

NUCLEAR ENGINEERING
READING ROOM - M.I.T.

Massachusetts Institute of Technology

Department of Nuclear Engineering

TRANSIENT RESPONSE OF A SINGLE HEATED CHANNEL

by

Min Lee and M. S. Kazimi



NUCLEAR ENGINEERING
READING ROOM - M.I.T.

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
ϕ_{lo}^2	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

	<u>Page</u>
Abstract	i
Nomenclature	ii
Table of Contents	iii
List of Figures	v
List of Tables	vi
1. Introduction	1
2. Sectionalized Compressible Flow Model	3
2.1 Differential Equation	3
2.2 Finite Difference Equation	5
2.3 Results	6
3. Momentum Integral Model	7
3.1 Differential Equation	7
3.2 Finite Difference Equation	9
3.3 Results	10
4. Single Mass Velocity Model	11
4.1 Differential Equations	12
4.2 Finite Difference Equations	13
4.3 Results	13
5. Channel Integral Model	14
5.1 Differential Equations	14
5.2 Finite Difference Equations	15
5.3 Results	17
6. Conclusion	17
7. References	20

Table of Contents (continued)

	<u>Page</u>
Appendix I.A General Information and Input Manual of Computer Codes.....	36
Appendix I.B Input Manual	37
Appendix II. Computer Codes for Transient Response of a Single Heated Channel	39
Appendix II.A Sectionalized Compressible Flow Model	40
Appendix II.B Momentum Integral Model	59
Appendix II.C Single Velocity Model	77
Appendix II.D Channel Integral Model	92

List of Figures

<u>Figures</u>		<u>Page</u>
1	Sectionalized Compressible Flow Model Calculation Cell	21
2	Sectionalized Compressible Flow Model (PWR pressure drop decrease transient)	22
3	Sectionalized Compressible Flow Model (BWR pressure drop decrease transient)	23
4	Sectionalized Compressible Flow Model (PWR heat flux increase transient)	24
5	Sectionalized Compressible Flow Model (PWR pressure drop decrease transient - long term)	25
6	Momentum Integral Model (PWR pressure drop decrease transient)	26
7	Momentum Integral Model (BWR pressure drop decrease transient)	27
8	Momentum Integral Model (PWR heat flux increase transient) .	28
9	Momentum Integral Model (BWR heat flux increase transient) .	29
10	Momentum Integral Model (PWR pressure drop decrease transient - long term)	30
11	Comparison between Single Mass Velocity Models and Others (PWR pressure drop decrease transient	31
12	Channel Integral (PWR pressure drop decrease transient)	32
13	Channel Integral Model (BWR drop decrease transient)	33
14	Channel Integral Model (PWR heat flux increase transient) ..	34
15	Channel Integral Model (PWR pressure drop decrease transient-long term)	35

List of Tables

<u>Table No.</u>		<u>Page</u>
1	Cases Analyzed	8

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:

$$\frac{\partial \rho_m}{\partial t} + \frac{\partial G_m}{\partial z} = 0 \quad (1)$$

$$\frac{\partial G_m}{\partial t} + \frac{\partial}{\partial z} (G_m^2 / \rho_m) = - \frac{\partial P}{\partial z} - \frac{\phi_{\lambda 0}^2 (f/\rho) |G_m| G_m}{2De} - \rho_m g \quad (2)$$

$$\rho_m \frac{\partial H_m}{\partial t} + G_m \frac{\partial H_m}{\partial z} = \frac{\partial P}{\partial t} + \frac{q'}{A} + \frac{G_m}{\rho_m} \left[\frac{\partial P}{\partial z} + \frac{\phi_{\lambda 0}^2 (f/\rho) |G_m| G_m}{2De} \right] \quad (3)$$

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 $G_m(z,t)$, $P(z,t)$, 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

$$\begin{aligned} \frac{\partial \rho_m}{\partial t} &= \left(\frac{\partial \rho_m}{\partial H_m} \right)_P \frac{\partial H_m}{\partial t} + \left(\frac{\partial \rho_m}{\partial P} \right)_{H_m} \frac{\partial P}{\partial t} \\ &= R_h \frac{\partial H_m}{\partial t} + R_p \frac{\partial P}{\partial t} \end{aligned} \quad (5)$$

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

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

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.

