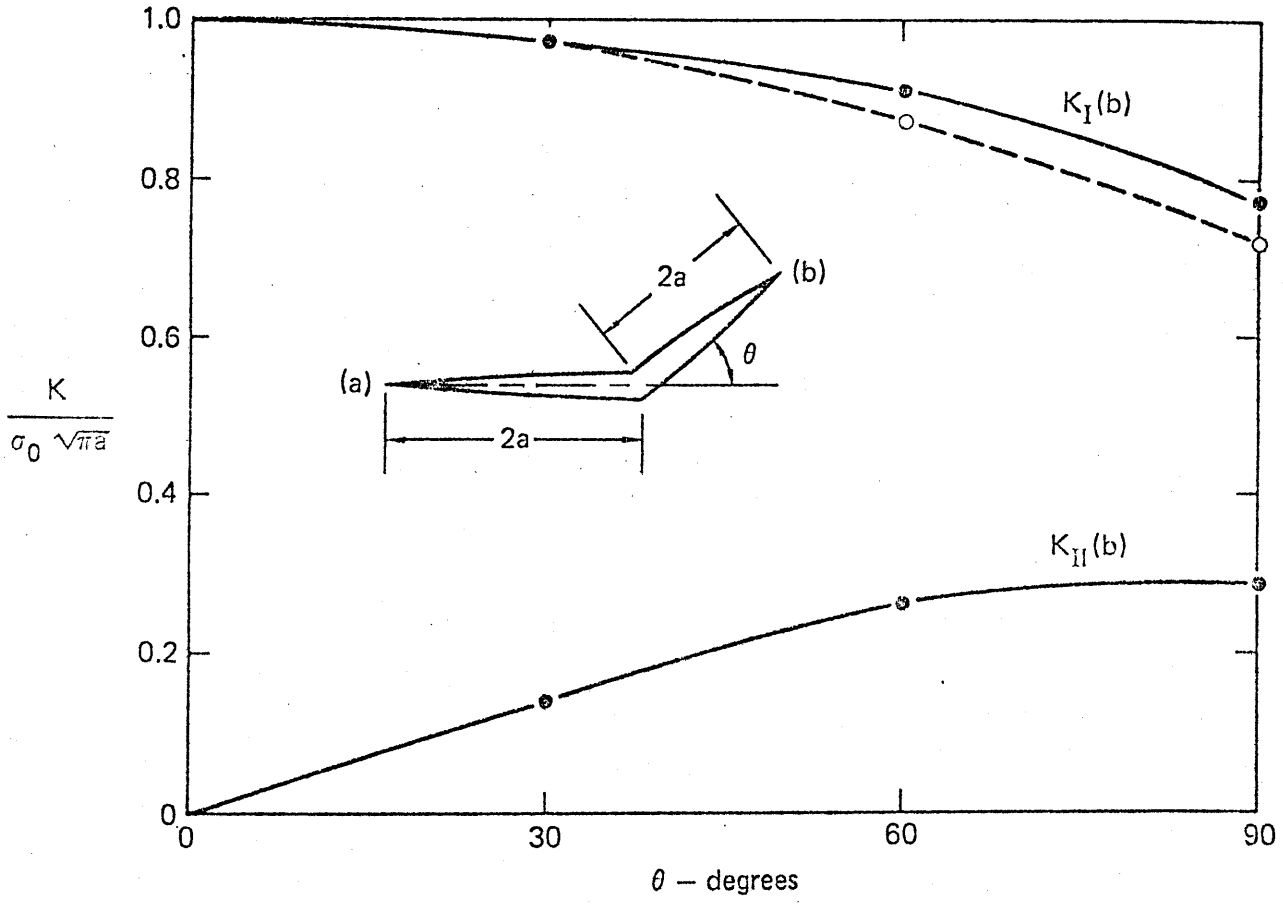


Fig. 2.6 Plot of stress-intensity factors as function of branching angle for the branched crack problem. The dashed line is a plot of Eq. 2.4 -- a rough preliminary estimate of the expected behavior of $K_I(b)$.

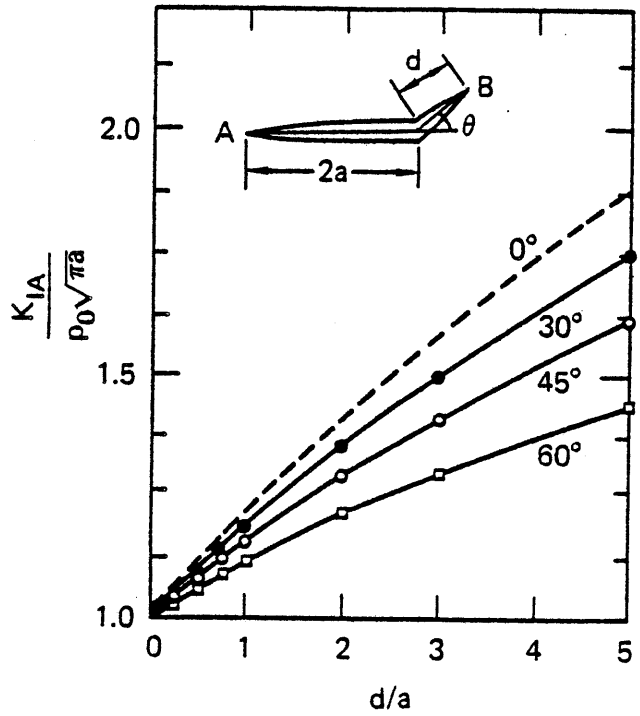


FIG. 7 Plot of K_I at tip A of an unsymmetrically branched crack. The dashed line represents the theoretical elevation of K_I when θ = 0 deg.

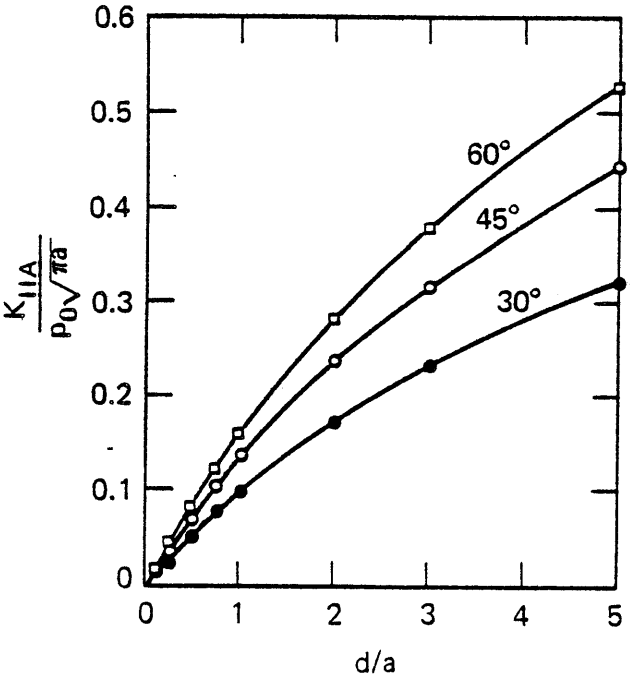


FIG. 8 Plot of K_{II} at tip A vs d/a for an asymmetrically branched crack.

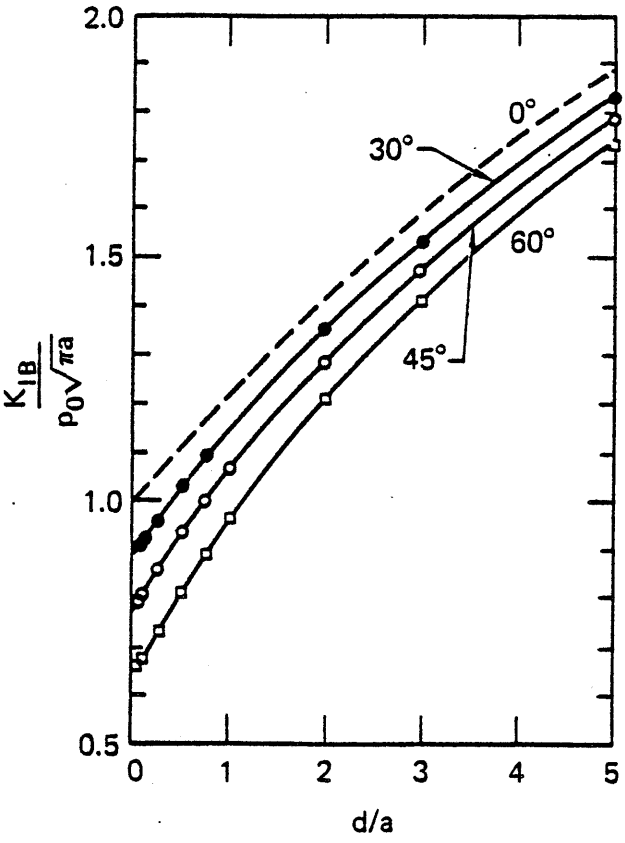


FIG. 9 Plot of K_I at tip B vs d/a for an asymmetrically branched crack.

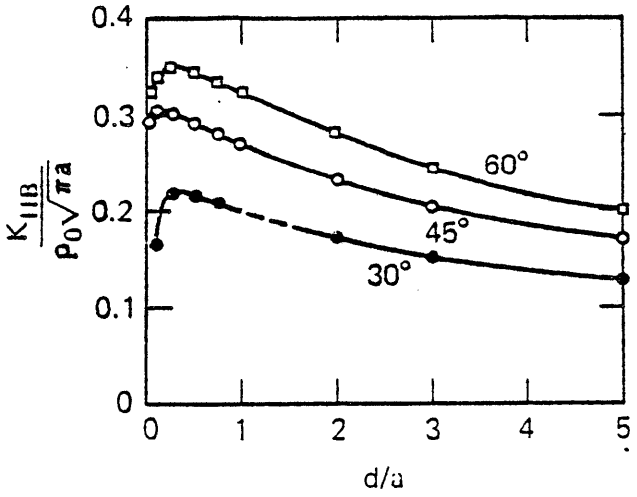


FIG. 10 Plot of K_{II} at tip B vs d/a for an asymmetrically branched crack. The dashed segment of the 30-deg curve is a region of unsatisfactory numerical behavior.

The relatively wide separation of K_I at the tip of a short branch for different branch angles is of great interest to us because of its implications for estimating the directional tendency of a hydraulic fracture. Apparently, based on any of the numerous branching criteria (e.g. [17]), we would not expect a straight hydraulic fracture in a homogeneous medium to deviate from its course if the tectonic stress field is consistent and it is driven by internal pressure: however, we have previously recognized [14] the various barriers and stress eccentricities that can easily make this branching more favorable.

2.5 The Behavior of Stress Intensity Factors at the Tips of Doubly Branched or Blunted Cracks

There are situations in which we might expect a propagating hydraulic fracture to form not one, but two branches. Perhaps the most likely (and the most important from the standpoint of containment) of these occurrences is crack blunting. This is a process by which the energy normally available to drive a crack across an interface would instead cause separation and frictional slippage on such an imperfectly bonded interface. Because of its importance in hydraulic fracturing, our investigations of doubly branched cracks focussed on crack geometries associated with such a blunting process. A complete study of blunting must include the frictional characteristics of the interface, as well as the tectonic stresses acting at the interface, since it is these

properties which may control the degree of blunting rather than the elastic moduli of the material on either side of the interface (Section 2.2); such a study has been undertaken by Papodopoulos [20].

Two different blunted crack models were evaluated. The simpler of the two is a two crack model in which the main crack and the blunted portion are two separate intersecting surfaces (Figure 2.11a). The second model, which yielded better numerical results, is the three crack model shown in Figure (2.11b), in which the blunt is imagined to be composed of two surfaces intersecting tip-to-tip at the point where the blunt joins the main crack, which is the third element.

Once again, additional equations are needed to complete the system formed by the governing integral equations. In the case of the two-crack model, we need four such conditions. The most important consideration is that there should not be any (even logarithmic) stress singularity in the material near the intersection, which is equivalent to requiring that there be no net jumps in dislocation density at the intersection. For our initial work with the two crack model for symmetric blunted cracks, we imposed this constraint through the following equations:

$$-\mu^{(11)}(0) + \mu^{(22)}(0+) - \mu^{(22)}(0-) = 0 \qquad (2.5a)$$

$$\mu^{(21)}(0) + \mu^{(12)}(0+) - \mu^{(12)}(0-) = 0 \qquad (2.5b)$$

The remaining two equations came from requiring closure (Eq. (1.12b)), as before. Equations (2.5) are unsatisfactory for use with unsymmetric problems since, although they ensure boundedness of σ_{11} and σ_{12}, and σ_{22} is not bounded unless the blunt is perpendicular to the main crack. We thus decided that a different way of requiring bounded stresses near the intersection was in order. Because our Chebyshev formulation is not well suited to providing discontinuous dislocation densities on a single crack, we concluded that better numerical stability and perhaps physical realism could be achieved by specifying that the opening and sliding dislocation densities on the main crack vanish at the intersection, while densities are the same on either side of the intersection on the blunted portion of crack surface; namely

$$\mu^{(N2)}(0-) = \mu^{(N3)}(0+), \quad \mu^{(T2)}(0-) = \mu^{(T3)}(0+) \; ; \; \mu^{(X1)}(0) = 0, \; \mu^{(Y1)}(0) = 0 \qquad (2.6)$$

In view of the recent findings regarding matching conditions for branched crack problems (Section 2.4), it is probably best to require $\mu^{\beta j} = 0$ at the intersection on at least two of the crack surfaces. However, in the work presented here, Equations (2.6) proved to be satisfactory and, along with two closure conditions, were used in the three crack model, where six additional equations were needed to complete the system.

The results of our investigation of the behavior of

symmetric blunted cracks are shown in Figure 2.12. The essential features are the behavior of the opening mode stress intensity factors at the tips of both the primary and secondary cracks; namely $K_I(a)$ and $K_I(b)$, respectively. We note that the elevation of $K_I(a)$ with increased blunting reverses as expected when the length of the secondary crack exceeds that of the primary crack, but that, with increasing secondary crack length, $K_I(b)$ rises much more strongly than we had anticipated.

The initial rise of $K_I(a)$ is probably due to the development of a free surface effect like that encountered in the branched crack problem: the secondary crack offers much less resistance to the opening of the primary crack than would the unbroken material. When the secondary crack exceeds the primary in length, the effect of the fluid pressure in the secondary crack overwhelms the free surface effect by producing a compressive stress on the prospective locus of the primary crack thus decreasing $K_I(a)$, and thus dominates its further behavior.

In the absence of the primary crack $K_I(b)$ would increase as $\sqrt{d/\ell}$. We find this to be the case for large d/ℓ. The relative behavior of $K_I(a)$ and $K_I(b)$ is substantial evidence that once the secondary crack becomes long enough, propagation of the primary crack away from the secondary crack will be virtually stopped. Thus, we may make the preliminary conclusion that while blunting may result in containment of an hydraulic fracture, it

may also inhibit propagation away from the interface.

Results for representative asymmetric blunted crack problems are shown in Figures 2.13 - 2.16. Here we examine the effects of blunting inclined at an angle θ to the axis of the main crack when one tip (tip B) is held stationary and the other (tip C) is advanced. These results were obtained from both the two and the three crack models, as noted on the plots. While the two-crack model offers the advantage of simplicity, we note that there are cases of numerical instability for certain combinations of blunt length and inclination. This situation was remedied by adopting the three crack model, with its greater facility for capturing the behavior of the dislocation densities at the intersection. We feel that the three-crack model is much more accurate and reliable than the two-crack model, and we plan to use it in our future work. The stress intensity factors at the three-crack tips display some mildly noteworthy behavior.

As usual the behavior that interests us most is that of K at the various crack tips. Regardless of the angle of inclination of the blunt there is an increase in K_{IA}, K_{IB}, and K_{IC} with increasing amounts of blunting. Both the rapidity of this increase and the initial magnitude of these stress intensity factors depend upon the angle θ, but the nature of the dependence is different for K_{IB} than for K_{IA} and K_{IC}: for any choice of ℓ, K_{IB} increases with increasing θ, but K_{IA} and K_{IC} decrease (albeit slightly).

It is most probable that K_{IB} is dominated by compressive stresses in the vicinity of the body of the main crack: as θ decreases, tip B moves into areas of larger compressive stresses which force K_{IB} to decrease with θ for a given frac fluid pressure. The behavior of K_I and K_{II} at tips A and C is quite similar to what we have seen in the branched crack results in the previous section (as might be expected from the shortness of the leg of the blunt between the intersection and tip B). The magnitudes of K_{IIA}, K_{IIB}, and K_{IIC} tend to level off and decline as the blunt becomes very large compared to the main cracks, thereby confirming some obvious intuitive predictions. A phenomenon which is best illustrated by Figure 2.16a is the reversal in sign of K_{IIA} which occurs when the relative shearing actions of the legs of the blunt reverse; in case of the 90° blunt this occurs when one leg surpasses the other in length. We quote all these observations in order to provide some confidence in the general correctness of the scheme, although there are many other more complicated phenomena of interest still to be pursued.

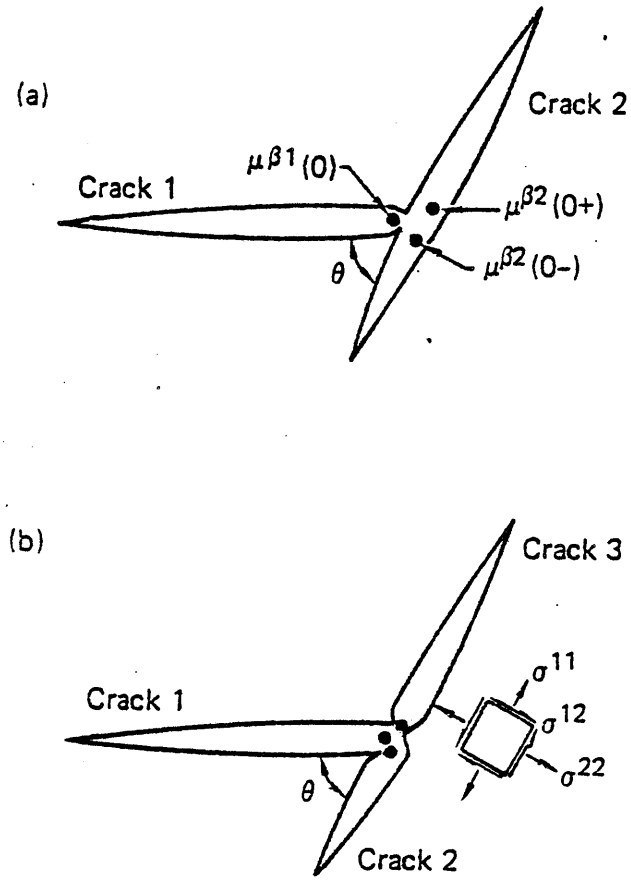


FIG. a.11 (a) Two-crack model for the blunted crack problem; (b) three-crack model, preferred because of its ability to capture the behavior of $\mu^{\beta 2}$ near the intersection.

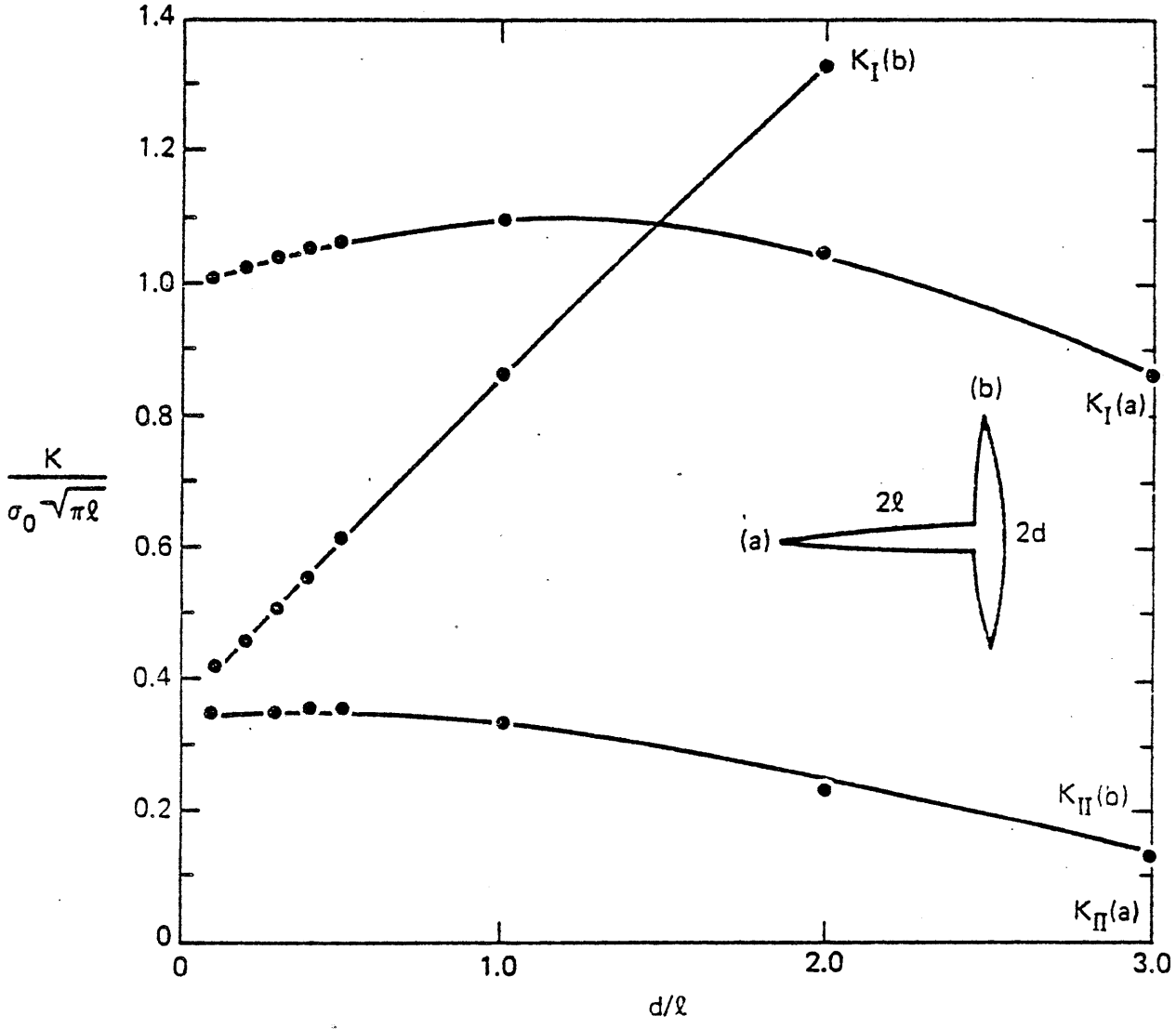


Fig. 2.12 Plot of stress intensity factors vs size of secondary crack for the blunted crack problem.

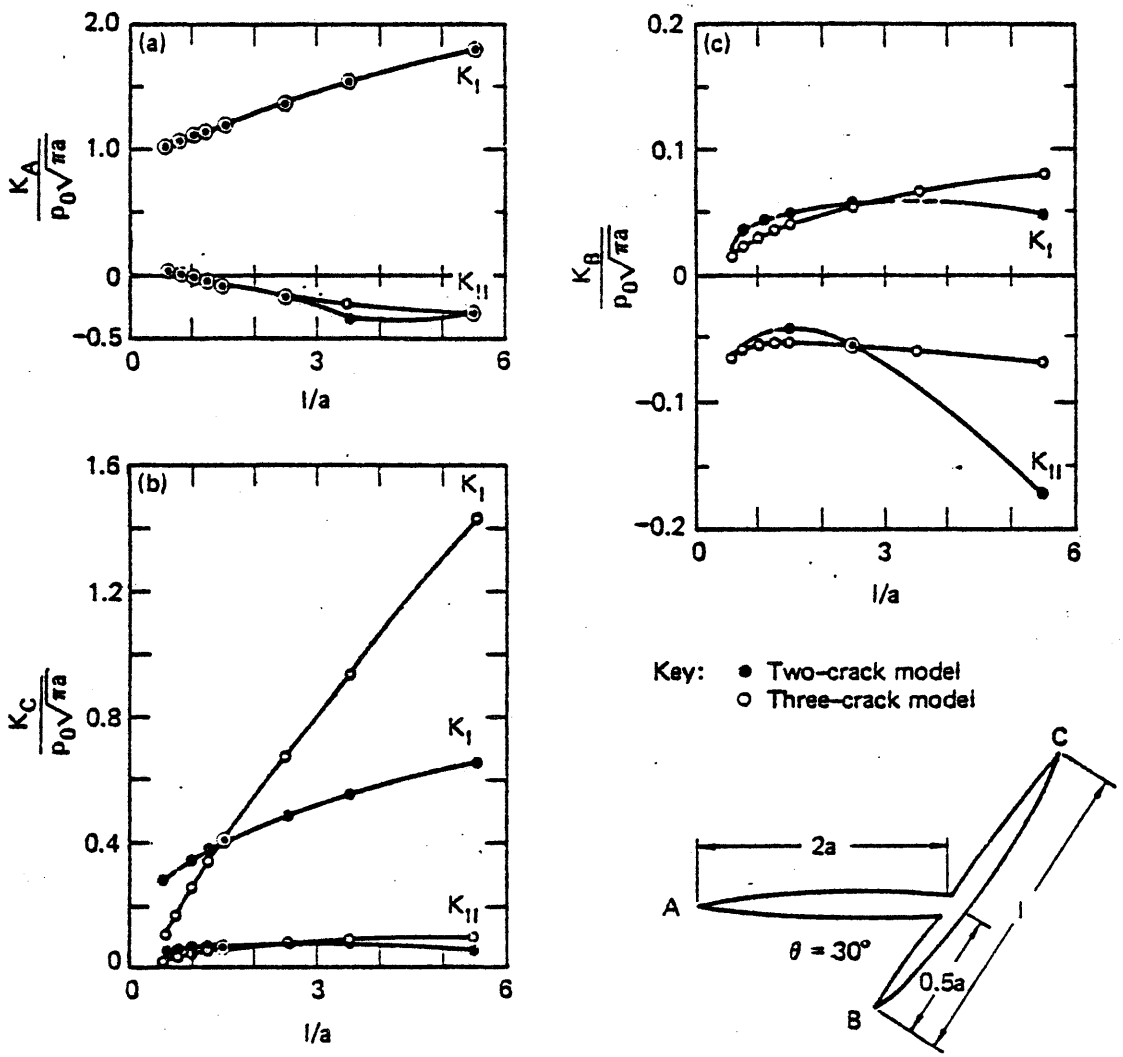


FIG. 2-13 Stress intensity factors at the tips of a 30-deg asymmetric blunted crack. (a) K_I, K_{II} at tip A; (b) K_I, K_{II} at tip B; (c) K_I, K_{II} at tip C. For the two-crack model, we modeled both cracks with 20 nodes. For the three-crack model, crack 1 had 20 nodes and cracks 2 and 3 had 10 nodes each (negligible changes in the results occurred when 15 nodes were used on each surface). The dashed segments indicate regions of unsatisfactory numerical behavior.

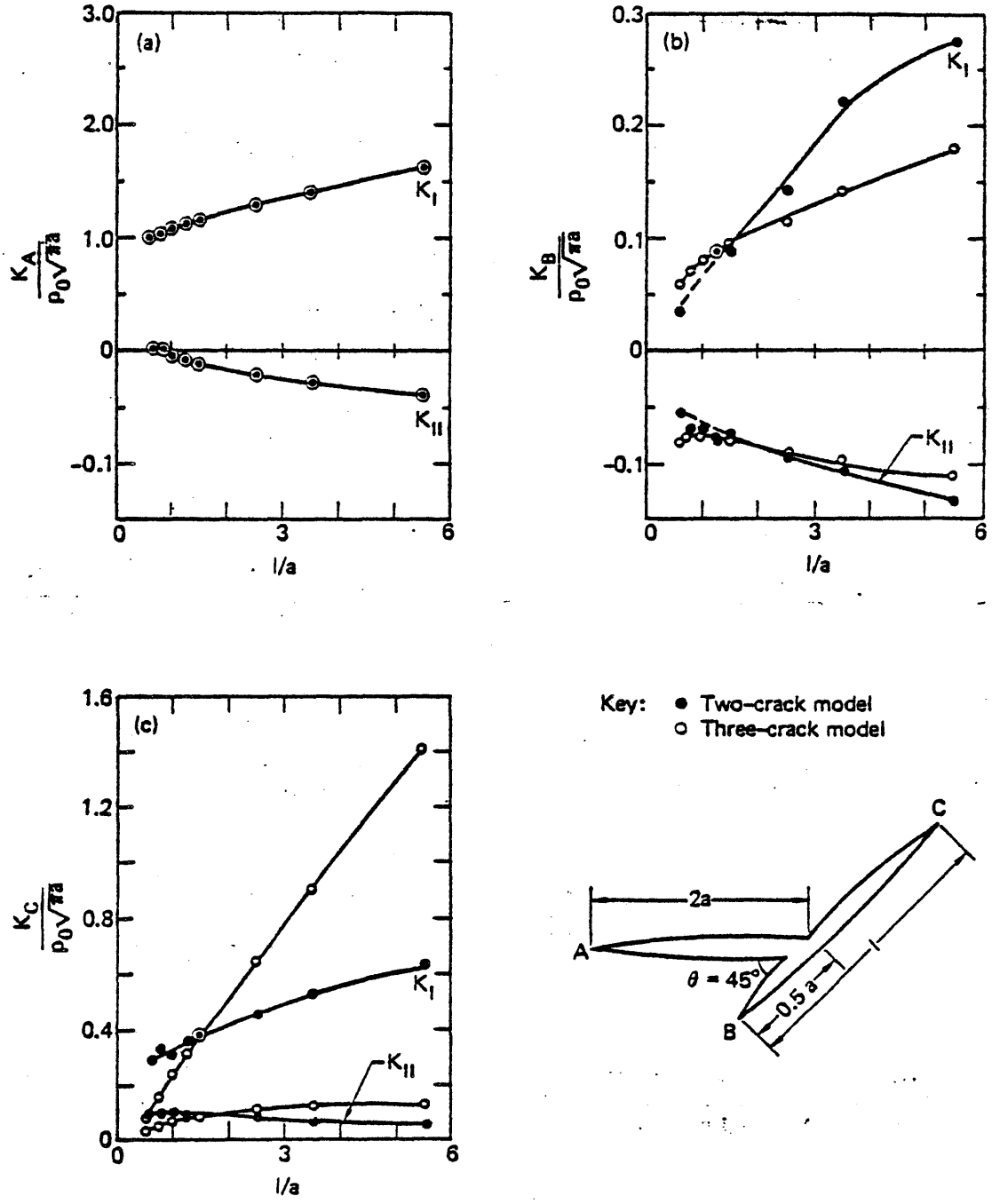


FIG. 2.14 Stress intensity factors at the tips of a 45-deg asymmetricallly blunted crack. (a) K_I, K_{II} at tip A; (b) K_I, K_{II} at tip B; (c) K_I, K_{II} at tip C. For the two-crack model, cracks 1 and 2 had 20 nodes. For the three-crack model crack 1 had 20 nodes, and cracks 2 and 3 had 10 nodes each. The dashed segments indicate regions of unsatisfactory numerical behavior.

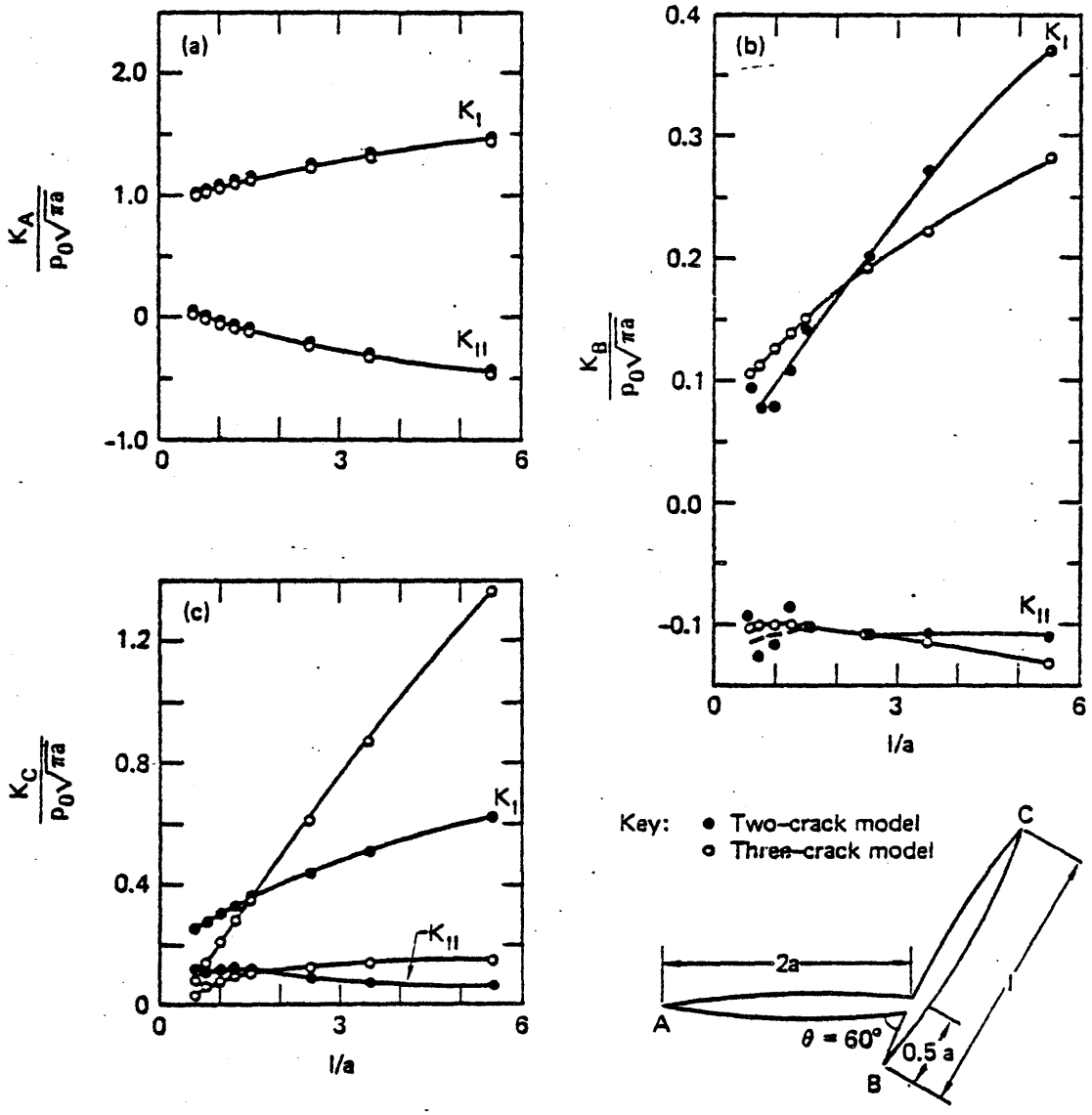


FIG. 2.15 Stress intensity factors at the tips of a 60-deg asymmetrically blunted crack. (a) K_I, K_{II} at tip A; (b) K_I, K_{II} at tip B; (c) K_I, K_{II} at tip C. For the two-crack model, cracks 1 and 2 each had 20 nodes. For the three-crack model, crack 1 had 20 nodes, while cracks 2 and 3 had 10 nodes each.

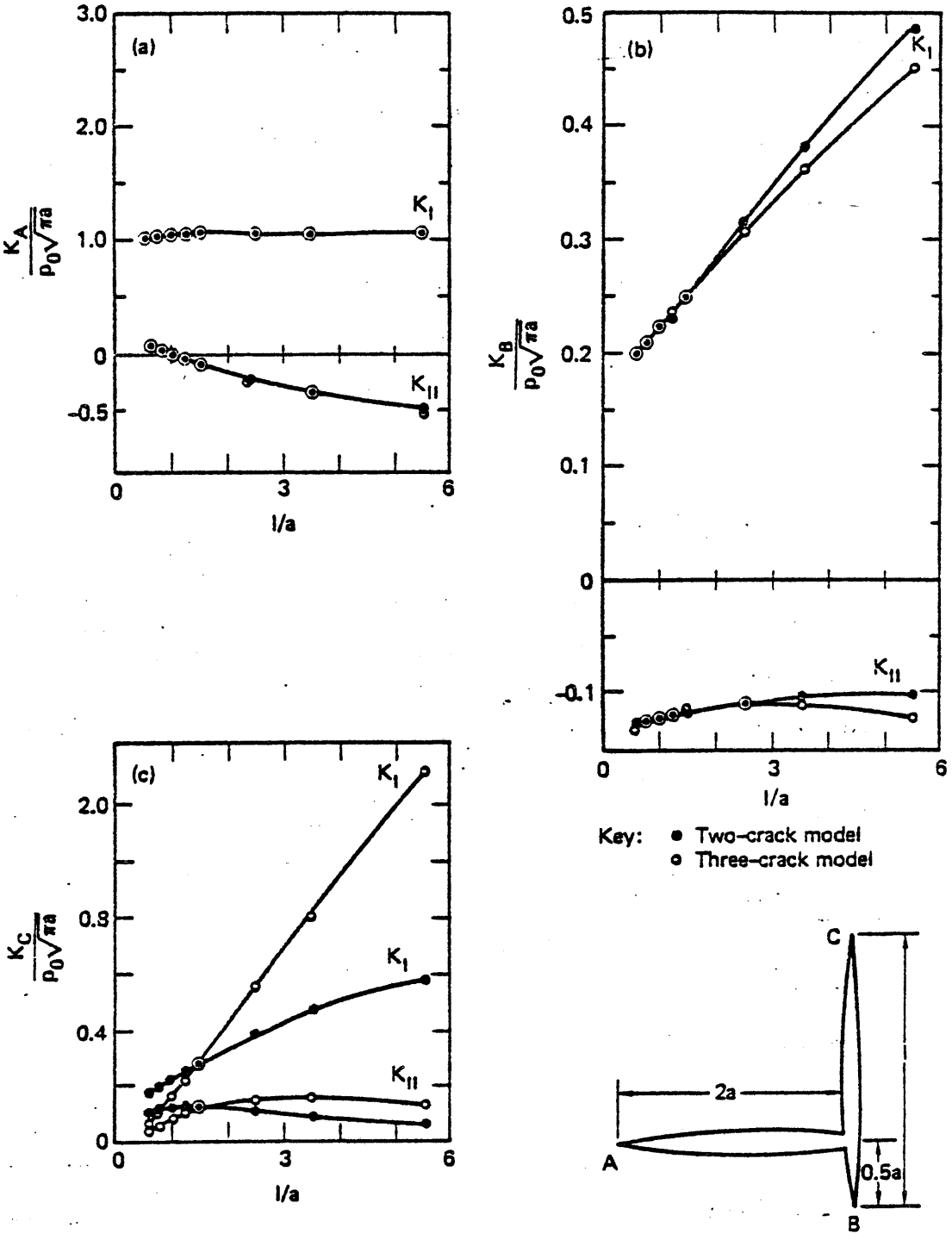


FIG. 16 Stress intensity factors at the tips of a 90-deg asymmetrically blunted crack. (a) K_I, K_{II} at tip A; (b) K_I, K_{II} at tip B; (c) K_I, K_{II} at tip C. For the two-crack model, both cracks had 20 nodes each. For the three-crack model, crack 1 had 20 nodes, and cracks 2 and 3 had 10 nodes each.

CHAPTER 3 QUASI-STATIC CRACK PROBLEMS

3.1 Introduction

Our studies of the static crack problems described in Chapters 1 and 2 have served two important purposes: they have provided some insight into the behavior of corresponding cracks in actual hydraulic fracturing operations, and they have served as stepping stones, providing us with modelling experience necessary to achieve our ultimate goal of full 3-D simulation of propagating hydraulic fractures. In order to reach that goal, we must have, in addition to the capability of modelling complex crack geometries, the capability of computing the characteristics of the flow of a viscous fracturing fluid in a propagating crack, as well as the effect of the fluid flow on the rate of propagation. Our approach to such quasi-static hydraulic fracturing problems has been to consider in sequence certain idealized models with increasing complexity. Thus, we first investigated the problem of fluid pressure evolution in a stationary plane crack filled with a quasi-statically flowing fluid; then we studied pressure evolution and fluid front advancement in such a crack. Work is now in progress on the problem of quasi-static propagation and fluid front motion in a plane crack, a problem which comes quite close to some actual field operations. We found, in the course of working on the pressure evolution problem, that the "explicit" scheme described in the next section seems totally inappropriate and that only the "implicit" formulation described in Section 3.3 is sufficiently stable.

3.2 Frac. Fluid Pressure Evolution: Explicit Formulation

The pressure evolution problem is illustrated schematically in Figure (3.1): the extremely viscous frac. fluid is pumped into a crack (already filled with the frac. fluid) whose length is held fixed. As the width of the crack increases, the fluid pressure distribution changes accordingly. Since we choose to pump the fluid at whatever rate is necessary to maintain a constant presssure at the borehole, the process will stop when the fluid pressure becomes uniform along the entire crack length.

In the early stages of our work on this problem, we felt that an "explicit" formulation following the general outline presented by Cleary [17], would be the simplest and most economical method of solution; since we anticipated having to carry out the solution over many discrete time steps, the latter are very important criteria. By explicit scheme we mean a method which allows the fluid pressure distribution at a time in the future to be calculated explicitly from the present crack opening and fluid pressure distributions. Such a method is considerably more economical than an "implicit scheme", in which the future pressure distribution depends implicitly upon the current state, thus requiring solution of a system of equations. Although some stability problems were anticipated, (as discussed below and in ref. [17]), we felt that they

could be adequately taken care of.

In the development that follows (and in later sections) some simplifications of the notation used in Chapter 1 will be possible, since from here on we will be dealing with one crack only and normal tractions. Specifically, the superscripts used in reference to the traction σ, the influence function Γ, and the dislocation density μ will be dropped; further, the traction will be designated by p as a reminder that it is due only to an internal fluid pressure. In other words; $p \equiv \sigma^{(11)}$, $\Gamma \equiv \Gamma^{(1111)}$ and $\mu \equiv \mu^{(11)}$. Also, since the crack will always be assumed to lie on the interval [-1,1], $E_1 \equiv 1$.

Our formulation starts with the equivalent of the integral equation (1.2):

$$p(x_0) = \int_{-1}^{1} \Gamma(x_0, x)\, \mu(x)\, dx \tag{3.1}$$

The appropriately specialized versions (presented in [17] along with the more general equations) of the equations of conservation of mass and momentum take the form

$$\frac{\partial}{\partial x}(v\delta) = -\frac{\partial \delta}{\partial t} \tag{3.2}$$

and

$$\frac{\partial p}{\partial x} = -\hat{\eta}\,\frac{v}{\delta^2} \tag{3.3}$$

Equations (3.2) and (3.3) are readily manipulated to get

$$\frac{\partial}{\partial x}\left[\delta^3 \frac{\partial p}{\partial x}\right] = \hat{\eta}\,\frac{\partial \delta}{\partial t} \tag{3.4}$$

Differentiating (1) with respect to time and (4) with respect to x and substituting, we arrive at

$$\hat{\eta}\,\frac{\partial p}{\partial t} = \int_{-\ell}^{\ell} \Gamma(x_o, x)\,\frac{\partial^2}{\partial x^2}\left[\delta^3 \frac{\partial p}{\partial x}\right] dx \tag{3.5}$$

The solution procedure involves:

(1) Selecting an appropriate initial frac. fluid pressure distribution

(2) Solving equations (3.1) for $\mu(x) = \delta'(x)$, hence δ

(3) Evaluating the integral in Equation (3.5) and adding the increment in pressure to the previous pressure.

(4) Updating the time, specifying the pressure at the borehole, and returning to step (2).

We have chosen to enforce a constant pressure at the borehole, a condition which is quite realistic; we can easily adapt to the more

usual field condition of constant pumping rate but that is not of fundamental importance yet. In general, however, the newly computed pressure curve at each time step must be corrected in some manner (Fig. 3.2). Perhaps the most appealing method is to simply set the pressure at the node x_0 in the borehole to the desired level. It may be more accurate, however, to apply some form of global renormalization, as shown in Fig. 3.2(a). It is important to scale borehole pressure p_0 and the frac. fluid viscosity η to the shear modulus G of the material (e.g., they might typically have relative magnitudes of $p_0/G = 10^{-4}$ and $\eta/G = 10^{-11}$ sec.). It is essential to relate the time steps assumed in iteration to a time τ_c which is based on the characteristics of fluid flow, relevant considerations of elastic crack opening, and the assumption of constant borehole pressure. The appropriate τ_c has been provided by Cleary [23] and, for the case of linear fluid equations used above, it takes the form

$$\tau_c \sim \left(\frac{G}{p_0}\right)^3 \left(\frac{\eta}{G}\right)$$

A corresponding expression for more general nonlinear fluid behavior has also been extracted by Cleary [24]. Considerable attention has thus been given to finding the appropriate fraction of τ_c for use in our marching schemes. The crucial aspect, from a numerical standpoint is the evaluation of the second derivative appearing in the integral equation (3.5). We start by non-dimensionalizing the variables: $\delta \leftarrow \frac{G\delta}{p_0 \ell}$, $p \leftarrow p/p_0$, $\dot{p} \leftarrow \frac{\tau_c \dot{p}}{p_0}$ and $\gamma \equiv \frac{\Gamma \ell}{G}$.

At first, we thought it best to explicitly expand this derivative:

$$[\delta^3 p']'' = 6\delta^2\delta' p'' + \delta^3 p''' + 6\delta(\delta')^2 p' + 3\delta^2\delta'' p' \qquad (3.6)$$

and then evaluate the component derivatives separately. We seemed to encounter no great difficulties in evaluating δ and its derivatives. The first derivative, μ, is obtained directly from the solution of equation (3.1). In order to calculate δ and δ'', we approximated μ by the polynomial

$$\mu(x) = \frac{a_0 x}{\sqrt{1-x^2}} + a_1 + a_2 x + a_3 x^2 + \dots + a_M x^{M-1} \qquad (3.7)$$

which can then be integrated to get δ and differentiated to get δ''.

Accurate differentiation of p proved to be a much greater problem. We have observed that for any realistic pressure distribution, the resulting dislocation density will be of a form characteristic of that obtained for a uniform distribution. Thus, while we may always be confident that μ can be approximated well by a polynomial of the form (6), we need a more fool-proof method for evaluating the first three derivatives of p. We first tried several different simple scheme for interpolating and differentiating p, all based upon finding an interpolating polynomial of some sort and differentiating it. Specifically, we tried (i) ordinary polynomials (of various orders), (ii) local third order polynomials,

and (iii) Chebyshev polynomials, as described later.

Ordinary Polynomials

Because of the simplicity of implementation our first attempts at finding the derivatives of p involved interpolation with ordinary polynomials. Two approaches were used for obtaining the values of p (initially known at the zeroes of second-order Chebyshev polynomials, ($x_r, r=1,\ldots,N-1$), plus p', p'', p''', at the first order Chebyshev zeroes, t_k, k=1,...,N. The first and simpler of the two was to collocate at the points x_r to obtain

$$p(x_r) = a_0 + a_1 x_r + a_2 x_r^2 + \ldots + a_M x_r^M \qquad (3.8)$$

and then evaluate this polynomial and its derivatives at the t_k. The second approach was to evaluate p at the points t_k first by low order Lagrangian interpolation and then collocate at the t_k to get a polynomial of the form (8). For an initial Gaussian pressure distribution ($p(x) = e^{-5x^2}$), the interpolation was done over the entire interval $-1 \leq x \leq 1$, but for a "square root" curve ($p(x)=\sqrt{1+|x|}$), the interpolation was carried out separately on the intervals $-1 \leq x \leq 0$ and $0 \leq x \leq 1$.

The results obtained by use of these interpolation schemes varied somewhat, but were bad in general. In particular, the approxi-

mation of the derivatives was much too inaccurate for purposes of stability and convergence toward the expected long-time response never was achieved.

Local Third Order Polynomials

Our final attempt to employ a simple, collocation-based polynomial scheme involved the use of ordinary third order polynomials, chosen so as to interpolate p at four consecutive points t_k. It was expected that by using low order polynomials, valid over a relatively short interval, interpolating functions could be found that not only gave reliable values of $p(t_k)$ but in addition were sufficiently smooth to provide good approximations of the derivatives of p. This method was also a relatively simple one: values of $p(t_k)$ were first obtained by low-order interpolations from the values of $p(x_r)$. Local cubic polynomials were then found by collocation at four consecutive t_k, then marching ahead one point and so forth. In other words, $p(t_1)$ was approximated by collocation at $t_1,\ldots,t_4$; $p(t_2)$ by collocation at $t_2,\ldots,t_5$, and so on to t_{n-4}. Values of $p(t_{n-4}),\ldots,p(t_n)$ were all obtained from the n-4th polynomial. This method gave very good approximations to $p(t_k)$ for both the Gaussian and square root pressure distributions, but it still gave highly unsatisfactory values of the derivatives.

While none of these relatively simple schemes provided close enough approximations to the derivatives of p, there were still several very promising alternatives. Since the problem seemed to lie in the lack of smoothness of the various interpolating functions tried so far, we expected that the use of functions of greater intrinsic smoothness could prove to be more fruitful.

The normal difficulties associated with numerical differentiation (especially in evaluating derivatives greater than first order) are made even more severe in the evaluation of $[\delta^3 p']'''$, because of the cumulative nature of δ and p. The future value of p is obtained by integration of $[\delta^3 p]''$ with γ, an operation which does not smooth out ripples in the usual fashion of regular integration: this, at best, can only cause "noise" to be passed along unfiltered to the new p. The future δ is also determined by adding $[\delta^3 p']'dt$; any inaccuracies in the computation $[\delta^3 p']'$ or its derivative will return in the next time step as noise in <u>both</u> δ and p. Thus, while it may be possible to project δ and p by one time step, subsequent computations can be extremely unstable. Clearly, a differentiation scheme which filters out all pre-existing noise in δ and p is required; it is imperative that at <u>each</u> time step, the integrand in Equation (3.5) be perfectly smooth.

The simplest such scheme involves differentiating p, then $\delta^3 p'$ (without prior expansion, as in Eq. (3.6)) by finite

differences and locally smoothing rough areas in each derivatives by fitting with a relatively low order "least squares" polynomial (Figure 3.3). These operations were carried out separately on either side of the borehole location in order to preserve (at the borehole) slope discontinuities in pressure. This scheme was tried using a very simple "triangular" initial pressure distribution.

In addition, during this trial we allowed the borehole pressure to assume whatever value was dictated by the governing equations, rather than correct it at each time step to maintain a specified $p(0,t)$. These simplifications were made because we can analytically predict with some confidence the results for the first time step under such circumstances. These tests were run using the algorithm described above in which we solve Equation (3.1) at each time step.

The first such trial involved computation of a new pressure curve after a very large time step (one quarter of the characteristic time τ_c), to permit easy visualization. The results were generally good, except for slight asymmetry (Fig. 3.4). A significant finding was that, while our differentiation routine was designed to identify rough regions of a derivative and smooth them locally, the derivatives exhibited sufficient roughness (e.g., $[\delta^3 p']'$ in Figure 3.4 (e)) that the smoothing was actually done globally on each side of the borehole. A similar test was run with a more reasonable time increment, but gross instability was observed in the computation by the third

time step. The problem seems to have been that the roughness in $[\delta^3 p']'$ after the first time step was of great enough magnitude that a low order least squares polynomial no longer provides a sufficiently accurate representation of the true curve. The required second differentiation only aggravates this inaccuracy.

Our experience with the tests described above and others like them indicate, perhaps predictably from the viewpoint of skilled numerical analysts, that it is undesirable to smooth derivates by approximation with other functions; the noise present after differentiation is of sufficient magnitude to confound efforts to capture the true form of the derivative. In particular, we expect that a "least squares" fit (because it minimizes the squares of the errors) would be rendered increasingly ineffective by pervasive noise of large-and random-amplitude. We conclude that all measures taken to ensure smoothness of derivatives should at least begin with the function that is to be differentiated.

Among the methods that did show some promise was that of "transferring" $\delta^3 p'$ -- known at 20-40 zeroes (t_k) of the Chebyshev polynomials of the first kind -- to several hundred uniformly spaced points on the same interval, via Lagrangian interpolating polynomials of fourth or fifth order. Since both p and δ are initially quite smooth, the transfer should not introduce any bad behavior. Differentiation can be accomplished with finite differences,

as before, but instead of smoothing the derivative with some sort of global function, we simply compute the average value of the derivative over a number of the uniformly spaced points in the vicinity of a particular t_k^*. This method is simple and does not require much computation time, but the quality of the results can be heavily dependent upon the size of the interval over which the averaging takes place. We found, therefore, that its usefulness for smoothing strongly singular functions such as $[\delta^3 p']$" was variable (although we have equipped our routine with the capability of smoothing over intervals of varying size on the crack surface, thus enabling it to capture anticipated sharp rises and falls in the derivative). Because of the mixed success, and the advent - before testing of this "filter" could be completed - of the scheme described below, this method has been relegated to the role of evaluating p' only.

While we have previously noted the difficulties attendant upon differentiating an interpolating function, this approach seems to be the only one capable of capturing the singular behavior of $[\delta^3 p']$". Some observations regarding the nature of $\delta^3 p'$ and the Chebyshev polynomials (and their derivatives) led us to examine their use: in particular, we noted that the derivative did not have the character desired to represent $\mu_{,t}$ and were thus led to

*Thus we have something akin to a zero order "hold-circuit" low pass filter.

explore expansions in these polynomials.

We consider again the case of a simple triangular pressure distribution which, with the resulting crack opening displacement, → , is sketched in Figure 3.5. Owing to the antisymmetry of $\delta^3 p'$ (Figure 3.5(b)), we may shift both sides (without affecting derivatives) to obtain the continuous curve passing through the origin, shown in Figure 3.5(b). This shifted curve has two properties which immediately and strongly suggest approximation by Chebyshev polynomials: it attains extreme values at ± 1 and it passes through the origin, as do the odd-ordered T_k (first kind). We note, further, that termwise differentiation of a Chebyshev series introduces a division by $\sqrt{1-x^2}$, (see Ref. [27]), which has the right character to represent $\mu_{,t}$. Furthermore, orthogonal functions offer the advantage of being independent: coefficients are chosen on the basis of the integrated degree of presence of the corresponding member of the orthogonal set in the curve being approximated, rather than in an attempt to find a combination of potentially similar functions that may pass through the collocation points. Thus, since our modified anticipated $[\delta^3 p']$ curve has the same general shape as would a combination of two or more Chebyshev polynomials (viz, T_1, T_3, etc.), we might expect a very good approximation to $\delta^3 p'$ and possibly a good approximation of $[\delta^3 p]'$; that is, we epxect both high accuracy and the required degree of smoothness in $[\delta^3 p']'$.

Thus, in implementing this scheme, we represent $\delta^3 p'$ by the series [25]

$$[\delta^3 p'](x) \approx \frac{a_0}{2} + \sum_{k=1}^{N} a_k T_k(x) \tag{3.9a}$$

$$a_k \approx \sum^{N} [\delta^3 p'](x_m) T_k(x_m) \tag{3.9b}$$

$$x_m = \cos(m\pi/N), \quad m = 0, \ldots, N \tag{3.9c}$$

The series (3.9) may be differentiated termwise using either the recursion relations

$$T_N(x) = 2x T_{N-1}(x) - T_{N-2}(x) \; ; \; T_0 = 1 \; , \; T_1 = x \tag{3.10}$$

or the more direct formula given in Ref. [27]. In our general hydrofac formulation, the values of $[\delta^3 p']$ are not known at the points x_m, but can be easily evaluated there by interpolation.

This scheme was tested using δ curves as computed by our fracture simulation program for various numbers of nodal points t_k and differing orders of the series (3.8). Typical results are shown in Figures 3.6 - 3.9. Two separate characteristics of the approximation of $[\delta^3 p']$" may be observed upon examination of these plots. Firstly, while the general shape of $[\delta^3 p']$" is right in all cases, it is plagued by noise, of which "frequency" is dependent

upon the order of the series (increasing with the number of terms, as might be expected); the amplitude seems to decrease with increasing number of t_k points employed to represent $\delta^3 p'$ before the expansion in equation (3.9a). Thus, we would expect the best performance from a series with a very large number of terms starting from an equally large number of t_k's.

Furthermore, it is likely that we could obtain a particularly smooth fit of δ at a large number of points by starting with a relatively small number of t_k's in the actual evaluation of equations (3.6)-(3.8), finding the coefficients from the Chebyshev series approximation of the integral of μ (see Appendix A), then evaluating the series at a larger number of points (preferably the x_m's used to evaluate the a_k). The result should be a curve whose initial high degree of smoothness has been enhanced by the process of integration.

This hypothesis was subjected to a preliminary test by assuming $\delta = \sqrt{1-t_k^2}$ (not much different from the actual shape) and that $p' = \pm 1$; this saved the cost of solving a 200 x 200 system. We then computed $\delta^3 p'$ at 200 t_k points and fitted with a 200 term Chebyshev series to get the results shown in Fig. 3.10. Note that while $[\delta^3 p']''$ still has some high frequency noise (and some bad behavior* at ± 1), $[\delta^3 p']$ is markedly smoother than in any of the

*This is to be expected from the high order of singularity introduced by double differentiation of Chebyshev polynomials (ref. 27).

previous trials. It seems, then, that better results for $[\delta^3 p']''$ might be obtained by differentiating the $[\delta^3 p']'$ computed with the 200 term series by finite differences, at the 40 t_k's, using the averaging procedure illustrated in Figure 3.11(a). The result is shown in Figure 3.11(b), and seems to be exactly what we want. Hence, this scheme is currently installed in our hydrofrac program.

Summary of explicit time marching scheme.

The differentiation schemes (for operations like $[\delta^3 p']''$) which we hve described above have produced quite satisfactory results in our explicit time integration procedures, insofar as accurate numerical representation is concerned.

Essentially, our results show that, even for a fairly small time step size, the solution becomes totally unacceptable after just one step forward. Examination of the trend at the borehole suggests an increasingly singular character in all variables (especially pressure). This instability is caused very simply by the failure of the algorithm to produce a rate of crack opening, $\dot{\delta}$, that simultaneously satisfies the equations of elasticity (in relation to p). For instance, a sharp cusp develops in $\dot{\delta}$ (Fig. (3.5)) -- a condition that would require a logarithmically infinite pressure at the borehole.

However, our work with explicit time integration has given us good insight into the pressure evolution problem. For instance, we have quickly recognized the need for a time integration scheme

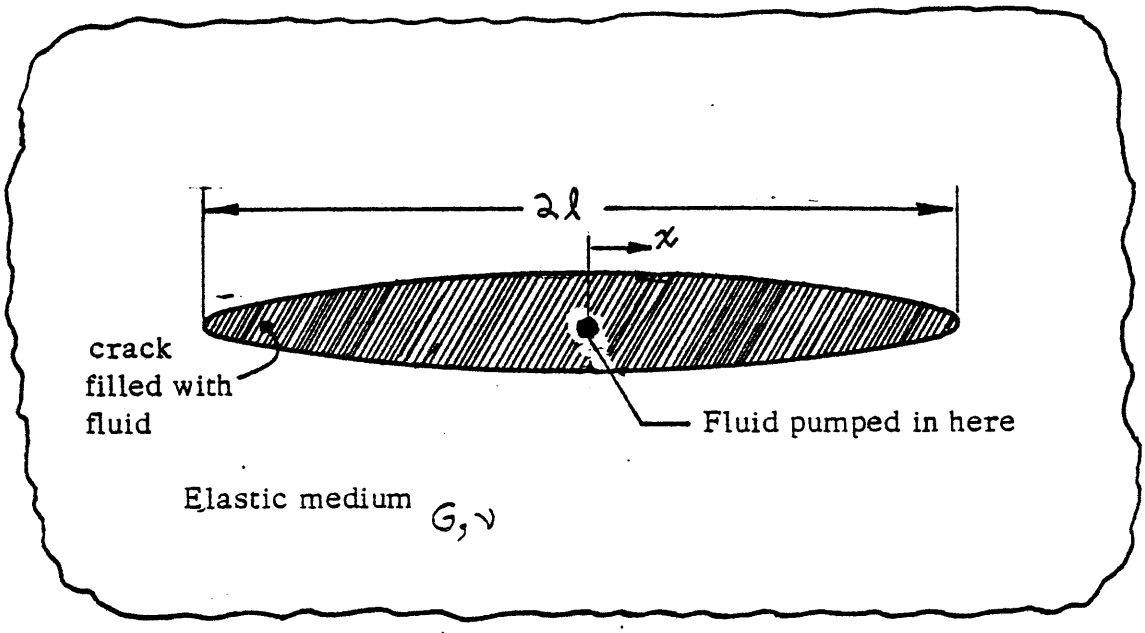


FIG. 3.1 Diagram of the pressure evolution problem. Frac. fluid is pumped in at constant pressure p_o while the crack is held at fixed length 2l.

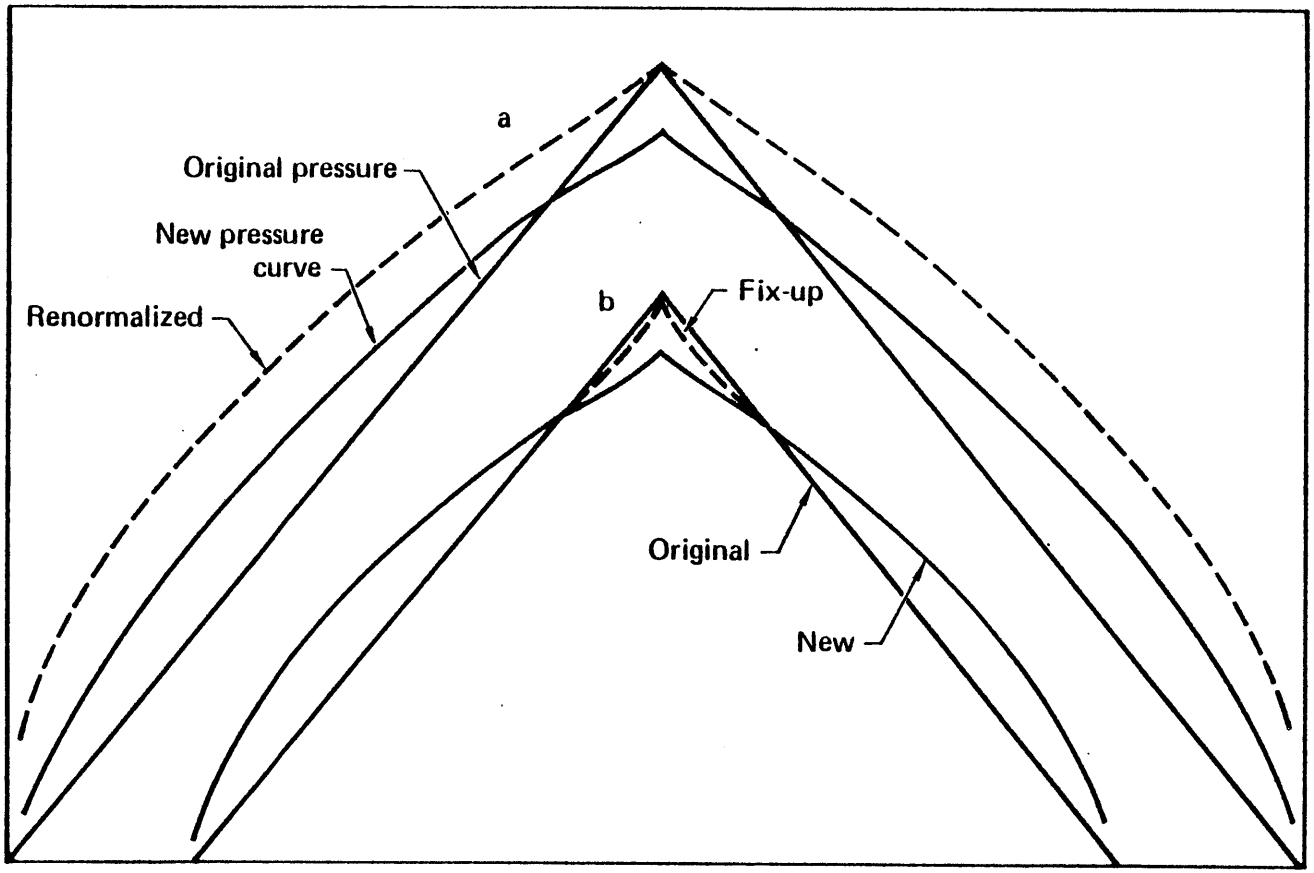


FIG.3.a Optional fixup schemes to retain specified borehole pressure. (a) Global renormalization; (b) local fixup.

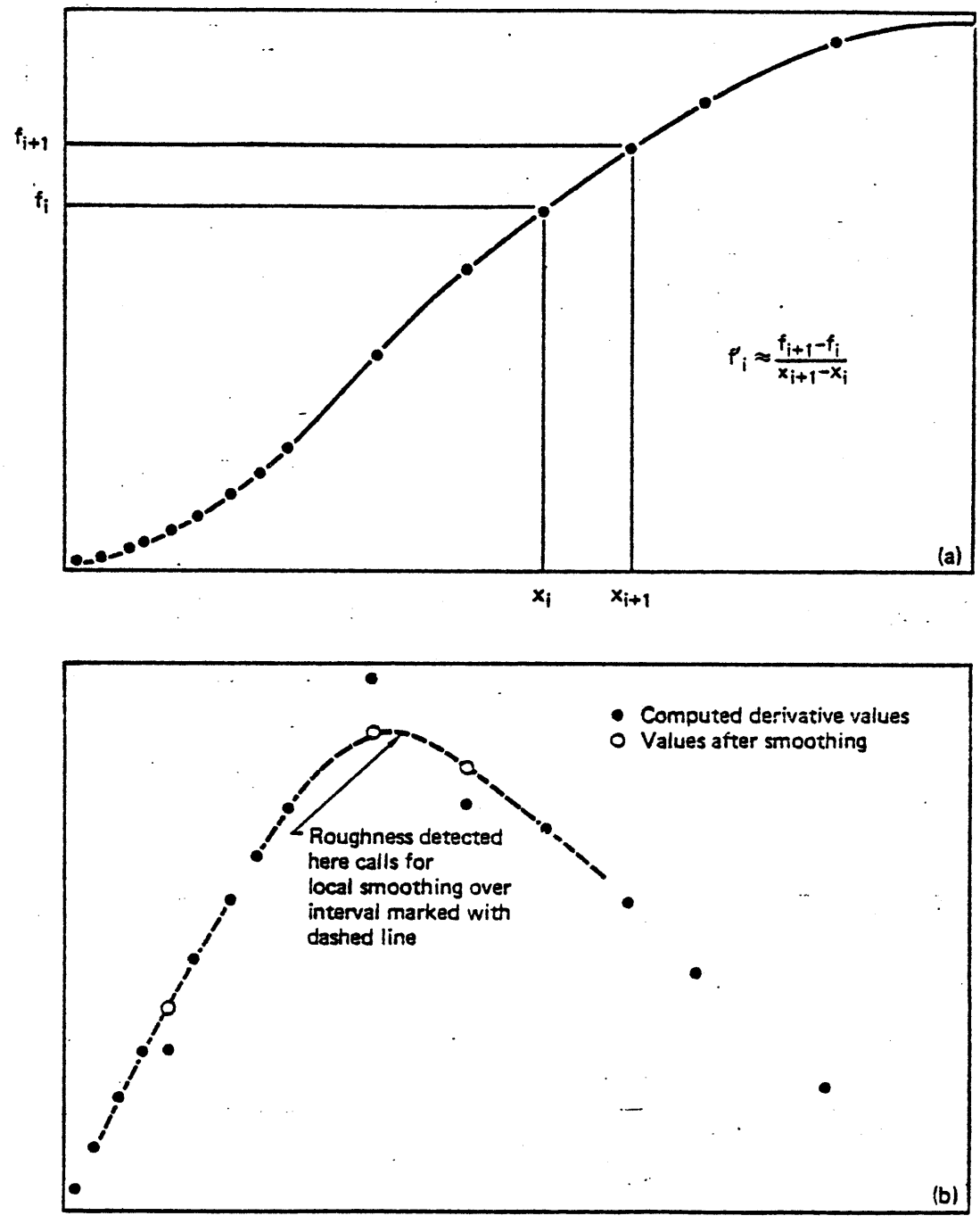


FIG.3.3 These diagrams illustrate the operation of local smoothing after differentiation. (a) The process of differentiation of half of a curve similar to $\delta^3 p'$. (b) The results of local smoothing.

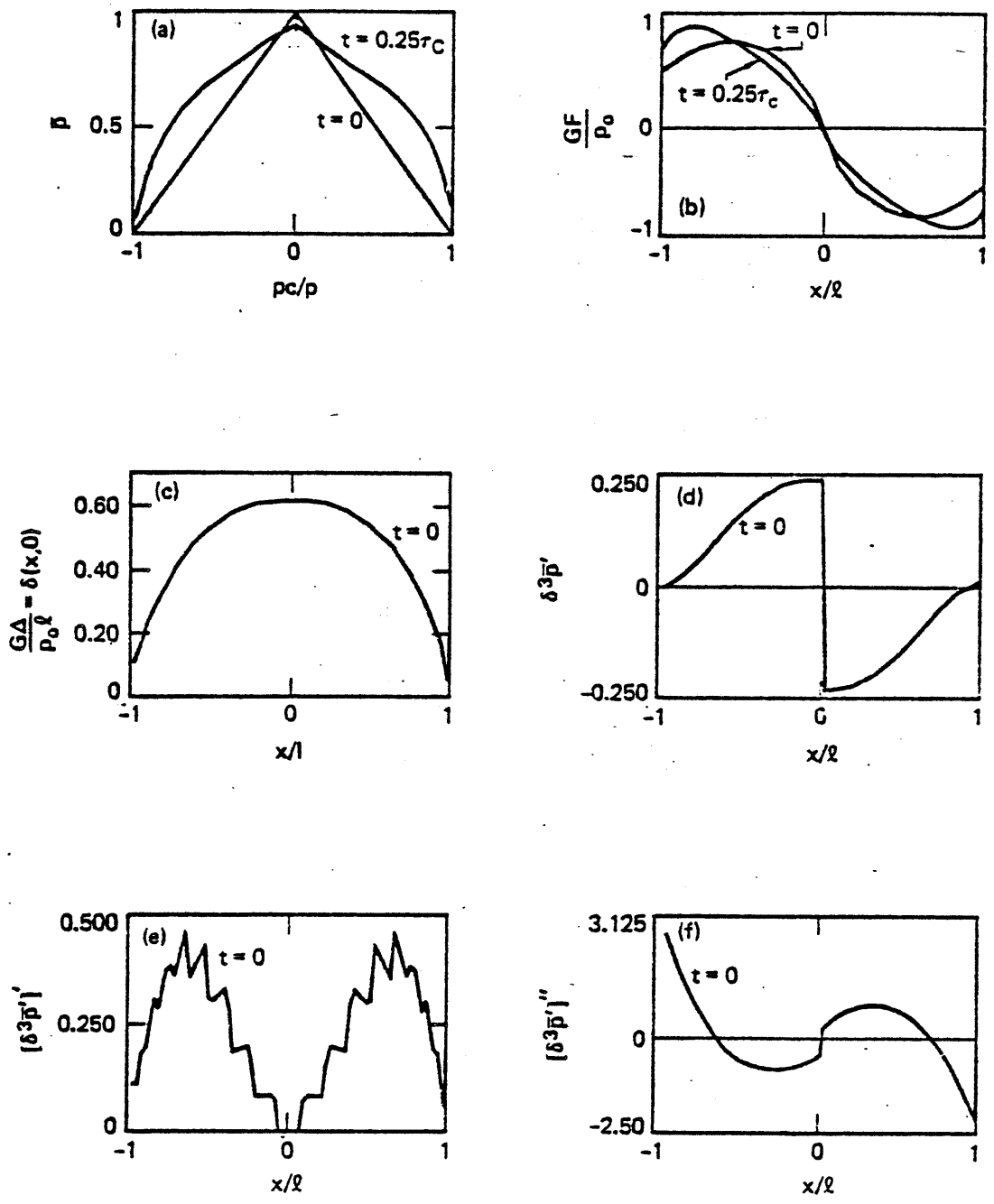


FIG3.4 Results of a trial computation of evolving fracture fluid pressure using our combined finite difference-local smoothing method for evaluating $[\delta^3 p']$. (a) Fracture fluid pressure; (b) solution of Eq. (1), $F = \mu\sqrt{1-x^2}$; (c) initial crack opening displacement, δ; (d) initial $\delta^3 p'$; (e) initial $[\delta^3 p']$ (before smoothing); (f) after smoothing.

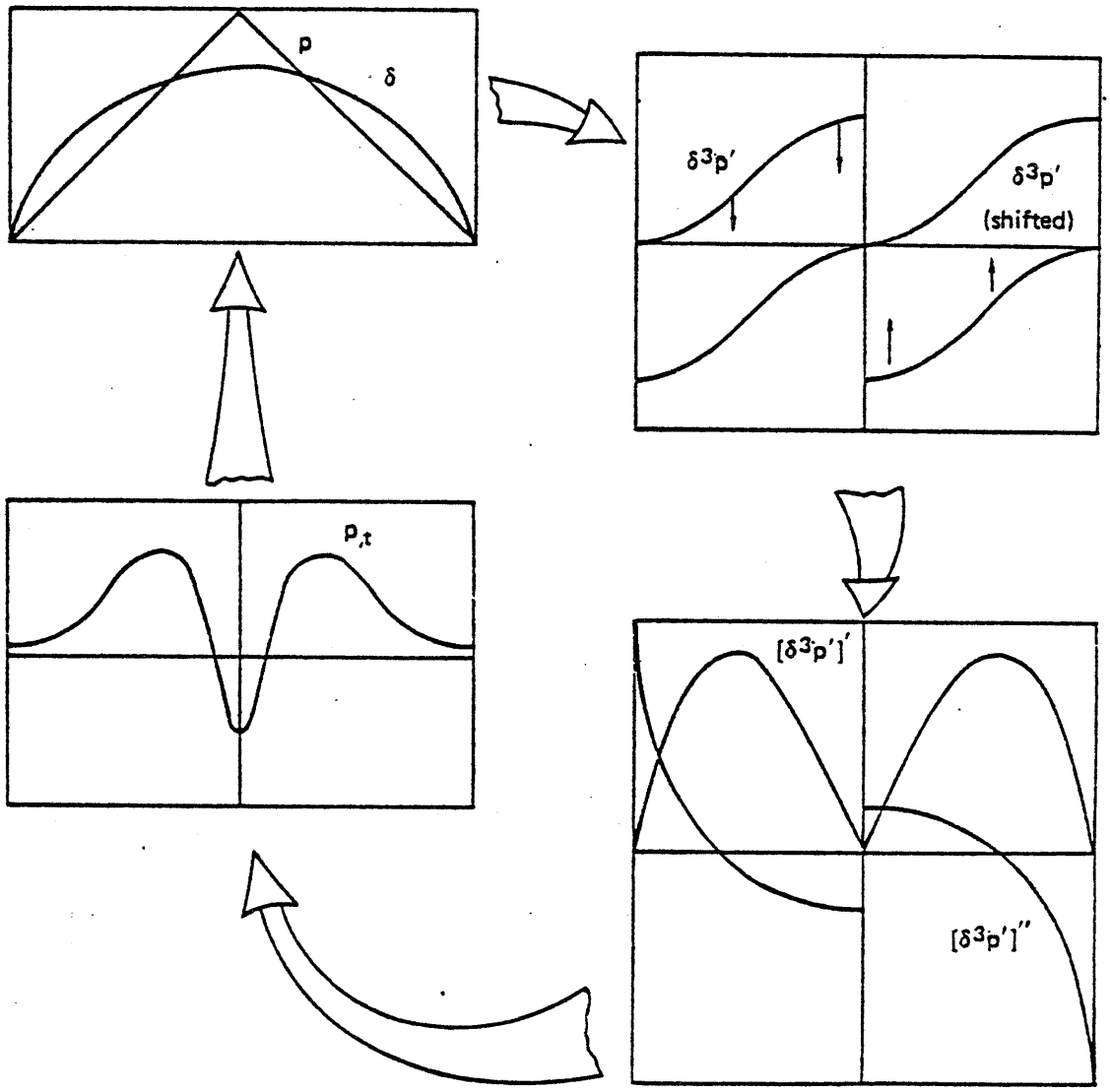


FIG. 3.5 Schematic of procedure for tracing fracture fluid pressure evolution (see Figs. 3.11 and 3.12 for details of typical cases).

