Massachusetts Institute of Technology

Department of Nuclear Engineering

TRANSIENT RESPONSE OF A SINGLE HEATED CHANNEL

by

Min Lee and M. S. Kazimi

MITNE-271

Department of Nuclear Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

July 25, 1984

TRANSIENT RESPONSE OF A SINGLE HEATED CHANNEL

by

Min Lee and M. S. Kazimi

ABSTRACT

The adequacy of four approaches to description of the transients in a heated channel are investigated. The four approaches are: the sectionalized compressible flow model, the momentum integral model, the single velocity model and the channel integral model. The transients investigated represent flow reduction and power increase under conditions representative of PWR and BWR pressure and flow rate conditions.

While the first approach is the most rigorous one, it requires much longer computational times. The constraints implied by the other models, and hence the class of transients they should not be used for are outlined.

Nomenclature

A	flow area
c	velocity of sound
De	equivalent hydraulic diameter
f	friction factor
g	gravitational acceleration
G	mass flux
H	enthalpy
q'	axial power input per unit length
p	pressure
t	time
u	fluid velocity
z	axial distance
$\phi^2_{\ell o}$	two phase friction multiplication factor
ρ	density

Subscripts

j	cell designation for finite difference in mean value within the area

Superscript

i	time step designation

Table of Contents

Page

Table of Contents (continued)

List of Figures

List of Tables

1. INTRODUCTION

By neglecting the lateral variation of fluid properties and velocity, the conservation equations of mass, momentum and energy for a single heated channel can be written in the following form:

$$\frac{\partial \rho_m}{\partial t} + \frac{\partial G_m}{\partial z} = 0 \tag{1}$$

$$\frac{\partial G_m}{\partial t} + \frac{\partial}{\partial z} (G_m^2/\rho_m) = - \frac{\partial P}{\partial z} - \frac{\phi_{\ell o}^2 (f/\rho)|G_m|G_m}{2De} - \rho_m g \tag{2}$$

$$\rho_m \frac{\partial H_m}{\partial t} + G_m \frac{\partial H_m}{\partial z} = \frac{\partial P}{\partial t} + \frac{q'}{A} + \frac{G_m}{\rho_m} [\frac{\partial P}{\partial z} + \frac{\phi_{\ell o}^2 (f/\rho)/|G_m|G_m}{2De}] \tag{3}$$

Under the assumption that the liquid and vapor can be considered as a homogeneous mixture, the above equations are applicable for two phase flow as well as for single phase flow.

In order to understand the transient response of a single heated channel, the solution for $G_m(z,a)$, $P(z,t)$, and $H_m(z,t)$ is desired for the above equations under appropriate boundary conditions. The difficulty in solving the general transient equations arises from the coupling between the solution of the momentum and the energy equations. However, several different levels of approximations, or models [1], can be used to decouple the momentum and energy equations. Those models are:

(1) Sectionalized Compressible Model

This model is the most direct and detailed representation of channel behavior. It is based on a direct numerical solution of multiple point (or sectionalized) difference equation approximations to the conservation equations (Eqs. (1 to 3)).

(2) Momentum Integral Model

It is assumed that the density can be evaluated as a function of enthalpy and a reference pressure where the latter is considered as a constant. This is equivalent to assuming that the fluid is incompressible. It is clear that, under this assumption, the local pressure gradient will not influence the mass flux of the fluid along the channel. Thus, the momentum equation is only useful in determining the spatially averaged mass velocity. Combining Eqs. (1) and (3) and with the help of the equation of state, we can use a difference approximation to find the variation of the local mass velocity about the average mass velocity. In this model, the sonic effects are neglected. The effects of thermal expansion and enthalpy transport are preserved.

(3) Single Mass Velocity Model

Further computational simplification can be obtained if the effect of thermal expansion is neglected in handling the mass velocity profile. That is the mass velocity is considered to be constant throughout the channel and is a function of time only. This model preserves the effect of enthalpy transport.

(4) Channel Integral Model

Channel Integral model solves all three conservation equations in an integrated manner. In order to perform the integration of mass and energy equation, an axial profile of the enthalpy is required. This is assumed to be the same as that in steady state condition. Hence, in this model, the effect of enthalpy transport is neglect. The effect of thermal expansion is preserved.

In solving these equations, we need to specify the boundary conditions. These are:

(1) The inlet and outlet pressures or the inlet velocity and the pressure value at one of the ends.

(2) The linear heat generation is given and is uniformly distributed.

(3) The inlet enthalpy is specified.

In this study, a computer code based on finite difference forms was written for each model. The following sections summarize the finite difference equations and numerical schemes. Examples for the results using each of the models are also presented.

2. SECTIONALIZED COMPRESSIBLE FLOW MODEL

2.1 Differential Equation

Assume that a differential equation of state is available:

$$\rho_m = \rho_m(H_m, P)$$

then

$$\frac{\partial \rho_m}{\partial t} = \left(\frac{\partial \rho_m}{\partial H_m}\right)_p \frac{\partial H_m}{\partial t} + \left(\frac{\partial \rho_m}{\partial P}\right)_{H_m} \frac{\partial P}{\partial t}$$

$$= R_h \frac{\partial H_m}{\partial t} + R_p \frac{\partial P}{\partial t} \qquad (5)$$

From the mass conservation equation (Eq. (1))

$$R_h \frac{\partial H_m}{\partial t} + R_p \frac{\partial P}{\partial t} + \frac{\partial G_m}{\partial z} = 0 \qquad (6)$$

Then Eqs. (3) and (6) may be combined to yield one equation only with $\partial p/\partial t$ and a second only with $\partial H_m/\partial t$ as a time derivative.

$$\frac{\rho_m}{c^2}\frac{\partial P}{\partial t} + \frac{\rho_m\,\partial G_m}{\partial z} + \left(\frac{R_h G_m}{\rho_m}\right)\frac{\partial P}{\partial z} - (R_h G_m)\frac{\partial H_m}{\partial z}$$

$$= - R_h \left[\frac{q'}{A} + \frac{\phi^2_{\ell o}(f/\rho)|G_m|G^2_m}{2\rho_m De}\right] \tag{7}$$

$$\frac{\rho_m}{c^2}\frac{\partial H_m}{\partial t} + \frac{\partial G_m}{\partial z} - \frac{R_p G_m}{\rho_m}\frac{\partial P}{\partial z} + (R_p G_m)\frac{\partial H_m}{\partial z}$$

$$= R_p \left[\frac{q'}{A} + \frac{\phi^2_{\ell o}(f/\rho)|G_m|G^2_m}{2\rho_m De}\right] \tag{8}$$

where c, the isentropic sonic velocity, is given

$$c = 1/(R_p + R_h/\rho_m)^{0.5} \tag{9}$$

The equations we are interested in are Eqs. (2), (7) and (8) which are in terms of p, H_m, G_m. By the relation $U_m = G_m/\rho_m$, we can change all three equations into the following form:

$$\rho_m\left(\frac{\partial U_m}{\partial t} + U_m\frac{\partial U_m}{\partial z}\right) = -\frac{\partial P}{\partial z} - \frac{\phi^2_{\ell o}(f/\rho)\rho^2_m U^2_m}{2De} - \rho_m g \tag{10}$$

$$\frac{\rho_m}{c^2}\left(\frac{\partial P}{\partial t} + U_m\frac{\partial P}{\partial z}\right) = -\rho^2_m\frac{\partial U_m}{\partial z} - R_h\left(\frac{q'}{A} + \frac{\phi^2_{\ell o}(f/\rho)\rho^2_m U^3_m}{2De}\right) \tag{11}$$

$$\frac{\rho_m}{c^2}\left(\frac{\partial H_m}{\partial t} + U_m\frac{\partial H_m}{\partial z}\right) = -\rho_m\frac{\partial U_m}{\partial z} + R_p\left(\frac{q'}{A} + \frac{\phi^2_{\ell o}(f/\rho)\rho^2_m U^3_m}{2De}\right) \tag{12}$$

2.2 Finite Difference Equation

Using staggered mesh (Fig. 1) and an explicit method [2], we have:

$$(\rho_m)_{j+1/2} \left[\frac{(U_m)^{i+1}_{j+1/2} - (U_m)^{i}_{j+1/2}}{\Delta t}\right] + \left[\frac{(U_m)^{i}_{j+1/2}[(U_m)^{i}_{j+1/2} - (U_m)^{i}_{j-1/2}]}{\Delta z}\right]$$

$$= \frac{P^i_{j+1} - P^i_j}{\Delta z} - \phi^2_{\ell o} \frac{(f/\rho)_{j+1/2}[(\rho_m)^i_{j+1/2} (U_m)^i_{j+1/2}]^2}{2De} - (\rho_m)^i_{j+1/2} g \quad (13)$$

$$\frac{(\rho_m)^i_{j-1/2}}{(c^2)^i_{j-1/2}} \left[\frac{P^{i+1}_j - P^i_j}{\Delta t} + (U_m)^{i+1}_{j-1/2} \frac{P^i_j - P^i_{j-1}}{\Delta z}\right]$$

$$= -(\rho^2_m)^i_{j-1/2} \frac{(U_m)^{i+1}_{j-1/2} - (U_m)^{i+1}_{j-1/2}}{\Delta z} - (R_h)^i_{j-1/2} \left[\frac{(q')^i_{j+1/2}}{A}\right.$$

$$\left. + \frac{[\phi^2_{\ell o}(f/\rho)]^{i+1}_{j-1/2} (\rho^2_m)^i_{j-1/2} (U^3_m)^{i+1}_{j-1/2}}{2De}\right] \quad (14)$$

$$\frac{(\rho_m)^i_{j-1/2}}{(c^2)^i_{j-1/2}} \left[\frac{(H_m)^{i+1}_j - (H_m)^i_j}{\Delta t} + (U_m)^{i+1}_{j-1/2} \frac{(H_m)^i_j - (H_m)^i_{j-1}}{\Delta z}\right]$$

$$= - (\rho_m)^i_{j-1/2} \frac{(U_m)^{i+1}_{j-1/2} - (U_m)^{i+1}_{j-1/2}}{\Delta z} + (R_p)^i_{j-1/2} \left[\frac{(q)^i_{j-1/2}}{A}\right.$$

$$\left. + \frac{[\phi^2_{\ell o}(f/\rho)]^{i+1}_{j-1/2} (\rho^2_m)^i_{j-1/2} (U^3_m)^{i+1}_{j-1/2}}{2De}\right] \quad (15)$$

In the above difference equation, the parameters $(R_h)_{j-1/2}$, $(R_p)_{j-1/2}$, $(c)_{j-1/2}$ are evaluated by the following equations:

$$(R_h)_{j-1/2} = - \frac{\rho_m((H_m)_j, P_{j-1/2}) - \rho_m((H_m)_{j-1}, P_{j-1/2})}{(H_m)_j - (H_m)_{j-1}}$$

$$(R_p)_{j-1/2} = \frac{\rho_m((H_m)_{j-1/2}, P_j) - \rho_m((H_m)_{j-1/2}, P_{j-1})}{P_j - P_{j-1}}$$

$$(c)_{j-1/2} = ((c)_j + (c)_{j-1})/2.$$

Where $(c)_j$ is evaluated by Eq. (9) with the values evaluated as the local derivative of density with respect to enthalpy and pressure.

Equation (13) can be solved explicitly. With the value of $(U_m)_j$ at the new time step, Equations (14) and (15) can also be solved explicitly. If the inlet velocity is specified in the calculation, the inlet pressure can be calculated by Eq. (14) at the first half cell, i.e., the equation is written in terms of $P_{1/2}$. For pressure boundary conditions the inlet velocity can be calculated by Eq. (13) written in terms of $(U_m)_{1/4}$. A half cell equation is also needed for calculating the outlet velocity.

2.3 Results

Several cases were analyzed (Table 1). Those are PWR heat flux increase transient (by 10%, step function), and PWR, BWR pressure drop decrease transient (by 1/2, exponential function with time constant equal to 0.025 sec^{-1}). The results are shown in Fig. 2 through Fig. 5. From Fig. 2 the effect of the sonic wave can easily by identified.

Table 1

Cases Analyzed

Operating Condition	PWR	BWR
Channel Length (m)	3.66	3.05
Rod Diameter (m)	9.70×10^{-3}	1.27×10^{-2} m
Pitch (m)	1.28×10^{-2}	1.595×10^{-2} m
q' Linear Heat (kW/m)	17.52	16.4
Mass Velocity (kg/m^2 sec)	4124.90	2302.4
P inlet (MPa)	15.5	6.9579
H inlet (kJ/kg)	1337.2	1225.5

Transients

(i) Pressure Drop Decrease Transient

$$P\ \text{inlet}\ (t) = (P\ \text{inlet}\ (0) + P\ \text{outlet}\ (0))/2.$$

$$+ \frac{P\ \text{inlet}\ (0) - P\ \text{outlet}\ (0)}{2} \times \exp\,(-400\ t)$$

(ii) Heat Flux Increase Transient

$$q'(t) = q'(0) \cdot 1.1.$$

The sonic velocity for the PWR operating condition is approximately 900 m/sec. The results for the BWR are slightly different, (Fig. 3), the sonic wave propagates faster in the single phase region and slows down considerably in the two-phase region. The sonic velocity in this region is approximately only 100 m/sec. The flow oscillation in the PWR heat flux transient (Fig. 4) is also induced by the sonic wave. The effect of thermal expansion on the distribution of mass flux can be seen in Fig. 5. It can be justified that the boiling begins around 1.1 sec. after the start of the transient.

Numerical stability of the difference solution requires that the time step size be less than the order of the time interval for sonic wave propagation between two points, i.e.,

$$\Delta t \leq \Delta z/(C+|U|) \tag{16}$$

Therefore, in a problem in which the fluid has a high-sonic velocity, the time step becomes prohibitively small from a computer time standpoint. Another difficulty for this model arises from the choice of the correct sonic velocity in finite difference equation. The sonic velocity decreases suddenly by more than one order of magnitude as the boiling begins. This imposes some extra problems on the stability of the numerical scheme.

3. Momentum Integral Model

3.1 Differential Equations [3]

It is assumed that

$$\rho_m = \rho_m(H_m, P^*) \tag{17}$$

where P* is a reference pressure. From the mass equation (1) we have

$$\frac{\partial G_m}{\partial z} = -\left(\frac{\partial \rho_m}{\partial H_m}\right)\left(\frac{\partial H_m}{\partial t}\right) \tag{18}$$

Integrate the momentum equation (2) over the channel and define G_{ave} as the average mass velocity we have:

$$\frac{\partial G_{ave}}{\partial t} = \frac{1}{L}\,(\Delta p - F) \tag{19}$$

where $\Delta P = -\int_o^L \frac{\partial P}{\partial z}\,dz = p_{inlet} - p_{outlet}$

$$G_{ave} = \frac{1}{L}\int_o^L G_m dz$$

$$F = \left(\frac{G_m^2}{\rho_m}\right)_{outlet} - \left(\frac{G_m^2}{\rho_m}\right)_{inlet} + \int_o^L \frac{\phi_{\ell o}^2 (f/\rho)|G_m|G_m}{2De}\,dz + \int_o^L \rho_m g dz$$

By neglecting the pressure and friction terms in the energy equation (3) we get:

$$\rho_m \frac{\partial H_m}{\partial t} + G_m \frac{\partial H_m}{\partial z} = \frac{q'}{A} \tag{20}$$

The equations we need to solve are Eqs. (18), (19), and (20).

3.2 Finite Difference Equations

Using a staggered mesh where $(G_m)_j^i$ is defined at the cell boundary, $(H_m)_{j-1/2}^i$ and $(\rho_m)_{j-1/2}^i$ are defined at the center of cell. Then, from Eq. (20) we have:

$$(H_m)_{j+1/2}^{i+1} = (H_m)_{j+1/2}^i + \frac{\Delta t}{(\rho_m)_{j-1/2}^i \Delta z}\left[(G_m)_{j+1}^i \frac{(H_m)_{j+3/2}^i - (H_m)_{j+1/2}^i}{2}\right.$$

$$\left. + (G_m)_j^i \frac{(H_m)_{j+1/2}^i - (H_m)_{j-1/2}^i}{2}\right] + (q')_{j+1/2}^i\, \Delta t/A\, (\rho_m)_{j-1/2}^i \tag{21}$$

Combining Eqs. (18) and (20) and rearranging, we have

$$\left(1 + \frac{1}{\rho_m}\frac{d\rho_m}{dH_m} H_m\right)\frac{\partial G_m}{\partial z} = -\frac{1}{\rho_m}\left(\frac{d\rho_m}{dH_m}\right)\left[\frac{q'}{A} - \frac{\partial G_m H_m}{\partial z}\right] \tag{22}$$

Convert the above equation into a finite difference form, we have

$$1 + \left(\frac{H_m}{\rho_m}\frac{d\rho_m}{dH_m}\right)_{j+1/2} \frac{(G_m)_{j+1} - (G_m)_j}{\Delta z} = - \left(\frac{1}{\rho_m}\frac{d\rho_m}{dH_m}\right)_{j+1/2}$$

$$\left[\frac{q'}{A} - \frac{(G_m)_{j+1}\frac{(H_m)_{j+1/2} + (H_m)_{j-1/2}}{2} - (G_m)_j\frac{(H_m)_{j+1/2} + (H_m)_{j-1/2}}{2}}{\Delta z}\right.$$

By rearranging the last equation we get:

$$(G_m)_{j+1} = (\alpha_j (G_m)_j + \beta_j) \tag{23}$$

where

$$\alpha_j = \frac{1 + \left(\frac{1}{\rho_m}\frac{d\rho_m}{dH_m}\right)_{j+1/2}\left[\frac{(H_m)_{j+1/2} - (H_m)_{j-1/2}}{2}\right]}{1 + \left(\frac{1}{\rho_m}\frac{d\rho_m}{dH_m}\right)_{j+1/2}\left[\frac{(H_m)_{j+1/2} - (H_m)_{j+3/2}}{2}\right]}$$

$$\beta_j = \frac{\left(\frac{1}{\rho_m}\frac{d\rho_m}{dH_n}\right)_{j+1/2} \frac{q'}{A} \cdot \Delta z}{1 + \left(\frac{1}{\rho_m}\frac{d\rho_m}{dH_m}\right)_{j+1/2}\left[\frac{(H_m)_{j+1/2} - (H_m)_{j+3/2}}{2}\right]}$$

All the above values are referred to new time step, i.e., i+1. Equation (19) can be changed into:

$$(G_{ave})^{i+1} = (G_{ave})^i + \frac{\Delta t}{L}(\Delta p^i - F^i) \tag{24}$$

From the initial conditions, the enthalpy distribution $(H_m)_{j+1}$ at a new time step can always be calculated with Eq. (21). For a flow transient, $(G_m)_1$ is specified, and the mass flux distribution can be known from Eq. (23). For a pressure transient, G_{ave} at new time step can be obtained from Eq. (24) and by following the relation we can find $(G_m)_1$.

$$(G_m)_1^{i+1} = \frac{1}{\hat{\gamma}^{i+1}} (G_{ave}^{i+1} - \hat{\delta}^{i+1}) \tag{25}$$

where

$$\hat{\gamma}^{i+1} = \frac{1}{2N} \sum_{j=1}^{N} (\gamma_j^{i+1} + \gamma_{j-1}^{i+1})$$

$$\hat{\delta}^{i+1} = \frac{1}{2N} \sum_{j=1}^{N} (\delta_j^{i+1} + \delta_{j-1}^{i+1})$$

and

$$\gamma_j^{i+1} = \alpha_j^{i+1} \gamma_{j-1}^{i+1}, \ \gamma_o = 1$$

$$\delta_j^{i+1} = \alpha_j^{i+1} \delta_{j-1}^{i+1} + \beta_j, \ \delta_o = 0$$

3.3 Results

The same cases as those of the sectionalized compressible flow model were analyzed. Besides that, a BWR heat flux increase transient was analyzed. The results are shown in Fig. 6 through 10. From Fig. 6, it can be seen that the flow oscillation predicted by the previous model was not observed in this model. For the PWR pressure drop decrease transient, the mass velocity distribution is relatively uniform. For the BWR, the outlet mass velocity decreases at a slower rate compared with that of the inlet mass velocity (Fig. 7). This is due to the fact that the thermal expansion has larger effect in a boiling channel. Figure 8 shows the results of a PWR heat flux increase transient. The mass velocity at the outlet is larger than that at the inlet due to the thermal expansion effect. Again we did not see the flow oscillation phenomenon predicted by previous models.

For the BWR heat flux increase transient, the variation of mass velocity is larger due to the more severe thermal expansion. Figure 10 shows the variations of mass velocity after boiling begins. Compare this figure with Fig. 5; it can be seen that both models can predict thermal expansion, however, the results from the sectionalized compressible flow model are more pronounced.