$$\begin{aligned} \frac{\rho_m}{c^2} \frac{\partial P}{\partial t} + \frac{\rho_m}{c^2} \frac{\partial G_m}{\partial z} + \left(\frac{R_h G_m}{\rho_m} \right) \frac{\partial P}{\partial z} - (R_h G_m) \frac{\partial H_m}{\partial z} \\ = - R_h \left[\frac{q'}{A} + \frac{\phi_{\ell o}^2 (f/\rho) |G_m| G_m^2}{2\rho_m De} \right] \end{aligned} \quad (7)$$

$$\begin{aligned} \frac{\rho_m}{c^2} \frac{\partial H_m}{\partial t} + \frac{\partial G_m}{\partial z} - \frac{R_p G_m}{\rho_m} \frac{\partial P}{\partial z} + (R_p G_m) \frac{\partial H_m}{\partial z} \\ = R_p \left[\frac{q'}{A} + \frac{\phi_{\ell o}^2 (f/\rho) |G_m| G_m^2}{2\rho_m De} \right] \end{aligned} \quad (8)$$

where c , the isentropic sonic velocity, is given

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

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:

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

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

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

2.2 Finite Difference Equation

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

$$\begin{aligned}
 & (\rho_m)_{j+1/2}^i \left[\frac{(U_m)_{j+1/2}^{i+1} - (U_m)_{j+1/2}^i}{\Delta t} \right] + \left[\frac{(U_m)_{j+1/2}^i [(U_m)_{j+1/2}^i - (U_m)_{j-1/2}^i]}{\Delta z} \right] \\
 & = \frac{P_{j+1}^i - P_j^i}{\Delta z} - \phi_{\ell o}^2 \frac{(f/\rho)_{j+1/2}^i [(\rho_m)_{j+1/2}^i (U_m)_{j+1/2}^i]^2}{2De} - (\rho_m)_{j+1/2}^i g \quad (13)
 \end{aligned}$$

$$\begin{aligned}
 & \frac{(\rho_m)_{j-1/2}^i}{(c^2)_{j-1/2}^i} \left[\frac{P_j^{i+1} - P_j^i}{\Delta t} + (U_m)_{j-1/2}^{i+1} \frac{P_j^i - P_{j-1}^i}{\Delta z} \right] \\
 & = -(\rho_m^2)_{j-1/2}^i \frac{(U_m)_{j-1/2}^{i+1} - (U_m)_{j-1/2}^i}{\Delta z} - (R_h)_{j-1/2}^i \left[\frac{(q')_{j+1/2}^i}{A} \right. \\
 & \quad \left. + \frac{[\phi_{\ell o}^2 (f/\rho)]_{j-1/2}^{i+1} (\rho_m^2)_{j-1/2}^i (U_m^3)_{j-1/2}^{i+1}}{2De} \right] \quad (14)
 \end{aligned}$$

$$\begin{aligned}
 & \frac{(\rho_m)_{j-1/2}^i}{(c^2)_{j-1/2}^i} \left[\frac{(H_m)_j^{i+1} - (H_m)_j^i}{\Delta t} + (U_m)_{j-1/2}^{i+1} \frac{(H_m)_j^i - (H_m)_{j-1}^i}{\Delta z} \right] \\
 & = -(\rho_m)_{j-1/2}^i \frac{(U_m)_{j-1/2}^{i+1} - (U_m)_{j-1/2}^i}{\Delta z} + (R_p)_{j-1/2}^i \left[\frac{(q)_{j-1/2}^i}{A} \right. \\
 & \quad \left. + \frac{[\phi_{\ell o}^2 (f/\rho)]_{j-1/2}^{i+1} (\rho_m^2)_{j-1/2}^i (U_m^3)_{j-1/2}^{i+1}}{2De} \right] \quad (15)
 \end{aligned}$$

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 be identified.