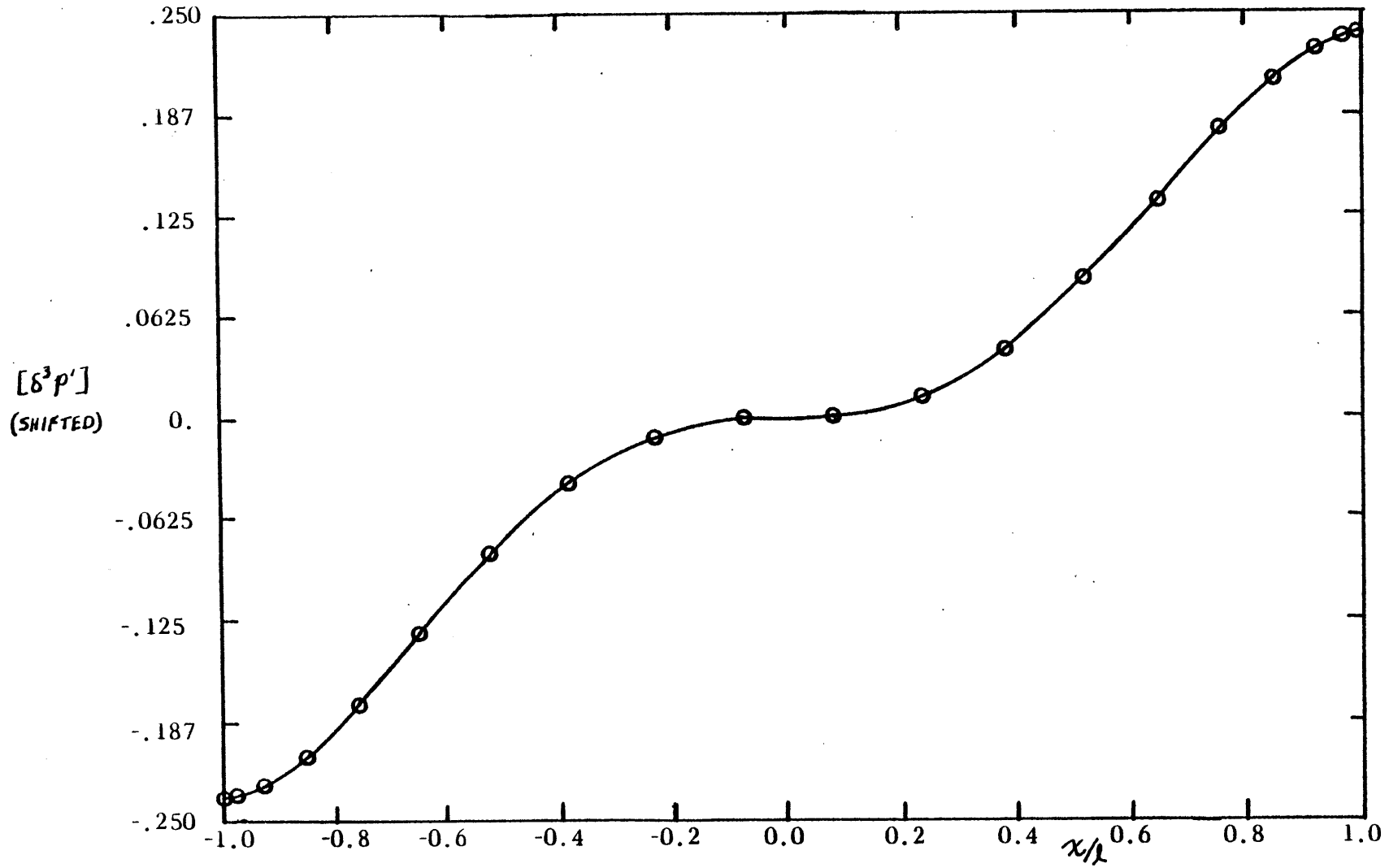


FIG. 3.6(a). This plot shows the approximation of the shifted $[\delta^3 p']$ data obtained at 20 t_k points (open circles) obtained from our hydrofrac program. The resulting series was evaluated at the 200 points joined by the solid line

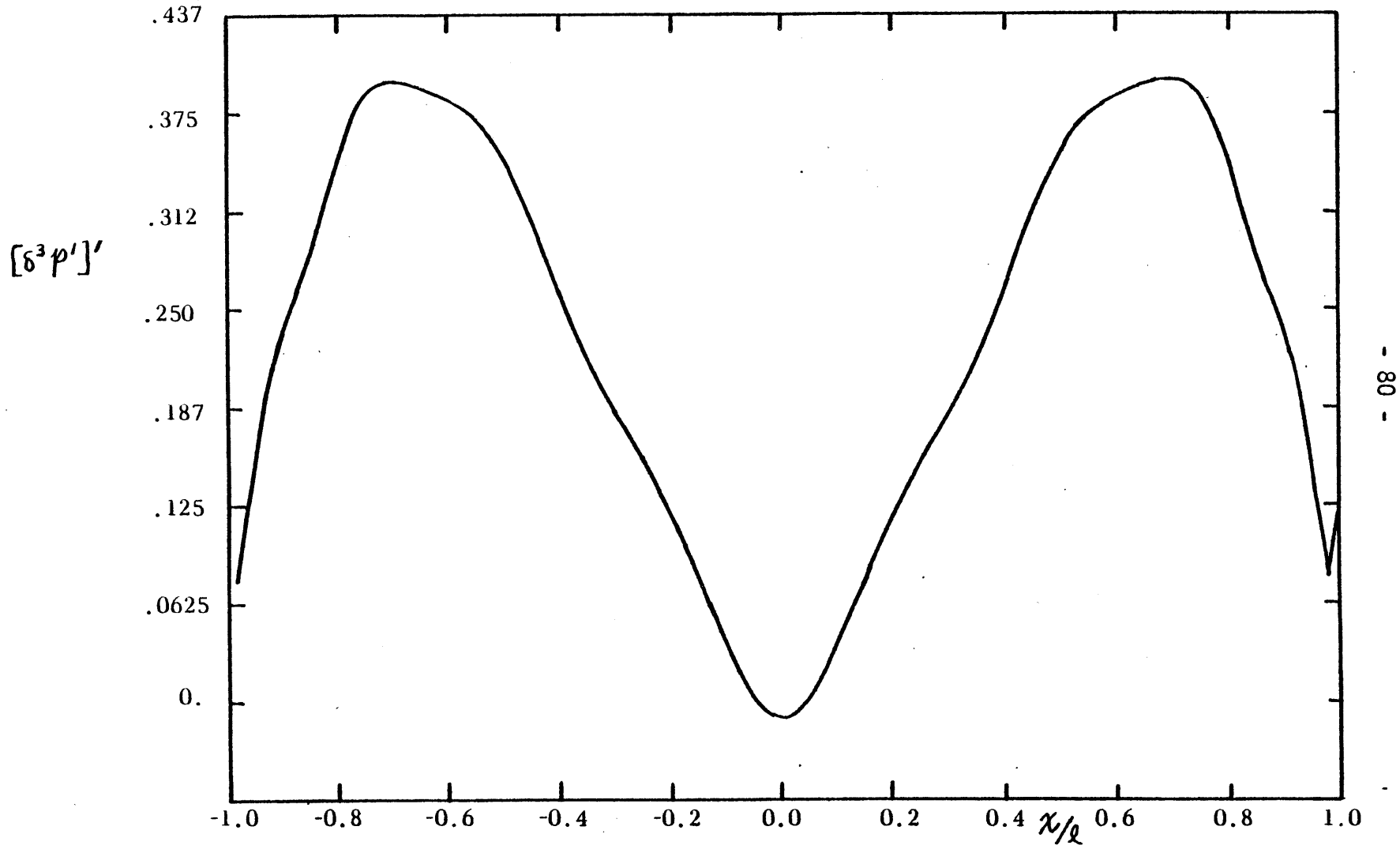


FIG. 3.6(b). Plot of $[\delta^3 p']'$ obtained by termwise differentiation of the Chebyshev series plotted in fig. 3.6(a). This curve, while reasonably smooth, is not accurate enough according to our predictions. In particular, the peaks in the vicinity of ± 0.6 seem to have the wrong character. The worst feature is the sharp upturn at +1.

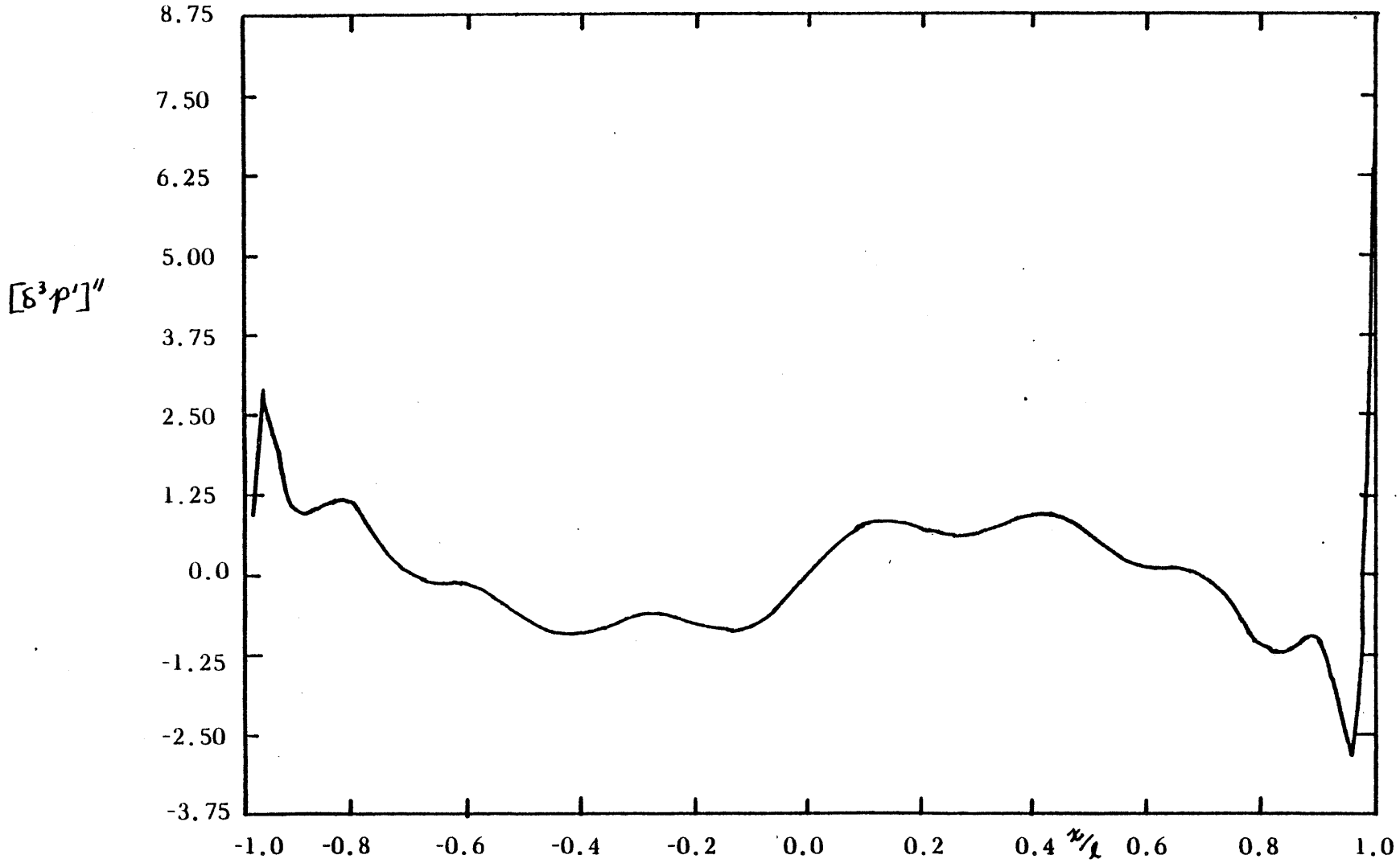


FIG. 3.6(c). Approximation of $[\delta^3 p']''$, obtained by termwise differentiation of the Chebyshev series shown in fig. 3.6(a). This curve has the right general appearance but is plagued by low frequency noise and sharp spikes, which go in the wrong direction, at each end.

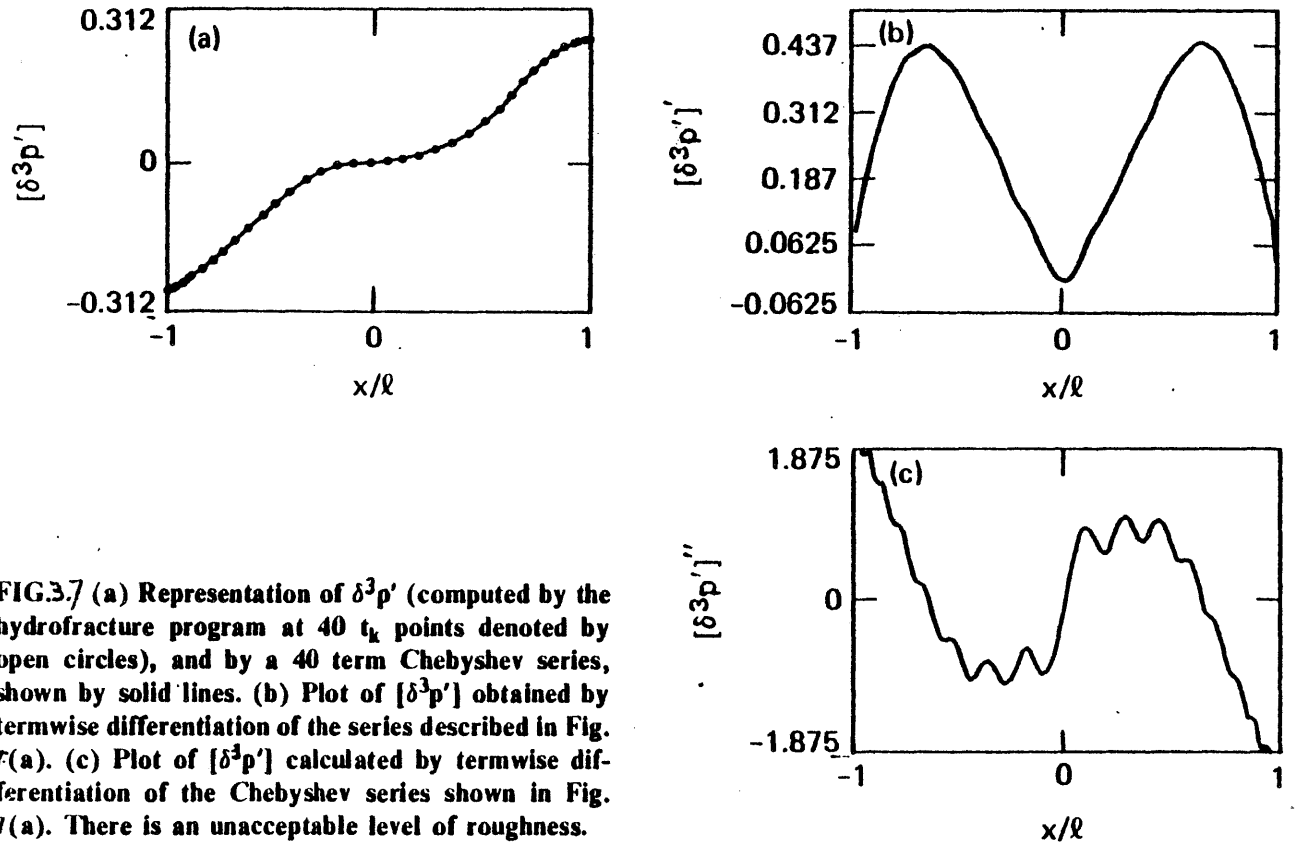
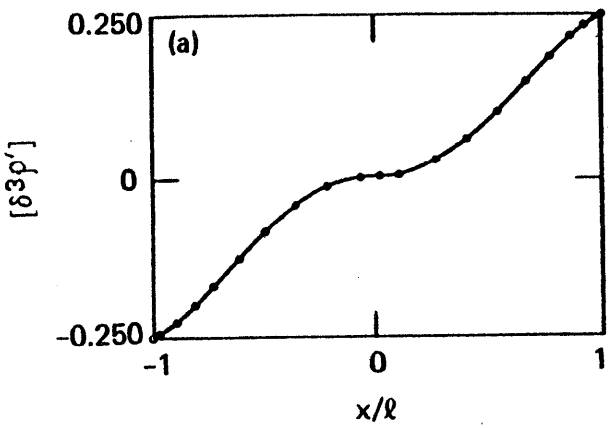
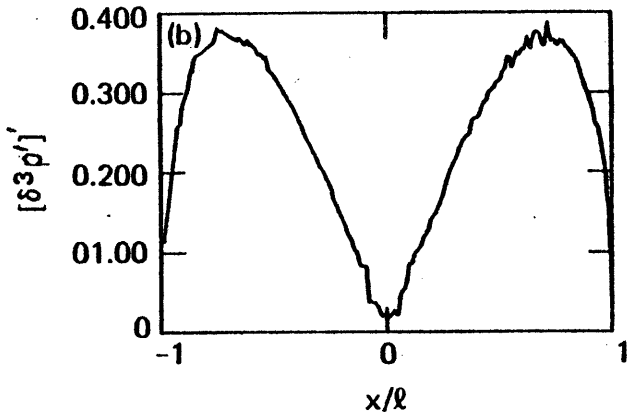


FIG.3.7 (a) Representation of $\delta^3 p'$ (computed by the hydrofracture program at 40 t_k points denoted by open circles), and by a 40 term Chebyshev series, shown by solid lines. (b) Plot of $[\delta^3 p']$ obtained by termwise differentiation of the series described in Fig. 3.7(a). (c) Plot of $[\delta^3 p']$ calculated by termwise differentiation of the Chebyshev series shown in Fig. 3.7(a). There is an unacceptable level of roughness.





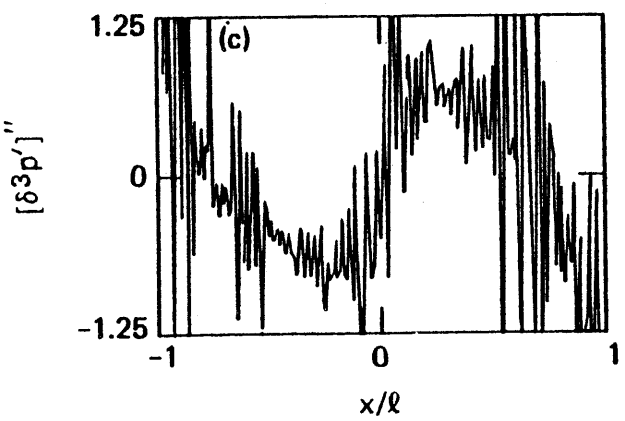
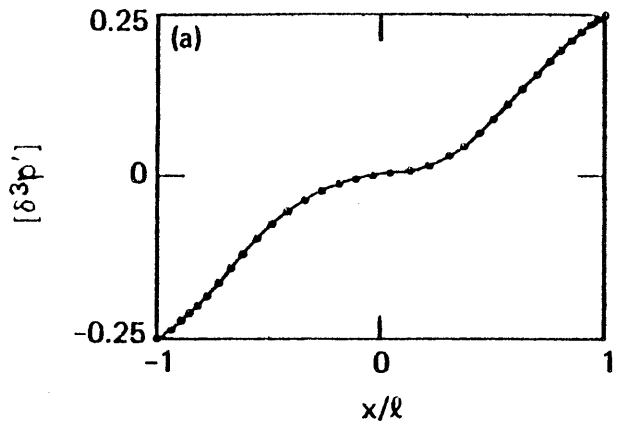
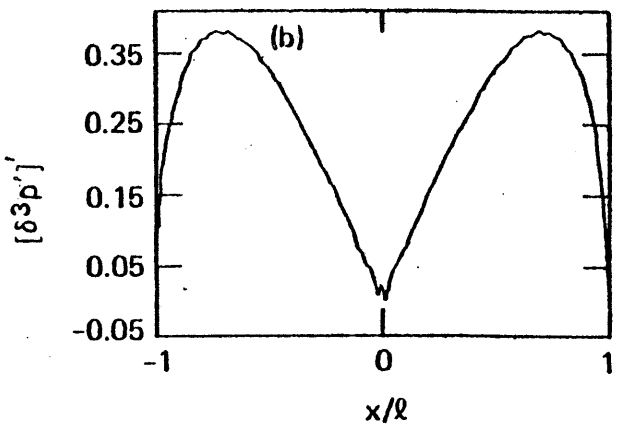


FIG.3.8 (a) Representation of $\delta^3\bar{p}'$ (calculated by the approximation $\delta^3 = (1 - t_k^2)^{3/2}$ at 20 t_k points shown by open circles) and by a 200 term Chebyshev series shown by solid line. (b) Approximation of $[\delta^3 p']'$ obtained by termwise differentiation of the series described in Fig.3.8(a) Here, although the number of t_k points (20) is the same as in the sequence of curves in Fig.3.6 the noise is of much higher frequency confirming the frequency of noise depends mostly on the number of terms in the series. (c) Approximation of $[\delta^3 p']''$ obtained by termwise differentiation of the series described in Fig.3.8(a). The true shape of this curve is completely marked by large amplitude, high frequency.





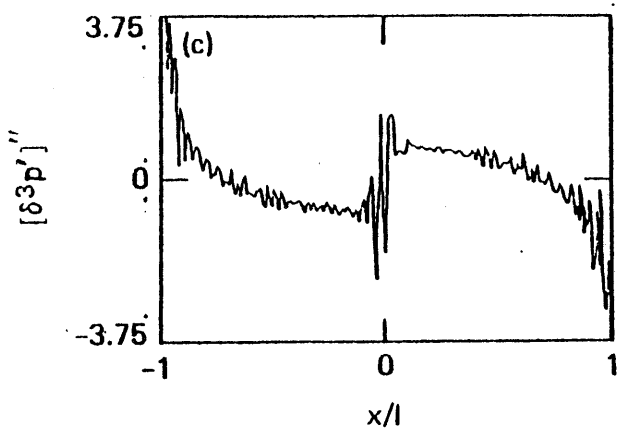
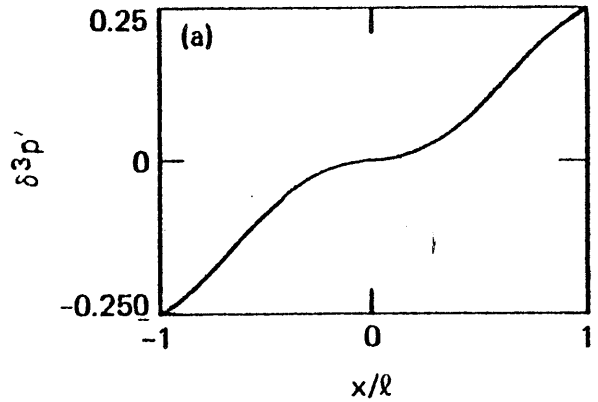


FIG. 3.9 Approximation of $\delta^3 p'$ (calculated at 40 t_k points (shown by open circles) by the approximate formula $\delta^3 \propto (1 - x^2)^{3/2}$) with a 200-term Chebyshev series solid line. (b) Plot of $[\delta^3 p']'$ computed by termwise differentiation of the 200-term Chebyshev series described in Fig. 3.9(a). Note the great improvement in smoothness, as well as accuracy near the borehole, caused by using twice the number of t_k points (40 vs 20), used to compute $\delta^3 p'$ in Fig. 3.8 (c) Plot of $[\delta^3 p']''$ obtained by termwise differentiation of the Chebyshev series described in Fig. 3.9 (a) although the noise has been substantially reduced from that in Fig. 3.8(c).



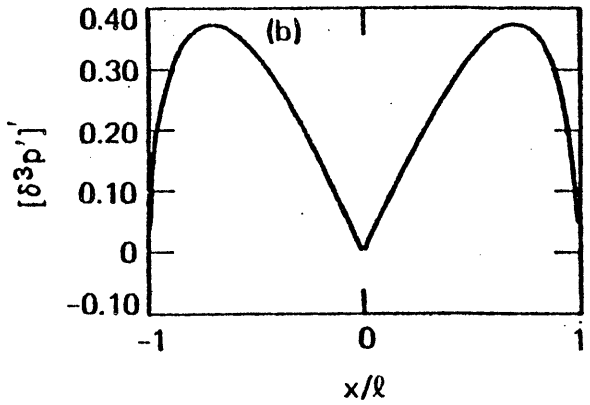
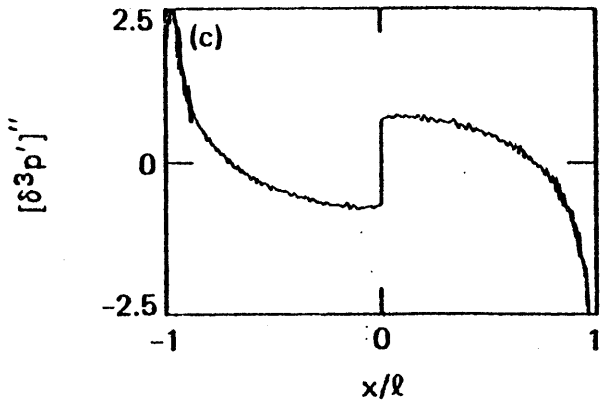
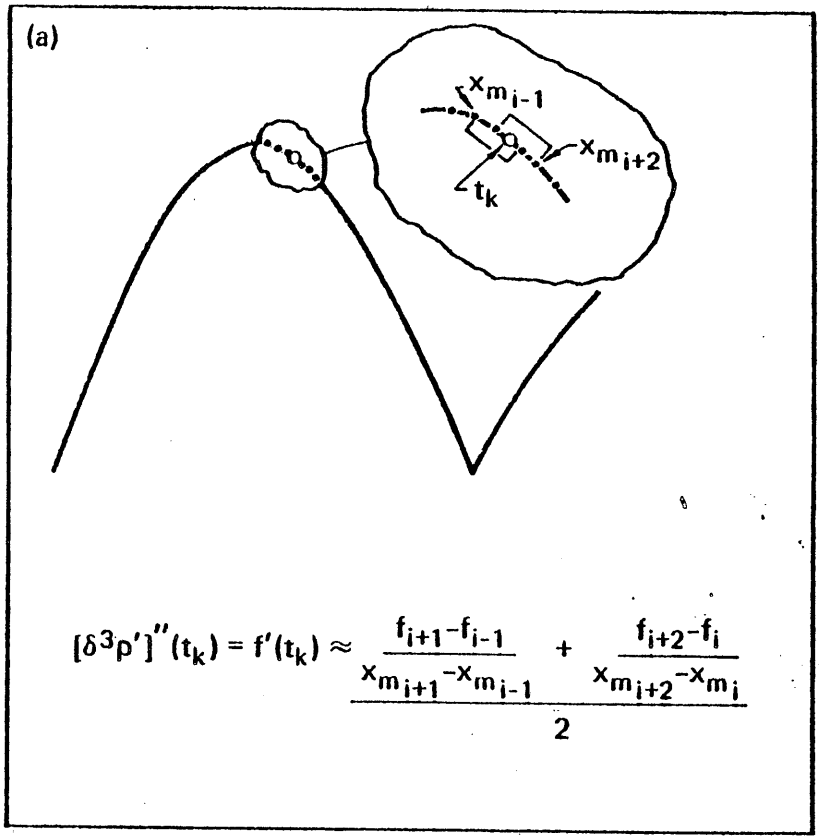


FIG.3.10 (a) Representation of $\delta^3 p'$ (computed approximately at 200 t_k points using the approximation of $\delta = \sqrt{1 - t_k^2}$) by a 200-term Chebyshev series. (b) Approximation of $[\delta^3 p']'$ obtained by termwise differentiation of the series described in Fig. 3.10(a) Smoothness and accuracy to be used in dislocation dipole model. (c) Approximation of $[\delta^3 p']''$ by termwise differentiation of the series described in Fig. 3.10(a) The noise in this curve is of about the same frequency as that in Figs.3.9(c)and3.8(c). Overall, the amplitude is much smaller (probably because the same number of terms is used in the series). This curve is not smooth or accurate enough for use in pressure evolution computations.





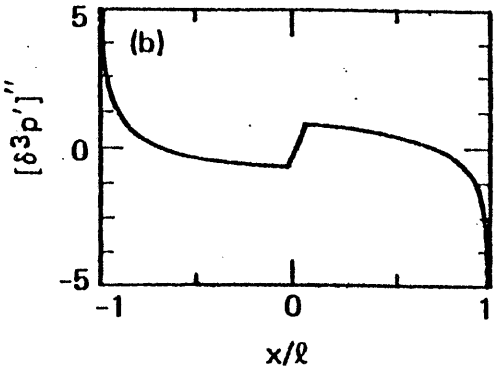


FIG.3.11(a) Illustration of the differentiation method used to obtain the curve in Fig3.11(b) from $[\delta^3 p']'$ in Fig.3.10(b). (b) This approximation of $[\delta^3 p']''$ was obtained by differentiating the curve shown in Fig.3.10(b), computed by termwise differentiation of a Chebyshev series, with the averaged finite-difference scheme illustrated in Fig. 3.11(a).

which allows simultaneous satisfaction of both mass conservation and elasticity equations. The implicit marching scheme (which uses many of the numerical techniques developed for the explicit scheme), discussed in the next section, has proved to be the best such method.

3.3 Implicit Scheme for Tracing of Frac-Fluid Pressure Evolution

Although we have been able to develop numerical procedures stable enough to allow explicit computation of evolving fluid pressure and crack geometry (Section 3.2), the results obtained were, inevitably, too little influenced by the elastic properties of the rock, and were effectively dominated by the requirement of frac-fluid mass conservation, which dictated the change in width from one step to the next. For this reason we felt it essential to employ a method by which frac-fluid pressure would be implicitly computed at each time step so as to satisfy simultaneously the requirements of elasticity and mass conservation. Further, because of the success achieved with the dislocation dipole scheme (as noted in Ref. 11), it was decided to base our implicit method on the latter, rather than the dislocation density scheme, in order to avoid anticipated trouble with the high-order differentiation.

Thus, we start with the integral equation relating frac-fluid pressure and dipole density or crack opening displacement [11], which is obtained by integrating Equation (1.1) by parts:

$$p(x_o) = \int_{-\ell}^{\ell} \gamma_D(x_o,x)\left[\delta(x)-\delta(x_o)\right]dx - \delta(x_o)\left[\gamma(x_o,\ell)-\gamma(x_o,-\ell)\right], \qquad (3.11a)$$

where, for a homogeneous isotropic infinite medium,

$$\gamma_D = \frac{-1}{2\pi(1-\nu)}(x_o-x)^{-2} \equiv -\frac{d\gamma}{dx}. \qquad (3.11b)$$

Here, γ_D is the influence function which gives the stress at point x_0 due to the difference in dipole strengths δ at points x and x_0, and γ is the analogous influence function associated with dislocations [8]. Differentiating Equation (3.11a) with respect to time gives (for time-dependent and stationary crack-tips)

$$\dot{p}(x_o) = \int_{-\ell}^{\ell} \gamma_D(x_o,x)\left[\dot{\delta}\right]_{x_o}^{x} dx - \dot{\delta}(x_o)\left[\gamma(x_o,+\ell)-\gamma(x_o,-\ell)\right] \qquad (3.12)$$

We have seen before (Section 3.2) that the simplest fluid flow model -- Poiseuille flow -- gives the result that $\hat{\eta}\dot{\delta} = [\delta^3 p']'$ (here the apostrophe denotes* spatial differentiation, while the "dot" indicates differentiation with respect to time). If we now make the following approximations (which can, of course, be refined),

$${}^{t}\dot{\delta} = \alpha\,{}^{t+\Delta t}\dot{\delta} + (1-\alpha)\,{}^{t}\dot{\delta}\,, \quad \dot{p} = \frac{{}^{t+\Delta t}p - {}^{t}p}{\Delta t} \qquad (3.13)$$

*Here $\hat{\eta}$ denotes an effective viscosity and we have used G, ν for shear modulus and Poisson ratio of the surrounding rock.

and substitute into Equation (3.12), we get an equation which may be re-arranged so that only terms evaluated at time $t+\Delta t$ are on the left and only those at time t are on the right. When we completely non-dimensionalize all terms, we get (assuming fluid penetration all the way to the tip, viz. a stationary crack)

$$
\begin{aligned}
&{}^{t+\Delta t}p(x_o) - \frac{\alpha\Delta t}{\tau_c}\int_{-1}^{1}\gamma_D(x_o,x)\left\{{}^{t+\Delta t}[\delta^3 p']'(x) - {}^{t+\Delta t}[\delta^3 p']'(x_o)\right\}dx \\
&\quad + \alpha\frac{\Delta t}{\tau_c}\,{}^{t+\Delta t}[\delta^3 p']'(x_o)\left[\gamma(x_o,1)-\gamma(x_o,-1)\right] \\
&= {}^{t}p(x_o) + \frac{(1-\alpha)\Delta t}{\tau_c}\int_{-1}^{1}\gamma_D(x_o,x)\left\{{}^{t}[\delta^3 p']'(x) - {}^{t}[\delta^3 p']'(x_o)\right\}dx \\
&\quad - \frac{(1-\alpha)\Delta t}{\tau_c}\,{}^{t}[\delta^3 p']'(x_o)\left[\gamma(x_o,1)-\gamma(x_o,-1)\right] \qquad (3.14)
\end{aligned}
$$

where, again, it is understood that p and δ are now dimensionless: that is,

$$
\delta \leftarrow \frac{G\delta}{p_o \ell}\ ,\quad p \leftarrow \frac{p}{p_o}\ ,\ \text{and}\ \ \tau_c = \frac{\hat{\eta}}{G}\left(\frac{G}{p_o}\right)^3
$$

is the characteristic time predicted by Cleary [23]. The parameter α is chosen to provide the most stable solution; in fact, it will be seen later that the best choice is $\alpha = 1$. If we make the assumption that ${}^{t+\Delta t}\delta = {}^{t}\delta$ (or any other relation between ${}^{t+\Delta t}\delta$ and ${}^{t}\delta$) we can re-write Equation (3.14) as a set of linear

algebraic equations by using the appropriate discrete formulas for integration and differentation. First, we approximate the integrals in Equation (3.14) by the Gauss-Chebyshev formula:

$$\int_{-1}^{1} \gamma_D(x_r, t_i)\left[\delta^3 p'\right]' \Big|_{x_r}^{t_i} dx = \frac{\pi}{N} \sum_{i=1}^{N} \gamma_D(x_r, t_i)\left[\delta^3 p'\right]' \Big|_{x_r}^{t_i} \sqrt{1-t_i^2} \qquad (3.15a)$$

$$x_r = -\cos(\pi r/N) \qquad r = 1, \ldots, N-1 \qquad (3.15b)$$

$$t_i = -\cos(\pi(2i-1)/2N) \qquad i = 1, \ldots, N \qquad (3.15c)$$

Since we wish to formulate the equations in terms of p, we need to represent $[\delta^3 p']$ in terms of δ and p. Our previous success in termwise differentiation of a Chebyshev series leads us to use that method here:

$$\left[\delta^3 p'\right]'(x) = \sum_{j=1}^{M} T_j'(x) \left\{ \frac{4}{\pi^2} \int_{-1}^{1} \left[\frac{T_j(x)\,\delta^3(x)}{\sqrt{1-x^2}} \sum_{\ell=1}^{L} T_\ell'(x) \int_{-1}^{1} \frac{T_\ell(x)\,p(x)\,dx}{\sqrt{1-x^2}} \right] dx \right\} \qquad (3.16)$$

Since we will need to impose two constraints on the solution for ${}^{t+\Delta t}p$ (viz. we will in particular maintain the borehole pressure at some desired value, and ${}^{t+\Delta t}p$ will be such that ${}^{t+\Delta t}\delta\big|_{-1}^{+1} = 0$) we will have one more equation than unknowns (i.e., N+1 equations, N unknowns) unless we obtain $p(x_r)$ from a set of N+1 points by interpolation. Again we make use of the Chebyshev series:

$$p(x_n) = \sum_{l=0}^{L-1} T_l(x_n) \left[\frac{2}{\pi} \int_{-1}^{1} \frac{T_l(x)\, p(x)}{\sqrt{1-x^2}}\, dx \right] \left(1 - \tfrac{1}{2}\delta_{l0}\right) \tag{3.17}$$

If we apply the Gauss-Chebyshev integration formula in Equations (3.16, 17) substitute into Equation (3.15) and thence to Equation (3.14) we obtain our system of equations:

$$\sum_{l=1}^{L} \left(1 - \tfrac{1}{2}\delta_{l1}\right) T_{l-1}(x_n) \sum_{\Delta=1}^{L} \frac{2}{L} T_{l-1}(t_\Delta) \left[{}^{t+\Delta t}p(t_\Delta) - {}^{t}p(t_\Delta) \right]$$

$$= \frac{4\Delta t}{LM\tau_e} \left(\frac{\pi}{N}\right) \sum_{i=1}^{N} \gamma_D(x_n, t_i) \sum_{j=1}^{M} \left\{ T_j'(t_i) - T_j'(x_n) \right\} \sqrt{1-t_i^2} \sum_{k=1}^{M} T_j(t_k)\, \delta^3(t_k)$$

$$\times \sum_{l=1}^{L} T_l'(t_\Delta) \sum_{\Delta=1}^{L} T_l(t_\Delta) \left[\alpha\, {}^{t+\Delta t}p(t_\Delta) + (1-\alpha)\, {}^{t}p(t_\Delta) \right]$$

$$- \frac{4\Delta t}{LM\tau_e} \left[\gamma(x_n, 1) - \gamma(x_n, -1) \right] \sum_{j=1}^{M} T_j'(x_n) \sum_{k=1}^{M} T_j(t_k)\, \delta^3(t_k)$$

$$\times \sum_{l=1}^{L} T_l'(t_\Delta) \sum_{\Delta=1}^{L} T_l(t_\Delta) \left[\alpha\, {}^{t+\Delta t}p(t_\Delta) + (1-\alpha)\, {}^{t}p(t_\Delta) \right] \tag{3.18a}$$

Here we find it natural to make the following identifications

$$t_{k,\Delta} = -\cos\left[\frac{\pi(2(k,\Delta)-1)}{2L} \right], \quad k, \Delta = 1, \ldots, L, \quad L \equiv M \tag{3.18b}$$

Equations (3.18) may be simplified, and the time required to set up and solve them reduced, through the use of the following matrices:

$$A_{rl} \equiv (1-\tfrac{1}{2}\delta_{l1})\, T_{l-1}(x_r) \qquad r=1,\ldots,N-1\ ,\ l=1,\ldots,L \tag{3.19a}$$

$$A'_{ls} \equiv \frac{2}{L}\, T'_{l-1}(t_s) \qquad s=1,\ldots,L\ ,\ l=1,\ldots,L \tag{3.19b}$$

$$B_{rj} \equiv \frac{\pi}{N}\sum_{i=1}^{N} \gamma_D(x_r,t_i)\left[T'_j(t_i)-T'_j(x_r)\right]\sqrt{1-t_i^2} \qquad r=1,\ldots,N-1,\ j=1,\ldots,L \tag{3.19c}$$

$$C_{rj} \equiv T'_j(x_r)\ , \tag{3.19d}$$

$$C'_{kj} \equiv T'_j(t_k)\ , \qquad k=1,\ldots,M \tag{3.19e}$$

$$D_{jk} \equiv \frac{2}{M}\, T_j(t_k), \tag{3.19f}$$

$$\Delta_{jk} \equiv \delta^3(t_k)\,\delta_{jk}, \tag{3.19g}$$

$$E_{kl} \equiv T'_l(t_k), \tag{3.19h}$$

$$F_{ls} \equiv \frac{2}{L}\, T_l(t_s), \qquad s=1,\ldots,L, \tag{3.19i}$$

$$G_{rj} \equiv \delta_{rj}\left[\gamma(x_r,1)-\gamma(x_r,-1)\right] \tag{3.19j}$$

where

$$\delta_{ij} = \begin{cases} 1, & i=j \\ 0, & i \neq j \end{cases}$$

Our experience with fitting functions by a Chebyshev series indicates that when the series is to be differentiated it is best to transform the function so as to make it pass through the origin and be antisymmetric, then re-transform the series if necessary (cf. Section 3.2). To effect the necessary transformations, we define these matrices:

$$H_{kq} \equiv -\delta_{kq}\,\mathrm{sign}(t_k) + \delta_{(\frac{L}{2})k}\,\mathrm{sign}(t_k) \tag{3.19k}$$

$$S_{ij} \equiv \delta_{ij} + \delta_{(\frac{M}{2})j}\,\mathrm{sign}(t_i) \tag{3.19l}$$

$$T_{jk} \equiv -\delta_{jk}\,\mathrm{sign}(t_k) \tag{3.19m}$$

where

$$\mathrm{sign}(x) = \begin{cases} +1, & x>0 \\ -1, & x<0 \end{cases} \tag{3.19n}$$

and the borehole is located at $t_{M/2}$.

Here $\underline{H}$ and $\underline{T}$ are used for transforming p before and after fitting and differentiation, and $\underline{S}$ is used for transforming $\delta^3 p'$ before fitting and differentiation.

We may now define the "secondary" matrices

$$\underline{M1} \equiv \underline{A}\underline{A}', \quad \underline{M2} \equiv \underline{B}\underline{D}\underline{S}, \quad \underline{M3} \equiv \underline{T}\underline{E}\underline{F}\underline{H}, \quad \underline{M4} \equiv \underline{G}\underline{C}\underline{D}\underline{S}, \quad \underline{M5} \equiv \underline{C}'\underline{D}\underline{S} \tag{3.20}$$

so that Equation (3.18) can be written more compactly:

$$\left\{\underline{M1} - \frac{\alpha\Delta t}{\tau_c}\underline{M2}\,\underline{\Delta}\,\underline{M3} + \frac{\alpha\Delta t}{\tau_c}\underline{M4}\,\underline{\Delta}\,\underline{M3}\right\}^{t+\Delta t}\underline{p}$$

$$= \left\{\underline{M1} + \frac{(1-\alpha)\Delta t}{\tau_c}\underline{M2}\,\underline{\Delta}\,\underline{M3} - \frac{(1-\alpha)\Delta t}{\tau_c}\underline{M4}\,\underline{\Delta}\,\underline{M3}\right\}^{t}\underline{p} \tag{3.21a}$$

or

$$\underline{M}^{t+\Delta t}\underline{p} = \underline{R} \tag{3.21b}$$

As mentioned before, we need to impose two constraints on the solution ${}^{t+\Delta t}p$. The first of these is the requirement that ${}^{t+\Delta t}\dot{\delta}\big|_{-1}^{+1} = 0$ (by analogy with closure in our dislocation density schemes), which can be realized by adding a row to $\underline{B}$ and $\underline{R}$

$$\underline{B}_{(N+1)j} = T_j'(+1) - T_j'(-1), \quad R_{N+1} = 0 \tag{3.22}$$

The secondary constraint is on the borehole pressure, the value of which we wish to specify. We impose this constraint by adding rows to $\underline{M}$ and $\underline{R}$:

$$\underline{M}_{(N+2)j} = \delta_{(\frac{M}{2})j}, \quad R_{N+2} = p_o \tag{3.23}$$

Our procedure for computing fluid pressure and crack opening starts by evaluating the matrices in Equation (3.21a), which need only be done once. Then, starting with an initial pressure distribution and crack geometry, we can compute the new pressure (viz. ${}^{t+\Delta t}p$). The new crack opening is obtained from the relation

$$ {}^{t+\Delta t}\delta = \left[(1-\alpha)\,{}^{t}\dot{\delta} + \alpha\,{}^{t+\Delta t}\dot{\delta} \right] \frac{\Delta t}{\tau_c} + {}^{t}\delta \qquad (3.24a) $$

where

$$ \dot{\delta} = \underline{M5}\,\underline{\Delta}\,\underline{M3}\,\underline{p} \qquad (3.24b) $$

We may then continue to compute the next pressure, and so on. Note that ${}^{t+\Delta t}\delta$ is necessarily consistent with ${}^{t+\Delta t}p$; iteration on δ^3 in $\delta^3 p'$, although rigorously needed, produces only small effects for reasonable time steps. The implicit scheme may also be formulated on the basis of local interpolation methods [11]; although the local matrices would be simpler to generate, global interpolation offers the advantage of greater accuracy for the same number of nodal points, and it may provide more stability.

Results

Typical results from the global formulation of our implicit integration scheme are presented in Figures 3.12 - 3.18. These results yield great insight on the effects of the value of α,

initial pressure distribution, and time-step size.

Figure 3.12 shows the result of a preliminary validation of the FORTRAN coding of our algorithm (see Chapter 4), especially the formulation and computation of the matrices in Eqn. (3.19). By using $p(t_s)=t_s$, $\delta^3(t_s)=\frac{1}{2}\,\mathrm{Sin}^{-1}(t_s) + \frac{1}{2}\,t_s\sqrt{1-t_s^2}$, and replacing $\underline{H}$, $\underline{S}$, and $\underline{I}$ with identity matrices, we have $[\delta^3 p']' = \sqrt{1-t_s^2}$ which, when integrated with χ_D, should produce a constant $\dot{p}$. This curve, while constant over most of the interval (-1,1), has "spikes" at either end which are apparently the result of slight inaccuracies in the explicit computation of the various derivatives; we plan to remedy this, but it has not caused any serious perturbations in the rest of our computations.

Figure 3.13 shows a set of pressure evolution curves, obtained with $\alpha = 1$ and $t=.25\tau_c$, in which the borehole pressure is maintained at a constant level at each time step; as is the case for all other figures except Fig. 3.18, it starts from the triangular pressure distribution shown in Figure (3.13a). We note that, near the borehole, p has the positive curvature necessary to produce an ever increasing crack opening at the borehole (since $\dot{\delta} = (\delta^3)'p' + \delta^3 p''$), which is consistent with the continuous addition of frac-fluid. At $t=1.5\tau_c$ the fluid pressure becomes essentially constant over the crack length, verifying that τ_c is an excellent estimate of the time required for pressure penetration to the crack tips. Also at

$t=1.5\,\tau_c$ we note that the crack opening is very close to the analytical result, $\delta = \sqrt{1-x^2}$, that we would expect from a uniform pressure.

Figure 3.14 shows results of computations similar to those in Figure 3.14 except that we have chosen $\alpha = .5$, bringing $t+\Delta t_p$ under the influence of the requirement of mass conservation at time t. The effect is that the algorithm tends to become unstable for t near τ_c. Similar calculations with $\alpha = .9$ produced the results shown in Figure 3.16: the solutions exhibit nearly as much stability as for $\alpha = 1$. We thus conclude that, in general, the best results are to be obtained when $\alpha = 1$ and that there is actually a slight computational disadvantage to using $\alpha < 1$.

The effect of changing the time step size is shown in Figures 3.16, 17 along with the previous results (Figure 3.13). Comparison reveals that there is enough difference between the curves obtained by various step sizes to warrant the use of $t = .10\tau_c$, or smaller, for all but rough calculations.

Figure 3.18 shows the effect of using a different initial pressure distribution, in this case $p(x,t=0) = \sqrt{1+|x|}$ rather than the triangular distribution used in the other cases. Two phenomena

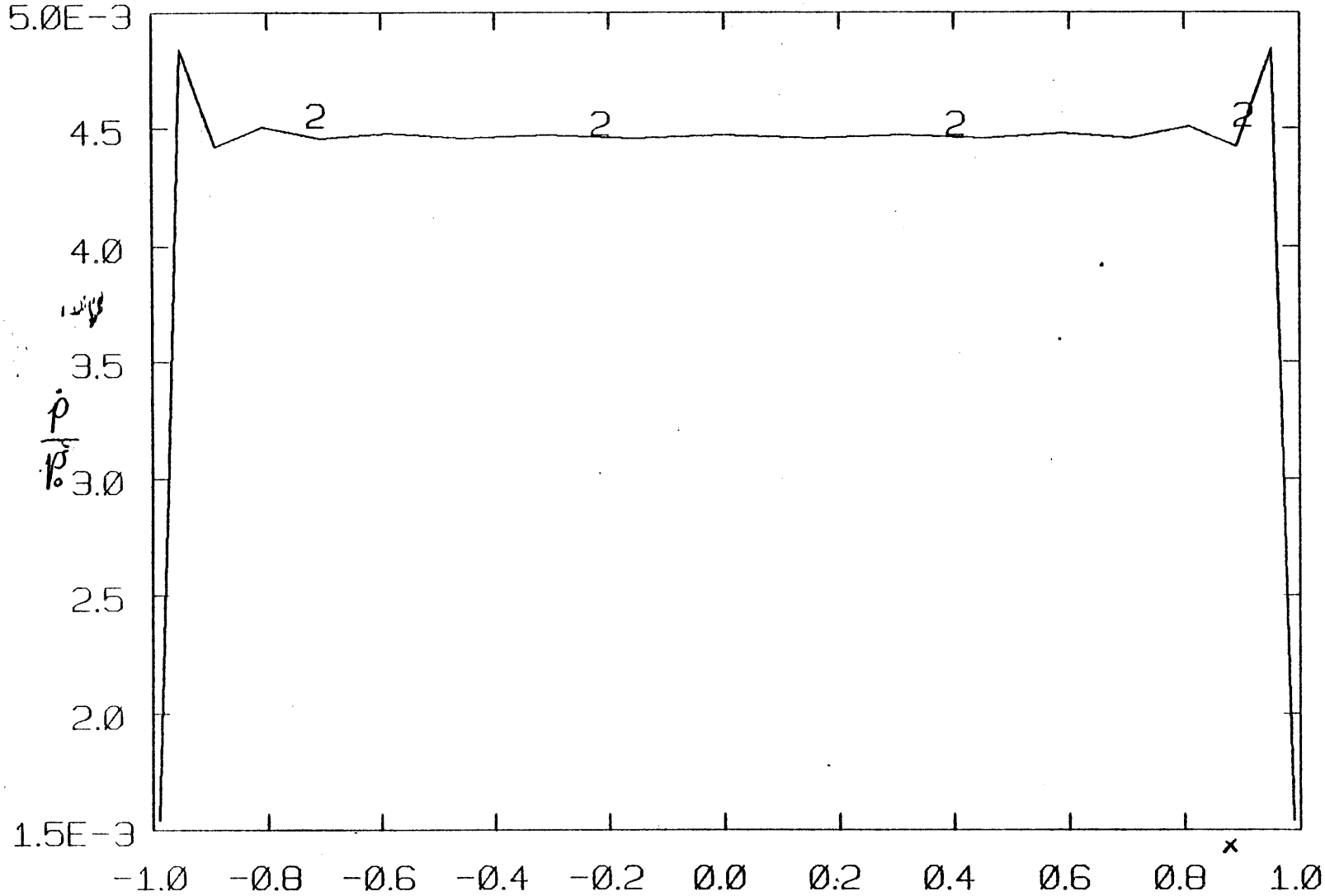


Fig. 3.12. Result of a test of our pressure evolution program in which $b = \sqrt{1-x^2}$ was simulated and then integrated with γ_D via the matrix operations described in section 3. The result is the expected constant "ρ/ρ_0", which deviates from uniformity at the tips because of errors in explicit differentiation.

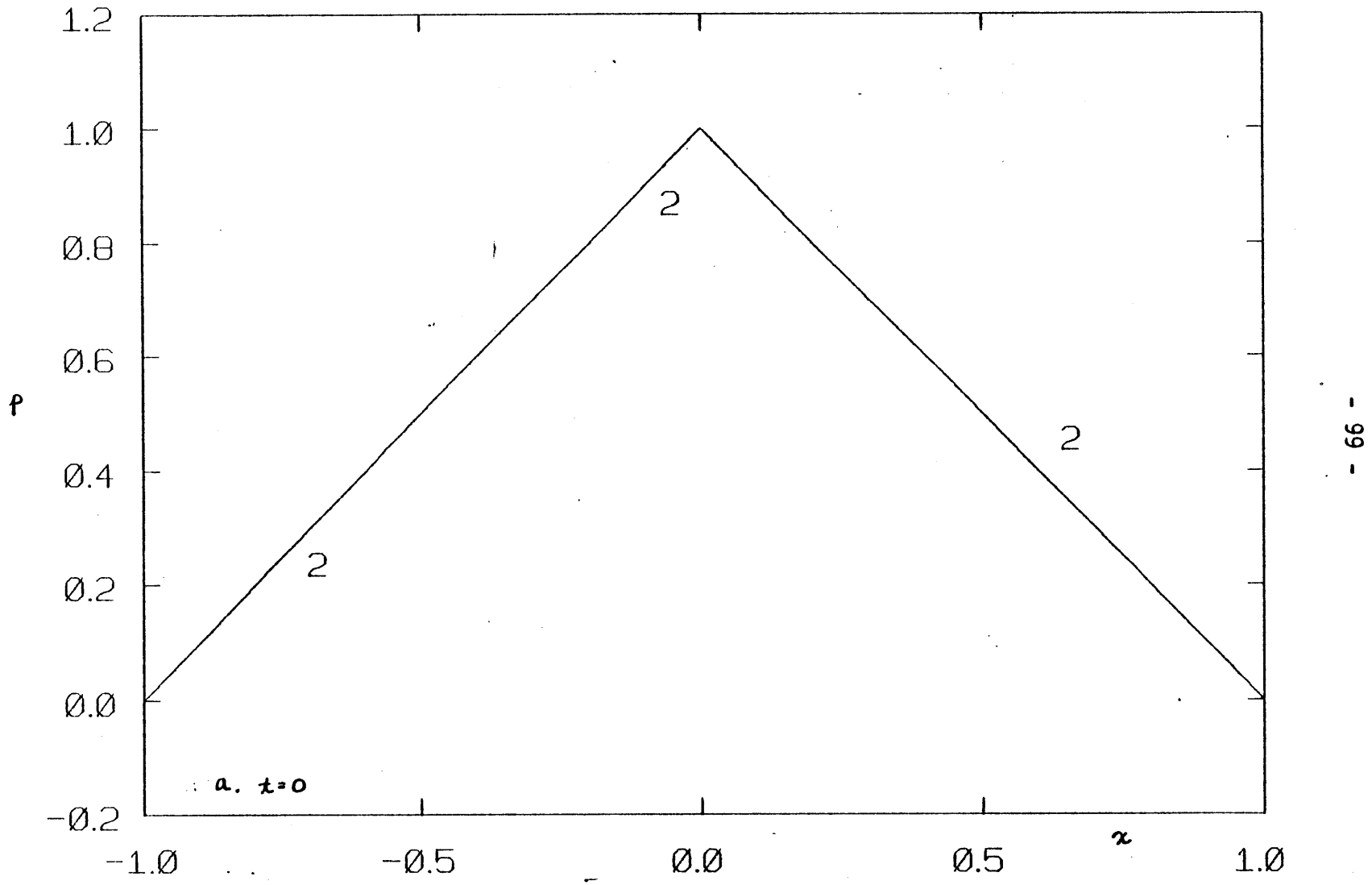
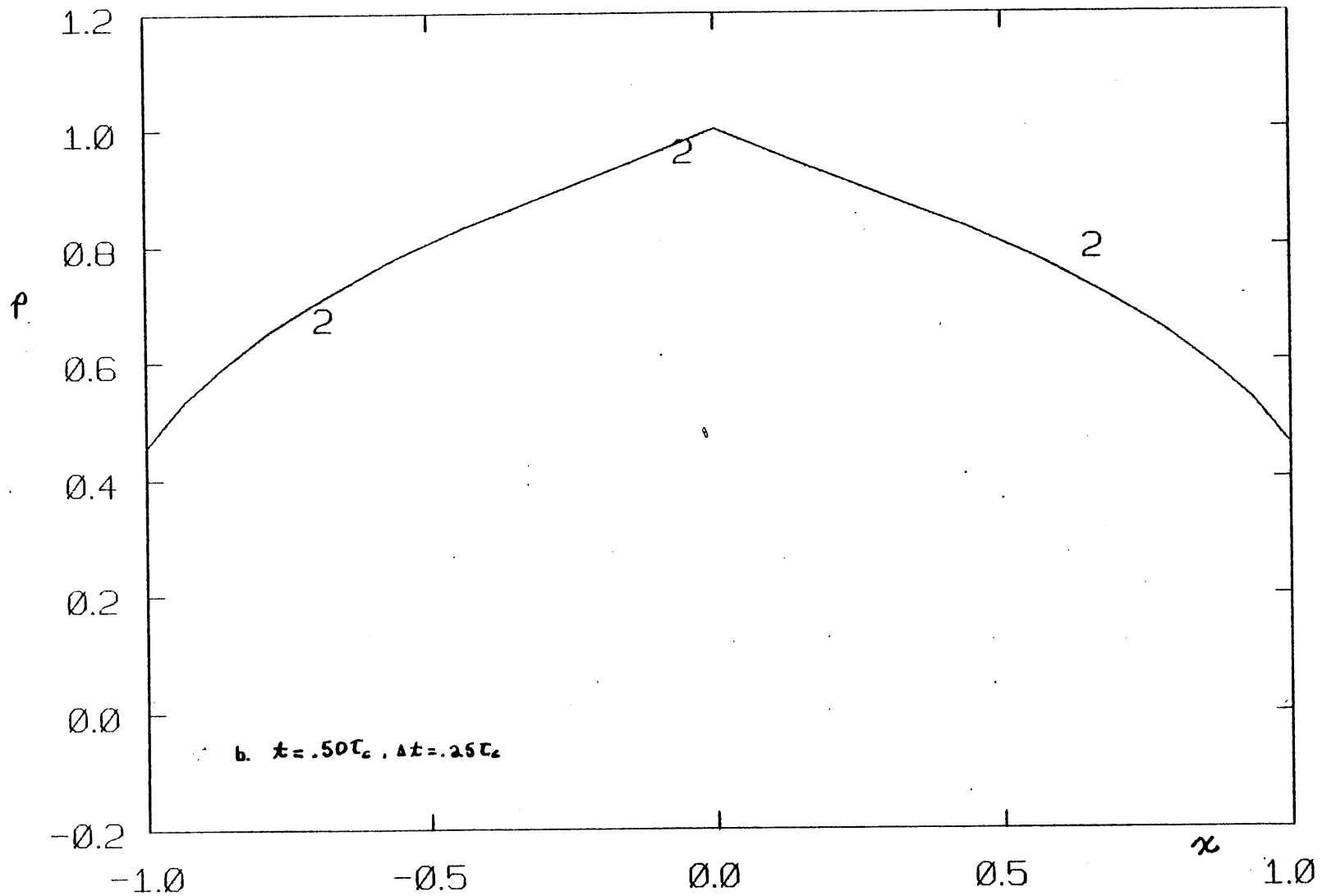
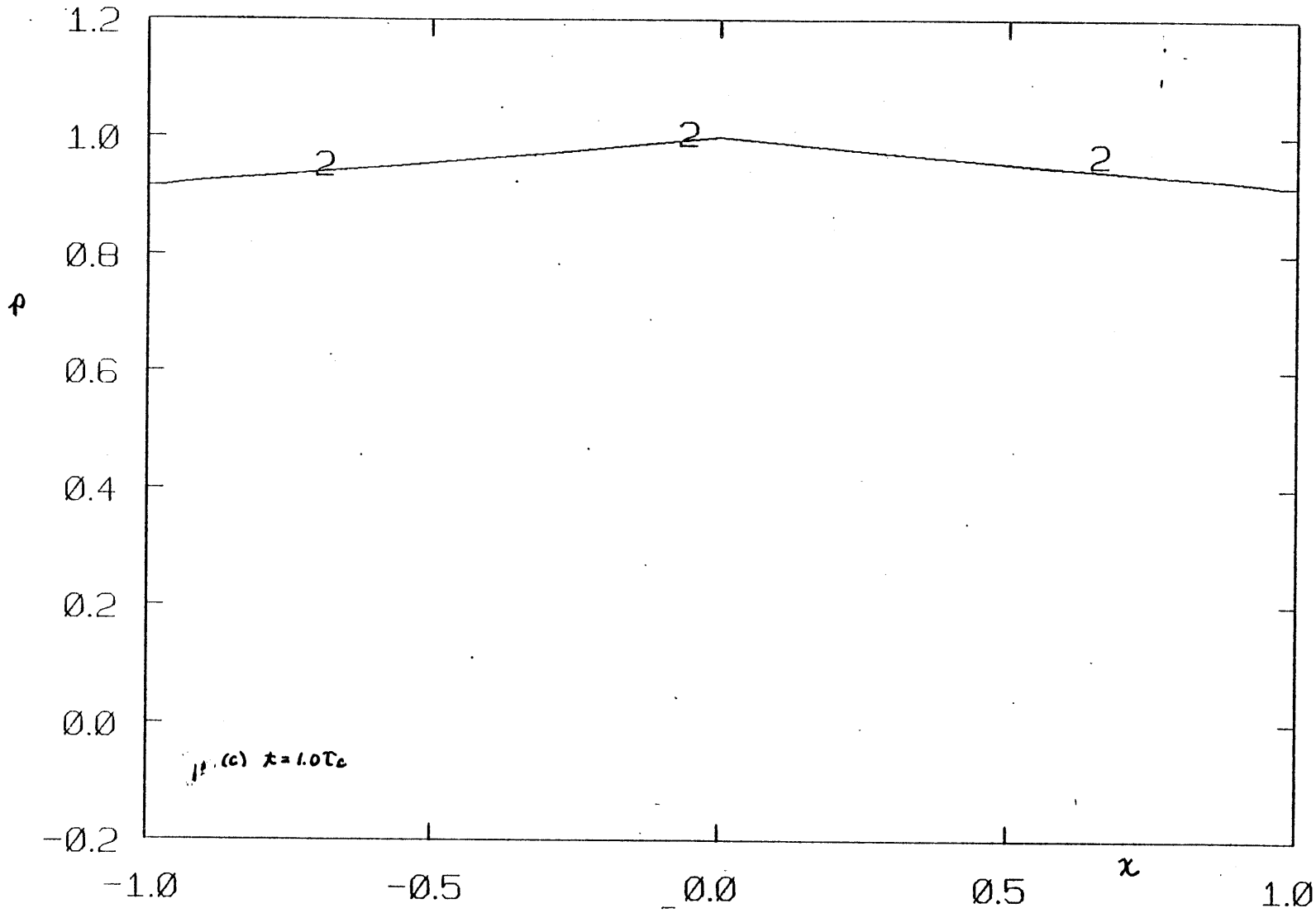
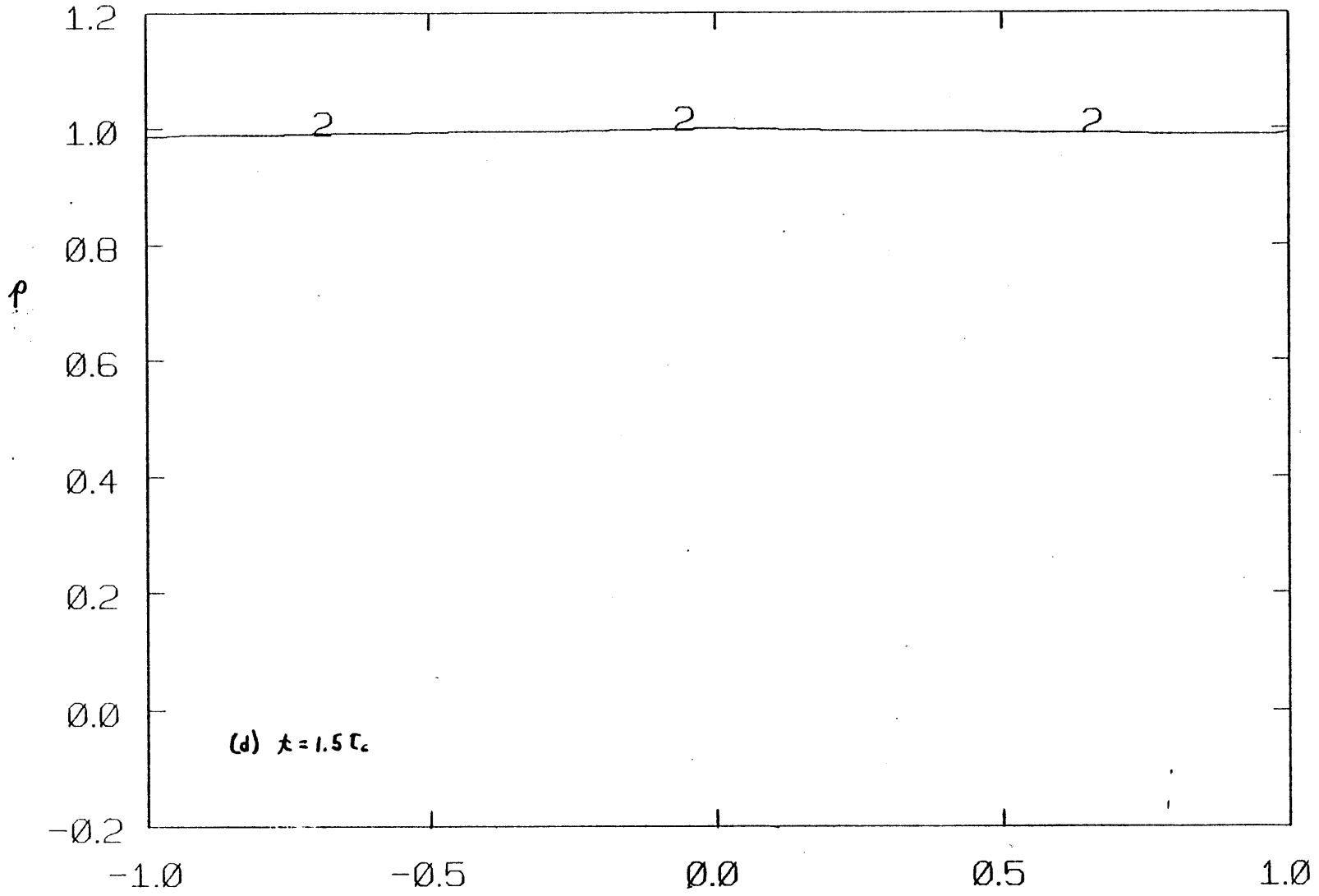
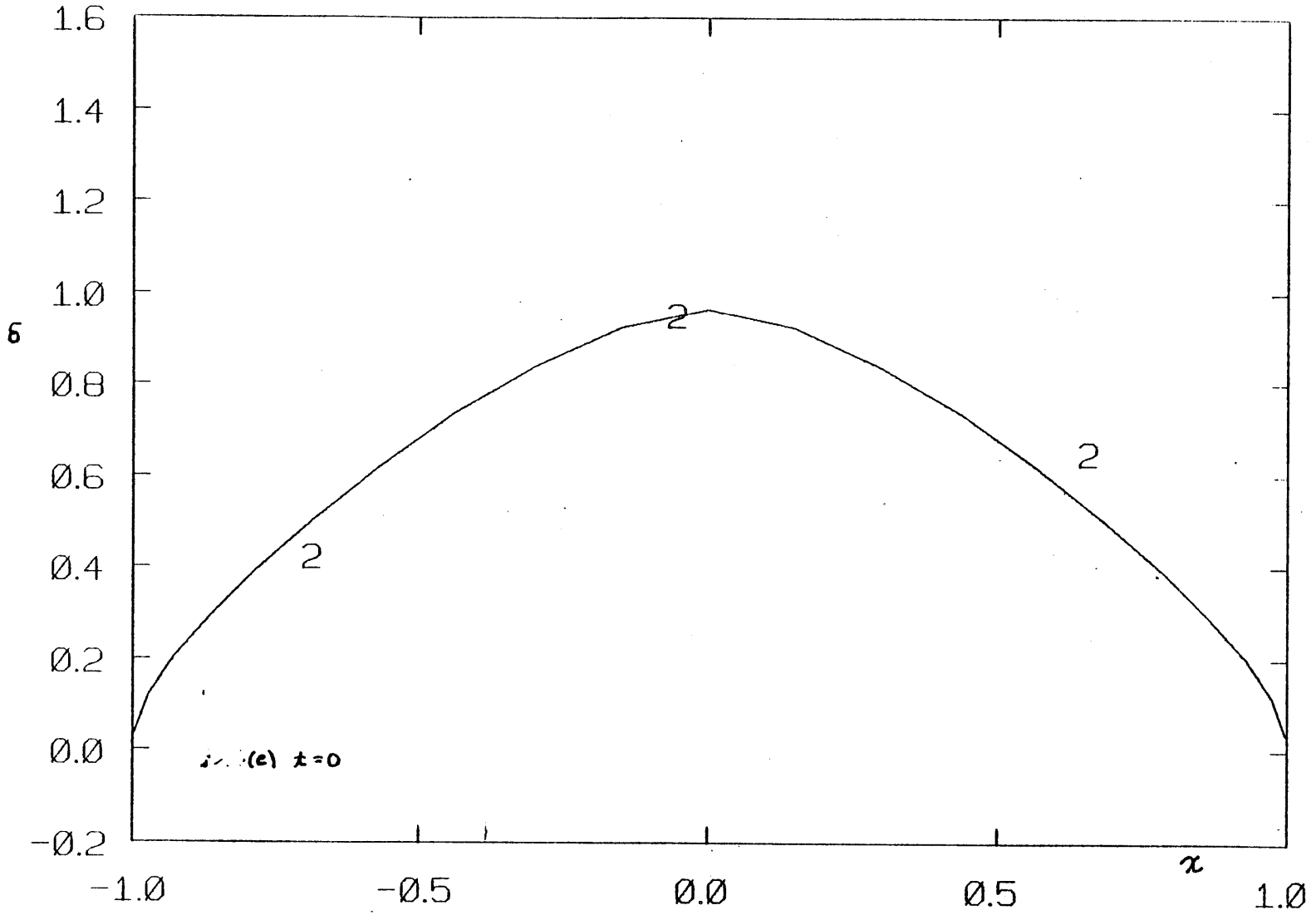


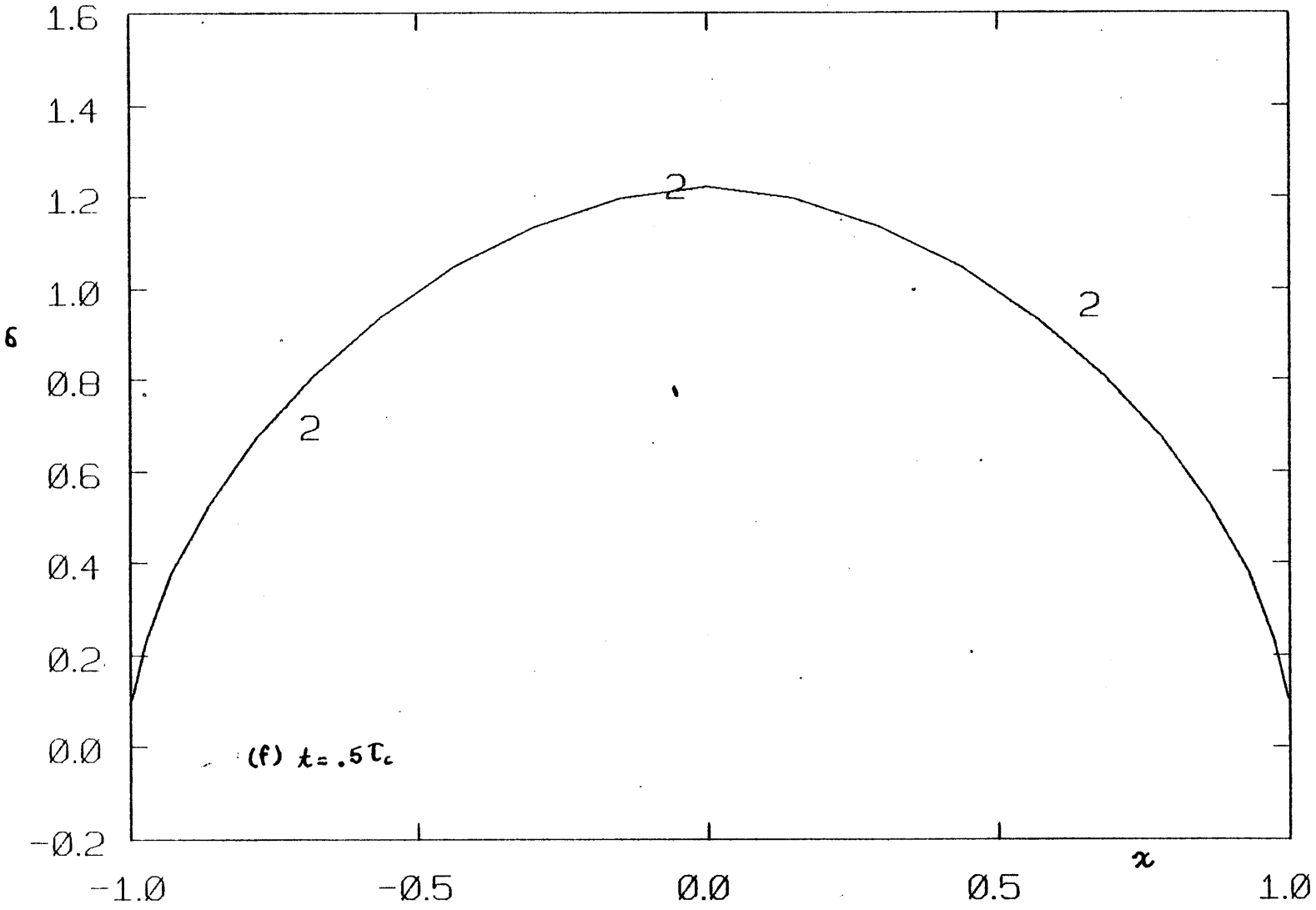
FIG. 3.13. Pressure evolution curves obtained using $\alpha=1$, $\Delta t=.25\tau_c$, $\nu=.3$, $p_o/G=1$. (a)-(d) p ; (e)-(h) crack opening, δ ; (i)-(k) rate of crack opening, $\dot{\delta}$.