As expected, the main advantage of the momentum integral method is that the numerical limitation of Eq. (16) is now replaced by the less stringent requirement, i.e.,

$$\Delta t \leq \Delta Z/\left|U_m\right| \tag{26}$$

In Eq. (23), the parameter $(d\rho_m/dH_m)_{j+1/2}$ is evaluated by the following equation.

$$\left(\frac{d\rho_m}{dH_m}\right)_{j+1/2} = \frac{(\rho_m)_{j-1} - (\rho_m)_j}{(H_m)_{j-1} - (H_m)_j} \tag{27}$$

The primary reason for doing so is to avoid the drastic change of the derivative as boiling begins.

4. SINGLE MASS VELOCITY MODEL

4.1 Differential Equations

It is assumed that the density change due to the fluid expansion is neglected. The mass equation Eq. (1) simplifies to

$$\frac{\partial G_m}{\partial z} = 0 \tag{28}$$

Integrate the momentum equation over the channel length. We have

$$\frac{\partial G}{\partial t} = \frac{1}{L}\left(\Delta p - F\right) \tag{29}$$

where

$$F = \int_o^L \frac{\phi_{lo}^2\ (f/\rho)\left|G_m\right|G}{2De}\,dz + \int_o^L \rho_m g dz + \left(\frac{G_m^2}{\rho_m}\right)_{out} - \left(\frac{G_m^2}{\rho_m}\right)_{in}$$

We also neglect the friction and pressure terms in the energy equation

$$\rho_m \frac{\partial H_m}{\partial t} + G_m \frac{\partial H_m}{\partial z} = \frac{q'}{A} \tag{30}$$

4.2 Finite Difference Equations

From Eq. (30)

$$(\rho_m)^i_{j-1/2} \frac{(H_m)^{i+1}_j - (H_m)^i_j + (H_m)^{i+1}_{j-1} - (H_m)^i_{j-1}}{2\Delta t}$$

$$+ G^i_m \frac{(H_m)^{i+1}_j - (H_m)^{i+1}_{j-1} + (H_m)^i_j - (H_m)^i_{j-1}}{2\Delta z} = \frac{q'}{A}$$

Rearrange it.

$$(H_m)^{i+1}_j = (H_m)^i_{j-1} - \frac{1-\alpha}{1+\alpha} \left[(H_m)^{i+1}_{j-1} - (H_m)^i_j\right] + \frac{q'/A \cdot 2\Delta t}{(\rho_m)^i_{j-1/2}\ (1+\alpha)} \tag{31}$$

where

$$\alpha = \frac{G^i \Delta t}{(\rho_m)^i_{j-1/2}\ \Delta z}$$

From Eq. (29)

$$G^{i+1}_m = G^i_m + \frac{\Delta t}{L} (\Delta p^i - F^i) \tag{32}$$

We can either use Eq. (32) to calculate the new mass velocity for a pressure transient or use it to calculate the pressure drop for a flow transient. After we know the mass velocity, we can use Eq. (31) to calculate the enthalpy and density distribution. Then determine the F^i accordingly.

4.3 Results

The results from this model and those from the momentum integral model for a PWR pressure drop decrease transient are compared in Fig. 11. In the momentum integral model, after boiling takes place, the fluid being expelled from the channel causes the exit mass velocity to be considerably greater than that at the inlet. The single mass velocity model follows the behavior of the momentum integral model up to the point of boiling. Afterwards, however, the single mass velocity remains within the limit of the other two velocities of the momentum integral model for only a short time and deviates considerably from both during the later stages of the transient.

The numerical scheme is stable for reasonable time step size. It is still unclear what is the numerical stability limitation.

5. CHANNEL INTEGRAL MODEL

5.1 Differential Equations [4]

The basis of this model is an integration of the laws of conservation of mass, momentum and energy over the length of the channel. During this integration, the shape of the enthalpy profile is considered known and invariant during the course of the transient. Those integrated balance operations are:

$$\frac{dM}{dt} = G_o - G_n \tag{33}$$

$$\frac{d\hat{G}}{dt} = \frac{1}{L}(\Delta p - F) \tag{34}$$

$$\frac{dE}{dt} = Q - G_n \Delta H \tag{35}$$

where

$$M = \int_o^L \rho_m dz$$

$$E = \int_o^L \rho_m((H_m) - H_{in})dz$$

$$\hat{G} = \frac{1}{L}\int_o^L G dz$$

The shape of enthalpy profile in the integration is

$$H_m(z,t) = H_{in} + B(z)(\hat{H}(t) - H_{in})$$

where

$$\hat{H}(t) = \frac{1}{L}\int_o^L H_m(z,t)dz$$

and

$$\frac{1}{L}\int_o^L \beta(z)dz = 1, \ \beta_o = 0$$

It can be shown that [5], the mass velocity distribution can be expressed as

$$G_m(z,t) = G_o(t) + \gamma(z,\hat{H}) \ [\hat{G}(t) - G_o(t)] \tag{36}$$

where

$$\frac{1}{L}\int_o^L \gamma(z,\hat{H})dz = 1, \ \gamma_o = 0$$

Define

$$C_1 = \frac{dM}{d\hat{H}}, \quad C_2 = \frac{dE}{d\hat{H}} \tag{37}$$

Then from Eq. (33) and (34)

$$C_1 \frac{d\hat{H}}{dt} = \gamma_n \ (G_{in} - \hat{G}) \tag{38}$$

From Eq. (35) and (36)

$$C_2 \frac{d\hat{H}}{dt} = (Q - \hat{G}\Delta H) + (\gamma_n - 1)(G_n - \hat{G})\Delta H \tag{39}$$

Solve Eqs. (38) and (39) for $\frac{d\hat{H}}{dt}$ and G_o, that is

$$\frac{d\hat{H}}{dt} = \frac{1}{C_3 L} [Q - \hat{G}\Delta H] \tag{40}$$

$$G_o = \hat{G} + \frac{C_1}{C_3 \gamma_n L} [Q - \hat{G}\Delta H] \tag{41}$$

where

$$C_3 = \frac{\gamma_n C_2 - (\gamma_n - 1) C_1 \Delta H}{\gamma_n L}$$

5.2 Finite Difference Equations

In the channel integral model we solve Eqs. (34) and (40) in finite differencd form. That is

$$\hat{H}^{i+1} = \hat{H}^i + \frac{\Delta t}{C_3 L} [Q^i - G^i_{ave} \Delta H^i] \tag{42}$$

$$\hat{G}^{i+1} = \hat{G}^i + \frac{\Delta t}{L} [\Delta p^i - F^i] \tag{43}$$

From the calculated $\hat{H}^{i+1}$, G^{i+1}, we can find the enthalpy and mass velocity distributions with the following relations:

$$(H_m)^{i+1}_j = H_{in} + \beta(z) [(\hat{H}_m)^{i+1} - H_{in}] \tag{44}$$

$$(G_m)^{i+1}_j = G_{in} + \gamma_j(z, \hat{H}_m{}^{i+1})[\hat{G}^{i+1} - G^{i+1}_{in}] \tag{45}$$

From the initial operating condition, we first calculate C_1, C_2 and C_3, then with those values and the enthalpy and density distributions we can evaluate γ_j. By Eq. (45), we can calculate the mass velocity distribution at a new time step. In determining C_1, C_2, C_3 and $\gamma_j(\hat{H})$, there is some iteration over $\hat{H}$ (Eq. 40). That is, using Eq. (40) and the C_1, C_2, C_3, $\gamma_j(\hat{H})$ of the previous time step to estimate $\hat{H}^{i+1}$. Then use this new $\hat{H}^{i+1}$ to calculate C_1, C_2, C_3 and $\gamma_j(\hat{H})$. Repeat this procedure until two successive $\hat{H}^{i+1}$ get close enough.

The numerical limitations of the channel integral model are unclear. It is only known that the time step can not be too small (on the order of ms). Otherwise, the situation of dividing by zero will happen in the iteration stated above.

5.3 Results

The same cases as those of Section 2 were analyzed and the results are shown in Fig. 12 to Fig. 15. From Fig. 12 and Fig. 3, it can be seen that owing to the neglect of the enthalpy transport effect (by assuming the enthalpy profile) we tend to overpredict the outlet mass velocity especially as the transient time is small. The situation is worse for the BWR. For PWR heat flux increase transient, we see approximately the same trends in the channel integral model as those of the momentum integral model. However, the mass velocity profiles of the two are not exactly the same (Fig. 13 and Fig. 8). Figure 15 shows the mass velocity profile after boilng begins in a PWR pressure drop decrease transient. The mass velocity profile keeps the same shape even after the boiling begins which is different from the prediction of the sectionalized compressible model and momentum integral model. Again, this is believed to be the effect of enthalpy transport.

6. CONCLUSION

The transient response of a single heated channel have been calculated by several different models. Those models involve different levels of assumptions. Among those, the sectionalized compressible flow model represents the most detailed approach. However, one pays for performing this kind of detailed calculation. The computer time is very long. The numerical scheme for sectionalized compressible model used in this analysis is not good enough. It is quite sensitive to which value of the sonic velocity is used in the finite difference equations. Sometimes, the numerical scheme does not work for a certain kind of transient, e.g., for a BWR heat flux increase transient. The sudden change of sonic velocity at the boiling boundary causes a lot of problems and give unreasonable results.

The time step size of the momentum integral model is less restrictive, order of magnitudes different, compared with that of sectionalized compressible flow model. The results of momentum integral model are only significantly different from those of sectionalized compressible flow model in the first few tenths of milliseconds. If we want to calculate the long term response of a transient, the momentum integral model seems good enough. The momentum integral model preserves the thermal expansion effect.

The single velocity is the simplest method we can use to solve the transient responses of a single heated channel. It neglects the sonic effect and thermal expansion effect, but preserves the enthalpy transport effect. Therefore, we can expect the single velocity will give reasonable results as long as thermal expansion of the coolant is not very large, i.e., it remains as a single phase.

In the channel integral model, a preassumed enthalpy profile is used throughout the transient. This will cause the results to deviate from the true solution. It is believed that the situation will be even worse for a nonuniform heat flux distribution or a transient heat flux that changes its profile. Another interesting phenomenon observed in channel integral model is that there are some oscillations in the outlet mass velocity in a PWR pressure drop decrease transient (Fig. 11). It is still unclear to the authors where these oscillations come from.

In all the calculations above, it has been assumed that the two phase flow can be treated as a homogeneous flow with no slip. It is interesting to know what is the impact of this assumption. For doing so, we need only to modify the definition of densities in the momentum and energy conservation equations and implement those new definitions into a computer code.

7. Reference

(1) J. E. Meyer, "Hydrodynamic Models for the Treatment of Reactor Thermal Transients," Nucl. Sci. and Eng. 19, 269-277 (1961).

(2) R. D. Richtmyer, "Difference Methods for Initial-Value Problems," Interscience, New York, 1957.

(3) J. E. Meyer, J. S. Williams, Jr., "A Momentum Integral Model for the Treatment of Transient Fluid Flow," WAPD BT-25 (1962).

(4) J. E. Meyer, W. D. Long, "A Channel Integral Model for the Treatment of Transient Fluid Flow," WAPD-BT-23 (1961).

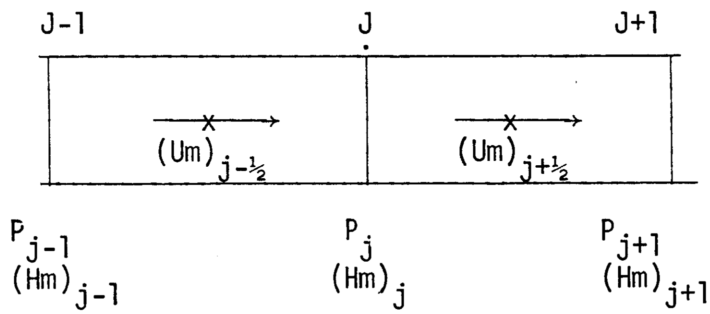

Fig. 1. Sectionalized Compressible Flow Model Calculation Cell.

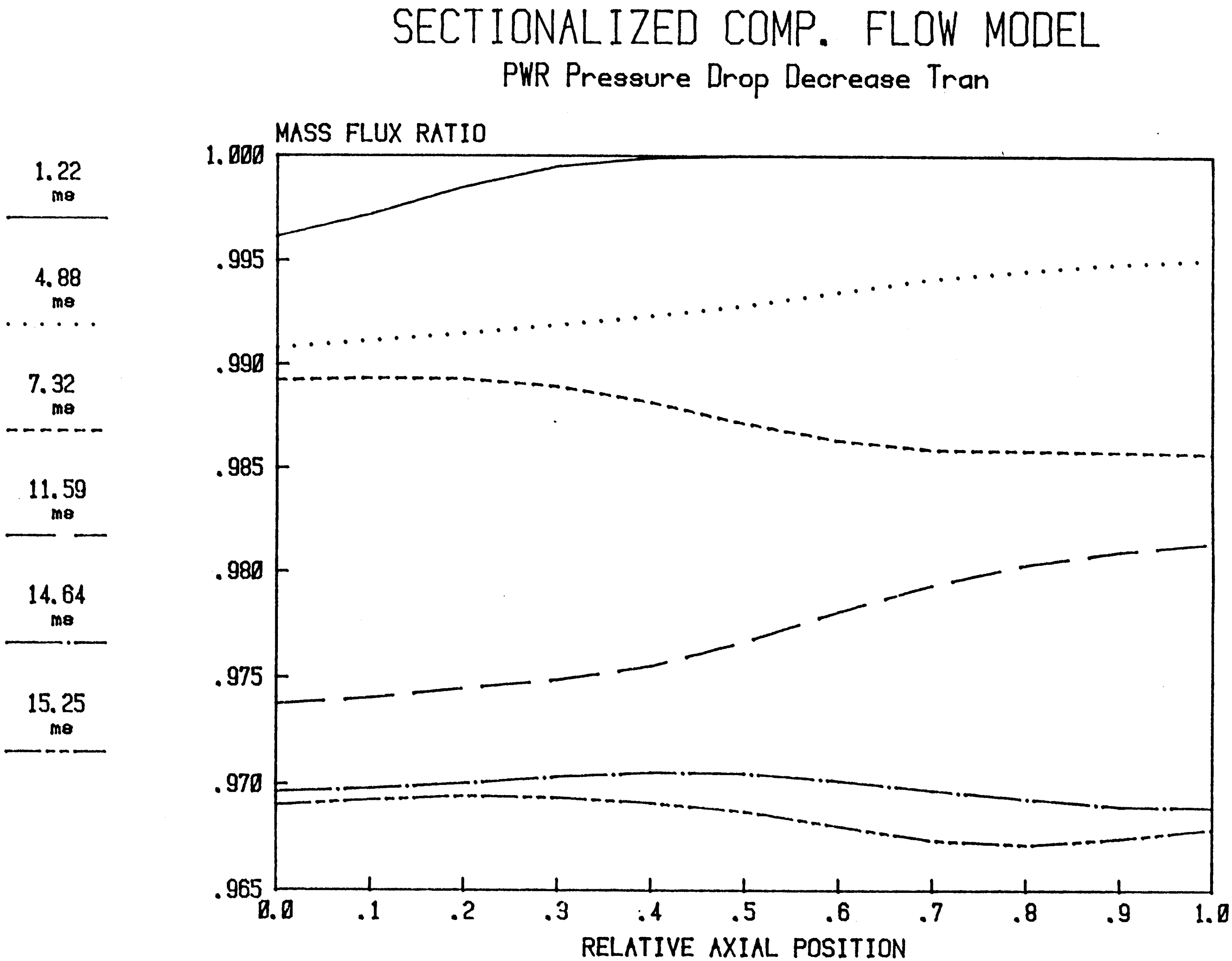

Fig. 2. Sectionalized Compressible Flow Model (PWR Pressure Drop Decrease Transient)

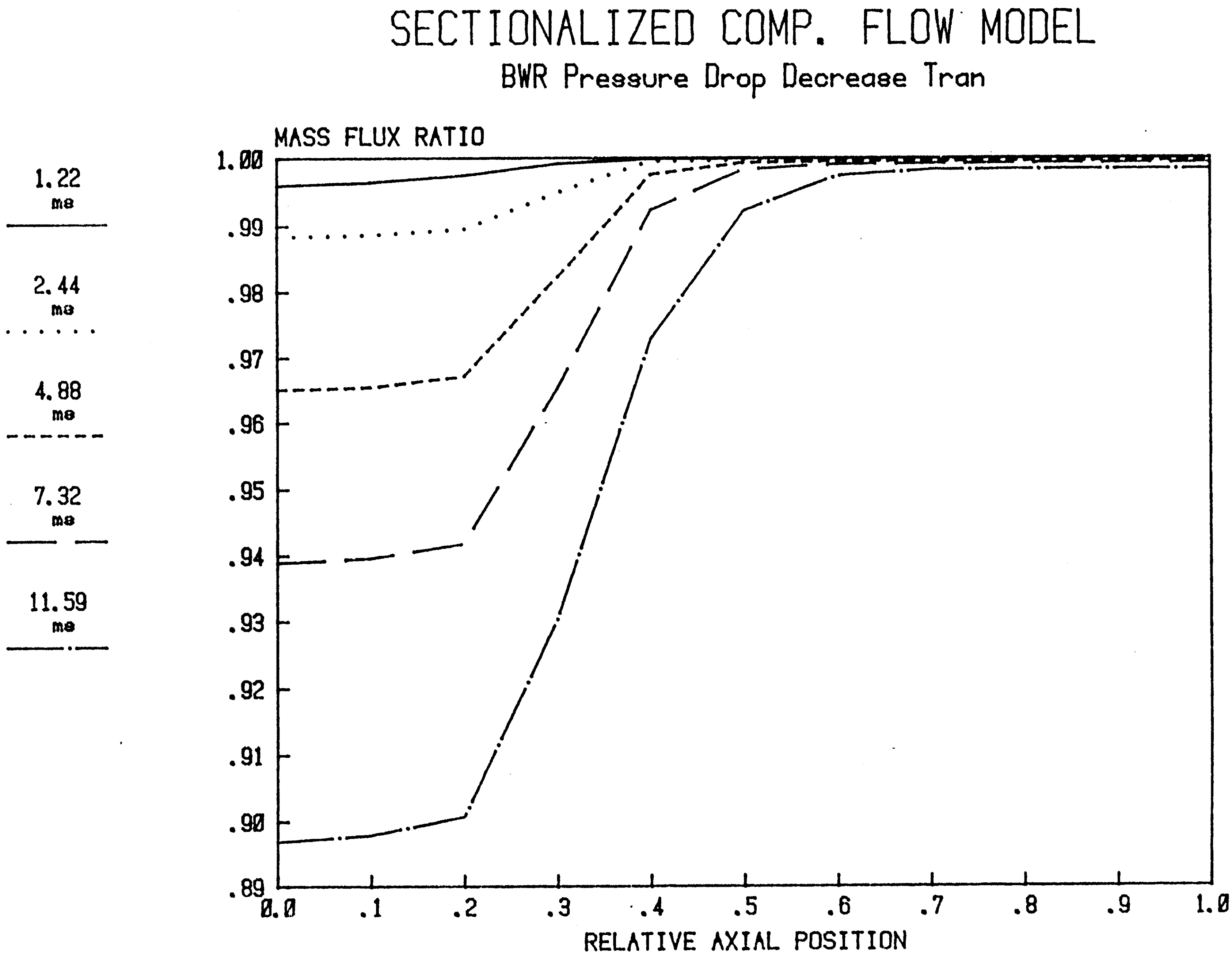

Fig. 3. Sectionalized Compressible Flow Model (BWR Pressure Drop Decrease Transient)