Table 1
Cases Analyzed

<u>Operating Condition</u>	<u>PWR</u>	<u>BWR</u>
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.,

$$\Delta t \leq \Delta z / (C + |U|) \quad (16)$$

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

$$\rho_m = \rho_m(H_m, P^*) \quad (17)$$

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

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

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

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

where $\Delta P = - \int_0^L \frac{\partial P}{\partial z} dz = p_{inlet} - p_{outlet}$

$$G_{ave} = \frac{1}{L} \int_0^L G_m dz$$

$$F = \left(\frac{G_m^2}{\rho_m} \right)_{outlet} - \left(\frac{G_m^2}{\rho_m} \right)_{inlet} + \int_0^L \frac{\phi_{\lambda o}^2 (f/\rho) |G_m| G_m}{2De} dz + \int_0^L \rho_m g dz$$

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

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

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:

$$\begin{aligned} (H_m)_{j+1/2}^{i+1} &= (H_m)_{j+1/2}^i + \frac{\Delta t}{(\rho_m)_{j-1/2}^i \Delta z} \left[(G_m)_{j+1}^i \frac{(H_m)_{j+3/2}^i - (H_m)_{j+1/2}^i}{2} \right. \\ &\quad \left. + (G_m)_j^i \frac{(H_m)_{j+1/2}^i - (H_m)_{j-1/2}^i}{2} \right] + (q')_{j+1/2}^i \Delta t / A (\rho_m)_{j-1/2}^i \quad (21) \end{aligned}$$

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

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

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:

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

where

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

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

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

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

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 $(G_m)_1$.

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

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

$$\gamma_j^{i+1} = \alpha_j^{i+1} \gamma_{j-1}^{i+1}, \quad \gamma_0 = 1$$

$$\delta_j^{i+1} = \alpha_j^{i+1} \delta_{j-1}^{i+1} + \beta_j, \quad \delta_0 = 0$$

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.,

$$\Delta t \leq \Delta Z / |U_m| \quad (26)$$

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

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

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

$$\frac{\partial G_m}{\partial z} = 0 \quad (28)$$

Integrate the momentum equation over the channel length. We have

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

where

$$F = \int_0^L \frac{\phi_{10}^2 (f/\rho) |G_m| G}{2De} dz + \int_0^L \rho_m g dz + \left(\frac{G_m^2}{\rho_m}\right)_{out} - \left(\frac{G_m^2}{\rho_m}\right)_{in}$$

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

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

4.2 Finite Difference Equations

From Eq. (30)

$$\begin{aligned} (\rho_m)_{j-1/2}^i \frac{(H_m)_j^{i+1} - (H_m)_j^i + (H_m)_{j-1}^{i+1} - (H_m)_{j-1}^i}{2\Delta t} \\ + G_m^i \frac{(H_m)_j^{i+1} - (H_m)_{j-1}^{i+1} + (H_m)_j^i - (H_m)_{j-1}^i}{2\Delta z} = \frac{q'}{A} \end{aligned}$$

Rearrange it.

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

where

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

From Eq. (29)

$$G_m^{i+1} = G_m^i + \frac{\Delta t}{L} (\Delta p^i - F^i) \quad (32)$$

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:

$$\frac{dM}{dt} = G_o - G_n \quad (33)$$

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

$$\frac{dE}{dt} = Q - G_n \Delta H \quad (35)$$

where

$$M = \int_0^L \rho_m dz$$

$$E = \int_0^L \rho_m ((H_m) - H_{in}) dz$$

$$\hat{G} = \frac{1}{L} \int_0^L G dz$$

The shape of enthalpy profile in the integration is

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

where

$$\hat{H}(t) = \frac{1}{L} \int_0^L H_m(z,t) dz$$

and

$$\frac{1}{L} \int_0^L \beta(z) dz = 1, \beta_0 = 0$$

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

$$G_m(z,t) = G_0(t) + \gamma(z,\hat{H}) [\hat{G}(t) - G_0(t)] \quad (36)$$

where

$$\frac{1}{L} \int_0^L \gamma(z,\hat{H}) dz = 1, \gamma_0 = 0$$

Define

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

Then from Eq. (33) and (34)

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

From Eq. (35) and (36)

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

Solve Eqs. (38) and (39) for $\frac{d\hat{H}}{dt}$ and G_0 , that is

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

$$G_0 = \hat{G} + \frac{C_1}{C_3 \gamma_n L} [Q - \hat{G}\Delta H] \quad (41)$$

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 differend form. That is

$$\hat{H}^{i+1} = \hat{H}^i + \frac{\Delta t}{C_3 L} [Q^i - G_{ave}^i \Delta H^i] \quad (42)$$

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

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

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

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

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 boiling 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).