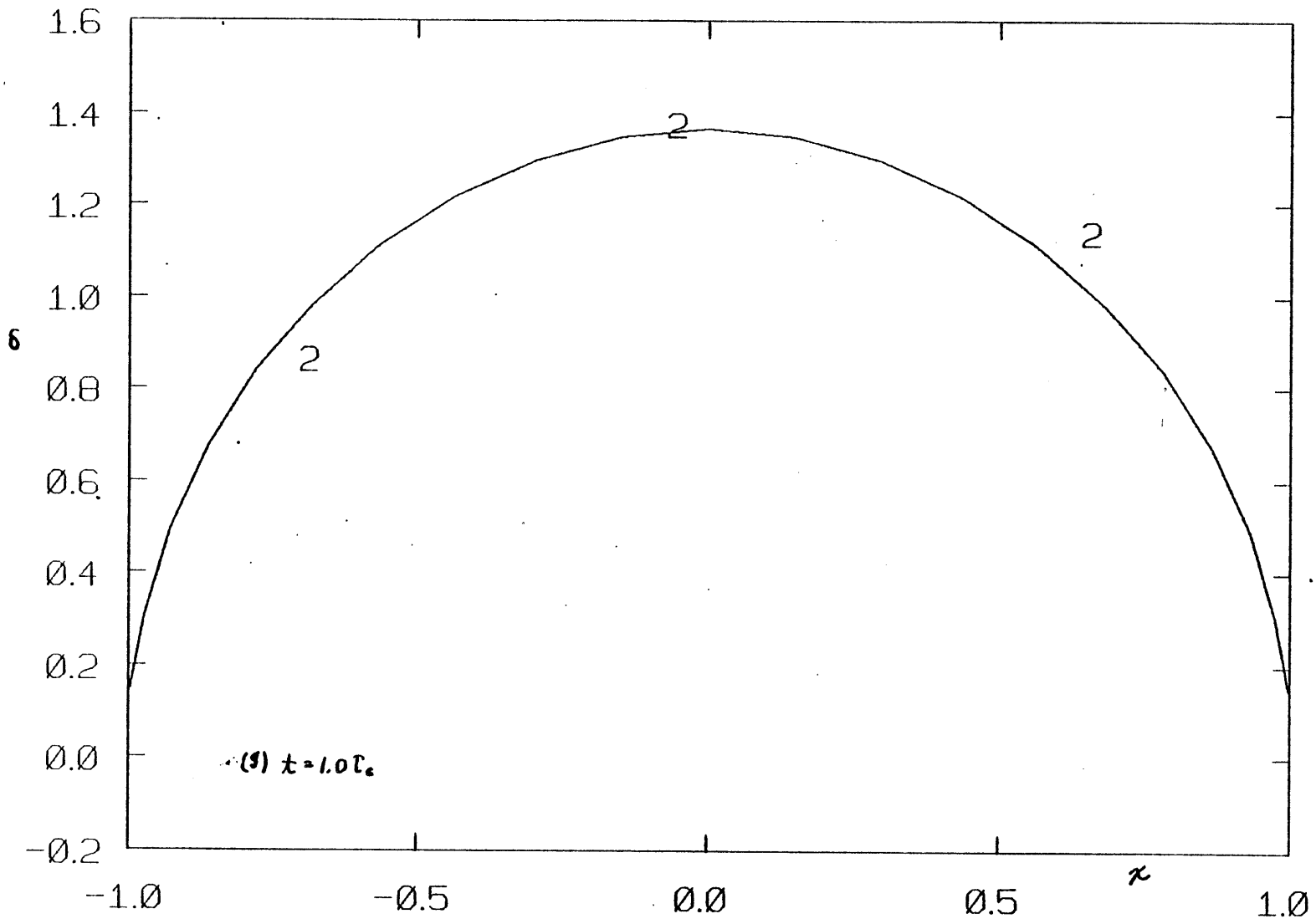

1.2
1.0
Ø.8
Ø.6
Ø.4
Ø.2
Ø.0
-Ø.2
ρ
-1.Ø
-Ø.5
Ø.0
Ø.5
1.Ø
x
2
2
2
b. t = .50τc , Δt = .25τc

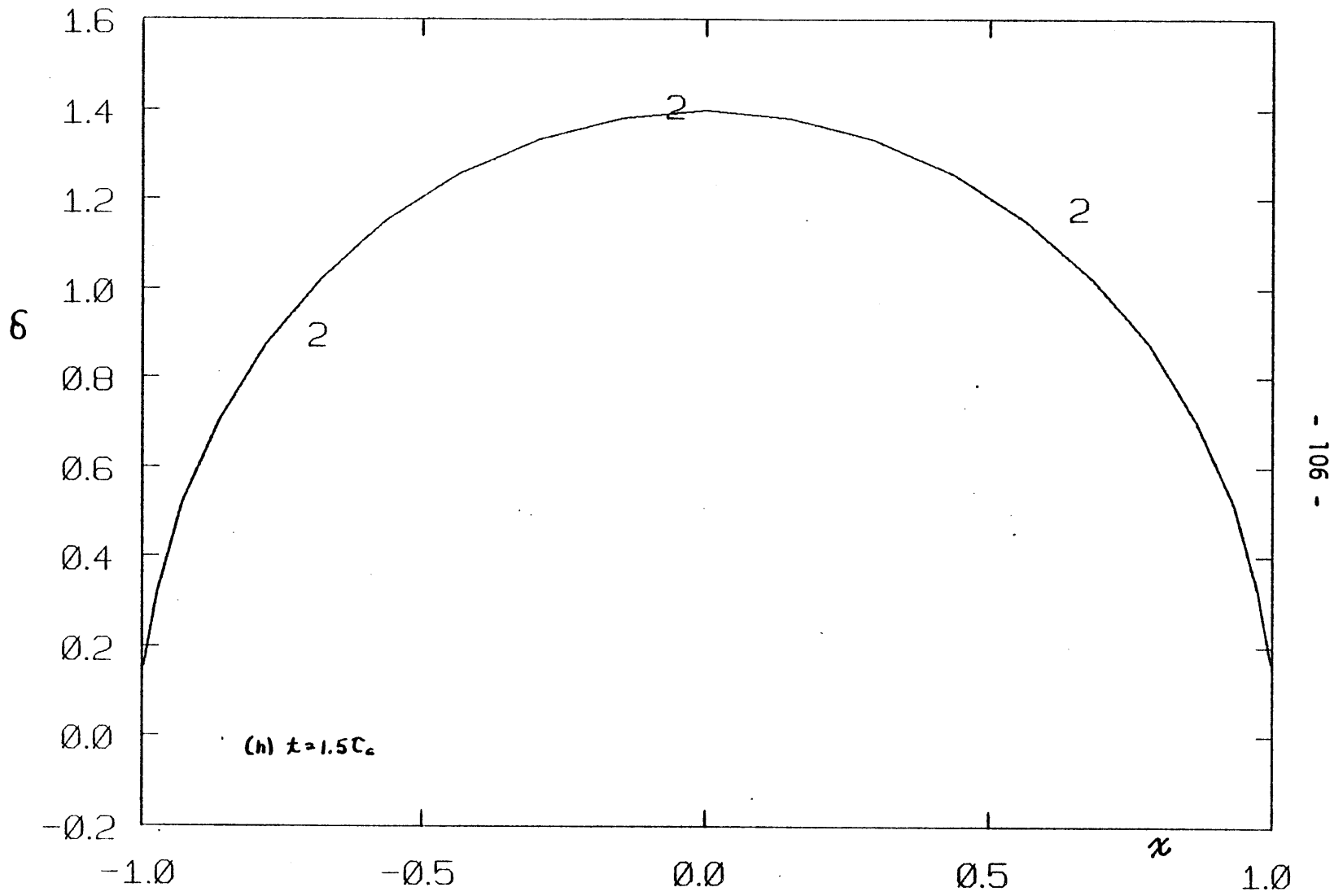

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
2
2
2
(c) t = 1.0τc
-1.0
-0.5
0.0
0.5
1.0
x

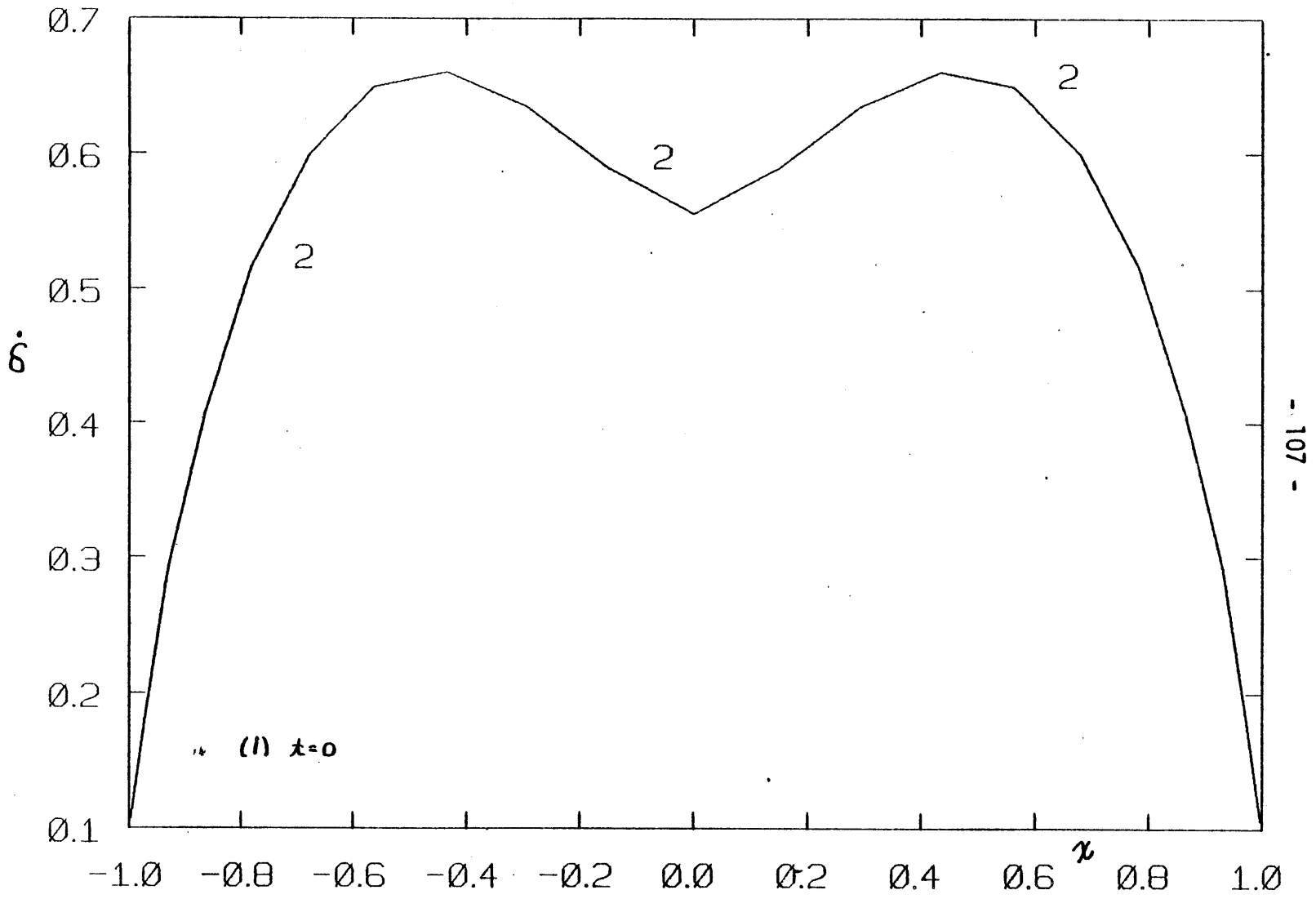

1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
ρ
2
2
2
(d) t = 1.5 τc
−1.0
−0.5
0.0
0.5
1.0

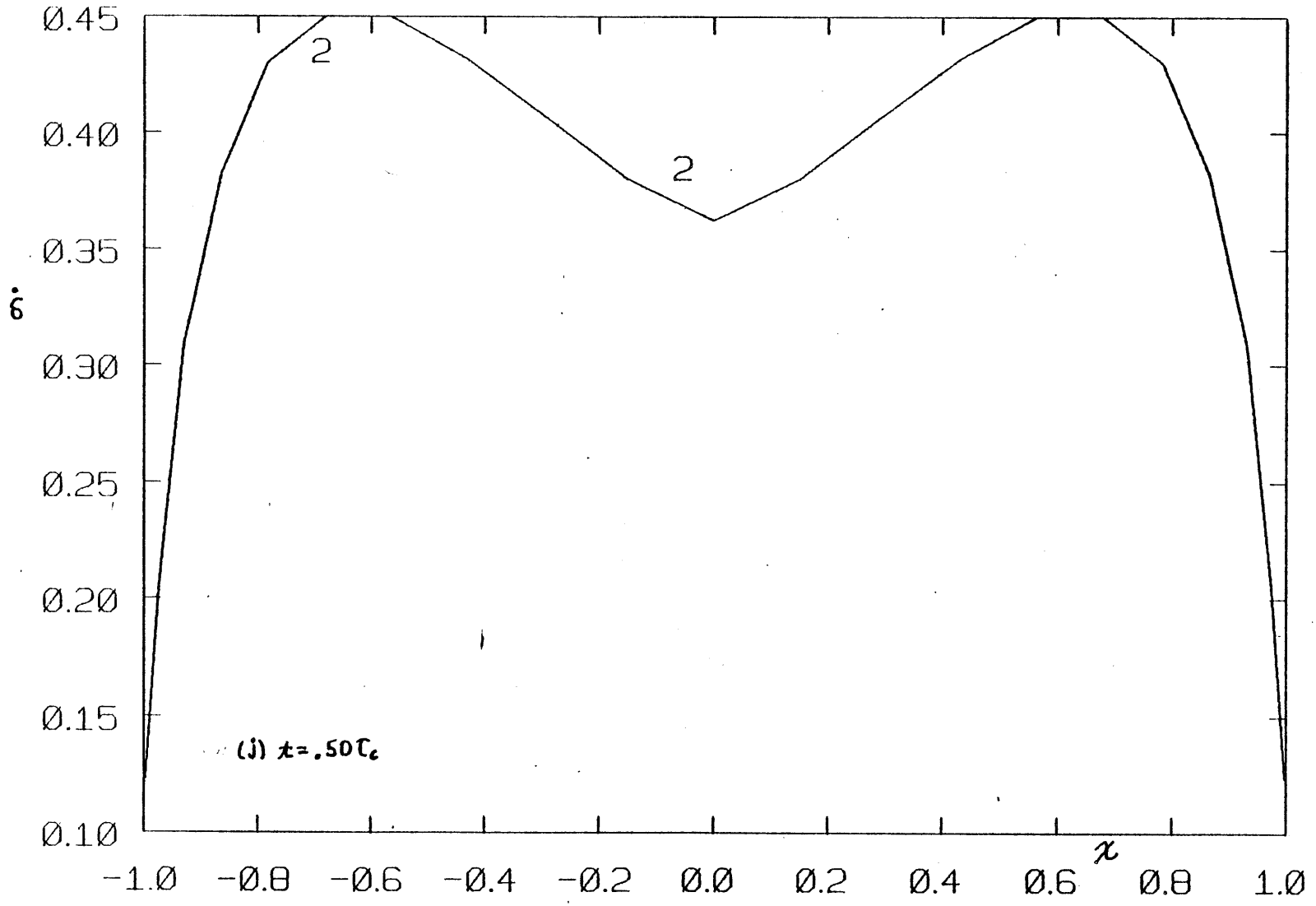

1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
2
2
2
(c) t = 0
-1.0
-0.5
0.0
0.5
1.0
x

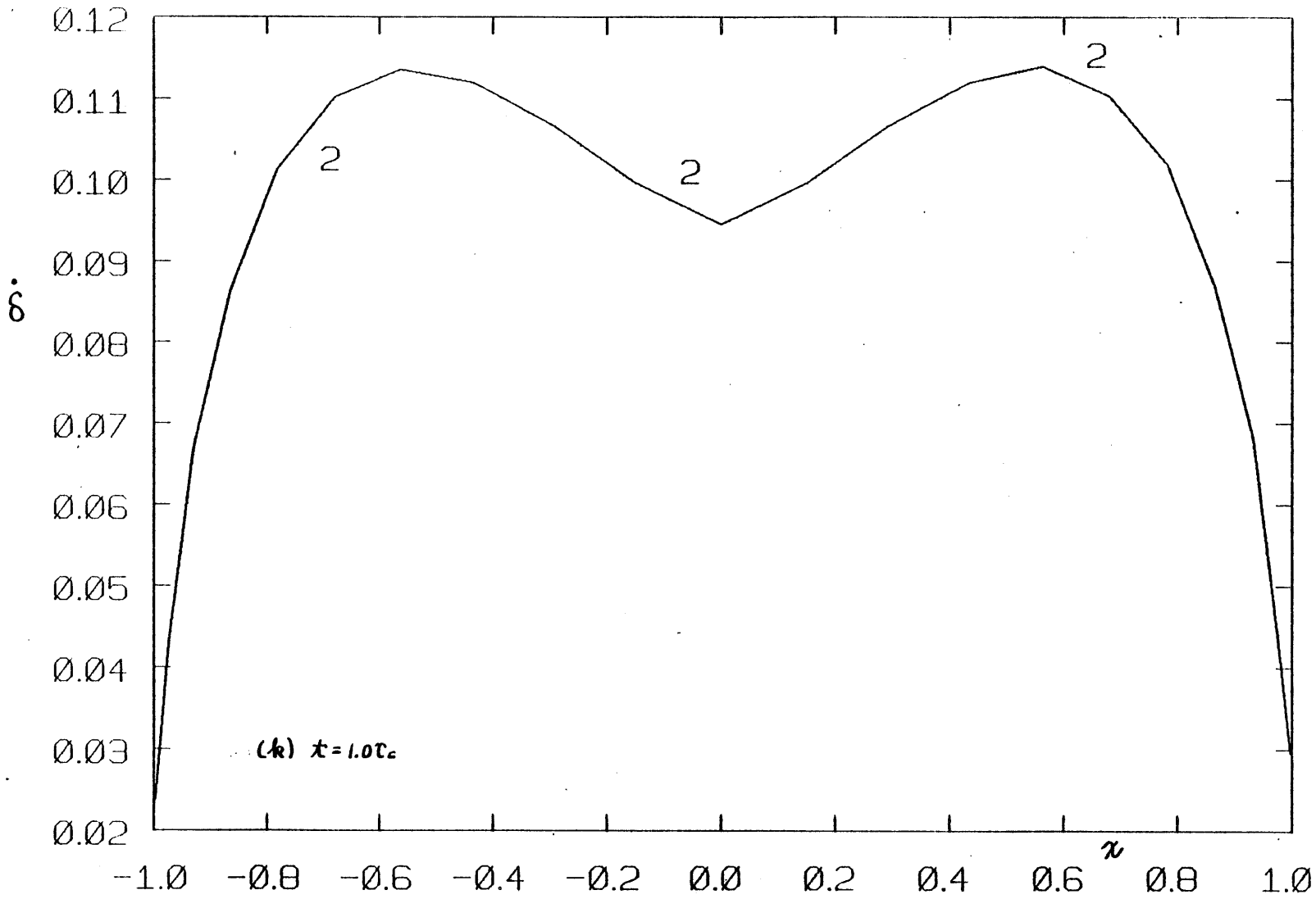

1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(f) t = .5 τc


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(1) t = 1.0 τc


1.6
1.4
1.2
1.0
Ø.8
Ø.6
Ø.4
Ø.2
Ø.0
-Ø.2
δ
2
2
2
(h) t=1.5C₆
-1.Ø
-Ø.5
Ø.Ø
Ø.5
1.Ø
x


0.7
0.6
0.5
0.4
0.3
0.2
0.1
δ̇
2
2
2
(1) t=0
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
χ


0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
$\dot{\varepsilon}$
2
2
(j) $t=.50\tau_c$
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x


0.12
0.11
0.10
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
δ̇
2
2
2
(k) t = 1.0τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x

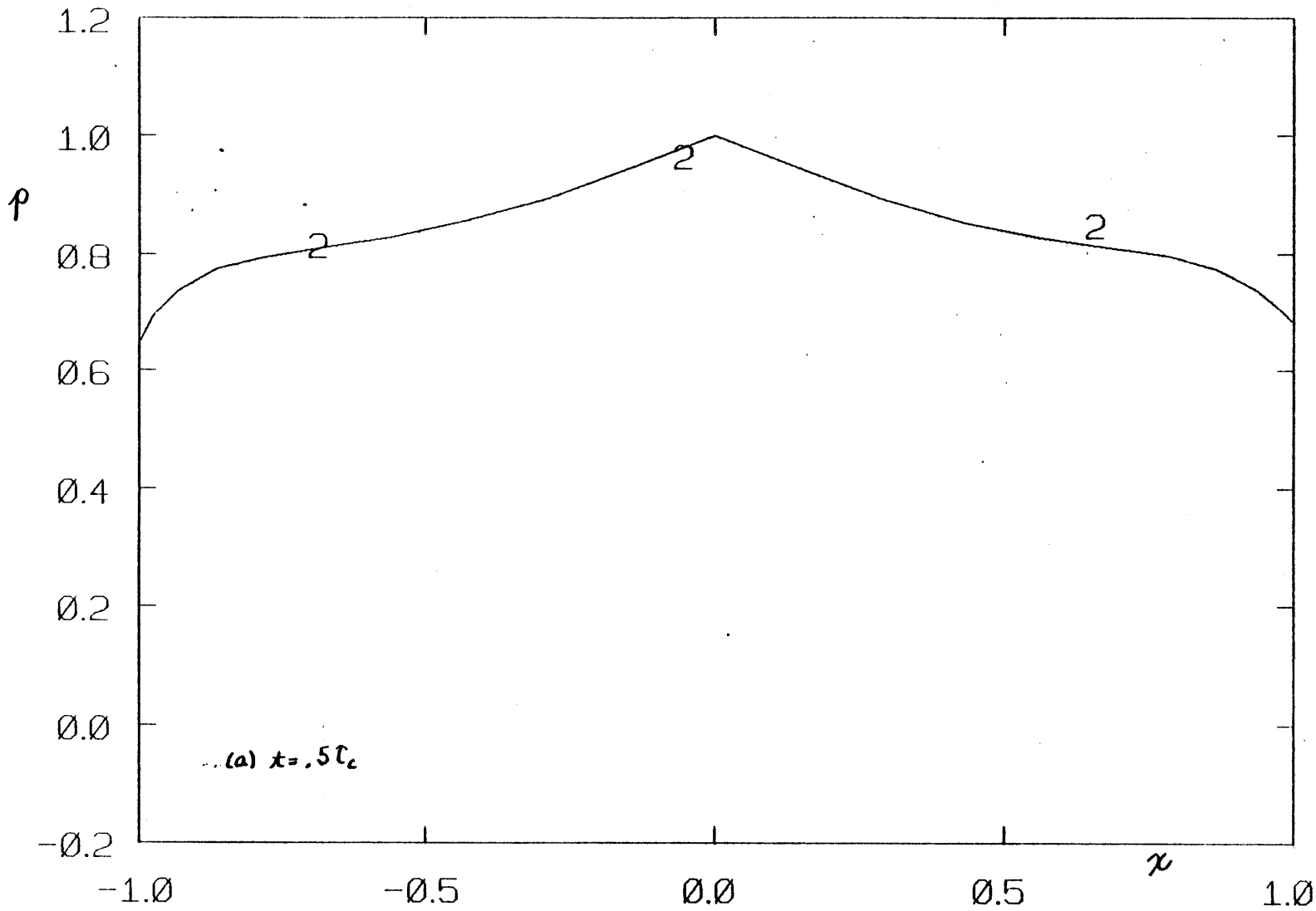
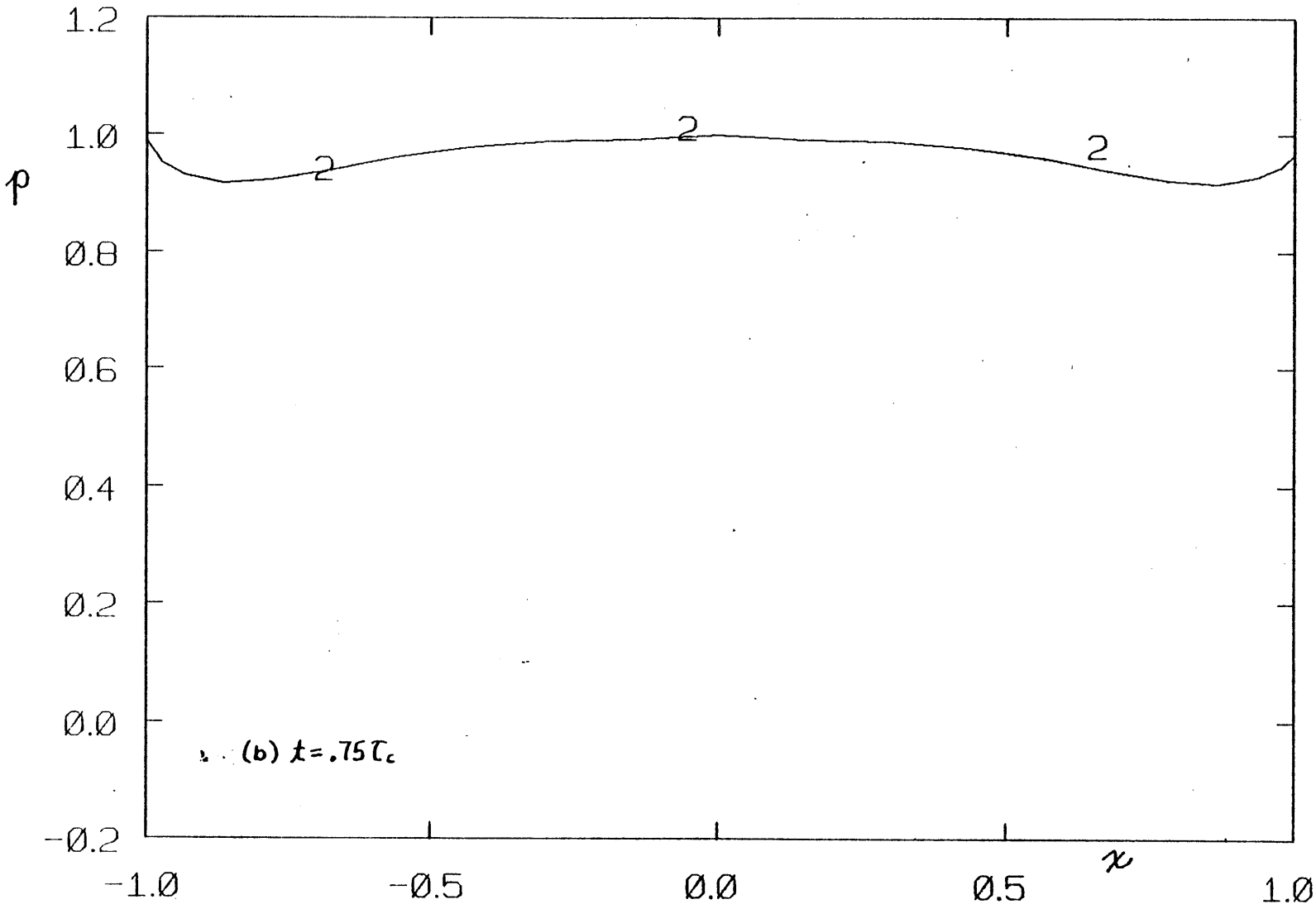
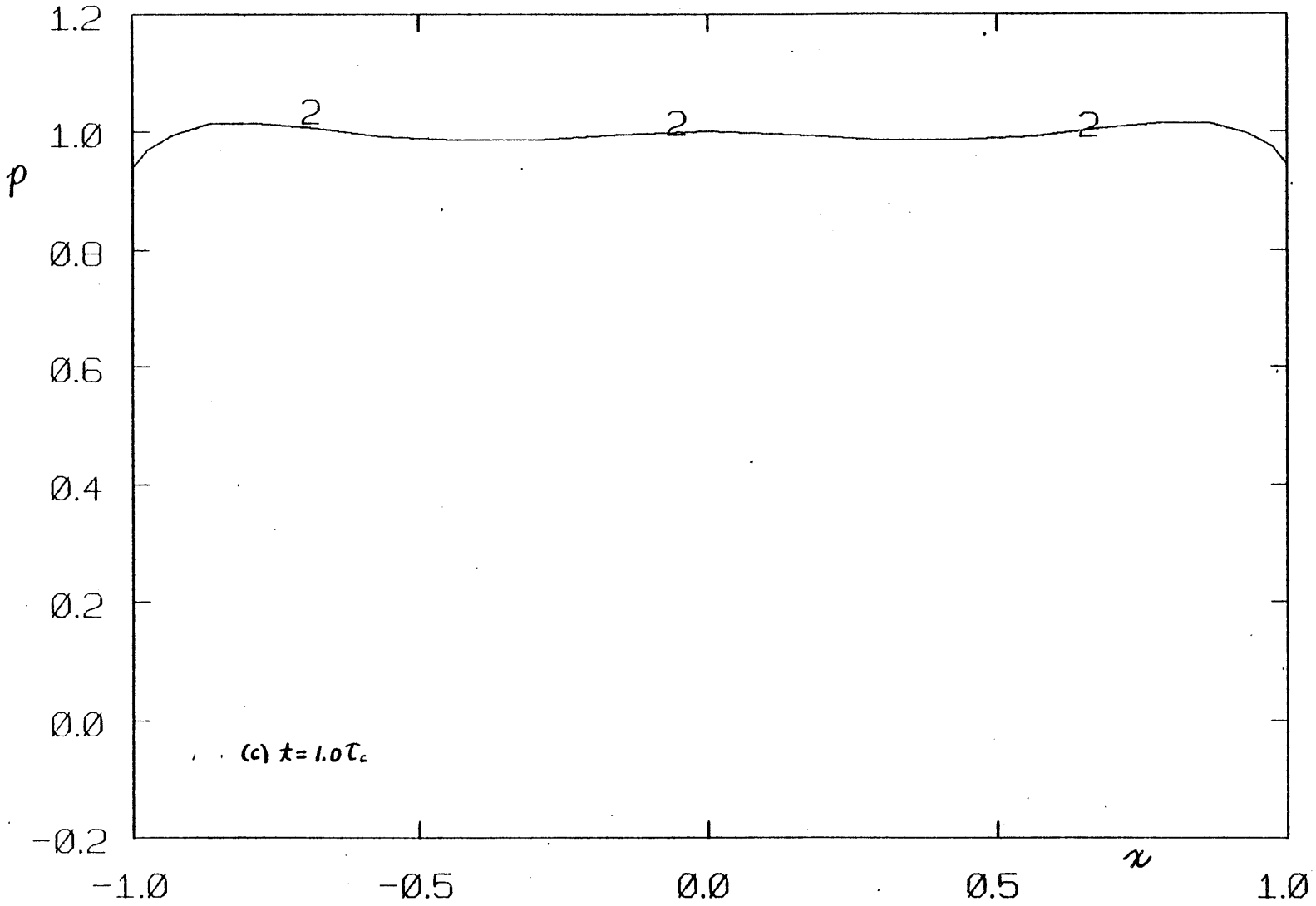
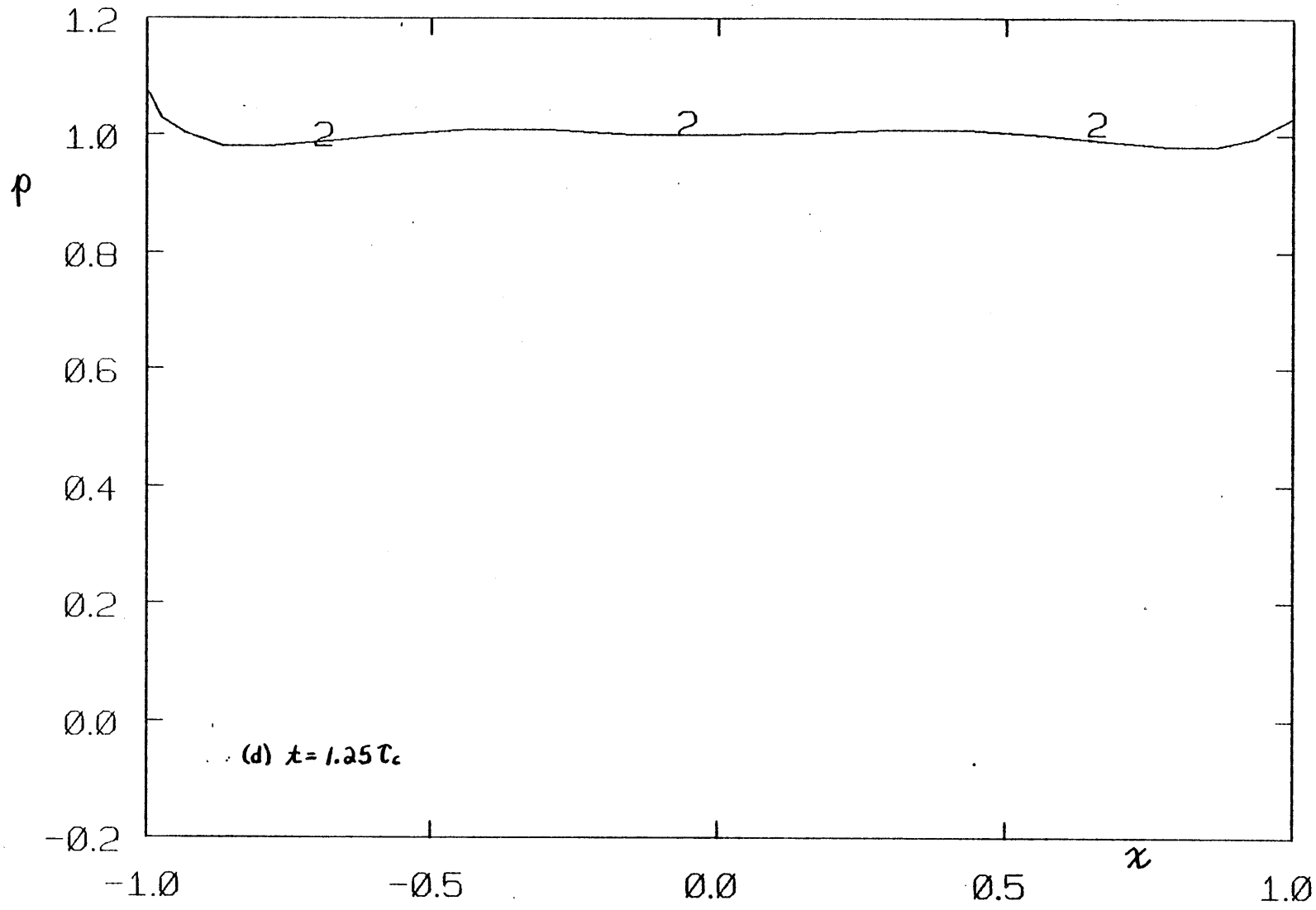
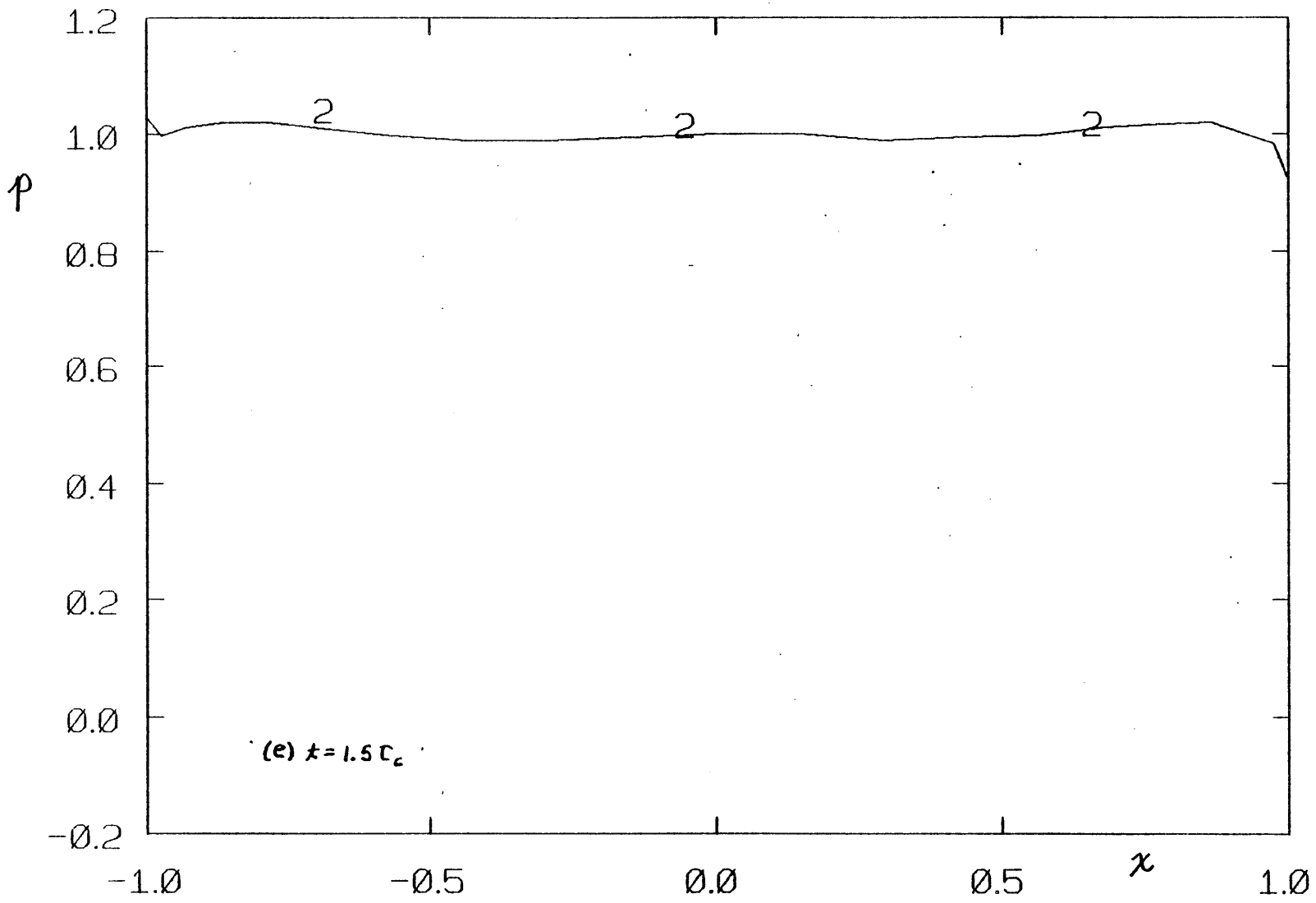


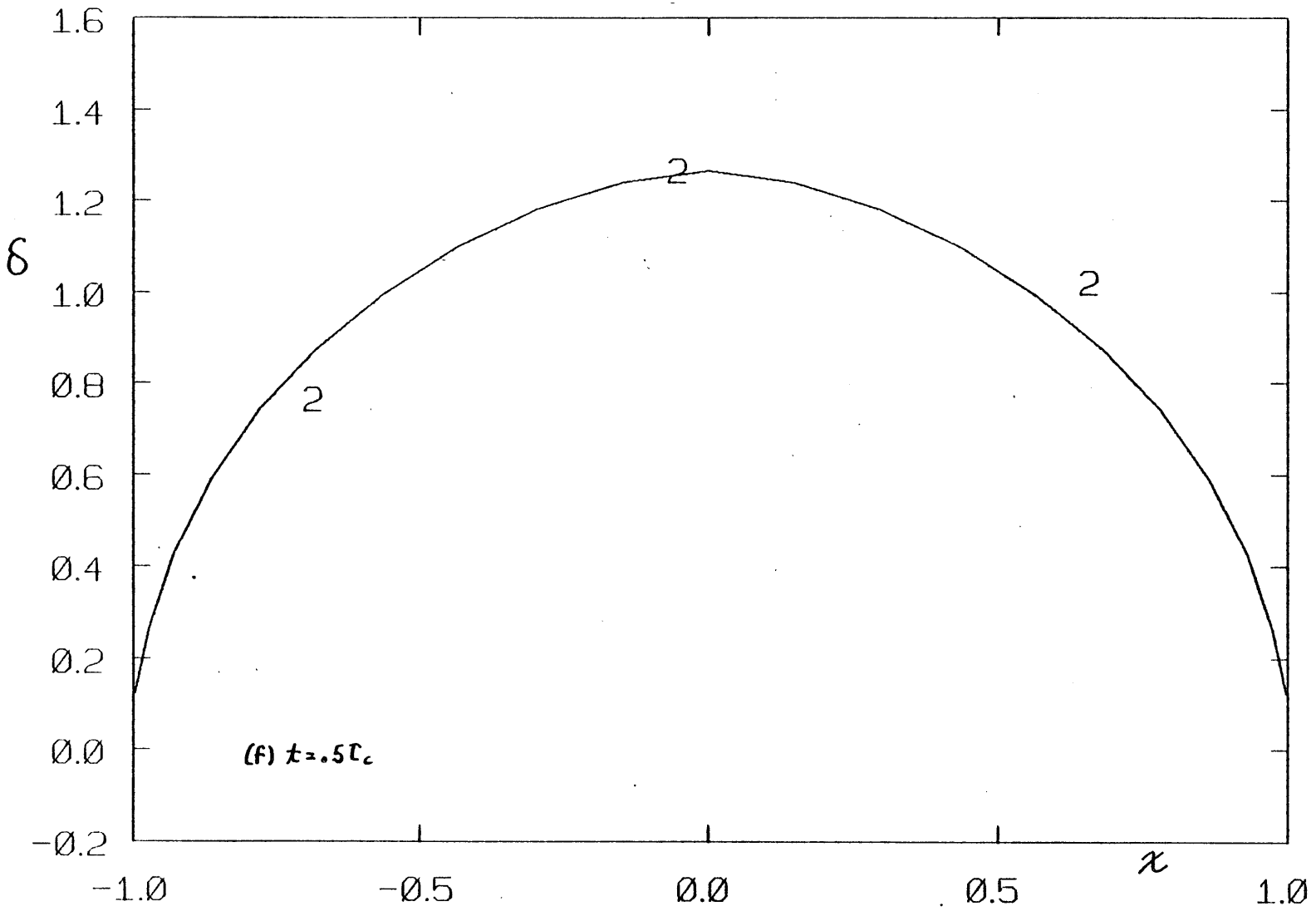
FIG. 3.14. Pressure evolution curves obtained under the same conditions as those in fig. 3.13. (i.e., $\Delta t = .25\tau_c$, same elastic constants and initial pressure distribution), but with $\alpha = .5$. Note the instability in p and $\dot{\delta}$, manifested in the oscillating pressure gradients at the crack tips. (a)-(e) p; (f)-(h) δ; (i),(j) $\dot{\delta}$.

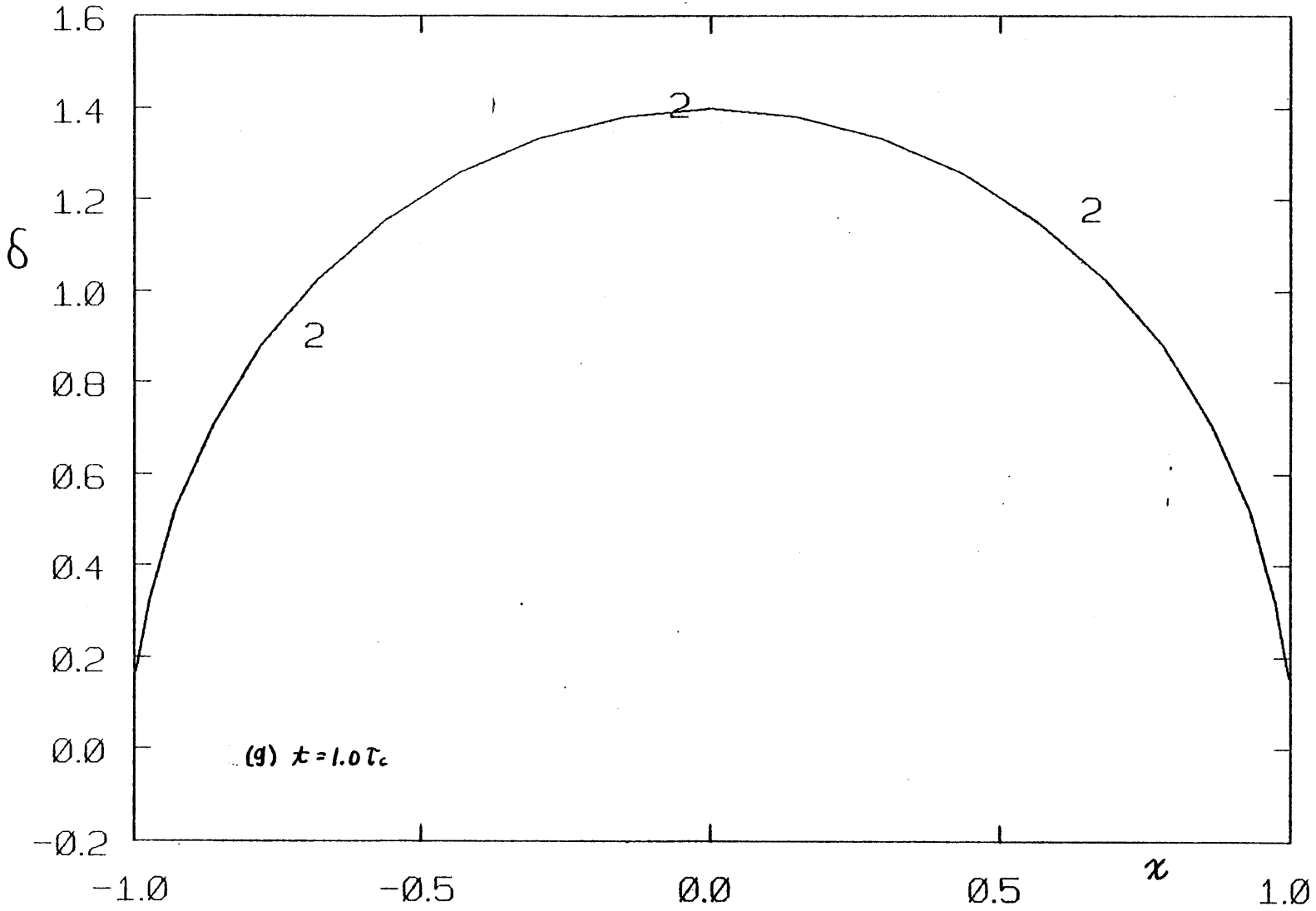

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
p
2
2
2
(b) $t = .75\tau_c$
-1.0
-0.5
0.0
0.5
1.0
x

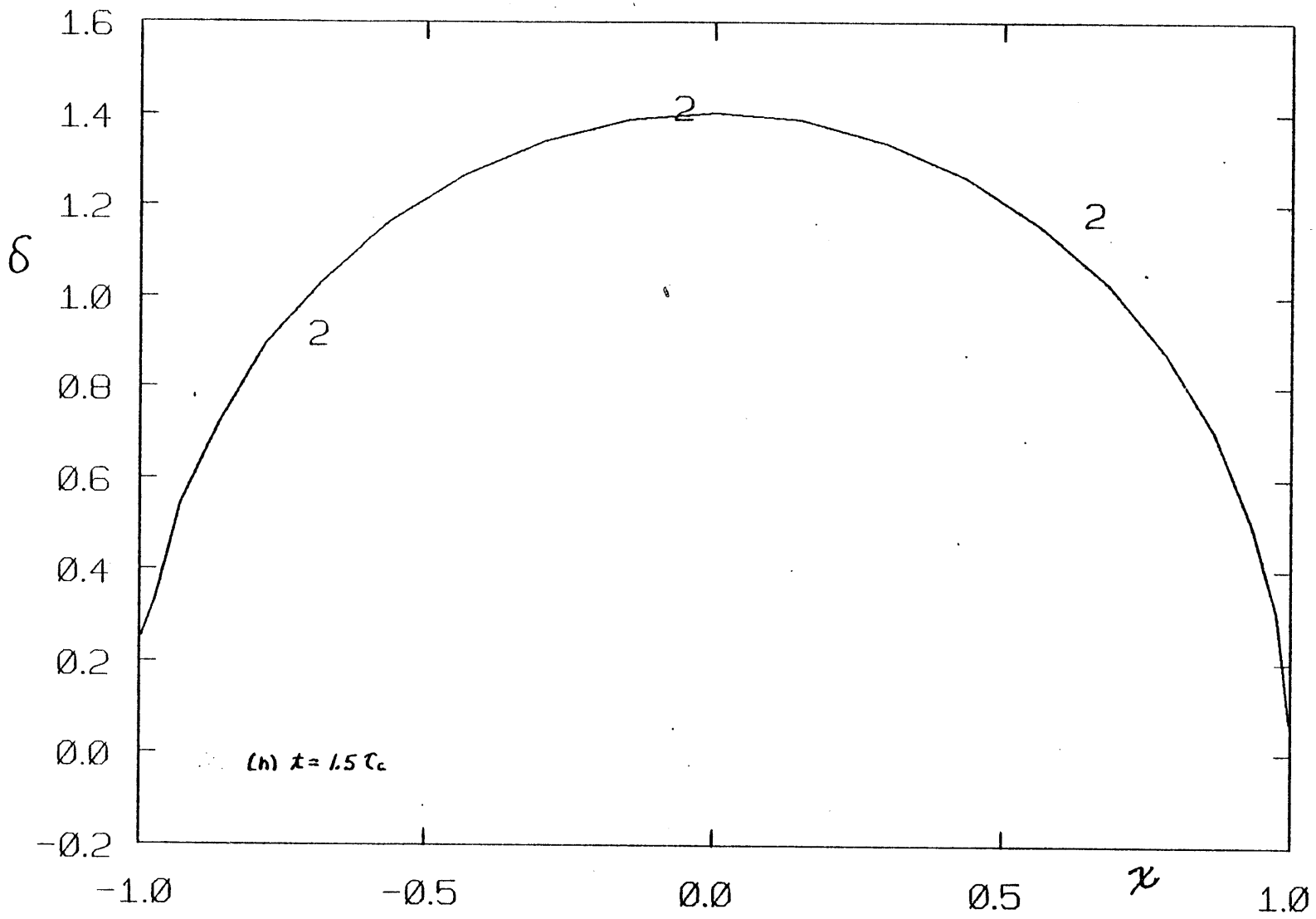

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
2
2
2
(c) t = 1.0 τc
-1.0
-0.5
0.0
0.5
1.0
x

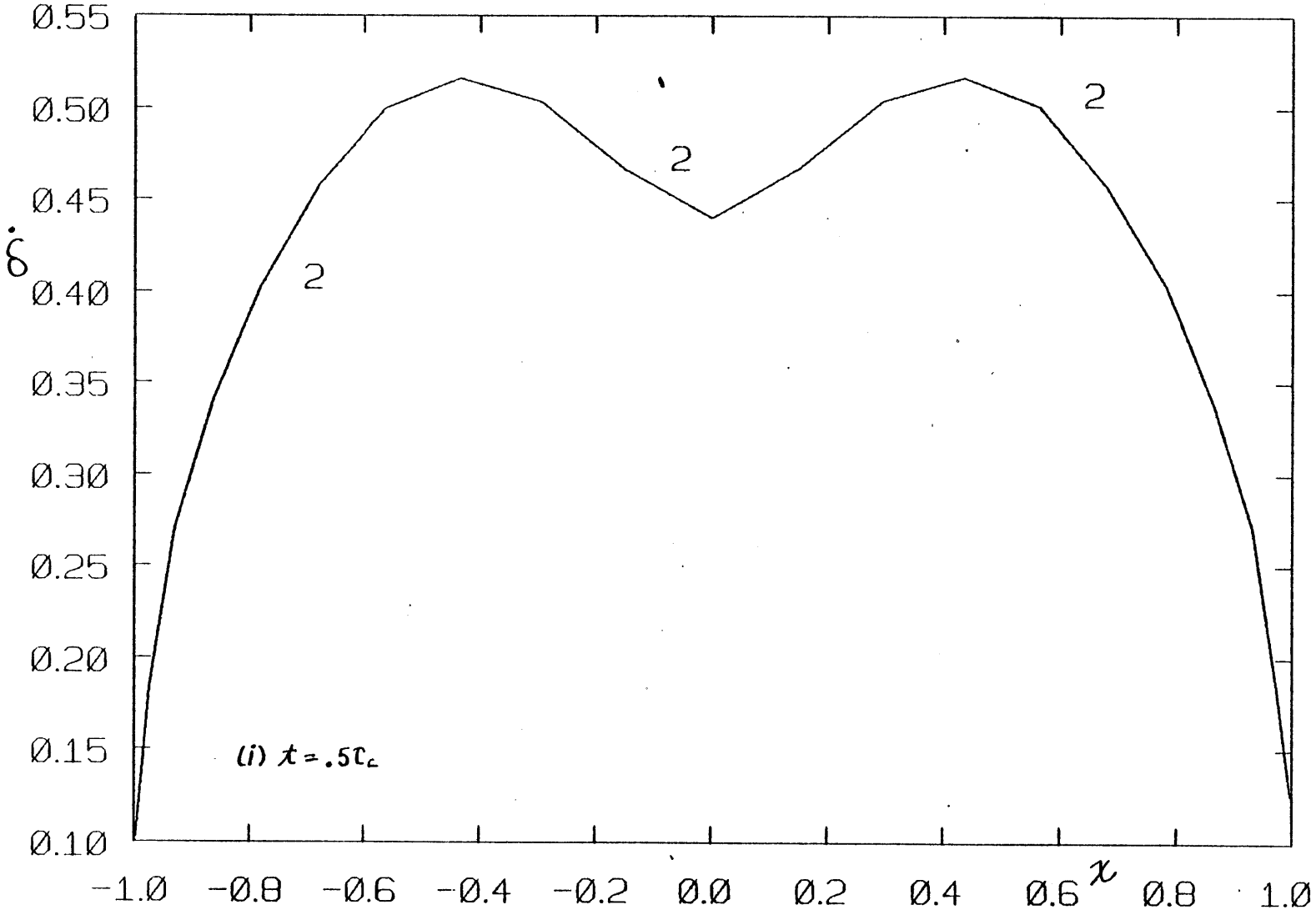

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
2
2
2
(d) t = 1.25 τc
-1.0
-0.5
0.0
0.5
1.0
x

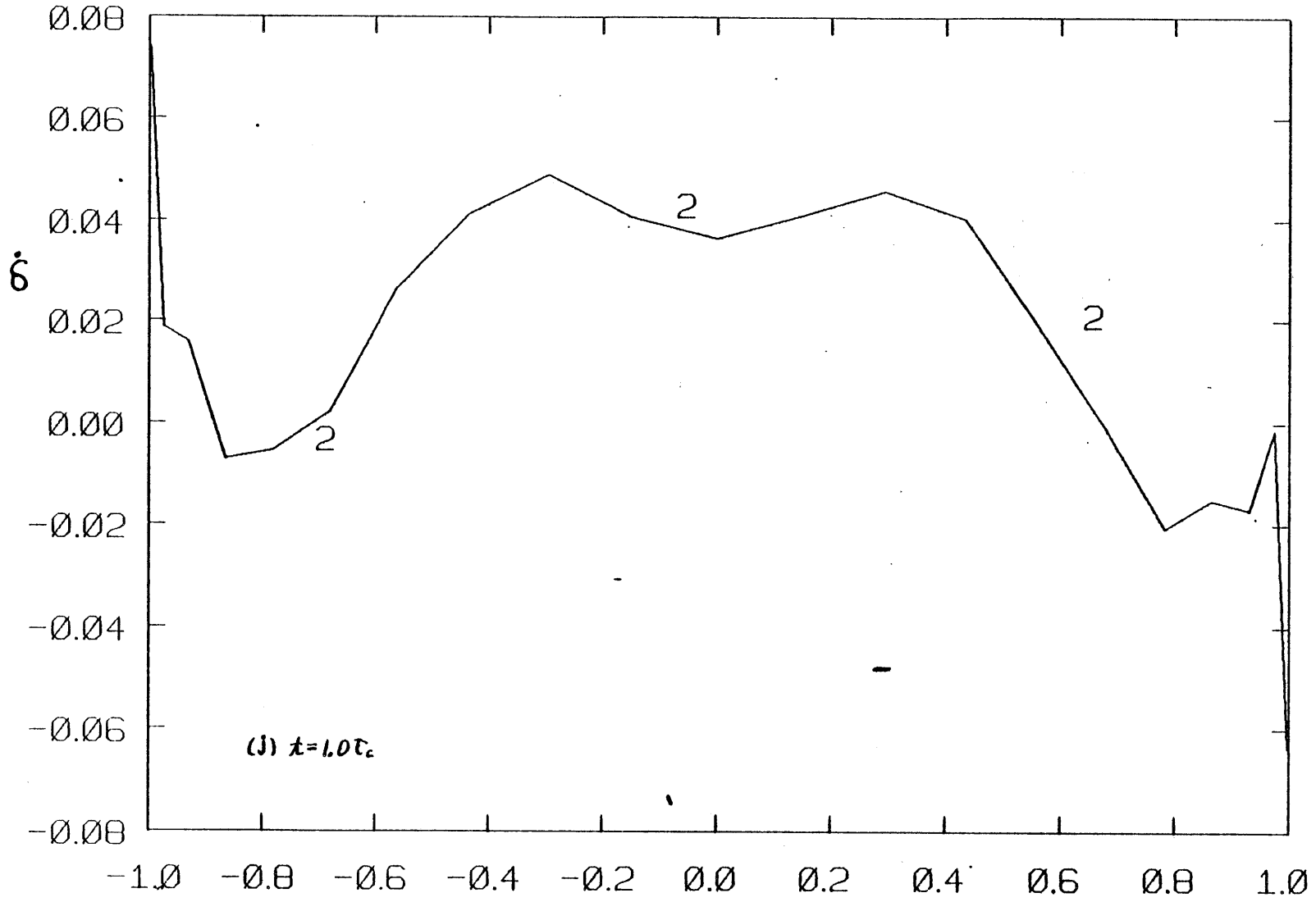

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
2
2
2
(e) t = 1.5 τc
-1.0
-0.5
0.0
0.5
1.0
x


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
2
2
2
(f) t=.5τc
-1.0
-0.5
0.0
0.5
1.0
x


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(g) t = 1.0 τc


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(h) t = 1.5 τc


0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
δ̇
2
2
2
(i) t = .5τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x


0.08
0.06
0.04
0.02
0.00
-0.02
-0.04
-0.06
-0.08
δ̇
2
2
2
(J) t=1.0τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0

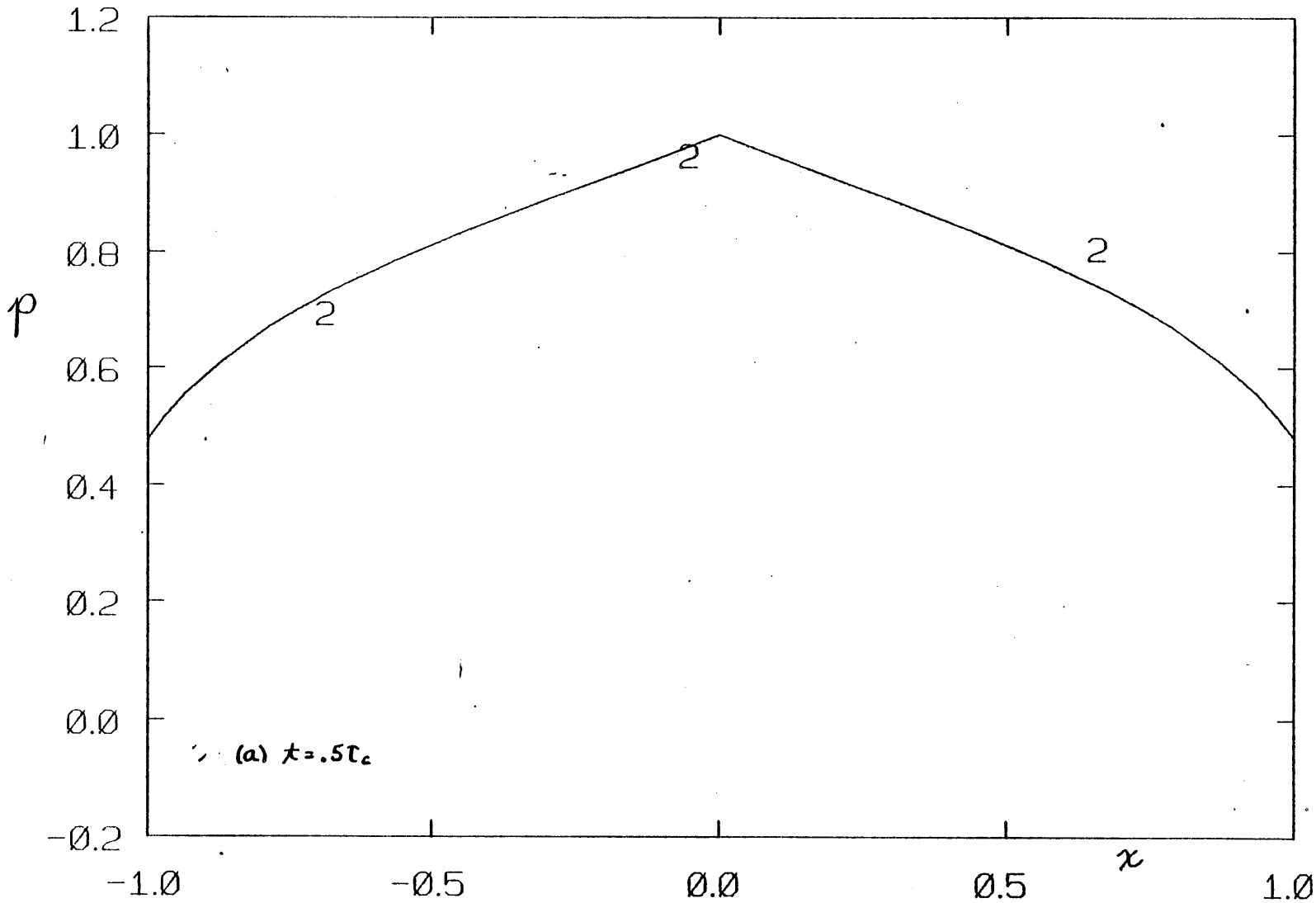
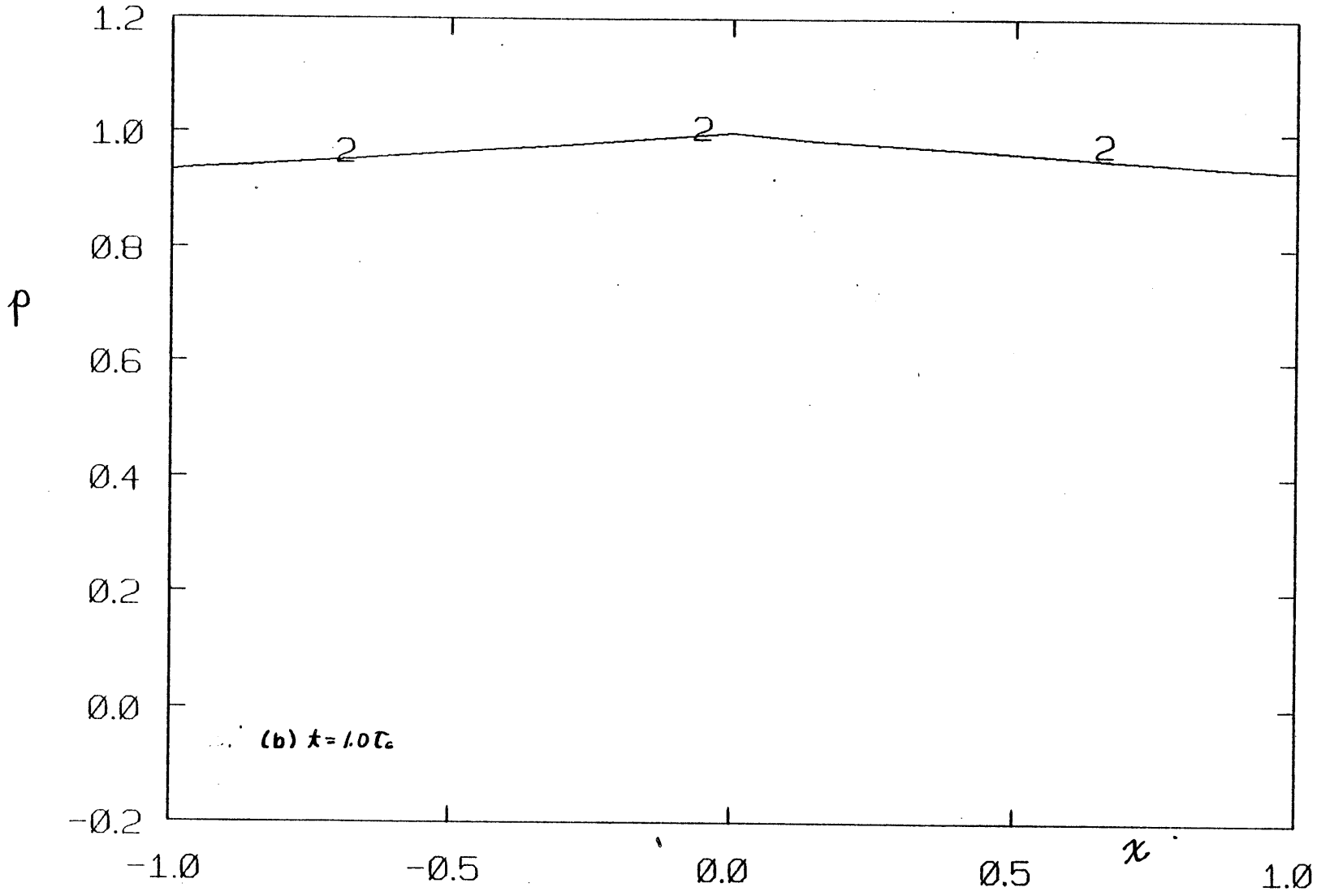
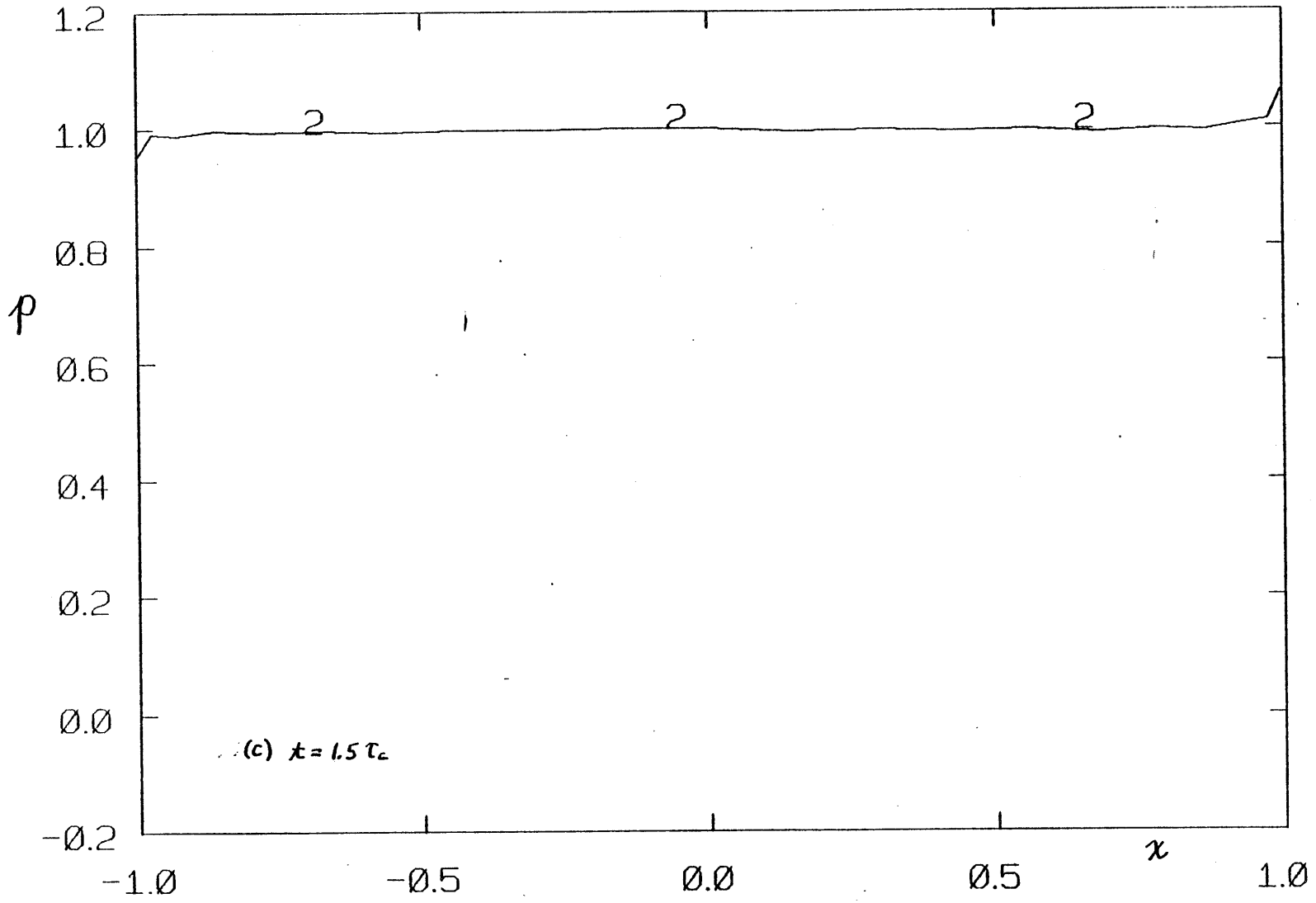
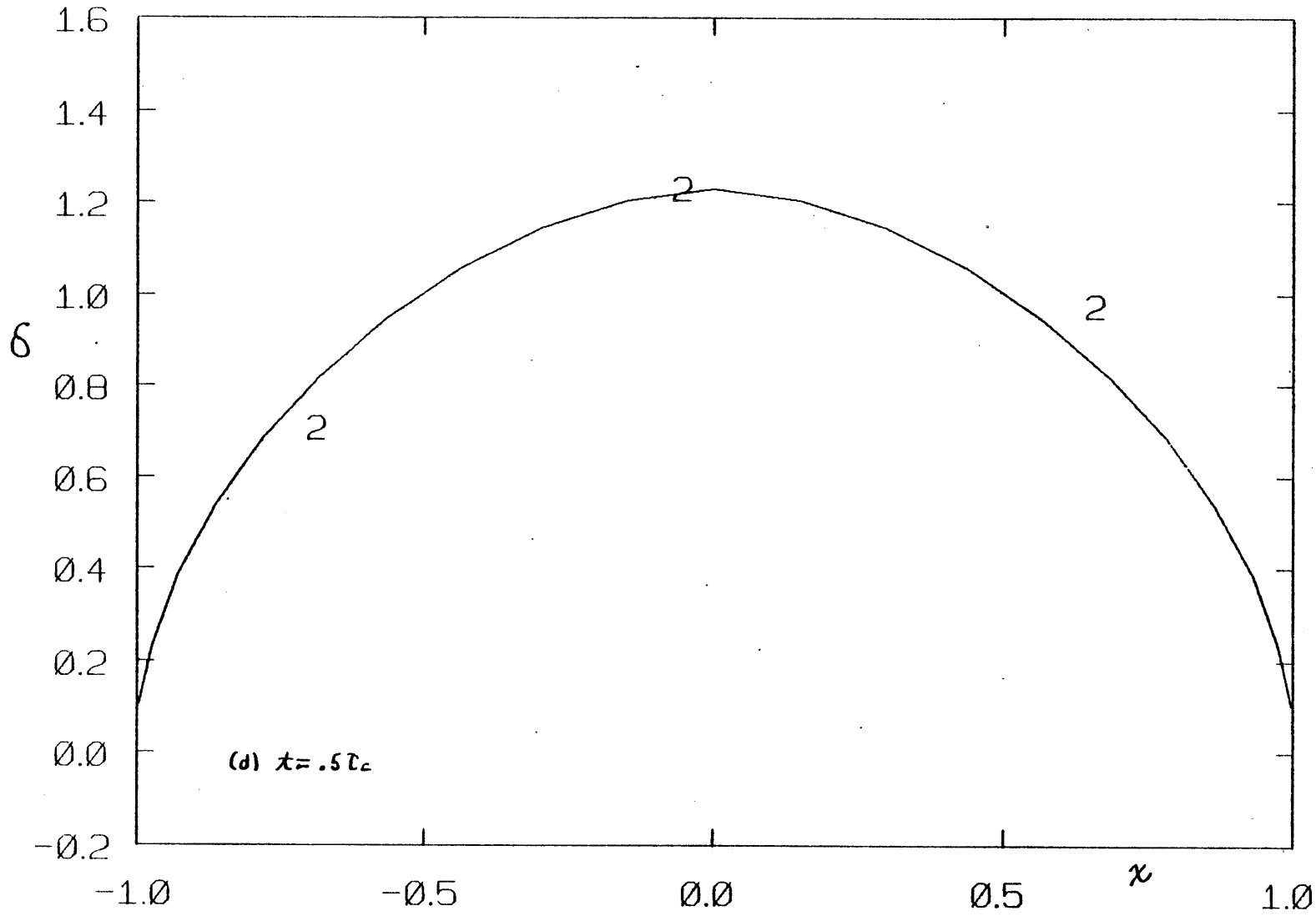
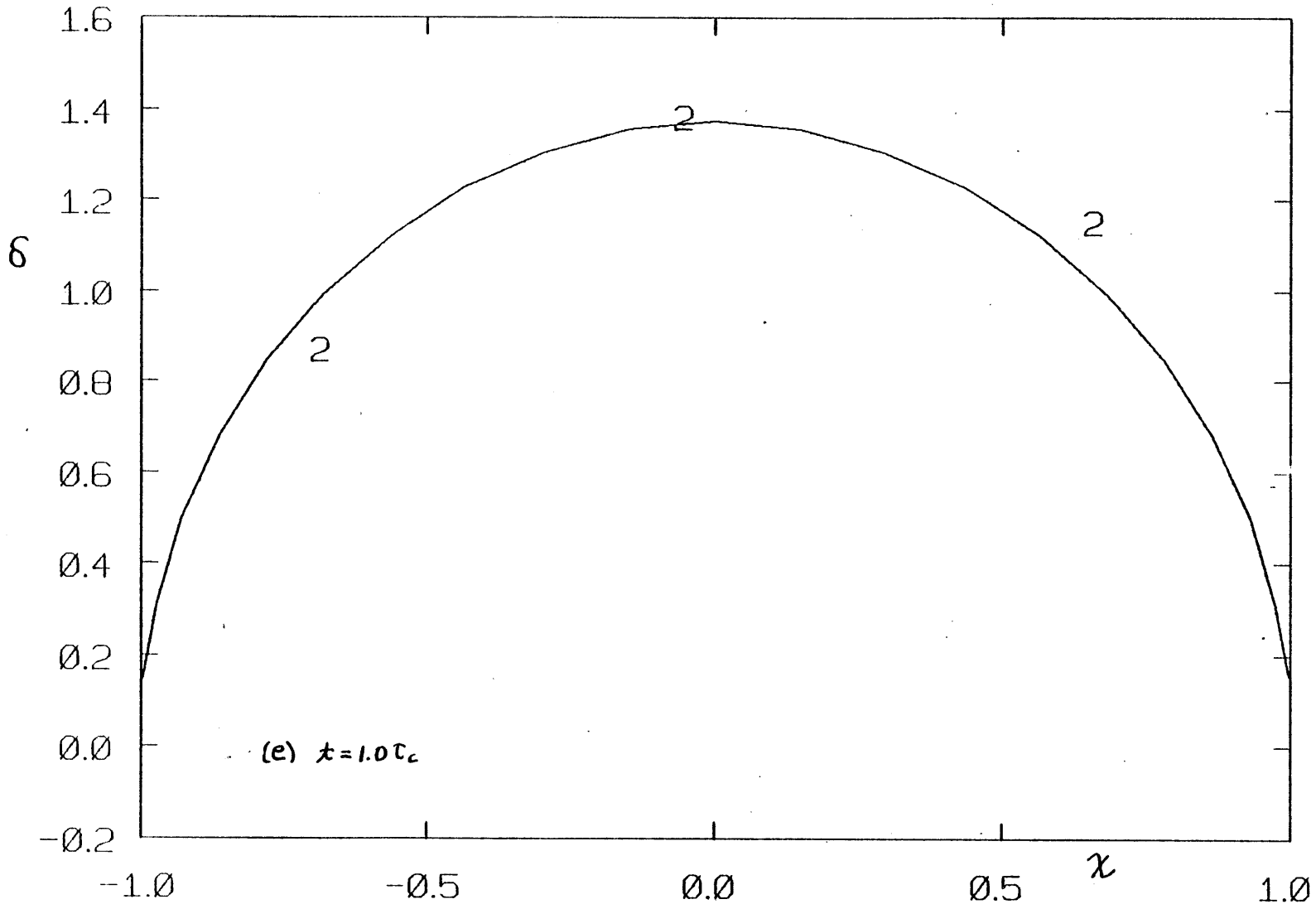


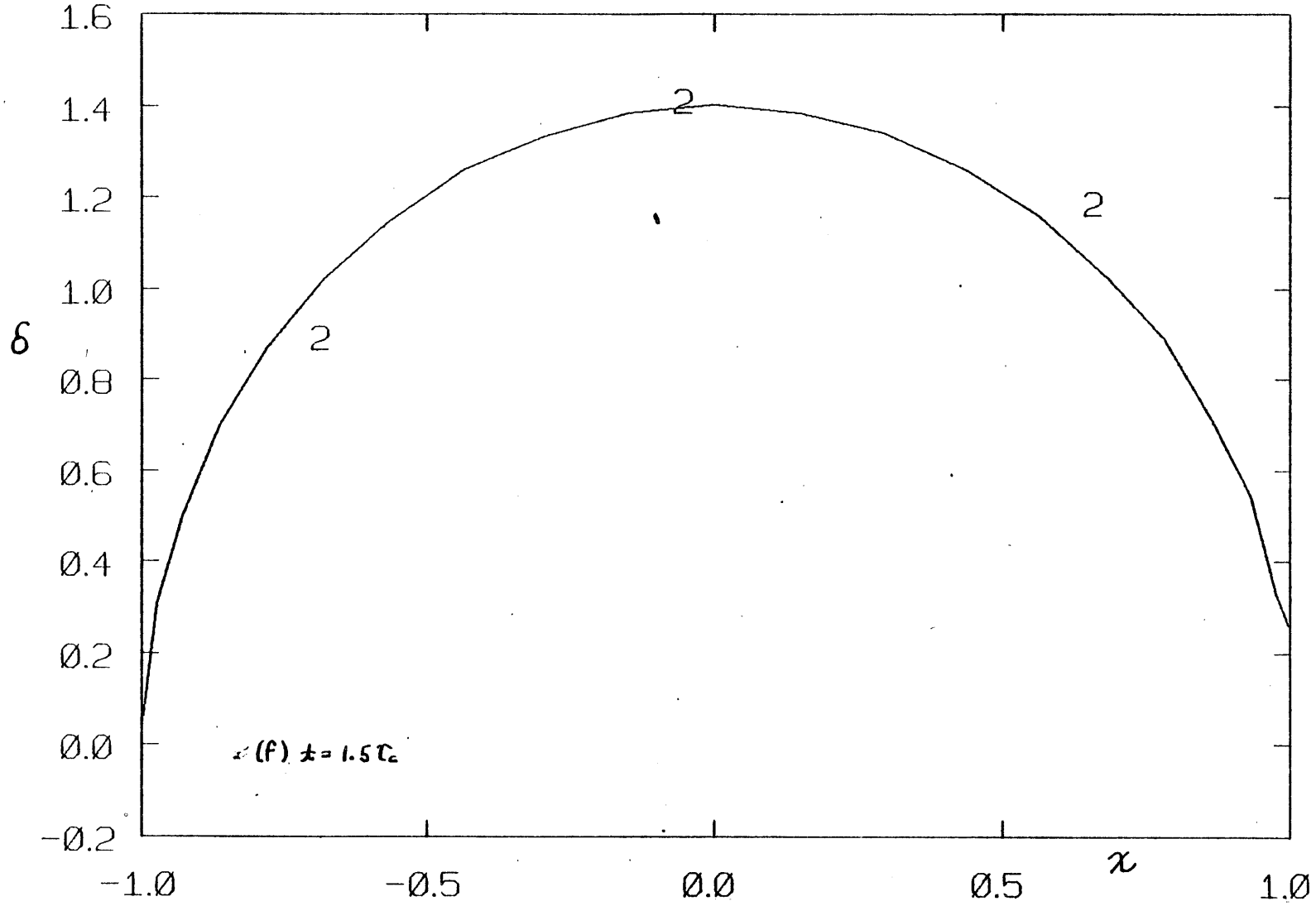
FIG. 3.15. Pressure evolution curves obtained with $\alpha=.9$, but otherwise under the same conditions as those in fig. 3.13. Only slight instability- at $t=1.5\tau_c$ - occurs, but it is apparent that the best results are obtained with $\alpha=1$. (a)- (c) P ; (d)-(f) δ ; (g),(h) $\hat{\delta}$.