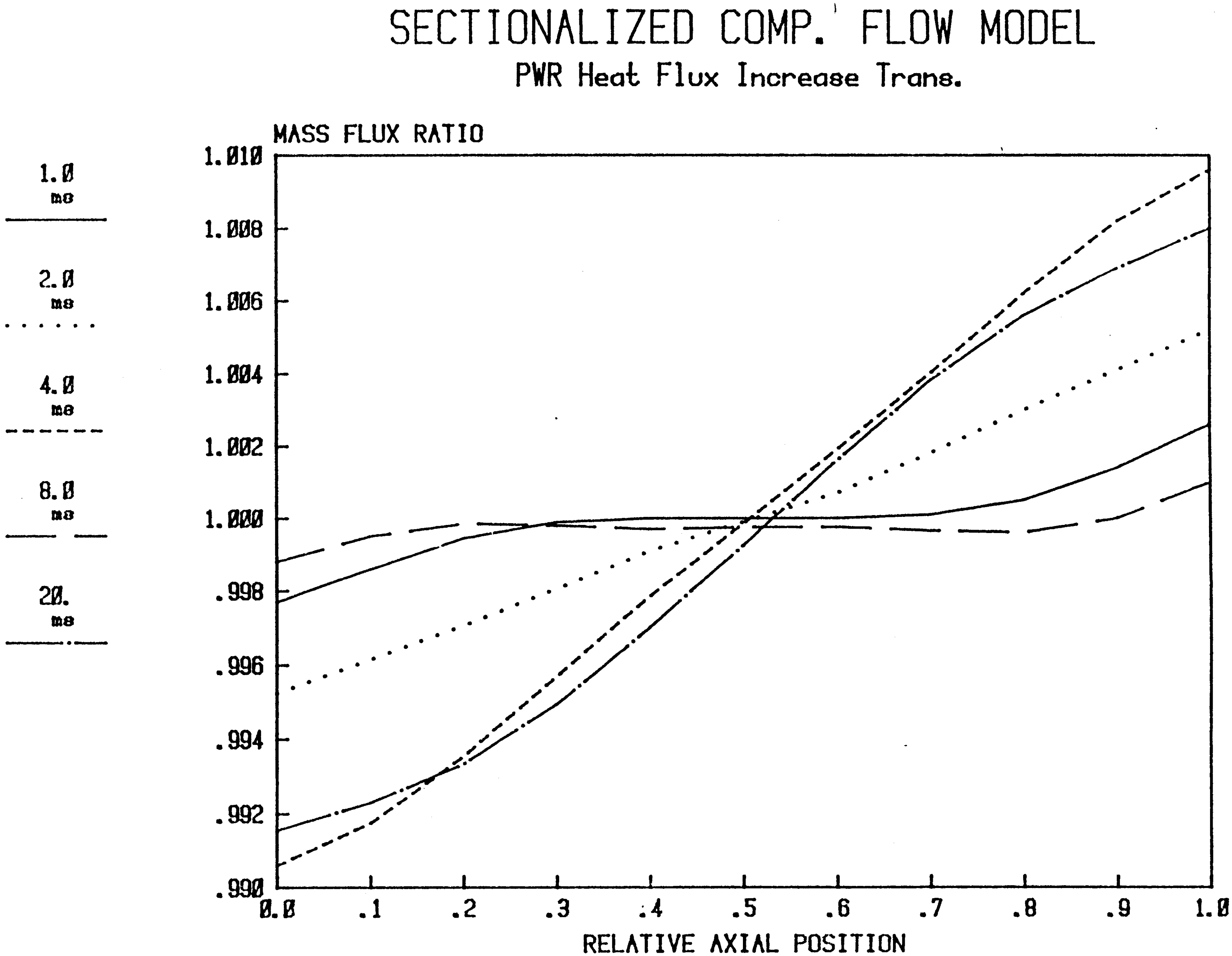

Fig. 4 Sectionalized Compressible Flow Model (PWR Heat Flux Increase Transient)

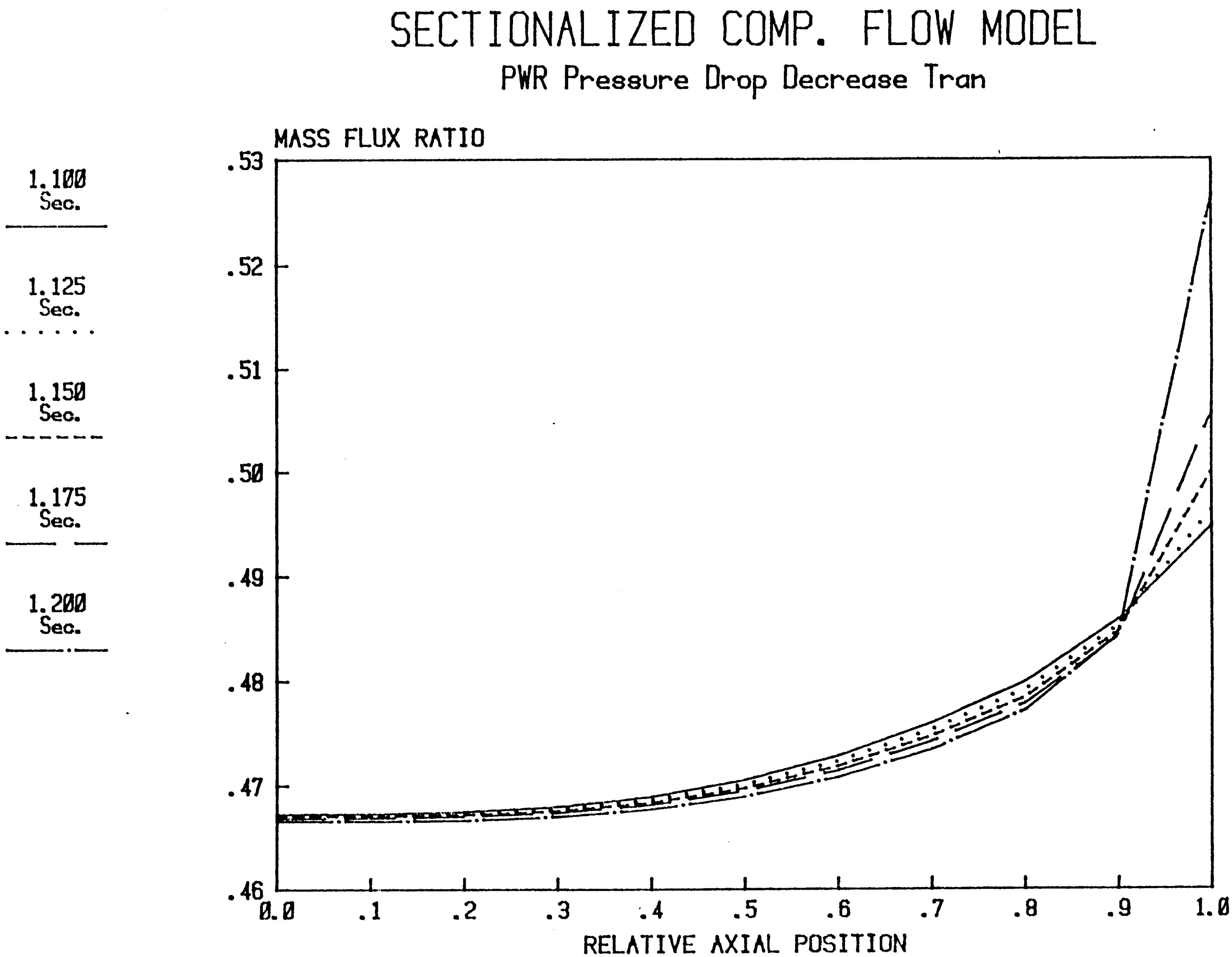

Fig. 5. Sectionalized Compressible Flow Model (PWR Pressure Drop Decrease Transient--Long Term)

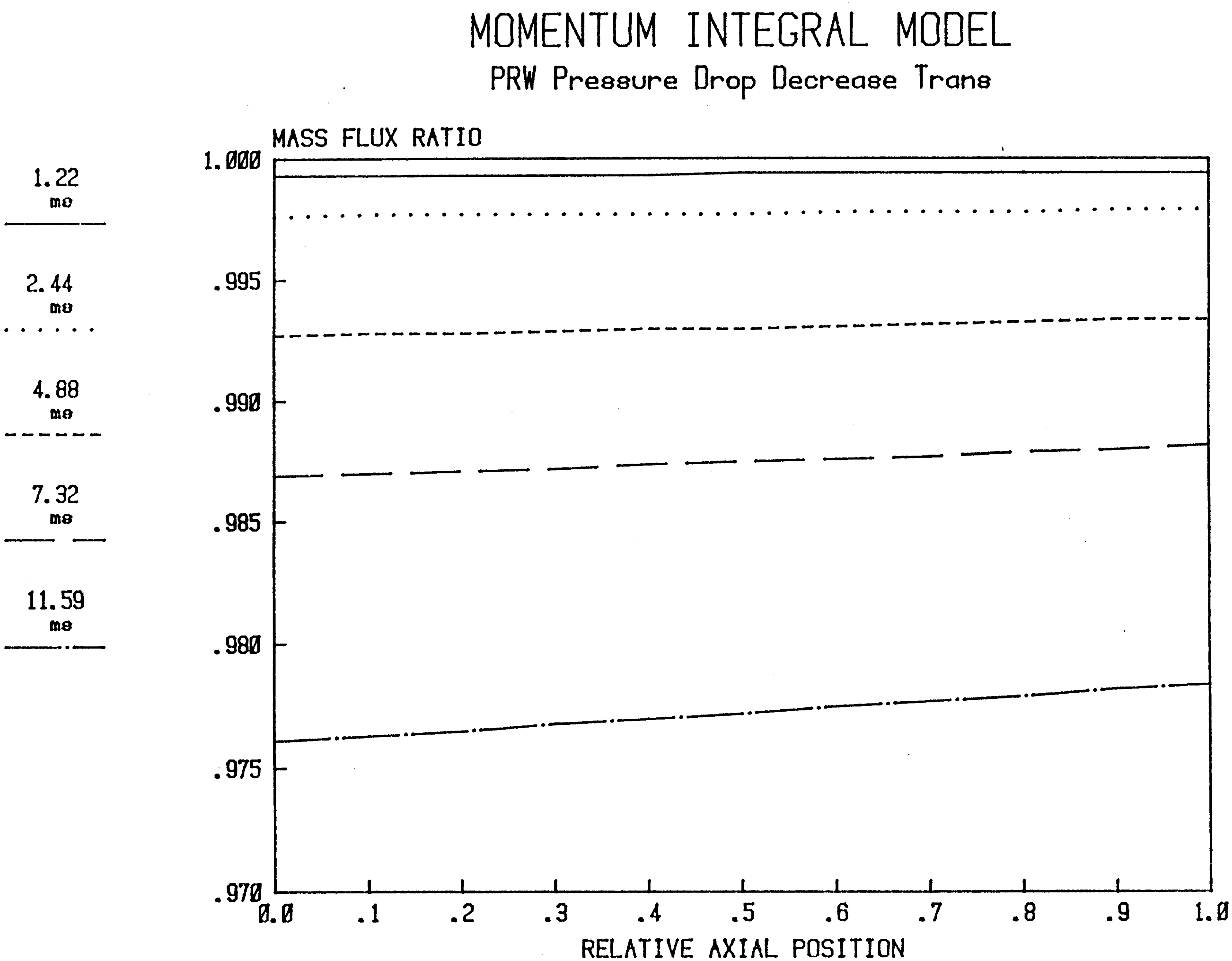

Fig. 6. Momentum Integral Model (PWR Pressure Drop Decrease Transient)

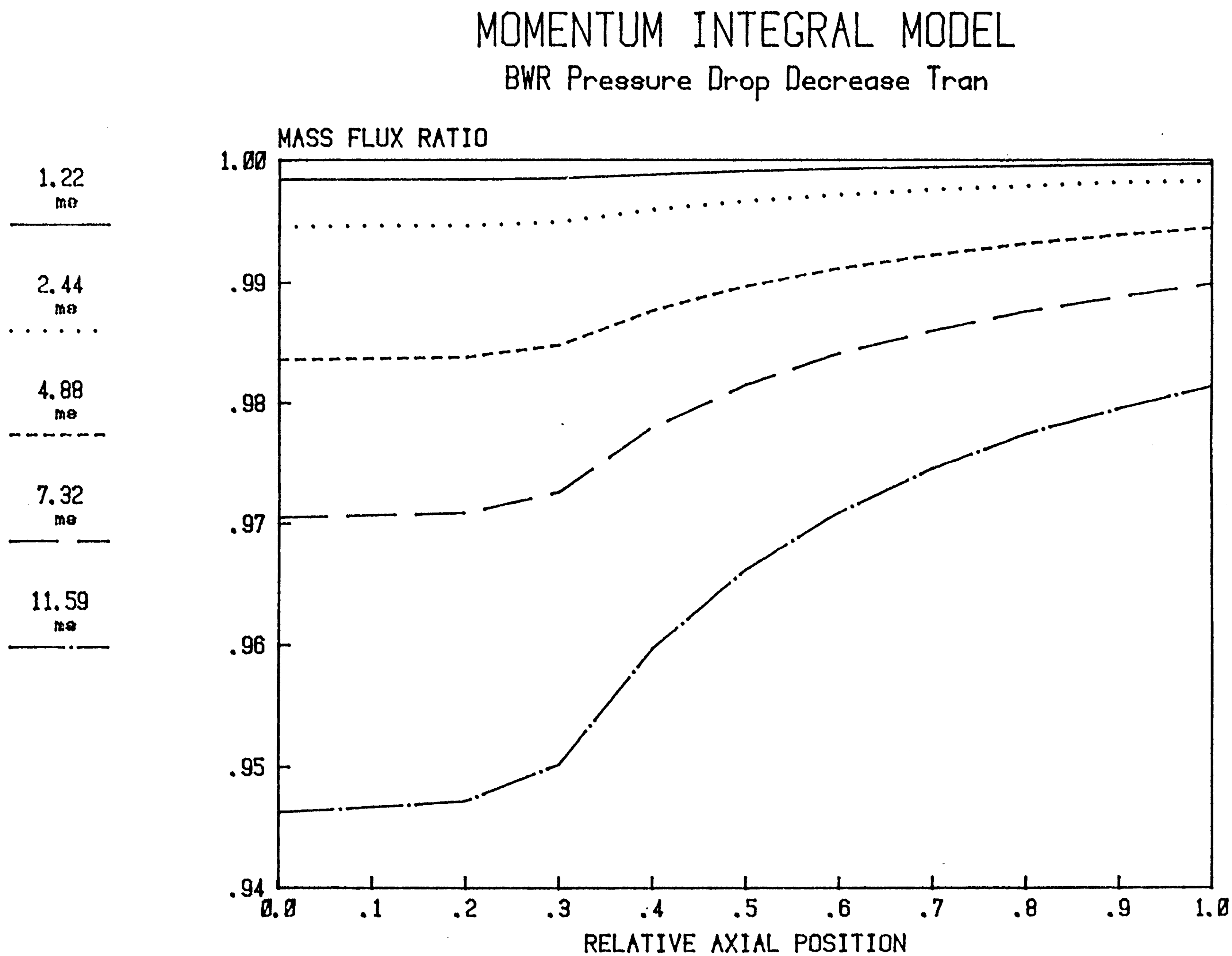

Fig. 7. Momentum Integral Model (BWR Pressure Drop Decrease Transient)

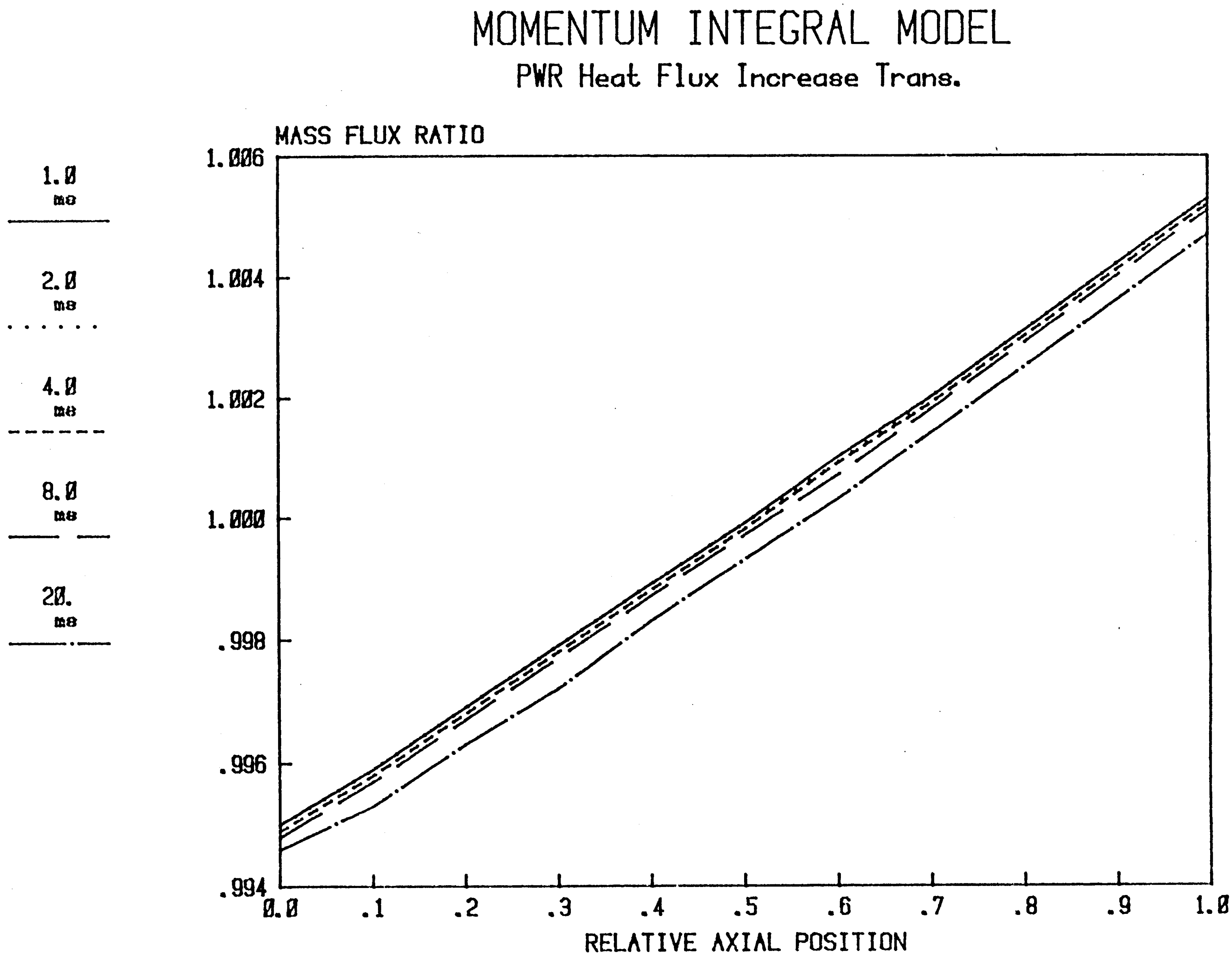

Fig. 8 Momentum Integral Model (PWR Heat Flux Increase Transient)

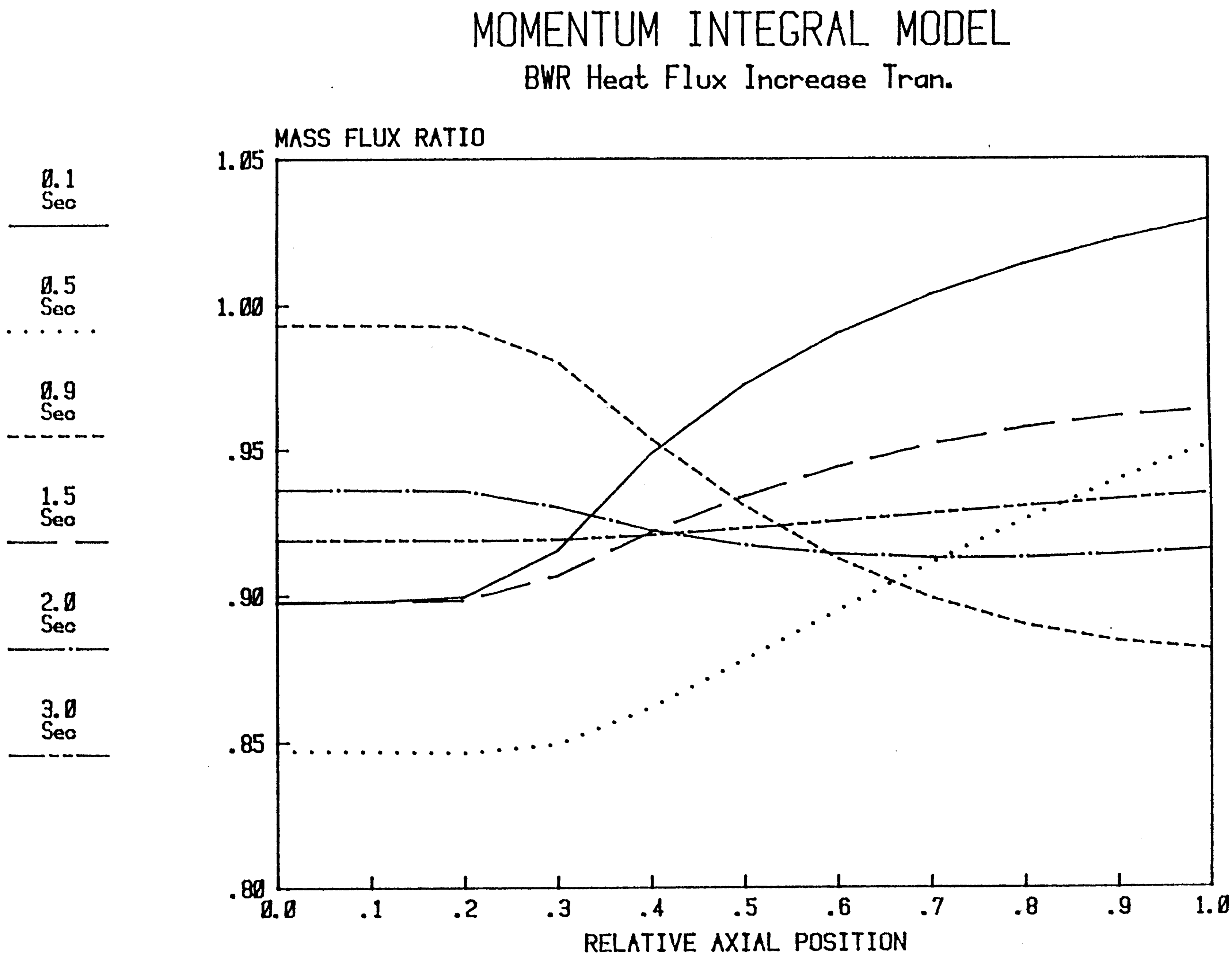

Fig. 9. Momentum Integral Model (BWR Heat Flux Increase Transient)