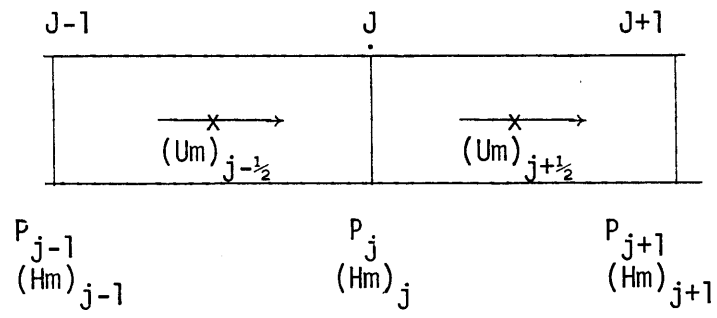


Fig. 1. Sectionalized Compressible Flow Model Calculation Cell.

SECTIONALIZED COMP. FLOW MODEL

PWR Pressure Drop Decrease Tran

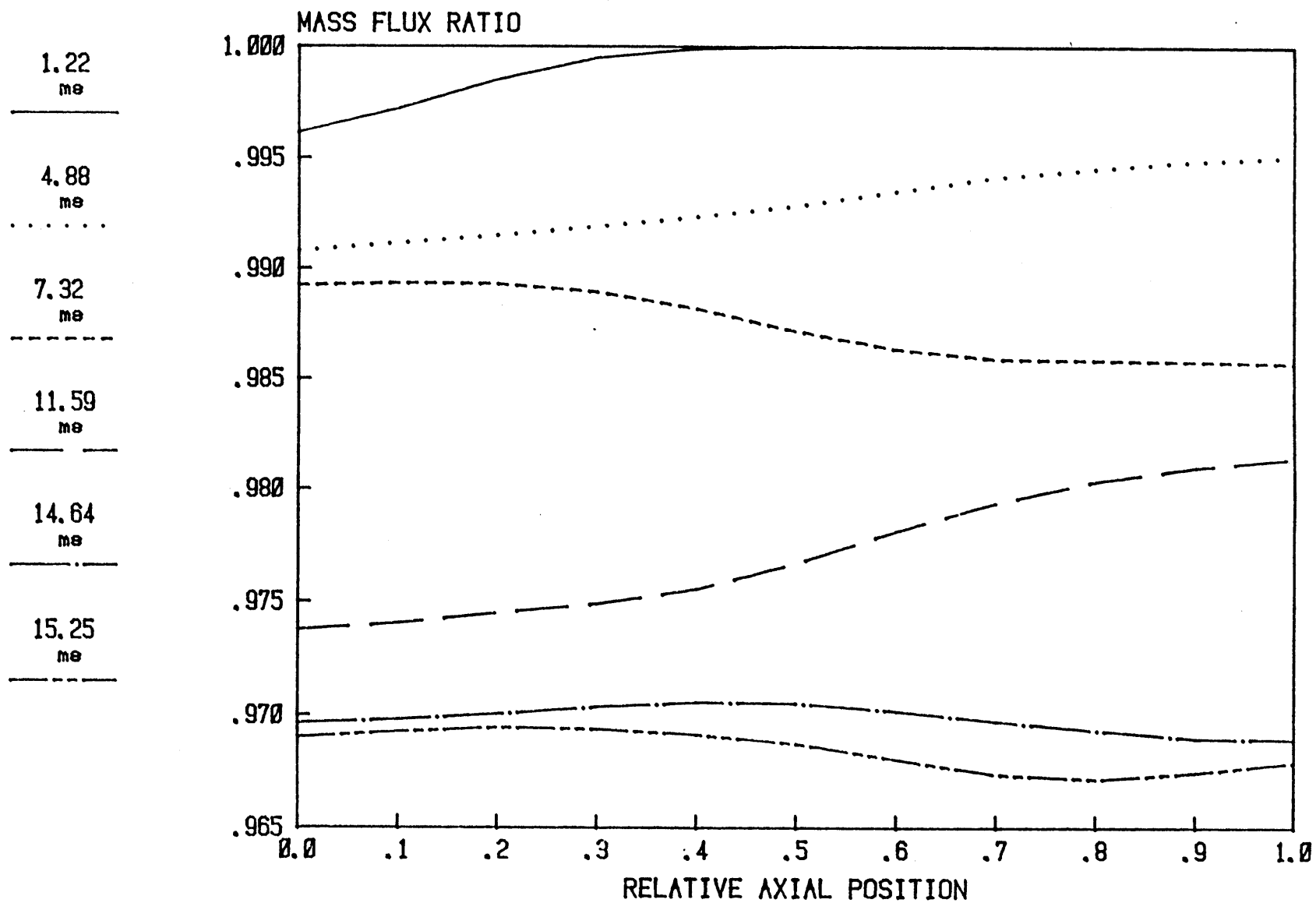


Fig. 2. Sectionalized Compressible Flow Model (PWR Pressure Drop Decrease Tran)