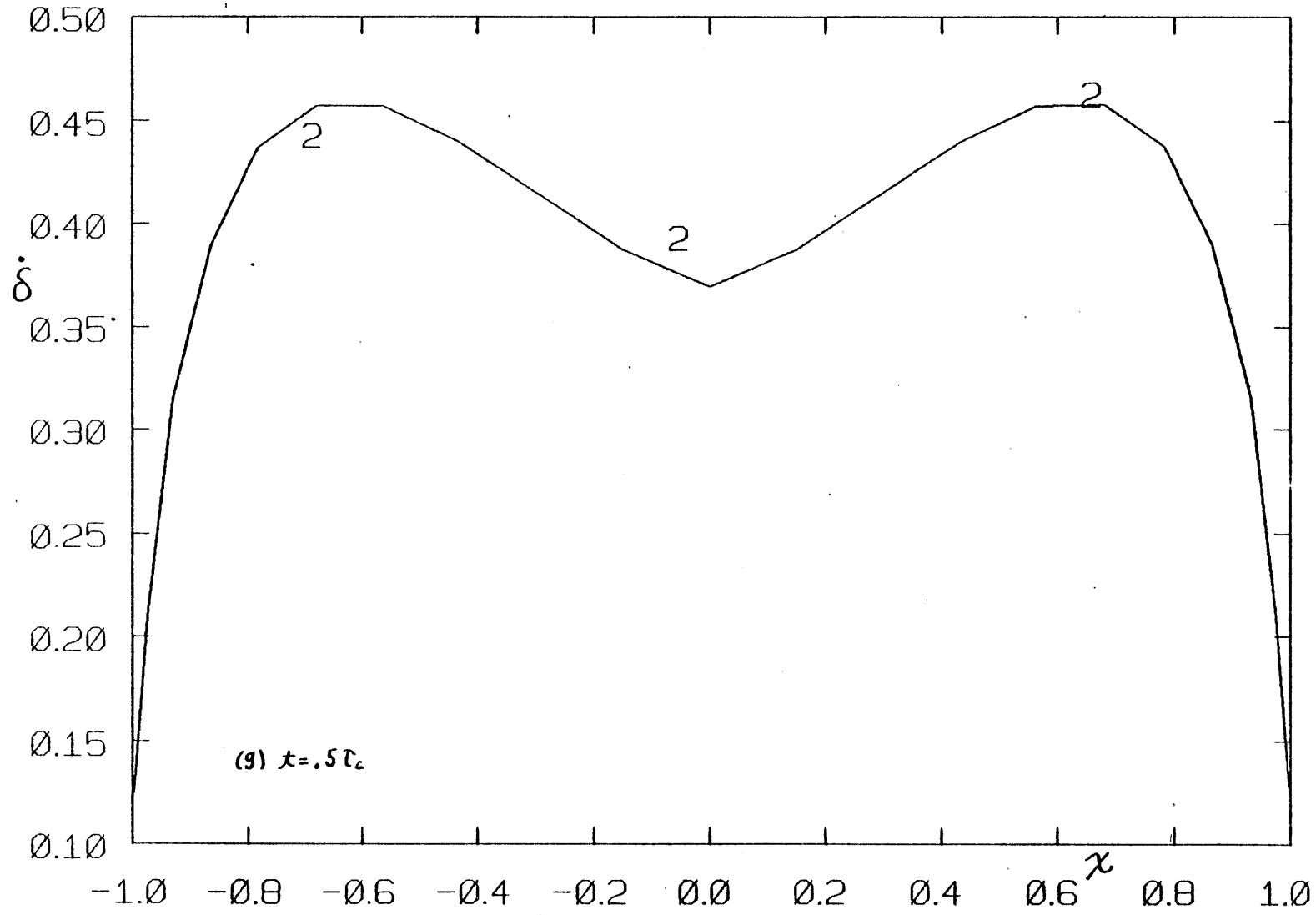

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
2
2
2
(b) t = 1.0 τ₀
-1.0
-0.5
0.0
0.5
1.0
x

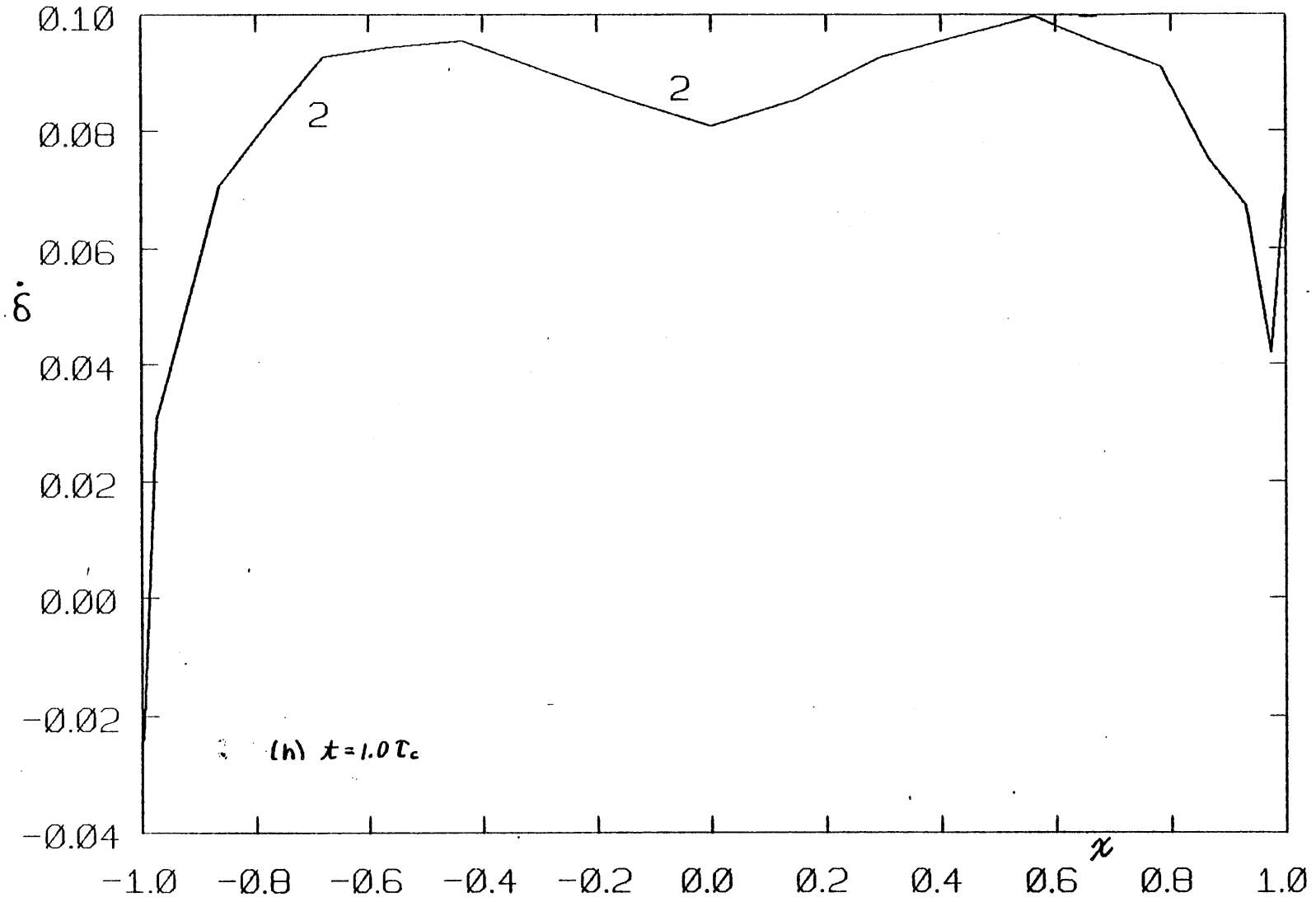

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
p
2
2
2
(c) t = 1.5 τc
-1.0
-0.5
0.0
0.5
1.0
x


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(d) t= .5 τc


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
χ
2
2
2
(e) t = 1.0 τc


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(f) t = 1.5 τc


0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
δ̇
2
2
2
(g) t=.5τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x


0.10
0.08
0.06
δ̇
0.04
0.02
0.00
-0.02
-0.04
2
2
(h) t = 1.0 τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x

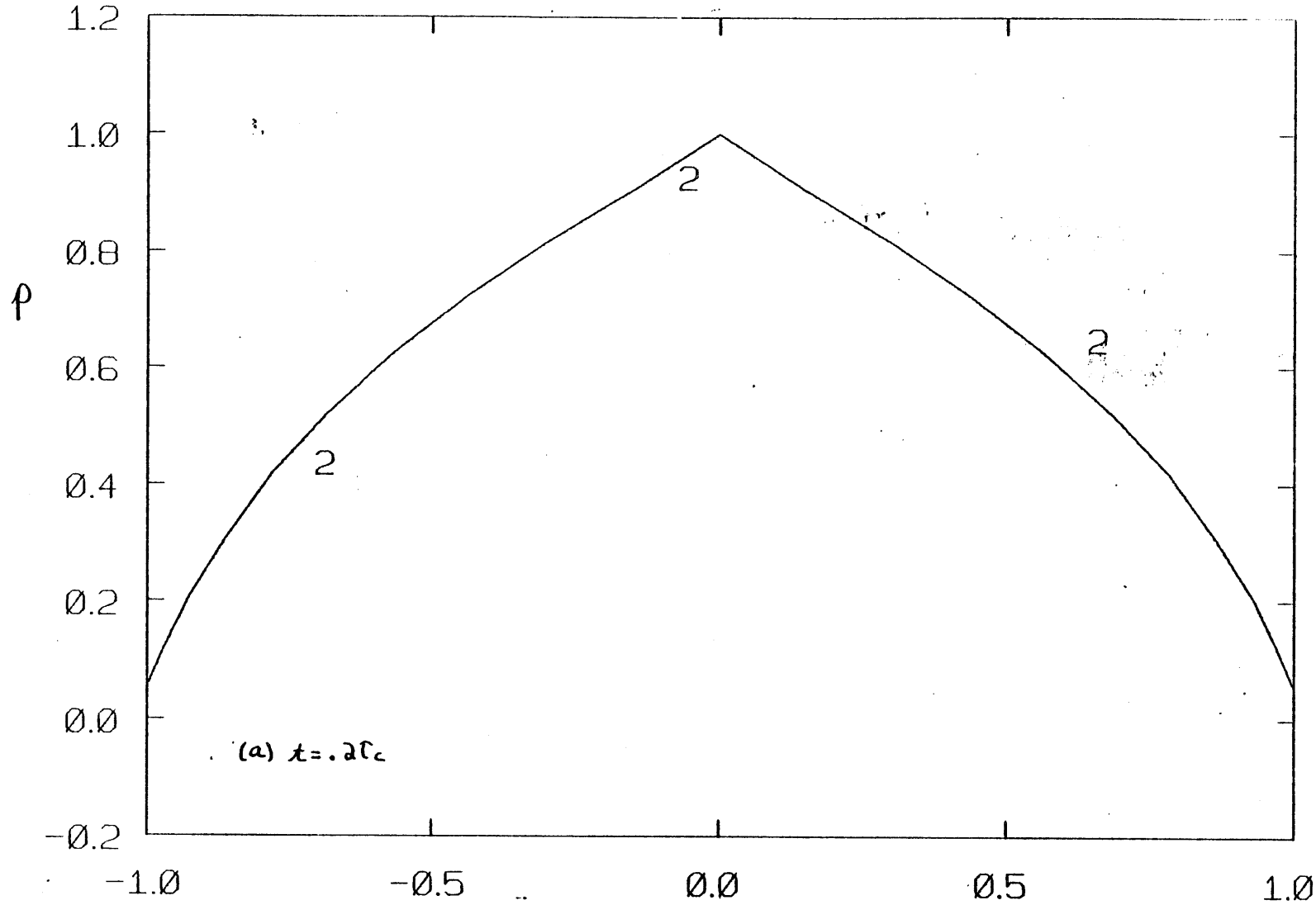
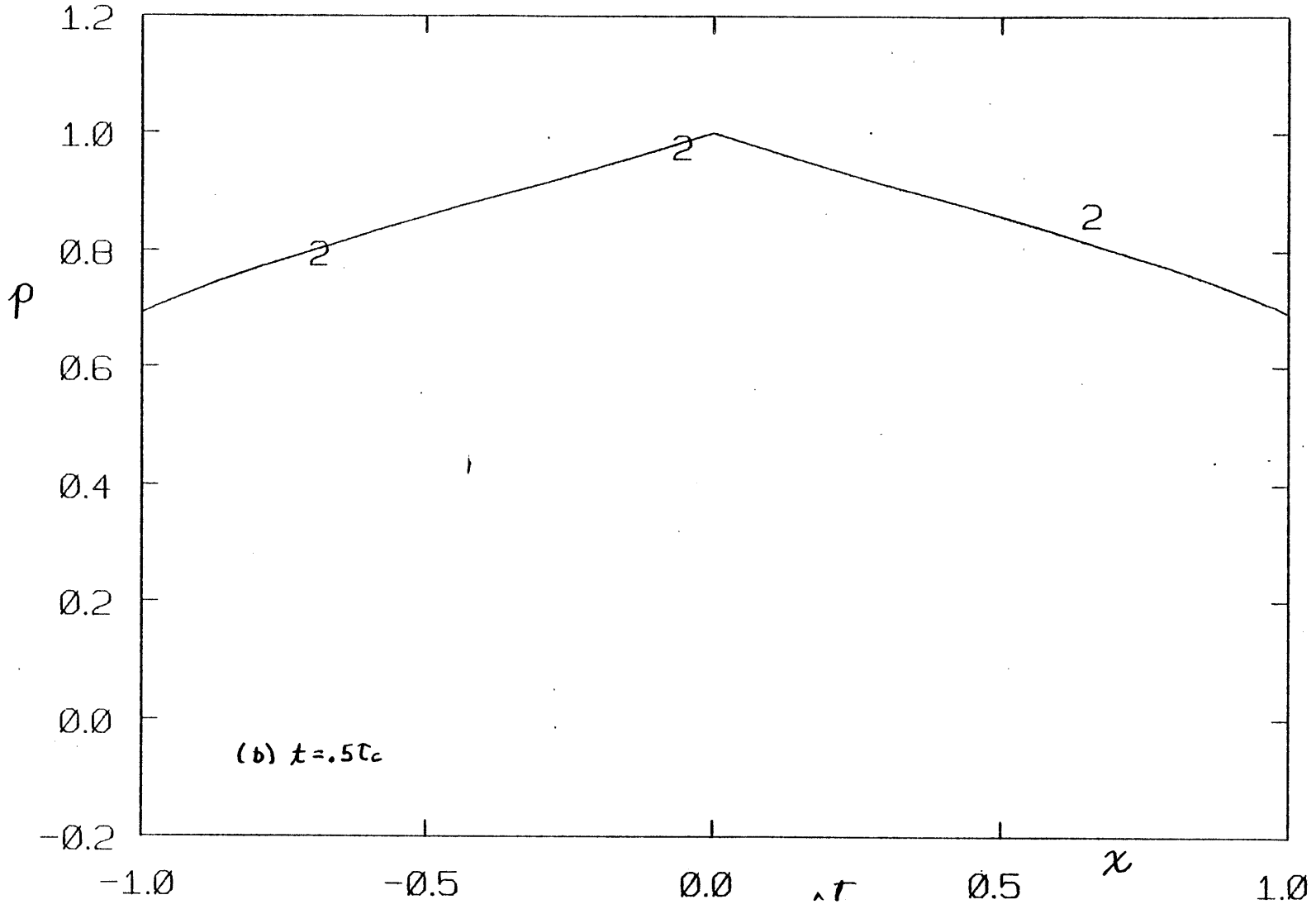
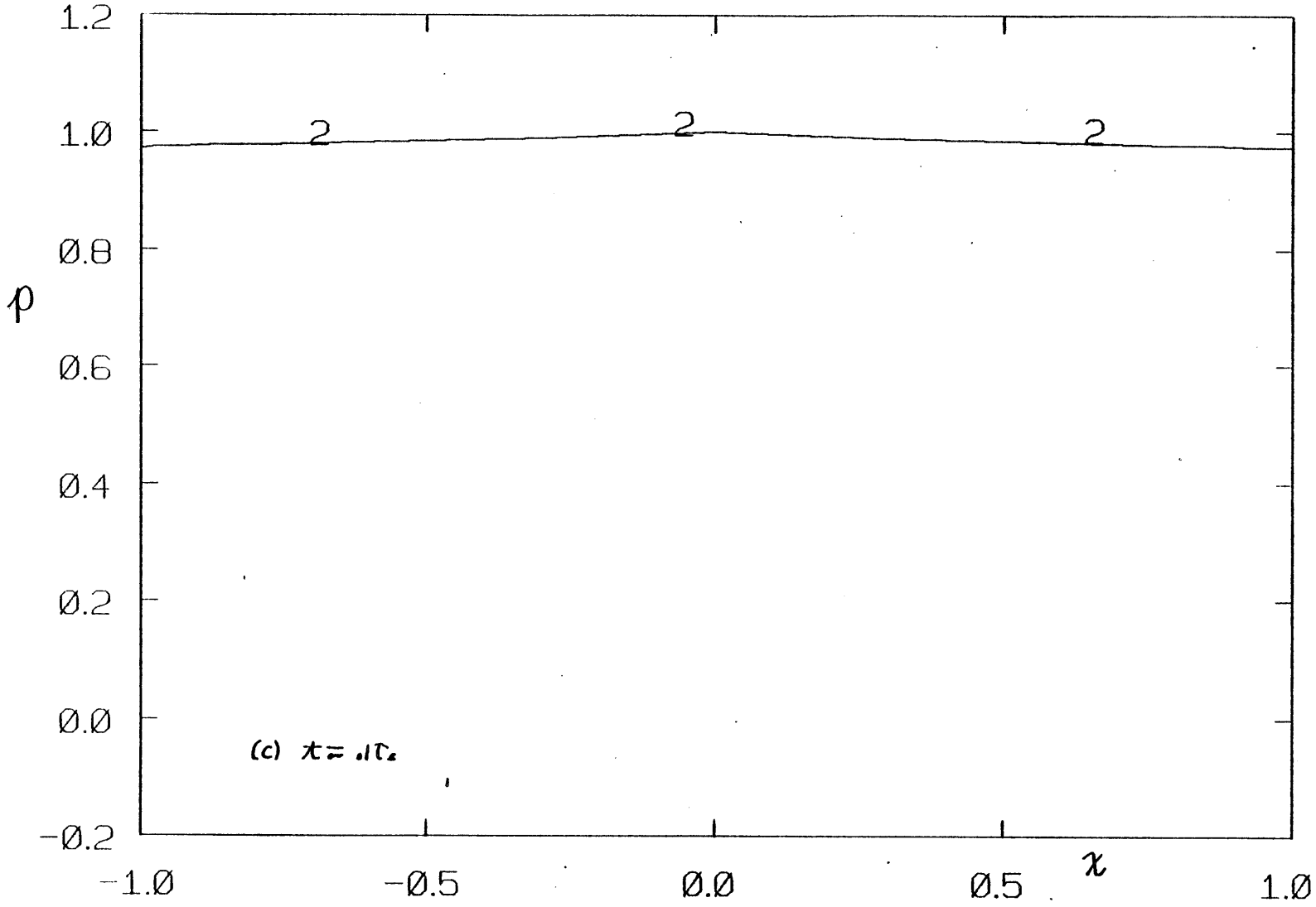
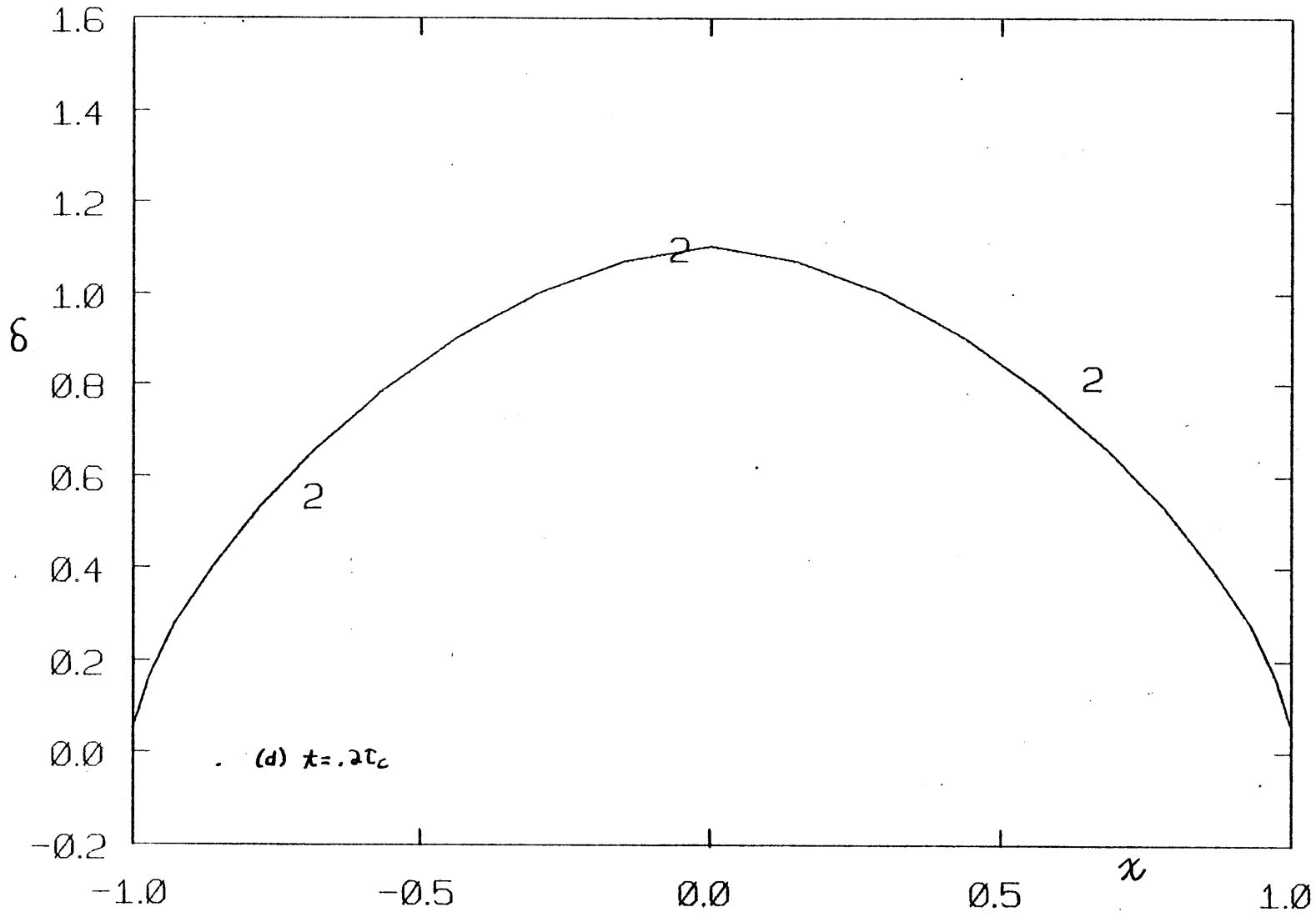
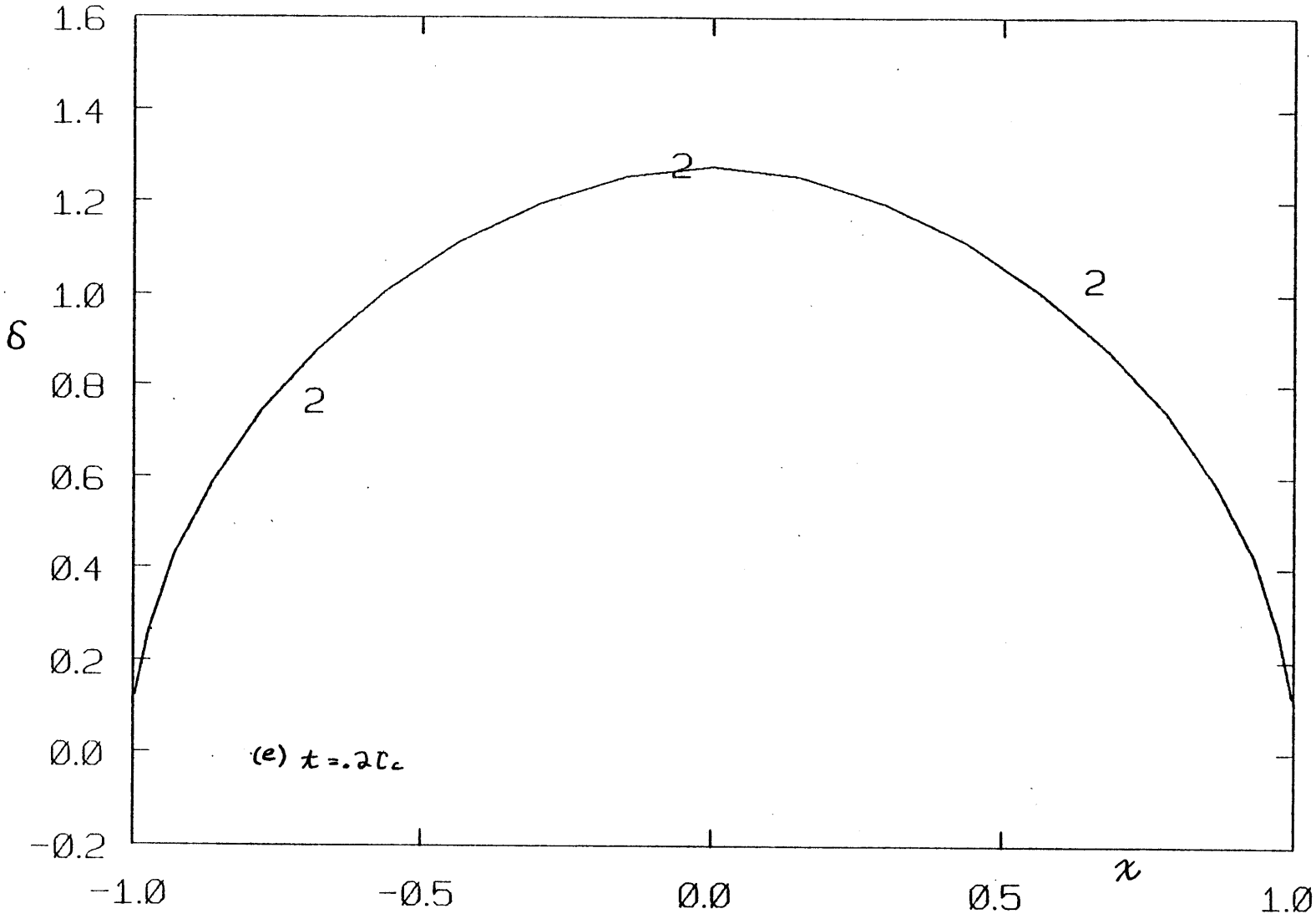


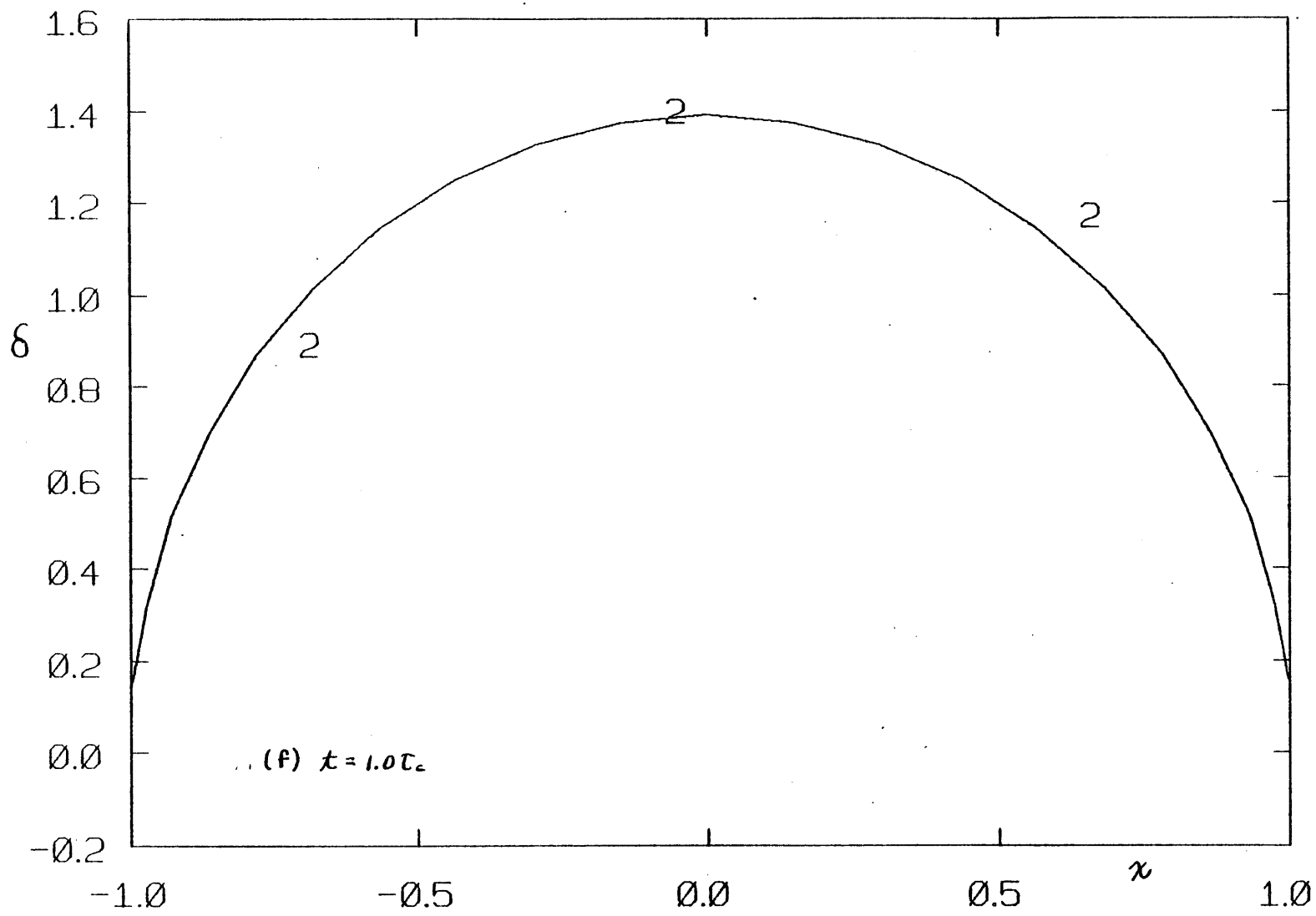
FIG. 3.16. Pressure evolution curves obtained with $\Delta t = .1\tau_c$, but otherwise under the same conditions as those in fig. 3.13: $\rho_0/G=1$, $\nu=.3$, $\alpha=1$. Time steps of $.1\tau_c$ or less are necessary for all but rough calculations. (a)-(c) p ; (d)-(f) δ ; (g),(h) $\dot{\delta}$.

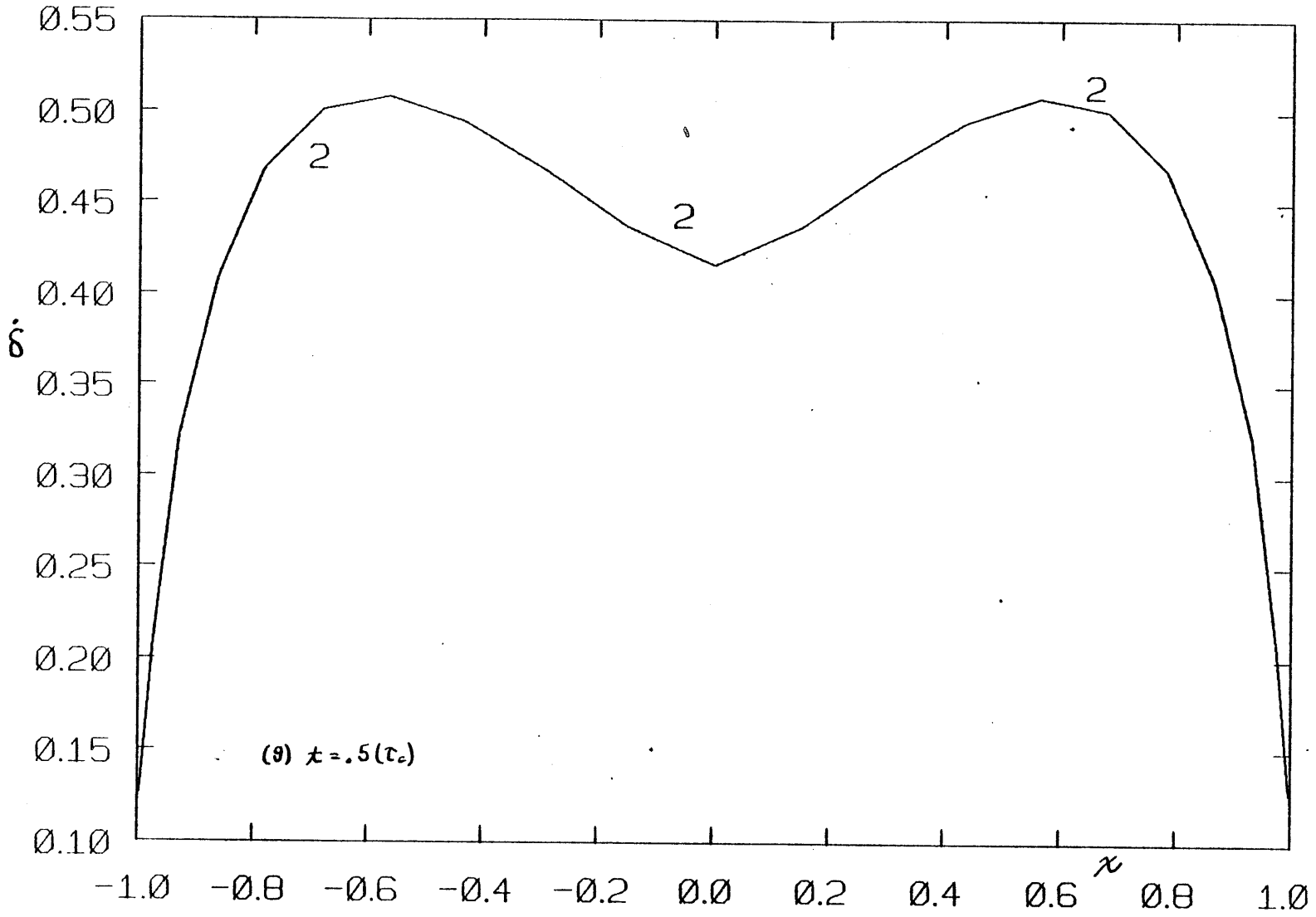

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
2
2
2
(b) t=.5τc
-1.0
-0.5
0.0
0.5
1.0
x

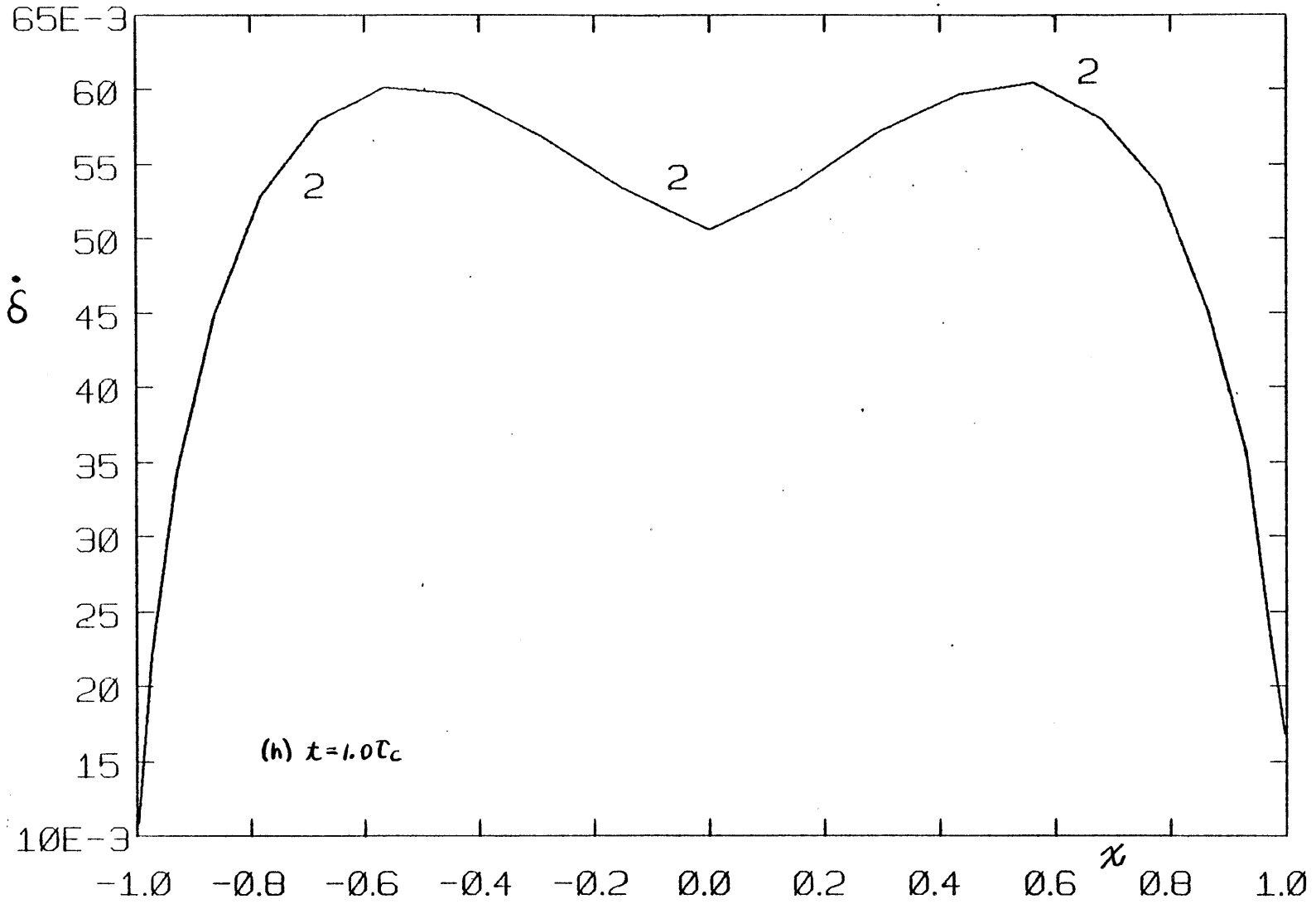

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
2
2
2
(c) t = .1τ₀
-1.0
-0.5
0.0
0.5
1.0
x


1.6
1.4
1.2
1.0
Ø.8
Ø.6
Ø.4
Ø.2
Ø.0
-Ø.2
δ
2
2
2
(d) t=.2τc
-1.Ø
-Ø.5
Ø.Ø
Ø.5
1.Ø
x


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(e) t = .2 t_c


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
2
2
2
(f) $t = 1.0\tau_c$
-1.0
-0.5
0.0
0.5
1.0
x


0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
δ̇
2
2
2
(θ) t=.5(τc)
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x


65E-3
60
55
50
45
40
35
30
25
20
15
10E-3
δ̇
2
2
2
(h) t=1.0τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x

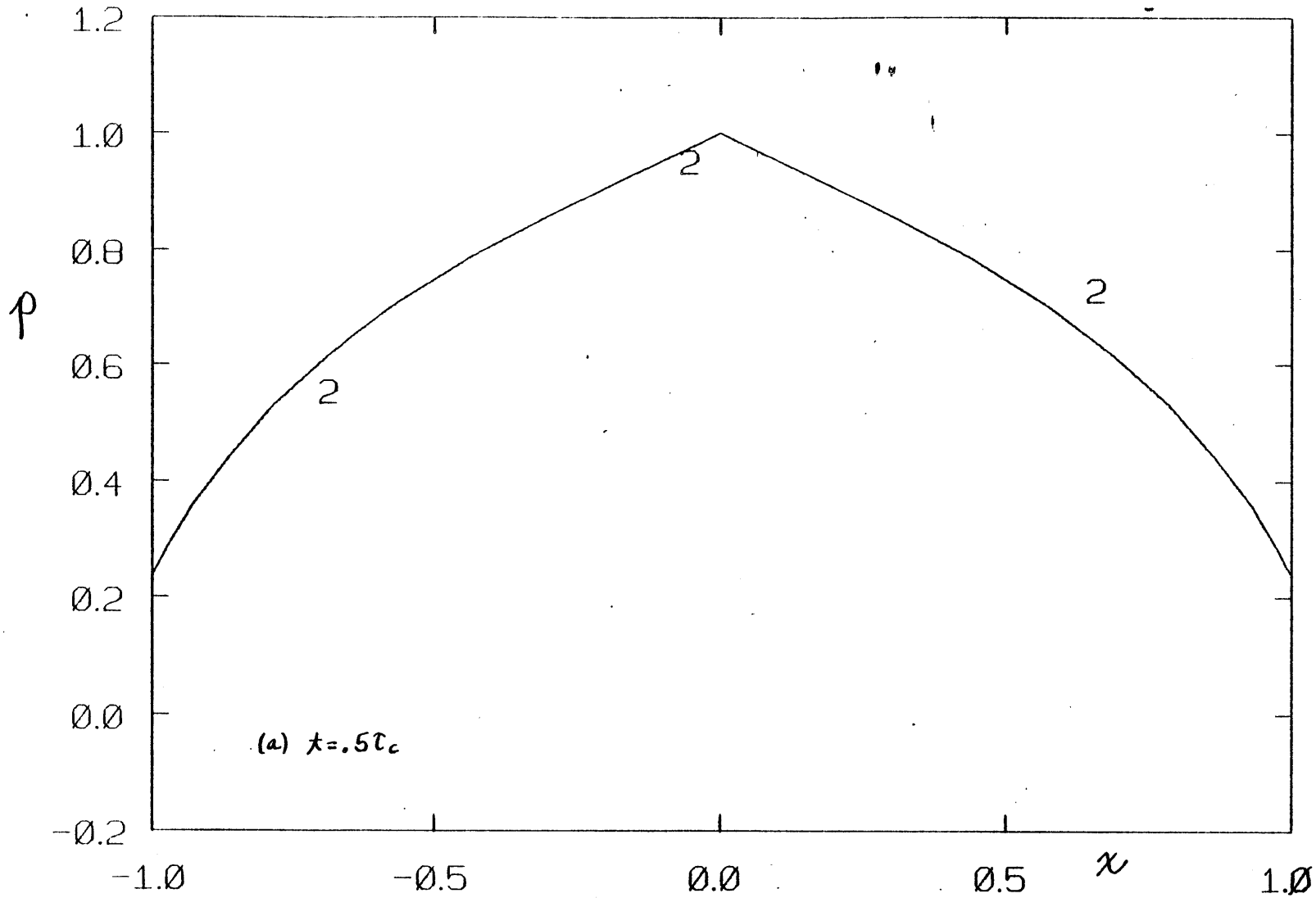
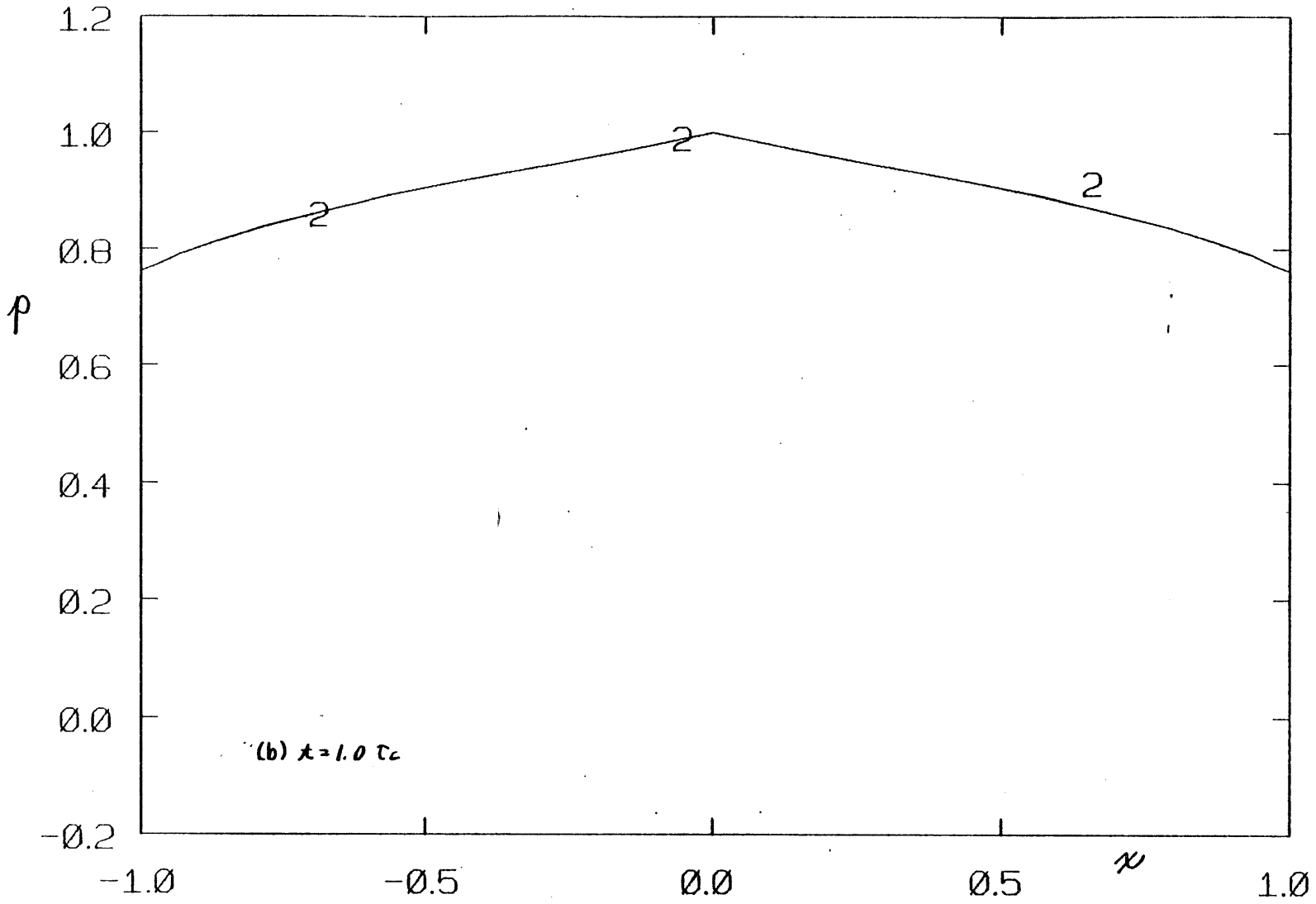
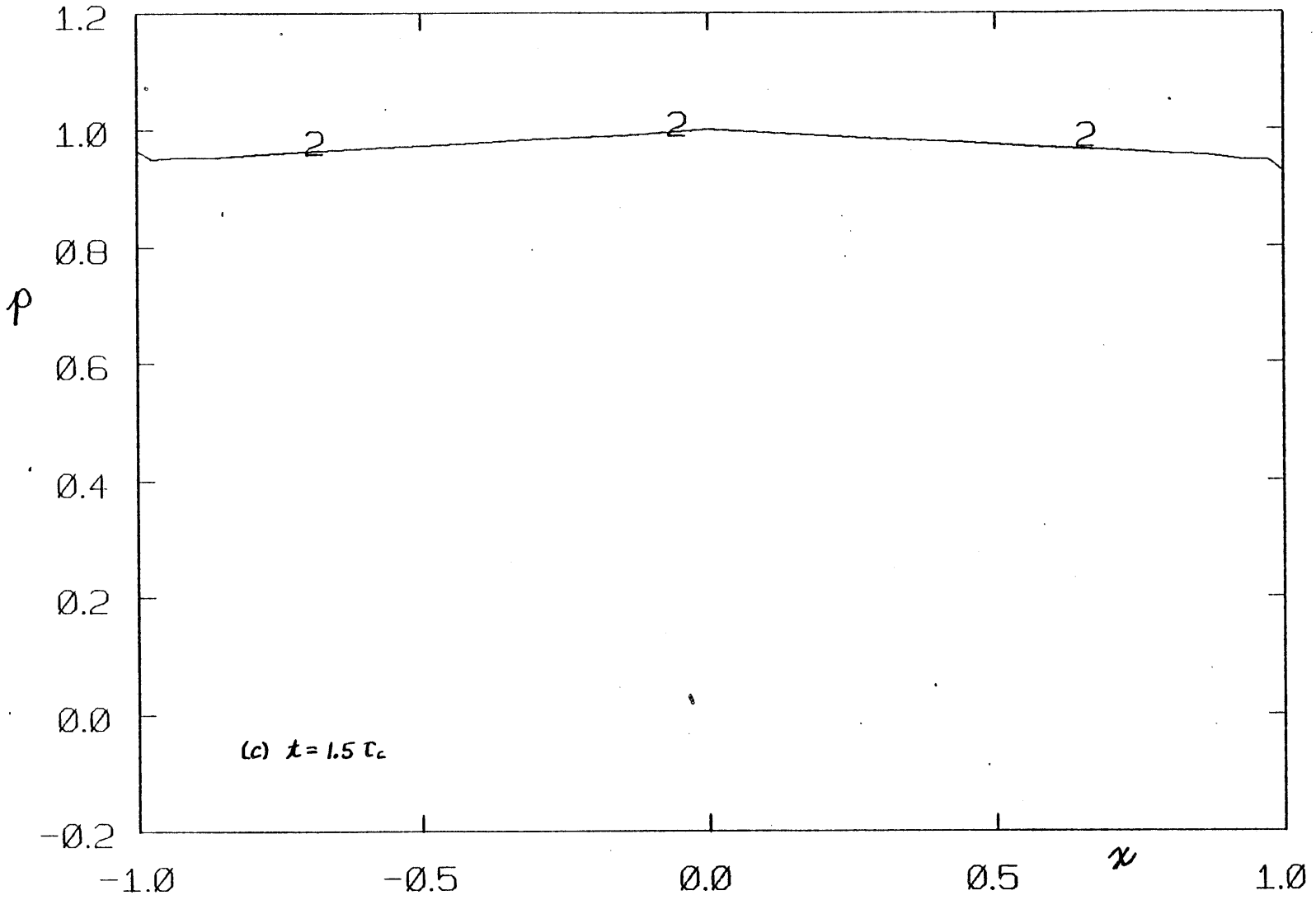
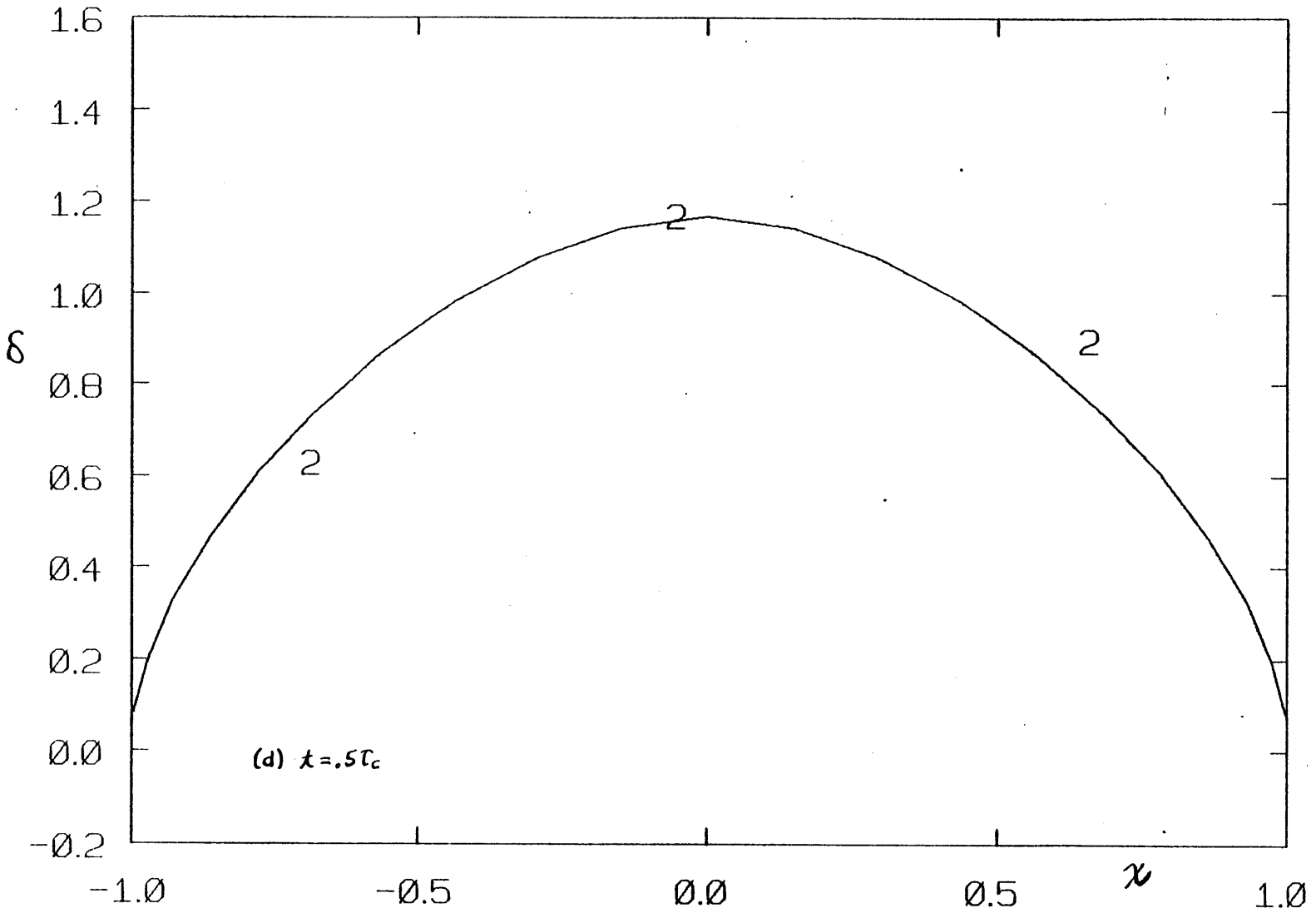
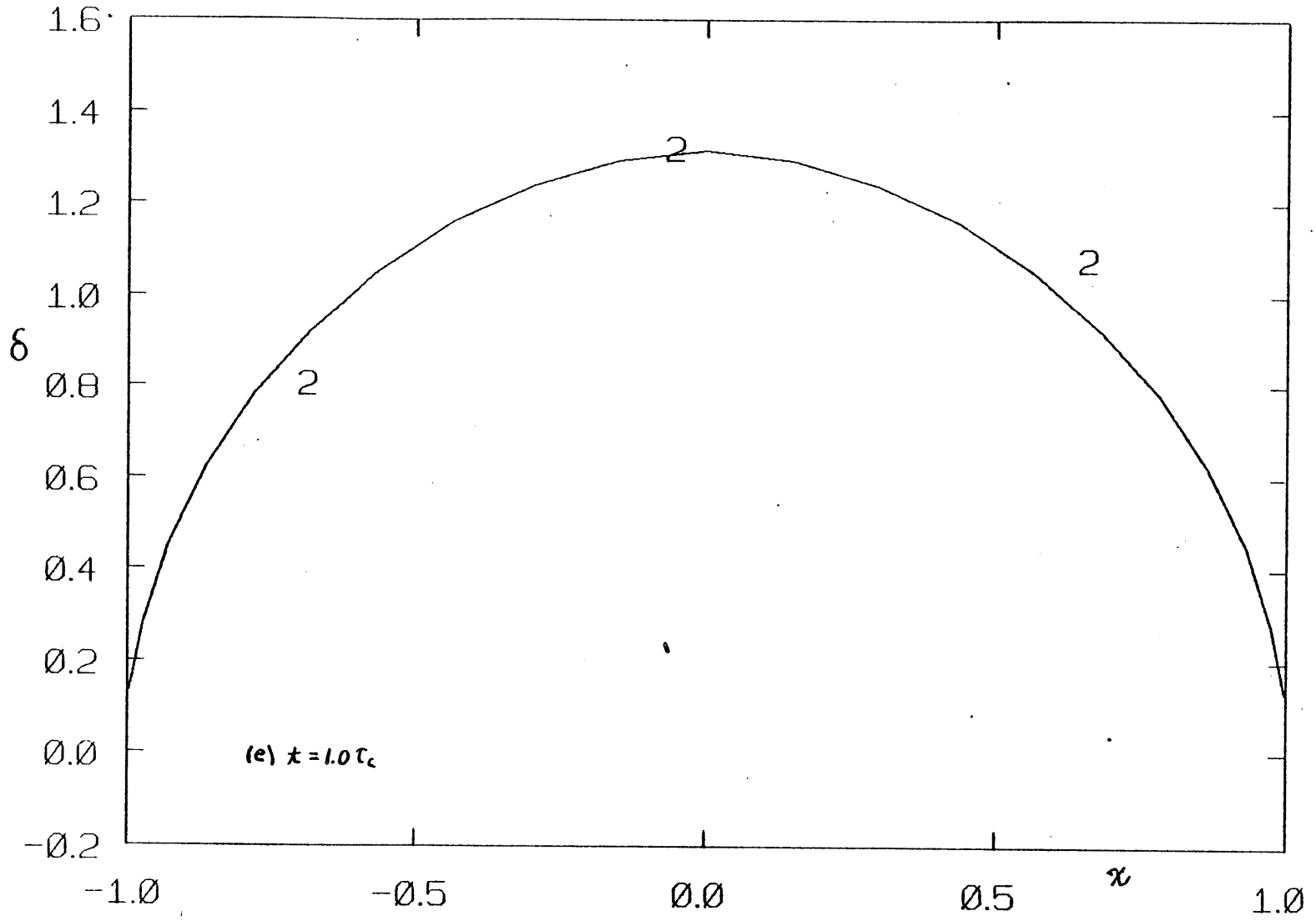


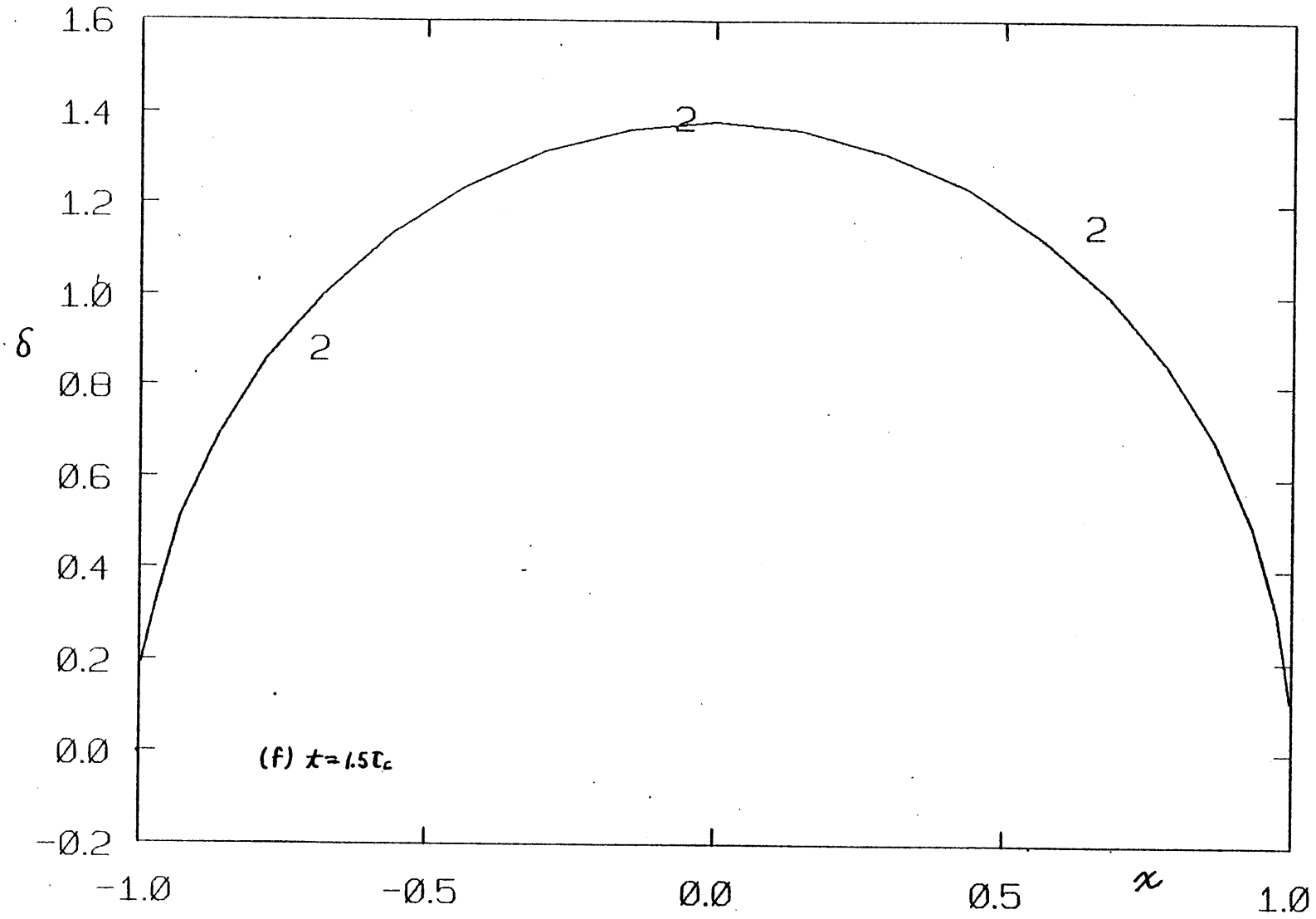
FIG. 3.17. Pressure evolution curves obtained with a large time step size ($\Delta t = .5\tau_c$) but otherwise under the same conditions as those in fig. 3.13. These plots demonstrate the stability of the implicit scheme. (a)-(c) ρ ; (d)-(f) δ ; (g),(h) $\dot{\delta}$.

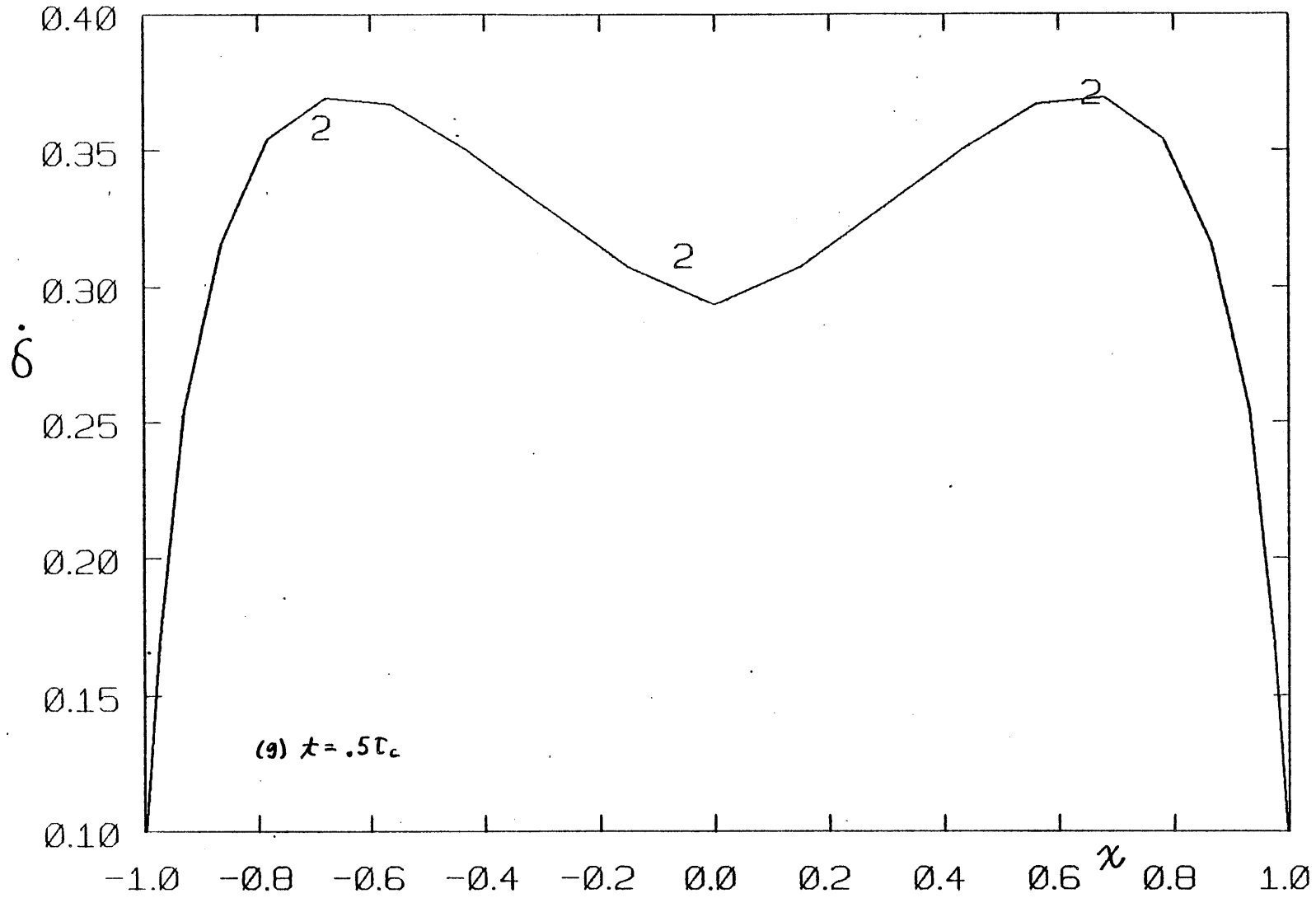

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(b) t = 1.0 τc

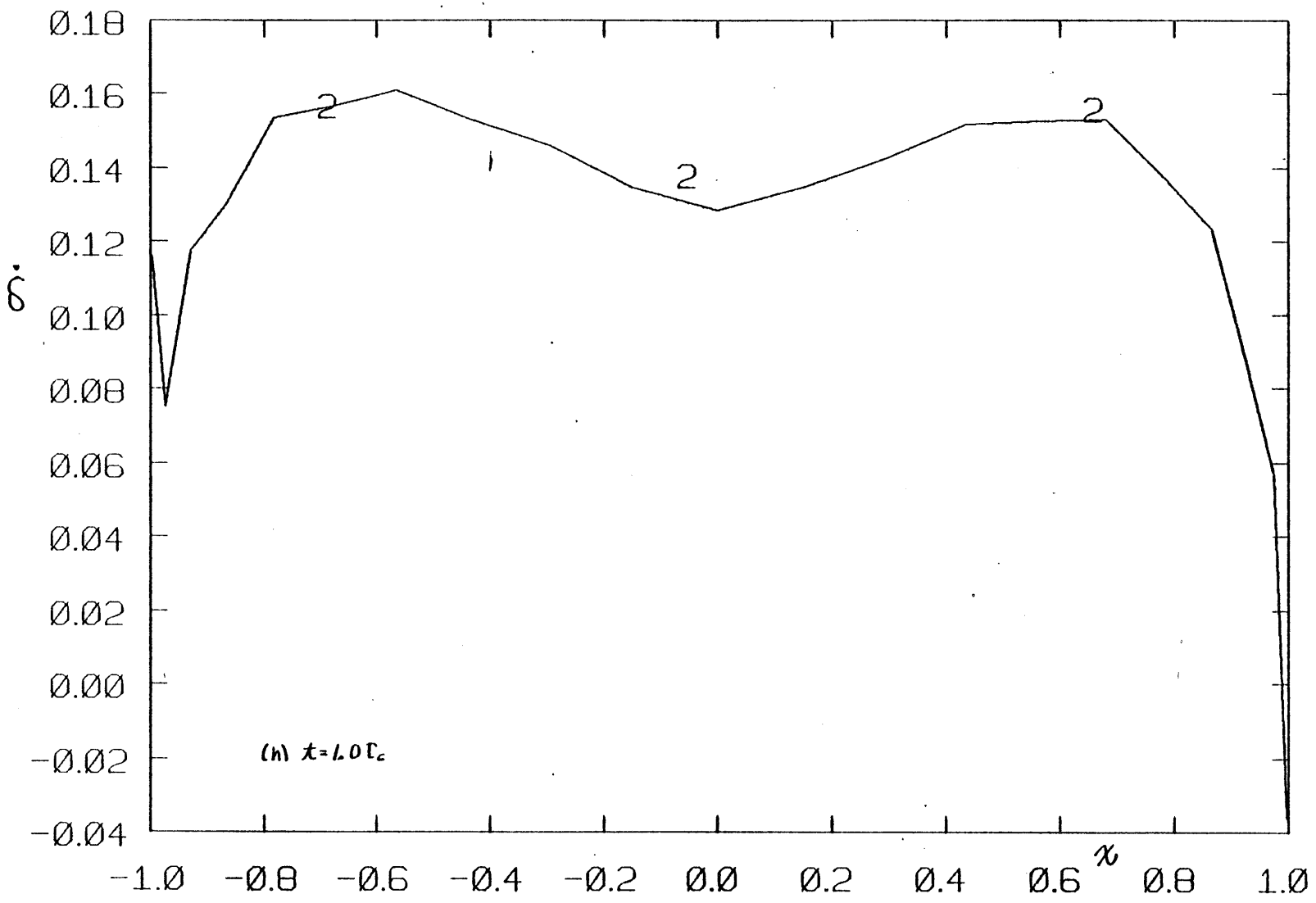

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
2
2
2
(c) t = 1.5 τc
-1.0
-0.5
0.0
0.5
1.0
x


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(d) $t = .5\tau_c$


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
2
2
2
(e) $t = 1.0\,\tau_c$
-1.0
-0.5
0.0
0.5
1.0
x


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(f) t=1.5τc


0.40
0.35
0.30
0.25
0.20
0.15
0.10
δ̇
2
2
2
(g) t = .5τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
χ


0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
-0.02
-0.04
δ
2
1
2
2
(h) t=1.0τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x