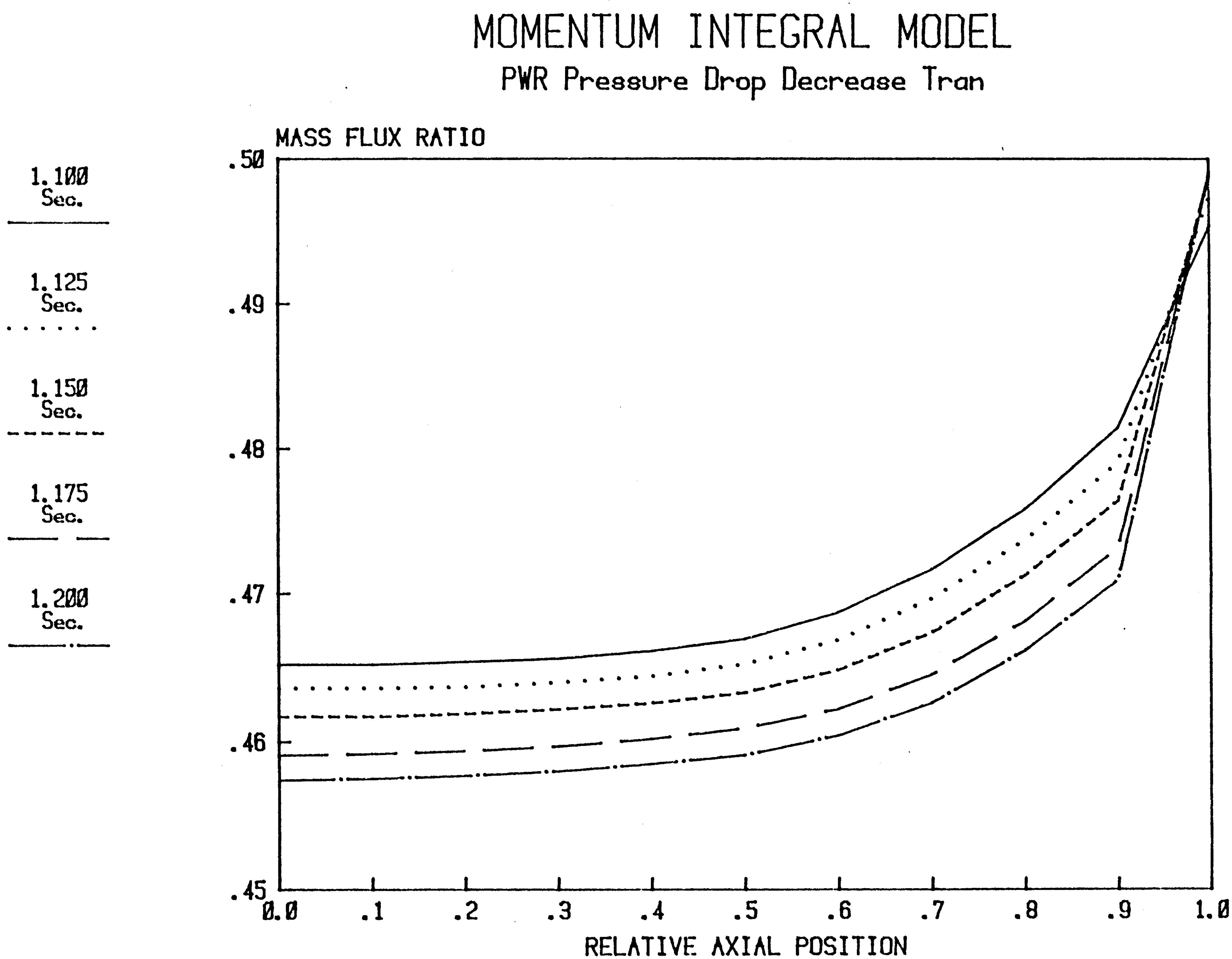

Fig. 10. Momentum Integral Model (PWR Pressure Drop Decrease Transient - Long Term)

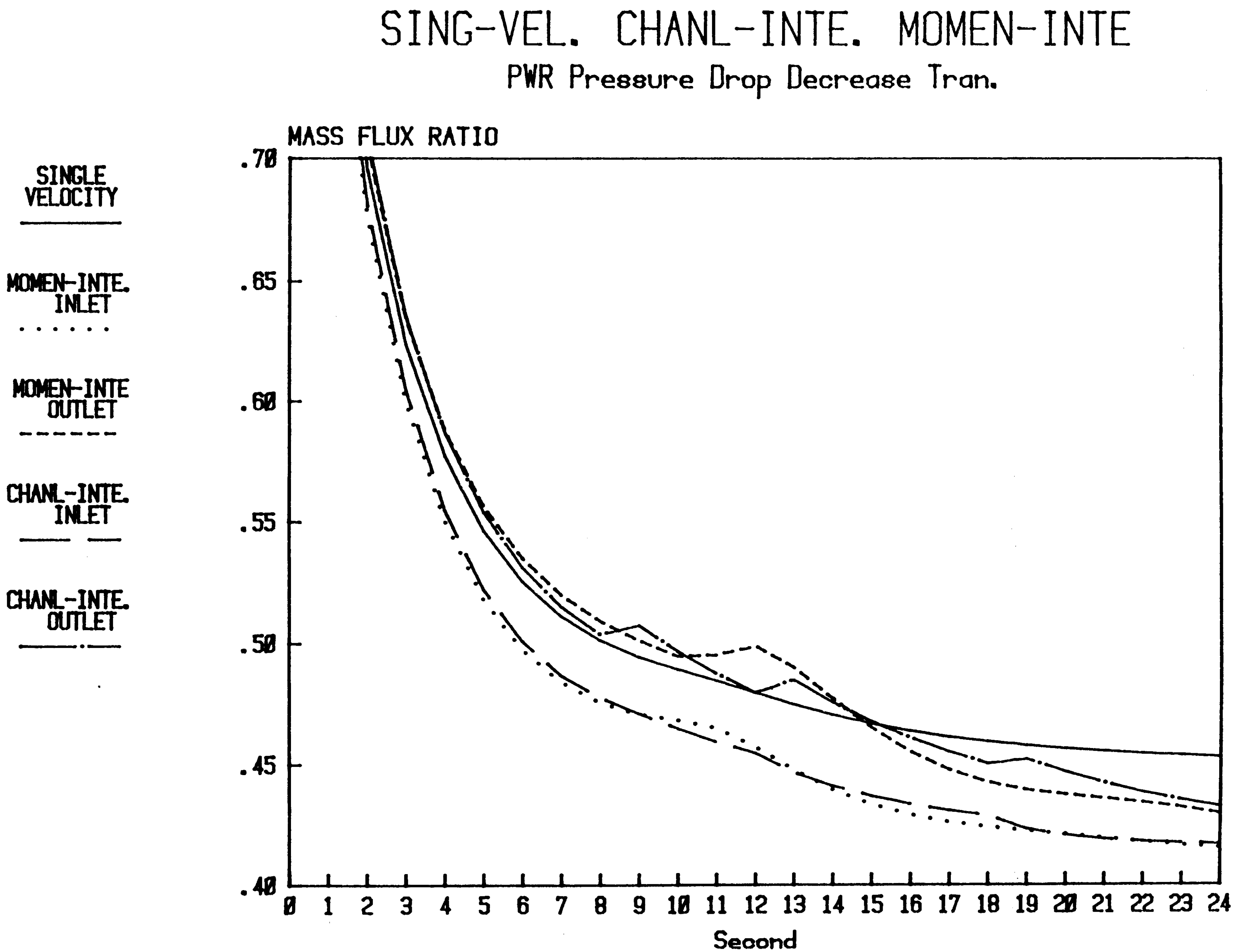

Fig. 11. Comparison between Single Mass Velocity Models and Others (PWR Pressure Drop Decrease Transient)

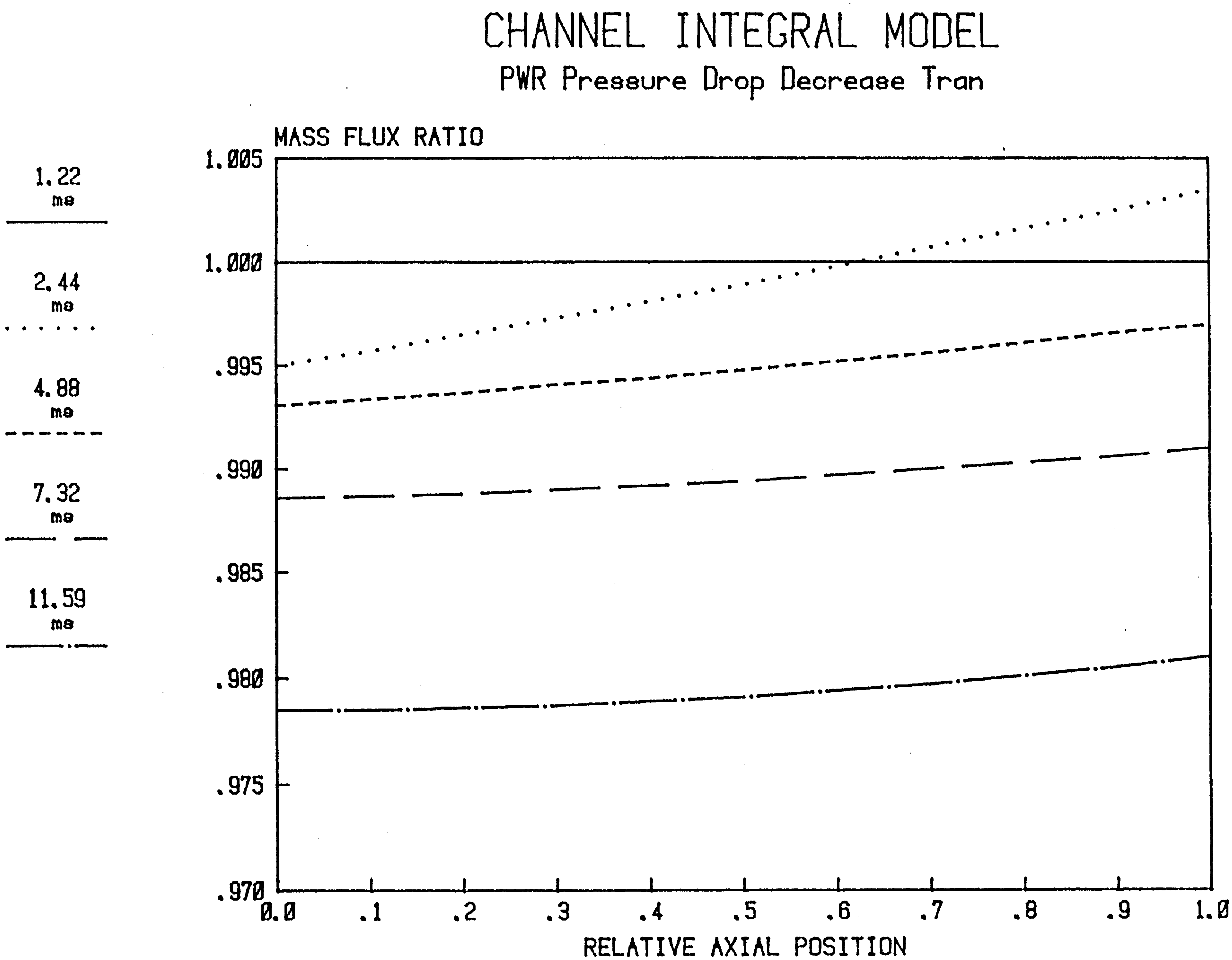

Fig. 12 Channel Integral (PWR Pressure Drop Decrease Transient)

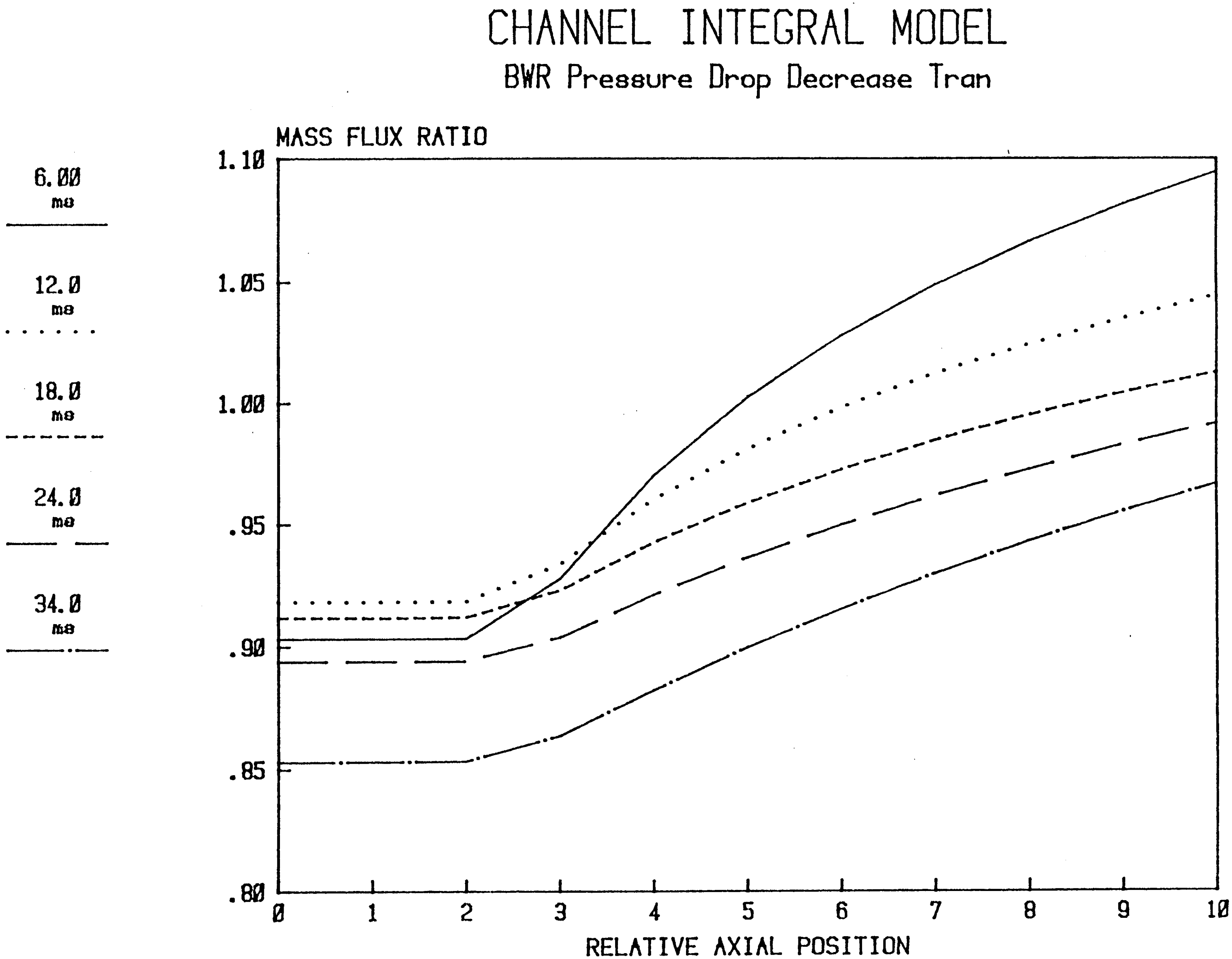

Fig. 13. Channel Integral Model (BWR Pressure Drop Decrease Transient)

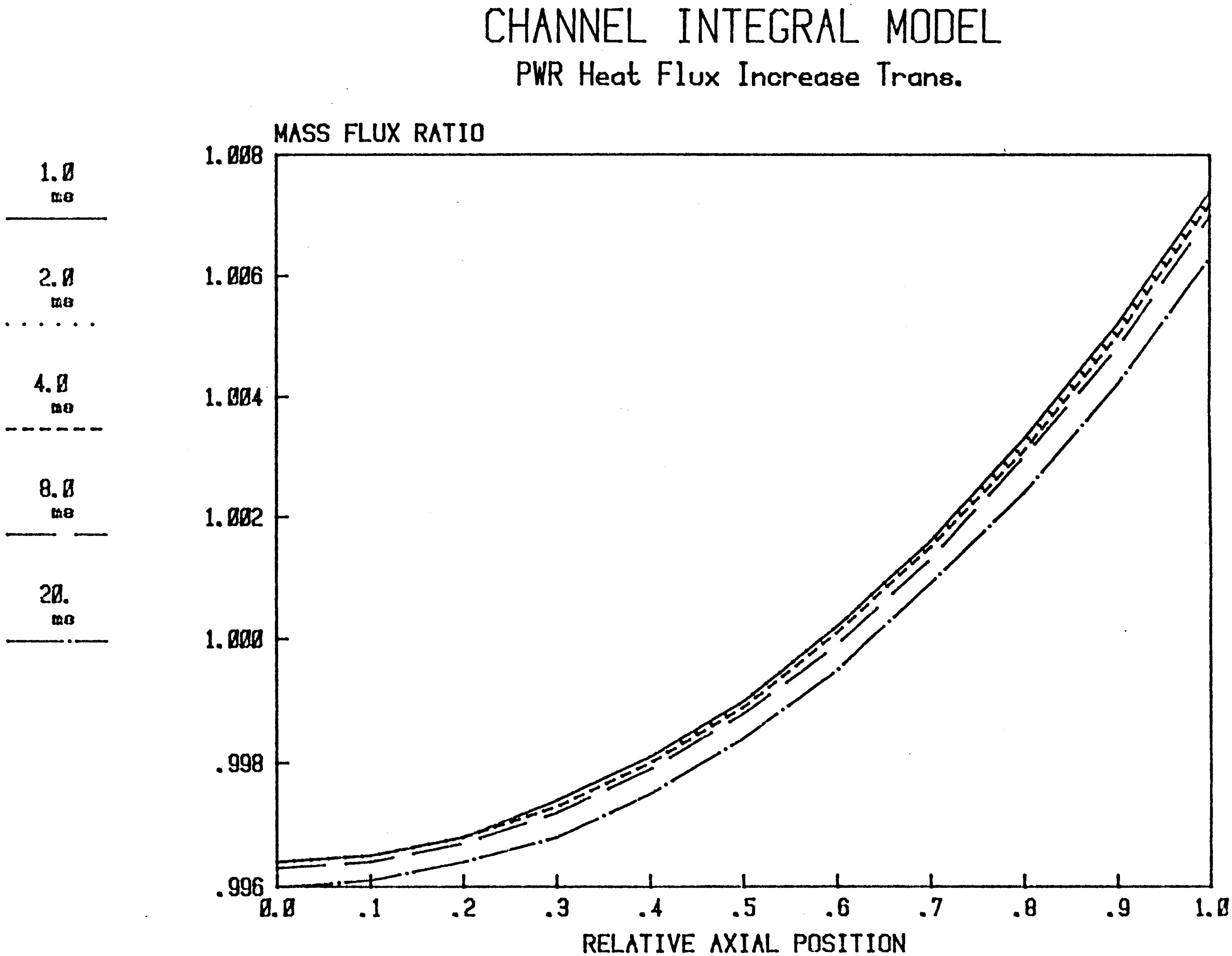

Fig. 14. Channel Integral Model (PWR Heat Flux Increase Transient)

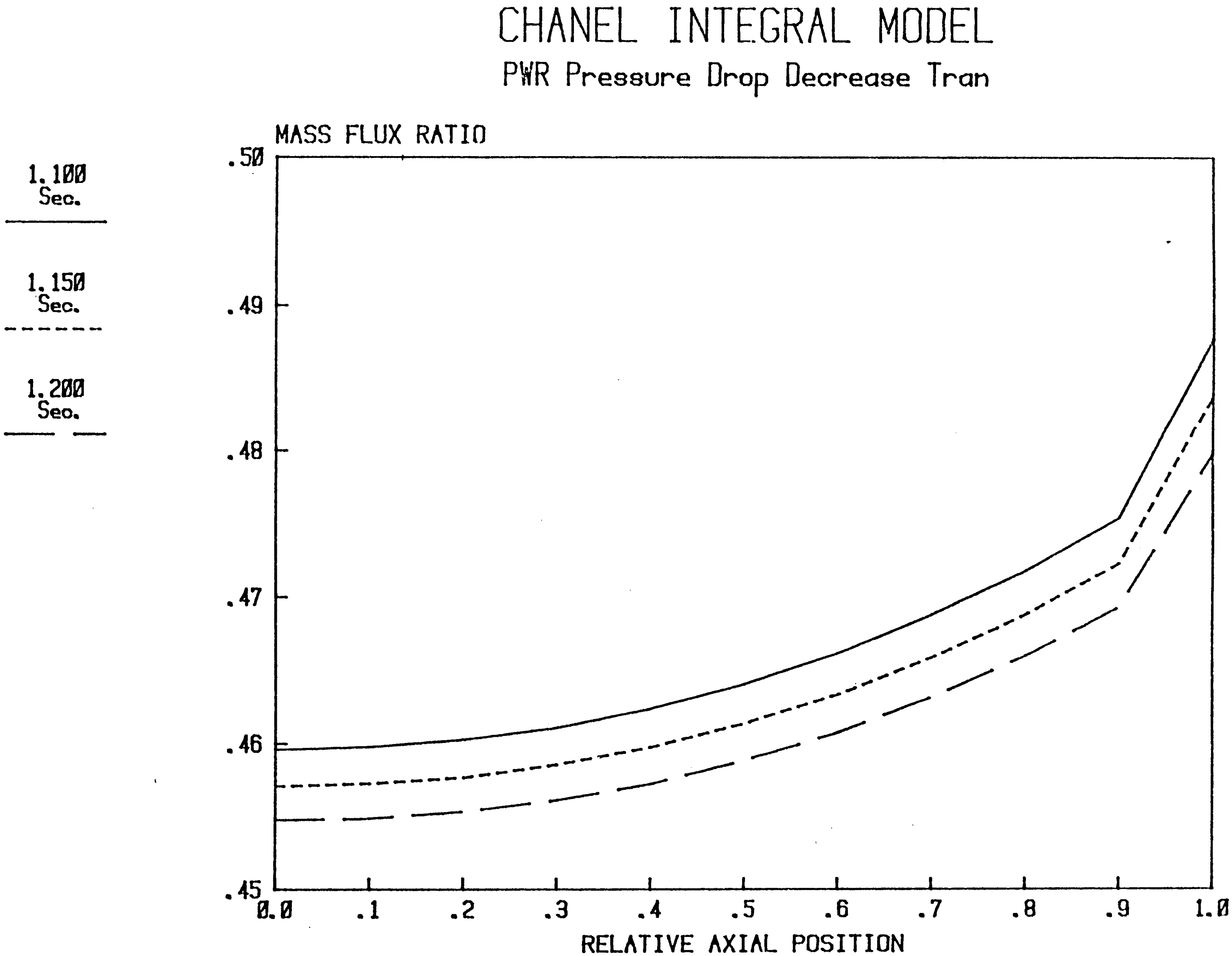

Fig. 15. Channel Integral Model (PWR Pressure Drop Decrease Transient--Long Term)

Appendix I.A: General Information

A separate code was written for each of the following methods:

(1) Sectionalized Compressible Flow Model

(2) Momentum Integral Model

(3) Single Mass Velocity Model

(4) Channel Integral Model.