SECTIONALIZED COMP. FLOW MODEL

BWR Pressure Drop Decrease Tran

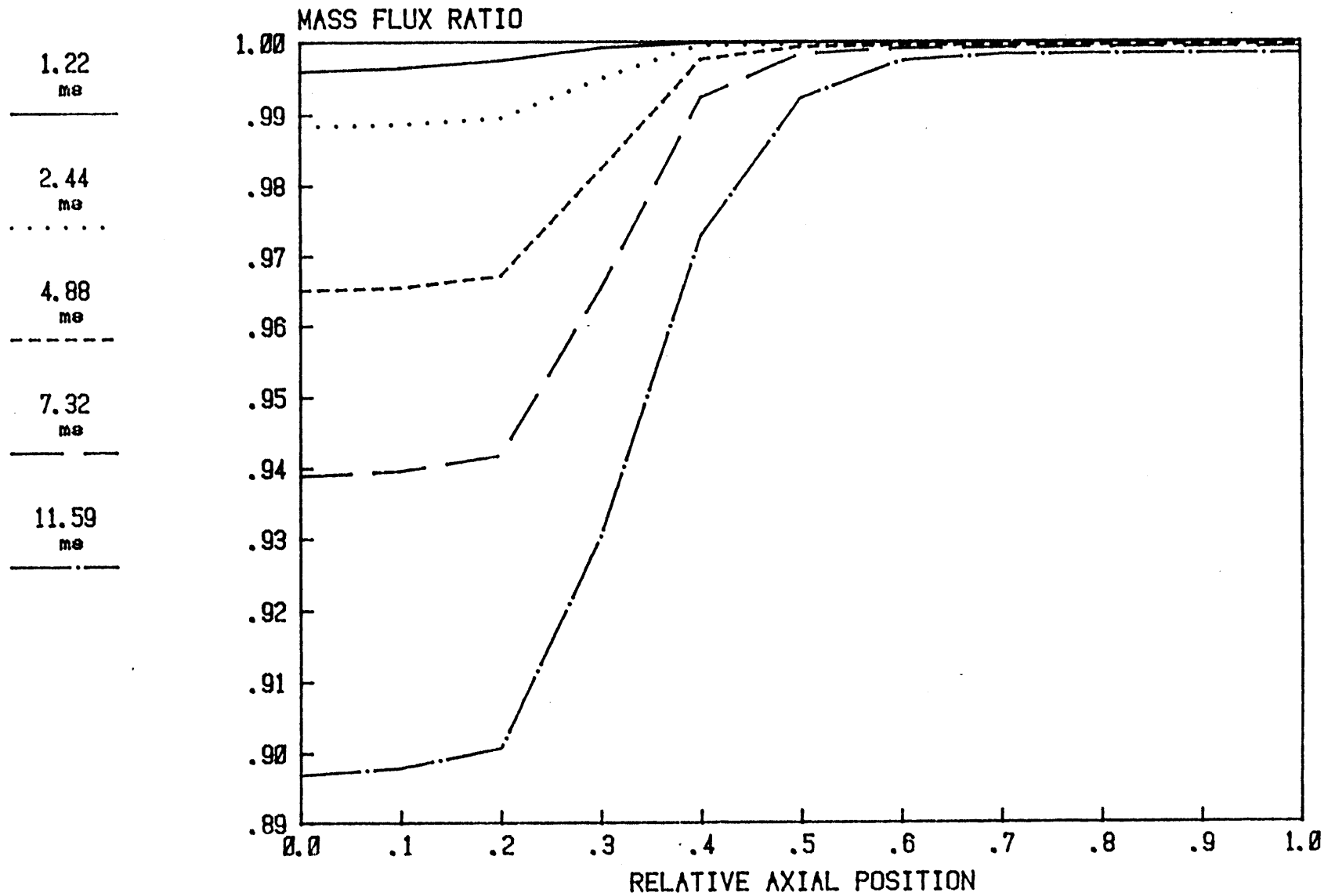


Fig. 3. Sectionalized Compressible Flow Model (BWR Pressure Drop Decrease Transient)

SECTIONALIZED COMP. FLOW MODEL

PWR Heat Flux Increase Trans.