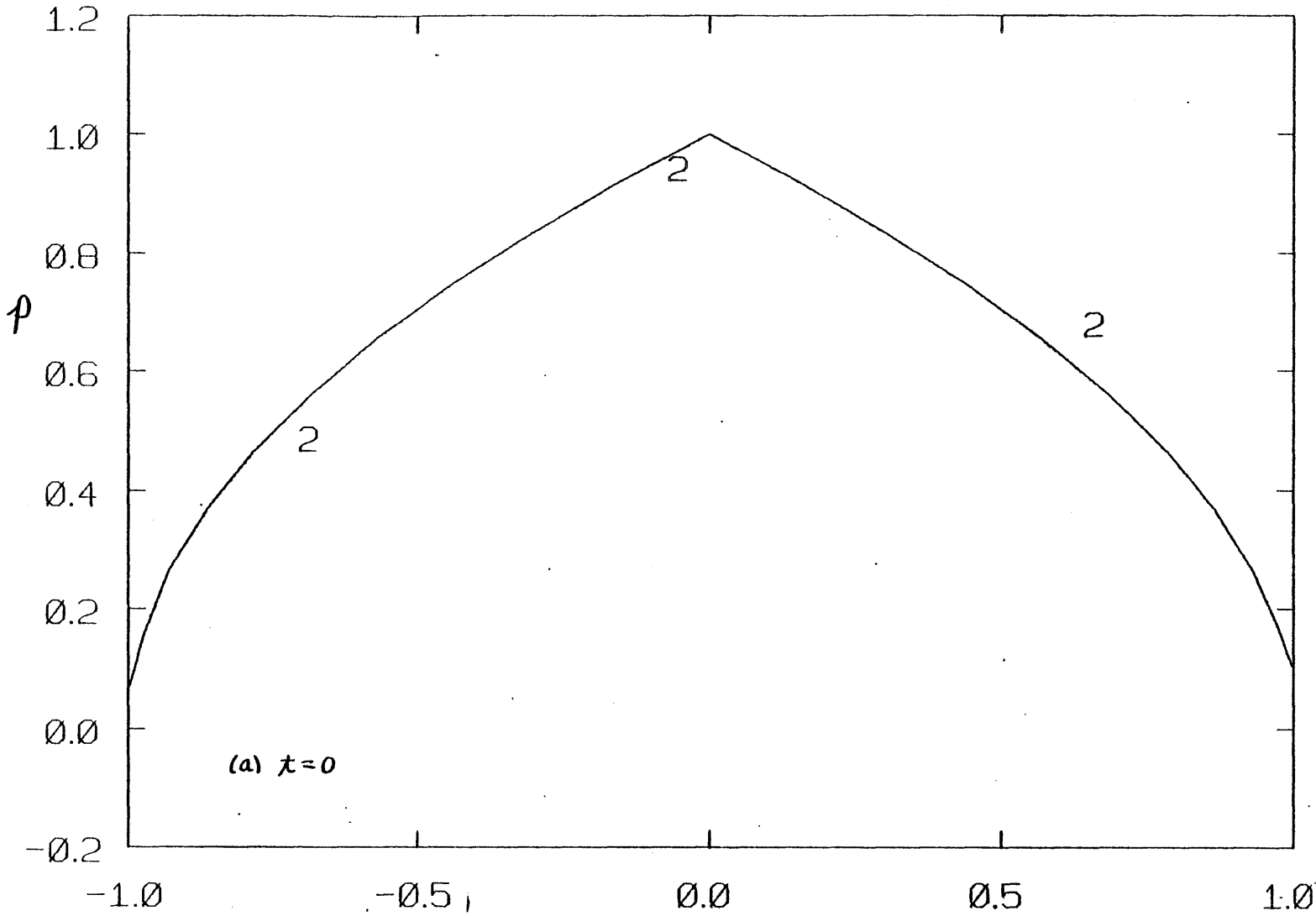
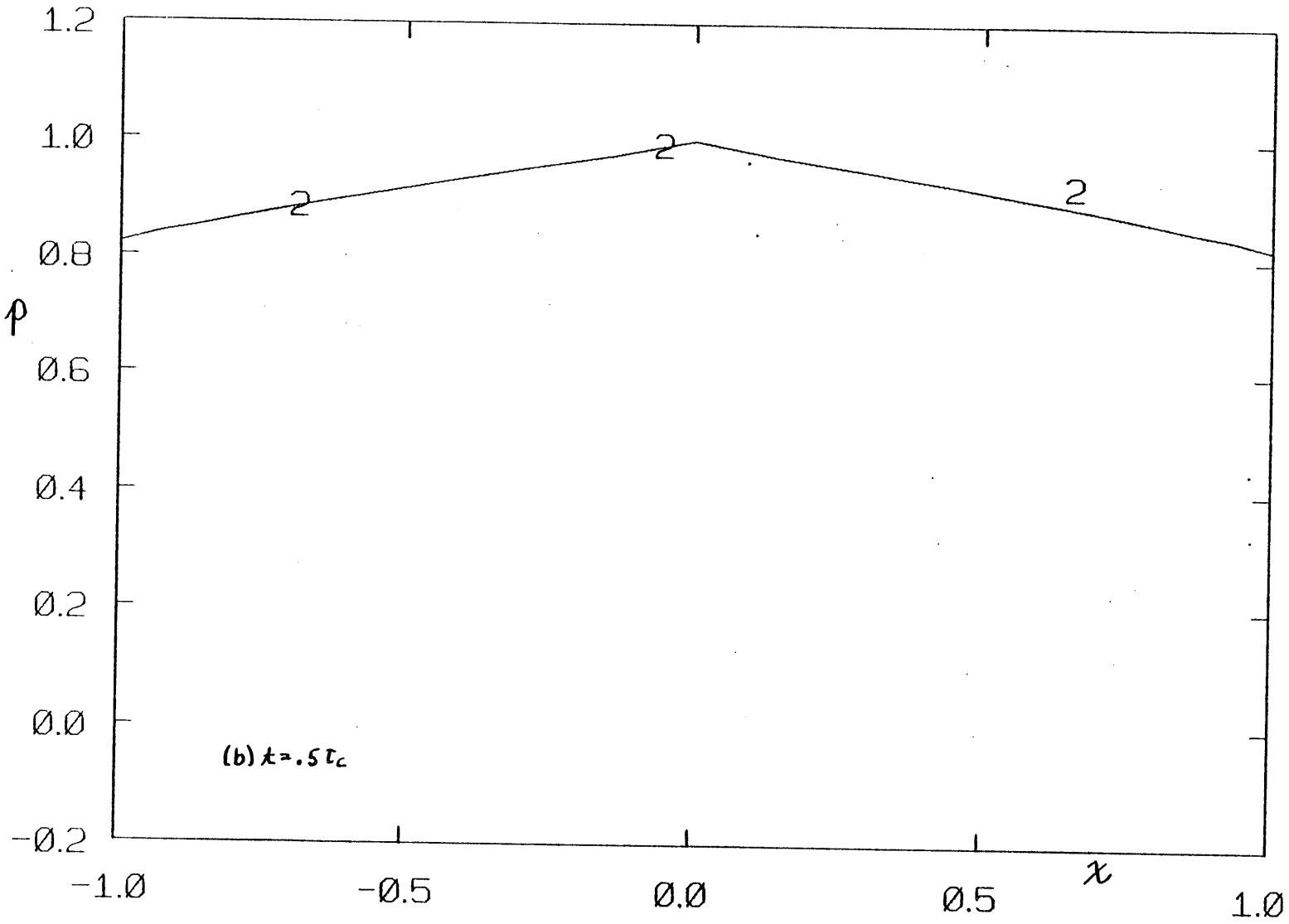
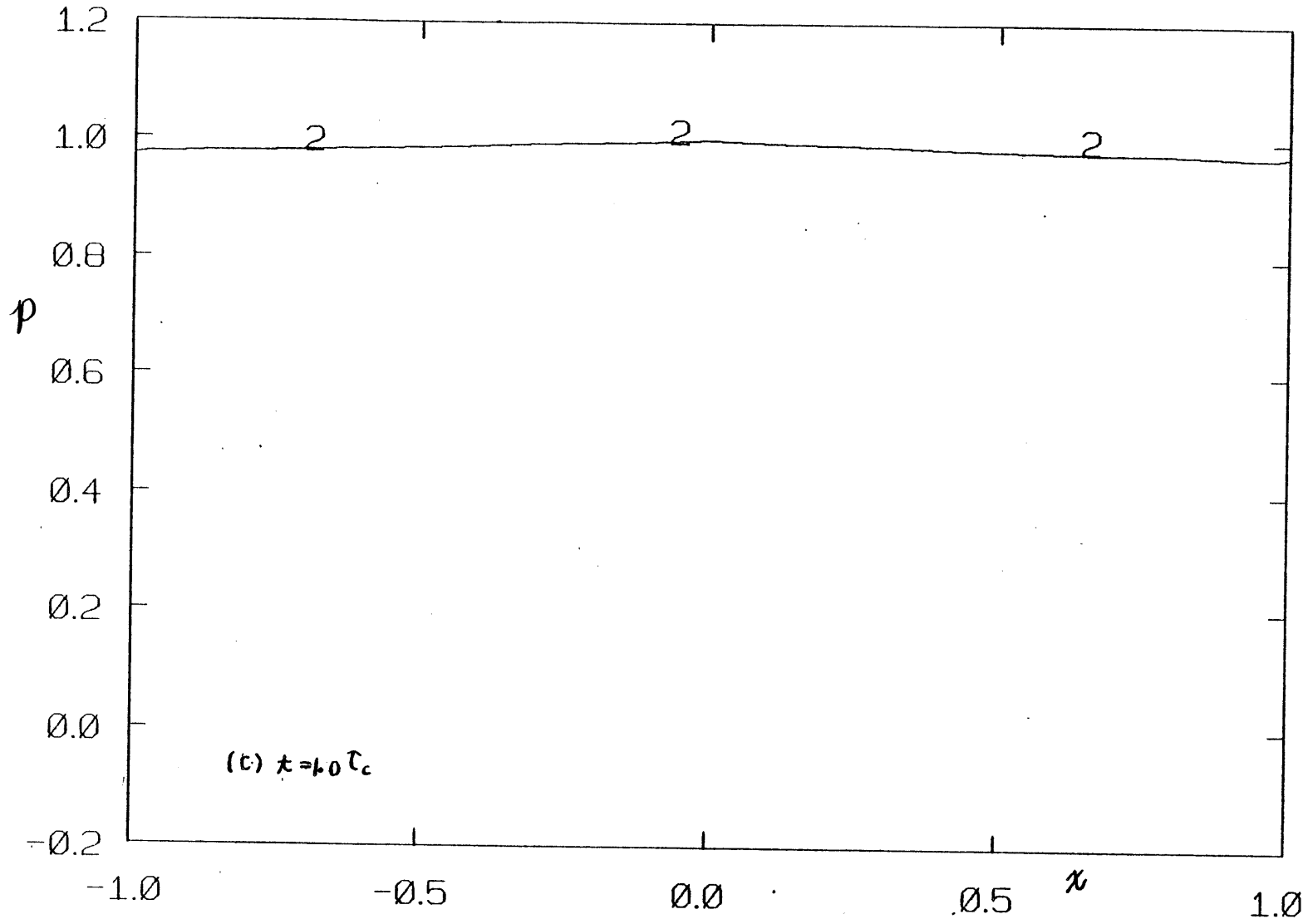
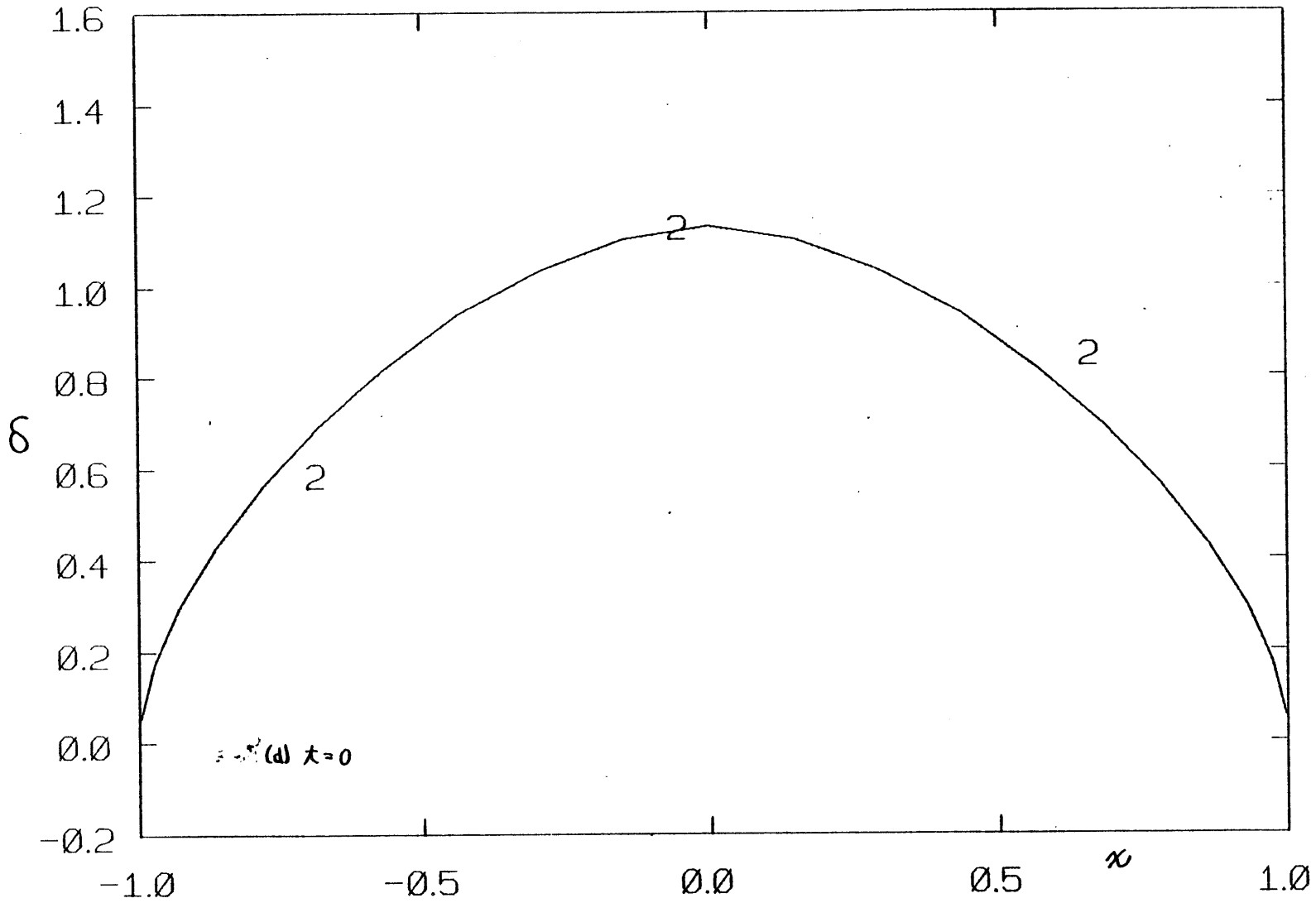
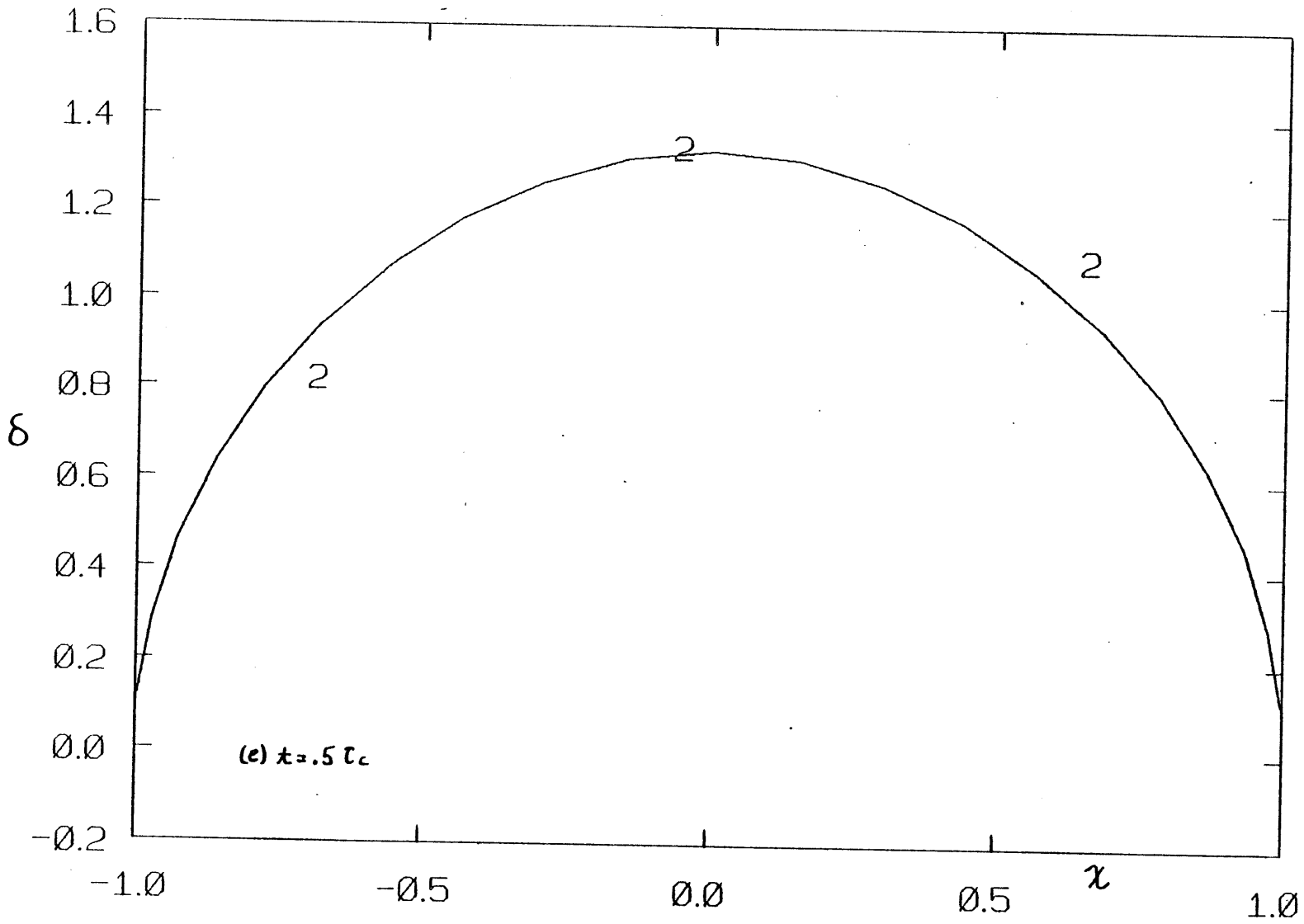


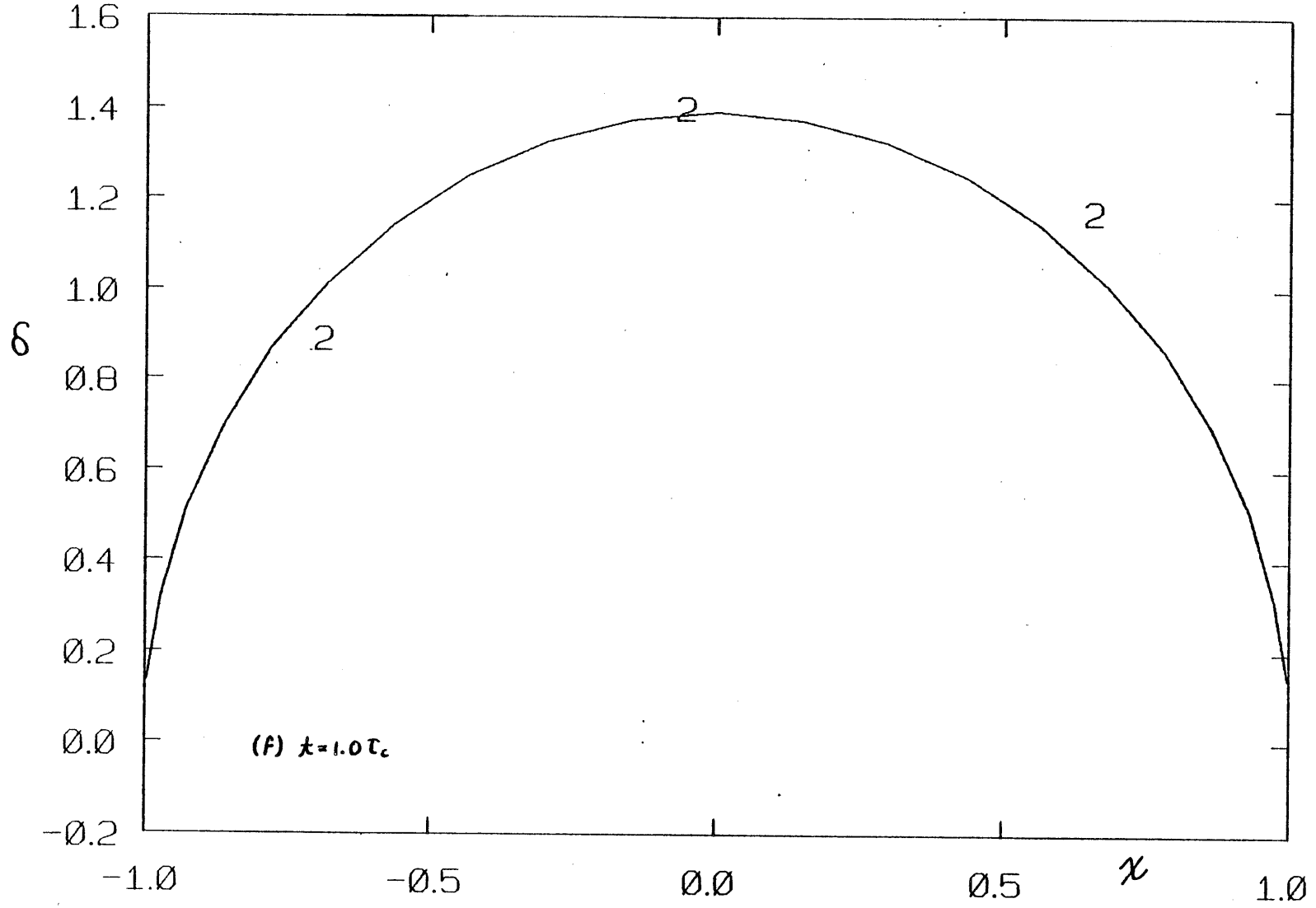
FIG. 3.18. Pressure evolution curves computed from a different initial pressure distribution ($p(x,t=0) = \sqrt{1+|x|}$). Note the reversal of curvature of p near the borehole after t=0 and the more rapid approach to uniform pressure than is obtained with a triangular initial pressure distribution. Again, $\Delta t = .25\,\tau_c$, $\alpha=1$, $p_o/G=1$, $\nu=.3$. (a)-(c) p ; (d)-(g) δ ; (h),(i) $\dot{\delta}$.

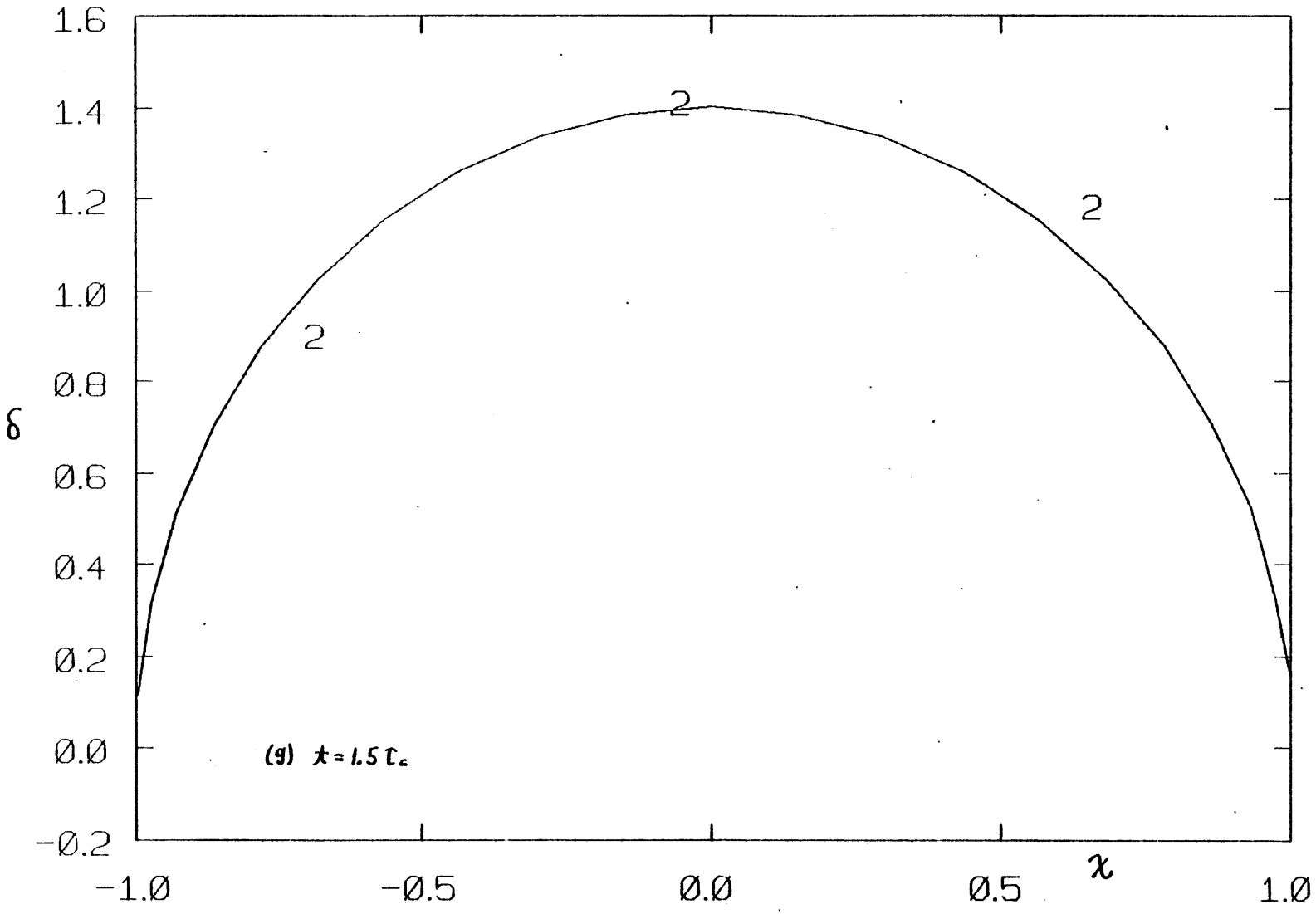

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
2
2
2
(b) t = .5 τc
-1.0
-0.5
0.0
0.5
1.0
x

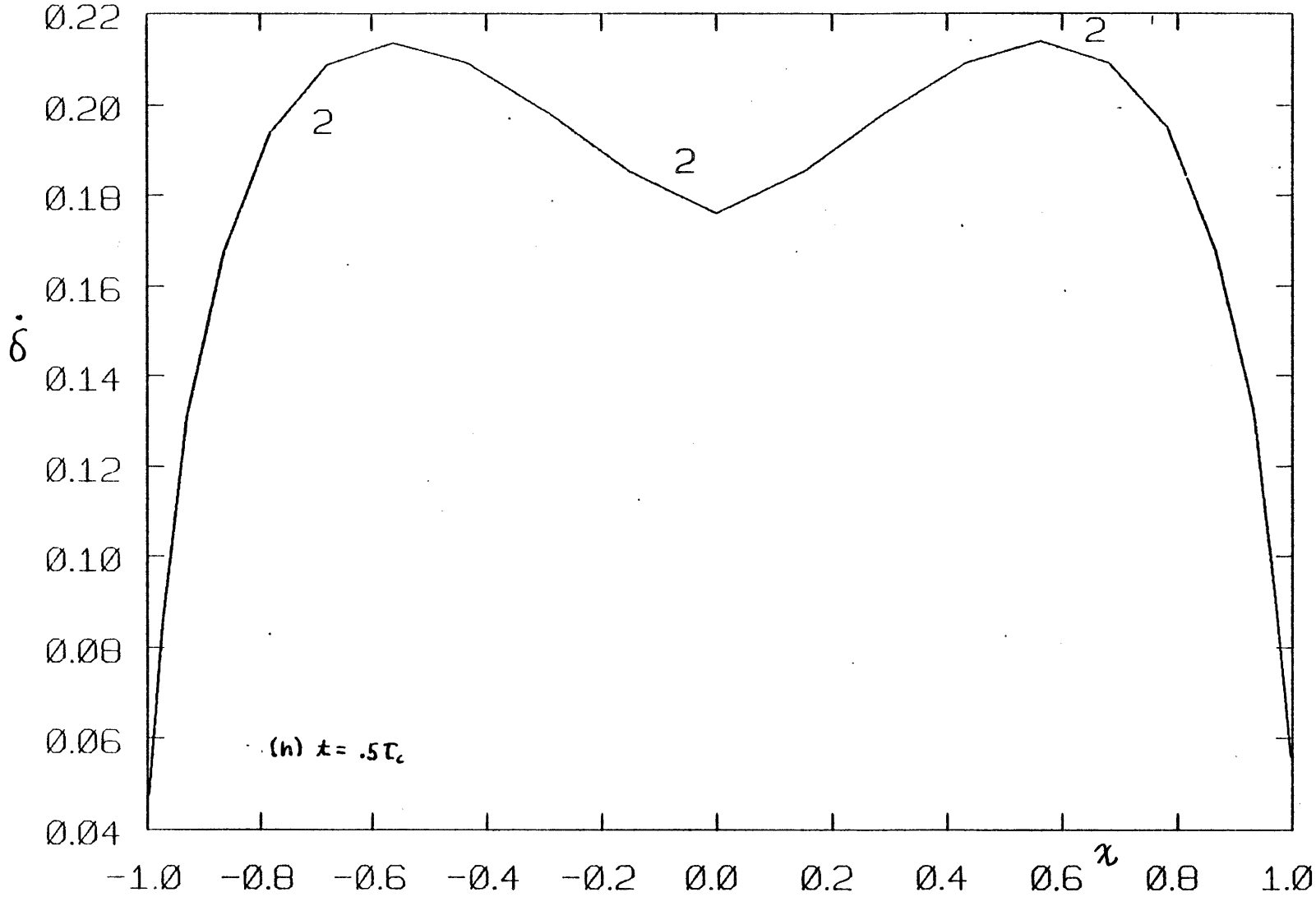

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
p
2
2
2
(c) t = 4.0 τc
-1.0
-0.5
0.0
0.5
1.0
x

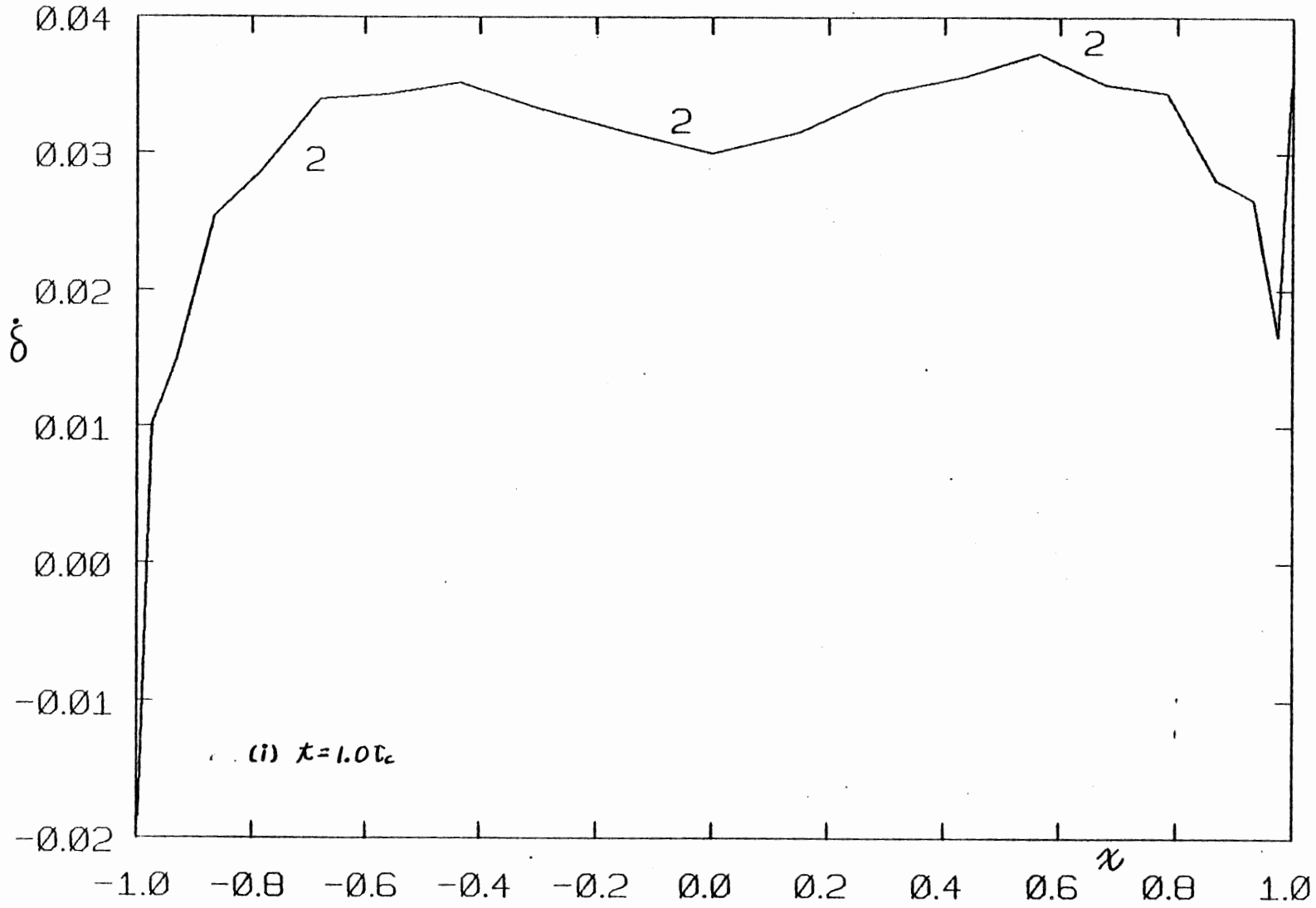

1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
2
2
2
(d) t = 0
-1.0
-0.5
0.0
0.5
1.0
x


1.6
1.4
1.2
1.0
Ø.8
Ø.6
Ø.4
Ø.2
Ø.0
-Ø.2
δ
2
2
2
(e) t=.5 τc
-1.0
-Ø.5
Ø.0
Ø.5
1.Ø
χ


1.6
1.4
1.2
1.0
Ø.8
Ø.6
Ø.4
Ø.2
Ø.0
-Ø.2
δ
2
2
2
(f) t=1.0τc
-1.0
-Ø.5
Ø.Ø
Ø.5
1.Ø
x


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
χ
2
2
2
(g) t = 1.5 τc


0.22
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
δ̇
2
2
2
(h) t = .5τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x


0.04
0.03
0.02
0.01
0.00
-0.01
-0.02
δ
2
2
2
(i) t = 1.0 τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x

are noteworthy: first, the pressure reaches an essentially uniform value more rapidly ($1.25\tau_c$ vs. $1.5\tau_c$), and as well, the negative (adverse) curvature of the initial pressure curve has reversed by time $.25\tau_c$. The latter observation provides evidence that we can start with a variety of initial distributions and be assured of stability of the solution, and that the various pressure distributions will quickly tend towards the same shape with ongoing time.

3.4 Fluid Front Advancement in Stationary Cracks

A typical fluid front advancement problem is illustrated in Figure 3.19. In general, we again have the overall elasticity equation*, suitably non-dimensionalized

$$\dot{p}(x_o) = \int_{-\ell}^{\ell} \gamma_D(x_o, x)\left[\dot{\delta}(x) - \dot{\delta}(x_o)\right] dx - \dot{\delta}(x_o)\left[\gamma(x_o, 1) - \gamma(x_o, -1)\right]$$

$$\gamma_D \equiv -\frac{d}{dx}\gamma \qquad (3.26a)$$

With reference to this equation, we may now phrase the distinct conditions, one pertaining to the non-penetrated zone (size ω) near each tip while the other prevails in the fluid-filled region (where laminar flow of Newtonian fluid is assumed for simplicity in early testing of our routines)

$$\dot{p}(\ell \geq |x_o| \geq \ell - \omega) = 0, \quad \dot{\delta}(\ell - \omega \geq |x| \geq 0) \simeq \frac{1}{\hat{\eta}}\left[\delta^3 p'\right]' \qquad (3.26b)$$

*Note that this equation also applies, remarkably, to the moving crack problem.

Thus we must solve for fluid pressure in the fluid filled region of the crack and crack opening rate in the empty region. Our experience with the pressure evolution problem (Ref. 26-28) demands that an implicit method be used for the fluid front advancement problem. Further, since we must solve for the opening rate over part of the crack, it will be convenient to construct our system of equations so as to solve for opening rate over the entire crack.

By simple approximation of time derivatives, we obtain from Eq. (3.26) an implicit equation for $^{t+\Delta t}\dot{\delta}$ and $^{t+\Delta t}p$ which may be written in the following numerical form [28]:

$$
\begin{aligned}
&{}^{t+\Delta t}p(x_n) - \alpha\Delta t\frac{\pi}{N}\sum_{i=1}^{N}\gamma_D(x_n,t_i)\left[{}^{t+\Delta t}\dot{\delta}(t_i) - {}^{t+\Delta t}\dot{\delta}(x_n)\right]\sqrt{1-t_i^2} \\
&\qquad - \alpha\Delta t\,{}^{t+\Delta t}\dot{\delta}(x_n)\left[\gamma(x_n,1)-\gamma(x_n,-1)\right] \\
&= {}^{t}p(x_n) - (1-\alpha)\Delta t\frac{\pi}{N}\sum_{i=1}^{N}\gamma_D(x_n,t_i)\left[{}^{t}\dot{\delta}(t_i) - {}^{t}\dot{\delta}(x_n)\right]\sqrt{1-t_i^2} \\
&\qquad + (1-\alpha)\Delta t\,{}^{t}\dot{\delta}(x_n)\left[\gamma(x_n,1)-\gamma(x_n,-1)\right] \qquad (3.28)
\end{aligned}
$$

Over the fluid-filled region ($1-\omega \geq |x_n, t_i| \geq 0$), this leads to the following matrix equations for the pressures at the "Chebyshev points" t_s,

$$
\sum_{\ell=1}^{L}\left(1-\tfrac{1}{2}\delta_{\ell 1}\right)T_{\ell-1}(x_n)\sum_{\Delta=1}^{L}\frac{2}{L}T_{\ell-1}(t_\Delta)\left[{}^{t+\Delta t}p(t_\Delta) - {}^{t}p(t_\Delta)\right]
$$

$$= \frac{4\Delta t}{LM\tau_c}\left(\frac{\pi}{N}\right)\sum_{i=1}^{N}\gamma_D(x_n,t_i)\sum_{j=1}^{M}\left\{T_j'(t_i)-T_j'(x_n)\right\}\sqrt{1-t_i^2}\sum_{k=1}^{M}T_j(t_k)\,\delta^3(t_k)$$

$$\times\sum_{l=1}^{L}T_l'(t_\Delta)\sum_{\Delta=1}^{L}T_l(t_\Delta)\left[\alpha^{t+\Delta t}p(t_\Delta)+(1-\alpha)^{t}p(t_\Delta)\right]$$

$$-\frac{4\Delta t}{LM\tau_c}\left[\gamma(x_n,1)-\gamma(x_n,-1)\right]\sum_{j=1}^{M}T_j'(x_n)\sum_{k=1}^{M}T_j(t_k)\,\delta^3(t_k)$$

$$\times\sum_{l=1}^{L}T_l'(t_\Delta)\sum_{\Delta=1}^{L}T_l(t_\Delta)\left[\alpha^{t+\Delta t}p(t_\Delta)+(1-\alpha)^{t}p(t_\Delta)\right];\quad t_{k,\Delta}=-\cos\left[\frac{\pi(2(k,\Delta)-1)}{2L}\right]$$

$k,\Delta=1,\dots,L$; $L\equiv M\equiv N+1$; $t_i=-\cos\left[\frac{\pi(2i-1)}{2N}\right]$, $i=1,\dots,N$; $x_n=-\cos(\pi n/N)$, $n=1,\dots,N-1$ (3.29a)

where $\tau_c\equiv\eta G^2/p_0^3$ is the characteristic time [23]. On the other hand $1-\omega<|x|\le 1$, we must use

$$\dot\delta(x)=\sum_{j=0}^{M-1}(1-\tfrac{1}{2}\delta_{j0})T_j(x)\sum_{\Delta=1}^{L}T_j(t_\Delta)\,\dot\delta(t_\Delta) \qquad (3.29b)$$

in Eq. (3.28) in order to guarantee that $\dot\delta$ is smoothly continued from $(\delta^3 p')'$ in the penetrated region. In the non-penetrated region we must impose $\dot p = 0$ to allow solution for the unknown ${}^{t+\Delta t}\dot\delta(t_s)$. Thus, we solve Eqns. (3.29a) in the penetrated region and Eq. (3.28) in the non-penetrated zone, subject to the constraints

$$^{t+\Delta t}p(t_\Delta)=0 \quad \begin{cases}\Delta=1,\dots,LF\\ \Delta=RF+2,\dots,L\end{cases} \qquad (3.30a)$$

$$^{t+\Delta t}\dot\delta(1)-{}^{t+\Delta t}\dot\delta(-1),\ {}^{t+\Delta t}p(t_{\frac{1}{2}})=p_0 \qquad (3.30b)$$

Note that Eqns. (3.30b) are the constraints of crack closure and constant borehole pressure. Simplicity and economy may be achieved by defining the appropriate matrices*:

$$A_{rl} \equiv \left(1-\tfrac{1}{2}\delta_{l1}\right) T_{l-1}(x_r);\ r = LF,\ldots,RF\ ,\ l=1,\ldots,L \tag{3.31a}$$

$$A'_{lk} \equiv \frac{2}{L}\, T_{l-1}(t_k) \qquad l=1,\ldots,L\ ;\ k=1,\ldots,L \tag{3.31b}$$

$$B_{ri} \equiv \frac{\pi}{NT_c}\,\gamma_D(x_r,t_i)\sqrt{1-t_i^2} \qquad r=LF,\ldots,RF\ ;\ i=LF,\ldots,RF+1$$

$$\equiv \frac{\pi}{N}\,\gamma_D(x_r,t_{i-L})\sqrt{1-t_{i-L}^2} \qquad r=LF,\ldots,RF\ ;\ i=L+1,\ldots,L+LF-1$$
$$i=L+RF+2,\ldots,L+N$$

$$\equiv \frac{\pi}{NT_c}\,\gamma_D(x_{r-L},t_i)\sqrt{1-t_{i-L}^2} \qquad r=L+1,\ldots,L+LF-1\ ;\ r=L+RF+1,\ldots,L+N-1$$
$$i=LF,\ldots,RF+1$$

$$\equiv \frac{\pi}{N}\,\gamma_D(x_{r-L},t_{i-L})\sqrt{1-t_{i-L}^2} \qquad r=L+1,\ldots,L+LF-1\ ;\ r=RF+L+1,\ldots,L+N-1 \tag{3.31c}$$
$$i=L+1,\ldots,L+LF-1\ ;\ i=L+RF+2,\ldots,L+N$$

*All of the matrices are $2M \times 2M$; any undefined elements are zero.

$$B'_{ri} \equiv \delta_{ri} \frac{\pi}{N\tau_c} \sum_{j=1}^{N} \gamma_D(x_r, t_j)\sqrt{1-t_j^2} \qquad r = LF,\ldots,RF \quad i = 1,\ldots,N$$

$$\equiv \delta_{(r-L)i} \qquad r = L+LF,\ldots,L+RF \quad i = 1,\ldots,N$$

$$\equiv \delta_{ri} \frac{\pi}{N} \sum_{j=1}^{N} \gamma_D(x_r, t_j)\sqrt{1-t_j^2} \qquad \begin{matrix} r = L+1,\ldots,L+LF-1 \quad r = L+RF+1,\ldots,L+N-1 \\ i = L+1,\ldots,L+N \end{matrix}$$

$$\equiv -\delta_{ri} \qquad r = L+LF,\ldots,L+RF \quad i = L+1,\ldots,L+N \qquad (3.31d)$$

$$C_{kj} \equiv T'_j(t_k) \qquad k = 1,\ldots,M \quad j = 1,\ldots M$$

$$\equiv \left(1-\tfrac{1}{2}\delta_{j(M+1)}\right) T_{j-M-1}(t_{k-M}) \qquad k = M+1,\ldots,2M;\ j = M+1,\ldots,2M \qquad (3.31e)$$

$$C'_{rj} \equiv T'_j(x_r) \qquad r = 1,\ldots,N-1 \quad j = 1,\ldots,M$$

$$\equiv \left(1-\tfrac{1}{2}\delta_{j(M+1)}\right) T_{j-M-1}(x_{r-M}) \qquad r = M+1,\ldots,M+N-1 \quad j = M+1,\ldots,2M \qquad (3.31f)$$

$$D_{jk} \equiv \frac{2}{L} T_j(t_k) \qquad j,k = 1,\ldots,L$$

$$\equiv \frac{2}{L} T_{j-L-1}(t_{k-L}) \qquad j,k = L+1,\ldots,2L \tag{3.31g}$$

From here on, all undefined elements are part of a unit matrix (e.g., $E_{kl} = \delta_{kl}$), the rest being given by

$$E_{kl} \equiv T_l'(t_k) \qquad k,l = 1,\ldots,L \tag{3.31h}$$

$$F_{l\Delta} \equiv \frac{2}{N} T_l(t_\Delta) \qquad l,\Delta = 1,\ldots,L \tag{3.31i}$$

$$G_{rj} \equiv \frac{\delta_{rj}}{\tau_c}\left[\gamma(x_{r,1}) - \gamma(x_{r,-1})\right] \qquad r = LF,\ldots,RF \qquad j = 1,\ldots,L$$

$$\equiv \delta_{rj}\left[\gamma(x_{r-L,1}) - \gamma(x_{r-L,-1})\right] \qquad r = L+1,\ldots,L+LF-1$$
$$j = L+1,\ldots,2L \tag{3.31j}$$

$$H_{\Delta\phi} \equiv -\delta_{\Delta\phi}\,\mathrm{sign}(t_\Delta) + \delta_{\left(\frac{L}{2}\right)\Delta}\,\mathrm{sign}(t_\Delta) \qquad \Delta,\phi = 1,\ldots,l \tag{3.31k}$$

$$S_{ij} \equiv \delta_{ij} + \delta_{\left(\frac{M}{2}\right)j}\,\mathrm{sign}(t_i) \qquad i,j = 1,\ldots,L \tag{3.31l}$$

$$T_{jk} \equiv -\delta_{jk}\,\mathrm{sign}(t_k) \qquad j,k = 1,\ldots,L \tag{3.31m}$$

$$\Delta_{jk} \equiv \delta_{jk}\delta^3(t_j) \qquad j,k = 1,\ldots,L \tag{3.31n}$$

Use of the following secondary matrices lends further simplification:

$$\underline{M1} \equiv \underline{D}\,\underline{S} \;, \quad \underline{M2} \equiv \underline{T}\,\underline{E}\,\underline{F}\,\underline{H} \;, \quad \underline{M3} \equiv \underline{C'}\,\underline{D}\,\underline{S} \;,$$

$$\underline{M4} \equiv \underline{A}\,\underline{A'} \;, \quad \underline{M5} \equiv \underline{B}\,\underline{C} - \underline{B'}\,\underline{C'} \;, \quad \underline{M6} \equiv \underline{T}\,\underline{C'}\,\underline{F}\,\underline{H} \tag{3.32}$$

Now M1, M2, M3, and M6 need be computed only once; only M4 and M5 are time dependent. The resulting system of equations is:

$$\{\underline{M4} - \alpha\Delta t\,\underline{M5}\,\underline{M1}\,\underline{\Delta}\,\underline{M2} + \alpha\Delta t\,\underline{G}\,\underline{M3}\,\underline{\Delta}\,\underline{M2}\}^{t+\Delta t}\underline{U}$$

$$= \{\underline{M4} - (1-\alpha)\Delta t\,\underline{M5}\,\underline{M1}\,\underline{\Delta}\,\underline{M2} + (1-\alpha)\Delta t\,\underline{G}\,\underline{M3}\,\underline{\Delta}\,\underline{M2}\}^{t}\underline{U} \tag{3.33a}$$

or

$$\underline{M}^{t+\Delta t}\underline{U} = \underline{R} \tag{3.33b}$$

where the vector of unknown variables is:

$$U_\ell \equiv \begin{matrix} p(t_\ell) & \ell = 1, \ldots, M \\ \dot{\delta}(t_{\ell-M}) & \ell = M+1, \ldots, 2M \end{matrix} \tag{3.33c}$$

The constraints (Eq. (3.30)) are imposed as follows: Eq. (3.30a) by setting $M_{sj} = \delta_{sj}$, $R_s = 0$ for $s = 1, \ldots, LF-1$, $s = RF+3, \ldots, L$ and $j = 1, \ldots, 2L$; Eq. (3.30b) by setting $B(N,1) = 1$ and $B(N,L) = -1$

before computing M5 and Eq. (3.30c) by setting $M_{\frac{L}{2}i} = \delta_{\frac{L}{2}j}, j=1,\ldots,$ 2L. The time step size (Δt) is computed, based on the velocity of the fluid front, so as to bring the front to the next node x_r at $t = t + \Delta t$. Thus, we employ

$$\frac{\Delta t}{\tau_c} = \frac{[\delta^2 p'](x_{RF})}{x_{RF+1} - x_{RF}} \tag{3.34}$$

Typical results for the fluid front advancement problem are shown in Figure (3.20). The fluid was allowed to advance to the crack tips, filling the crack entirely, and the pressure was then allowed to build up for some time afterward. The pressure distribution behaves as one might expect: the curves become steeper near the tips as time progresses and the crack fills out very quickly. One notable feature is the rather sudden increase in the fluid pressure at the tip just after the fluid reaches it (Figures 3.20e, f).

The curves showing the crack opening rate (Figures 3.20 (n-t)) undergo a change of character between the initial step (Figure (3.20n)) and the final step (Figure (3.20t)). Before the fluid front reaches the tip (Figure (3.20q)), $\dot{\delta}$ shows high narrow peaks near, but somewhat behind the points corresponding to the location of the fluid fronts. This phenomenon seems to be consistent with the large pressure gradient that develops at the fluid fronts. After the fluid fills the crack, the peaks broaden out and the overall magnitude begins to decline (Figures (3.20 (r-t)). The shape of the initial curve (Figure 3.20(n)) is not unlike that of the final curve,

although the initial curve has a much smaller magnitude.

The velocity of the fluid front is compared to the corresponding velocity (also calculated via Eq. 3.34) of the same fluid flowing between two parallel plates, with a space of δ (t;x=0) between them and being driven by a uniform pressure gradient of p'/p_0=1.0 in Figure (3.21). Initially there is a large discrepancy between this latter velocity (dashed curve) and the calculated fluid front velocity; this result is consistent with the difference between the crack opening at the fluid front (.4) and the maximum opening (1.0). As time progresses, this difference in velocities decreases somewhat, and seems to stabilize. We conclude that predictions of fluid penetration times based on estimates of the crack opening and fluid pressure at the borehole may be quite conservative, but are of the right order to provide useful information; thus, estimates based on τ_c (e.g., as given by Cleary [23] are useful guides to the process. We note, however, that such characteristic time estimates are easily made only in the case of a crack in a homogeneous medium, with constant borehole pressure. It would be difficult, for instance to include the effects of adjacent strata, inclusions, and the like. When our computer program is extended so that crack propagation in non-homogeneous regions can be simulated, we will probably be able to develop correlations for τ_c based on our numerical calculations.

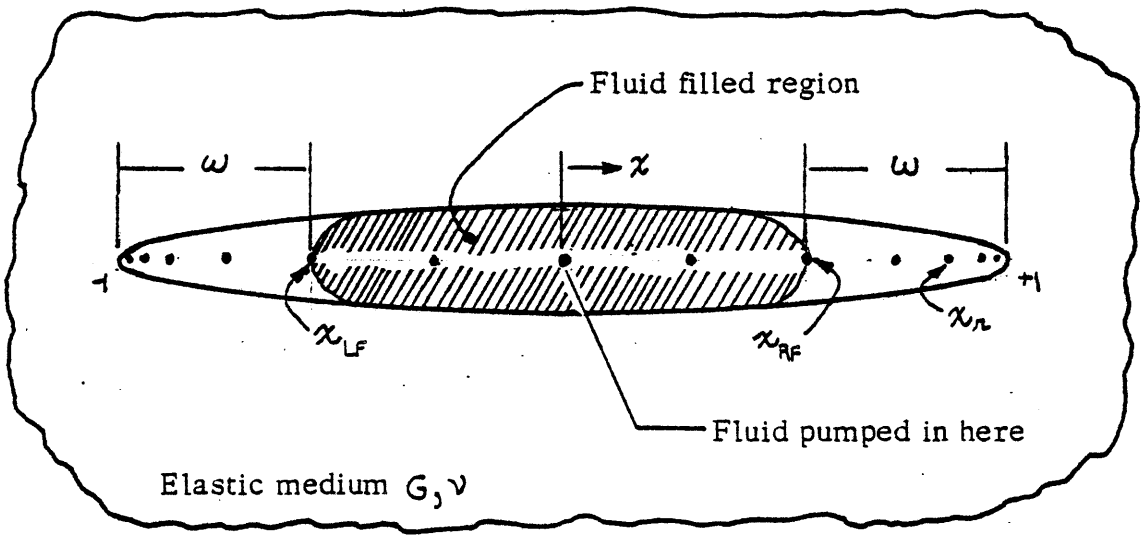


FIG. 3.19 Diagram of the fluid front advancement problem. Fluid is pumped into the crack at the borehole at constant pressure p_o. The fluid front advances from one node x_r (r= LF, RF) to the next, while the crack tips are held stationary.

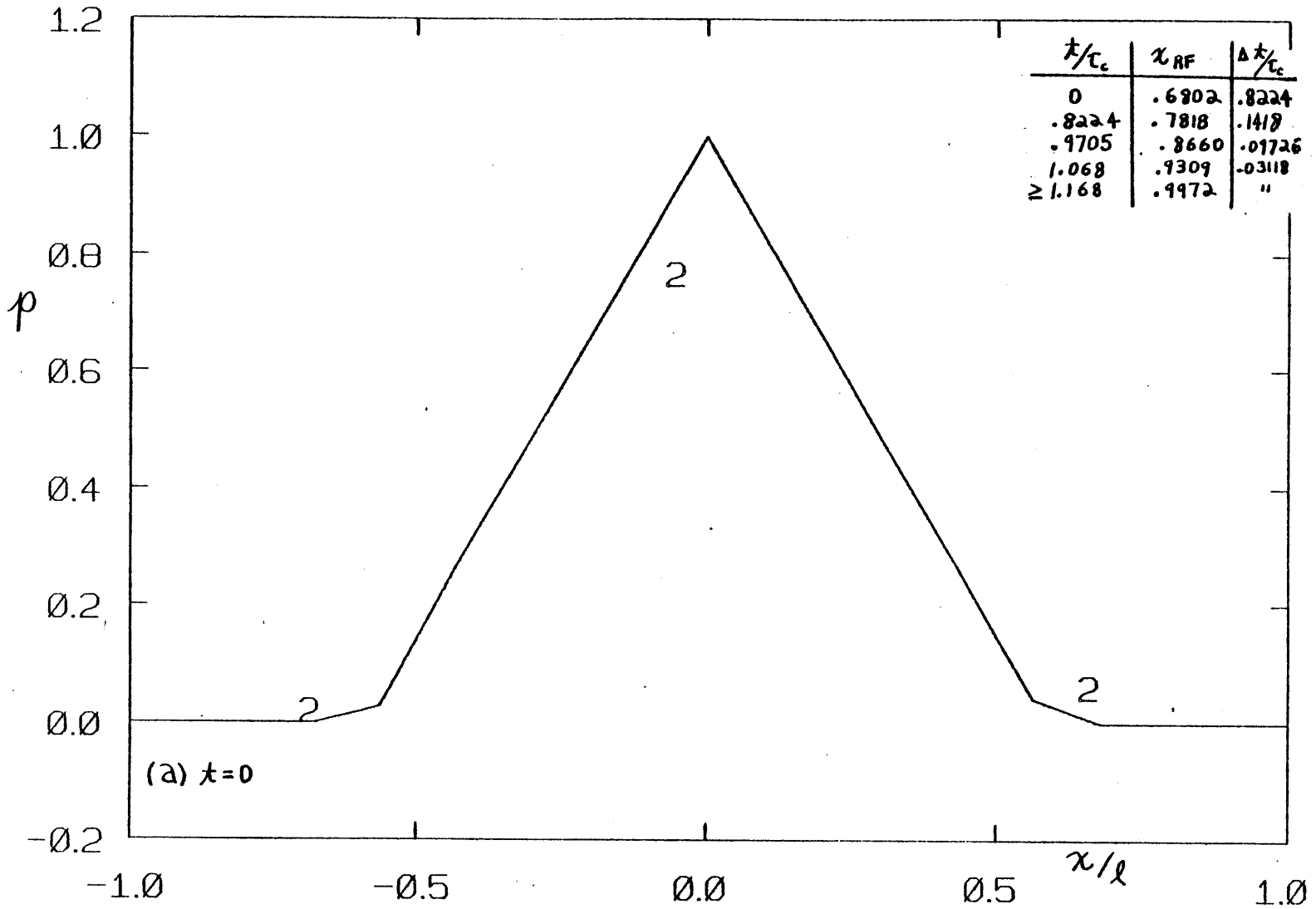
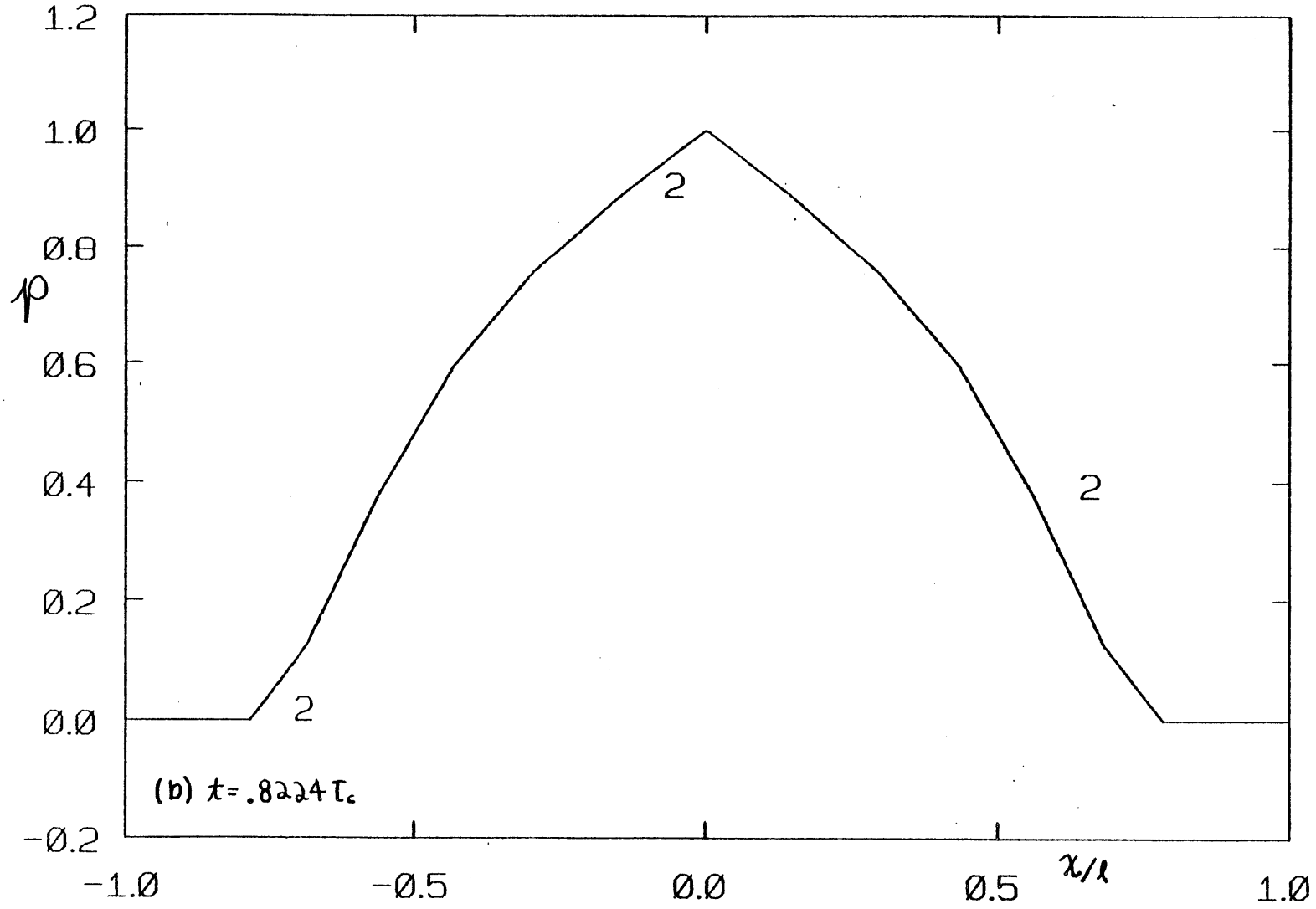
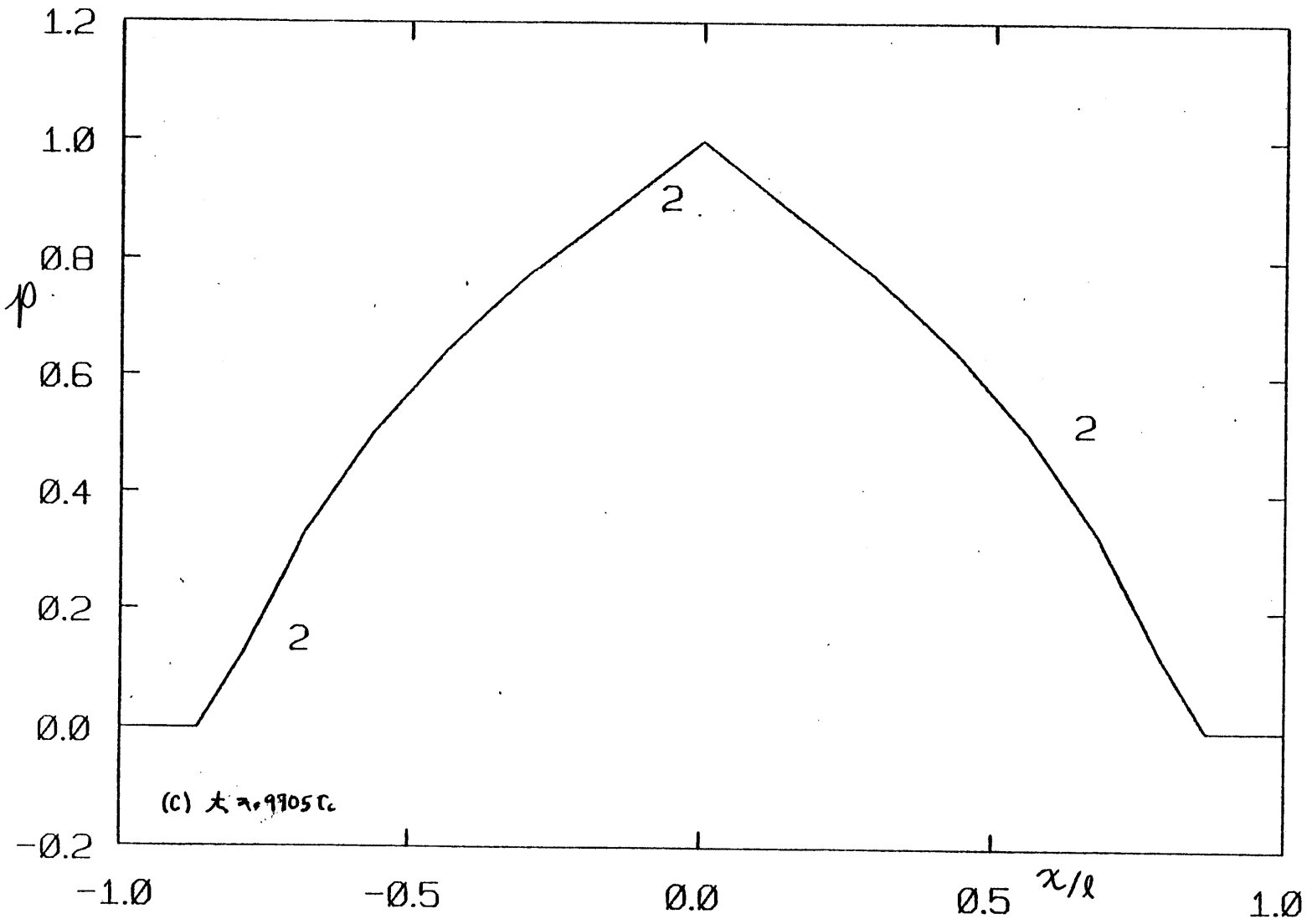
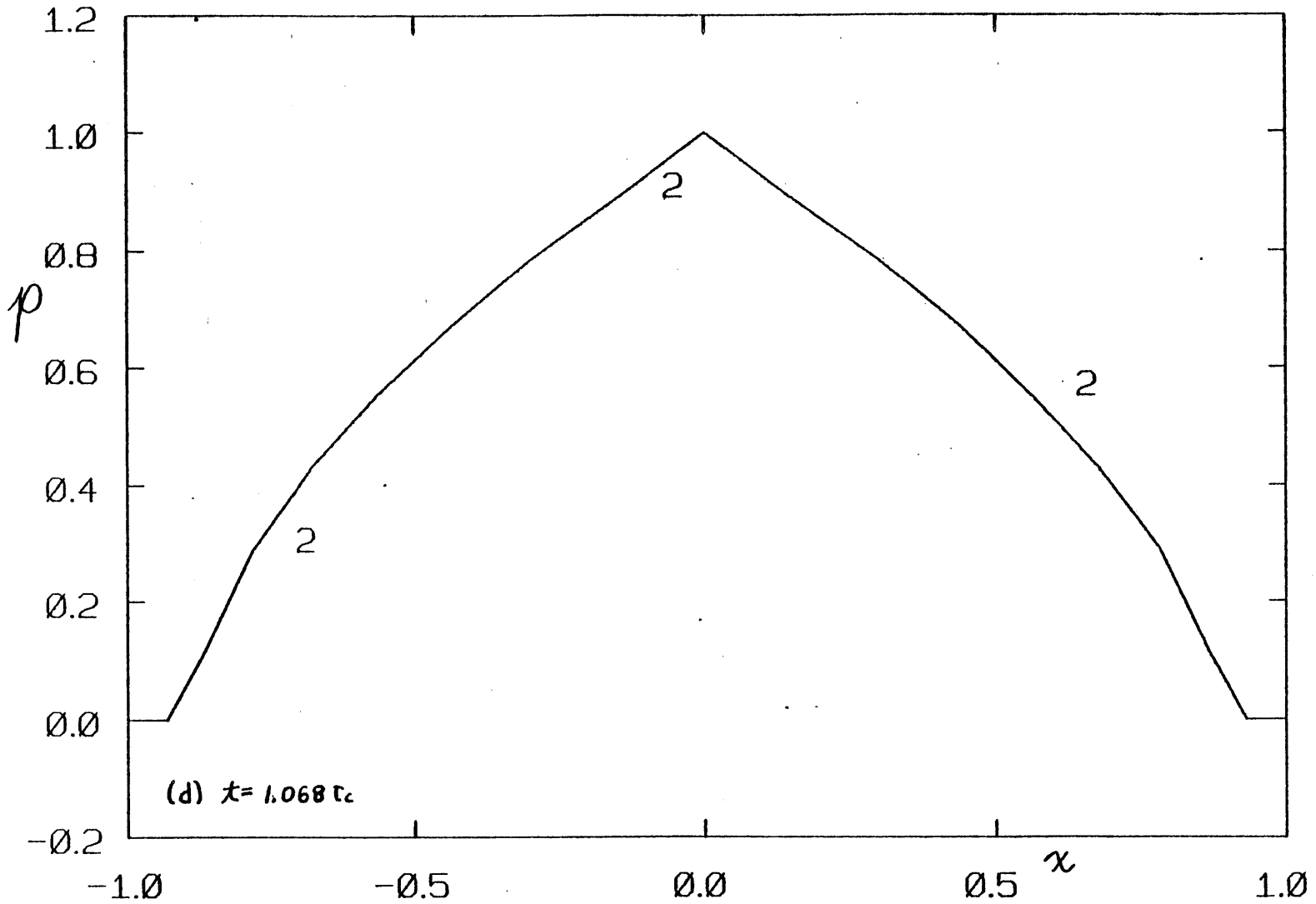
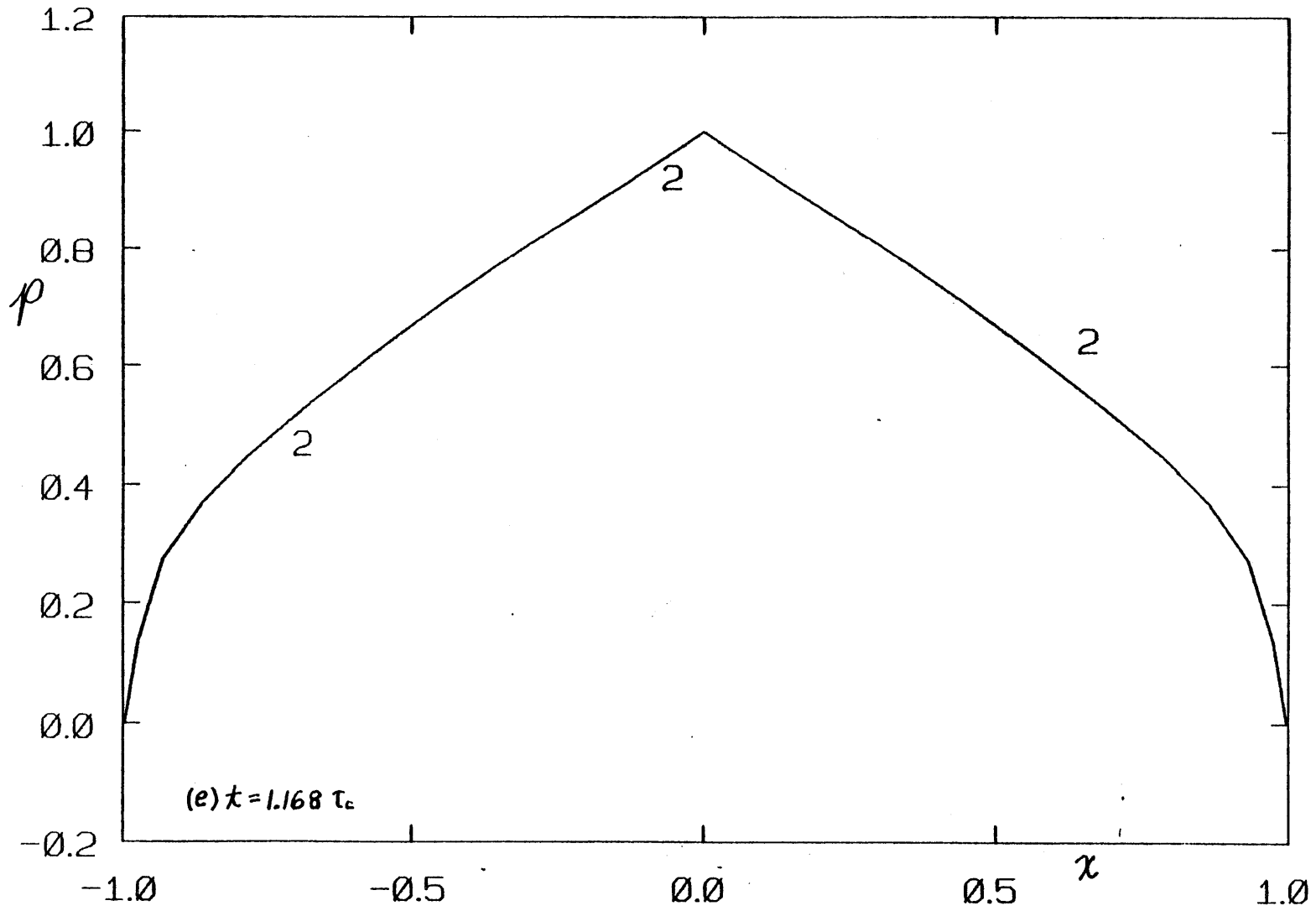


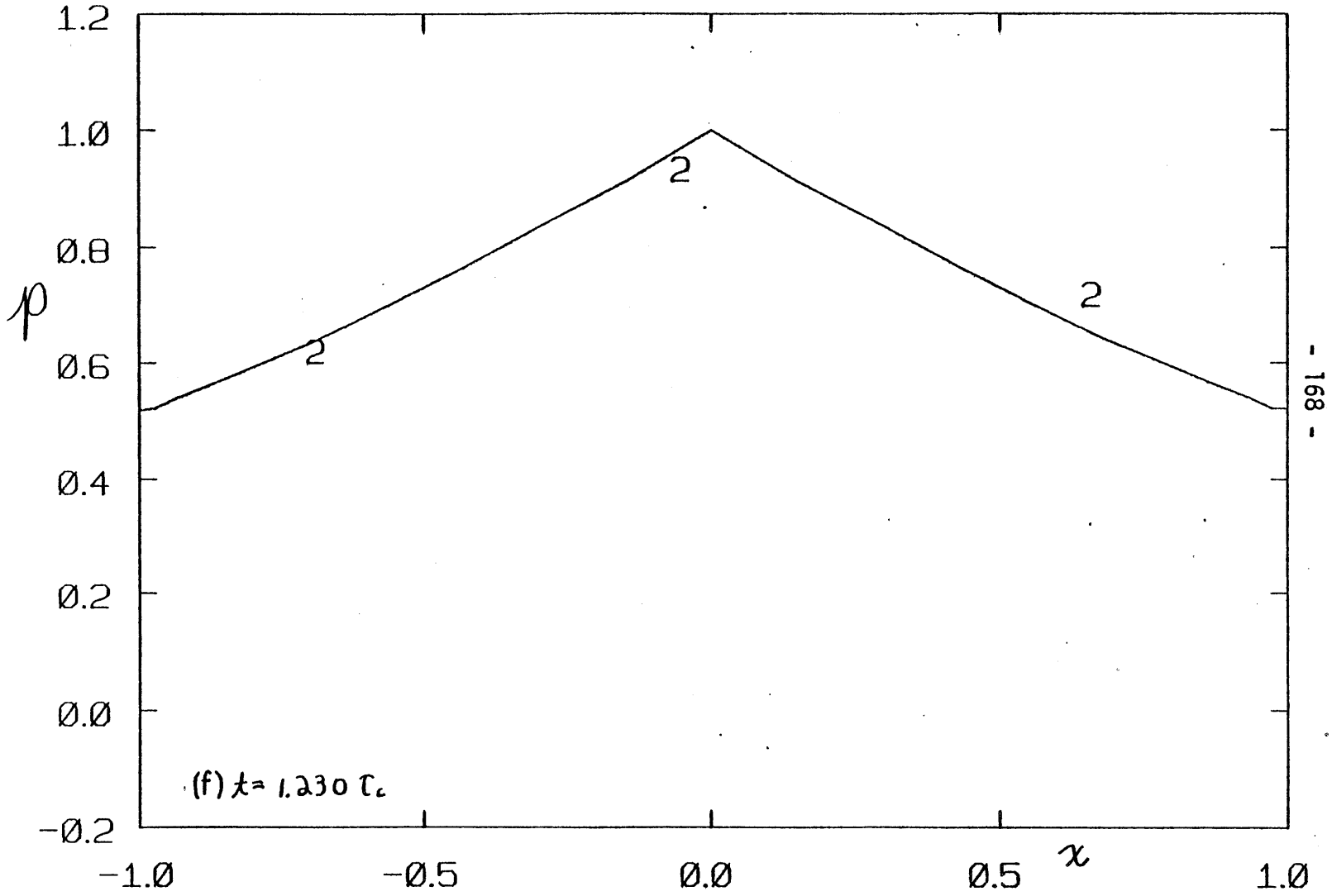
FIG. 3.20. Curves showing fluid front advancement and pressure evolution in a stationary crack. Note the rapid change in the pressure distribution when the fluid reaches the crack tips (e),(f) and the changes in the shape of the opening rate ($\dot{\delta}$) curves as the crack is being filled. (a)-(g) p ; (h)-(m) δ ;(n)-(t) $\dot{\delta}$. Here we have used $p_0/G=1$, $\hat{\eta}/G=1$, $\nu=.3$.