(1) McAdames correlation was used to calculate the friction factor.

(2) The Equation (5.209) p. 230 in Lahey and Moody was used to calculate the two phase multiplier.

(3) The Equations of State are the same as those of THERMIT [5]. However, the routine has been rewritten.

(4) Subroutine INIT was used to calculate the pressure distribution across the channel. The inlet mass velocity must be given. The initial condition can also be input by the user.

(5) For different kinds of transient, the corresponding routines must be modified. These are pilt, polt, power and gilt. The transient in current coding is for a pressure drop decrease transient.

Appendix I.B: Input Manual

Card 1 Problem Title

80 characters string

Card 2 (1) chanl (e10.5) Channel Length (m)

(2) area (e10.5) Channel Flow Area (m^2)

(3) equd (310.5) Channel Equivalent Diameter (m)

Card 3 (1) is (I5)

1: initial condition calculated by code

Others: initial condition supplied by user

(2) isb (I5)

1: Given mass flux for initial condition calculation

Others: Specified pressure drop for initial condition calculation (only 1 can be input)

(3) iϕ (I5)

Others: Flow rate are specified for B.C.

(4) NPP (I5)

Output will be given for each NPP time steps

Card 4 (1) Time (e10. 6) Total Transient Time (sec)

(2) nots (i5) No. of Time Steps

(3) nosn (i5) No. of Spatial Nodes

Card 5 (1) gilt0 (e10.5) inlet enthalpy (J/kg)

(2) hilto (e10.5) inlet enthalpy (J/kg)

(3) pilto (e10.5) initial inlet pressure (Pa)

(4) polto (e10.5) initial outlet pressure (Pa)

(* this value is unimportant for is=1)

(5) poweo (e10.5) initial linear power (w/m)

Card 6 only need if is ≠ 1

(1) po(i) pressure for ith node (Pa)

(2) go(i) mass velocity for ith node (kg/m^2 sec)

(3) ho(i) enthalpy for ith node (J/kg)

Totally NOSN cards are needed.

Files:

5: input

6: output

0: terminal display

Appendix II: Computer Codes for Transient Response of a Single Heated Channel

A. Sectionalized Compressible Flow Model

B. Momentum Integral Model

C. Single Velocity Model

D. Channel Integral Model

A. Sectionalized Compressible Flow Model

```
c   solutions of transient balance equations for single heated channel
c
c          **  SECTIONALIZED COMPRESSIBLE FLOW MODEL  **
c
c               june  1983
c
      dimension title(20)
      common /param/hn(21),ho(21),gn(21),go(21),dn(21),do(21),pn(21),
     1  po(21),x(21),un(21),uo(21),uin,uio,din,dio
      common /sonvel/ csq(21)
      common  /data1/  chanl,area,equd,nosn,nots,dz,dt
      read (5,1000)(title(i),i=1,20)
 1000 format (20a4)
      read (5,1010)chanl,area,equd
      read(5,1050)is,isb,io,npp
      read(5,1030)time,nots,nosn
 1050 format(4i5)
 1010 format(3e10.5)
      read(5,1020)gilt0,hilt0,pilt0,polt0,powe0
 1020 format(5e10.5)
 1030 format(e10.6,2i5)
      dz=chanl/(nosn-1)
      dt=time/nots
      if(is.eq.1)go to 130
      do 10 i=1,nosn
      read (5,1040)po(i),go(i),ho(i)
   10 continue
 1040 format(3e10.5)
      go to 140
c
c
c   calculate the steady state density distribution
c
c
c  the initial condition is calculated by sub. init
c
  130 call init(isb,gilt0,hilt0,pilt0,polt0,powe0,
     1      po,ho,do,go,x)
  140 continue
      nosn1=nosn-1
      do 90 i=1,nosn1
      ha=(ho(i)+ho(i+1))/2
      pa=(po(i)+po(i+1))/2.
      call densi(pa,ha,do(i),x(i))
      uo(i)=go(i)/do(i)
   90 continue
      call densi(po(nosn),ho(nosn),do(nosn),x(nosn))
      call densi(po(1),ho(1),dio,xa)
      uio=gilt0/dio
      uo(nosn)=go(nosn)/do(nosn)
      write(6,2000)
 2000 format (//," transient solutions of single heated channel by",
     1 " sectionalized compressible fluid model ")
      write(6,2010)(title(i),i=1,20)
 2010 format (/,20a4)
      write(6,2020)
 2020 format(/," channel geometry ")
      write (6,2030)chanl,area,equd
 2030 format(1x," channel length=",f6.3," m",
```

```
     1/," flow area  = ",e13.6," m**2",
     1        /,"  equi  diame = ",f6.3," m")
      write(6,2040)
 2040 format(/," operating condition ")
      hiltw=hilt0/1000
      piltw=pilt0/1.e6
      poltw=polt0/1.e6
      powew=powe0/1000
      write (6,2050)gilt0,hiltw,piltw,poltw,powew
 2050 format(1x," inlet mass flux =",f8.3," kg/m**2.sec",
     1        /,"  inlet enthalpy = ",f8.3," kj/kg",
     1        /,"  inlet pressure = ",f7.4," Mpa",
     1        /,"  outlet pressure = ",f7.4," Mpa",
     1      /,"   power  =  ",f7.4," kw/m")
      write(6,2300)time,dt
 2300 format(/," total transient time =",e13.6,
     1 /," time step size ",e13.6)
      write(6,2110)
      write(6,2070)
      do 150 i=1,nosn
      pw=po(i)/1.e6
      hw=ho(i)/1000
      write(6,3080)i,pw,go(i),hw,do(i),uo(i),x(i)
  150 continue
 2110 format(/,"  initial conditions for transient calculation ")
c
      np=0
      do 30 i=1,nots
 9000 format(1x," the time step ",i6)
      np=np+1
      ttime=dt*i
      time1=ttime-dt
      tpolt=polt(polt0,ttime)
      qipt=power(powe0,time1)
      hilt=hilt0
      call calc(tpolt,hilt,qipt,ieror)
      if(ieror.eq.1) go to 71
      dtdz=dt/dz
c
c     determine the inlet velocity or inlet pressure
c
      if(io.eq.1) go to 40
c
c      determine the inlet pressure for a flow reduced transient
c
      tgilt=gilt(gilt0,ttime)
      call densi(po(1),hn(1),din,xa)
      ha1=(3*ho(1)+ho(2))/4
      pa=(3*po(1)+po(2))/4
      call densi(pa,ha1,da,xa1)
      call rph(pa,ha1,da,xa1,rp1,rha1)
      csqa1=1/(rp1+rha1/da)
      if(csqa1.le.0.0)write(0,8888)csqa1
 8888 format(1x,"  csqa1",1x,e13.6)
      uin=tgilt/din
      uan=(uin+un(1))/2.
      ga1=da*uan
      call frici(xa,pa,ga1,da,foda,ieror)
      if(ieror.eq.1) go to 71
      pa=(3*po(2)+po(1))/4.
```

```
      ha2=(3*ho(2)+ho(1))/4.
      call densi(pa,ha2,daa,xaa)
      call rph(pa,ha2,daa,xaa,rpa2,rha2)
      csqa2=1./(rpa2+rha2/daa)
      csq1=((csqa1**.5+csqa2**.5)/2.)**2.
      pa=(po(1)+po(2))/2.
      call densi(pa,ha1,da1,xa1)
      call densi(pa,ha2,da2,xa2)
      rh1=(da2-da1)/(ha2-ha1)
      daaa=(daa+da)/2.
      c1=(daaa*csq1)*(un(1)+un(2)-2*uin)*dtdz/2.
      c2=rh1/(daaa/csq1)*(qipt/area+foda*da**2*uan**3/(
     1    2*equd))*dt
      c3=(uin+un(1))*(po(2)-po(1))/2.*dtdz
      pn(1)=po(2)+po(1)-pn(2)-2*(c1+c2+c3)
      go to 60
c
c     determine the inlet velocity for pressure transient
c
   40 pn(1)=pilt(pilt0,polt0,ttime)
      call densi(pn(1),hn(1),din,xa)
      ha1=(3*ho(1)+ho(2))/4
      pa=(3*po(1)+po(2))/4
      ua=(uio+uo(1))/2
      call densi(pa,ha1,da,xa)
      call rph(pa,ha1,da,xa,rpa1,rha1)
      csqa1=1./(rpa1+rha1/da)
      ga=ua*da
      call fric1(xa,pa,ga,da,foda,ieror)
      if(ieror.eq.1)go to 71
      pa=(3*po(2)+po(1))/4.
      ha2=(3*ho(2)+ho(1))/4.
      call densi(pa,ha2,daa,xaa)
      call rph(pa,ha2,daa,xaa,rpa2,rha2)
      csqa2=1./(rpa2+rha2/daa)
      if(csqa2.le.0.0) write(0,8887)csqa2
 8887 format(1x," csqa=",e13.6)
      csqa=((csqa1**.5+csqa2**.5)/2.)**2.
      pa=(po(2)+po(1))/2.
      call densi(pa,ha1,da1,xa1)
      call densi(pa,ha2,da2,xa2)
      rha=(da2-da1)/(ha2-ha1)
      daaa=(da+daa)/2.
      c1=(foda*da*ua**2/(2*equd)+9.8)*dt*2
      c2=(po(2)-po(1))/da*dtdz*2.
      c3=(uo(1)+uio)*(uo(1)-uio)*dtdz*2
      uin=uo(1)+uio-un(1)-c1-c2-c3
      udif=0.1
      int=0
      itest=0
      it1=0
      d1=0.
      uio1=uin
      uin1=uin
   61 int=int+1
      if(itest.eq.0)uin=uin1
c
c      iteration over the pressure boundary
c
      uan=(uin+un(1))/2.
```

```
      c1=(daaa*csqa)*(un(1)+un(2)-2*uin)*dtdz/2.
      c2=rha/(daaa/csqa)*(qipt/area+foda*da**2.*uan**3./(2*equd))*dt
      c3=(uin+un(1))*(po(2)-po(1))/2.*dtdz
      pn1=po(2)+po(1)-pn(2)-2*(c1+c2+c3)
      d2=pn1-pn(1)
      if(abs(d2).le.1.0)go to 62
      if(itest.eq.1)go to 66
      if(d1.eq.0.0)go to 63
      if((d1*d2).le.0.0)go to 64
   63 uio1=uin1
      if(it1.eq.1)go to 67
      if(d2.gt.0.0)uin1=uin1-udif
      if(d2.lt.0.0)uin1=uin1+udif
      if(d1.eq.0.0)go to 65
      if(abs(d2).le.abs(d1))go to 65
      it1=1
   67 if(d2.gt.0.0)uin1=uin1+udif
      if(d2.lt.0.0)uin1=uin1-udif
      go to 65
   64 itest=1
      if(d1.gt.0.0)go to 68
      uh=uin1
      ul=uio1
      dh=abs(d2)
      dl=abs(d1)
      go to 69
   68 uh=uio1
      ul=uin1
      dh=abs(d1)
      dl=abs(d2)
   69 uin=(ul*dh+uh*dl)/(dl+dh)
      go to 65
   66 if(d2.gt.0.0)go to 59
      ul=uin
      dl=abs(d2)
      go to 69
   59 uh=uin
      dh=abs(d2)
      go to 69
   65 if(int.gt.50)go to 70
      d1=d2
      go to 61
   62 continue
      tgilt=uin*din
   60 predi=pn(1)-pn(nosn)
c
      do 260 i1=1,nosn1
      ha=(hn(i1)+hn(i1+1))/2
      pa=(pn(i1)+pn(i1+1))/2.
      call densi(pa,ha,dn(i1),x(i1))
      gn(i1)=dn(i1)*un(i1)
  260 continue
      call densi(pn(nosn),hn(nosn),dn(nosn),x(nosn))
      gn(nosn)=un(nosn)*dn(nosn)
      do 110 i1=1,nosn
      po(i1)=pn(i1)
      ho(i1)=hn(i1)
      go(i1)=gn(i1)
      do(i1)=dn(i1)
      uo(i1)=un(i1)
```

```
  110 continue
      uio=uin
      dio=din
      if(i.eq.20000)npp=100
      if(i.eq.20000)np=0
      if(np.ne.npp)go to 31
   71 continue
c      write(0,9000)i
      np=0
      predw=predi/1000
      qiptw=qipt/1000
      write (6,2060)ttime,predw,qiptw,uin,tgilt
      write(6,2070)
      do 120 i1=1,nosn
      pw=pn(i1)/1.e6
      hw=hn(i1)/1000
      gr=gn(i1)/gilt0
      sonic=csq(i1)**0.5
      write(6,2080)i1,pw,gn(i1),hw,dn(i1),un(i1),x(i1),gr,sonic
  120 continue
      if(ieror.eq.1) go to 160
 2060 format(//," transient time=",e13.6," sec ",
     1       /,"  pressure drop = ",f8.4," kpa",
     1       /,"    power   =  ",f8.4, " kw/m",
     1       /,"   inlet velocity = ",f7.4, "  m/sec",
     1       /,   "   inlet mass fux = ",f8.3,"  kg/m**2.sec")
 2070 format(/"       pressure  ",
     1    "  mass flux  enthalpy   density",
     1    "   velocity   quality ",
     1    /,7x, "  Mpa    kg/m**2.sec   kj/kg     kg/m**3    m/sec.  ")
 2080 format(1x,i2,3x,f7.4,6x,f7.2,3x,f8.3,3x,
     1   f7.3,3x,f7.4,4x,f6.3,3x,f7.4,4x,f7.2)
 3080 format(1x,i2,3x,f7.4,6x,f7.2,3x,f8.3,3x,
     1  f7.3,3x,f7.4,4x,f6.3)
   31 continue
   30 continue
      go to 160
c
   70 write(6,2100)ttime
 2100 format (1x," iteration not converge at time eq. ",e13.6)
  160 continue
c
      end
```

```
      subroutine init(is1,gilt0,hilt0,pilt0,polt0,powe0,
     1    p,h,d,g,x)
      dimension h(21),d(21),p(21),g(21),x(21)
      common /data1/ chanl,area,equd,nosn,nots,dz,dt
      call densi(pilt0,hilt0,d(1),x(1))
      h(1)=hilt0
      p(1)=pilt0
      g(1)=gilt0
      i=1
      n1=1
   90 do 10 i=2,nosn
      h1=h(i-1)+powe0*dz/(area*gilt0)
      p1=p(i-1)
      call densi(p1,h1,d(i),x(i))
   60 ha=(h1+h(i-1))/2.
      pa=(p2+p(i-1))/2.
      call densi(pa,ha,da,x1)
      call frict (x1,pa,gilt0,da,fod,ieror)
      p2=p(i-1)-fod*gilt0**2.*dz/(2*equd)
     1  -da*9.80*dz-gilt0**2.*(1/d(i)-1/d(i-1))
      call densi(p2,h1,d(i),x(i))
      pa=(p2+p(i-1))/2.
      call densi(pa,ha,da,x1)
      call frict(x1,p2,gilt0,da,fod,ieror)
   31 h1=h(i-1)+powe0*dz/(area*gilt0)+(p2-p(i-1)+
     1   (fod*gilt0**2*dz)/(2*equd))/da
      if(abs((p2-p1)/p2).le.1.0e-8)go to 40
      n=n+1
      if(n.gt.50)goto 50
      p1=p2
      goto 60
   40 p(i)=p2
      h(i)=h1
      g(i)=gilt0
   10 continue
      if(is1.eq.1)polt0=p(nosn)
      if(abs((p(nosn)-polt0)/polt0).le.1.0e-4)return
      if(n1.gt.50) go to 80
      n1=n1+1
      gilt0=gilt0*((p(nosn)-p(1))/(polt0-pilt0))**0.5
      goto 90
   80 write(6,2110)
 2110 format(1x," iteration not converge for pressure b.c.")
      return
   50 write(6,2090)i
 2090 format(1x,"iteration not converge for spatial node ",i2)
      return
      end
```

```
function power(poweO,time1)
power=poweO
return
end
```

```
function polt(polt0,time1)
polt=polt0
return
end
```

```
function gilt(gilt0,time1)
gilt=gilt0
return
end
```

```
      subroutine fric1(x1,p,g,d,fod,ieror)
      common /data1/ chanl,area,equd,nosn,nots,dz,dt
      data conv1,conv2,vics/737.4643,1.4504e-4,91.7e-6/
      data vics1/19.73e-6/
c
      ieror=0
      if(g.lt.0.0)ieror=1
      if(g.lt.0.0)go to 60
c
c   use McAdames correlation to calculate the friction factor
      thm=1.0
      dpc=d
      if(x1.eq.-1.0)go to 10
      if(x1.eq.1.)go to 20
      call satt(p,hls,hvs)
      call lden(p,hls,rol)
      call vden(p,hvs,rov)
      gtest=g*conv1/1.0e6
      if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest
     1   -0.000714*p*conv2*gtest
      if(gtest.gt.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest
     1+0.00028*p/gtest*conv2
      thm=c1*(1.2*(rol/rov-1)*x1**0.824)+1.0
      dpc=rol
   10 fric=0.184*(g*equd/vics)**(-0.2)
      go to 21
   20 fric=0.184*(g*equd/vics1)**(-0.2)
   21 fod=thm*fric/dpc
   60 continue
      return
      end
```

```
      subroutine calc(tpolt,hilt,qipt,ieror)
      common /sonvel/csq(21)
      common /param/ hn(21),ho(21),gn(21),go(21),dn(21),do(21),pn(21)
     1    ,po(21),x(21),un(21),uo(21),uin,uio,din,dio
      common /data1/ chanl,area,equd,nosn,nots,dz,dt
      dimension  fod(21),rp(21),rh(21)
      nosn1=nosn-1
      nosn2=nosn-2
      pa=po(1)/2.+po(2)/2.
      do 10 i=1,nosn1
      ha=(ho(i)+ho(i+1))/2
      pa=(po(i)+po(i+1))/2.
      call rph(pa,ha,do(i),x(i),rp(i),rh(i))
      csq(i)=1/(rp(i)+rh(i)/do(i))
      if(csq(i).le.0.0)write(0,9999)i,csq(i)
 9999 format(1x,"i=",i4,"csq(i)=",e13.6)
      call frici(x(i),po(i),go(i),do(i),fod(i),ieror)
      if(ieror.eq.1)goto 100
   10 continue
      do 11 i=1,nosn2
      ha1=(ho(i)+ho(i+1))/2.
      ha2=(ho(i+1)+ho(i+2))/2.
      pa1=(po(i)+po(i+1))/2.
      pa2=(po(i+1)+po(i+2))/2.
      call densi(po(I+1),ha1,da1,xa1)
      call densi(po(i+1),ha2,da2,xa2)
      call densi(pa1,ho(i+1),da3,xa3)
      call densi(pa2,ho(i+1),da4,xa4)
      rp(i)=(da3-da4)/(pa1-pa2)
      rh(i)=(da2-da1)/(ha2-ha1)
   11 continue
      pa=(3*po(nosn)+po(nosn1))/4.
      ha1=(ho(nosn1)+ho(nosn))/2.
      call densi(pa,ha1,da1,xa1)
      call densi(pa,ho(nosn),da2,xa2)
      rh(nosn1)=(da2-da1)/(ho(nosn)-ha1)
      ha=(3*ho(nosn)+ho(nosn1))/4.
      pa1=(po(nosn1)+po(nosn))/2.
      call densi(pa1,ha,da3,xa3)
      call densi(po(nosn),ha,da4,xa4)
      rp(nosn1)=(da4-da3)/(po(nosn)-pa1)
      hn(1)=hilt
      dtdz=dt/dz
      c1=uo(1)*(uo(1)-uio)*2*dtdz
      c2=(po(2)-po(1))/do(1)*dtdz
      c3=fod(1)*do(1)*uo(1)**2.*dt/(2*equd)+9.8*dt
      un(1)=uo(1)-c1-c2-c3
      do 40 i=2,nosn1
      c1=uo(i)*(uo(i)-uo(i-1))*dtdz
      c2=(po(i+1)-po(i))/do(i)*dtdz
      c3=fod(i)*do(i)*uo(i)**2.*dt/(2*equd)+9.8*dt
      un(i)=uo(i)-c1-c2-c3
      go1=un(i)*do(i)
      call frici(x(i),po(i),go1,do(i),fod(i),ieror)
       if(ieror.eq.1) go to 100
   40 continue
      do 50 i=2,nosn1
      da=(do(i-1)+do(i))/2.
      csqa=((csq(i-1)**.5+csq(i)**.5)/2.)**2.
```

```
      rha=rh(i-1)
      c1=un(i-1)*(po(i)-po(i-1))*dtdz
      c2=(da*csqa)*(un(i)-un(i-1))*dtdz
      c3=rha/(da/csqa)*(qipt/area+fod(i-1)*
     1   do(i-1)**2.*un(i-1)**3./(2*equd))*dt
      pn(i)=po(i)-c1-c2-c3
   50 continue
      pn(nosn)=tpolt
      ua=un(nosn1)
      call rph(po(nosn),ho(nosn),do(nosn),x(nosn),rp(nosn),rh(nosn))
      csq(nosn)=1./(rp(nosn)+rh(nosn)/do(nosn))
      if(csq(nosn).le.0.0)write(0,8886)csq(nosn)
 8886 format(1x," csqa11 ",e13.6)
      uan=un(nosn1)
      csqa=((csq(nosn1)**.5+csq(nosn)**.5)/2.)**2.
      rha=rh(nosn1)
      daa=(do(nosn1)+do(nosn))/2.
      c1=(3*pn(nosn)+pn(nosn1)-3*po(nosn)-po(nosn1))/(8*dtdz)
      c2=un(nosn1)*(po(nosn)-po(nosn1))
      c3=(c1+c2)/(daa*csqa)
      c4=rha*dz/daa**2.*(qipt/area+fod(nosn1)*do(nosn1)**2.
     1  *un(nosn1)**3./(2*equd))/2.
      un(nosn)=un(nosn1)-c3-c4
      do 130 i=2,nosn1
      csqa=((csq(i-1)**.5+csq(i)**.5)/2.)**2.
      da=(do(i-1)+do(i))/2.
      rpa=rp(i-1)
      c1=un(i-1)*(ho(i)-ho(i-1))*dtdz
      c2=csqa*(un(i)-un(i-1))*dtdz
      c3=rpa/(da/csqa)*(qipt/area+fod(i-1)
     1  *do(i-1)**2.*un(i-1)**3./(2*equd))*dt
      hn(i)=ho(i)+c3-c1-c2
  130 continue
      csqa=((csq(nosn)**.5+csq(nosn1)**.5)/2.)**2.
      rpa=rp(nosn1)
      daa=(do(nosn1)+do(nosn))/2.
      c1=un(nosn1)*(ho(nosn)-ho(nosn1))*dtdz
      c2=csqa*(un(nosn)-un(nosn1))
     1    *2*dtdz
      c3=rpa/(daa/csqa)*(qipt/area+fod(nosn1)
     1    *do(nosn1)**2.*un(nosn1)**3./(2*equd))*dt
      hn(nosn)=ho(nosn)+c3-c1-c2
  100 continue
      return
      end
```

```
      subroutine densi(p,h,d,x)
c
      call satt(p,hls,hvs)
      if(h.le.hls) go to 10
      if (h.gt.hvs) go to 20
      call lden(p,hls,rol)
      call vden(p,hvs,rov)
      x=(h-hls)/(hvs-hls)
      svl=1./rol
      svv=1./rov
      sv=svl*(1-x)+svv*x
      d=1./sv
      return
   10 x=-1.
      call lden(p,h,rol)
      d=rol
      return
   20 x=1.
      call vden(p,h,rov)
      d=rov
      return
      end
```

```
      subroutine lden(p,h,rol)
      data f11,f12/999.65,4.9737e-7/
      data f21,f22/-2.5847e-10,6.1767e-19/
      data f31,f32/1.2696e-22,-4.9223e-31/
c
      data f41,f42/1488.64,1.3389e-6/
      data f51,f52/1.4695e9,8.85736/
      data f61,f62/3.20372e6,1.20483e-2/
c
      if(h.gt.6.513e5) go to 10
      f1=f11+f12*p
      f2=f21+f22*p
      f3=f31+f32*p
      rol=f1+h*h*(f2+f3*h*h)
      go to 20
   10 f4=f41+f42*p
      f5=f51+f52*p
      f6=f61+f62*p
      rol=f4+f5/(h-f6)
   20 return
      end
```

```
      subroutine vden(p,h,rov)
c
      data g00,g01,g02,g10,g11,g12/-5.1026e-5,1.1208e-10,
     1    -4.4506e5,-1.6893e-10,-3.3980e-17,2.3058e-1/
c
      g0p=g00+g01*p+g02/p
      g1p=g10+g11*p+g12/p
      rov=1./(g0p+g1p*h)
      return
      end
```

```
      subroutine satt(p,hls,hvs)
      data tsc1,tsc2, tsexp  /9.0395, 255.2, 0.223/
      data cps1,cps2, cpsexp  /9.5875e2, .00132334, -0.8566/
c    for hls and hvs
        data  hl0,hl1,hl2,hl3,hl4,hl5/5.7474718e5,2.0920624e-1,
     1    -2.8051070e-8,2.3809828e-15,-1.0042660e-22,1.6586960e-30/
        data  hv0,hv1,hv2,hv3,hv4/2.7396234e6,3.758844e-2,
     1        -7.1639909e-9,4.2002319e-16,-9.8507521e-24 /
c
      hvs = hv0 + p*(hv1 + p*(hv2 + p*(hv3 + p*hv4)))
      hls = hl0 + p*(hl1 + p*(hl2 + p*(hl3 + p*(hl4 + p*hl5))))
      return
      end
```

```
      subroutine rph(p,h,d,x,rp,rh)
c
      data hl11,hl22,hl33,hl44,hl55/2.0920624e-1,-5.610214e-8,
     1  7.142948e-15,-4.0170640e-22,8.293480e-30/
c
      data hv11,hv22,hv33,hv44/3.758844e-2,-14.33279818e-9,
     1    12.6006957e-16,-39.4030084e-24/
c
      data f11,f12/999.65,4.9737e-7/
      data f21,f22/-2.5847e-10,6.1767e-19/
      data f31,f32/1.2696e-22,-4.9223e-31/
c
      data f41,f42/1488.64,1.3389e-6/
      data f51,f52/1.4695e9,8.85736/
      data f61,f62/3.20372e6,1.20483e-2/
c
      data g00,g01,g02,g10,g11,g12/-5.1026e-5,1.1208e-10,
     1     -4.4506e5,-1.6893e-10,-3.3980e-17,2.3058e-1/
c
      if(x.eq.-1.)go to 10
      call satt(p,hls,hvs)
      call lden(p,hls,rol)
      call vden(p,hvs,rov)
      dhldp=hl11+p*(hl22+p*(hl33+p*(hl44+p*hl55)))
      dhvdp=hv11+p*(hv22+p*(hv33+p*hv44))
      dvvdp=(g01-g02/p**2.)+(g11-g12/p**2.)*hvs
      if(hls.gt.6.513e5) go to 20
      f2=f21+f22*p
      f3=f31+f32*p
      dvldp=-(f12+hls*hls*(f22+f32*hls*hls))/rol**2.
      go to 30
   20 f5=f51+f52*p
      f6=f61+f62*p
      dvldp=-(f42+f52/(hls-f6)+f5*f62/(hls-f6)**2.)/rol**2.
   30 continue
      dxdp=-dhldp/(hvs-hls)-(dhvdp-dhldp)*(h-hls)/(hvs-hls)**2.
      dvdp=(1-x)*dvldp+x*dvvdp+(1./rov-1./rol)*dxdp
      dvdh=(1./rov-1./rol)/(hvs-hls)
      rp=-d**2.*dvdp
      rh=-d**2.*dvdh
      return
   10 if(h.gt.6.513e5)go to 40
      f2=f21+f22*p
      f3=f31+f32*p
      rp=(f12+h*h*(f22+f32*h*h))
      rh=(2*h*f2+4*h*h*h*f3)
      go to 50
   40 f5=f51+f52*p
      f6=f61+f62*p
      rh=-f5/(h-f6)**2.
      rp=(f42+f52/(h-f6)+f5*f62/(h-f6)**2.)
   50 continue
      return
      end
```

```
      function pilt(pilt0,polt0,time1)
      cn=400.*time1
      if(cn.gt.79.99)go to 10
      pilt=(pilt0+polt0)/2.+(pilt0-polt0)/2.*(exp(-400*time1))
      go to 20
   10 pilt=(pilt0+polt0)/2.
   20 continue
      return
      end
```