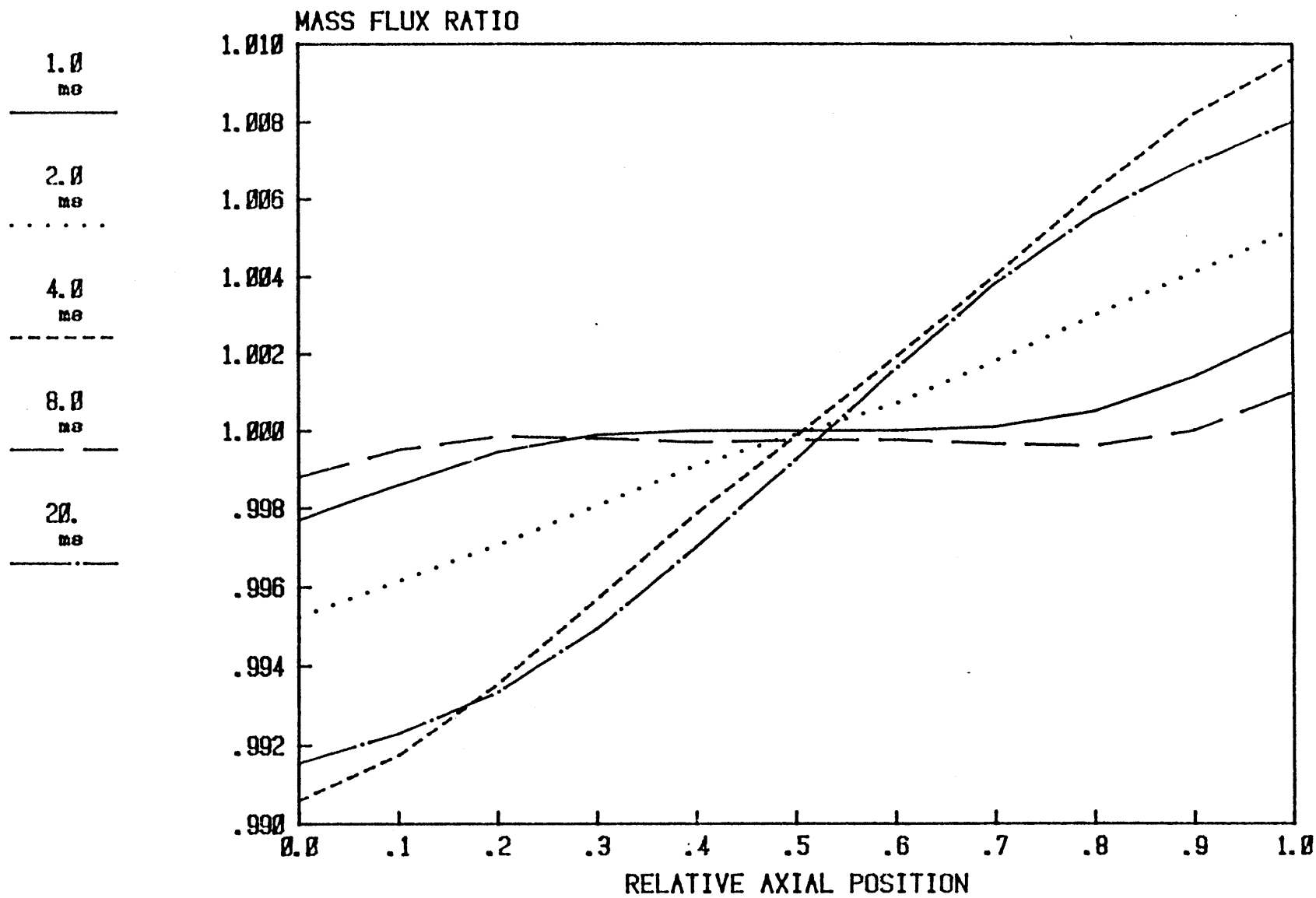


Fig. 4 Sectionalized Compressible Flow Model (PWR Heat Flux Increase Transient)