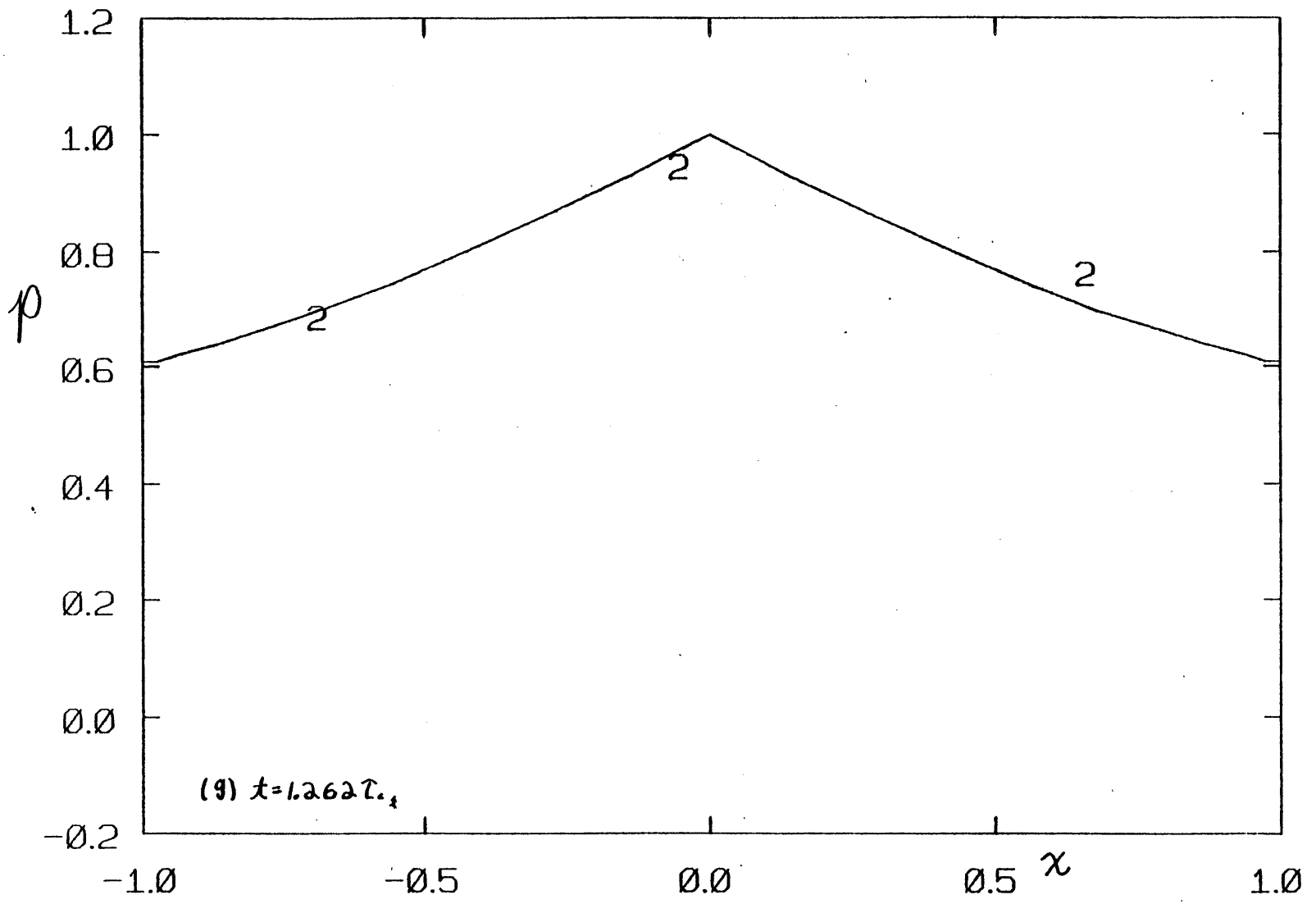

1.2
1.0
0.8
ρ
0.6
0.4
0.2
0.0
-0.2
-1.0
-0.5
0.0
0.5
1.0
x/ℓ
2
2
2
(b) t=.8224 τc

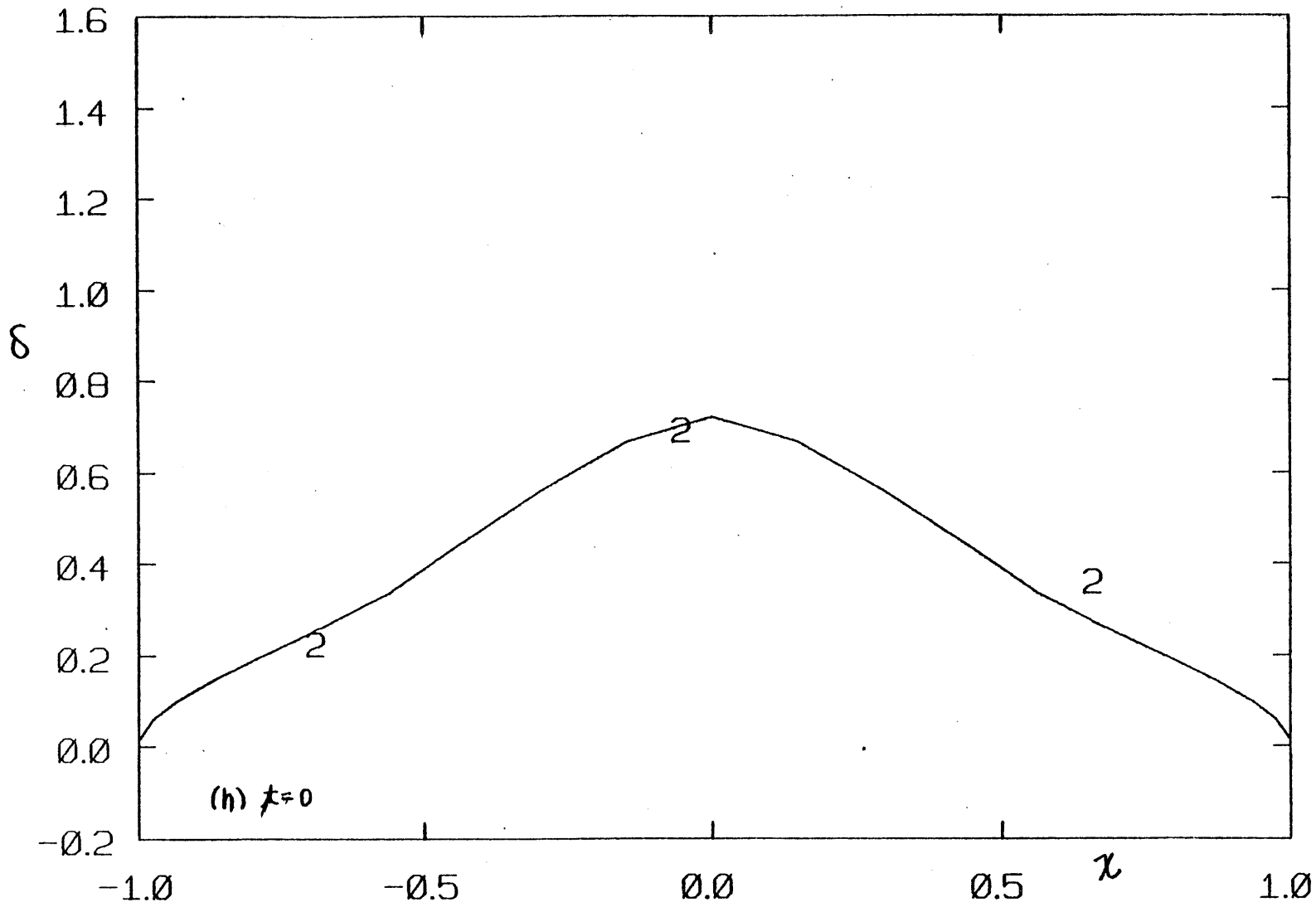

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
p
-1.0
-0.5
0.0
0.5
1.0
x/ℓ
2
2
2
(c) 大=0.9905 Cc

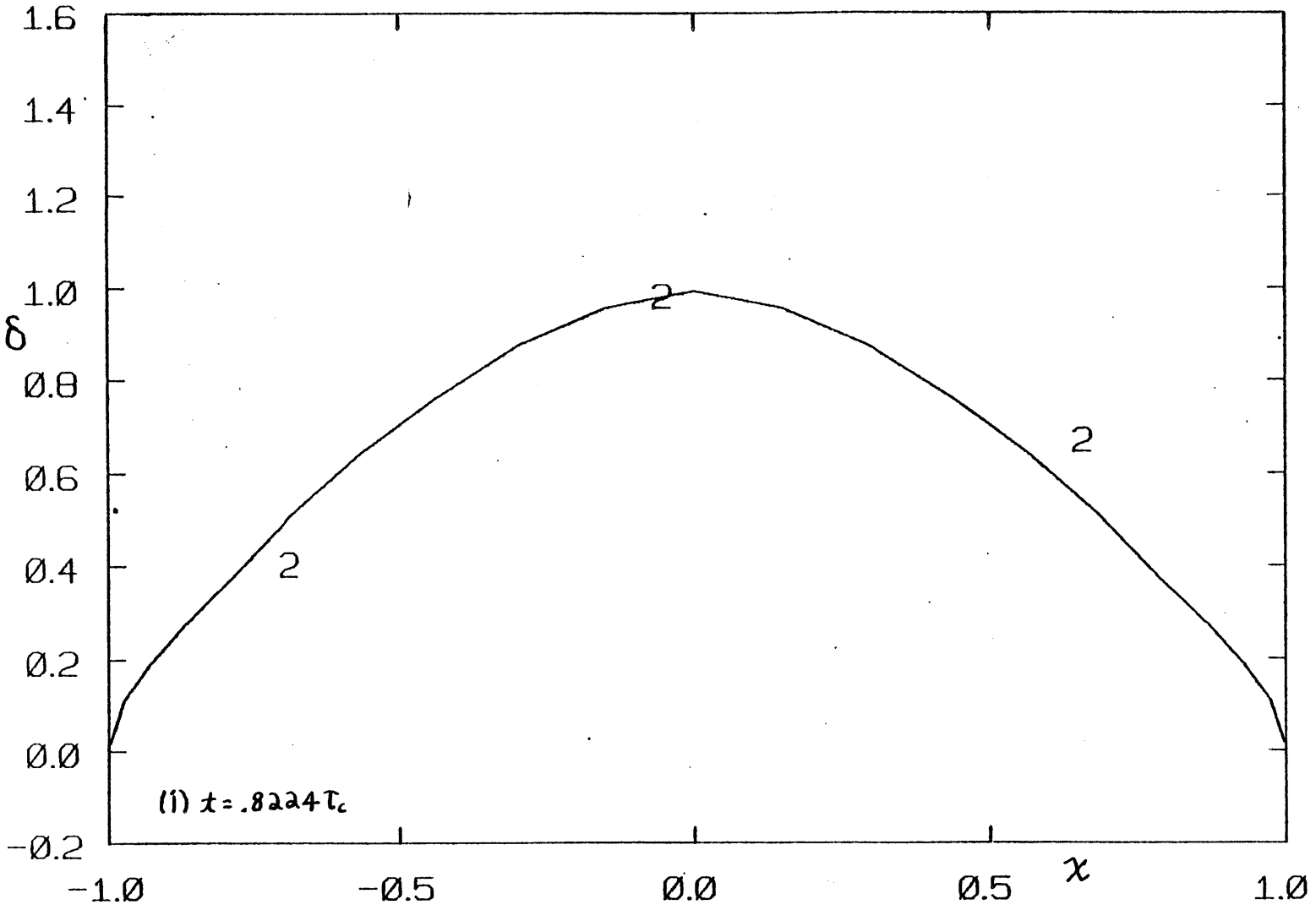

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
2
2
2
(d) t= 1.068 τc
-1.0
-0.5
0.0
0.5
1.0
x

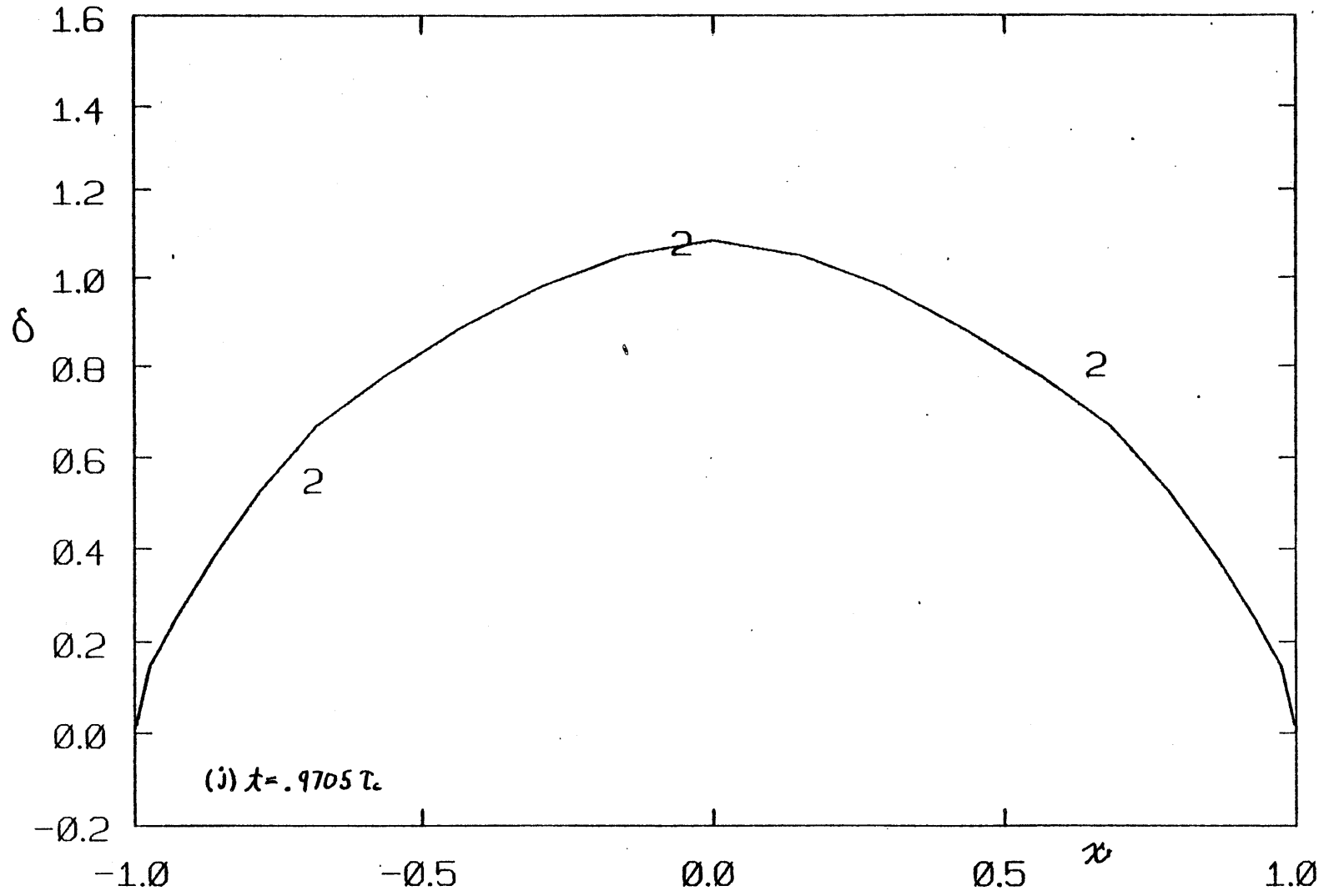

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
2
2
2
(e) t = 1.168 τc
-1.0
-0.5
0.0
0.5
1.0
χ

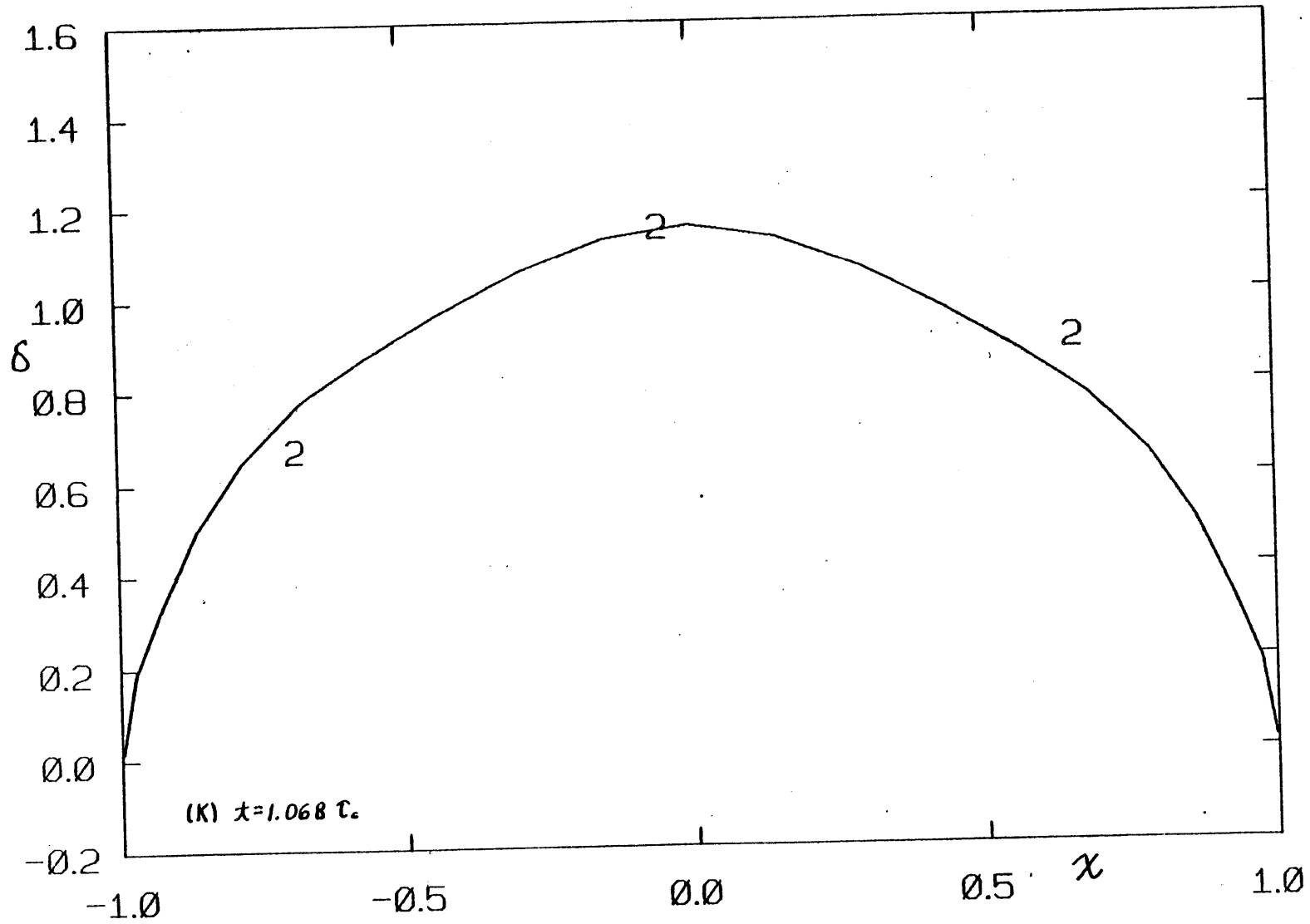

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
2
2
2
(f) t= 1.230 τc
-1.0
-0.5
0.0
0.5
1.0
x

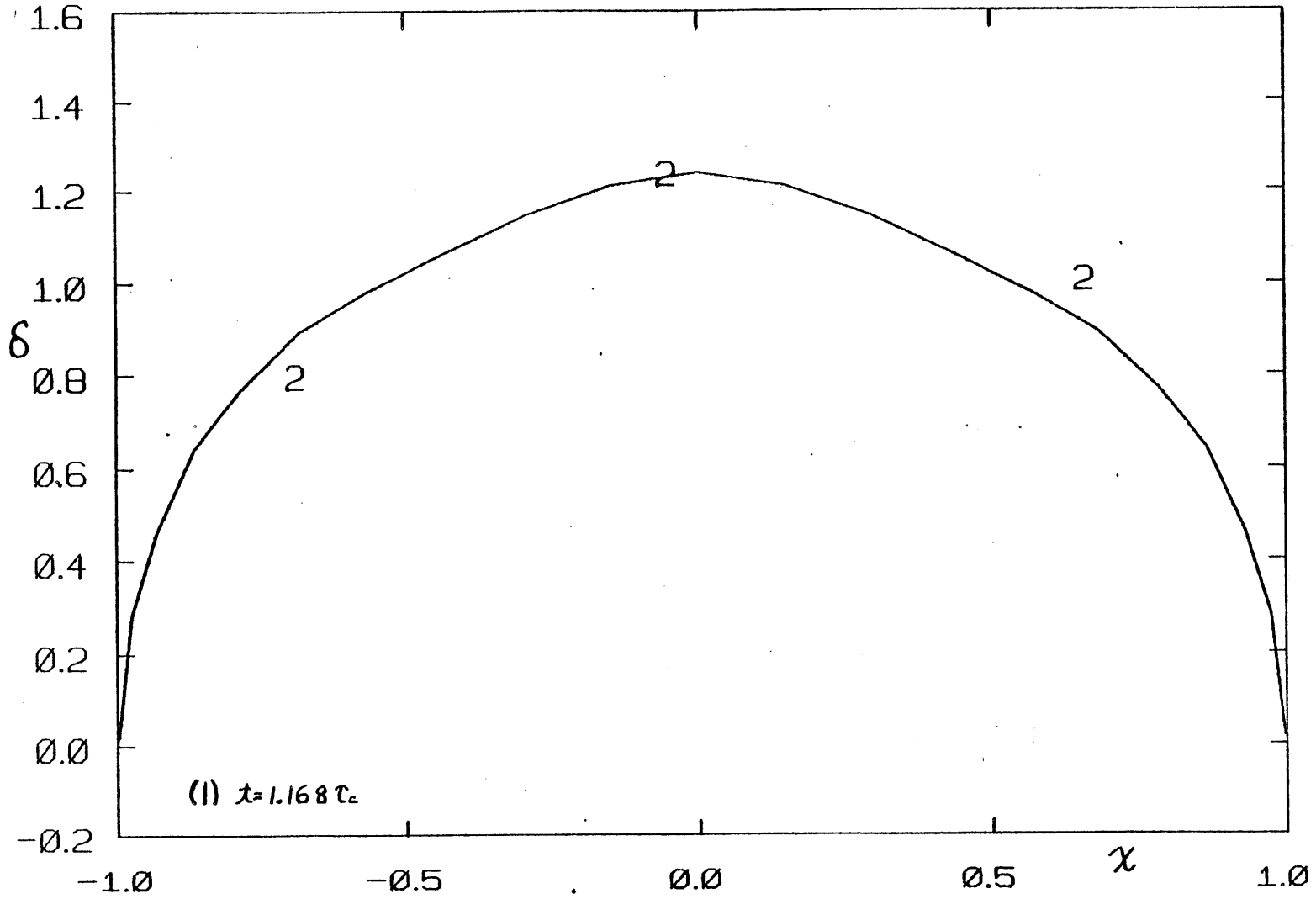

1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
ρ
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(9) t=1.262τ.

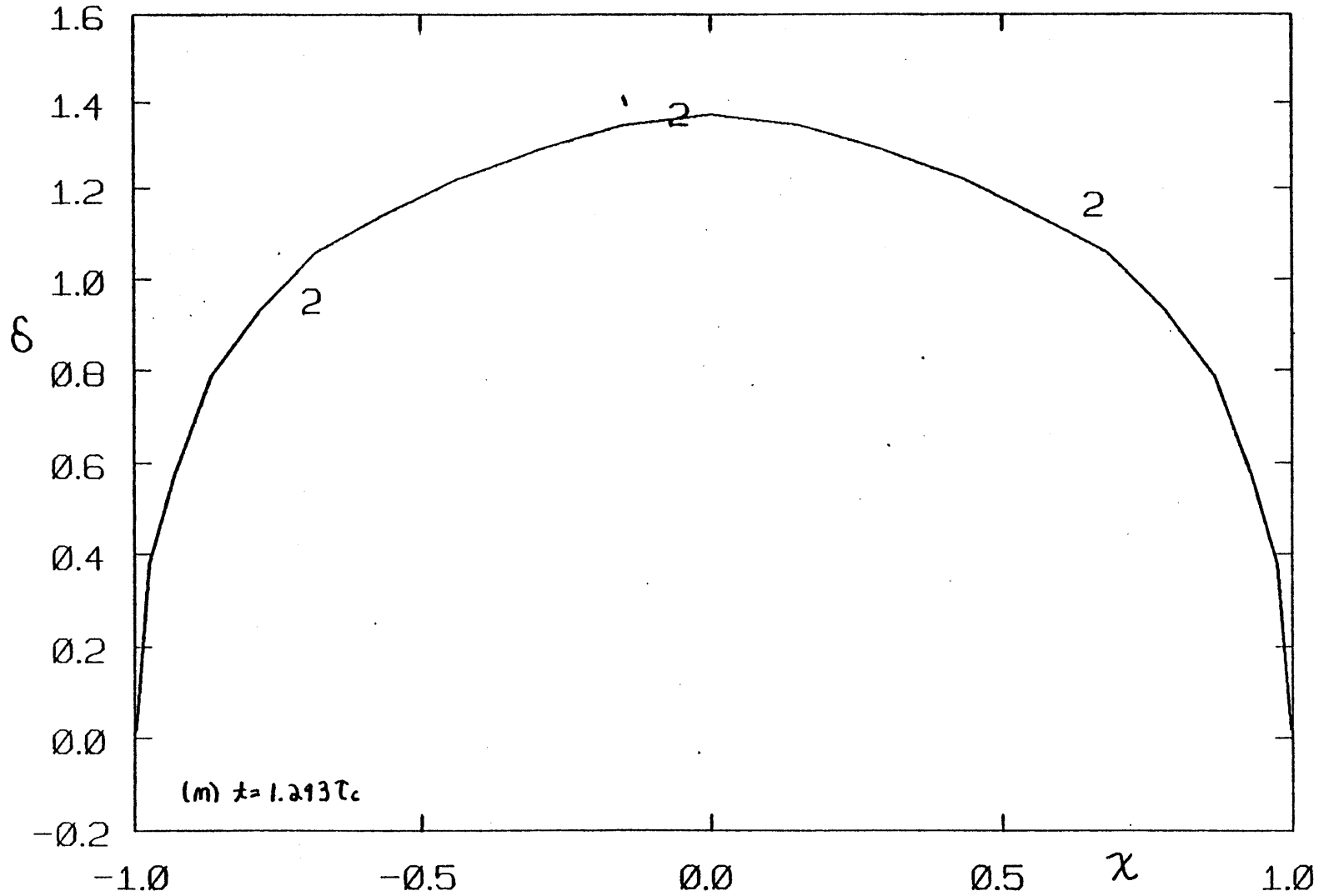

1.6
1.4
1.2
1.Ø
δ
Ø.8
Ø.6
Ø.4
Ø.2
Ø.Ø
-Ø.2
2
2
2
(h) t=0
-1.Ø
-Ø.5
Ø.Ø
Ø.5
1.Ø
x

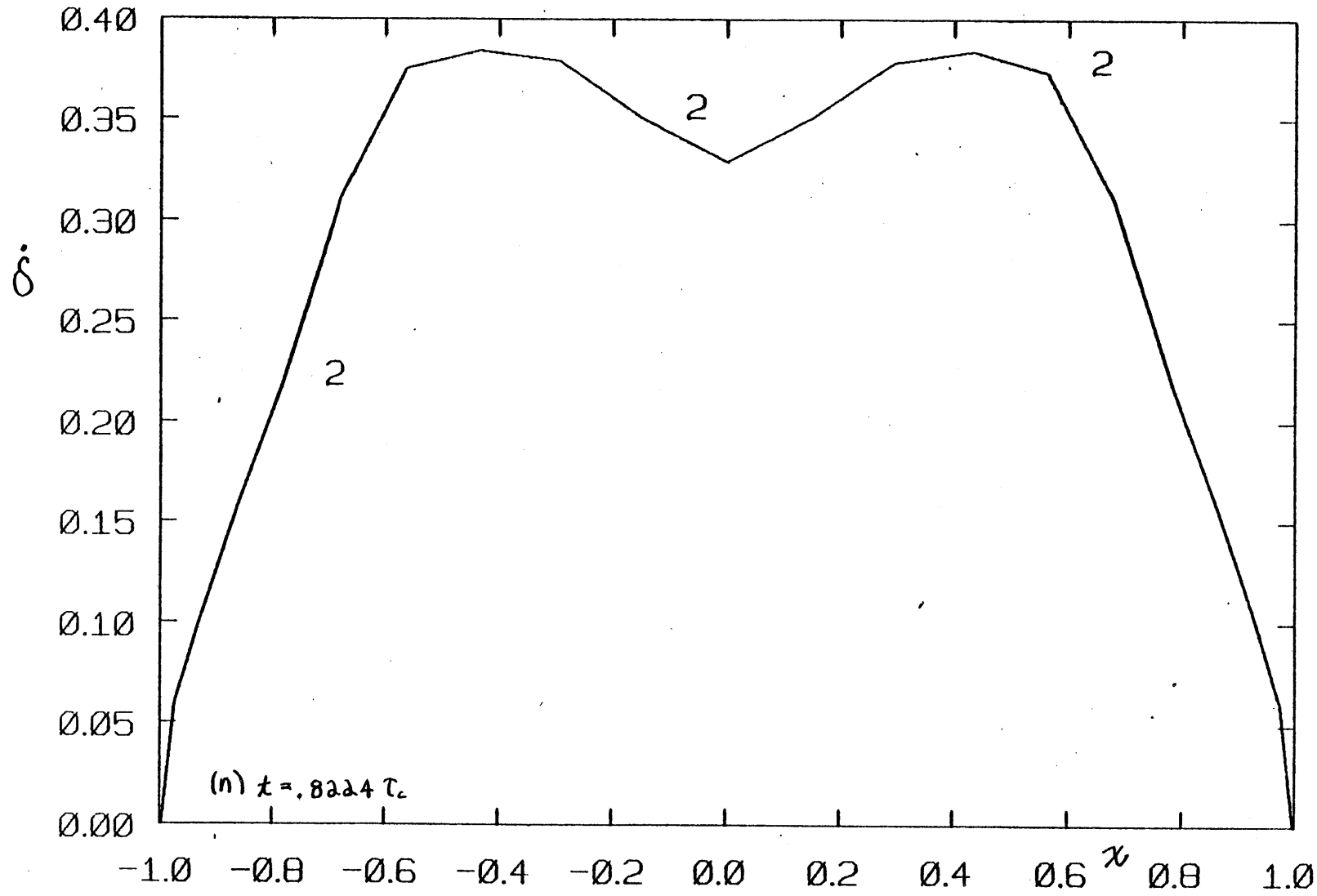

1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(1) t = .8224 τc

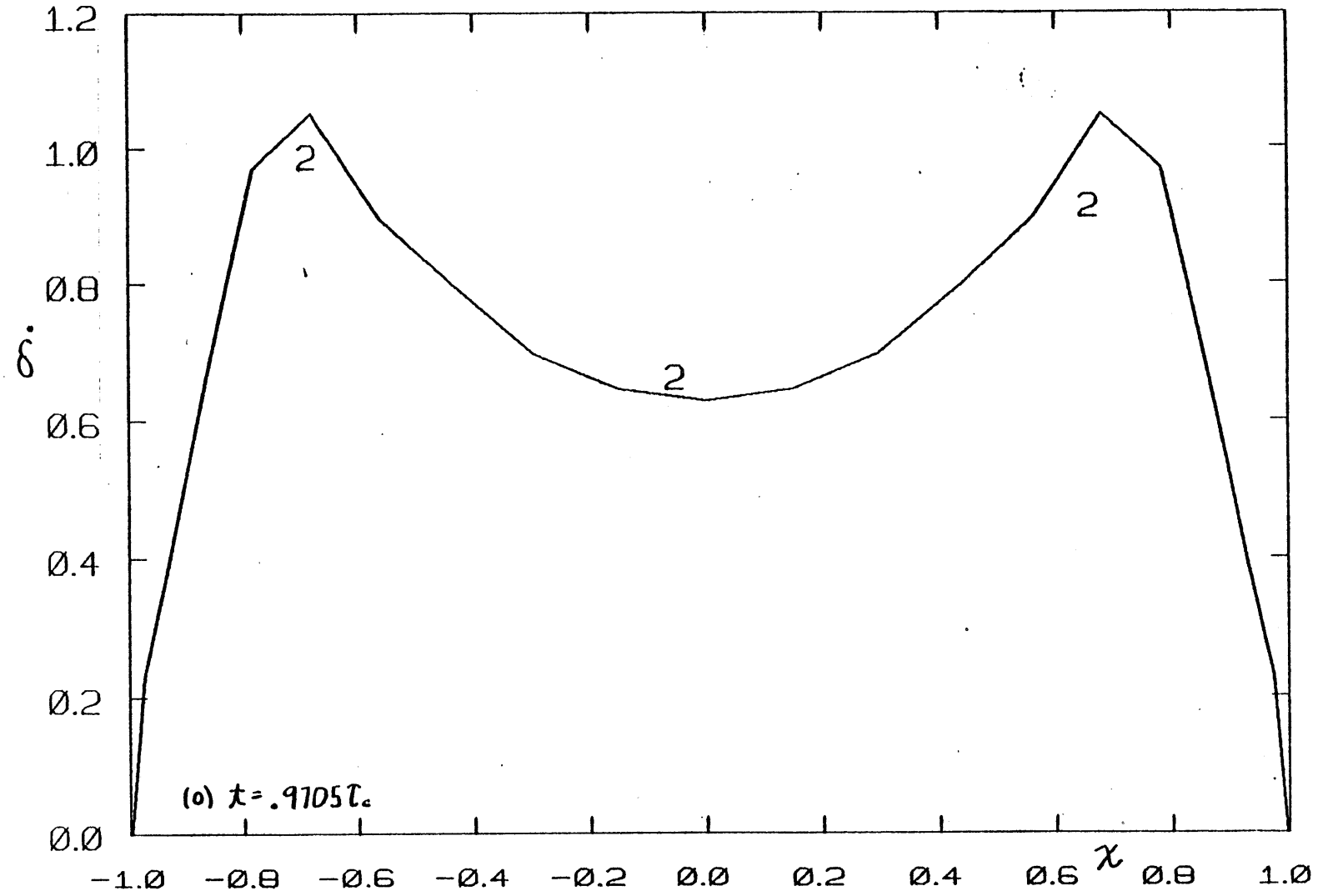

1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
x
2
2
2
(j) t=.9705 τc

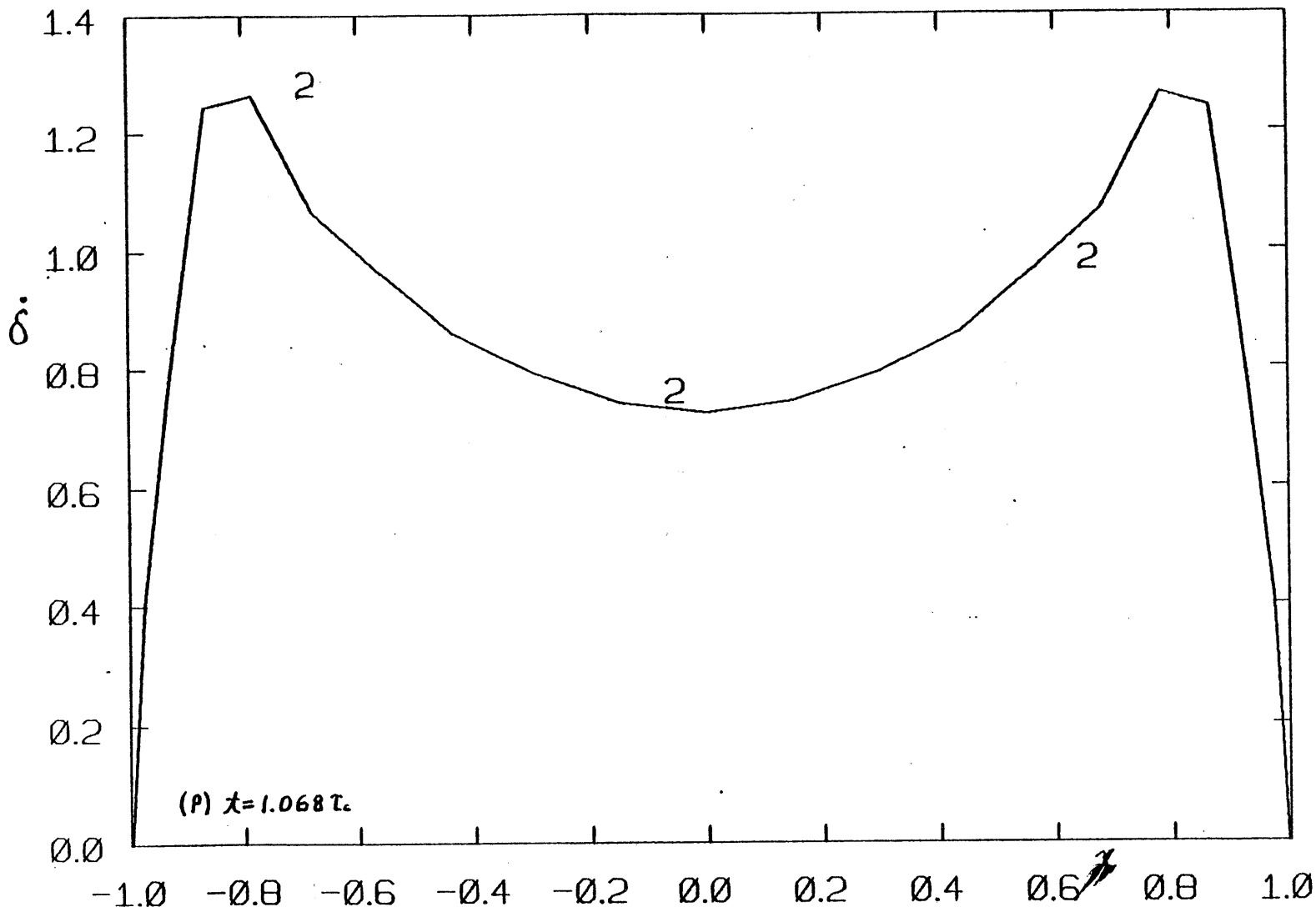

1.6
1.4
1.2
1.Ø
δ
Ø.8
Ø.6
Ø.4
Ø.2
Ø.Ø
-Ø.2
-1.Ø
-Ø.5
Ø.Ø
Ø.5
χ
1.Ø
2
2
2
(K) t=1.068 τc

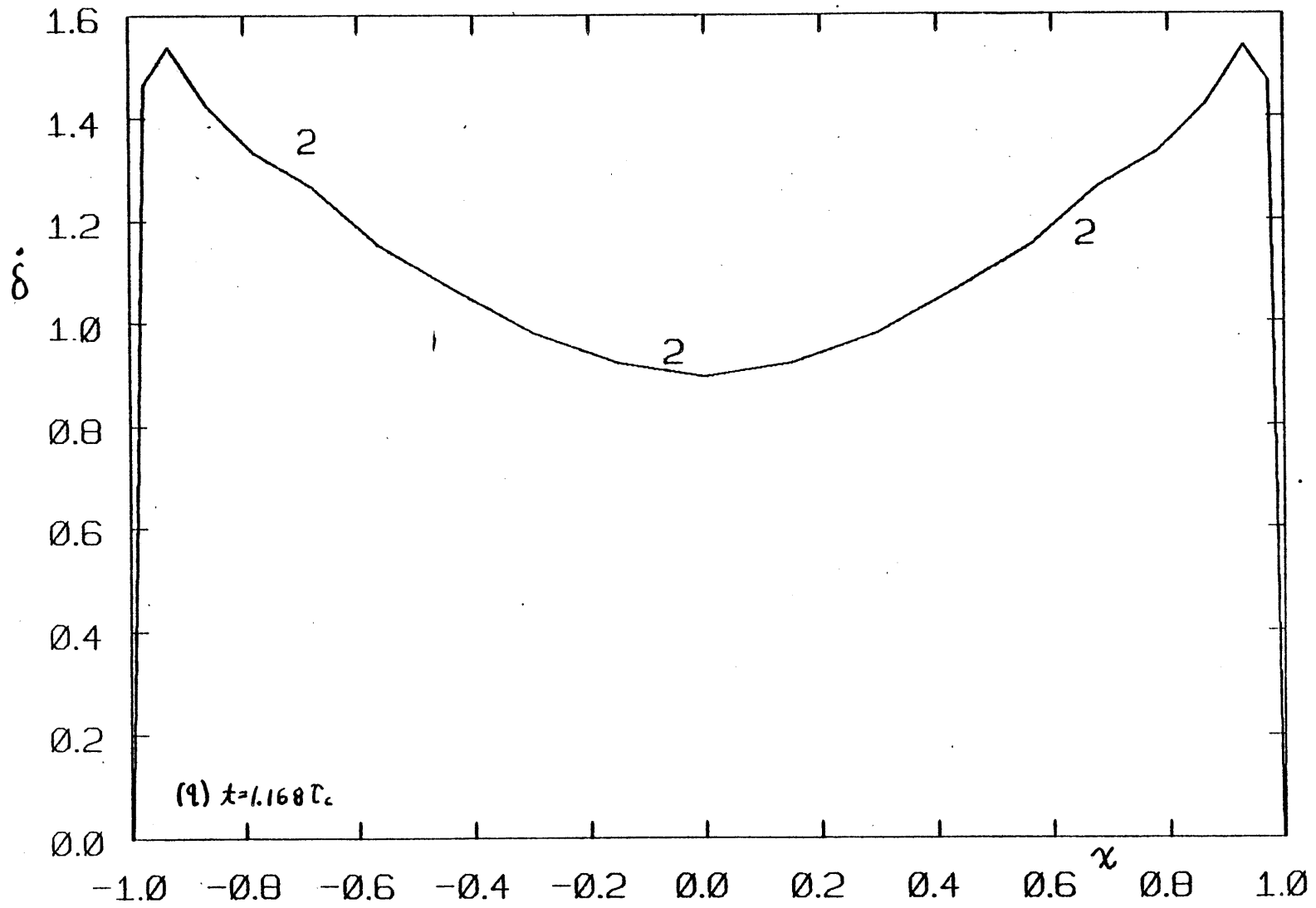

1.6
1.4
1.2
1.Ø
Ø.8
Ø.6
Ø.4
Ø.2
Ø.Ø
-Ø.2
δ
-1.Ø
-Ø.5
Ø.Ø
Ø.5
1.Ø
χ
2
2
2
(1) t=1.168 τc

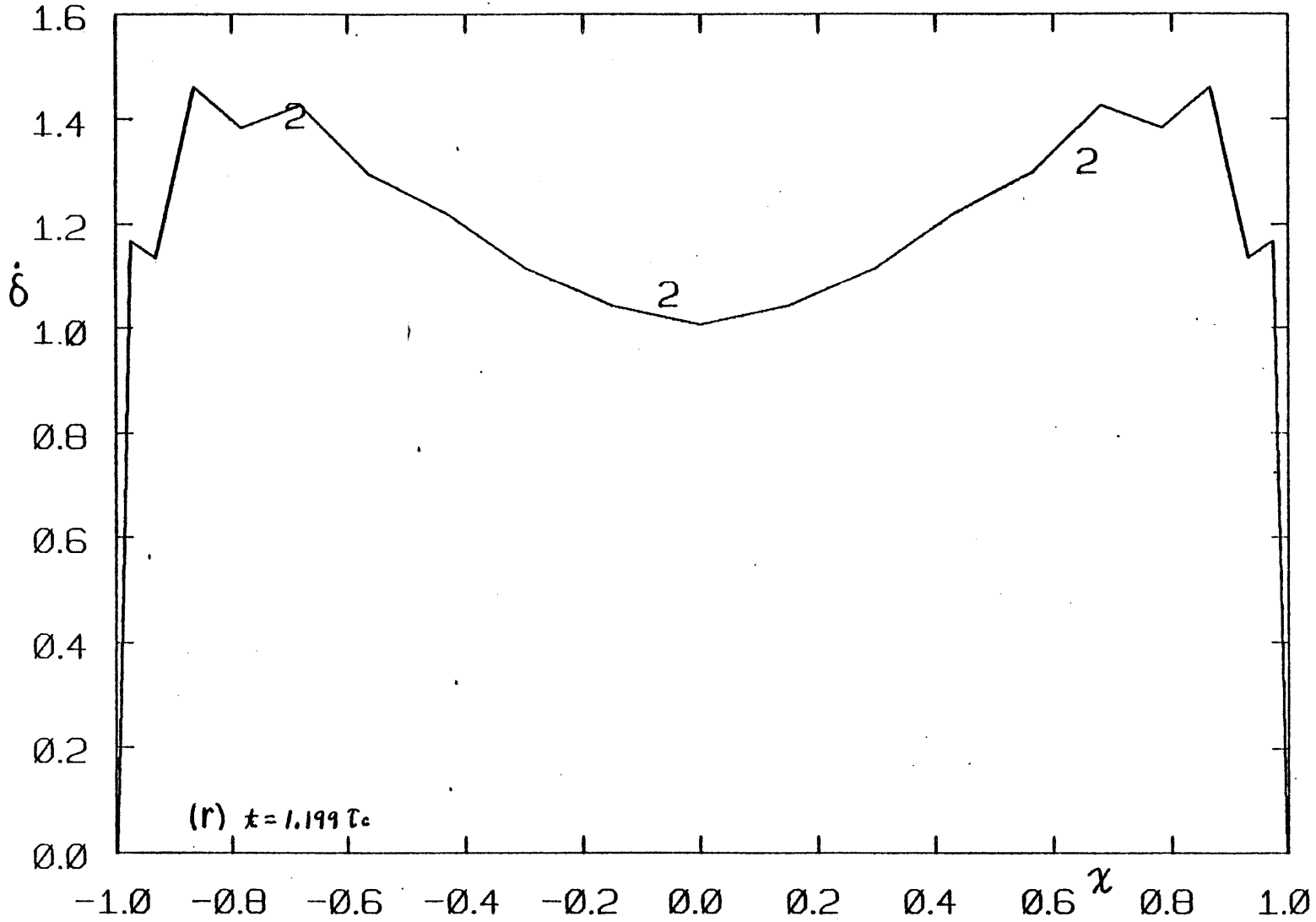

1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
δ
-1.0
-0.5
0.0
0.5
1.0
χ
2
2
2
(m) t= 1.293 τc

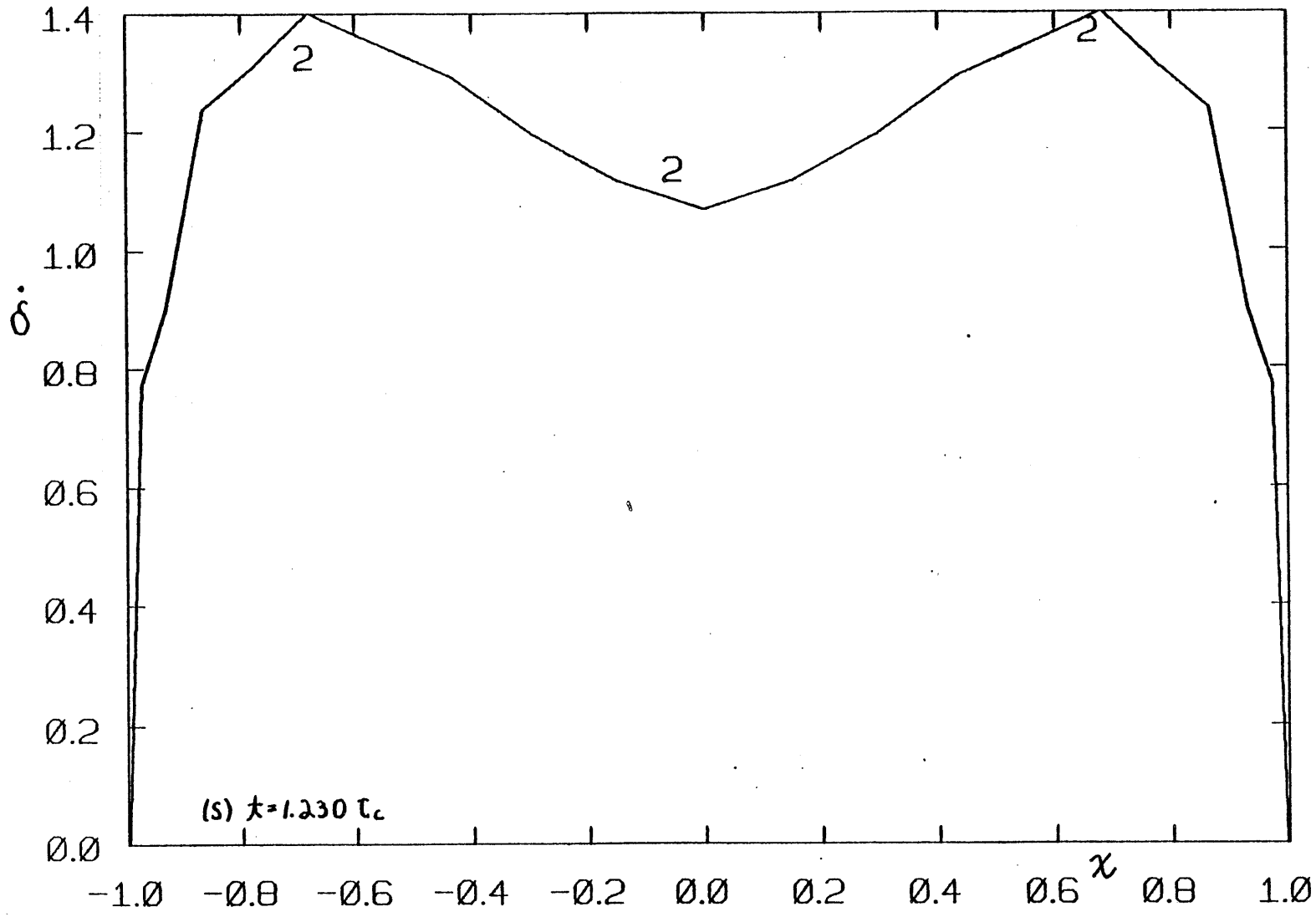

0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
δ̇
2
2
2
(n) t = .8224 τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x

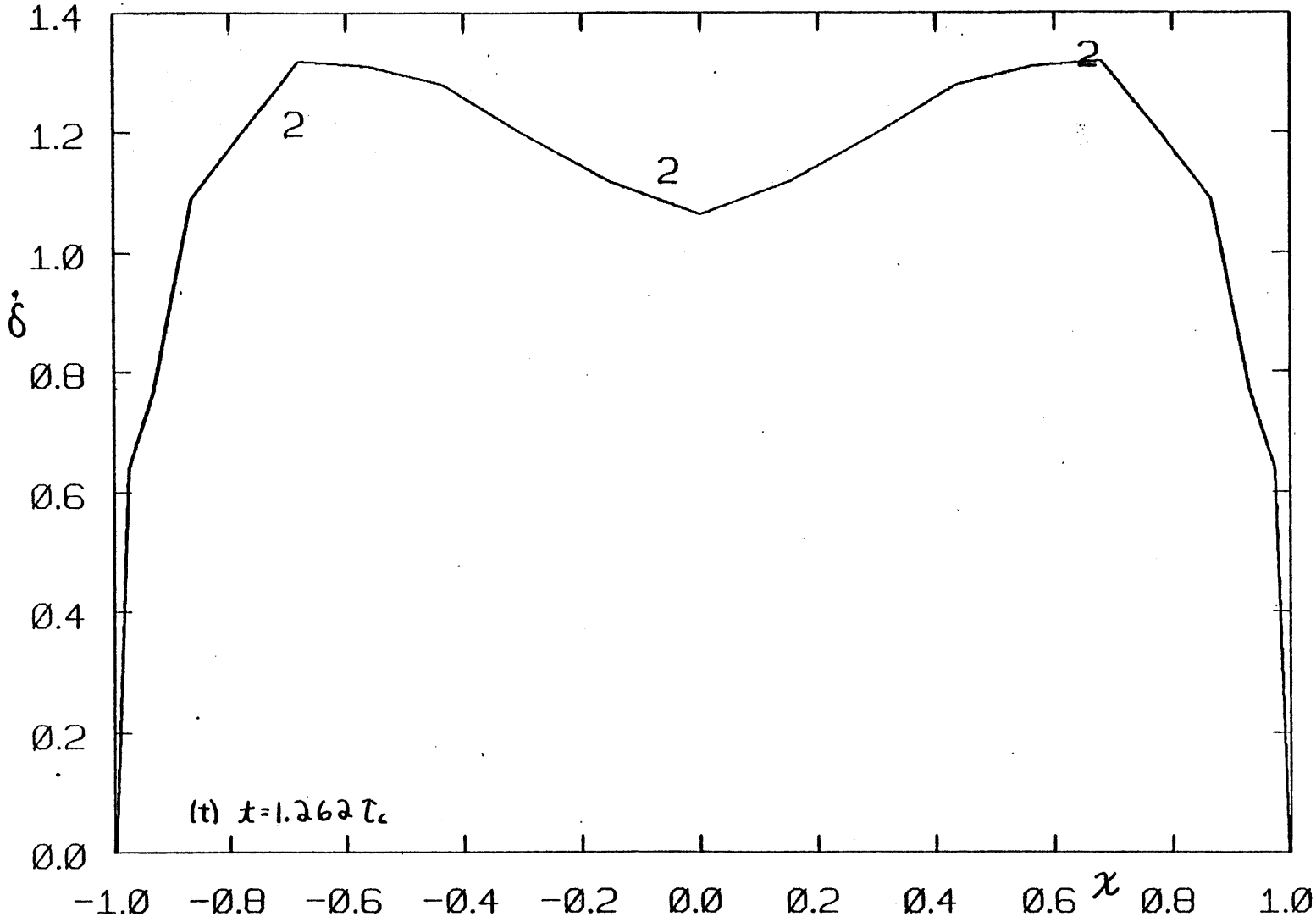

1.2
1.0
0.8
0.6
0.4
0.2
0.0
δ
2
2
2
(a) t = .9705 t_c
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
δ̇
2
2
2
(P) t=1.068 τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x


1.6
1.4
1.2
1.Ø
Ø.8
Ø.6
Ø.4
Ø.2
Ø.Ø
δ
2
2
2
(9) t=1.168 τc
-1.Ø
-Ø.8
-Ø.6
-Ø.4
-Ø.2
Ø.Ø
Ø.2
Ø.4
Ø.6
Ø.8
1.Ø
χ


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
δ̇
2
2
2
(r) t = 1.199 τc
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
χ


1.4
1.2
1.0
Ø.8
Ø.6
Ø.4
Ø.2
Ø.0
δ̇
2
2
2
(S) t*=1.230 τc
-1.Ø
-Ø.8
-Ø.6
-Ø.4
-Ø.2
Ø.Ø
Ø.2
Ø.4
Ø.6
Ø.8
1.Ø
χ


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
δ
2
2
2
(t) t=1.262 τc
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
x

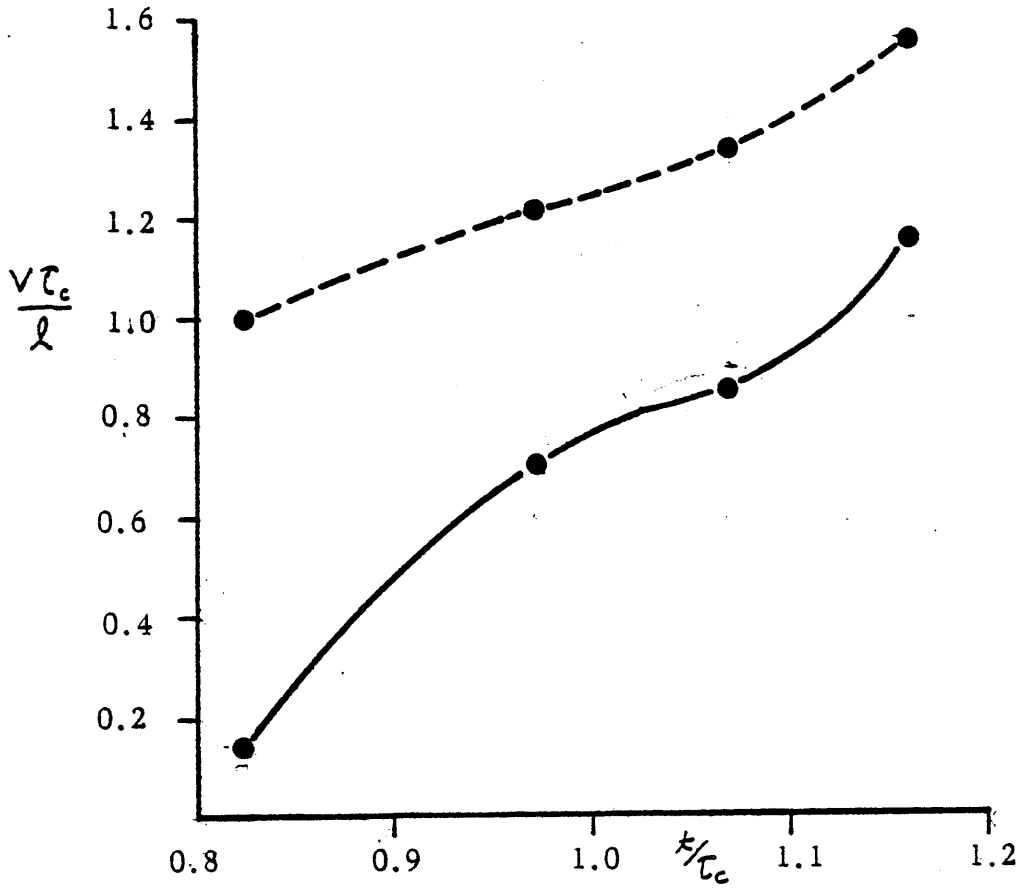


FIG. 3.21. Plot of fluid front velocity as a function of time (solid curve). The velocity of the same fluid flowing between two parallel plates with spacing $\delta(x=0, \frac{t}{\tau_c})$ and uniform pressure gradient ($P\ell/p_0 = 1.0$) is plotted as the dashed curve.

CHAPTER 4: DESCRIPTION AND STATUS OF FRACSIM: A GENERAL PURPOSE COMPUTER PROGRAM FOR HYDRAULIC FRACTURE SIMULATION

4.1 Introduction

Our ultimate goal is to have written a computer program capable of full three dimensional simulation of arbitrary hydraulic fracturing operations: these would include (but not necessarily be limited to) problems involving interaction of several arbitrarily shaped cracks, one or more of which is being propagated through a porous region containing interfaces, inclusions and/or other irregularities by internal hydraulic pressure. Thus, while developing the numerical techniques necessary for such problems (as described in the last two chapters), we have also been developing the computer program itself. In the course of our work this program has undergone several major revisions in order to incorporate increasingly versatile architecture. Currently the program, which has been dubbed FRACSIM (FRacture SIMulation), is capable of solving all of the problems discussed so far and can, in addition, solve any other plane static problems involving arbitrary numbers of arbitrarily oriented cracks (some or all of which may intersect) near an interface; however, some of the less important auxiliary subroutines for input/output, post-solution calculations, etc. are not up-to-date because of the number of different techniques tried in our work on pressure evolution). The basic structure of FRACSIM seems quite satisfactory and easily extendable for future work: it includes some important

simplifying features (such as the use of generic elements and connectivity" arrays [29]), which are apparently fairly widely used in SIE programs [7.9] and seem to be inspired by techniques used in many programs for finite element analysis.

4.2 Functional Organization of FRACSIM

FRACSIM consists of a main program (written in FORTRAN), which controls the various input/output and computational tasks that are performed by a group of subroutines*. As shown in Figure (4.1), these tasks are quite distinct and, with the exception of the various time dependent computations (viz., those required for the problems discussed in Chapter 3), are each associated with a particular subroutine. The subroutines fall naturally into five categories: control (FRACSIM), automatic data generation (AUTO, STRESS); assembly of matrices (MATRIX, STRCMP, DECOMP, CLOSRE); input/output (DUMP, RESTRT, OUTPUT, PLOT), and computation (all of the remaining subroutines). The roles of the subroutines are further clarified by arranging them in the calling hierarchy shown in Figure (4.2.). Subroutines NEWSTR, STATFL and MOVFL perform the computations

*Technically, the name FRACSIM refers only to the main program. The subroutines are stored separately in a subroutine library named FRACLIB. This type of organization permits editing and re-compiling a particular subroutine independently of the rest. Great savings in time and cost are thus realized.

necessary for the fluid pressure evolution (explicit and implicit methods) and moving fluid front problems, respectively (which computes pressure evolution by explicit integration). NEWSTR is currently not used at all, and STATFL (pressure evolution by the implicit method) has been superceded by MOVFL; (moving fluid front computation). They are retained for possible future reference and comparison. A complete listing of FRACSIM and all of the subroutines is included as Appendix B.

4.3 Program Structure

The versatility of a program such as ours is largely dependent upon the quality of what might best be called the "bookkeeping": the internal representation of the various elements, nodes and crack surfaces and the manner in which each such piece of information is used in performing the necessary computations. Also, the program should be structured in such a way that all problems can be couched in terms of its normal input requirements. For example, in terms of the input data there is no fundamental difference between the branched crack and micro-crack problems; only the location of the cracks, the type of closure or matching conditions specified, and the relative elastic moduli need to be changed to solve one problem or the other. Outside of the steps required for automatic calculation of matching condition coefficients (which we regard as purely a convenience feature) each type of problem is handled by identically the same FORTRAN statements. There is no branching to one part of FRACSIM

for branched crack problems or to another for three-crack-model blunted crack problems (except for the quasi static problems).

Toward these ends we have developed a bookkeeping system and have structured the major operations (such as evaluating matrix elements) so that they are applicable to all problems involving straight, plane cracks, and easily extendable to be completely universal when such capabilities are required.

In Chapter 1 it was noted that our special SIE method requires the division of crack loci into one or more elements, each of which is sub-divided into a number of discrete nodal points. The object of the method is to evaluate the dislocation density at some of these nodes based upon the known tractions at other nodes; (it is conceivable that, in some local interpolation schmes, these two sets of nodes would be identical). The purpose of phrasing the description of our SIE method as it was done in the first chapter (not the only possible -- nor even the simplest -- statement of it) is, in fact, that such a verbal description suggests a very powerful program structure. Thus, the entities which our program deals with are nodal points and boundary elements.

The information required by FRACSIM consists of two tables of nodal point coordinates (one set associated with the known tractions, the other with the unknown disloaction densities), two other tables which contain lists of the nodes that constitute each element, and the tractions (in local coordinates) at each of the "traction nodes". These tables correspond to the internal arrays

XNODE, TNODE, ELMNTT, ELMNTX, and STRSL, whose organization is shown in Figure 4.3. While these arrays could be typed manually, line by line, by the analyst and read directly by FRACSIM, we have obviated this tedious work by writing an automatic data generating subroutine (AUTO) which will fill in the bookkeeping arrays, based on a few input parameters, for 2-D crack problems in which we elect to use the Gauss-Chebyshev global interpolation scheme. A second task which has been automated for the convenience of the use is that of translating the appropriate closure and matching conditions into actual matrix elements. Subroutine CLOSRE is equipped to supply any combination of the closure and matching conditions discussed in Chapters 1 and 2, based upon choices that are made by the user and ready by AUTO.

Perhaps the most important point to note here is that in problems involving more than one crack there is no direct internal distinction between the various cracks; stated differently, none of the basic operations performed by FRACSIM require knowledge of the number of cracks or the particular crack upon which is located the node or element being operated on. Note also that, with the lack of internal distinction between elements and crack surfaces, there is really no internal "conceptual" difference between local and global interpolation methods. The implications of this last point are important: for instance, we should be able to use the same fundamental procedure (in subroutine MATRIX) for setting up a matrix for a 3-D problem, where we may <u>have</u> to use local inter-

polation, as for a 2-D problem where we can use the more convenient global method.

Another important feature of our program structure, although not explicitly apparent at this time because of our exclusive use of global interpolation, is that we use a single (set of) generic interpolation function(s) (i.e., functions defined on [-1,1] for the integration in Equation (1.5)). Use of this standard technique obviates a separate set of interpolation functions for each element.

Since the subroutines used for quasi-static (fluid injection) problems are still being developed, we have not yet endowed them with the same multi-crack and near-interface capabilities as we have those sections of FRACSIM which are devoted to static problems. Consequently, they lack the generalized structure as well: they are currently restricted to solving problems involving a single crack on the interval [-1,1] by global Gauss-Chebyshev interpolation.

4.4 Format of Required Input Data

The automatic data generation subroutine (AUTO) requires that values for its input parameters be arranged in the following order and format (all lines or cards must be included, except as indicated):

line or card number:

1. ISTART (I2) — Choice of whether to read input data for a new problem (ISTART=1) or to re-start a previous problem from the state at which computation had stopped (ISTART=2: currently not supported).

2. NREG (I2) — Number of material regions. For each region, one set of the following cards (RTYPE; RPOS: E,NU,G) is required.

3. RTYPE (I2) — Specific geometric type of region. Current options:

   RTYPE = 1 infinite plane

   RTYPE = 2 half plane

4. RPOS (array: 3F10.4) — Specific location of material region. For a half plane:

   RPOS(1) x-coordinate of interface

   RPOS(2) = -1. region is to the left of the interface

   RPOS(2) = +1. region is to the right of the interface

   RPOS(3) is irrelevant in this case

5. E,NU,G (3F10.4) — Elastic constants of region

6. IDOF (I2) — The range of and in Eq. (1.2). IDOF=2 unless all cracks are collinear and have purely normal loading, in which case the solution is more economical if IDOF=1.

7. NCC (I2) The number of closure/matching conditions required. For each condition, one set of the following three cards (ITYPE, ELMNT, DIR) is required.

8. ITYPE (I2) The specific type of closure or matching condition:

   ITYPE = 1 closure condition

   ITYPE = 2 branched crack matching condition

   ITYPE = 3 blunted crack matching condition (two crack model)

   ITYPE = 4 blunted crack matching condition (three crack model).

9. ELMNT (array, 6I2) The surface number of each surface associated with the particular matching condition (See SURFNO).

10. DIR (array: 6I2) The component of dislocation density to be used in the particular matching condition in reference to the crack surface specified by the corresponding element of ELMNT:

    DIR = 1 normal component

    DIR = 2 tangential component

11. NSURF (I2) The number of cracks in the problem. For each crack, one set of the following cards (SURFNO, NBPTS, LTYPE, AA-BB, LOAD) must be supplied.

12. SURFNO (I2) The number assigned to each crack (surface). The numbering scheme used is arbitrary.

13. NBPTS (I2) The number of the points on the surface.

14. LTYPE (12) The type of load distribution desired on the surface (see also LOAD). The most frequently needed choices are:

    LTYPE = 2 "square root" load distribution (see Chapter 3)

    LTYPE = 3 uniform loading with desired magnitudes of both normal and severe components.

    LTYPE = 4 triangular loading (see Chapter 3)

    LTYPE = 7 triangular loading from x_{lfront} to x_{rfront}.

15. LFRONT, RFRONT (2I3) XNODE numbers at which the left and right fluid fronts are located (meaningful only if LTYPE = 4).

16. LOAD (array: 2E15.4) Magnitudes of normal and shearing tractions, respectively. For nonuniform loading, LOAD (1) is the magnitude of the fluid pressure at the borehole and the value of LOAD (2) is irrelevant.

17. AA,BB (arrays: 2F10.4 ea) Global x and y coordinates, respectively, of the left (AA) and right (B) crack tips

18. TFIN, DT (F10.4 ea) Stop time and time increments for quasi-static problems (irrelevant for static problems)

19. VISCO (E15.4) Frac. fluid viscosity.

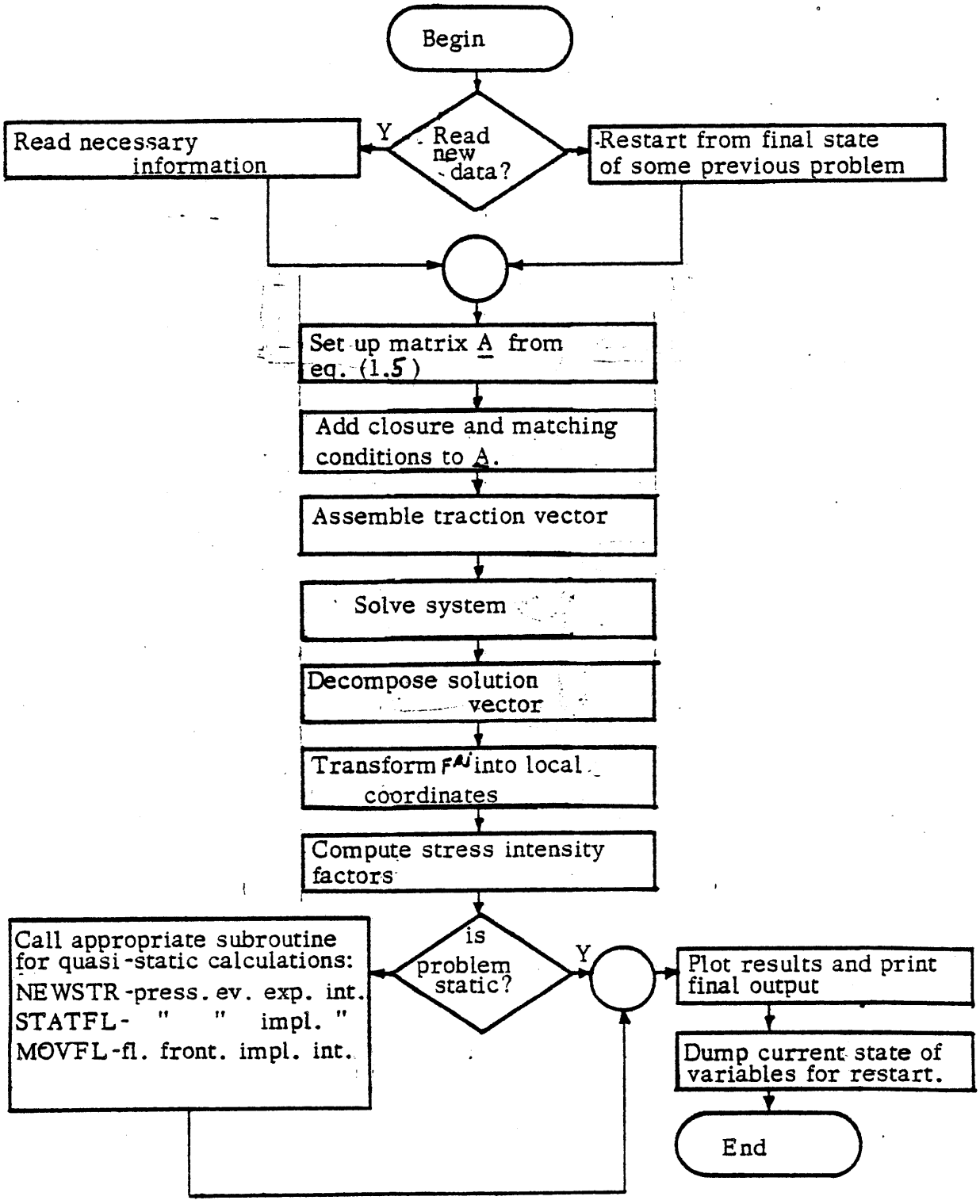


FIG. 4.1. Functional flow diagram of FRACSIM.

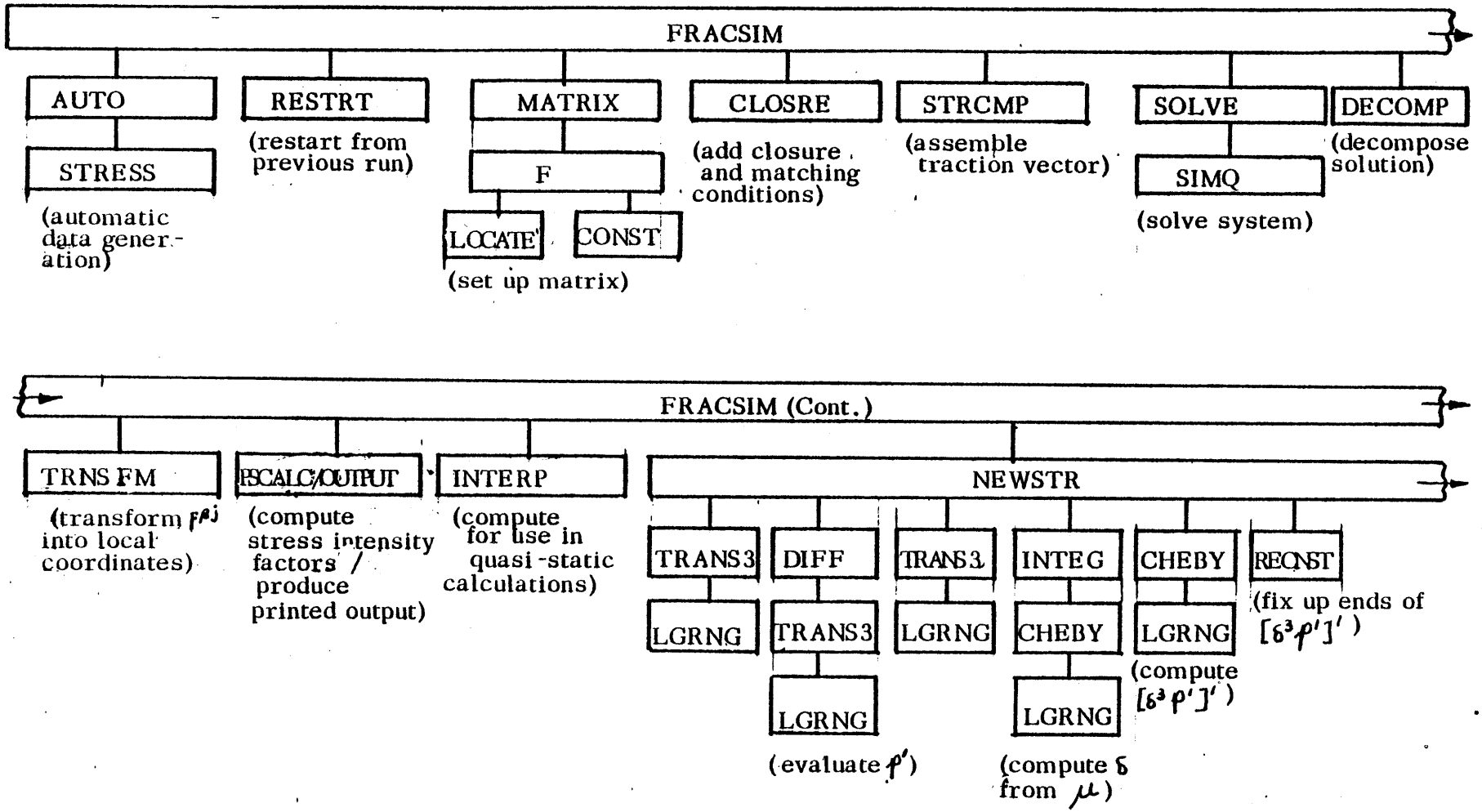


FIG. 4.2. Diagram showing both the hierarchy among the subroutines and the calling sequence followed in the course of a run. The sequence of events starts from the upper left end of the box representing the main program (FRACSIM) and ends at the lower right (next page). Note the correspondence to the flow diagram in fig. 4.1.

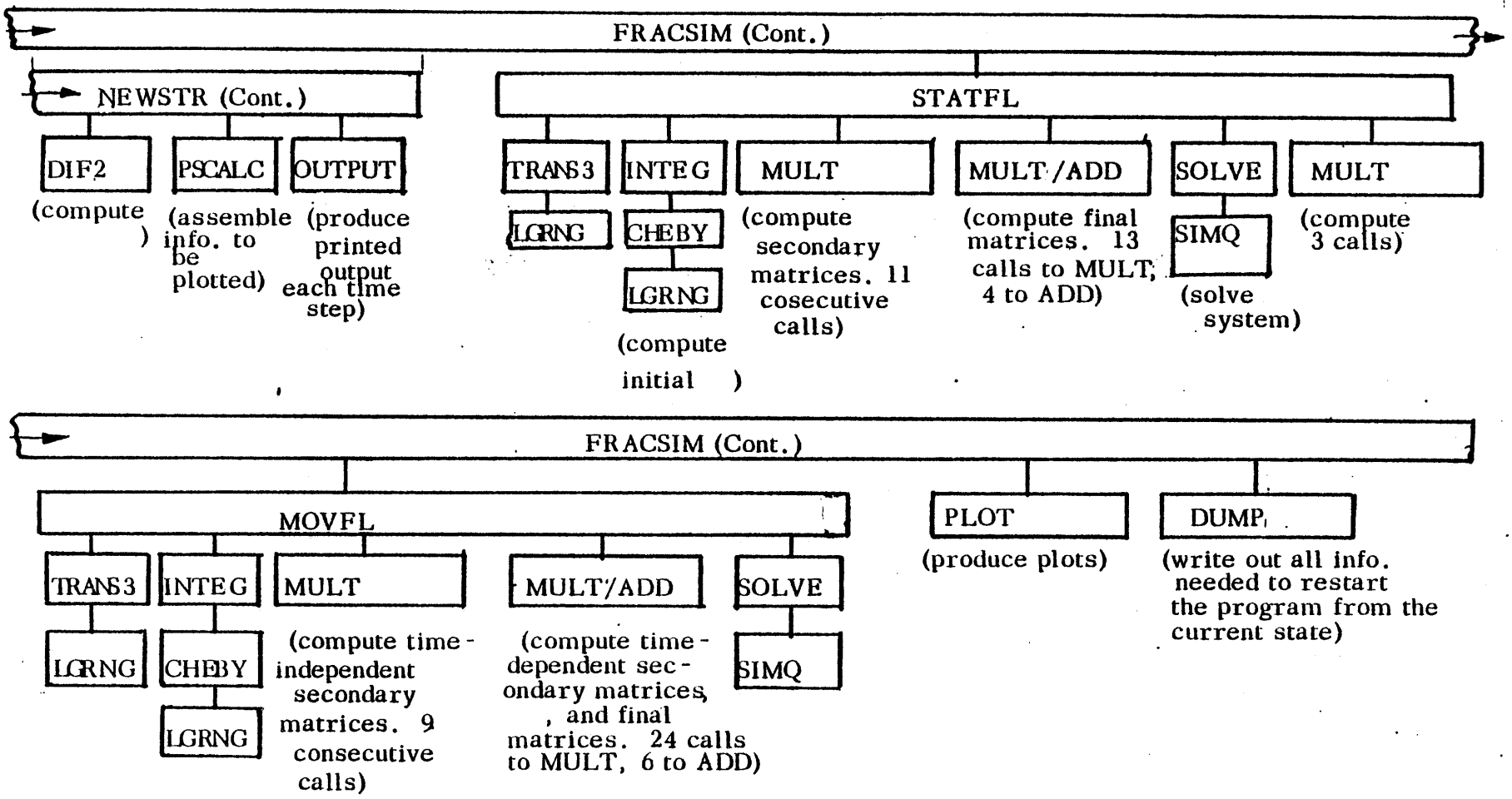


FIG. 4.2(continued).

XNODE, TNODE

node #	x_1 coord.	x_2 coord.	x_3 coord.
1			
2			
3			
.			
.			
.			

ELMNTT, ELMNTX

element #	#of TNODES or XNODES	1.	2.	3.	nodes .....
1					
2					
3					
.					
.					
.					

STRSL

node #	σ_{nn}	σ_{nt}
1		
2		
3		
.		
.		
.		

FIG. 4.3. Organization of bookkeeping arrays in FRACSIM.

CHAPTER 5: CONCLUSIONS

The problems discussed in the previous Chapters have served as a proving ground for the numerical techniques required to simulate more complicated fracture events, and at the same time have provided some valuable preliminary insights into some of the more important situations in actual hydraulic fracturing. From the modelling standpoint, we have established that it is best to employ a two crack model for a crack penetrating an interface (and for branched cracks, of course) whereas a three-crack model is needed for blunted crack simulation. Also, we have noted that there now appears to be a completely satisfactory "matching condition" (viz, setting. The dislocation density singularity to zero at the intersection) for branched crack problems, which will probably also prove to be the best extra condition for blunted cracks and cracks through interfaces. Our work with quasi-static problems (involving coupled fluid flow and crack opening) has led to the rejection of explicit time integration methods, because of insufficient coupling of the requirements of fluid mass conservation and elasticity. We must use instead the very stable implicit scheme, in which mass conservation and elasticity requirements are satisfied simultaneously.

From our studies of some relevant static models, we have seen, for instance, how adjacent strata may make it easy for a propagating hydraulic fracture to break out of the pay zone (if the adjacent stratum is relatively soft) or provide an "elasticity barrier" to enhance containment (if the adjacent stratum is relatively stiff). It

is apparent that micro-cracks in the adjacent stratum can be induced to break through the interface and link up with an hydraulic fracture, thereby circumventing this elasticity barrier. We note that there is a possibility that the blunting process may ensure containment, but it may also helpfully inhibit propagation away of the other tip from the interface. Branching, if it occurs, could also play a major part in the containment process. Our work on quasi-static problems has given us some insight on the nature of fluid flow within cracks; and in particular has shown that some early expectations (such as the characteristic time for fluid penetration [23]) are essentially correct.

We have incorporated what we feel to be a very powerful basic structure into a general purpose computer program whose capabilities will be extended as our work continues.

REFERENCES

1. Howard, G. C. and Fast, C. R. Hydraulic Fracturing, Society of Petroleum Engineers of AIME, 1970.

2. Perkins, J. K. and Kern, L. R. "Widths of Hydraulic Fractures", J. Pet. Tech., 1961, p. 937.

3. Geertsma, J. and de Klerk, F. "A Rapid Method of Predicting Width and Extent of Hydraulically-Induced Fractures", J. Pet. Tech., Dec., 1969, p. 157.