B. Momentum Integral Model

```
c    solution of transient balance equations for single heated channel
c
c    ***   MOMEMTUM INTEGRAL MODEL   ***
c
c          june  1983
c
      dimension hn(21),ho(21),gn(21),go(21),dn(21),do(21),
     1   x(21),    title(20),ddh(21)
      common /data1/  chanl,area,equd,nosn,nots,dz,dt
      read (5,1000)(title(i),i=1,20)
 1000 format (20a4)
      read (5,1010)chanl,area,equd
      read(5,1050)is,isb,io,npp
      read(5,1030)time,nots,nosn
 1050 format(4i5)
      nosn1=nosn-1
      nosn2=nosn-2
 1010 format(3e10.5)
      read(5,1020)gilt0,hilt0,pilt0,polt0,powe0
 1020 format(5e10.5)
 1030 format(e10.6,2i5)
      dz=chanl/(nosn-1)
      dt=time/nots
      if(is.eq.1)go to 130
      do 10 i=1,nosn
      read (5,1040)go(i),ho(i)
   10 continue
 1040 format(2e10.5)
c
c
c    calculate the steady state density distribution
c
      p=(pilt0+polt0)/2
      go to 140
c
c   the initial condition is calculated by sub. init
c
  130 call init(isb,gilt0,hilt0,pilt0,polt0,powe0,
     1     p,ho,do,go,x)
c
  140 write(6,2000)
c
      call densi(p,hilt0,din,xin)
      do 141 i=1,nosn1
      hn(i)=(ho(i)+ho(i+1))/2.
  141 continue
      hn(nosn)=ho(nosn)
      do 142 i=1,nosn
      ho(i)=hn(i)
      call densi(p,ho(i),do(i),x(i))
  142 continue
c    calculate the average of the initial mass flux
c
      sum=0
      do 30 i=2,nosn
      sum=sum+(go(i-1)+go(i))/2.
   30 continue
      gavo=sum/(nosn-1)
      dz=chanl/(nosn-1)
```

```
      dt=time/nots
c
c   calculate the friction term of the integral momemtum equation
c
      call intrgl(p,go,ho,do,din,x,fric)
 2000 format (//," transient solutions of single heated channel by "
     1"momentumum integral model")
      write(6,2010)(title(i),i=1,20)
 2010 format (/,20a4)
      write(6,2020)
 2020 format(/," channel geometry ")
      write (6,2030)chanl,area,equd
 2030 format(1x," chanlnel length=",f6.3," m",
     1/,"  flow area  = ",e13.6," m**2",
     1        /,"  equi  diame = ",f6.3," m")
      write(6,2040)
 2040 format(/," operating condition ")
      hiltw=hilt0/1000
      piltw=pilt0/1.e6
      poltw=polt0/1.e6
      powew=powe0/1000
      write (6,2050)gilt0,hiltw,piltw,poltw,powew
 2050 format(1x," inlet mass flux =",f8.3," kg/m**2.sec",
     1        /,"  inlet enthalpy = ",f9.4," kj/kg",
     1        /,"  inlet pressure = ",f8.4," Mpa",
     1        /,"  outlet pressure = ",f8.4," Mpa",
     1      /,"   power   =   ",f8.4," kw/m")
      write(6,3000)time,dt
 3000 format(/," total transient time = ",e13.6," sec",
     1    /,"  time step size = ",e13.6," sec")
      write(6,3010)
 3010 format(/,"  initial condition for transient calculation ")
      write(6,2070)
      do 240 i=1,nosn
      hw=ho(i)/1000
      write(6,2080)i,go(i),hw,do(i),x(i)
  240 continue
c
      np=0
      do 40 i=1,nots
      np=np+1
 9000 format(1x,"  time step ",i5)
      ttime=dt*i
      time1=ttime-dt
      qipt=power(powe0,time1)
c
c   the power should be evaluate at time=i
c
c calculate the enthalpy of the new time stop i.e.  i+1
c
      hin=hilt0
      c1=dt/(dz*do(1))
      c2=(ho(2)-ho(1))/2.*go(2)-(hin-ho(1))*go(1)
      c3=qipt*dt/(area*do(1))
      hn(1)=ho(1)-c1*c2+c3
      do 50 i1=2,nosn2
      c1=dt/(dz*do(i1))
      c2=(ho(i1+1)-ho(i1))/2.*go(i1+1)-(ho(i1-1)-ho(i1))/2.*go(i1)
      c3=qipt*dt/(area*do(i1))
      hn(i1)=ho(i1)-c1*c2+c3
```

```
   50 continue
      c1=dt/(dz*do(nosn1))
      c2=(ho(nosn)-ho(nosn1))*go(nosn)-(ho(nosn2)-ho(nosn1))/2.
     1  *go(nosn1)
      c3=qipt*dt/(area*do(nosn1))
      hn(nosn1)=ho(nosn1)-c1*c2+c3
      hoh=(ho(nosn)+ho(nosn1))/2.
      call densi(p,hoh,da,xa)
      c1=2*dt/(da*dz)
      c2=ho(nosn)*go(nosn)-ho(nosn1)*(go(nosn)+go(nosn1))/2.
     1    -hoh*(go(nosn)-go(nosn1))/2.
      c3=qipt*dt/(area*da)
      hn(nosn)=-hn(nosn1)+2*hoh-2*c1*c2+2*c3
c
c   calculate the new density distribution from the equation of the state
      call densi(p,hin,din,xin)
c
      do 60 i1=1,nosn
      call densi(p,hn(i1),dn(i1),x(i1))
   60 continue
c
c   the mass flux distribution at new time step
c
      if(io.eq.1)goto 70
c   io.eq.2  flow reduced transient
      tgilt=gilt(gilt0,ttime)
      tpilt=pilt0
      ddh(1)=(dn(1)-din)/(hn(1)-hin)
      do 62 i1=2,nosn
      ddh(i1)=(dn(i1)-dn(i1-1))/(hn(i1)-hn(i1-1))
   62 continue
      call masscal(p,qipt,dn,hn,hin,ddh,tgilt,gn)
c
c   calculate the average of the mass flux
c
      sum=0
      do 80 i1=2,nosn
      sum=sum+(gn(i1-1)+gn(i1))/2.
   80 continue
      gavn=sum/(nosn-1)
      predi=(gavn-gavo)/dt+fric
      tpolt=tpilt+predi
      goto 90
c   io.eq.1   pressure transient
   70 continue
      tpilt=pilt(pilt0,polt0,time1)
      tpolt=polt(polt0,time1)
      predi=tpilt-tpolt
      gavn=gavo+dt/chanl*(predi-fric)
c
calculate the inlet mass flux
c
      ddh(1)=(dn(1)-din)/(hn(1)-hin)
      do 71 i1=2,nosn
      ddh(i1)=(dn(i1)-dn(i1-1))/(hn(i1)-hn(i1-1))
   71 continue
      sum1=0
      sum2=0
      qipt=power(powe0,ttime)
      gamm0=1
```

```
      delt0=0
      c1=ddh(1)/dn(1)
      alpha=(1+(hn(1)-hin)*c1)/(1+(hn(1)-hn(2))/2.*c1)
      belta=-dz*c1*qipt/(area*(1+(hn(1)-hn(2))/2.*c1))
      gamm1=gamm0*alpha
      delt1=delt0*alpha+belta
      sum1=sum1+(gamm1+gamm0)/2.
      sum2=sum2+(delt0+delt1)/2.
      gamm0=gamm1
      delt0=delt1
      do 100 i1=3,nosn1
      c1=ddh(i1-1)/(dn(i1-1)*2)
      alpha=(1+(hn(i1-1)-hn(i1-2))*c1)/(1+(hn(i1-1)-hn(i1))*c1)
      belta=-2*dz*c1*qipt/(area*(1+(hn(i1-1)-hn(i1))*c1))
      gamm1=alpha*gamm0
      delt1=alpha*delt0+belta
      sum1=sum1+(gamm1+gamm0)/2.
      sum2=sum2+(delt1+delt0)/2.
      gamm0=gamm1
      delt0=delt1
  100 continue
      c1=ddh(nosn1)/dn(nosn1)
      alpha=(1+(hn(nosn1)-hn(nosn2))/2.*c1)/(1+(hn(nosn1)-hn(nosn)
     1  )*c1)
      belta=-dz*c1*qipt/(area*(1+(hn(nosn1)-hn(nosn))*c1))
      gamm1=alpha*gamm0
      delt1=delt0*alpha+belta
      sum1=sum1+(gamm1+gamm0)/2.
      sum2=sum2+(delt0+delt1)/2.
      ave1=sum1/(nosn-1)
      ave2=sum2/(nosn-1)
      tgilt =(gavn-ave2)/ave1
 9900 format(1x,"gavn,ave2,ave1",3(2x,e10.5),2x,i5)
      call masscal(p,qipt,dn,hn,hin,ddh,tgilt,gn)
c
c   calculate the friction term of the integral momemtum equation
c
   90 continue
      do 91 i1=1,nosn
      if(gn(i1).lt.0.0)go to 45
   91 continue
      call intrgl(p,gn,hn,dn,din,x,fric)
      do 110 i1=1,nosn
      ho(i1)=hn(i1)
      go(i1)=gn(i1)
      do(i1)=dn(i1)
  110 continue
      p=(tpilt+tpolt)/2.
      gavo=gavn
      if(np.ne.npp)go to 41
      if(gn(1).lt.0.0)write(0,9999)gn(1)
 9999 format(1x," something wrong gn(1)=",e13.6)
      write(0,9000)i
      np=0
      predw=predi/1000
      qiptw=qipt/1000
      write (6,2060)ttime,predw,qiptw,gavn
      write(6,2070)
      do 120 i1=1,nosn
      hw=hn(i1)/1000
```

```
      gr=gn(i1)/gilt0
      write(6,2080)i1,gn(i1),hw,dn(i1),x(i1),gr
  120 continue
 2060 format(//,"  transient time=",e13.6," sec ",
     1       /,"   pressure drop = ",f8.4," kpa",
     1       /,"    power   =  ",f8.4, " kw/m",
     1      /,"   average mass flux =",f9.4," kg/m**2.sec")
 2070 format(/,"       mass flux     enthalpy     density  quality",
     1             /,"    (kg/m**2.sec)   (kj/kg)     (kg/m**3)")
 2080 format(1x,i3,2x,f8.3,6x,f8.3,4x,f8.3,2x,f6.3,3x,f7.4)
   41 continue
   40 continue
      go to 46

   45 write(0,5000)i,i1,gn(i1)
      write(6,5000)i,i1,gn(i1)
 5000 format(1x," flow rate is negative at time step no"
     1 ,1x,i5,/," at node ",i4,"   gn=",e13.6)
   46 continue
      stop
      end
c
c   *****************************************************
c
c
c      ******************************************
c
```

```
      subroutine intrgl(p,g,h,d,din,xa,fric)
      dimension h(21),g(21),d(21),xa(21)
      common /data1/  chanl,area,equd,nosn,nots,dz,dt
      data conv1,conv2,vics,g1/737.4643,1.4504e-4,91.7e-6,9.8/
      data vics1/19.73e-6/
c
      fric=0
      nosn1=nosn-1
c   acceleration pressure drop
c
      fric=g(nosn)**2/d(nosn)-g(1)**2/din
c
c   thm:  two phase pressure drop multiplier
c
      thm=1
c
c   get the saturation liquid and vapor density rol,rov
c
      do 10 i=1,nosn1
      ga=(g(i+1)+g(i))/2.
c
c   elavation pressure drop
c
      fric=fric+d(i)*g1*dz
      dpc=d(i)
c
c   if x>0  calculate the two phase multiplier
c   eq(5.209) p.230  Lahey and Moody was used
c
      if(xa(i).eq.-1.) goto 20
      if(xa(i).eq.1.) go to 30
      call satt(p,hls,hvs)
      call lden(p,hls,rol)
      call vden(p,hvs,rov)
      dpc=rol
      gtest=ga*conv1/1.0e6
      if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest
     1    -0.000714*p*conv2*gtest
      if(gtest.ge.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest
     1     +0.00028*p/gtest*conv2
      thm=c1*(1.2*(rol/rov-1)*xa(i)**0.824)+1.0
c
c   use Blasius for friction factor
c
   20 fact=0.184*(ga*equd/vics)**(-0.20)
      go to 40
   30 fact=0.184*(ga*equd/vics1)**-.20
   40 fric=fric+thm*fact*ga**2/(2*equd*dpc)*dz
   10 continue
      return
      end
c
c  ***********************************************
c
```

```
      subroutine init(is1,gilt0,hilt0,pilt0,polt0,powe0,
     1    p1,h,d,g,x)
      dimension h(21),d(21),p(21),g(21),x(21)
      common /data1/ chanl,area,equd,nosn,nots,dz,dt
      call densi(pilt0,hilt0,d(1),x(1))
      h(1)=hilt0
      p(1)=pilt0
      g(1)=gilt0
      i=1
      n1=1
   90 do 10 i=2,nosn
      h1=h(i-1)+powe0*dz/(area*gilt0)
      p1=p(i-1)
      call densi(p1,h1,d(i),x(i))
   60 ha=(h1+h(i-1))/2.
      call densi(p1,ha,da,x1)
      call frici (x1,p1,gilt0,da,fod)
      p2=p(i-1)-fod*gilt0**2.*dz/(2*equd)
     1  -da*9.80*dz-gilt0**2.*(1/d(i)-1/d(i-1))
      call densi(p2,h1,d(i),x(i))
      call densi(p2,ha,da,x1)
      call frici(x1,p2,gilt0,da,fod)
   31 h1=h(i-1)+powe0*dz/(area*gilt0)+(p2-p(i-1)+
     1   (fod*gilt0**2*dz)/(2*equd))/da
      if(abs((p2-p1)/p2).le.1.0e-8)go to 40
      n=n+1
      if(n.gt.50)goto 50
      p1=p2
      goto 60
   40 p(i)=p2
      h(i)=h1
      g(i)=gilt0
   10 continue
      if(is1.eq.1)polt0=p(nosn)
      if(abs((p(nosn)-polt0)/polt0).le.1.0e-4)go to 160
      if(n1.gt.50) go to 80
      n1=n1+1
      gilt0=gilt0*((p(nosn)-p(1))/(polt0-pilt0))**0.5
      goto 90
   80 write(6,2110)
 2110 format(1x," iteration not converge for pressure b.c.")
      stop
   50 write(6,2090)i
 2090 format(1x,"iteration not converge for spatial node ",i2)
      stop
  160 p1=(polt0+pilt0)/2.
      return
      end
```

```
      subroutine frici(x1,p,g,d,fod)
      common /data1/ chanl,area,equd,nosn,nots,dz,dt
      data conv1,conv2,vics/737.4643,1.4504e-4,91.7e-6/

c   use McAdames correlation to calculate the friction factor

      fric=0.184*(g*equd/vics)**(-0.2)
      if(x1.eq.-1.0)go to 10
      call satt(p,hls,hvs)
      call lden(p,hls,rol)
      call vden(p,hvs,rov)
      gtest=g*conv1/1.0e6
      if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest
     1    -0.000714*p*conv2*gtest
      if(gtest.gt.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest
     1+0.00028*p/gtest*conv2
      thm=c1*(1.2*(rol/rov-1)*x1**0.824)+1.0
      fod=thm*fric/rol
      go to 21
   10 fod=fric/d
   21 continue
      return
      end
```

```
function power(powe0,time1)
power=powe0
return
end
```

```
      function pilt(pilt0,polt0,time1)
      cn=400.*time1
      if(cn.gt.79.99)go to 10
      pilt=(pilt0+polt0)/2.+(pilt0-polt0)/2.*(exp(-400*time1))
      go to 20
   10 pilt=(pilt0+polt0)/2.
   20 continue
      return
      end
```

```
function polt(polt0,time1)
polt=polt0
return
end
```

```
      function gilt(gilt0,time1)
      gilt=gilt0
      return
      end
c
```

```fortran
      subroutine densi(p,h,d,x)
c
      call satt(p,hls,hvs)
      if(h.le.hls) go to 10
      if(h.gt.hvs) go to 20
      call lden(p,hls,rol)
      call vden(p,hvs,rov)
      x=(h-hls)/(hvs-hls)
      svl=1./rol
      svv=1./rov
      sv=svl*(1-x)+svv*x
      d=1./sv
      return
   10 x=-1.
      call lden(p,h,rol)
      d=rol
      return
   20 x=1.0
      call vden(p,hvs,rov)
      d=rov
      return
      end
```

```
      subroutine lden(p,h,rol)
      data f11,f12/999.65,4.9737e-7/
      data f21,f22/-2.5847e-10,6.1767e-19/
      data f31,f32/1.2696e-22,-4.9223e-31/
c
      data f41,f42/1488.64,1.3389e-6/
      data f51,f52/1.4695e9,8.85736/
      data f61,f62/3.20372e6,1.20483e-2/
c
      if(h.gt.6.513e5) go to 10
      f1=f11+f12*p
      f2=f21+f22*p
      f3=f31+f32*p
      rol=f1+h*h*(f2+f3*h*h)
      go to 20
   10 f4=f41+f42*p
      f5=f51+f52*p
      f6=f61+f62*p
      rol=f4+f5/(h-f6)
   20 return
      end
```

```
      subroutine vden(p,h,rov)
c
      data g00,g01,g02,g10,g11,g12/-5.1026e-5,1.1208e-10,
     1    -4.4506e5,-1.6893e-10,-3.3980e-17,2.3058e-1/
c
      g0p=g00+g01*p+g02/p
      g1p=g10+g11*p+g12/p
      rov=1./(g0p+g1p*h)
      return
      end
```

```
      subroutine satt(p,hls,hvs)
      data tsc1,tsc2, tsexp  /9.0395, 255.2, 0.223/
      data cps1,cps2, cpsexp  /9.5875e2, .00132334, -0.8566/
c       cps2 = -cpsexp * tcrinv
c    for hls and hvs
       data  hl0,hl1,hl2,hl3,hl4,hl5/5.747471 8e5,2.0920624e-1,
     1    -2.8051070e-8,2.3809828e-15,-1.0042660e-22,1.6586960e-30/
       data  hv0,hv1,hv2,hv3,hv4/2.7396234e6,3.758844e-2,
     1        -7.1639909e-9,4.2002319e-16,-9.8507521e-24 /
c
      tsat = tsc1* p**tsexp
      hvs = hv0 + p*(hv1 + p*(hv2 + p*(hv3 + p*hv4)))
      hls = hl0 + p*(hl1 + p*(hl2 + p*(hl3 + p*(hl4 + p*hl5))))
      pinv = 1./ p
      dtsdp = tsat*tsexp*pinv
      tsat = tsat + tsc2
      return
      end
```

```
      subroutine masscal(p,qipt,dn,hn,hin,ddh,tgilt,gn)
      dimension dn(21),hn(21),gn(21),ddh(21)
      common /data1/   chanl,area,equd,nosn,nots,dz,dt
      nosn1=nosn-1
      nosn2=nosn-2
      gn(1)=tgilt
      c1=ddh(1)/dn(1)
      belta=-dz*c1*qipt/(area*(1+(hn(1)-hn(2))/2.*c1))
      alpha=(1+(hn(1)-hin)*c1)/(1+(hn(1)-hn(2))/2.*c1)
      gn(2)=gn(1)*alpha+belta
      do 100 i1=3,nosn1
      c1=ddh(i1-1)/(dn(i1-1)*2)
      alpha=(1+(hn(i1-1)-hn(i1-2))*c1)/(1+(hn(i1-1)-hn(i1))*c1)
      belta=-2*dz*c1*qipt/(area*(1+(hn(i1-1)-hn(i1))*c1))
      gn(i1)=gn(i1-1)*alpha+belta
  100 continue
      c1=ddh(nosn1)/dn(nosn1)
      alpha=(1+(hn(nosn1)-hn(nosn2))/2.*c1)/(1+(hn(nosn1)-hn(nosn)
     1  )*c1)
      belta=-dz*c1*qipt/(area*(1+(hn(nosn1)-hn(nosn))*c1))
      gn(nosn)=alpha*gn(nosn1)+belta
      return
      end
```