SECTIONALIZED COMP. FLOW MODEL

PWR Pressure Drop Decrease Tran

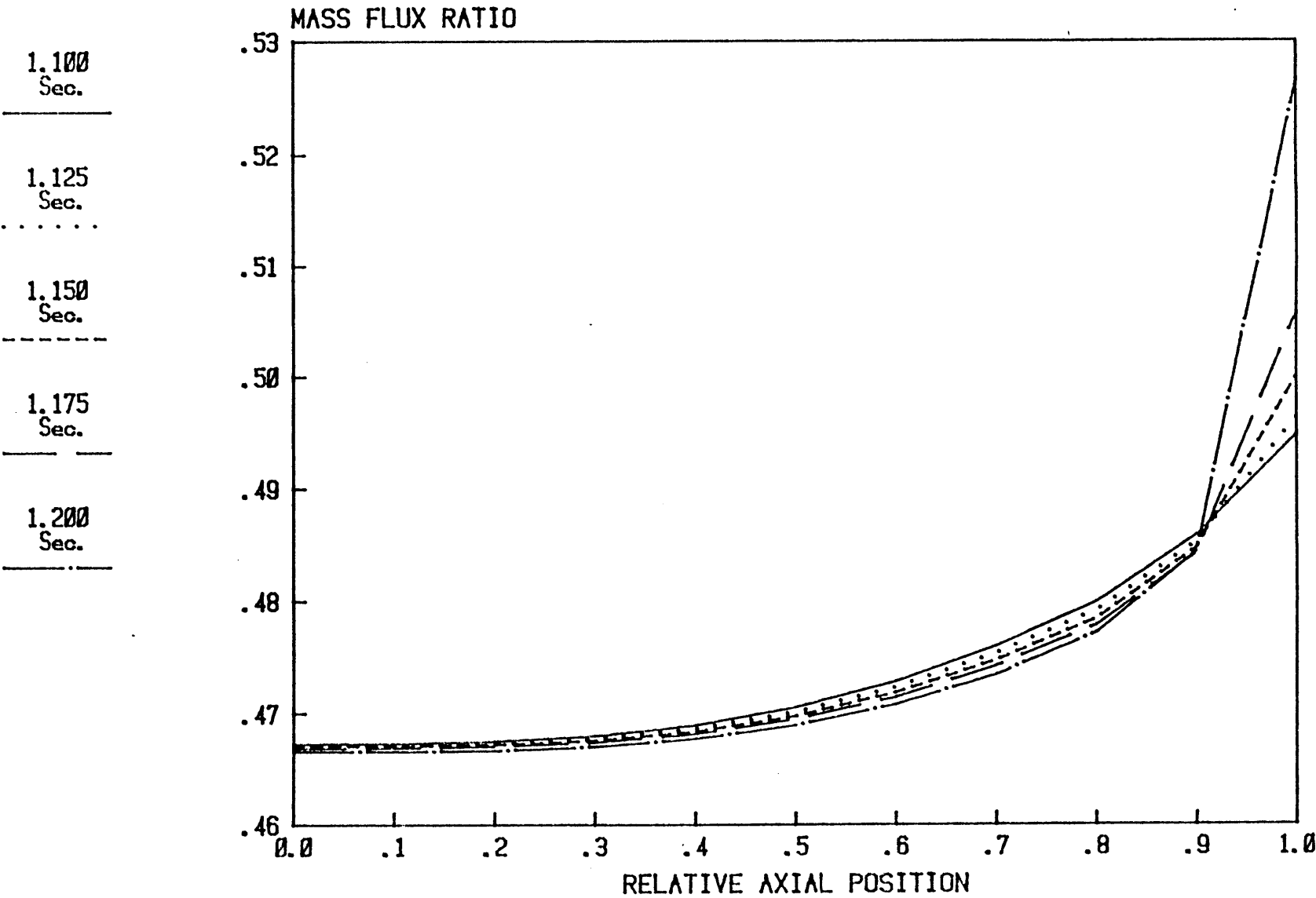


Fig. 5. Sectionalized Compressible Flow Model (PWR Pressure Drop Decrease Transient--Long Term)