4. Daneshy, A. A. "On the Design of Vertical Hydraulic Fractures", J. Pet. Tech., 1973, pp. 83-93.

5. Abe, H., Mura, J. and Keer, L.M. "Growth Rate of a Penny-Shaped Crack in Hydraulic Fracturing of Rocks", J. Geophys. Res., 81, (29), pp. 5335-5340, 1976.

6. Mahmoud-Mohammed, N.O. "On The Development of a Numerical Method for Hydrofracturing Calculations", M.S. Thesis at Brown Univ., June, 1977.

7. Cruse, J.A. and Rizzo, F. (eds). Boundary Integral Equation Methods: Computational Methods in Applied Mechanics, published by ASME, 1975.

8. Cleary, M.P. First and Second Quarterly Reports to Lawrence Livermore Laboratories, 1978.

9. Brebbia, C.A., The Boundary Element Method for Engineers, John Wiley and Sons, New York.

10. Cleary, M.P. "Fundamental Solutions for Fluid Saturated Porous Media and Application to Localised Rupture Phenomena", PhD Thesis at Brown University, Dec., 1975.

11. Wong, S.K. S.M. Thesis in progress, M.I.T., Dept. of Mech. Eng. 1981.

12. Erdogan, F. and Gupta, G. D. "On The Numerical Solution of Singular Integral Equations", Quarterly of Applied Mathematics, January, 1972.

13. Erdogan, F. and Biricikoglu, V. "Two Bonded Half Planes with a Crack Going Through the Interface", Int. J. Engng. Sci., 1973, Vol. 11, pp 745-766.

REFERENCES (CONTINUED)

14. Cleary, M. P. "Moving Singularities in Elasto-Diffusive Solids with Application to Fracture Propagation", Int. J. Solids Structures, 1978, Vol. 14, pp. 81-97.

15. Cleary, M.P. and Petersen, D.R., Quarterly Reports to Lawrence Livermore Laboratories, 1978,1979.

16. Cleary, M.P. "Continuously Distributed Dislocation Model for Shear Bands in Softening Materials", International Journal for Numerical Methods in Engineering, Vol. 10, 679-702 (1976).

17. Cleary, M.P. "Some Primary Factors Governing the Behavior of Hydraulic Fractures in Heterogeneous Stratified Porous Formations", ASME Paper No. 78-Pet-47, 1978.

18. Gupta, G.D. "Strain Energy Release Rate for Mixed Mode Crack Problem", ASME Paper No. 76-WA/PVP-7.

19. Lo, K.K. "Analysis of Branched Cracks", Journal of Applied Mechanics, Vol. 45, 1978, pp. 797-802.

20. Papadopoulos, J.B. "A Model for Crack Blunting at a Frictional Interface". S.B. Thesis at MIT, May 1979.

21. Cook, J.S. and Erdogan, F. "Stresses in Bonded Materials with a Crack Perpendicular to the Interface", Int. J. Engng. Sci., 1972, Vol. 10, pp. 677-697.

22. Barr, D. "Thermal Cracking in Non-Porous Geothermal Reservoirs", S.M. Thesis at MIT, January 1980.

23. Cleary, M. P., "Rate and Structure Sensitivity in Hydraulic Fracturing of Fluid -- Saturated Porous Formations", to appear in the Proceedings of the 20th U.S. Symposium on Rock Mechanics, University of Texas at Austin, Juen 1979.

24. Cleary, M. P., "Comprehensive Design Formulae for Hydraulic Fracturing Operations", in preparation for publication, 1979.

25. Clenshaw, C. W. "Chebyshev Series for Mathematical Functions", Math. Tables, Vol. 5, Nat. Phys. Lab., Great Britain, 1962.

26. Cleary, M. P. and Petersen, D. R., Sixth Quarterly Report to Lawrence Livermore Laboratories, April, 1979.

REFERENCES (CONTINUED)

27. Cleary, M. P. and Petersen, D.R. Seventh Quarterly Report to Lawrence Livermore Laboratories, July, 1979.

28. Cleary, M. P. and Petersen, D.R. Second Annual Report to Lawrence Livermore Laboratories, October 1979.

29. Bathe, K. J. and Wilson, E. L., Numerical Methods in Finite Element Analysis, Prentice-Hall, Inc., 1976.

APPENDIX A: NUMERICAL INTEGRATION FORMULA FOR OBTAINING CRACK OPENING FROM DISLOCATION DENSITY

A very convenient and accurate method of integrating the dislocation density μ to obtain the crack opening δ by fitting F (see Chapter 1) with a Chebyshev series can be derived as follows:

$$\mu(x) = \frac{F(x)}{\sqrt{1-x^2}} \simeq \sum a_k T_k(x) / \sqrt{1-x^2} \tag{A.1a}$$

$$a_k = \frac{2}{N} \sum_{j=0}^{N} F(x_j) T_k(x_j) \; ; \; x_j \equiv \cos\left(\frac{\pi j}{N}\right), \; k = 0, \ldots, N \tag{A.1b}$$

We now proceed to integrate both sides of Eq. (A.1a):

$$\int_0^x \mu(\xi)\, d\xi = \delta(x) \simeq \sum_{k=1}^{N} a_k \int_0^x \frac{T_k(\xi)\, d\xi}{\sqrt{1-\xi^2}} \tag{A.2}$$

If we make the substitution

$$\xi = \cos(\theta), \quad d\xi = -\sin(\theta)\, d\theta$$

we obtain

$$\int_0^x \frac{T_k(\xi)\, d\xi}{\sqrt{1-\xi^2}} = -\int_0^{\cos^{-1}(x)} \frac{\cos(k\theta)\sin(\theta)\, d\theta}{\sqrt{1-\cos^2(\theta)}} \tag{A.3}$$

or

$$\int_0^x \frac{T_k(\xi)\,d\xi}{\sqrt{1-\xi^2}} = -\frac{1}{k}\sin\left(k\cos^{-1}(x)\right) \tag{A.4}$$

so that

$$\delta(x) = -\sum_{k=1}^{N} \frac{a_k}{k}\sin\left(k\cos^{-1}(x)\right) \tag{A.5}$$