C. Single Velocity Model

```
c   solution of transient balance equations for single heated channel
c
c   ***   SINGLE VELOCITY MODEL   ***
c
c         june  1983
c
      dimension hn(21),ho(21), dn(21),do(21),
     1   x(21),xa(21),da(21),    title(20)
      common /data1/  chanl,area,equd,nosn,nots,dz,dt
      read (5,1000)(title(i),i=1,20)
 1000 format (20a4)
      read (5,1010)chanl,area,equd
      read(5,1050)is,isb,io,npp
      read(5,1030)time,nots,nosn
 1050 format(4i5)
 1010 format(3e10.5)
      read(5,1020)gilt0,hilt0,pilt0,polt0,powe0
 1020 format(5e10.5)
 1030 format(e10.6,2i5)
      dz=chanl/(nosn-1)
      dt=time/nots
      if(is.eq.1)go to 130
      do 10 i=1,nosn
      read (5,1040)ho(i)
   10 continue
 1040 format(e10.5)
c
c
c
      p=(pilt0+polt0)/2
      go to 140
c
c  the initial condition is calculated by sub. init
c
  130 call init(isb,gilt0,hilt0,pilt0,polt0,powe0,
     1     p,ho,do,x)
c
  140 write(6,2000)
c
c      calculate the steady state density distribution
c
      do 20 i=1,nosn
      call densi(p,ho(i),do(i),x(i))
   20 continue
c
      do 141 i=2,nosn
      hoh=(ho(i)+ho(i-1))/2.
      call densi(p,hoh,da(i),xa(i))
  141 continue
      go=gilt0
      call intrgl(p,go,ho,do,da,xa,fric)
 2000 format (1h1," transient solutions of single heated channel "
     1    ," by single velocity model ")
      write(6,2010)(title(i),i=1,20)
 2010 format (/,20a4)
      write(6,2020)
 2020 format(/," channel geometry ")
      write (6,2030)chanl,area,equd
 2030 format(1x," channel length=",f6.3," m",
```

```
     1/," flow area  = ",e13.6," m**2",
     1       /,"  equi  diame = ",f6.3," m")
      write(6,2040)
 2040 format(/," operating condition ")
      hiltw=hilt0/1000
      piltw=pilt0/1000000
      poltw=polt0/1.e6
      powew=powe0/1000
      write (6,2050)gilt0,hiltw,piltw,poltw,powew
 2050 format(1x,"  inlet mass flux = ",f9.3," kg/m**2.sec",
     1       /,"  inlet enthalpy = ",f9.3," kj/kg",
     1       /,"  inlet pressure = ",f8.4," Mpa",
     1       /,"  outlet pressure = ",f8.4," Mpa",
     1    /,"   power  =  ",f8.4," kw/m")
      write(6,3200)time,dt

 3200 format(/," total transient= ",e13.6," sec",
     1       /" time step size = ",e13.6," sec")
      write(6,3000)
 3000 format(/,"  initial condition for transient calculation ")
      write(6,2070)
      do 200 i=1,nosn
      hw=ho(i)/1000
      write(6,2080)i,hw,do(i),x(i)
  200 continue
c
      np=0
      do 40 i=1 ,nots
 6000 format(1x,"  time step ",i5)
      np=np+1
      ttime=dt*i
      time1=ttime-dt
      qipt=power(powe0,time1)
c
c   the power should be evaluate at time=i
c
c calculate the enthalpy of the new time step i.e.  i+1
c
      hn(1)=hilt0
      do 50 i1=2,nosn
      alpha=go*dt/(da(i1)*dz)
      belta=(2*dt*qipt/area)/(da(i1)*(1+alpha))
      hn(i1)=ho(i1-1)-(hn(i1-1)-ho(i1))*((1-alpha)/(1+alpha))+belta
   50 continue
c
c   calculate the new density distribution from the equation of the state
c
      do 60 i1=1,nosn
      call densi(p,hn(i1),dn(i1),x(i1))
   60 continue
      do 61 i1=2,nosn
      hnh=(hn(i1)+hn(i1-1))/2.
      call densi(p,hnh,da(i1),xa(i1))
   61 continue
c
c   the mass flux distribution at new time step
c
       if(io.eq.1)goto 70
c   io.eq.2  flow reduced transient
      gn=gilt(gilt0,ttime)
```

```
      tpilt=pilt0
      predi=(gn-go)/dt+fric
      tpolt=tpilt+predi
      goto 90
c   io.eq.1   pressure transient
   70 continue
      tpilt=pilt(pilt0,polt0,time1)
      tpolt=polt(polt0,time1)
      predi=tpilt-tpolt
      gn=go+dt/chanl*(predi-fric)
c
c   calculate the friction term of the integral momemtum equation
c
   90 continue
      call intrgl(p,gn,hn,dn,da,xa,fric)
      do 110 i1=1,nosn
      ho(i1)=hn(i1)
      do(i1)=dn(i1)
  110 continue
      go=gn
      if(np.ne.npp)go to 40
c     write(0,6000)i
      np=0
      predw=predi/1000
      qiptw=qipt/1000
      gnr=gn/gilt0
      write (6,2060)ttime,predw,qiptw,gn,gnr
      write(6,2070)
      do 120 i1=1,nosn
      hw=hn(i1)/1000
      write(6,2080)i1,hw,dn(i1),x(i1)
  120 continue
 2060 format(//," transient time=",f8.4," sec ",
     1       /," pressure drop = ",f8.4," kpa",
     1       /,"    power   =  ",f8.4, " kw/m",
     1         /,"  mass flux  = ",f9.3,"  kg/m**2.sec",3x,f8.4)

 2070 format(/,"       enthalpy     density  ", "   quality ",
     1        /   ,"        (kj/kg)    (kg/m**3)  ")
 2080 format(1x,i2,2x,f9.3,5x,f8.4,4x,f6.3)
   41 continue
   40 continue
      end
```

```
      subroutine intrgl(p,g,h,d,da,xa,fric)
      dimension h(21),d(21),xa(21),da(21)
      common /data1/  chanl,area,equd,nosn,nots,dz,dt
      data conv1,conv2,vics,g1/737.4643,1.4504e-4,91.7e-6,9.8/
      data vics1/19.73e-6/
c
      fric=0
c       acceleration pressure drop
c
      fric=g**2/d(nosn)-g**2/d(1)
c
c   thm:  two phase pressure drop multiplier
c
      thm=1
c
c   get the saturation liquid and vapor density rol,rov
c
      call satt(p,hls,hvs)
      call lden(p,hls,rol)
      call vden(p,hvs,rov)
      do 10 i=2,nosn
c
c   elavation pressure drop
c
      fric=fric+da(i)*g1*dz
      dpc=da(i)
c
c   if x>0  calculate the two phase multiplier
c   eq(5.209) p.230  Lahey and Moody was used
c
      if(xa(i).eq.-1) goto 20
      if(xa(i).eq.1.0) go to 30
      dpc=rol
      gtest=g*conv1/1.0e6
      if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest
     1     -0.000714*p*conv2*gtest
      if(gtest.ge.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest
     1     +0.00028*p/gtest*conv2
      thm=c1*(1.2*(rol/rov-1)*xa(21)**0.824)+1.0
c
c
   20 fact=0.184*(g*equd/vics)**(-0.20)
      go to 40
   30 fact=0.184*(g*equd/vics1)**(-0.20)
   40 fric=fric+thm*fact*g**2/(2*equd*dpc)*dz
   10 continue
      return
      end
```

```
      subroutine init(is1,gilt0,hilt0,pilt0,polt0,powe0,
     1    p1,h,d,x)
      dimension h(21),d(21),p(21),g(21),x(21)
      common /data1/ chanl,area,equd,nosn,nots,dz,dt
      tsub=600
      call densi(pilt0,hilt0,d(1),x(1))
      h(1)=hilt0
      p(1)=pilt0
      g(1)=gilt0
      i=1
      n1=1
   90 do 10 i=2,nosn
      h1=h(i-1)+powe0*dz/(area*gilt0)
      p1=p(i-1)
      call densi(p1,h1,d(i),x(i))
   60 ha=(h1+h(i-1))/2.
      call densi(p1,ha,da,x1)
      call frict (x1,p1,gilt0,da,fod)
      p2=p(i-1)-fod*gilt0**2.*dz/(2*equd)
     1  -da*9.80*dz-gilt0**2.*(1/d(i)-1/d(i-1))
      call densi(p2,h1,d(i),x(i))
      call densi(p2,ha,da,x1)
      call frict(x1,p2,gilt0,da,fod)
   31 h1=h(i-1)+powe0*dz/(area*gilt0)+(p2-p(i-1)+
     1   (fod*gilt0**2*dz)/(2*equd))/da
      if(abs((p2-p1)/p2).le.1.0e-8)go to 40
      n=n+1
      if(n.gt.50)goto 50
      p1=p2
      goto 60
   40 p(i)=p2
      h(i)=h1
      g(i)=gilt0
   10 continue
      if(is1.eq.1)polt0=p(nosn)
      if(abs((p(nosn)-polt0)/polt0).le.1.0e-4)go to 160
      if(n1.gt.50) go to 80
      n1=n1+1
      gilt0=gilt0*((p(nosn)-p(1))/(polt0-pilt0))**0.5
      goto 90
   80 write(6,2110)
 2110 format(1x," iteration not converge for pressure b.c.")
      stop
   50 write(6,2090)i
 2090 format(1x,"iteration not converge for spatial node ",i2)
      stop
  160 p1=(polt0+pilt0)/2.
      return
      end
```

```
      subroutine frici(x1,p,g,d,fod)
      common /data1/ chanl,area,equd,nosn,nots,dz,dt
      data conv1,conv2,vics/737.4643e0,1.4504e-4,91.7e-6/

c   use McAdames correlation to calculate the friction factor

      fric=0.184*(g*equd/vics)**(-0.2)
      if(x1.eq.-1.0)go to 10
      call satt(p,hls,hvs)
      call lden(p,hls,rol)
      call vden(p,hvs,rov)
      gtest=g*conv1/1.0e6
      if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest
     1   -0.000714*p*conv2*gtest
      if(gtest.gt.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest
     1+0.00028*p/gtest*conv2
      thm=c1*(1.2*(rol/rov-1)*x1**0.824)+1.0
      fod=thm*fric/rol
      go to 21
   10 fod=fric/d
   21 continue
      return
      end
```

```
function power(powe0,time1)
power=powe0
return
end
```

```
      function polt(polt0,time1)
      polt=polt0
      return
      end
```

```
      function gilt(gilt0,time1)
      gilt=gilt0
      return
      end
c
```

```
      subroutine densi(p,h,d,x)
c
      call satt(p,hls,hvs)
      if(h.le.hls) go to 10
      if(h.gt.hvs) go to 20
      call lden(p,hls,rol)
      call vden(p,hvs,rov)
      x=(h-hls)/(hvs-hls)
      svl=1./rol
      svv=1./rov
      sv=svl*(1-x)+svv*x
      d=1./sv
      return
   10 x=-1.
      call lden(p,h,rol)
      d=rol
      return
   20 x=1.0
      call vden(p,hvs,rov)
      d=rov
      return
      end
```

```
      subroutine lden(p,h,rol)
      data f11,f12/999.65,4.9737e-7/
      data f21,f22/-2.5847e-10,6.1767e-19/
      data f31,f32/1.2696e-22,-4.9223e-31/
c
      data f41,f42/1488.64,1.3389e-6/
      data f51,f52/1.4695e9,8.85736/
      data f61,f62/3.20372e6,1.20483e-2/
c
      if(h.gt.6.513e5) go to 10
      f1=f11+f12*p
      f2=f21+f22*p
      f3=f31+f32*p
      rol=f1+h*h*(f2+f3*h*h)
      go to 20
   10 f4=f41+f42*p
      f5=f51+f52*p
      f6=f61+f62*p
      rol=f4+f5/(h-f6)
   20 return
      end
```

```
      subroutine vden(p,h,rov)
c
      data g00,g01,g02,g10,g11,g12/-5.1026e-5,1.1208e-10,
     1    -4.4506e5,-1.6893e-10,-3.3980e-17,2.3058e-1/
c
      g0p=g00+g01*p+g02/p
      g1p=g10+g11*p+g12/p
      rov=1./(g0p+g1p*h)
      return
      end
```

```
      subroutine satt(p,hls,hvs)
      data tsc1,tsc2, tsexp  /9.0395, 255.2, 0.223/
      data cps1,cps2, cpsexp  /9.5875e2, .00132334, -0.8566/
c       cps2 = -cpsexp * tcrinv
c    for hls and hvs
        data  hl0,hl1,hl2,hl3,hl4,hl5/5.7474718e5,2.0920624e-1,
     1     -2.8051070e-8,2.3809828e-15,-1.0042660e-22,1.6586960e-30/
        data  hv0,hv1,hv2,hv3,hv4/2.7396234e6,3.758844e-2,
     1        -7.1639909e-9,4.2002319e-16,-9.8507521e-24 /
c
      tsat = tsc1* p**tsexp
      hvs = hv0 + p*(hv1 + p*(hv2 + p*(hv3 + p*hv4)))
      hls = hl0 + p*(hl1 + p*(hl2 + p*(hl3 + p*(hl4 + p*hl5))))
      pinv = 1./ p
      dtsdp = tsat*tsexp*pinv
      tsat = tsat + tsc2
      return
      end
```

```
      function pilt(pilt0,polt0,time1)
      cn=400.*time1
      if(cn.gt.79.99)go to 10
      pilt=(pilt0+polt0)/2.+(pilt0-polt0)/2.*(exp(-400*time1))
      go to 20
   10 pilt=(pilt0+polt0)/2.
   20 continue
      return
      end
```