This formula is quite convenient because it makes use of the non-singular function F and employs already existing subroutines (eq. CHEBY).

```
C
C                    F R A C S I M
C
C          A GENERAL PURPOSE PROGRAM (UNDER DEVELOPMENT)
C              FOR HYDRAULIC FRACTURE SIMULATION.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER TYPE(5),ELMNT(6),DIR(6),SURFNO
      INTEGER CLROW(6),ALPHA,BETA
      INTEGER CELMNT(6,6),CDIR(6,6),COL(6,6)
      INTEGER TSTEP,RTYPE,ORDER,R,RMAX
      REAL*8 LOAD(2),MU(80,3),KAPPA1,NU
C
      DIMENSION RLOC(5,3),RPOS(3),AA(2),B(2),NPTS(4)
      DIMENSION A1(4,2),A2(4,2),ALFA(4)
      DIMENSION A(80,80)
      DIMENSION COEFF(80),SIGMA(80)
      DIMENSION ELCON(2,3)
      DIMENSION XA1(4),XB1(4),XA2(4),XB2(4)
      DIMENSION ZETA(4,80),ETA(4,80)
      DIMENSION ICOL(6)
      DIMENSION C(6,6)
C
      COMMON /ENDPTS/ XA1,XB1,XA2,XB2,THETA
      COMMON /REG/ RLOC,TYPE,NREG
      COMMON /BKPING/ XNODE(80,3),TNODE(80,3),ELMNTT(4,80),ELMNTY(4,80)
      COMMON /CLOSE/ A1,A2,ALFA,CLROW,NCL,CELMNT,CDIR,COL,C
     1,ITYPE(10)
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /START/ IOLD,JOLD
      COMMON /ARRAYS/ A,SIGMA,COEFF
      COMMON /OUT/ STRSL(80,3),STRSC(80,3),ALOC(80,3),ACRT(80,3)
      COMMON /ELAST/ S1,KAPPA1,NU,ELCON
```

```
3500          COMMON /GP/ ZETA,NPTS
3600          COMMON /TIME/ TSTART,TFIN,DT,TSTEP,T
3700          COMMON /DOF/ IDOF
3800          COMMON /INTMETH/ ICHCE
3900    C
4000        5 CONTINUE
4100          READ(5,500) ISTART
4200      500 FORMAT(I2)
4300          IF(ISTART.EQ.1) GO TO 7
4400    C
4500          CALL RESTRT
4600    C
4700    C
4800    C
4900    C
5000    C
5100    C
5200    C
5300    C
5400    C
5500    C
5600    C
5700        7 CONTINUE
5800          TSTEP=0
5900    C
6000          CALL AUTO
6100    C
6200          T=TSTART-DT
6300    C
6400          CALL MATRIX
6500          CALL CLOSRE
6600    C
6700       10 CONTINUE
6800          IF(T.GT.TFIN) GO TO 100
6900          TSTEP=TSTEP+1
```

```
7000        20 CONTINUE
7100           CALL STRCMP
7200           CALL SOLVE(A,SIGMA,COEFF,ORDER)
7300           CALL DECOMP
7400           CALL TRNSFM
7500           CALL PSCALC
7600           CALL OUTPUT
7700           CALL INTERP(ALOC,TNODE,NTNODE,MU)
7800           T=T+DT
7900           IF(ICHCE.EQ.1) CALL NEWSTR(MU,A)
8000           IF(ICHCE.EQ.2) CALL STATFL(MU,A)
8100           IF(ICHCE.EQ.3) CALL MOVFL(MU,A)
8200      100 CONTINUE
8300    C
8400           CALL PLOT
8500           CALL DUMP
8600    C
8700           GO TO 5
8800           END
```

```
100           SUBROUTINE ADD(A,IROW,ICOL,B,JROW,JCOL,C,IER)
200           IMPLICIT REAL*8(A-H,O-Z)
300           DIMENSION A(80,80),B(80,80),C(80,80)
400     C
500           IF((IROW.NE.JROW).OR.(ICOL.NE.JCOL)) IER=1
600           IF(IER.EQ.1) GO TO 1000
700     C
800           DO 20 I=1,IROW
900           DO 10 J=1,ICOL
1000          C(I,J)=A(I,J)+B(I,J)
1100       10 CONTINUE
1200       20 CONTINUE
1300    C
1400     1000 CONTINUE
1500          RETURN
1600          END
```

```
C   SUBROUTINE AUTO
C
C                THIS SUBROUTINE AUTOMATICALLY
C           GENERATES THE NODAL POINT AND ELEMENT
C           DATA NECESSARY FOR PLANE STATIC AND
C           QUASI-STATIC CRACK PROBLEMS WHICH ARE
C           TO BE SOLVED WITH GLOBAL GAUSS-CHEBYSHEV
C           INTERPOLATION.
C
C
      SUBROUTINE AUTO
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 NONGAM,NONDEL,NONP,NONMU,LENGTH
      INTEGER TYPE(5),ELMNT(6),DIR(6),SURFNO
      INTEGER RTYPE,RFRONT,ORDER,R,RMAX,REGNO
      INTEGER CLROW(6),ELMNTT(4,80),ELMNTX(4,80),ALPHA,BETA
      INTEGER CELMNT(6,6),CDIR(6,6),COL(6,6),TSTEP
      REAL*8 LOAD(2),MU,KAPPA1,NU
C
      DIMENSION RLOC(5,3),RPOS(3),AA(2),B(2),NPTS(4)
      DIMENSION A1(4,2),A2(4,2),ALFA(4)
      DIMENSION XNODE(80,3),TNODE(80,3),STRSL(80,3),STRSC(80,3)
      DIMENSION A(80,80),C(6,6),ICOL(6)
      DIMENSION COEFF(80),SIGMA(80),ACART(80,3)
      DIMENSION ALOC(80,3),ELCON(2,3)
      DIMENSION XA1(4),XB1(4),XA2(4),XB2(4)
      DIMENSION ZETA(4,80),ETA(4,80)
      DIMENSION NDX(2),INTSZ(400),INTMID(400)
C
      COMMON /DIFPAR/ NDX,INTSZ,INTMID
      COMMON /ENDPTS/ XA1,XB1,XA2,XB2
      COMMON /REG/ RLOC,TYPE,NREG
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
```

```
      COMMON /CLOSE/ A1,A2,ALFA,CLROW,NCC,CELMNT,CDIR,COL,C
     1,ITYPE(10)
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /START/ IOLD,JOLD
      COMMON /ARRAYS/ A,SIGMA,COEFF
      COMMON /OUT/ STRSL,STRSC,ALOC,ACART
      COMMON /ELAST/ G1,KAPPA1,MU,ELCON
      COMMON /GP/ ZETA,NPTS
      COMMON /TIME/ TFIN,DT,TSTEP,T
      COMMON /FLUID/ VISCO,LOAD
      COMMON /LPAR/ LTYPE
      COMMON /DOF/ IDOF
      COMMON /NONDIM/ NONGAM,NONDEL,NONP,NONMU,TAUC
      COMMON /INTMETH/ ICHCE
      COMMON /FILL/ LFRONT,RFRONT
C
      SIN(Q)=DSIN(Q)
      COS(Q)=DCOS(Q)
      ATAN(Q)=DATAN(Q)
      SQRT(Q)=DSQRT(Q)
C
      ORDER=0
      IOLD=0
      JOLD=0
      JJK=0
      L=0
      M=0
      NREG=1
      RTYPE=1
      CLC=0
      DO 1 I=1,4
      ELMNT(I)=0
      DIR(I)=0
      ALFA(I)=0.
      ICOL(I)=0
```

```
    1 CONTINUE
      AA(1)=0.
      AA(2)=0.
      B(1)=0.
      B(2)=0.
      LOAD(1)=1.
      LOAD(2)=0.
      E=0.
      NU=.3
      G=1.
      RPOS(1)=0.
      RPOS(2)=0.
      RPOS(3)=0.
C
C           READ DATA FOR MATERIAL
C           REGIONS
C
      READ(5,502) NREG
      DO 27 IREG=1,NREG
C      READ(5,502) REGNO
      READ(5,502) RTYPE
      READ(5,504) (RPOS(II),II=1,3)
      READ(5,504) E,NU,G
      ELCON(REGNO,1)=E
      ELCON(REGNO,2)=NU
      ELCON(REGNO,3)=G
      RLOC(REGNO,1)=RPOS(1)
      RLOC(REGNO,2)=RPOS(2)
      RLOC(REGNO,3)=RPOS(3)
      TYPE(REGNO)=RTYPE
C
    5 CONTINUE
C
      READ(5,502) IDOF
C
```

```
C          READ CLOSURE/ MATCHING CONDITION
C          PARAMETERS
C
      READ(5,502) NCC
      DO 3 I=1,NCC
      READ(5,502) ITYPE(I)
      READ(5,500) (ELMNT(ICNT),ICNT=1,6)
      READ(5,500) (DIR(ICNT),ICNT=1,6)
C     READ(5,500) (ICOL(ICNT),ICNT=1,6)
C     READ(5,501) (ALFA(ICNT),ICNT=1,6)
      DO 4 J=1,6
      CELMNT(I,J)=ELMNT(J)
      CDIR(I,J)=DIR(J)
      C(I,J)=ALFA(J)
      COL(I,J)=ICOL(J)
    4 CONTINUE
    3 CONTINUE
C
C          READ SURFACE DATA
C
      READ(5,502) NSURF
      ORDER=0
      L=0
      M=0
      NELMNT=NSURF
      J=0
      DO 35 I=1,NSURF
      READ(5,502) SURFNO
      READ(5,502) NBPTS
      READ(5,502) LTYPE
      READ(5,517) LFRONT,RFRONT
      READ(5,503) LOAD(1),LOAD(2)
      READ(5,501) AA(1),AA(2),B(1),B(2)
  501 FORMAT(4F10.4)
  503 FORMAT(2E15.4)
```

```
  502 FORMAT(I2)
  504 FORMAT(3F10.4)
  517 FORMAT(2I3)
      XA1(SURFNO)=AA(1)
      XA2(SURFNO)=AA(2)
      XB1(SURFNO)=B(1)
      XB2(SURFNO)=B(2)
      NPTS(SURFNO)=NBPTS
      JMAX=NBPTS
      NONP=1./LOAD(1)
  500 FORMAT(4I2)
C
C
      JMAX=NPTS(I)
      DO 20 J=1,JMAX
      ARG=(2.*J-1.)*3.1415926535898/(2.*NPTS(I))
      ZETA(1,J)=-COS(ARG)
   20 CONTINUE
      KMAX=JMAX-1
      DO 22 J=1,KMAX
      ARG=3.14159265898*J/NPTS(I)
      ETA(I,J)=-COS(ARG)
   22 CONTINUE
C
      DO 25 J=1,JMAX
      L=L+1
      ELMNTT(I,1)=NPTS(I)
      ELMNTT(I,J+1)=L
      TNODE(L,1)=.5*XA1(I)*(1.-ZETA(I,J))+.5*XB1(I)*(1.+ZETA(I,J))
      TNODE(L,2)=.5*XA2(I)*(1.-ZETA(I,J))+.5*XB2(I)*(1.+ZETA(I,J))
      TNODE(L,3)=0.
      DO 25 DETA=1,IDOF
      ORDER=ORDER+1
   25 CONTINUE
      DO 30 J=1,KMAX
```

```
      M=M+1
      ELMNTX(I,1)=NPTS(I)-1
      ELMNTX(I,J+1)=M
      XNODE(M,1)=.5*XA1(I)*(1.-ETA(I,J))+.5*XB1(I)*(1.+ETA(I,J))
      XNODE(M,2)=.5*XA2(I)*(1.-ETA(I,J))+.5*XB2(I)*(1.+ETA(I,J))
      XNODE(M,3)=0.
   30 CONTINUE
      DO 190 JJ=2,JMAX
      JJK=JJK+1
      CALL STRESS(STRSL,LOAD,I,JJK)
  190 CONTINUE
   35 CONTINUE
C
      READ(5,510) TFIN,DT
C
  562 FORMAT(F10.4)
  510 FORMAT(2F10.4)
      NXNODE=M
      NTNODE=L
      NCC=0
      KMAX=L
      II=0
C
C
      READ(5,288) VISCO
  288 FORMAT(E15.4)
   52 CONTINUE
      DO 55 ISURF=1,NSURF
      DO 50 ALPHA=1,IDOF
      II=II+NPTS(ISURF)
      NCLC=NCLC+1
      CLROW(NCLC)=II
   50 CONTINUE
   55 CONTINUE
C
```

```
C
C             READ CHOICE OF TIME INTEGRATION
C             METHOD:
C
C                   ICHCE = 1  EXPLICIT INTEGRATION (NEWSTR)
C                         = 2  IMPLICIT INTEGRATION (STATFL)
C                         = 3  IMPLICIT INTEGRATION (MOVFL)
C
C
      READ(5,924) ICHCE
C
C           READ PARAMETERS FOR SUBROUTINE DIFF
C           IF SUBROUTINE NEWSTR IS TO BE USED
C           FOR PRESSURE EVOLUTION COMPUTATIONS
C
      READ(5,924) NDR
  924 FORMAT(I3)
      DO 943 II=1,NDR
      READ(5,925) INTSZ(II),INTMID(II)
  943 CONTINUE
  925 FORMAT(2I2)
      NDX(1)=NDR
      NDX(2)=NDR
C
      LENGTH=1.
      NONSAM=LENGTH/G
      NONDEL=G/(LOAD(1)*LENGTH)
      NONP=1./LOAD(1)
      NONMU=G/LOAD(1)
      TAUC=(12.*VISCO/G)*(G/LOAD(1))**3
C
      RETURN
      END
```

```
C  SUBROUTINE CHEBY
C
C                  THIS SUBROUTINE FITS THE FUNCTION WHOSE
C               VALUES AT THE POINTS ARG1 ARE TRANSMITTED IN
C               TAB0 WITH A CHEBYSHEV SERIES.  CHEBY COMPUTES
C               THE VALUE OF THE SERIES, ITS TERMWISE DERIVATIVES
C               AND THE INTEGRAL OF THE FUNCTION*SQRT(1-X**2)
C               AT THE POINTS ARG2 AND RETURNS
C               THEM IN F.
C
C
C
      SUBROUTINE CHEBY(ARG1,ARG2,NDIM1,NDIM2,TAB0,NDEG,NXC,NDERIV,F,
     1IROW,JROW,IEVAL)
C
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER R
      DIMENSION ARG1(IROW,1),ARG2(JROW,1),TAB(400,3),F(JROW,5)
      DIMENSION T(400,4),TAB0(IROW,1)
      DIMENSION Y(400),A(400)
C
      COS(Q)=DCOS(Q)
      ACOS(Q)=DACOS(Q)
C
      IF((IEVAL.NE.0).AND.(IEVAL.NE.1)) GO TO 5000
C
      DO 5 I=1,NXC
      XC=-COS(3.1415926535898*(I-1)/(NXC-1))
      CALL LGRNG(ARG1,TAB0,XC,PT,NDIM1,5,IROW)
      TAB(I,1)=PT
    5 CONTINUE
C
      DO 20 R=1,NDEG
       SUM=0.
      DO 10 J=1,NXC
```

```
3500            FUDGE=1.
3600            IF((J.EQ.1).OR.(J.EQ.NDEG)) FUDGE=.5
3700            XXA=-COS((R-1)*3.1415926535898*(J-1)/(NXC-1))
3800            SUM=SUM+TAB(J,1)*XXA*FUDGE
3900         10 CONTINUE
4000            A(R)=2.*SUM/(NXC-1)
4100         20 CONTINUE
4200            A(1)=.5*A(1)
4300      C
4400       5000 CONTINUE
4500      C
4600            T(1,1)=1.
4700            T(1,2)=0.
4800            T(1,3)=0.
4900      CX      T(1,4)=0.
5000      C
5100            T(2,2)=1.
5200            T(2,3)=0.
5300      CX      T(2,4)=0.
5400      C
5500            DO 40 I=1,NDIM2
5600            DO 35 J=1,5
5700            F(I,J)=0.
5800         35 CONTINUE
5900      C
6000            T(2,1)=ARG2(I,1)
6100      C
6200            DO 32 N=3,NDEG
6300            IF(IEVAL.EQ.1) GO TO 33
6400      C
6500            T(N,1)=2.*ARG2(I,1)*T(N-1,1)-T(N-2,1)
6600            T(N,2)=(2.*T(N-1,1)+2.*ARG2(I,1)*T(N-1,2)-T(N-2,2))
6700      CX     T(N,3)=(4.*T(N-1,2)+2.*ARG2(I,1)*T(N-1,3)-T(N-2,3))
6800      CX      T(N,4)=(6.*T(N-1,3)+2.*ARG2(I,1)*T(N-1,4)-T(N-2,4))
6900      C
```

```
7000            F(I,1)=F(I,1)+A(N)*T(N,1)
7100            F(I,2)=F(I,2)+A(N)*T(N,2)
7200    CX        F(I,3)=F(I,3)+A(N)*T(N,3)
7300    CX        F(I,4)=F(I,4)+A(N)*T(N,4)
7400    C
7500            IF(IEVAL.EQ.0) GO TO 32
7600         33 CONTINUE
7700            KKL=N-1
7800    CX        F(I,5)=F(I,5)-A(N)*DSQRT(1.-TCH(KKL,ARG2(I,1))**2)/DFLOAT(N-1)
7900            ARG=DFLOAT(KKL)*DACOS(ARG2(I,1))
8000            TCH=DSIN(ARG)/DFLOAT(KKL)
8100            F(I,5)=F(I,5)-A(N)*TCH
8200    C
8300         32 CONTINUE
8400            F(I,1)=F(I,1)+A(1)*T(1,1)+A(2)*T(2,1)
8500            F(I,2)=F(I,2)+A(1)*T(1,2)+A(2)*T(2,2)
8600    C
8700            ARG=DACOS(ARG2(I,1))
8800            TCH=DSIN(ARG)
8900            F(I,5)=F(I,5)-0.5*A(1)*ARG2(I,1)-
9000           1A(2)*TCH
9100    C
9200         40 CONTINUE
9300            WRITE(6,536) (A(II),II=1,NDEG)
9400        536 FORMAT(' ',8E15.4)
9500    C
9600            RETURN
9700            END
```

```
C  SUBROUTINE CONST
C
C
C                  THIS SUBROUTINE COMPUTES THE
C            COMBINATIONS OF ELASTICITY CONSTANTS
C            REQUIRED FOR EVALUATION OF THE CHOSEN
C            INFLUENCE FUNCTION.   CONST IS CALLED
C            ONLY WHEN A NEW SET OF PARAMETERS IS
C            REQUIRED, AS DETERMINED BY SUBROUTINE
C            LOCATE.
C
C
      SUBROUTINE CONST(IREG,JREG,EMOD,A,B)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 KAPPA1,KAPPA2,MU
      DIMENSION ELCON(2,3)
      COMMON /ELAST/ G,KAPPA1,MU,ELCON
C
C        PARAMETERS FOR THE INFLUENCE FUNCTION FOR
C        A DISLOCATION NEAR AN INTERFACE.  FOR
C        CONVENIENCE, WE CURRENTLY ASSUME THAT THE
C        REGIONS WILL BE NUMBERED 1 AND 2.
C
      G=ELCON(JREG,3)
      KAPPA1=3.-4.*ELCON(JREG,2)
      PI=3.1415926535898
      EMOD=G/(PI*(KAPPA1+1.))
      IF(JREG.EQ.2) GAMMA=ELCON(1,3)/ELCON(2,3)
      IF(JREG.EQ.1) GAMMA=ELCON(2,3)/ELCON(1,3)
      IF(JREG.EQ.1) KAPPA2=3.-4.*ELCON(2,2)
      IF(JREG.EQ.2) KAPPA2=3.-4.*ELCON(1,2)
      A=(1.-GAMMA)/(1.+GAMMA*KAPPA1)
      B=(KAPPA2-GAMMA*KAPPA1)/(KAPPA2+GAMMA)
C
      RETURN
      END
```

```
C  SUBROUTINE CLOSRE
C
C                THIS SUBROUTINE COMPUTES AND INSERTS
C           THE MATRIX ELEMENTS CORRESPONDING TO VARIOUS
C           CLOSURE AND MATCHING CONDITIONS.
C
      SUBROUTINE CLOSRE
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 KAPPA1,MU
      INTEGER ORDER
      INTEGER ELMNTT(2,80),ELMNTX(2,80),BETA,CLROW(6),CELMNT(6,6)
      INTEGER CDIR(6,6)
      INTEGER ELMNT(6)
      INTEGER COL(6,6)
C
      DIMENSION XA1(4),XA2(4),XB1(4),XB2(4),A1(4,4),A2(4,4),ALPHA(4)
      DIMENSION XNODE(80,3),TNODE(80,3),STRSL(80,3),STRSC(80,3)
      DIMENSION A(80,80)
      DIMENSION COEFF(80),SIGMA(80),ACART(80,3)
      DIMENSION ALOC(80,3),ELCON(2,3)
      DIMENSION C(6,6)
      DIMENSION ZETA(2,80),NPTS(4)
C
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /CLOSE/ A1,A2,ALPHA,CLROW,NCC,CELMNT,CDIR,COL,C
     1,ITYPE(10)
      COMMON /ENDPTS/ XA1,XB1,XA2,XB2,THETA
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
      COMMON /ARRAYS/ A,SIGMA,COEFF
      COMMON /OUT/ STRSL,STRSC,ALOC,ACART
      COMMON /ELAST/ G1,KAPPA1,MU,ELCON
      COMMON /GP/ ZETA,NPTS
      COMMON /DOF/ IDOF
C
```

```
      SIN(Q)=DSIN(Q)
      COS(Q)=DCOS(Q)
      ATAN(Q)=DATAN(Q)
      SQRT(Q)=DSQRT(Q)
C
      PI=3.1415926535898
      DO 1000 ICC=1,NCC
      ICLC=ITYPE(ICC)
      GO TO (100,500,700,800),ICLC
C
C
C
  100 CONTINUE
C
C     NET ENTRAPPED DISLOCATION=0:
C
      DO 10 J=1,ORDER
      A(CLROW(ICC),J)=0.
   10 CONTINUE
      SIGMA(CLROW(ICC))=0.
      DO 20 L=1,4
      JEL=CELMNT(ICC,L)
      IF(JEL.EQ.0) GO TO 25
      JJ=0
      DO 19 J=1,NELMNT
      DO 18 BETA=1,IDOF
      KMAX=ELMNTT(J,1)+1
      DO 16 K=2,KMAX
      JJ=JJ+1
      IF((J.EQ.JEL).AND.(BETA.EQ.CDIR(ICC,L))) A(CLROW(ICC),JJ)=SQRT((XB
     11(JEL)-XA1(JEL))**2+(XB2(JEL)-XA2(JEL))**2)*PI/(2.*ELMNTT(JEL,1))
C
   16 CONTINUE
   18 CONTINUE
   19 CONTINUE
```

```
 7000       20 CONTINUE
 7100       25 CONTINUE
 7200          GO TO 1000
 7300    C
 7400      500 CONTINUE
 7500          TEST=5000.
 7600          TEST1=5000.
 7700    C
 7800          DO 510 J=1,ORDER
 7900          A(CLROW(ICC),J)=0.
 8000      510 CONTINUE
 8100          SIGMA(CLROW(ICC))=0.
 8200    C
 8300    C        MATCHING CONDITIONS FOR BRANCHED CRACKS:
 8400    C
 8500    C
 8600          JJ1=NPTS(1)
 8700          JJ2=1
 8800          PHI=PI/2.
 8900          THETA=PI/2.
 9000          ARG=(XB2(2)-XA2(2))/(XB1(2)-XA1(2))
 9100          ARG1=(XB2(1)-XA2(1))/(XB1(1)-XA1(1))
 9200          IF(XA1(2).NE.XB1(2)) PHI=ATAN(ARG)
 9300          THETA=ATAN(ARG1)
 9400    C
 9500          JJ=0
 9600          DO 526 L=1,2
 9700          DO 525 BETA=1,IDOF
 9800          IMAX=NPTS(L)
 9900          DO 524  I=1,IMAX
10000          JJ=JJ+1
10100          IF((BETA.EQ.1).AND.(L.EQ.1).AND.(I.EQ.JJ1)) J1=JJ
10200          IF((BETA.EQ.2).AND.(L.EQ.1).AND.(I.EQ.JJ1)) J2=JJ
10300          IF((BETA.EQ.1).AND.(L.EQ.2).AND.(I.EQ.JJ2)) J3=JJ
10400          IF((BETA.EQ.2).AND.(L.EQ.2).AND.(I.EQ.JJ2)) J4=JJ
```

```
10500      524 CONTINUE
10600      525 CONTINUE
10700      526 CONTINUE
10800          COL(3,1)=J1
10900          COL(3,2)=J2
11000          COL(3,3)=J3
11100          COL(3,4)=J4
11200          COL(4,1)=J1
11300          COL(4,2)=J2
11400          COL(4,3)=J3
11500          COL(4,4)=J4
11600          C(3,1)=SIN(THETA)
11700          C(3,2)=COS(THETA)
11800          C(3,3)=SIN(PHI)
11900          C(3,4)=COS(PHI)
12000          C(4,1)=COS(THETA)
12100          C(4,2)=-SIN(THETA)
12200          C(4,3)=COS(PHI)
12300          C(4,4)=-SIN(PHI)
12400      850 CONTINUE
12500          ALNTH1=SQRT((XB2(1)-XA2(1))**2+(XB1(1)-XA1(1))**2)
12600          ALNTH2=SQRT((XB2(2)-XA2(2))**2+(XB1(2)-XA1(2))**2)
12700    C
12800    C
12900          SIGMA(CLROW(ICC))=0.
13000    C
13100          DO 620 IK=1,ORDER
13200          A(CLROW(ICC),IK)=0.
13300      620 CONTINUE
13400          DO 630 J=1,4
13500          A(CLROW(ICC),COL(ICC,J))=C(ICC,J)
13600      630 CONTINUE
13700          GO TO 1000
13800    C
13900    C
```

```
14000    C          MATCHING CONDITIONS FOR BLUNTED CRACKS
14100    C          ( TWO CRACK MODEL):
14200    C
14300       700 CONTINUE
14400           NMAX=NPTS(2)
14500           DO 720 N=2,NMAX
14600           J=ELMNTT(2,N)
14700           K=ELMNTT(2,N+1)
14800           IF((TNODE(J,2).GT.0.).OR.(TNODE(K,2).LT.0.)) GO TO 719
14900           JJ1=N-1
15000           JJ2=N
15100       719 CONTINUE
15200       720 CONTINUE
15300           FACTOR=SQRT(1.-ZETA(2,JJ1)**2)
15400           C(3,3)=1./FACTOR
15500           C(4,3)=1./FACTOR
15600           FACTOR=SQRT(1.-ZETA(2,JJ2)**2)
15700           C(3,4)=1./FACTOR
15800           C(4,4)=1./FACTOR
15900           FACTOR=SQRT(1.-ZETA(1,NPTS(1))**2)
16000           C(3,1)=1./FACTOR
16100           C(3,2)=0.
16200           C(4,2)=1./FACTOR
16300           C(4,1)=0.
16400           JJ=0
16500           DO 726 L=1,2
16600           DO 727 BETA=1,IDOF
16700           IMAX=NPTS(L)
16800           DO 724 I=1,IMAX
16900           JJ=JJ+1
17000           IF((BETA.EQ.1).AND.(L.EQ.2).AND.(I.EQ.JJ1)) J1=JJ
17100           IF((BETA.EQ.1).AND.(L.EQ.2).AND.(I.EQ.JJ2)) J2=JJ
17200           IF((BETA.EQ.2).AND.(L.EQ.2).AND.(I.EQ.JJ1)) J3=JJ
17300           IF((BETA.EQ.2).AND.(L.EQ.2).AND.(I.EQ.JJ2)) J4=JJ
17400       724 CONTINUE
```

```
17500      727 CONTINUE
17600      726 CONTINUE
17700          COL(3,3)=J1
17800          COL(3,4)=J2
17900          COL(4,3)=J3
18000          COL(4,4)=J4
18100          ALNTH1=SQRT((XB2(1)-XA2(1))**2+(XB1(1)-XA1(1))**2)
18200          ALNTH2=SQRT((XB2(2)-XA2(2))**2+(XB1(2)-XA1(2))**2)
18300    C
18400          DO 730 J=1,4
18500          A(CLROW(ICC),COL(ICC,J))=C(ICC,J)
18600      730 CONTINUE
18700    C
18800          GO TO 1000
18900    C
19000      900 CONTINUE
19100    C
19200          SIGMA(CLROW(ICC))=0.
19300    C
19400          DO 920 IK=1,ORDER
19500          A(CLROW(ICC),IK)=0.
19600      920 CONTINUE
19700          DO 930 J=1,4
19800          A(CLROW(ICC),COL(ICC,J))=C(ICC,J)
19900      930 CONTINUE
20000      800 CONTINUE
20100    C
20200    C             MATCHING CONDITIONS FOR BLUNTED CRACKS
20300    C             (THREE CRACK MODEL):
20400    C
20500          THETA=3.1415926535898
20600          IF(XA1(2).NE.XB1(2)) THETA=DATAN((XA2(2)-XB2(2))/
20700         1(XA1(2)-XB1(2)))
20800          PHI=THETA
20900          NMAX=NPTS(2)
```

```
21000           DO 977 I=1,4
21100           DO 977 J=1,6
21200           C(I,J)=0.
21300       977 CONTINUE
21400           DO 520 N=2,NMAX
21500           J=ELMNTT(2,N)
21600           K=ELMNTT(2,N+1)
21700           IF((TNODE(J,2).GT.0.).OR.(TNODE(K,2).LT.0.)) GO TO 519
21800           JJ1=N-1
21900           JJ2=N
22000       519 CONTINUE
22100       520 CONTINUE
22200     C
22300           DO 687 I=1,6
22400           DO 687 J=1,6
22500           C(I,J)=0.
22600       687 CONTINUE
22700     C
22800           J1=NPTS(1)
22900           J2=J1+NPTS(1)
23000           J3=J2+NPTS(2)
23100           J4=J3+NPTS(2)
23200           J5=J4+1
23300           J6=J5+NPTS(3)
23400     C
23500           DO 683 II=3,6
23600           COL(II,1)=J1
23700           COL(II,2)=J2
23800           COL(II,3)=J3
23900           COL(II,4)=J4
24000           COL(II,5)=J5
24100           COL(II,6)=J6
24200       683 CONTINUE
24300     C
24400           FACTOR=SQRT(1.-ZETA(1,NPTS(1))**2)
```

```
24500           C(3,1)=1./FACTOR
24600           C(4,2)=1.
24700     C
24800           FACTOR=SQRT(1.-ZETA(2,NPTS(2))**2)
24900           C(5,3)=SIN(THETA)
25000           C(5,4)=COS(THETA)
25100           C(6,3)=COS(THETA)
25200           C(6,4)=-SIN(THETA)
25300     C
25400           FACTOR=SQRT(1.-ZETA(3,1)**2)
25500           C(5,5)=-SIN(THETA)
25600           C(5,6)=-COS(THETA)
25700           C(6,5)=-COS(THETA)
25800           C(6,6)=SIN(THETA)
25900     C
26000           DO 610 J=1,ORDER
26100           A(CLROW(ICC),J)=0.
26200       610 CONTINUE
26300           SIGMA(CLROW(ICC))=0.
26400     C
26500           DO 623 J=1,6
26600           A(CLROW(ICC),COL(ICC,J))=C(ICC,J)
26700       623 CONTINUE
26800     C
26900     C
27000      1000 CONTINUE
27100           RETURN
27200           END
```

```
C  SUBROUTINE DECOMP
C                THIS SUBROUTINE DECOMPOSES THE SOLUTION
C              VECTOR, COEFF, INTO A TABLE OF VALUES (ACART) OF
C              "F" IN THE GLOBAL Y (COLUMN 1) AND X (COLUMN 2)
C              DIRECTIONS.
C
      SUBROUTINE DECOMP
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER CLROW(6),ELMNTT(4,80),ELMNTX(4,80),ALPHA,BETA
      INTEGER ORDER
      DIMENSION XNODE(80,3),TNODE(80,3),STRSL(80,3),STRSC(80,3)
      DIMENSION A(80,80)
      DIMENSION COEFF(80),SIGMA(80),ACART(80,3)
      DIMENSION ALOC(80,3),ELCON(2,3)
C
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /ARRAYS/ A,SIGMA,COEFF
      COMMON /OUT/ STRSL,STRSC,ALOC,ACART
      COMMON /DOF/ IDOF
      IMAX=0
      IMIN=1
      II=1
      DO 400 K=1,NELMNT
      IMAX=IMAX+ELMNTT(K,1)
      DO 395 BETA=1,IDOF
      DO 390 I=IMIN,IMAX
      ACART(I,BETA)=COEFF(II)
      II=II+1
  390 CONTINUE
  395 CONTINUE
      IMIN=IMAX+1
  400 CONTINUE
      RETURN
      END
```

```
C  SUBROUTINE DIF2
C
C                  THIS SUBROUTINE COMPUTES THE DERIVATIVE
C              OF A FUNCTION BY AVERAGING OF FINITE DIFFERENCES.
C              THE DIFFERENTIATION IS DONE SEPARATELY ON EITHER
C              SIDE OF THE ORIGIN IN ORDER TO PRESERVE SLOPE
C              DISCONTINUITIES.
C
      SUBROUTINE DIF2(XC,F,NXC,TNODE,NTNODE)
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*4 XY(2,400),XSCL(4)
      DIMENSION XC(400,3),F(400,5),TNODE(80,3)
C
C
      ITK=1
      DO 10 IXC=1,NXC
C
      IF(((XC(IXC,1).GT.TNODE(ITK,1)).OR.
     1(TNODE(ITK,1).GT.XC(IXC+1,1))).OR.(ITK.GT.NTNODE))
     2GO TO 10
C
      WRITE(26,400) ITK
  400 FORMAT(' ','ITK=',I4)
C
      F(ITK,3)=((XC(IXC+2,1)-XC(IXC,1))*(F(IXC+1,2)-F(IXC-1,2))
     1+(XC(IXC+1,1)-XC(IXC-1,1))*(F(IXC+2,2)-F(IXC,2)))/
     2 (2.0*(XC(IXC+1,1)-XC(IXC-1,1))*(XC(IXC+2,1)-XC(IXC,1)))
C
      IF(ITK.GE.NTNODE) GO TO 20
      ITK=ITK+1
C
   10 CONTINUE
   20 CONTINUE
```

```
C
      DO 30 I=1,NTNODE
      XY(1,I)=TNODE(I,1)
      XY(2,I)=F(I,3)
   30 CONTINUE
C
      ISCL=-2
      XSCL(1)=-1.0
      XSCL(2)=1.0
      XSCL(3)=-20.
      XSCL(4)=20.
C
      CALL QPICTR(XY,2,NTNODE,QX(1))
C
      WRITE(13,356) ((XC(II,JJ),JJ=1,3),(F(II,JK),JK=1,5),
     1II=1,200)
  356 FORMAT(' ',8E15.4)
C
      RETURN
      END
```

```
 100    C  SUBROUTINE DIFF
 200    C
 300    C                  THIS SUBROUTINE DIFFERENTIATES A FUNCTION
 400    C              BY FINITE DIFFERENCES AT A LARGE NUMBER OF POINTS,
 500    C              THEN AVERAGING THE DIFFERENCES OVER INTERVALS OF
 600    C              SELECTED SIZES.  THE DIFFERENTIATION IS CARRIED
 700    C              OUT SEPARATELY ON EITHER SIDE OF THE ORIGIN IN
 800    C              ORDER TO PRESERVE SLOPE DISCONTINUITIES.
 900    C
1000          SUBROUTINE DIFF(X,Y,NX,NDERIV)
1100          IMPLICIT REAL*8(A-H,O-Z)
1200    C
1300          DIMENSION X(400,3),Y(400,5),YTEMP(400),NDX(2)
1400          DIMENSION IMIN(2,2),IMAX(2,2),XNEW(400,3),YNEW(400,5)
1500          DIMENSION INTSZ(400),INTMID(400)
1600          DIMENSION NDR(2)
1700    C
1800          COMMON /DIFPAR/ NDR,INTSZ,INTMID
1900    C
2000          WRITE(10,500)
2100      500 FORMAT(///)
2200          NDX(1)=NDR(1)/2
2300          NDX(2)=NDR(2)/2
2400          IMIN(1,1)=1
2500          IMAX(1,1)=160
2600          IMIN(2,1)=161
2700          IMAX(2,1)=320
2800          IMIN(1,2)=1
2900          IMAX(1,2)=160
300     -     IMIN(2,2)=161
3100          IMAX(2,2)=320
3200    C
3300    C
3400          DO 100 JDERIV=1,NDERIV
```

```
3500          IDERIV=JDERIV+1
3600          INT0=1
3700          ISTART=1
3800    C
3900          DO 95 ISIDE=1,2
4000          INT1=INT0+NDX(JDERIV)-1
4100    C
4200    C                COMPUTE DERIVATIVE
4300    C
4400          IF(ISIDE.NE.1) GO TO 30
4500          H=X(2,1)-X(1,1)
4600          ILO=1
4700          IHI=IMAX(1,JDERIV)-1
4800          DO 20 I=ILO,IHI
4900          YTEMP(I)=(Y(I+1,JDERIV)-Y(I,JDERIV))/H
5000       20 CONTINUE
5100          GO TO 50
5200    C
5300       30 CONTINUE
5400          IF(ISIDE.NE.2) GO TO 40
5500          ILO=IMIN(2,JDERIV)+1
5600          IHI=IMAX(2,JDERIV)
5700          DO 40 I=ILO,IHI
5800          YTEMP(I)=(Y(I,JDERIV)-Y(I-1,JDERIV))/H
5900       40 CONTINUE
6000    C
6100       50 CONTINUE
6200          WRITE(10,510) JDERIV,ISIDE,ILO,IHI
6300      510 FORMAT(' ','JDERIV=',I2,'ISIDE=',I2,'ILO=',I3,'IHI=',I3)
6400    C
6500    C                    COMPUTE AVERAGES
6600    C
6700          DO 70 INT=INT0,INT1
6800          SUM=0.
6900          ISTOP=ISTART+INTSZ(INT)-1
```

```
7000            IMID=ISTART+INTMID(INT)-1
7100            IF((INT.NE.INT1).OR.(ISIDE.NE.1)) GO TO 55
7200            ISTOP=ISTART+INTSZ(INT)-2
7300            IMID=ISTART+INTMID(INT)-2
7400         55 CONTINUE
7500            IF((INT.NE.INT0).OR.(ISIDE.NE.2)) GO TO 60
7600            ISTART=IMIN(2,JDERIV)+1
7700            ISTOP=ISTART+INTSZ(INT)-2
7800            IMID=ISTART+INTMID(INT)-2
7900      C
8000         60 CONTINUE
8100            DO 65 I=ISTART,ISTOP
8200            SUM=SUM+YTEMP(I)
8300         65 CONTINUE
8400            Y(INT,IDERIV)=SUM/DFLOAT(ISTOP-ISTART+1)
8500            X(INT,IDERIV)=X(IMID,1)
8600      C
8700            WRITE(10,520) JDERIV,ISIDE,INT0,INT1,ISTART,ISTOP
8800        520 FORMAT(' ','JDERIV=',I2,'ISIDE=',I2,'INT0=',I3,'INT1=',I3,
8900           1'ISTART=',I3,'ISTOP=',I3)
9000            ISTART=ISTOP+1
9100         70 CONTINUE
9200      C
9300            INT0=NDX(JDERIV)+1
9400         95 CONTINUE
9500      C
9600            DO 97 II=1,320
9700            XNEW(II,1)=X(II,1)
9800         97 CONTINUE
9900      C
10000           CALL TRANS3(X,Y,NDR(JDERIV),XNEW,YNEW,320,400,400,IDERIV)
10100           DO 98 II=1,320
10200           Y(II,IDERIV)=YNEW(II,1)
10300        98 CONTINUE
10400     C
```

```
10500      100 CONTINUE
10600    C
10700          RETURN
10800          END
```

```
C SUBROUTINE DUMP
C
C                THIS SUBROUTINE WRITES OUT ALL OF THE
C              INFORMATION NEEDED TO RESTART THE RUN
C              FROM THE CURRENT STATE.
C
      SUBROUTINE DUMP
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER ORDER,TSTEP
      INTEGER ELMNTT(4,80),ELMNTX(4,80)
C
      DIMENSION STRSL(80,3),STRSC(80,3),ALOC(80,3)
      DIMENSION ACART(80,3)
      DIMENSION XNODE(80,3),TNODE(80,3)
      DIMENSION XA1(4),XB1(4),XA2(4),XB2(4)
C
      COMMON /TIME/ TSTART,TFIN,DT,TSTEP,T
      COMMON /OUT/ STRSL,STRSC,ALOC,ACART
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
      COMMON /ENDPTS/ XA1,XB1,XA2,XB2,THETA
C
      WRITE(8,100) T,TSTEP,NXNODE,NTNODE
      WRITE(8,200) ((STRSL(II,JJ),JJ=1,3),II=1,NXNODE)
      WRITE(8,300) ((XNODE(II,JJ),JJ=1,3),II=1,NXNODE)
      WRITE(8,300) ((TNODE(II,JJ),JJ=1,3),II=1,NTNODE)
      WRITE(8,350) ((ELMNTX(II,JJ),II=1,4),JJ=1,80)
      WRITE(8,350) ((ELMNTT(II,JJ),II=1,4),JJ=1,80)
      WRITE(8,400) (XA1(II),II=1,4)
      WRITE(8,400) (XA2(II),II=1,4)
      WRITE(8,400) (XB1(II),II=1,4)
      WRITE(8,400) (XB2(II),II=1,4)
C
  100 FORMAT(' ',E15.4,3I2)
```

```
3500      200 FORMAT(' ',3E15.4)
3600      300 FORMAT(' ',3E15.4)
3700      350 FORMAT(' ',4I2)
3800      400 FORMAT(' ',4F15.4)
3900    C
4000          RETURN
4100          END
```

```
C  FUNCTION F
C
C                  THIS SUBPROGRAM COMPUTES THE APPROPRIATE
C              VALUE OF THE INFLUENCE FUNCTION FOR A
C              DISLOCATION NEAR AN INTERFACE.
C
      REAL FUNCTION F*8(J,K,ISURF,RR,ALPHA,BETA)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER RR
      REAL*8 MU
      INTEGER ALPHA,BETA,ELMNTT(4,80),ELMNTX(4,80)
      DIMENSION X(2),T(2),XNODE(80,3),TNODE(80,3)
      DIMENSION XA1(4),XB1(4),XA2(4),XB2(4)
C
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
      COMMON /START/ IOLD,JOLD
      COMMON /ENDPTS/ XA1,XB1,XA2,XB2,THETA
C
      SIN(Q)=DSIN(Q)
      COS(Q)=DCOS(Q)
      ATAN(Q)=DATAN(Q)
      SQRT(Q)=DSQRT(Q)
      ABS(Q)=DABS(Q)
C
      X(1)=XNODE(ELMNTX(ISURF,RR),1)
      X(2)=XNODE(ELMNTX(ISURF,RR),2)
      T(1)=TNODE(ELMNTT(J,K),1)
      T(2)=TNODE(ELMNTT(J,K),2)
C
      CALL LOCATE(T,JREG)
      CALL LOCATE(X,IREG)
      IF((IREG.NE.IOLD).AND.(JREG.NE.JOLD)) CALL CONST(IREG,JREG,EMOD,
     1A,B)
C
```

```
3500            X1=X(1)-T(1)
3600            Y1=X(2)-T(2)
3700            Y2=Y1
3800            C=T(1)
3900            X2=X1-2.*C
4000            R1=SQRT(X1**2+Y1**2)
4100            R2=SQRT(X2**2+Y2**2)
4200            C=ABS(C)
4300    C
4400            IF(BETA.NE.1) GO TO 10
4500    C
4600    C       NORMAL DISLOCATION:
4700    C
4800            BX=0.
4900            BY=1.
5000            GO TO 15
5100       10   CONTINUE
5200    C
5300    C       TANGENTIAL DISLOCATION:
5400    C
5500            BX=1.
5600            BY=0.
5700    C
5800       15   CONTINUE
5900            RMOD=EMOD
6000            AA=X1/R1**2
6100            BB=X2/R2**2
6200            DD=Y2/R2**2
6300            CC=Y1/R1**2
6400    C
6500    C            SYY
6600    C
6700            SA=BX*EMOD*(2.*CC*(2.*X1*AA-1.)+(3.*A-B-4.*A*X2*BB)*DD-(4.*A*C/R2**
6800           12)*(2.*X2*Y2*(4.*X2*BB-3.)/R2**2-2.*C*Y2*(4.*X2*BB-1.)/R2**2))
6900    C
```

```
      SB=BY*EMOD*(2.*(3.-2.*X1*AA)*AA-(5.*A+B-4.*A*X2*BB)*BB-(4.*A*C/R2*
     1*2)*(1.-6.*X2*BB+8.*X2*X2*BB*BB+2.*C*BB*(3.-4.*X2*BB)))
C
      SYY=SA+SB
C
C           SXX
C
      SA=BX*EMOD*(-2.*CC*(2.*X1*AA+1.)+(A+B+4.*A*X2*BB)*BB-(4.*A*C/R2**2
     1)*(2.*(2.*X2*BB-1.)-(3.-4.*X2*BB)*4.*BB*X2+(2.*C*BB)*(6.-8.*X2*BB)
     2))
C
      SXX=SA+SB
C
C     SXY
C
      SA=BX*EMOD*(2.*AA*(2.*X1*AA-1.)+(3.*A-B-4*A*X2*BB)*BB-(4.*A*C/R2**
     12)*(1.-8.*X2*BB+8.*BB**2+2.*C*BB*(3.-4.*X2*BB)))

C
      SB=BY*EMOD*(2.*CC*(2.*X1*AA-1.)+(B+A-4.*A*X2*BB)*DD+(2.*A*C/R2**2)
     1*(4.*X2*DD-8.*X2*DD*(1.-2.*X2*BB)+2.*C*DD*(1.-4.*BB)))
C
      SXY=SA+SB
C
C     STRESS TRANSFORMATION:
C
      THETA=3.1415926535898/2.
      IF(XB1(ISURF).NE.XA1(ISURF)) THETA=ATAN((XB2(ISURF)-XA2(ISURF)
     1)/(XB1(ISURF)-XA1(ISURF)))
      THETA=THETA+3.1415926535898/2.
C
      IF(ALPHA.NE.1) GO TO 20
C     SIGMA NN:
C
      F=.5*(SXX+SYY)+.5*(SXX-SYY)*COS(2.*THETA)+SXY*SIN(2.*THETA)
```

```
10500          GO TO 30
10600       20 CONTINUE
10700    C
10800          F=-.5*(SXX-SYY)*SIN(2.*THETA)+SXY*COS(2.*THETA)
10900       30 CONTINUE
11000          RETURN
11100          END
```

```
C  SUBROUTINE INTEG
C
C                  THIS SUBROUTINE COMPUTES THE INTEGRAL OF A
C            FUNCTION EITHER BY THE TRAPEZOIDAL RULE OR BY THE
C            METHOD SHOWN IN D. R. PETERSEN'S S.M. THESIS, INVOLVING
C            FITTING WITH A CHEBYSHEV SERIES.
C
      SUBROUTINE INTEG(DMUCHK,DDCHK,TNODE,NTNODE,XC,NXC,ICHCE,IROW)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*4 A(2,400),B(2,400)
      REAL*8 TNODE(80,3),DMUCHK(80,1),XC(400,3)
      REAL*8 DDCHK(IROW,1),TAB(80,3),F(400,5)
C
C        COMPUTE INTEGRAL OF D(MU)/DT=D(DELTA)/DT:
C
      IF(ICHCE.NE.1) GO TO 20
      SUM=0.
      DO 10 I=2,NTNODE
      DX=TNODE(I,1)-TNODE(I-1,1)
      SUM=SUM+DMUCHK(I,1)*DX-.5*(DMUCHK(I,1)-DMUCHK(I-1,1))*DX
      DDCHK(I,1)=SUM
   10 CONTINUE
      GO TO 1000
C
   20 CONTINUE
      DO 30 I=1,NTNODE
CPLOT      A(1,I)=TNODE(I,1)
CPLOT      A(2,I)=DMUCHK(I,1)
      TAB(I,1)=DMUCHK(I,1)*DSQRT(1.-TNODE(I,1)**2)
   30 CONTINUE
C
      CALL CHEBY(TNODE,XC,NTNODE,NXC,TAB,NTNODE,NTNODE,
     1           2,F,80,400,1)
C
```

```
3500          DO 40 I=1,NXC
3600          B(1,I)=XC(I,1)
3700          B(2,I)=F(I,5)
3800          DDCHK(I,1)=F(I,5)
3900       40 CONTINUE
4000    CPLOT       CALL QPICTR(A,2,NTNODE,QX(1))
4100          CALL QPICTR(B,2,NXC,QX(1))
4200    C
4300     1000 CONTINUE
4400          STOP
4500          END
```

```
C   SUBROUTINE INTERP
C
C                  THIS SUBROUTINE COMPUTES THE OPENING
C              DISLOCATION DENSITY ON CRACK NO. 1 FOR USE
C              IN FUTURE QUASI-STATIC COMPUTATIONS.
C
      SUBROUTINE INTERP(ALOC,TNODE,NTNODE,MU)
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 MU(80,3)
C
      DIMENSION ALOC(80,3),TNODE(80,3)
      DIMENSION XA1(4),XB1(4),XA2(4),XB2(4)
      DIMENSION ZETA(4,80),NPTS(4)
      COMMON /ENDPTS/ XA1,XB1, A2, B2, HE A
      SQRT(Q)=DSQRT(Q)
C
      DO 10 I=1,NTNODE
      MU(I,1)=ALOC(I,1)/SQRT(1.-ZETA(1,I)*ZETA(1,I))
   10 CONTINUE
C
      RETURN
      END
```

```
C  SUBROUTINE LGRNG
C
C                  THIS SUBROUTINE LOCALLY INTERPOLATES A
C               FUNCTION WITH LAGRANGIAN INTERPOLATING POLYNOMIALS.
C
      SUBROUTINE LGRNG(X,Y,XBAR,YBAR,IDIM,IDEG,IROW)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 L(400)
      DIMENSION X(IROW,3),Y(IROW,3)
C
      ICNT=1
      LIM=IDIM-1
      DO 10 I=1,LIM
      IF(X(I,1).LE.XBAR.AND.XBAR.LE.X(I+1,1)) GO TO 20
   10 CONTINUE
   20 IBAR=I
      IF(XBAR.LE.X(1,1)) IBAR=1
      IF(XBAR.GE.X(IDIM,1)) IBAR=IDIM-1
C
      TEMP=0.
      IMIN=IBAR
      IMAX=IBAR+1
   25 ISWTCH=0
C
   30 DO 50 I=IMIN,IMAX
      L(I)=1.0
      DO 40 J=IMIN,IMAX
      IF(J.EQ.I) GO TO 40
      L(I)=L(I)*(XBAR-X(J,1))/(X(I,1)-X(J,1))
   40 CONTINUE
   50 CONTINUE
      YBAR=0.
      DO 60 I=IMIN,IMAX
      YBAR=YBAR+L(I)*Y(I,1)
```

```
3500        60 CONTINUE
3600     C
3700           IF(ABS(YBAR-TEMP).LE..001) GO TO 90
3800           IF(ICNT.EQ.IDEG) GO TO 90
3900           ICNT=ICNT+1
4000           TEMP=YBAR
4100           ISWTCH=ISWTCH+1
4200           IF(ISWTCH.EQ.2) GO TO 70
4300           IF(IMIN.EQ.1) GO TO 70
4400        69 IMIN=IMIN-1
4500           GO TO 30
4600     C
4700        70 IF(IMAX.EQ.IDIM) GO TO 80
4800           IMAX=IMAX+1
4900           GO TO 25
5000     C
5100        80 IF(IMIN.EQ.1) GO TO 90
5200           GO TO 69
5300     C
5400        90 CONTINUE
5500           RETURN
5600           END
```

```
C  SUBROUTINE LOCATE
C
C                  THIS SUBROUTINE DETERMINES WHICH MATERIAL
C              REGION A POINT WITH GIVEN COORDINATES IS LOCATED
C              IN.
C
      SUBROUTINE LOCATE(TT,IREG)
C
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER TYPE(5)
      DIMENSION RLOC(5,3)
      DIMENSION TT(2)
C
      COMMON /REG/ RLOC,TYPE,NREG
C
      T=TT(1)
      DO 100 I=1,NREG
      LINE=TYPE(I)
      GO TO (10,20,30,40,50),LINE
C
C     INFINITE SPACE:
C
   10 CONTINUE
      IREG=1
      GO TO 100
C
C     HALF PLANE:
C
   20 CONTINUE
      IF((RLOC(I,2).GT.0).AND.(T.GE.RLOC(I,1))) IREG=I
      IF((RLOC(I,2).LT.0).AND.(T.LE.RLOC(I,1))) IREG=I
      GO TO 100
C
   30 CONTINUE
```

```
3500          40 CONTINUE
3600          50 CONTINUE
3700     C
3800         100 CONTINUE
3900             RETURN
4000             END
```

```
C  SUBROUTINE MATRIX
C
C                  THIS SUBROUTINE COMPUTES THE MATRIX
C                FOR PLANE STATIC CRACK PROBLEMS.
C
      SUBROUTINE MATRIX
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 NONGAM,NONDEL,NONP,NONMU
      INTEGER CLROW(6),ELMNTT(4,80),ELMNTX(4,80),ALPHA,BETA
      INTEGER ORDER,R,RMAX
C
      DIMENSION XNODE(80,3),TNODE(80,3),STRSL(80,3),STRSC(80,3)
      DIMENSION A(80,80)
      DIMENSION COEFF(80),SIGMA(80),ACART(80,3),ALOC(80,3)
      DIMENSION XA1(4),XB1(4),XA2(4),XB2(4)
C
      COMMON /ENDPTS/ XA1,XB1,XA2,XB2,THETA
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /ARRAYS/ A,SIGMA,COEFF
      COMMON /OUT/ STRSL,STRSC,ALOC,ACART
      COMMON /DOF/ IDOF
      COMMON /NONDIM/ NONGAM,NONDEL,NONP,NONMU,TAUC
C
      SQRT(Q)=DSQRT(Q)
C
C
      II=0
      DO 105 IEL=1,NELMNT
      RMAX=ELMNTX(IEL,1)+1
      DO 100 ALPHA=1,IDOF
      DO 95 R=2,RMAX
      II=II+1
      JJ=0
```

```
3500            DO 90 J=1,NELMNT
3600            DO 85 BETA=1,IDOF
3700            KMAX=ELMNTT(J,1)+1
3800            DO 80 K=2,KMAX
3900            JJ=JJ+1
4000      C
4100            SCALE=(XB1(J)-XA1(J))**2+(XB2(J)-XA2(J))**2
4200            A(II,JJ)=3.1415926535898*F(J,K,IEL,R,ALPHA,BETA)*SQRT(SCALE)
4300            A(II,JJ)=NONGAM*A(II,JJ)/(2.*ELMNTT(J,1))
4400      C
4500         80 CONTINUE
4600         85 CONTINUE
4700         90 CONTINUE
4800         95 CONTINUE
4900            II=II+1
5000        100 CONTINUE
5100        105 CONTINUE
5200            RETURN
5300            END
```

```
C  SUBROUTINE MOVFL
C
C                THIS SUBROUTINE PERFORMS THE COMPUTATIONS
C             FOR THE QUASI-STATIC FLUID FRONT ADVANCEMENT
C             PROBLEM (STATIONARY CRACK).  NOTE THAT THE ONLY
C             OPTION CURRENTLY SUPPORTED IS ALPHA=1.0 .
C
C
      SUBROUTINE MOVFL(MU,BB)
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 M(80,80),M1(80,80),M2(80,80),M3(80,80),M4(80,80)
      REAL*8 M5(80,80),,MU(80,3),M6(80,80),KAPPA1
      REAL*4 XY(2,80),XSCL(4),DUMMY
      INTEGER R,S,ORDER,ELMNTT,ELMNTX,Q,TSTEP,RINDEX,RFRONT,RECOL
C
      DIMENSION A(80,80),APRIME(80,80),B(80,80),TEMP2(80,80),DDELDT(80)
      DIMENSION C(80,80),CPRIME(80,80),D(80,80),E(80,80)
      DIMENSION F(80,80),G(80,80),H(80,80),SA(80,80),T(80,80),DELTA0(80)
      DIMENSION DELTA1(80),P0(80),P1(80),TEMP(80,80),TEMP1(80,80)
      DIMENSION BB(80,80),YNODE(80,3),RR(80),YT(80),DDEL0(80),DDEL1(80)
      DIMENSION DEL1(80,80),DEL0(80,80),BPRIME(80,80),PPRIME(80)
      DIMENSION B1(80,80),BPRIM1(80,80)
C
      COMMON /OUT/ STRSL(80,3),STRSC(80,3),ALOC(80,3),ACART(80,3)
      COMMON /BKPING/ XNODE(80,3),TNODE(80,3),ELMNTT(4,80),ELMNTX(4,80)
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /TIME/ TSTART,FTIME,DT,TSTEP,TIME
      COMMON /FILL/ LFRONT,RFRONT
C
      TCH(N,X)=DCOS(DFLOAT(N)*DACOS(X))
      TCHPR(N,X)=N*DSIN(DFLOAT(N)*DACOS(X))/DSQRT(1.-X**2)
      SIGN(X)=X/DABS(X)
C
```

```
      PI=3.1415926535898
      PHIL=1.0
C      READ(5,1) ALPHA,BETA,ILIM
C    1 FORMAT(2F10.4,I3)
      ALPHA=1.
      ILIM=9
C          *** TEMPORARY CARD ***
      TAUC=1.
C          **********************
      NLNODE=NTNODE+1
      NMNODE=NLNODE
      NSIZE=2*NLNODE
      ICOUNT=0
      RFCOL=RFRONT+1
C
      NT2=NTNODE/2
      NL2=NLNODE/2
      ITEST=2.*NL2
      IF(ITEST.LT.NLNODE) NL2=NL2+1
C
      NM2=NMNODE/2
      ITEST=2*NM2
      IF(ITEST.LT.NMNODE) NM2=NM2+1
C
      CALL CONST(IREG,JREG,EMOD,AA,BB)
      EMOD=2.*EMOD
C
      DO 5 I=1,NLNODE
      ARG=PI*(2.*I-1.)/(2.*NLNODE)
      YNODE(I,1)=-DCOS(ARG)
    5 CONTINUE
C
      DO 19 I=1,NSIZE
      APRIME(I,I)=1.
      DELO(I,I)=1.
```

```
      DEL1(I,I)=1.
      E(I,I)=1.
      F(I,I)=1.
      H(I,I)=1.
      SA(I,I)=1.
      T(I,I)=1.
   19 CONTINUE
C
      CALL TRANS3(XNODE,STRSL,NXNODE,YNODE,PO,NLNODE,80,80,1)
C
C
    6 CONTINUE
      CALL INTEG(MU,DELTAO,TNODE,NTNODE,YNODE,NLNODE,0,80)
      SUM=0.
C
C
C
C
      DO 15 L=1,NLNODE
      DO 15 S=1,NLNODE
      APRIME(L,S)=TCH(L-1,YNODE(S,1))*2./(NLNODE)
   15 CONTINUE
C      WRITE(6,500) ((APRIME(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
C      WRITE(6,501)
C  500 FORMAT(4(5E15.4,/),///)
C  501 FORMAT('1')
C
      DO 40 R=1,NXNODE
      DO 40 J=1,NMNODE
      CPRIME(R,J)=TCHPR(J,XNODE(R,1))
   40 CONTINUE
C
      RSTART=NMNODE+1
      RSTOP=NMNODE+NXNODE
      ISTART=NMNODE+1
```

```
10500           ISTOP=2*NMNODE
10600           RINDEX=0
10700           DO 25 R=RSTART,RSTOP
10800           RINDEX=RINDEX+1
10900           JINDEX=0
11000           DO 20 J=ISTART,ISTOP
11100           JINDEX=JINDEX+1
11200           CPRIME(R,J)=TCH(JINDEX-1,YNODE(RINDEX,1))
11300           IF(JINDEX.EQ.1) CPRIME(R,J)=.5*CPRIME(R,J)
11400        20 CONTINUE
11500        25 CONTINUE
11600     C
11700     C       WRITE(6,500) ((CPRIME(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
11800     C       WRITE(6,501)
11900     C
12000     C
12100           DO 45 K=1,NTNODE
12200           DO 45 J=1,NMNODE
12300           C(K,J)=TCHPR(J,TNODE(K,1))
12400        45 CONTINUE
12500     C
12600           RSTOP=NMNODE+NTNODE
12700           KINDEX=0
12800           DO 35 K=RSTART,RSTOP
12900           KINDEX=KINDEX+1
13000           JINDEX=0
13100           DO 30 J=ISTART,ISTOP
13200           JINDEX=JINDEX+1
13300           C(K,J)=TCH(JINDEX-1,TNODE(KINDEX,1))
13400           IF(JINDEX.EQ.1) C(K,J)=.5*C(K,J)
13500        30 CONTINUE
13600        35 CONTINUE
13700     C
13800     C       WRITE(6,500) ((C(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
13900     C       WRITE(6,501)
```

```
14000    C
14100          DO 65 J=1,NMNODE
14200          DO 60 K=1,NMNODE
14300          D(J,K)=2.*TCH(J,YNODE(K,1))/NLNODE
14400       60 CONTINUE
14500       65 CONTINUE
14600    C
14700          RSTOP=2*NMNODE
14800          ISTOP=2*NMNODE
14900          JINDEX=0
15000          DO 85 J=RSTART,RSTOP
15100          JINDEX=JINDEX+1
15200          KINDEX=0
15300          DO 80 K=ISTART,ISTOP
15400          KINDEX=KINDEX+1
15500          D(J,K)=2.*TCH(JINDEX-1,YNODE(KINDEX,1))/NLNODE
15600       80 CONTINUE
15700       85 CONTINUE
15800    C
15900    C       WRITE(6,500) ((D(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
16000    C       WRITE(6,501)
16100    C
16200    C
16300          DO 50 K=1,NMNODE
16400          DO 50 L=1,NLNODE
16500          E(K,L)=TCHPR(L,YNODE(K,1))
16600       50 CONTINUE
16700    C       WRITE(6,500) ((E(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
16800    C       WRITE(6,501)
16900    C
17000    C
17100          DO 55 L=1,NLNODE
17200          DO 55 S=1,NLNODE
17300          F(L,S)=TCH(L,YNODE(S,1))*2./NLNODE
17400       55 CONTINUE
```

```
C       WRITE(6,500) ((F(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
C       WRITE(6,501)
C
        DO 75 S=1,NLNODE
        DO 70 Q=1,NLNODE
        H(S,Q)=0.
        SA(S,Q)=0.
        T(S,Q)=0.
   70 CONTINUE
C
        H(S,S)=-SIGN(YNODE(S,1))
        H(S,NL2)=H(S,NL2)+SIGN(YNODE(S,1))
C
        SA(S,S)=1.
        SA(S,NM2)=SA(S,NM2)-SIGN(YNODE(S,1))
C
        T(S,S)=-SIGN(YNODE(S,1))
C
   75 CONTINUE
C       WRITE(6,500) ((H(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
C       WRITE(6,501)
C       WRITE(6,500) ((SA(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
C       WRITE(6,501)
C       WRITE(6,500) ((T(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
C       WRITE(6,501)
C
C
        CALL MULT(D,NSIZE,NSIZE,SA,NSIZE,NSIZE,M1,IER)
C
C
        CALL MULT(T,NSIZE,NSIZE,E,NSIZE,NSIZE,TEMP,IER)
        CALL MULT(TEMP,NSIZE,NSIZE,F,NSIZE,NSIZE,TEMP1,IER)
        CALL MULT(TEMP1,NSIZE,NSIZE,H,NSIZE,NSIZE,M2,IER)
C
C
```

```
      CALL MULT(CPRIME,NSIZE,NSIZE,D,NSIZE,NSIZE,TEMP,IER)
      CALL MULT(TEMP,NSIZE,NSIZE,SA,NSIZE,NSIZE,M3,IER)
C
      CALL MULT(T,NSIZE,NSIZE,CPRIME,NSIZE,NSIZE,TEMP,IER)
      CALL MULT(TEMP,NSIZE,NSIZE,F,NSIZE,NSIZE,TEMP1,IER)
      CALL MULT(TEMP1,NSIZE,NSIZE,H,NSIZE,NSIZE,M6,IER)
C
CTEST TEST TEST TEST TEST TEST
C      WRITE(16,500) ((C(II,JJ),JJ=1,NMNODE),II=1,NXNODE)
C      WRITE(17,500) ((CPRIME(II,JJ),JJ=1,NMNODE),II=1,NMNODE)
C      WRITE(18,500) ((D(II,JJ),JJ=1,NMNODE),II=1,NMNODE)
C      WRITE(19,500) ((M3(II,JJ),JJ=1,NMNODE),II=1,NMNODE)
C      WRITE(20,500) ((M5(II,JJ),JJ=1,NMNODE),II=1,NMNODE)
C      WRITE(21,500) ((M2(II,JJ),JJ=1,NMNODE),II=1,NXNODE)
C      WRITE(22,500) ((M4(II,JJ),JJ=1,NMNODE),II=1,NMNODE)
CTEST TEST TEST TEST TEST TEST
C
C
C
C=======================================================================
C=======================================================================
C=======================================================================
C=======================================================================
C
C
C
C
  100 CONTINUE
C
C
      DO 101 I=1,NLNODE
      XY(1,I)=YNODE(I,1)
      XY(2,I)=PO(I)
      WRITE(6,467) YNODE(I,1),PO(I)
  101 CONTINUE
```

```
24500           CALL QPICTR(XY,2,NLNODE,QX(1))
24600           ISCL=-2
24700           XSCL(1)=-1.0
24800           XSCL(2)=1.0
24900           XSCL(3)=-.10
25000           XSCL(4)=1.2
25100           CALL QPICTR(XY,2,NLNODE,QX(1),QISCL(ISCL),QXSCL(XSCL))
25200           WRITE(6,468)
25300       467 FORMAT(2E15.4)
25400       468 FORMAT(////)
25500     C
25600           XSCL(4)=1.5
25700           DO 102 I=1,NLNODE
25800           XY(1,I)=YNODE(I,1)
25900           XY(2,I)=DELTA1(I)
26000       102 CONTINUE
26100     CX      CALL QPICTR(XY,2,NLNODE,QTNIT(DUMMY),QX(1))
26200           CALL QPICTR(XY,2,NLNODE,QX(1),QISCL(ISCL),QXSCL(XSCL))
26300     C
26400           DO 103 I=1,NLNODE
26500           XY(1,I)=YNODE(I,1)
26600           XY(2,I)=DDELDT(I)
26700       103 CONTINUE
26800           CALL QPICTR(XY,2,NLNODE,QINIT(DUMMY),QX(1))
26900     C
27000           IF(ICOUNT.GE.ILIM)
27100          1GO TO 1000
27200     C
27300           DO 850 I=1,NSIZE
27400           DO 845 J=1,NSIZE
27500           A(I,J)=0.
27600           B(I,J)=0.
27700           BPRIME(I,J)=0.
27800           G(I,J)=0.
27900       845 CONTINUE
```

```
28000      850 CONTINUE
28100    C
28200           DO 110 J=1,NMNODE
28300           DO 105 K=1,NMNODE
28400           DEL0(J,K)=0.
28500           DEL1(J,K)=0.
28600      105  CONTINUE
28700           DEL0(J,J)=DELTA0(J)**3
28800           DEL1(J,J)=DELTA1(J)**3
28900      110  CONTINUE
29000    C
29100    C                    **** COMPUTE A ****
29200    C
29300    C
29400           DO 112 R=LFRONT,RFRONT
29500           DO 112 L=1,NLNODE
29600           A(R,L)=TCH(L-1,XNODE(R,1))
29700           IF(L.EQ.1) A(R,L)=.5*A(R,L)
29800      112  CONTINUE
29900    C
30000           DO 115 I=1,NMNODE
30100           A(NTNODE,I)=0.
30200           A(NMNODE,I)=0.
30300      115  CONTINUE
30400    C       WRITE(6,500) ((A(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
30500    C       WRITE(6,501)
30600    C
30700    C                    **** COMPUTE B ****
30800    C
30900    C              FIRST QUADRANT:
31000           DO 125 R=1,NXNODE
31100           IF((R.LT.LFRONT).OR.(R.GT.RFRONT)) GO TO 125
31200    C
31300           DO 120 I=1,NTNODE
31400           IF((I.LT.LFRONT).OR.(I.GT.RFCOL)) GO TO 120
```

```
31500          B(R,I)=DSQRT(1.-TNODE(I,1)**2)
31600         1        /((XNODE(R,1)-TNODE(I,1))**2)
31700          B(R,I)=PI*EMOD*B(R,I)/(NTNODE*TAUC)
31800        120 CONTINUE
31900        125 CONTINUE
32000      C
32100      C                SECOND QUADRANT:
32200          ISTART=NMNODE+1
32300          ISTOP=2.*NMNODE
32400          DO 135 R=1,NXNODE
32500          IF((R.LT.LFRONT).OR.(R.GT.RFRONT)) GO TO 135
32600          IINDEX=0
32700      C
32800          DO 130 I=ISTART,ISTOP
32900          IINDEX=IINDEX+1
33000          IF((LFRONT.LE.IINDEX).AND.(IINDEX.LE.RFCOL).OR.
33100         1(IINDEX.GT.NTNODE)) GO TO 130
33200      C
33300          B(R,I)=DSQRT(1.-TNODE(IINDEX,1)**2)
33400         1/((XNODE(R,1)-TNODE(IINDEX,1))**2)
33500          B(R,I)=PI*EMOD*B(R,I)/NTNODE
33600      C
33700        130 CONTINUE
33800        135 CONTINUE
33900      C
34000          DO 137 I=1,ISTOP
34100          B(NTNODE,I)=0.
34200        137 CONTINUE
34300          B(NTNODE,NT2-2)=1.
34400          B(NTNODE,NT2+3)=-1.
34500      C
34600      C                THIRD QUADRANT:
34700          RSTART=NMNODE+1
34800          RSTOP=2*NMNODE
34900          RINDEX=0
```

```
35000           DO 145 R=RSTART,RSTOP
35100           RINDEX=RINDEX+1
35200           IF((LFRONT.LE.RINDEX).AND.(RINDEX.LE.RFRONT).OR.
35300          1(RINDEX.GT.NXNODE)) GO TO 145
35400           B(R,RINDEX)=0.
35500     C
35600           DO 140 I=1,NTNODE
35700           IF((I.LT.LFRONT).OR.(I.GT.RFCOL)) GO TO 140
35800           B(R,I)=DSQRT(1.-TNODE(I,1)**2)
35900          1        /((XNODE(RINDEX,1)-TNODE(I,1))**2)
36000           B(R,I)=PI*EMOD*B(R,I)/(NTNODE*TAUC)
36100     C
36200        140 CONTINUE
36300        145 CONTINUE
36400     C
36500     C                    FOURTH QUADRANT:
36600           RINDEX=0
36700           DO 155 R=RSTART,RSTOP
36800           RINDEX=RINDEX+1
36900     CX       B(R,R)=-1.
37000           IF((LFRONT.LE.RINDEX).AND.(RINDEX.LE.RFRONT).OR.
37100          1(RINDEX.GT.NXNODE)) GO TO 155
37200           B(R,R)=0.
37300           IINDEX=0
37400     C
37500           DO 150 I=ISTART,ISTOP
37600           IINDEX=IINDEX+1
37700           IF((LFRONT.LE.IINDEX).AND.(IINDEX.LE.RFCOL).OR.
37800          1(IINDEX.GT.NTNODE)) GO TO 150
37900     C
38000           B(R,I)=DSQRT(1.-TNODE(IINDEX,1)**2)
38100          1        /((XNODE(RINDEX,1)-TNODE(IINDEX,1))**2)
38200           B(R,I)=PI*EMOD*B(R,I)/NTNODE
38300     C
38400        150 CONTINUE
```

```
38500      155 CONTINUE
38600    C      WRITE(6,500) ((B(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
38700    C      WRITE(6,501)
38800    C
38900    C                  **** COMPUTE BPRIME ****
39000    C
39100    C            FIRST QUADRANT:
39200          DO 170 R=1,NXNODE
39300          IF((R.LT.LFRONT).OR.(R.GT.RFRONT)) GO TO 170
39400    C
39500          SUM=0.
39600          DO 165 I=1,NTNODE
39700          SUM=SUM+DSQRT(1.-TNODE(I,1)**2)
39800         1        /((XNODE(R,1)-TNODE(I,1))**2)
39900      165 CONTINUE
40000          BPRIME(R,R)=PI*EMOD*SUM/(NTNODE*TAUC)
40100      170 CONTINUE
40200    C
40300    C            THIRD QUADRANT:
40400          RINDEX=0
40500          DO 172 R=RSTART,RSTOP
40600          RINDEX=RINDEX+1
40700          BPRIME(R,RINDEX)=1.
40800          IF((LFRONT.LE.RINDEX).AND.(RINDEX.LE.RFRONT)) GO TO 172
40900          BPRIME(R,RINDEX)=0.
41000      172 CONTINUE
41100    C
41200    C            FOURTH QUADRANT:
41300          RINDEX=0
41400          DO 180 R=RSTART,RSTOP
41500          RINDEX=RINDEX+1
41600          BPRIME(R,R)=-1.
41700          IF((LFRONT.LE.RINDEX).AND.(RINDEX.LE.RFRONT)) GO TO 180
41800          BPRIME(R,R)=0.
41900    C
```

```
42000           SUM=0.
42100           DO 175 I=1,NTNODE
42200           SUM=SUM+DSQRT(1.-TNODE(I,1)**2)
42300          1        /((XNODE(RINDEX,1)-TNODE(I,1))**2)
42400       175 CONTINUE
42500           BPRIME(R,R)=PI*EMOD*SUM/NTNODE
42600       180 CONTINUE
42700     C
42800           DO 183 I=1,ISTOP
42900           BPRIME(NTNODE,I)=0.
43000       183 CONTINUE
43100     C      WRITE(6,500) ((BPRIME(II,JJ),JJ=1,NSIZF),II=1,NSIZF)
43200     C      WRITE(6,501)
43300     C
43400     C
43500     C                **** COMPUTE G ****
43600     C
43700     C              FIRST QUADRANT:
43800           DO 190 R=1,NXNODE
43900           DO 185 J=1,NXNODE
44000           G(R,J)=0.
44100       185 CONTINUE
44200           IF((R.LT.LFRONT).OR.(R.GT.RFRONT)) GO TO 190
44300           G(R,R)=EMOD*((1./(XNODE(R,1)-1.))
44400          1        -(1./(XNODE(R,1)+1.)))
44500           G(NTNODE,R)=0.
44600           G(NTNODE+1,R)=0.
44700       190 CONTINUE
44800     C
44900     C              FOURTH QUADRANT:
45000           RINDEX=0
45100           DO 195 R=RSTART,RSTOP
45200           RINDEX=RINDEX+1
45300           G(R,R)=0.
45400           IF((LFRONT.LE.RINDEX).AND.(RINDEX.LE.RFRONT)) GO TO 195
```

```
45500           G(R,R)=EMOD*((1./(XNODE(RINDEX,1)-1.))
45600          1        -(1./(XNODE(RINDEX,1)+1.)))
45700         195 CONTINUE
45800       C
45900           DO 196 I=1,ISTOP
46000           G(NTNODE,I)=0.
46100         196 CONTINUE
46200       C
46300       C    WRITE(6,500) ((G(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
46400       C    WRITE(6,501)
46500       C
46600       C
46700       C                  **** COMPUTE M4 ****
46800       C
46900           CALL MULT(A,NSIZE,NSIZE,APRIME,NSIZE,NSIZE,M4,IER)
47000       C
47100       C                  **** COMPUTE M5 ****
47200       C
47300           CALL MULT(B,NSIZE,NSIZE,C,NSIZE,NSIZE,TEMP,IER)
47400           FACTOR=-1.
47500           CALL MULT(FACTOR,1,1,BPRIME,NSIZE,NSIZE,TEMP1,IER)
47600           CALL MULT(TEMP1,NSIZE,NSIZE,CPRIME,NSIZE,NSIZE,TEMP2,IER)
47700           CALL ADD(TEMP,NSIZE,NSIZE,TEMP2,NSIZE,NSIZE,M5,IER)
47800       C
47900       C                  **** COMPUTE DT ****
48000       C
48100           IF(LFRONT.EQ.1) GO TO 820
48200           CALL MULT(M6,NSIZE,NSIZE,P0,NSIZE,1,PPRIME,IER)
48300           DX=XNODE(RFRONT+1,1)-XNODE(RFRONT,1)
48400           V=PPRIME(RFRONT)*(DELTA0(RFRONT)**2)
48500           DT=-DX/V
48600         820 CONTINUE
48700       C
48800       C                  **** COMPUTE M ****
48900       C
```

```
49000    C
49100    C
49200          FACTOR=ALPHA*DT
49300    C
49400          CALL MULT(FACTOR,1,1,M5,NSIZE,NSIZE,TEMP,IER)
49500          CALL MULT(TEMP,NSIZE,NSIZE,M1,NSIZE,NSIZE,TEMP1,IER)
49600          CALL MULT(TEMP1,NSIZE,NSIZE,DEL1,NSIZE,NSIZE,TEMP2,IER)
49700          CALL MULT(TEMP2,NSIZE,NSIZE,M2,NSIZE,NSIZE,TEMP,IER)
49800    C
49900    C
50000          CALL ADD(M4,NSIZE,NSIZE,TEMP,NSIZE,NSIZE,TEMP1,IER)
50100    C
50200    C
50300          FACTOR=FACTOR
50400          CALL MULT(FACTOR,1,1,G,NSIZE,NSIZE,TEMP,IER)
50500          CALL MULT(TEMP,NSIZE,NSIZE,M3,NSIZE,NSIZE,TEMP2,IER)
50600          CALL MULT(TEMP2,NSIZE,NSIZE,DEL1,NSIZE,NSIZE,TEMP,IER)
50700          CALL MULT(TEMP,NSIZE,NSIZE,M2,NSIZE,NSIZE,TEMP2,IER)
50800    C
50900          CALL ADD(TEMP1,NSIZE,NSIZE,TEMP2,NSIZE,NSIZE,M,IER)
51000    C
51100    C                    **** COMPUTE BR ****
51200    C
51300          FACTOR=(1.-ALPHA)*DT
51400          IF(ALPHA.NE.1) GO TO 800
51500          CALL MULT(M4,NSIZE,NSIZE,PO,NSIZE,1,BR,IER)
51600          GO TO 810
51700      800 CONTINUE
51800    C
51900          CALL MULT(FACTOR,1,1,M5,NSIZE,NSIZE,TEMP,IER)
52000          CALL MULT(TEMP,NSIZE,NSIZE,M1,NSIZE,NSIZE,TEMP1,IER)
52100          CALL MULT(TEMP1,NSIZE,NSIZE,DELO,NSIZE,NSIZE,TEMP2,IER)
52200          CALL MULT(TEMP2,NSIZE,NSIZE,M2,NSIZE,NSIZE,TEMP,IER)
52300    C
52400          CALL ADD(M4,NSIZE,NSIZE,TEMP,NSIZE,NSIZE,TEMP1,IER)
```

```
C
C
      FACTOR=FACTOR
      CALL MULT(FACTOR,1,1,G,NSIZE,NSIZE,TEMP,IER)
      CALL MULT(TEMP,NSIZE,NSIZE,M3,NSIZE,NSIZE,TEMP2,IER)
      CALL MULT(TEMP2,NSIZE,NSIZE,DELO,NSIZE,NSIZE,TEMP,IER)
      CALL MULT(TEMP,NSIZE,NSIZE,M2,NSIZE,NSIZE,TEMP2,IER)
C
C
      CALL ADD(TEMP1,NSIZE,NSIZE,TEMP2,NSIZE,NSIZE,TEMP,IER)
      CALL MULT(TEMP,NSIZE,NSIZE,PO,NSIZE,1,RR,IER)
  810 CONTINUE
C
C                    **** APPLY BOUNDARY CONDITIONS ****
C
      DO 300 R=1,NXNODE
      IF((LFRONT.LE.R).AND.(R.LE.RFRONT)) GO TO 300
      DO 295 I=1,NSIZE
      M(R,I)=0.
  295 CONTINUE
      IF(R.LT.LFRONT) M(R,R)=1.
      IF(R.GT.RFRONT) M(R,R+2)=1.
      RR(R)=0.
  300 CONTINUE
      DO 671 JJ=1,NSIZE
      M(NSIZE-1,JJ)=0.
      M(NSIZE,JJ)=0.
      M(NMNODE,JJ)=0.
  671 CONTINUE
      M(NSIZE-1,NLNODE+1)=1.
      M(NSIZE,NSIZE)=1.
      RR(NSIZE-1)=0.
      RR(NSIZE)=0.
C
      M(NMNODE,NM2)=1.
```

```
56000           RR(NMNODE)=PBHL
56100     C      WRITE(6,500) ((M(II,JJ),JJ=1,NSIZE),II=1,NSIZE)
56200     C      WRITE(6,501)
56300     C      WRITE(6,500) (RR(II),II=1,NSIZE)
56400     C
56500     C                    **** SOLVE SYSTEM ****
56600     C
56700           CALL SOLVE(M,RR,P1,NSIZE)
56800     C
56900     C
57000     C                    **** DECOMPOSE RESULTS ****
57100     C
57200           INDEX=NMNODE+1
57300           DO 200 I=1,NMNODE
57400           PO(I)=P1(I)
57500           DDELDT(I)=P1(INDEX)
57600           DELTA1(I)=DELTA1(I)+DDELDT(I)*DT
57700           DELTAO(I)=DELTA1(I)
57800           PO(INDEX)=P1(INDEX)
57900           INDEX=INDEX+1
58000       200 CONTINUE
58100     C      WRITE(6,501)
58200     C      WRITE(6,500) (PO(II),II=1,NSIZE)
58300     C
58400     C
58500     C
58600           TIME=TIME+DT
58700           ICOUNT=ICOUNT+1
58800           IF(LFRONT.EQ.1) GO TO 830
58900           RFRONT=RFRONT+1
59000           LFRONT=LFRONT-1
59100           RFCOL=RFCOL+1
59200       830 CONTINUE
59300           WRITE(6,572) DT,TIME,RFRONT
59400       572 FORMAT('DT=',E15.4,'TIME=',E15.4,'RFRONT=',I3)
```

```
59500           GO TO 100
59600     C
59700     C


  100     1000 CONTINUE
  200          RETURN
  300          END
```

```
C  SUBROUTINE MULT
C
C                  THIS SUBROUTINE MULTIPLIES TWO MATRICES, OR
C              ONE MATRIX BY A SCALAR.
C
      SUBROUTINE MULT(A,IROW,ICOL, B,JROW,JCOL, PROD, IER)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(80,80),B(80,80),PROD(80,80)
C
      IF(ICOL.EQ.JROW) GO TO 10
      IF((IROW.EQ.1).AND.(ICOL.EQ.1)) GO TO 100
      IER=1
      GO TO 1000
C
   10 CONTINUE
      DO 40 I=1,IROW
      DO 30 J=1,JCOL
      PROD(I,J)=0.
C
      DO 20 K=1,ICOL
      PROD(I,J)=PROD(I,J)+A(I,K)*B(K,J)
   20 CONTINUE
C
   30 CONTINUE
   40 CONTINUE
      GO TO 1000
C
  100 CONTINUE
C
C                MULTIPLICATION OF B BY A SCALAR
C
      DO 120 I=1,JROW
      DO 120 J=1,JCOL
      PROD(I,J)=A(1,1)*B(I,J)
```

```
3500       120 CONTINUE
3600     C
3700     C
3800      1000 CONTINUE
3900           RETURN
4000           END
```

```
C SUBROUTINE NEWSTR
C
C                THIS SUBROUTINE PERFORMS QUASI-STATIC PRESSURE
C              EVOLUTION COMPUTATIONS FOR STATIONARY CRACKS BY
C              EXPLICIT TIME INTEGRATION.  NEWSTR IS NOT CURRENTLY
C              USED, BUT IS RETAINED FOR FUTURE REFERENCE AND
C              COMPARISONS.
C
      SUBROUTINE NEWSTR(MU,A)
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 MU(80,3),LOAD(2),NONGAM,NONDEL,NONP,NONMU
      REAL*4 D3P(10,400),D3P1(10,400),D3P2(10,400)
      REAL*4 DPLOT,XY(2,400),XSCL(4)
      INTEGER ORDER,ELMNTT(4,80),ELMNTX(4,80)
      INTEGER TSTEP
C
      DIMENSION DELTA(400,5),P(400,5),DERIV(80),DP(80),PINC(80)
      DIMENSION STRSL(80,3),TEMP(80,3)
      DIMENSION TNODE(80,3),XNODE(80,3)
      DIMENSION A(80,80),DMUDT(400,5)
      DIMENSION DPLOT(10,400),DTEMP(400,5)
      DIMENSION ZETA(4,80),NPTS(4)
      DIMENSION XA1(4),XB1(4),XA2(4),XB2(4),XC(400,3)
      DIMENSION XTEMP(400,3),PTEMP(400,5)
      DIMENSION PROUGH(80),PSMTH(80)
      DIMENSION XT(400,3),DMUCHK(80,1),DPCHK(80),DDCHK(80,1)
      DIMENSION STRSC(80,3),ALOC(80,3),ACART(80,3)
C
      COMMON /TEST/ ITEST
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /DEBUG/ DELTA,DPLOT,DMUCHK,PTEMP,DDCHK,DPCHK
      COMMON /GP/ ZETA,NPTS
```

```
      COMMON /ENDPTS/ XA1,XB1,XA2,XB2
      COMMON /MITBR/ JDEG
      COMMON /FLUID/ VISCO,LOAD
      COMMON /CRM114/ ALPHA,BETA,IFREQ
      COMMON /DEL3PR/ DMUDT,XTEMP,NXTEMP,D3P,D3P1,D3P2
      COMMON /OUT/ STRSL,STRSC,ALOC,ACART
      COMMON /TIME/ TSTART,TFIN,DT,TSTEP,T
      COMMON /TCHPTS/ XC,NDEG
      COMMON /NONDIM/ NONGAM,NONDEL,NONP,NONMU,TAUC
C
      SGN(AA,Q)=DSIGN(AA,Q)
      ATAN(Q)=DATAN(Q)
C
C
C
C              COMPUTE CRACK OPENING DISPLACEMENT AND DERIVATIVES,
C           INTERPOLATE PRESSURE AND COMPUTE DERIVATIVES:
C
C
      IBHL=NTNODE/2
      PBH=STRSL(IBHL,1)
      NXTEMP=320
      NDEG=200
      NSIDE=100
C
      X0=-1.00
      DO 5 I=1,NXTEMP
      XTEMP(I,1)=X0+2.*DFLOAT(I)/DFLOAT(NXTEMP)
    5 CONTINUE
C
      DO 6 I=1,NDEG
      XC(I,1)=-DCOS(3.1415926535898*(I-1)/DFLOAT(NDEG-1))
      XC(I,2)=XC(I,1)
    6 CONTINUE
C
```

```
   25 CONTINUE
C
C                              PLOT PRESSURE
C
      DO 26 I=1,NXNODE
      XY(1,I)=XNODE(I,1)
      XY(2,I)=STRSL(I,1)
   26 CONTINUE
      ISCL=-2
      XSCL(1)=-1.0
      XSCL(2)=1.0
      XSCL(3)=0.0
      XSCL(4)=1.5*SNGL(LOAD(1))
      CALL QPICTR(XY,2,NXNODE,QX(1),QISCL(ISCL),QXSCL(XSCL))
C
C
C             EVALUATE DMUDT:
C
C                    TRANSFORM STRSL FROM XNODE TO XTEMP:
C
        CALL TRANS3(XNODE,STRSL,NXNODE,XTEMP,P,320,80,400,1)
CX      WRITE(6,536) ((XTEMP(II,JJ),JJ=1,3),(PTEMP(II,JK),JK=1,5),
CX     1II=1,325)
  536 FORMAT(8E15.4)
C                    EVALUATE P*:
      CALL DIFF(XTEMP,P,320,1)
      WRITE(9,536) ((XTEMP(II,JJ),JJ=1,3),(P(II,JK),JK=1,5),
     1II=1,325)
      DO 22 I=1,NXTEMP
      XT(I,2)=XTEMP(I,1)
      PTEMP(I,2)=P(I,2)
   22 CONTINUE
      CALL TRANS3(XT,P,NXTEMP,XC,PTEMP,NDEG,400,400,2)
      DO 544 II=1,NDEG
      XY(1,II)=XC(II,1)
```

```
      XY(2,II)=PTEMP(II,1)
  544 CONTINUE
      CALL QPICTR(XY,2,NDEG,QINIT(DUMMY),QX(1))
C
C                 COMPUTE DELTA:
C
      CALL INTEG(MU,DELTA,TNODE,NTNODE,XC,NDEG,2,400)
      DO 694 I=1,NDEG
      XY(1,I)=XC(I,1)
      XY(2,I)=DELTA(I,1)
  694 CONTINUE
      CALL QPICTR(XY,2,NDEG,QX(1))
C
C
      WRITE(13,536) ((XC(II,JJ),JJ=1,3),(DELTA(II,JK),JK=1,5),
     1II=1,400)
C
C                   COMPUTE (DELTA**3)*P' :
C
      DO 20 I=1,NDEG
      DMUDT(I,1)=PTEMP(I,1)*DELTA(I,1)**3
      WRITE(13,532) XC(I,1),DMUDT(I,1)
  532 FORMAT(' ',2E15.4)
   20 CONTINUE
C
CX      CONST=DMUDT(NSIDE,1)
      CONST=DMUDT(NSIDE,1)
     1      -((DMUDT(NSIDE,1)-DMUDT(NSIDE-1,1))
     2      /(XC(NSIDE,1)-XC(NSIDE-1,1)))
     3      *XC(NSIDE,1)
C
      DO 81 I=1,NSIDE
      DMUDT(I,1)=DMUDT(I,1)-CONST
   81 CONTINUE
C
```

```
      IMIN=NSIDE+1
CX     CONST=DMUDT(IMIN,1)
      CONST=DMUDT(IMIN,1)
     1      +((DMUDT(IMIN+1,1)-DMUDT(IMIN,1))
     2      /(XC(IMIN+1,1)-XC(IMIN,1)))
     3      *(-XC(IMIN,1))
C
      DO 82 I=IMIN,NDEG
      DMUDT(I,1)=DMUDT(I,1)-CONST
   82 CONTINUE
C
      DO 83 I=1,NDEG
      XY(1,I)=XC(I,1)
      XY(2,I)=DMUDT(I,1)
   83 CONTINUE
      CALL QPICTR(XY,2,NDEG,QINIT(DUMMY),QX(1))
C
C
C
C                    DIFFERENTIATE:
C
      CALL CHEBY(XC,XC,NDEG,NDEG,DMUDT,NDEG,NDEG,2,DTEMP,
     1400,400,0)
C
C
      DO 84 I=1,NDEG
      XY(1,I)=XC(I,1)
      XY(2,I)=DTEMP(I,2)
   84 CONTINUE
      CALL QPICTR(XY,2,NDEG,QX(1))
C
C
      CALL RECNST(XC,DTEMP,400,2,NDEG,0)
      DO 85 I=1,NDEG
      XY(1,I)=XC(I,1)
```

```
      XY(2,I)=DTEMP(I,2)
   85 CONTINUE
      CALL QPICTR(XY,2,NDEG,QX(1))
C
      CALL DIF2(XC,DTEMP,NDEG,TNODE,NTNODE)
CX      CALL REBILD(TNODE,DTEMP,NTNODE)
C
      DO 30 I=1,NTNODE
      XY(1,I)=TNODE(I,1)
      XY(2,I)=DTEMP(I,3)
      WRITE(6,673) TNODE(I,1),DMUDT(I,3)
  673 FORMAT(' ','TNODE=',E15.4,'DMUDT=',E15.4)
      DO 29 J=1,5
      DMUDT(I,J)=DTEMP(I,J)
   29 CONTINUE
      DDCHK(I,1)=DMUDT(I,2)
      DMUCHK(I,1)=DMUDT(I,3)
   30 CONTINUE
      CALL QPICTR(XY,2,NTNODE,QX(1))
C
C
C
C           COMPUTE TIME DERIVATIVE OF PRESSURE:
C
      I=0
      DO 90 IEL=1,NELMNT
      IIMAX=ELMNTX(IEL,1)
      DO 80 II=1,IIMAX
      I=I+1
      DP(I)=0.
      DO 70 J=1,NTNODE
      DP(I)=DP(I)+A(I,J)*DTEMP(J,3)
     1*DSQRT(1.-TNODE(J,1)**2)
   70 CONTINUE
      DPCHK(I)=DP(I)
```

```
   80 CONTINUE
      I=I+1
   90 CONTINUE
C
      DO 95 I=1,NXNODE
      XY(1,I)=XNODE(I,1)
      XY(2,I)=DP(I)
   95 CONTINUE
      ISCL=-2
      XSCL(1)=-1.0
      XSCL(2)=1.0
      XSCL(3)=0.
      XSCL(4)=1.5*PBH
      CALL QPICTR(XY,2,NXNODE,QX(1))
C
C              ADD PRESSURE INCREMENT TO PREVIOUS PRESSURE:
C
      DO 100 I=1,NXNODE
      STRSL(I,1)=STRSL(I,1)+DP(I)*DT
  100 CONTINUE
C
C              FIX UP NEW PRESSURE CURVE:
C
C*************************TEMPORARY CARD*****************************
      STRSL(IBHL,1)=PBH
C*********************************************************************
CNEW        CALL FIXUP(STRSL,DP,LOAD(1),IBHL,NXNODE)
C
C              SMOOTH FINAL PRESSURE CURVE:
C
C       DO 950 II=1,NXNODE
C       PROUGH(II)=STRSL(II,1)
C  950 CONTINUE
C
C      CHECK=10.D 07
```

```
C      DO 962 ICNT=2,15
C       SDEV=RGRESS(XNODE,PROUGH,PSMTH,15,NXNODE)
CX     CHECK=1.5*CHECK
C      IF(SDEV.GE.CHECK) GO TO 963
C      WRITE(6,567) SDEV
C      CHECK=SDEV
C 962 CONTINUE
C 963 CONTINUE
C
C
C      WRITE(6,567) SDEV
C 567 FORMAT(' ','SDEV=',E15.4)
C       DO 975 II=1,NXNODE
C       STRSL(II,1)=PSMTH(II)
C  975 CONTINUE
C********************* TEMPORARY CARD *******************************
CX       IF(NXNODE.GE.1) GO TO 58
C**********************************************************************
C
C              COMPUTE NEW DELTA':
C
       DO 53 I=1,NTNODE
       MU(I,1)=MU(I,1)+DMUDT(I,3)*DT
   53 CONTINUE
C
C              GO THROUGH NEXT TIME STEP:
C
       CALL PSCALC
       CALL OUTPUT
       TSTEP=TSTEP+1
       IF(T.GE.TFIN) GO TO 1000
       T=T+DT
C
       GO TO 25
C***************************** TEMPORARY CARD ***************************
```

```
CX    58 CONTINUE
C*************************************************************************
C
C
 1000 CONTINUE
        RETURN
        END
```

```
C  SUBROUTINE OUTPUT
C
C                  THIS SUBROUTINE CURRENTLY PRODUCES
C               THE PRINTED OUTPUT ASSOCIATED WITH STATIC
C               PROBLEMS.
C
      SUBROUTINE OUTPUT
C
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER ELMNTT(4,80),ELMNTX(4,80)
      INTEGER ORDER
      INTEGER TYPE(5),CLROW(6),CELMNT(6,6),CDIR(6,6)
      INTEGER COL(6,6)
      INTEGER TSTEP
      REAL*8 MU,KAPPA1
      REAL*4 PPLOT,FPLOT,DMUPLT(10,80)
      REAL*4 DPLOT
C
      DIMENSION XA1(4),XB1(4),XA2(4),XB2(4)
      DIMENSION XNODE(80,3),TNODE(80,3)
      DIMENSION RLOC(5,3),ELCON(2,3)
      DIMENSION A(80,80),DDCHK(80,1),DPCHK(80)
      DIMENSION SIGMA(80),COEFF(80)
      DIMENSION A1(4,2),A2(4,2),ALFA(4)
      DIMENSION STRSL(80,3),STRSC(80,3),ALOC(80,3),ACART(80,3)
      DIMENSION ICOL(6)
      DIMENSION C(6,6)
      DIMENSION PPLOT(10,80),FPLOT(10,80)
      DIMENSION DPLOT(10,400)
      DIMENSION DELTA(400,5),DMUCHK(80),P(400,5)
C
      COMMON /ENDPTS/ XA1,XB1,XA2,XB2,THETA
      COMMON /REG/ RLOC,TYPE,NREG
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
```

```
      COMMON /CLOSE/ A1,A2,ALFA,CLROW,NCL,CELMNT,CDIR,COL,C
     1,ITYPE
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /ARRAYS/ A,SIGMA,COEFF
      COMMON /OUT/ STRSL,STRSC,ALOC,ACART
      COMMON /ELAST/ G1,KAPPA1,MU,ELCON
      COMMON /TIME/ TSTART,TFIN,DT,TSTEP,T
      COMMON /PLOTS/ FPLOT,PPLOT,DMUPLT
      COMMON /DEBUG/ DELTA,DPLOT,DMUCHK,P,DDCHK,DPCHK
C
      IF(TSTEP.NE.1) GO TO 783
      WRITE(6,100)
  100 FORMAT('1',T50,'M A T E R I A L S',///,
     1T30,'REGION',T40,'E',T60,'MU',T80,'G',T100,'INTERFACE',/
     2,T30,6('-'),T40,'-',T60,'--',T80,'-',T100,9('-'),/)
      DO 10 II=1,NREG
      WRITE(6,150) II,ELCON(II,1),ELCON(II,2),ELCON(II,3)
     1,RLOC(II,2)
   10 CONTINUE
  150 FORMAT(' ',T35,I2,T35,F15.4,T55,E15.4,T75,E15.4,
     1T95,E15.4)
  783 CONTINUE
      WRITE(6,200) T
  200 FORMAT('1',T50,30('*'),/,T50,'*',T80,'*',/,
     1T50,'*',T55,'TIME = ',F15.4,T80,'*',/,T50,'*',T80,'*',
     2/,T50,30('*'),///)
      WRITE(6,250)
      DO 20 II=1,NXNODE
      WRITE(6,300) II,(XNODE(II,JJ),JJ=1,3),(STRSL(II,JJ),JJ=1,3)
   20 CONTINUE
  250 FORMAT(' ',////,T5,'TNODE',T15,'X',T35,'Y',T55,'Z',
     1T75,'F OPEN',T95,'F SLIDE',/,T5,5('-'),T15,
     2'-',T35,'-',T55,'-',T75,6('-'),T95,7('-'),/)
  300 FORMAT(' ',T5,I2,T10,E10.4,T30,E10.4,T50,E10.4,T70,E10.4
     1,T90,E15.4)
```

```
7000          WRITE(6,350)
7100          DO 30 II=1,NTNODE
7200          WRITE(6,400) II,(TNODE(II,JJ),JJ=1,3),(ALOC(II,JJ),JJ=1,2)
7300       30 CONTINUE
7400      250 FORMAT(' ',T5,'XNODE',T15,'X',T35,'Y',T55,'Z',
7500         1T75,'SNN',T95,'SNT',T105,'THETA',/,T5,5('-'),T15,
7600         2'-',T35,'-',T55,'-',T75,'---',T95,'---',T105,5('-'),/)
7700      300 FORMAT(' ',T5,I2,T10,E10.4,T30,E10.4,T50,E10.4,T70,E10.4
7800         1,T90,E15.4,T100,E15.4)
7900    C
8000          RETURN
8100          END
```

```
C  SUBROUTINE PLOT
C
C                  THIS SUBROUTINE PRODUCES PLOTS OF
C               BOTH STATIC AND QUASI-STATIC CALCULATIONS.
C               PLOT IN TURN CALLS PLOTTING ROUTINES THAT
C               ARE USED AT THE M.I.T. JOINT COMPUTER
C               FACILITY.  PLOT IS CURRENTLY OUT OF DATE
C               BECAUSE OF RECENT CHANGES THAT HAVE BEEN
C               MADE IN THE QUASI-STATIC ROUTINES.
C
      SUBROUTINE PLOT
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*40 XLAB1,XLAB2,XLAB3,XLAB4,XLAB5,XLAB6,XLAB7,XLAB8,
     1             XLAB9,XLAB10,XLAB11,XLAB12,XLAB13,XLAB14,
     2             YLAB1,YLAB2,YLAB3,YLAB4,YLAB5,YLAB6,YLAB7,YLAB8,
     3             YLAB9,YLAB10,YLAB11,YLAB12,YLAB13,YLAB14,
     4             XLAB15,XLAB16,XLAB17,YLAB15,YLAB16,YLAB17
      INTEGER ELMNTT(4,80),ELMNTX(4,80)
      INTEGER ORDER
      INTEGER TYPE(5),CLROW(6),CELMNT(6,6),CDIR(6,6)
      INTEGER COL(6,6)
      INTEGER TSTEP
      REAL*8 MU,KAPPA1
      REAL*4 PPLOT,FPLOT,D3P,D3P1,D3P2
      REAL*4 DPLOT,DMUPLT,DDPLT,DPPLT
      REAL*4 P1PLT(10,80),P2PLT(10,400),P3PLT(10,80),P4PLT(10,80)
      REAL*4 D1PLT(10,80),D2PLT(10,80),D3PLT(10,80),D4PLT(10,80)
C
      DIMENSION XA1(4),XB1(4),XA2(4),XB2(4)
      DIMENSION XC(400,3),DPCHK(80),DDCHK(80,1)
      DIMENSION XNODE(80,3),TNODE(80,3)
      DIMENSION RLOC(5,3),ELCON(2,3)
      DIMENSION A(80,80)
      DIMENSION SIGMA(80),COEFF(80)
```

```
      DIMENSION A1(4,2),A2(4,2),ALFA(4)
      DIMENSION STRSL(80,3),STRSC(80,3),ALOC(80,3),ACART(80,3)
      DIMENSION ICOL(4),XTEMP(400,3),DMUDT(400,5),D3P(10,400)
      DIMENSION C(4,4),D3P1(10,400),D3P2(10,400)
      DIMENSION PPLOT(10,80),FPLOT(10,80)
      DIMENSION DPLOT(10,400),DMUPLT(10,80),F(400,5)
      DIMENSION DELTA(400,5),DMUCHK(80,1),DDPLT(10,80)
      DIMENSION DPPLT(10,80)
C
      COMMON /ENDPTS/ XA1,XB1,XA2,XB2,THETA
      COMMON /TCHPTS/ XC,NDEG
      COMMON /REG/ RLOC,TYPE,NREG
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
      COMMON /CLOSE/ A1,A2,ALFA,CLROW,NCL,CELMNT,CDIR,COL,C
     1,ITYPE
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /ARRAYS/ A,SIGMA,COEFF
      COMMON /OUT/ STRSL,STRSC,ALOC,ACART
      COMMON /ELAST/ G1,KAPPA1,MU,ELCON
      COMMON /TIME/ TSTART,TFIN,DT,TSTEP,T
      COMMON /PLOTS/ FPLOT,PPLOT,DMUPLT
      COMMON /PLOTS1/ DDPLT,DPPLT
      COMMON /DEBUG/ DELTA,DPLOT,DMUCHK,P,DDCHK,DPCHK
      COMMON /PLTEMP/ P1PLT,P2PLT,P3PLT,P4PLT,D1PLT,D2PLT,D3PLT,D4PLT
      COMMON /DEL3PR/ DMUDT,XTEMP,NXTEMP,D3P,D3P1,D3P2
C
C          ASSIGN AXIS LABELS:
C
      XLAB1='PLOT#1   XNODE'
      YLAB1='   FRAC FLUID PRESSURE P(XNODE,T)'
C
      XLAB2='PLOT#2   TNODE'
      YLAB2='   SOLUTION  F(TNODE,T)'
C
      XLAB3='PLOT#3   TNODE'
```

```
 7000          YLAB3='   PLOT OF D(NU)/DT'
 7100    C
 7200          XLAB4='PLOT#4    TNODE'
 7300          YLAB4='  PLOT OF D(DELTA)/DT'
 7400    C
 7500          XLAB5='PLOT#5    XNODE'
 7600          YLAB5='   FRAC. FLUID PRESS. INC. DP'
 7700    C
 7800          XLAB6='PLOT#6    XNODE'
 7900          YLAB6='P  (DIAGNOSTIC)'
 8000    C
 8100          XLAB7='PLOT#7    XNODE'
 8200          YLAB7='P/   (DIAGNOSTIC)'
 8300    C
 8400    CX      XLAB8='PLOT#8    XNODE'
 8500    CX      YLAB8='P//  (DIAGNOSTIC)'
 8600    CX
 8700    CX      XLAB9='PLOT#9    XNODE'
 8800    CX      YLAB9='P///   (DIAGNOSTIC)'
 8900    C
 9000          XLAB10='PLOT#10   TNODE'
 9100          YLAB10='DELTA/  (DIAGNOSTIC)'
 9200    C
 9300    CX      XLAB11='PLOT#11   TNODE'
 9400    CX      YLAB11='DELTA//  (DIAGNOSTIC)'
 9500    CX
 9600    CX      XLAB12='PLOT#12   TNODE'
 9700    CX      YLAB12='DELTA///  (DIAGNOSTIC)'
 9800    C
 9900          XLAB13='PLOT#12   TNODE'
10000          YLAB13='DELTA    (DIAGNOSTIC)'
10100    C
10200          XLAB14='PLOT#13    TNODE'
10300          YLAB14='CRACK OPENING DISP.  (DELTA(TNODE,T))'
10400    C
```

```
10500           XLAB15='PLOT#14     XTEMP'
10600           YLAB15='    (D**3)*P/ '
10700     C
10800           XLAB16='PLOT#15     XTEMP'
10900           YLAB16='    ((D**3)*P/)/ '
11000     C
11100           XLAB17='PLOT#16     XTEMP'
11200           YLAB17='    ((D**3)*P/)//'
11300     C
11400     C
11500     C           PLOT RESULTS:
11600           NP=NTNODE
11700           NP1=NXNODE
11800           NP2=NXTEMP
11900           NP3=NDEG
12000           NP4=NDEG
12100     C
12200     C
12300           DO 10 I=1,NXNODE
12400           PPLOT(1,I)=XNODE(I,1)
12500           DPPLT(1,I)=XNODE(I,1)
12600        10 CONTINUE
12700     C
12800           DO 11 I=1,NTNODE
12900           FPLOT(1,I)=TNODE(I,1)
13000           DMUPLT(1,I)=TNODE(I,1)
13100           DDPLT(1,I)=TNODE(I,1)
13200     C
13300     CX      P1PLT(1,I)=TNODE(I,1)
13400     CX      P2PLT(1,I)=TNODE(I,1)
13500     CX   X   P3PLT(1,I)=TNODE(I,1)
13600     CX      P4PLT(1,I)=TNODE(I,1)
13700     C
13800           D1PLT(1,I)=TNODE(I,1)
13900     CX      D2PLT(1,I)=TNODE(I,1)
```

```
14000    CX       D3PLT(1,I)=TNODE(I,1)
14100    CX       D4PLT(1,I)=TNODE(I,1)
14200    C
14300        11 CONTINUE
14400    C
14500           DO 20 I=1,NDEG
14600           D3P(1,I)=XC(I,1)
14700           DPLOT(1,I)=XC(I,1)
14800           D3P1(1,I)=XC(I,1)
14900        20 CONTINUE
15000    C
15100           DO 30 I=1,NXTEMP
15200           P2PLT(1,I)=XTEMP(I,1)
15300        30 CONTINUE
15400    C
15500           DO 40 I=1,NDEG
15600           D3P2(1,I)=XC(I,1)
15700        40 CONTINUE
15800    C
15900           CALL QPICTR(PPLOT,10,NP1,QINIT(DUMMY),QX(1),QLABEL(11004),
16000          1QXLAB(XLAB1),QYLAB(YLAB1))
16100    C
16200           CALL QPICTR(FPLOT,10,NP,QINIT(DUMMY),QX(1),QLABEL(11004),
16300          1QXLAB(XLAB2),QYLAB(YLAB2))
16400    C
16500           CALL QPICTR(DMUPLT,10,NP,QINIT(DUMMY),QX(1),QLABEL(11004),
16600          1QXLAB(XLAB3),QYLAB(YLAB3))
16700    C
16800           CALL QPICTR(DDPLT,10,NP,QINIT(DUMMY),QX(1),QLABEL(11004),
16900          1QXLAB(XLAB4),QYLAB(YLAB4))
17000    C
17100           CALL QPICTR(DPPLT,10,NP1,QINIT(DUMMY),QX(1),QLABEL(11004),
17200          1QXLAB(XLAB5),QYLAB(YLAB5))
17300    C
17400    CX       CALL QPICTR(P1PLT,10,NP2,QX(1),QLABEL(11004),QXLAB(XLAB6),
```

```
17500    CX      1QYLAB(YLAB6))
17600    C
17700           CALL QPICTR(P2PLT,10,NP2,QINIT(DUMMY),QX(1),QLABEL(11004),
17800           1QXLAB(XLAB7),QYLAB(YLAB7))
17900    C
18000    CX      CALL QPICTR(P3PLT,10,NP2,QX(1),QLABEL(11004),QXLAB(XLAB8),
18100    CX      1QYLAB(YLAB8))
18200    C
18300    CX      CALL QPICTR(P4PLT,10,NP2,QX(1),QLABEL(11004),QXLAB(XLAB9),
18400    CX      1QYLAB(YLAB9))
18500    C
18600    C
18700           CALL QPICTR(D1PLT,10,NP,QINIT(DUMMY),QX(1),QLABEL(11004),
18800           1QXLAB(XLAB10),QYLAB(YLAB10))
18900    C
19000    CX      CALL QPICTR(D2PLT,10,NP2,QX(1),QLABEL(11004),QXLAB(XLAB11),
19100    CX      1QYLAB(YLAB11))
19200    C
19300    CX      CALL QPICTR(D3PLT,10,NP2,QX(1),QLABEL(11004),QXLAB(XLAB12),
19400    CX      1QYLAB(YLAB12))
19500    C
19600    CX      CALL QPICTR(D4PLT,10,NP2,QX(1),QLABEL(11004),QXLAB(XLAB13),
19700    CX      1QYLAB(YLAB13))
19800    C
19900    C
20000    CX      CALL QPICTR(DPLOT,10,NP2,QX(1),QLABEL(11004),QXLAB(XLAB14),
20100    CX      1QYLAB(YLAB14))
20200    C
20300           CALL QPICTR(D3P,10,NP3,QINIT(DUMMY),QX(1),QLABEL(11004),
20400           1QXLAB(XLAB15),QYLAB(YLAB15))
20500    C
20600           CALL QPICTR(D3P1,10,NP3,QINIT(DUMMY),QX(1),QLABEL(11004),
20700           1QXLAB(XLAB16),QYLAB(YLAB16))
20800    C
20900           CALL QPICTR(D3P2,10,NP,QINIT(DUMMY),QX(1),QLABEL(11004),
```

```
21000          1QXLAB(XLAB17),QYLAB(YLAB17))
21100     C
21200           RETURN
21300           END
```

```
C  SUBROUTINE PSCALC
C
C                  THIS SUBROUTINE IS INTENDED TO
C              PERFORM POST-SOLUTION COMPUTATIONS
C              (SUCH AS CALCULATING STRESS INTENSITY
C              FACTORS) AND TO STORE CERTAIN RESULTS
C              FROM EACH TIME STEP FOR COMPOSITE
C              PLOTTING.  LIKE SUBROUTINE PLOT,IT IS
C              SOMEWHAT OUT OF DATE, AND IS BEING
C              RE-WRITTEN.
C
      SUBROUTINE PSCALC
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 MU,KAPPA1,K1A1,K2A1,K1A2,K2A2
      INTEGER ELMNTT(4,80),ELMNTX(4,80)
      INTEGER TSTEP,ORDER
      REAL*4 DPLOT(10,400),FPLOT,DMUPLT,PPLOT
      REAL*4 DDPLT,DPPLT
      REAL*4 D1PLT(10,80),D2PLT(10,80),D3PLT(10,80),D4PLT(10,80)
      REAL*4 P1PLT(10,80),P2PLT(10,400),P3PLT(10,80),P4PLT(10,80)
      REAL*4 D3P(10,400),D3P1(10,400),D3P2(10,400)
C
      DIMENSION XNODE(80,3),TNODE(80,3),XA1(4),XB1(4),XA2(4),XB2(4)
      DIMENSION STRSL(80,3),STRSC(80,3),ALOC(80,3),ACART(80,3)
      DIMENSION ELCON(2,3),DMUDT(400,5),XTEMP(400,3)
      DIMENSION PPLOT(10,80),FPLOT(10,80),DMUPLT(10,80)
      DIMENSION DELTA(400,5),DMUCHK(80,1),P(400,5),XC(400,3)
      DIMENSION DDCHK(80,1),DPCHK(80),DDPLT(10,80),DPPLT(10,80)
C
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
      COMMON /ELAST/ G1,KAPPA1,MU,ELCON
      COMMON /ENDPTS/ XA1,XB1,XA2,XB2,THETA
      COMMON /OUT/ STRSL,STRSC,ALOC,ACART
```

```
3500          COMMON /PLOTS/ FPLOT,PPLOT,DMUPLT
3600          COMMON /PLOTS1/ DDPLT,DPPLT
3700          COMMON /TIME/ TSTART,TFIN,DT,TSTEP,T
3800          COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
3900          COMMON /DEBUG/ DELTA,DPLOT,DMUCHK,P,DDCHK,DPCHK
4000          COMMON /PLTEMP/ P1PLT,P2PLT,P3PLT,P4PLT,D1PLT,D2PLT,D3PLT,D4PLT
4100          COMMON /DEL3PR/ DMUDT,XTEMP,NXTEMP,D3P,D3P1,D3P2
4200          COMMON /TCHPTS/ XC,NDEG
4300    C
4400          SIN(Q)=DSIN(Q)
4500          COS(Q)=DCOS(Q)
4600          ATAN(Q)=DATAN(Q)
4700          SQRT(Q)=DSQRT(Q)
4800    C
4900    C
5000          IST=TSTEP+1
5100          DO 10 I=1,NTNODE
5200          FPLOT(IST,I)=ALOC(I,1)
5300          DMUPLT(IST,I)=DMUCHK(I,1)
5400          DDPLT(IST,I)=DDCHK(I,1)
5500    C
5600          D1PLT(IST,I)=DELTA(I,1)
5700    CX      D2PLT(IST,I)=DELTA(I,2)
5800    CX      D3PLT(IST,I)=DELTA(I,3)
5900    CX      D4PLT(IST,I)=DELTA(I,4)
6000    C
6100          P1PLT(IST,I)=P(I,1)
6200    CX      P3PLT(IST,I)=P(I,3)
6300    CX      P4PLT(IST,I)=P(I,4)
6400          D3P2(IST,I)=DMUDT(I,1)
6500       10 CONTINUE
6600    C
6700          DO 20 I=1,NXNODE
6800          PPLOT(IST,I)=STRSL(I,1)
6900          DPPLT(IST,I)=DPCHK(I)
```

```
7000        20 CONTINUE
7100    C
7200           DO 30 I=1,NDEG
7300           D3P(IST,I)=DMUDT(I,1)
7400    CX       DPLOT(IST,I)=DELTA(I,1)
7500        30 CONTINUE
7600
7700           DO 40 I=1,NDEG
7800           D3P1(IST,I)=DMUDT(I,2)
7900           D3P2(IST,I)=DMUDT(I,3)
8000        40 CONTINUE
8100    C
8200           DO 50 I=1,NXTEMP
8300           P2PLT(IST,I)=P(I,2)
8400        50 CONTINUE
8500    C
8600           RETURN
8700           END
```

```
C  SUBROUTINE RECNST
C
C                  THIS SUBROUTINE FIXES UP THE NEAR-TIP
C              REGIONS OF THE [(DELTA**3)*P']' CURVE BY
C              SPLICING ELLIPTICAL ARCS OF APPROPRIATE LENGTH.
C
      SUBROUTINE RECNST(X,Y,NROW,ICOL,NY,IODD)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(NROW,3),Y(NROW,5),YTEMP(400,5),XTEMP(400,5)
C
      A=Y(22,ICOL)/DSQRT(1.-X(22,1)**2)
      B=Y(NY-21,ICOL)/DSQRT(1.-X(NY-21,1)**2)
C
      DO 10 I=1,21
      Y(I,ICOL)=A*DSQRT(1.-X(I,1)**2)
      Y((NY-21+I),ICOL)=B*DSQRT(1.-X((NY-21+I),ICOL)**2)
   10 CONTINUE
C
      RETURN
      END
```

```
C  SUBROUTINE RESTRT
C
C                THIS SUBROUTINE READS DATA THAT HAS
C              BEEN WRITTEN OUT BY SUBROUTINE DUMP.
C
      SUBROUTINE RESTRT
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER ORDER,TSTEP
      INTEGER ELMNTT(4,80),ELMNTX(4,80)
C
      DIMENSION STRSL(80,3),STRSC(80,3),ALOC(80,3)
      DIMENSION ACART(80,3)
      DIMENSION XNODE(80,3),TNODE(80,3)
      DIMENSION XA1(4),XB1(4),XA2(4),XB2(4)
      DIMENSION ZETA(4,80),NPTS(4)
C
      COMMON /TIME/ TSTART,TFIN,DT,TSTEP,T
      COMMON /OUT/ STRSL,STRSC,ALOC,ACART
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
      COMMON /ENDPTS/ XA1,XB1,XA2,XB2,THETA
      COMMON /GP/ ZETA,NPTS
      COMMON /TEM/ NSURF
C
      READ(8,100) T,TSTEP,NXNODE,NTNODE
      READ(8,200) ((STRSL(II,JJ),JJ=1,3),II=1,NXNODE)
      READ(8,300) ((XNODE(II,JJ),JJ=1,3),II=1,NXNODE)
      READ(8,300) ((TNODE(II,JJ),JJ=1,3),II=1,NTNODE)
      READ(8,350) ((ELMNTX(II,JJ),II=1,4),JJ=1,80)
      READ(8,350) ((ELMNTT(II,JJ),II=1,4),JJ=1,80)
      READ(8,400) (XA1(II),II=1,4)
      READ(8,400) (XA2(II),II=1,4)
      READ(8,400) (XB1(II),II=1,4)
      READ(8,400) (XB2(II),II=1,4)
```

```
3500    C
3600       100 FORMAT(X,E15.4,3I2)
3700       200 FORMAT(X,3E15.4)
3800       300 FORMAT(X,3E15.4)
3900       350 FORMAT(X,4I2)
4000       400 FORMAT(X,4E15.4)
4100    C
4200           RETURN
4300           END
```

```
  100    C   SUBROUTINE SOLVE
  200    C
  300    C                  THIS SUBROUTINE TRANSFERS MATRICES AND
  400    C                VECTORS TO TEMPORARY ARRAYS, WHICH ARE THEN
  500    C                USED AS CALLING ARGUMENTS TO A LIBRARY ROUTINE
  600    C                FOR SOLVING LINEAR SYSTEMS OF EQUATIONS.  LEQT1F
  700    C                IS IN THE IMSL LIBRARY AT MIT'S INFORMATION
  800    C                PROCESSING CENTER.  SIMQ IS A MODIFIED VERSION OF
  900    C                A SUBROUTINE IN THE SSP LIBRARY, AND IS ONE OF
 1000    C                THE ROUTINES IN FRACLIB.
 1100    C
 1200           SUBROUTINE SOLVE(A,SIGMA,COEFF,NSIZE)
 1300           IMPLICIT REAL*8(A-H,O-Z)
 1400           INTEGER ORDER
 1500           DIMENSION B(80),Y1(80),WKAREA(80)
 1600           DIMENSION A(80,80),SIGMA(80),COEFF(80)
 1700           DIMENSION ATEMP(80,80)
 1800           DO 10 I=1,NSIZE
 1900           B(I)=0.
 2000           Y1(I)=SIGMA(I)
 2100           DO 10 J=1,NSIZE
 2200           ATEMP(I,J)=A(I,J)
 2300        10 CONTINUE
 2400           M=1
 2500           IA=80
 2600           IDGT=4
 2700    C
 2800    C      CALL LEQT1F(A,M,NSIZE,IA,Y1,IDGT,WKAREA,IER)
 2900           CALL SIMQ(ATEMP,Y1,NSIZE,KS,IA)
 3000    C
 3100           DO 20 I=1,NSIZE
 3200           COEFF(I)=Y1(I)
 3300        20 CONTINUE
 3400    C      WRITE(6,100) IER
```

```
3500          WRITE(6,100) KS
3600      100 FORMAT(' ',T10,'IER=',I5)
3700          RETURN
3800          END
```

```
C  SUBROUTINE STATFL
C
C                THIS SUBROUTINE PERFORMS QUASI-STATIC
C             PRESSURE EVOLUTION COMPUTATIONS FOR STATIONARY
C             CRACKS USING IMPLICIT TIME INTEGRATION.
C
C
      SUBROUTINE STATFL(MU,BB)
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 M(80,80),M1(80,80),M2(80,80),M3(80,80),M4(80,80)
      REAL*8 M5(80,80),MU(80,3),KAPPA1
      REAL*4 XY(2,80),XSCL(4),DUMMY
      INTEGER R,S,ORDER,ELMNTT,ELMNTX,Q,TSTEP
C
      DIMENSION A(80,80),APRIME(80,80),B(80,80),TEMP2(80,80),DDELDT(80)
      DIMENSION C(80,80),CPRIME(80,80),D(80,80),E(80,80)
      DIMENSION F(80,80),G(80,80),H(80,80),SA(80,80),T(80,80),DELTA0(80)
      DIMENSION DELTA1(80),P0(80),P1(80),TEMP(80,80),TEMP1(80,80)
      DIMENSION BB(80,80),YNODE(80,3),RR(80),YT(80),DDEL0(80),DDEL1(80)
      DIMENSION DEL1(80,80),DEL0(80,80)
C
      COMMON /OUT/ STRSL(80,3),STRSC(80,3),ALOC(80,3),ACART(80,3)
      COMMON /BKPING/ XNODE(80,3),TNODE(80,3),ELMNTT(4,80),ELMNTX(4,80)
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /TIME/ TSTART,FTIME,DT,TSTEP,TIME
C
      TCH(N,X)=DCOS(DFLOAT(N)*DACOS(X))
      TCHPR(N,X)=N*DSIN(DFLOAT(N)*DACOS(X))/DSQRT(1.-X**2)
      SIGN(X)=X/DABS(X)
C
      PI=3.1415926535898
      PBHL=1.0
      IF(TSTEP.LT.0) GO TO 9
```

```
      READ(5,1) ALPHA,BETA
      READ(5,3) KILL,ICUT,LOOP
    1 FORMAT(2F10.4)
    3 FORMAT(3I3)
      NLNODE=NTNODE+1
      NMNODE=NLNODE
C
      NL2=NLNODE/2
      ITEST=2.*NL2
      IF(ITEST.LT.NLNODE) NL2=NL2+1
C
      NM2=NMNODE/2
      ITEST=2*NM2
      IF(ITEST.LT.NMNODE) NM2=NM2+1
C
      CALL CONST(IREG,JREG,EMOD,AA,BB)
      EMOD=2.*EMOD
C
      DO 5 I=1,NLNODE
      ARG=PI*(2.*I-1.)/(2.*NLNODE)
      YNODE(1,1)=-DCOS(ARG)
    5 CONTINUE
C
C
    9 CONTINUE
      IF(KILL.NE.1) GO TO 248
CSHIFT
      DO 246 I=1,NXNODE
      P1(I)=STRSL(ICUT+I,1)
      YT(I)=XNODE(ICUT+I,1)
  246 CONTINUE
CTRANSFER
      NT=NXNODE-2*ICUT
      CALL TRANS3(YT,P1,NT,YNODE,P0,NLNODE,80,80,1)
      GO TO 249
```

```
C
  248 CONTINUE
C
      CALL TRANS3(XNODE,STRSL,NXNODE,YNODE,P0,NLNODE,80,80,1)
C
  249 CONTINUE
      IF(KILL.NE.1) GO TO 6
      DO 8 I=1,ICUT
      P0(I)=0.
      P0(NLNODE-ICUT+I)=0.
    8 CONTINUE
C
C
    6 CONTINUE
      CALL INTEG(MU,DELTA0,TNODE,NTNODE,YNODE,NLNODE,0,80)
      SUM=0.
      DO 7 I=1,NLNODE
      DDELDT(I)=DELTA0(I)-DELTA1(I)
      DELTA1(I)=(1.+BETA)*DELTA0(I)
      SUM=SUM+DELTA0(I)*DSQRT(1.-YNODE(I,1)**2)
    7 CONTINUE
      SUM=PI*SUM/NLNODE
      WRITE(6,471) SUM
  471 FORMAT('********** FLUID VOLUME =',E15.4,' **********')
      IF(TSTEP.LT.0) GO TO 100
C
C
      DO 10 R=1,NXNODE
      DO 10 L=1,NLNODE
      A(R,L)=TCH(L-1,XNODE(R,1))
      IF(L.EQ.1) A(R,L)=.5*A(R,L)
   10 CONTINUE
C
C
      DO 15 L=1,NLNODE
```

```
      DO 15 S=1,NLNODE
      APRIME(L,S)=TCH(L-1,YNODE(S,1))*2./(NLNODE)
   15 CONTINUE
C
C
      NM2PL=NM2+1
      NM2MI=NM2-1
      DO 25 R=1,NXNODE
      DO 25 J=1,NMNODE
      SUM=0.
      SUM1=0.
      DO 20 I=1,NTNODE
C
      SUM=SUM+TCHPR(J,TNODE(I,1))*DSQRT(1.-TNODE(I,1)**2)
     1    /((XNODE(R,1)-TNODE(I,1))**2)
C
      SUM1=SUM1+TCHPR(J,XNODE(R,1))* SQRT(1.-TNODE(I,1)**2)
     1     /((XNODE(R,1)-TNODE(I,1))**2)
C
   20 CONTINUE
      B(R,J)=-PI*EMOD*(SUM-SUM1)/NTNODE
   25 CONTINUE
      DO 26 J=1,NMNODE
      B(NTNODE,J)=TCHPR(J,YNODE(NM2PL,1))-TCHPR(J,YNODE(NM2MI,1))
      B(NTNODE+1,J)=0.
   26 CONTINUE
C
C
      DO 40 R=1,NXNODE
      DO 40 J=1,NMNODE
      C(R,J)=TCHPR(J,XNODE(R,1))
   40 CONTINUE
C
C
      DO 45 K=1,NMNODE
```

```
      DO 45 J=1,NMNODE
C
      CPRIME(K,J)=TCHPR(J,YNODE(K,1))
C
      D(J,K)=TCH(J,YNODE(K,1))*2./NLNODE
C
   45 CONTINUE
C
C
      DO 50 K=1,NMNODE
      DO 50 L=1,NLNODE
      E(K,L)=TCHPR(L,YNODE(K,1))
   50 CONTINUE
C
C
      DO 55 L=1,NLNODE
      DO 55 S=1,NLNODE
      F(L,S)=TCH(L,YNODE(S,1))*2./NLNODE
   55 CONTINUE
C
C
      DO 65 R=1,NXNODE
      DO 60 J=1,NXNODE
      G(R,J)=0.
   60 CONTINUE
      G(R,R)=EMOD*((1./(XNODE(R,1)-1.))
     1          -(1./(XNODE(R,1)+1.)))
      G(NTNODE,R)=0.
      G(NTNODE+1,R)=0.
   65 CONTINUE
C
C
      DO 75 S=1,NLNODE
      DO 70 Q=1,NLNODE
      H(S,Q)=0.
```

```
      SA(S,Q)=0.
      T(S,Q)=0.
   70 CONTINUE
C
      H(S,S)=-SIGN(YNODE(S,1))
      H(S,NL2)=H(S,NL2)+SIGN(YNODE(S,1))
C
      SA(S,S)=1.
      SA(S,NM2)=SA(S,NM2)-SIGN(YNODE(S,1))
C
      T(S,S)=-SIGN(YNODE(S,1))
C
   75 CONTINUE
C      WRITE(1,500) ((H(II,JJ),JJ=1,NLNODE),II=1,NLNODE)
C      WRITE(2,500) ((SA(II,JJ),JJ=1,NLNODE),II=1,NLNODE)
C      WRITE(3,500) ((T(II,JJ),JJ=1,NLNODE),II=1,NLNODE)
  500 FORMAT(8E15.4)
C
C
      CALL MULT(A,NXNODE,NLNODE,APRIME,NLNODE,NLNODE,M1,IER)
      DO 76 J=1,NLNODE
      M1(NTNODE,J)=0.
      M1(NTNODE+1,J)=0.
   76 CONTINUE
C
C
      CALL MULT(B,NMNODE,NMNODE,D,NMNODE,NMNODE,TEMP,IER)
      CALL MULT(TEMP,NMNODE,NMNODE,SA,NMNODE,NMNODE,M2,IER)
C
C
      CALL MULT(T,NMNODE,NMNODE,E,NMNODE,NLNODE,TEMP,IER)
      CALL MULT(TEMP,NMNODE,NLNODE,F,NLNODE,NLNODE,TEMP1,IER)
      CALL MULT(TEMP1,NMNODE,NLNODE,H,NLNODE,NLNODE,M3,IER)
C
C
```

```
      CALL MULT(G,NMNODE,NXNODE,C,NXNODE,NMNODE,TEMP,IER)
      CALL MULT(TEMP,NMNODE,NMNODE,D,NMNODE,NMNODE,TEMP1,IER)
      CALL MULT(TEMP1,NMNODE,NMNODE,SA,NMNODE,NMNODE,M4)
C
C
      CALL MULT(CPRIME,NMNODE,NMNODE,D,NMNODE,NMNODE,TEMP,IER)
      CALL MULT(TEMP,NMNODE,NMNODE,SA,NMNODE,NMNODE,M5,IER)
C
C
CTEST TEST TEST TEST TEST TEST
C      WRITE(16,500) ((C(II,JJ),JJ=1,NMNODE),II=1,NXNODE)
C      WRITE(17,500) ((CPRIME(II,JJ),JJ=1,NMNODE),II=1,NMNODE)
C      WRITE(18,500) ((D(II,JJ),JJ=1,NMNODE),II=1,NMNODE)
C      WRITE(19,500) ((M3(II,JJ),JJ=1,NMNODE),II=1,NMNODE)
C      WRITE(20,500) ((M5(II,JJ),JJ=1,NMNODE),II=1,NMNODE)
C      WRITE(21,500) ((M2(II,JJ),JJ=1,NMNODE),II=1,NXNODE)
C      WRITE(22,500) ((M4(II,JJ),JJ=1,NMNODE),II=1,NMNODE)
CTEST TEST TEST TEST TEST TEST
C
C=======================================================================
C=======================================================================
C
  100 CONTINUE
C
C
      DO 101 I=1,NLNODE
      XY(1,I)=YNODE(I,1)
      XY(2,I)=P0(I)
  101 CONTINUE
      CALL QPICTR(XY,2,NLNODE,QX(1))
      ISCL=-2
      XSCL(1)=-1.0
      XSCL(2)=1.0
      XSCL(3)=-.10
      XSCL(4)=1.2
```

```
      CALL QPICTR(XY,2,NLNODE,QX(1),QISCL(ISCL),QXSCL(XSCL))
C
      XSCL(4)=1.5
      DO 102 I=1,NLNODE
      XY(1,I)=YNODE(I,1)
      XY(2,I)=DELTA1(I)
  102 CONTINUE
CX      CALL QPICTR(XY,2,NLNODE,QINIT(DUMMY),QX(1))
      CALL QPICTR(XY,2,NLNODE,QX(1),QISCL(ISCL),QXSCL(XSCL))
C
      DO 103 I=1,NLNODE
      XY(1,I)=YNODE(I,1)
      XY(2,I)=DDELDT(I)
  103 CONTINUE
      CALL QPICTR(XY,2,NLNODE,QINIT(DUMMY),QX(1))
C
      IF(TIME.GT.FTIME) GO TO 1000
C
C
      DO 110 J=1,NMNODE
      DO 105 K=1,NMNODE
      DEL0(J,K)=0.
      DEL1(J,K)=0.
  105 CONTINUE
      DEL0(J,J)=DELTA0(J)**3
      DEL1(J,J)=DELTA1(J)**3
  110 CONTINUE
C
C
C                    COMPUTE M:
C
C
      FACTOR=-ALPHA*DT
C
C
```

```
      CALL MULT(FACTOR,1,1,M2,NMNODE,NMNODE,TEMP1,IER)
      CALL MULT(TEMP1,NMNODE,NMNODE,DEL1,NMNODE,NMNODE,TEMP,IER)
      CALL MULT(TEMP,NMNODE,NMNODE,M3,NMNODE,NLNODE,TEMP1,IER)
C
C
      CALL ADD(M1,NMNODE,NLNODE,TEMP1,NMNODE,NLNODE,TEMP2,IER)
C
C
      FACTOR=-FACTOR
      CALL MULT(FACTOR,1,1,M4,NMNODE,NMNODE,TEMP,IER)
      CALL MULT(TEMP,NMNODE,NMNODE,DEL1,NMNODE,NMNODE,TEMP1,IER)
      CALL MULT(TEMP1,NMNODE,NMNODE,M3,NMNODE,NMNODE,TEMP,IER)
C
C
      CALL ADD(TEMP2,NMNODE,NMNODE,TEMP,NMNODE,NMNODE,M,IER)
C
      M(NLNODE,NL2)=1.
C
C                     COMPUTE RR:
C
C
      FACTOR=(1.-ALPHA)*DT
C
      CALL MULT(FACTOR,1,1,M2,NMNODE,NMNODE,TEMP,IER)
      CALL MULT(TEMP,NMNODE,NMNODE,DEL0,NMNODE,NMNODE,TEMP1,IER)
      CALL MULT(TEMP1,NMNODE,NMNODE,M3,NMNODE,NMNODE,TEMP2,IER)
C
C
      CALL ADD(M1,NMNODE,NMNODE,TEMP2,NMNODE,NMNODE,TEMP,IER)
C
C
      FACTOR=-FACTOR
      CALL MULT(FACTOR,1,1,M4,NMNODE,NMNODE,TEMP1,IER)
      CALL MULT(TEMP1,NMNODE,NMNODE,DEL0,NMNODE,NMNODE,TEMP2,IER)
      CALL MULT(TEMP2,NMNODE,NMNODE,M3,NMNODE,NMNODE,TEMP1,IER)
```

```
C
C
      CALL ADD(TEMP,NMNODE,NMNODE,TEMP1,NMNODE,NMNODE,TEMP2,IER)
      CALL MULT(TEMP2,NMNODE,NMNODE,P0,NMNODE,1,RR,IER)
      RR(NTNODE)=0.
      RR(NTNODE+1)=PBHL
C
C
C
      CALL SOLVE(M,RR,P1,NLNODE)
C
C
      IF(LOOP.NE.1) GO TO 235
      TSTEP=-100
      IF(KILL.NE.1) GO TO 232
CSHIFT
      DO 239 I=1,NLNODE
      P0(I)=P1(ICUT+I)
      YT(I)=YNODE(ICUT+I,1)
  239 CONTINUE
CTRANSFER
      NT=NLNODE-2*ICUT
      CALL TRANS3(YT,P0,NT,XNODE,STRSL,NXNODE,80,80,1)
CCUT
      DO 241 I=1,ICUT
      STRSL(I,3)=0.
      STRSL(NXNODE-ICUT+I,1)=0.
  241 CONTINUE
C
  232 CONTINUE
      GO TO 1000
C
  235 CONTINUE
      CALL MULT(M5,NMNODE,NMNODE,DEL1,NMNODE,NMNODE,TEMP,IER)
      CALL MULT(TEMP,NMNODE,NMNODE,M3,NMNODE,NMNODE,TEMP1,IER)
```

```
      CALL MULT(TEMP1,NMNODE,NMNODE,P1,NMNODE,1,DDEL1,IER)
C
      CALL MULT(M5,NMNODE,NMNODE,DEL0,NMNODE,NMNODE,TEMP,IER)
      CALL MULT(TEMP,NMNODE,NMNODE,M3,NMNODE,NMNODE,TEMP1,IER)
      CALL MULT(TEMP1,NMNODE,NMNODE,P0,NMNODE,1,DDEL0,IER)
C
      IF(KILL.NE.1) GO TO 148
      DO 146 I=1,ICUT
      P1(I)=0.
      P1(NLNODE-ICUT+I)=0.
  146 CONTINUE
C
C
  148 CONTINUE
      SUM=0.
      DO 150 I=1,NLNODE
      DDELDT(I)=ALPHA*DDEL1(I)+(1.-ALPHA)*DDEL0(I)
      DELTA0(I)=DELTA0(I)+DDELDT(I)*DT
      SUM=SUM+DELTA0(I)*DSQRT(1.-YNODE(I,1)**2)
      P0(I)=P1(I)
      DELTA1(I)=(1.+BETA)*DELTA0(I)
      WRITE(6,466) P1(I)
  466 FORMAT(E15.4)
  150 CONTINUE
      SUM=PI*SUM/NLNODE
      WRITE(6,471) SUM
      WRITE(6,467)
  467 FORMAT(///)
C
C
      TIME=TIME+DT
      GO TO 100
C
C
 1000 CONTINUE
      RETURN
      END
```

```
C  SUBROUTINE STRCMP
C
C                THIS SUBROUTINE ASSEMBLES THE TRACTION
C              COMPONENTS CONTAINED IN THE ARRAY STRSL
C              INTO THE "RIGHT- HAND SIDE" VECTOR OF THE
C              SYSTEM OF EQUATIONS.
C
      SUBROUTINE STRCMP
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER ORDER
      DIMENSION RLOC(5,3),RPOS(3),AA(2),B(2),NPTS(4)
      DIMENSION XNODE(80,3),TNODE(80,3),STRSL(80,3),STRSC(80,3)
      DIMENSION A(80,80)
      DIMENSION COEFF(80),SIGMA(80),ACART(80,3)
      DIMENSION ALOC(80,3),ELCON(2,3)
      DIMENSION ZETA(2,80),ETA(2,80)
C
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /ARRAYS/ A,SIGMA,COEFF
      COMMON /OUT/ STRSL,STRSC,ALOC,ACART
      COMMON /GP/ ZETA,NPTS
      COMMON /DOF/ IDOF
C
C
      II=0
      IMIN=1
      IMAX=0
      DO 185 J=1,NELMNT
      IMAX=IMAX+NPTS(J)-1
      DO 180 BETA=1,IDOF
      DO 170 I=IMIN,IMAX
      II=II+1
      SIGMA(II)=STRSL(I,BETA)
  170 CONTINUE
```

```
3500            II=II+1
3600        180 CONTINUE
3700            IMIN=IMAX+1
3800        185 CONTINUE
3900            RETURN
4000            END
```

```
C SUBROUTINE STRESS
C
C                THIS SUBROUTINE AUTOMATICALLY
C              GENERATES INITIAL TRACTIONS WITH
C              THE DESIRED FUNCTIONAL FORM.
C
      SUBROUTINE STRESS(STRSL,LOAD,I,J)
C
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER ELMNTT(4,80),ELMNTX(4,80),RFRONT
      REAL*8 LOAD(2),NONGAM,NONDEL,NONP,NONMU
C
      DIMENSION XNODE(80,3),TNODE(80,3),STRSL(80,3),STRSX(80,3)
C
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
      COMMON /LPAR/ LTYPE
      COMMON /NONDIM/ NONGAM,NONDEL,NONP,NONMU,TAUC
      COMMON /FILL/ LFRONT,RFRONT
C
      EXP(Q)=DEXP(Q)
      SQRT(Q)=DSQRT(Q)
      ABS(Q)=DABS(Q)
C
      A=ABS(TNODE(1,1))
      IF((LTYPE.EQ.1).OR.(LTYPE.EQ.5)) B=LOAD(2)
      CONST=LOAD(1)*EXP(-B*0.25)
C
      IF(LTYPE.EQ.1) STRSL(J,1)=LOAD(1)*EXP(-B*XNODE(J,1)**2)
C
      IF(LTYPE.EQ.6) STRSL(J,1)=LOAD(1)*(1.-(XNODE(J,1)/A)**2)
      IF(LTYPE.EQ.2) STRSL(J,1)=LOAD(1)*SQRT(1.-ABS(XNODE(J,1)))
C
      IF(LTYPE.EQ.3) STRSL(J,1)=LOAD(1)
C
```

```
3500          IF((LTYPE.EQ.4).AND.(XNODE(J,1).LT.0.)) STRSL(J,1)=LOAD(1)*
3600         1(1.+XNODE(J,1))
3700      C
3800          IF((LTYPE.EQ.7).AND.(XNODE(J,1).LT.0.)) STRSL(J,1)=LOAD(1)*
3900         1(XNODE(RFRONT,1)+XNODE(J,1))/XNODE(RFRONT,1)
4000      C
4100          IF((LTYPE.EQ.4).AND.(XNODE(J,1).GE.0.)) STRSL(J,1)=LOAD(1)*
4200         1(1.-XNODE(J,1))
4300      C
4400          IF((LTYPE.EQ.7).AND.(XNODE(J,1).GE.0.)) STRSL(J,1)=LOAD(1)*
4500         1(XNODE(RFRONT,1)-XNODE(J,1))/XNODE(RFRONT,1)
4600      C
4700          IF((LTYPE.EQ.7).AND.(J.LT.LFRONT)) STRSL(J,1)=0.
4800          IF((LTYPE.EQ.7).AND.(J.GT.RFRONT)) STRSL(J,1)=0.
4900      C
5000          IF((LTYPE.EQ.5).AND.(ABS(XNODE(J,1)).GT.0.5)) STRSL(J,1)=0.
5100      C
5200          IF((LTYPE.EQ.5).AND.(ABS(XNODE(J,1)).LE.0.5)) STRSL(J,1)=
5300         1LOAD(1)*EXP(-B*XNODE(J,1)**2)-CONST
5400      C
5500          STRSL(J,1)=STRSL(J,1)*NONP
5600          STRSL(J,2)=LOAD(2)*NONP
5700          IF((LTYPE.EQ.1).OR.(LTYPE.EQ.5)) STRSL(J,2)=0.
5800      C
5900          RETURN
6000          END
```

```
100    C  SUBROUTINE TRANS3
200    C
300    C                  THIS SUBROUTINE INTERPOLATES A FUNCTION
400    C              KNOWN FOR THE ARGUMENTS XTEMP AT THE NEW SET
500    C              OF ARGUMENTS XOLD (THESE NAMES ARE NO LONGER
600    C              MNEMONIC, BUT WHEN THE SUBROUTINE WAS WRITTEN
700    C              THEY WERE RELEVANT).  TRANS3 USES LGRNG TO
800    C              PERFORM THE ACTUAL INTERPOLATION, BUT DOES SO
900    C              IN SUCH A WAY THAT THE FUNCTION IS INTERPOLATED
1000   C              SEPARATELY ON EITHER SIDE OF THE ORIGIN, TO
1100   C              PRESERVE DISCONTINUITIES.
1200   C
1300         SUBROUTINE TRANS3(XTEMP,YTEMP,NXTEMP,XOLD,YOLD,NXOLD,
1400        1IROW,JROW,JCOL)
1500   C
1600         IMPLICIT REAL*8(A-H,O-Z)
1700         DIMENSION XSIDE(400,3),YSIDE(400,3)
1800         DIMENSION XOLD(JROW,3),YOLD(JROW,1),XTEMP(IROW,3),YTEMP(IROW,5)
1900   C
2000         NXT=NXTEMP/2
2100         WRITE(25,100) NXT
2200         NSIDE=NXOLD/2
2300         ITEST=2*NXT
2400         IF(ITEST.LT.NXTEMP) NXT=NXT+1
2500         WRITE(25,100) NXT
2600     100 FORMAT(' ',I3)
2700   C
2800         DO 5 I=1,NXT
2900         XSIDE(I,1)=XTEMP(I,JCOL)
3000         YSIDE(I,1)=YTEMP(I,JCOL)
3100       5 CONTINUE
3200   C
3300         K=1
3400         DO 10 I=1,NSIDE
```

```
3500          CALL LGRNG(XSIDE,YSIDE,XOLD(I,1),PT,NXT,5,IROW)
3600          YOLD(K,1)=PT
3700          K=K+1
3800       10 CONTINUE
3900    C
4000          IF(ITEST.LT.NXTEMP) NXT=NXT-1
4100          N=NXT+1
4200          DO 15 I=1,NXT
4300          XSIDE(I,1)=XTEMP(N,JCOL)
4400          YSIDE(I,1)=YTEMP(N,JCOL)
4500          N=N+1
4600       15 CONTINUE
4700    C
4800          NSIDE=NSIDE+1
4900          DO 20 I=NSIDE,NXOLD
5000          CALL LGRNG(XSIDE,YSIDE,XOLD(I,1),PT,NXT,5,IROW)
5100          YOLD(K,1)=PT
5200          K=K+1
5300       20 CONTINUE
5400    C
5500          RETURN
5600          END
```

```
C  SUBROUTINE TRNSFM
C
C                THIS SUBROUTINE USES THE VECTOR
C           TRANSFORMATION LAWS TO TRANSFORM THE
C           VALUES OF THE SOLUTION "F" FROM GLOBAL
C           INTO LOCAL COORDINATES.
C
      SUBROUTINE TRNSFM
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER ELMNTT(4,80),ELMNTX(4,80)
      INTEGER ORDER
      DIMENSION XNODE(80,3),TNODE(80,3),STRSL(80,3),STRSC(80,3)
      DIMENSION COEFF(80),SIGMA(80),ACART(80,3)
      DIMENSION ALOC(80,3),ELCON(2,3)
      DIMENSION XA1(4),XB1(4),XA2(4),XB2(4)
C
      COMMON /ENDPTS/ XA1,XB1,XA2,XB2,THETA
      COMMON /SIZE/ ORDER,NELMNT,NXNODE,NTNODE
      COMMON /OUT/ STRSL,STRSC,ALOC,ACART
      COMMON /BKPING/ XNODE,TNODE,ELMNTT,ELMNTX
      ATAN(Q)=DATAN(Q)
      SIN(Q)=DSIN(Q)
      COS(Q)=DCOS(Q)
      JMAX=0
      JMIN=1
      DO 380 I=1,NELMNT
      GAMMA=3.1415926535898/2.
      ARG=(XB2(I)-XA2(I))/(XB1(I)-XA1(I))
      IF(XB1(I).NE.XA1(I)) GAMMA=ATAN(ARG)
      IF(XB1(I).LT.XA1(I)) GAMMA=3.1415926535898+GAMMA
      GAMMA=-GAMMA
      JMAX=JMAX+ELMNTT(I,1)
      DO 370 J=JMIN,JMAX
C
```

```
3500          ALOC(J,1)=ACART(J,1)*COS(GAMMA)+ACART(J,2)*SIN(GAMMA)
3600          ALOC(J,2)=-ACART(J,1)*SIN(GAMMA)+ACART(J,2)*COS(GAMMA)
3700      370 CONTINUE
3800          JMIN=JMAX+1
3900      380 CONTINUE
4000          RETURN
4100          END
```