D. Channel Integral Model

```
c   solution of transient balance equations for single heated channel
c
c   ***   CHANNEL INTEGRAL MODEL   ***
c
c         june  1983
c
      dimension hn(21),ho(21),gn(21),go(21),dn(21),do(21),
     1   xn(21),xo(21),hn1(21),dn1(21),gamma(21),beta(21),title(20)
      common /data1/  chanl,area,equd,nosn,nots,dz,dt
      read (5,1000)(title(i),i=1,20)
 1000 format (20a4)
      read (5,1010)chanl,area,equd
      read(5,1050)is,isb,io,npp
      read(5,1030)time,nots,nosn
 1050 format(4i5)
 1010 format(3e10.5)
      read(5,1020)gilt0,hilt0,pilt0,polt0,powe0
 1020 format(5e10.5)
 1030 format(e10.6,2i5)
      dz=chanl/(nosn-1)
      dt=time/nots
      if(is.eq.1)go to 130
      do 10 i=1,nosn
      read (5,1040)go(i),ho(i)
   10 continue
 1040 format(2e10.5)
c
c
      p=(pilt0+polt0)/2.
      go to 140
c
c  the initial condition is calculated by sub. init
c
  130 call init(isb,gilt0,hilt0,pilt0,polt0,powe0,
     1     p,ho,do,go,xo)
c
c
c
c      calculate the steady state density distribution
c
      p=(pilt0+polt0)/2.
      do 20 i=1,nosn
      call densi(p,ho(i),do(i),xo(i))
   20 continue
c
  140 write(6,2000)
c
c   calculate the average of the initial mass flux
c
      sum=0
      do 30 i=2,nosn
      sum=sum+(go(i-1)+go(i))/2.
   30 continue
      gavo=sum/(nosn-1)
      nosn1=nosn-1
      sum=ho(1)/2.
      do 40 j=2,nosn1
      sum=sum+ho(j)
   40 continue
```

```
      sum=sum+ho(nosn)/2
      havo=sum/nosn1
      deltho=ho(nosn)-hilt0
      do 50 j=1,nosn
      beta(j)=(ho(j)-hilt0)/(havo-hilt0)
   50 continue
      dz=chanl/(nosn-1)
      dt=time/nots
c
c
 2000 format (//," transient solutions of single heated channel by ",
     1"channel integral model")
      write(6,2010)(title(i),i=1,20)
 2010 format (/,20a4)
      write(6,2020)
 2020 format(/," channel geometry ")
      write (6,2030)chanl,area,equd
 2030 format(1x," chanlnel length=",f6.3," m",
     1/,"  flow area  = ",e13.6," m**2",
     1      /,"  equi  diame = ",f6.3," m")
      write(6,2040)
 2040 format(/," operating condition ")
      hiltw=hilt0/1000
      piltw=pilt0/1.e6
      poltw=polt0/1.e6
      powew=powe0/1000
      write (6,2050)gilt0,hiltw,piltw,poltw,powew
 2050 format(1x," inlet mass flux =",f8.3," kg/m**2.sec",
     1      /,"  inlet enthalpy = ",f9.4," kj/kg",
     1      /,"  inlet pressure = ",f8.4," Mpa",
     1      /,"  outlet pressure = ",e13.6," Mpa",
     1    /,"   power  =  ",f8.4," kw/m")
      write(6,3000)time,dt
 3000 format(/," total transient time = ",e13.6," sec",
     1    /,"  time step size = ",e13.6," sec")
      write(6,3010)
 3010 format(/,"  initial condition for transient calculation ")
      write(6,3200)
      do 240 i=1,nosn
      hw=ho(i)/1000
      write(6,3300)i,go(i),hw,do(i),xo(i)
  240 continue
      havn= havo*1.0001
      havn=havo+1.
      do 60 j=1,nosn
      hn(j)=hilt0+beta(j)*(havn-hilt0)
      call densi(p,hn(j),dn(j),xn(j))
   60 continue
      call cst(havo,ho,do,havn,hn,dn,deltho,gamma,hilt0,cst1
     1,cst2,cst3)
c
      np=0
      do 70 i=1,nots
      n=1
      np=np+1
      ttime=dt*i
      time1=ttime-dt
      qipt=power(powe0,time1)
      thilt=hilt0
      if(io.eq.1)go to 80
```

```
c
c     flow reduce transient
c
      tgilt=gilt(gilt0,time1)
      const=cst1/(cst3*gamma(nosn)*chanl)
      gavo=(tgilt-const*qipt*chanl/area)/(1-const*deltho)
   80 havn=havo+dt/(cst3*chanl)*(qipt*chanl/area-gavo*deltho)
      do 110 j=1,nosn
      hn(j)=thilt+beta(j)*(havn-thilt)
      call densi(p,hn(j),dn(j),xn(j))
  110 continue
  180 continue
        call cst(havo,ho,do,havn,hn,dn,deltho,
     1      gamma,thilt,cst1,cst2,cst3)
      if(io.eq.1)go to 81
      const=cst1/(cst3*gamma(nosn)*chanl)
      gavo=(tgilt-const*qipt*chanl/area)/(1-const*deltho)
   81 havn1=havo+dt/(cst3*chanl)*(qipt*chanl/area-gavo*deltho)
      if(abs((havn-havn1)/havn1).le.1.0e-7) go to 90
      n=n+1
      if(n.gt.100)go to 100
      do 112 j=1,nosn
      hn1(j)=thilt+beta(j)*(havn1-thilt)
      call densi(p,hn1(j),dn1(j),xn(j))
  112 continue
      havn=havn1
      do 120 j=1,nosn
      dn(j)=dn1(j)
      hn(j)=hn1(j)
  120 continue
      go to 180
   90 tgilt=gavo+cst1/(cst3*gamma(nosn)*chanl)*(qipt*chanl/area
     1   -gavo*deltho)
      gn(1)=tgilt
      do 190 j=2,nosn
      gn(j)=gn(1)+gamma(j)*(gavo-gn(1))
  190 continue
      if(io.eq.1)go to 150
c
c     flow reduce transient
c
c     for this transient the calculated pressure drop
c        is corresponding to ttime-2dt  hence there is no meaning
c        to calculate it for first time step
c
      if(i.eq.1)go to 151
      time2=time1-dt
      tpolt=polt(polt0,time2)
      do 191 i1=1,nosn
      if(go(i1).lt.0.0)go to 73
  191 continue
      call intrgl(p,go,ho,do,xo,fric)
      predi=(gavo-gavoo)*chanl/dt+fric
      tpilt=tpolt+predi
      p=(tpilt+tpolt)/2.
  151 gavoo=gavo
      go to 160
  150 do 192 i1=1,nosn
      if(gn(i1).lt.0.0)go to 73
  192 continue
```

```
      call intrgl(p,gn,hn,dn,xn,fric)
      tpolt=polt(polt0,time1)
      tpilt=pilt(pilt0,polt0,time1)
      predi=tpilt-tpolt
      p=(tpolt+tpilt)/2.
      gavn=gavo+dt/chanl*(predi-fric)
  160 gavo=gavn
      deltho=hn(nosn)-thilt
      do 170 j=1,nosn
      do(j)=dn(j)
      ho(j)=hn(j)
      xo(j)=xn(j)
  170 continue
      havo=havn
      if(i.gt.1000)npp=10
      if(np.ne.npp)go to 71
      np=0
c     write(0,9000)i
 9000 format(1x," timme steps ",i5)
      write(6,3600)ttime
 3600 format(//," transient time = ",e13.6," sec")
      havw=havn/1000
      qiptw=qipt/1000
      predw=predi/1000
      write(6,3100)gavn,havw,predw,qiptw
 3100 format(/,"  average mass flux = ",f8.3," kg/m**2.sec",
     1     /, "  average enthalpy = ",f8.3," kj/kg",
     1     /,  "  pressure drop  = ",f8.3,"  kpa",
     1     /, "  linear power = ",f8.3," kw/m")
      write(6,3200)
 3200 format(/,"     mass flux   enthalpy   density  quality ",
     1      /,"     kg/m**2.sec   kj/kg    kg/m**3    ")
      do 200 j=1,nosn
      gr=gn(j)/gilt0
      hw=hn(j)/1000
      write(6,3300)j,gn(j),hw,dn(j),xn(j),gr
 3300 format(1x,i3,2x,f8.3,3x,f8.3,3x,f8.4,2x,f7.4,3x,f7.4)
  200 continue
   71 continue
   70 continue
      go to 72
   73 write(0,3500)i,i1
      write(6,3500)i,i1
 3500 format(1x," flow rate is negative at time step ",i5,/,
     1  " for node ",i4)
      go to 72
  100 write(6,3400)i
 3400 format(1x," iteration not converenge for time step",2x,i2)
   72 continue
      end
```

```
      subroutine intrgl(p,g,h,d,x,fric)
      dimension h(21),g(21),d(21),x(21)
      common /data1/  chanl,area,equd,nosn,nots,dz,dt
      fric=0
      data conv1,conv2,vics,g1/737.4643,1.4504e-4,91.7e-6,9.8/
      data vics1/19.73e-6/
c
c   acceleration pressure drop
c
      fric=g(nosn)**2/d(nosn)-g(1)**2/d(1)
c
c   thm:  two phase pressure drop multiplier
c
      thm=1
c
c   get the saturation liquid and vapor density rol,rov
c
      call satt(p,hls,hvs)
      call lden(p,hls,rol)
      call vden(p,hvs,rov)
      do 10 i=2,nosn
      ha=(h(i-1)+h(i))/2.
      call densi(p,ha,da,xa)
      ga=(g(i-1)+g(i))/2.
c
c   elavation pressure drop
c
      fric=fric+da*g1*dz
c
c   if x>0  calculate the two phase multiplier
c   eq(5.209) p.230  Lahey and Moody was used
c
      if(xa.eq.-1) goto 20
      if(xa.eq.1.) go to 30
      da=rol
      gtest=ga*conv1/1.0e6
      if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest
     1     -0.000714*p*conv2*gtest
      if(gtest.ge.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest
     1     +0.00028*p/gtest*conv2
      thm=c1*(1.2*(rol/rov-1)*xa**0.824)+1.0
c
c   use Blasius for friction factor
c
   20 fact=0.184*(ga*equd/vics)**(-0.20)
      go to 40
   30 fact=0.184*(ga*equd/vics1)**(-0.2)
   40 fric=fric+thm*fact*ga**2/(2*equd*da)*dz
   10 continue
      return
      end
c
c  **********************************************
c
```

```
      subroutine init(is1,gilt0,hilt0,pilt0,polt0,powe0,
     1    p1,h,d,g,x)
      dimension h(21),d(21),p(21),g(21),x(21)
      common /data1/ chanl,area,equd,nosn,nots,dz,dt
      call densi(pilt0,hilt0,d(1),x(1))
      h(1)=hilt0
      p(1)=pilt0
      g(1)=gilt0
      i=1
      n1=1
   90 do 10 i=2,nosn
      h1=h(i-1)+powe0*dz/(area*gilt0)
      p1=p(i-1)
      call densi(p1,h1,d(i),x(i))
   60 ha=(h1+h(i-1))/2.
      call densi(p1,ha,da,x1)
      call fric1 (x1,p1,gilt0,da,fod)
      p2=p(i-1)-fod*gilt0**2.*dz/(2*equd)
     1  -da*9.80*dz-gilt0**2.*(1/d(i)-1/d(i-1))
      call densi(p2,h1,d(i),x(i))
      call densi(p2,ha,da,x1)
      call fric1(x1,p2,gilt0,da,fod)
   31 h1=h(i-1)+powe0*dz/(area*gilt0)+(p2-p(i-1)+
     1   (fod*gilt0**2*dz)/(2*equd))/da
      if(abs((p2-p1)/p2).le.1.0e-8)go to 40
      n=n+1
      if(n.gt.50)goto 50
      p1=p2
      goto 60
   40 p(i)=p2
      h(i)=h1
      g(i)=gilt0
   10 continue
      if(is1.eq.1)polt0=p(nosn)
      if(abs((p(nosn)-polt0)/polt0).le.1.0e-4)go to 160
      if(n1.gt.50) go to 80
      n1=n1+1
      gilt0=gilt0*((p(nosn)-p(1))/(polt0-pilt0))**0.5
      goto 90
   80 write(6,2110)
 2110 format(1x," iteration not converge for pressure b.c.")
      stop
   50 write(6,2090)i
 2090 format(1x,"iteration not converge for spatial node ",i2)
      stop
  160 p1=(polt0+pilt0)/2.
      return
      end
```

```
      subroutine fric1(x1,p,g,d,fod)
      common /data1/ chanl,area,equd,nosn,nots,dz,dt
      data conv1,conv2,vics/737.4643,1.4504e-4,91.7e-6/

c   use McAdames correlation to calculate the friction factor

      fric=0.184*(g*equd/vics)**(-0.2)
      call satt(p,hls,hvs)
      call lden(p,hls,rol)
      call vden(p,hvs,rov)
      if(x1.eq.-1.0)go to 10
      gtest=g*conv1/1.0e6
      if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest
     1   -0.000714*p*conv2*gtest
      if(gtest.gt.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest
     1+0.00028*p/gtest*conv2
      thm=c1*(1.2*(rol/rov-1)*x1**0.824)+1.0
      fod=thm*fric/rol
      go to 21
   10 fod=fric/d
   21 continue
      return
      end
```

```
      subroutine cst(havo,ho,do,havn1,hn1,dn1,deltho,gamma,
     1          hilt0,cst1,cst2,cst3)
      dimension ho(21),hn1(21),do(21),dn1(21),b(21),gamma(21)
      common/data1/  chanl,area,equd,nosn,nots,dz,dt
      nosn1=nosn-1
      smtot=do(1)/2.
      do 30 j=2,nosn1
      smtot=smtot+do(j)
   30 continue
      smtot=smtot+do(nosn)/2.
      smtot=smtot*chanl/nosn1
      etot=0
      do 40 j=2,nosn
      et=1/2.*(do(j-1)+do(j))*((ho(j-1)+ho(j))/2.-hilt0)
      etot=etot+et
   40 continue
      etot=etot*chanl/nosn1
      smtt1=1/2.*dn1(1)
      do 60 j=2,nosn1
      smtt1=smtt1+dn1(j)
   60 continue
      smtt1=smtt1+1/2.*dn1(nosn)
      smtt1=smtt1*chanl/nosn1
      etot1=0
      do 70 j=2,nosn
      et=1/2.*(dn1(j-1)+dn1(j))*((hn1(j)+hn1(j-1))/2.-hilt0)
      etot1=etot1+et
   70 continue
      etot1=etot1*chanl/nosn1
      cst1=(smtt1-smtot)/(havn1-havo)
      cst2=(etot1-etot)/(havn1-havo)
      b(1)=0
      do 80 j=2,nosn
      b(j)=dn1(j)-do(j)+dn1(j-1)-do(j-1)+b(j-1)
   80 continue
      btot=0
      btot=1/2.*b(1)
      do 90 j=2,nosn1
      btot=btot+b(j)
   90 continue
      btot=btot+1/2.*b(nosn)
      bave=btot/nosn1
      do 100 j=1,nosn
      gamma(j)=b(j)/bave
  100 continue
      cst3=(gamma(nosn)*cst2-cst1*(gamma(nosn)-1)*deltho)/
     1(gamma(nosn)*chanl)
      return
      end
c
```

```
      subroutine densi(p,h,d,x)
c
      call satt(p,hls,hvs)
      if(h.le.hls) go to 10
      if(h.gt.hvs) go to 20
      call lden(p,hls,rol)
      call vden(p,hvs,rov)
      x=(h-hls)/(hvs-hls)
      svl=1./rol
      svv=1./rov
      sv=svl*(1-x)+svv*x
      d=1./sv
      return
   10 x=-1.
      call lden(p,h,rol)
      d=rol
      return
   20 x=1.
      call vden(p,h,rov)
      d=rov
      return
      end
```

```
      subroutine lden(p,h,rol)
      data f11,f12/999.65,4.9737e-7/
      data f21,f22/-2.5847e-10,6.1767e-19/
      data f31,f32/1.2696e-22,-4.9223e-31/
c
      data f41,f42/1488.64,1.3389e-6/
      data f51,f52/1.4695e9,8.85736/
      data f61,f62/3.20372e6,1.20483e-2/
c
      if(h.gt.6.513e5) go to 10
      f1=f11+f12*p
      f2=f21+f22*p
      f3=f31+f32*p
      rol=f1+h*h*(f2+f3*h*h)
      go to 20
   10 f4=f41+f42*p
      f5=f51+f52*p
      f6=f61+f62*p
      rol=f4+f5/(h-f6)
   20 return
      end
```

```
      subroutine vden(p,h,rov)
c
      data g00,g01,g02,g10,g11,g12/-5.1026e-5,1.1208e-10,
     1    -4.4506e5,-1.6893e-10,-3.3980e-17,2.3058e-1/
c
      g0p=g00+g01*p+g02/p
      g1p=g10+g11*p+g12/p
      rov=1./(g0p+g1p*h)
      return
      end
```

```
      subroutine satt(p,hls,hvs)
      data tsc1,tsc2, tsexp  /9.0395, 255.2, 0.223/
      data cps1,cps2, cpsexp  /9.5875e2, .00132334, -0.8566/
c       cps2 = -cpsexp * tcrinv
c     for hls and hvs
        data  hl0,hl1,hl2,hl3,hl4,hl5/5.7474718e5,2.0920624e-1,
     1     -2.8051070e-8,2.3809828e-15,-1.0042660e-22,1.6586960e-30/
        data  hv0,hv1,hv2,hv3,hv4/2.7396234e6,3.758844e-2,
     1          -7.1639909e-9,4.2002319e-16,-9.8507521e-24 /
c
      tsat = tsc1* p**tsexp
      hvs = hv0 + p*(hv1 + p*(hv2 + p*(hv3 + p*hv4)))
      hls = hl0 + p*(hl1 + p*(hl2 + p*(hl3 + p*(hl4 + p*hl5))))
      pinv = 1./ p
      dtsdp = tsat*tsexp*pinv
      tsat = tsat + tsc2
      return
      end
```

```
function power(powe0,time1)
power=powe0
return
end
```

```
function polt(polt0,time1)
polt=polt0
return
end
```

```
      function gilt(gilt0,timei)
      gilt=gilt0
      return
      end
c
```

```
      function pilt(pilt0,polt0,time1)
      cn=100.*time1
      if(cn.gt.79.99)go to 10
      pilt=(pilt0+polt0)/2.+(pilt0-polt0)/2.*(exp(-100*time1))
      go to 20
   10 pilt=(pilt0+polt0)/2.
   20 continue
      return
      end
```