MOMENTUM INTEGRAL MODEL

PRW Pressure Drop Decrease Trans

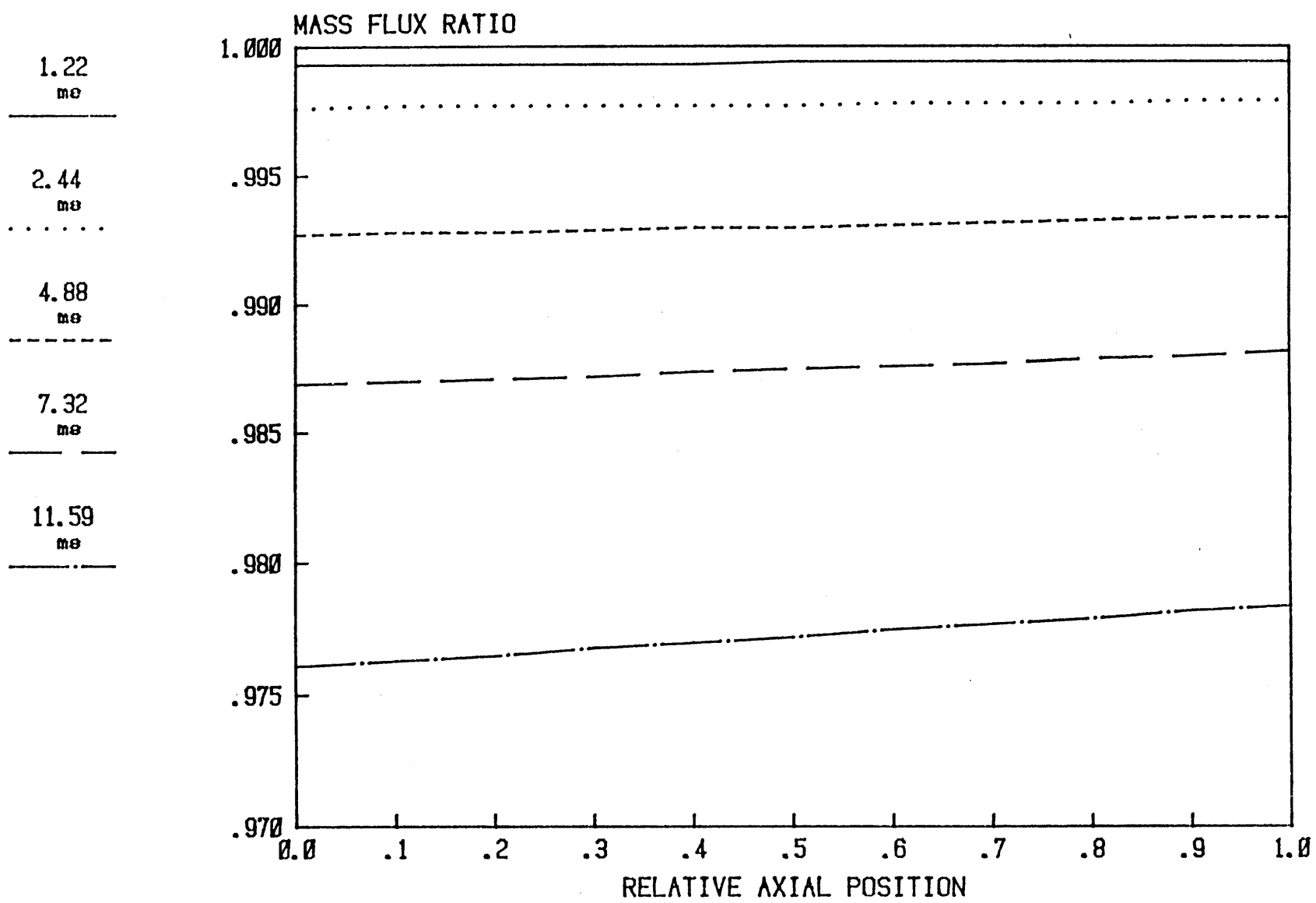


Fig. 6. Momentum Integral Model (PWR Pressure Drop Decrease Transient)

MOMENTUM INTEGRAL MODEL

BWR Pressure Drop Decrease Tran

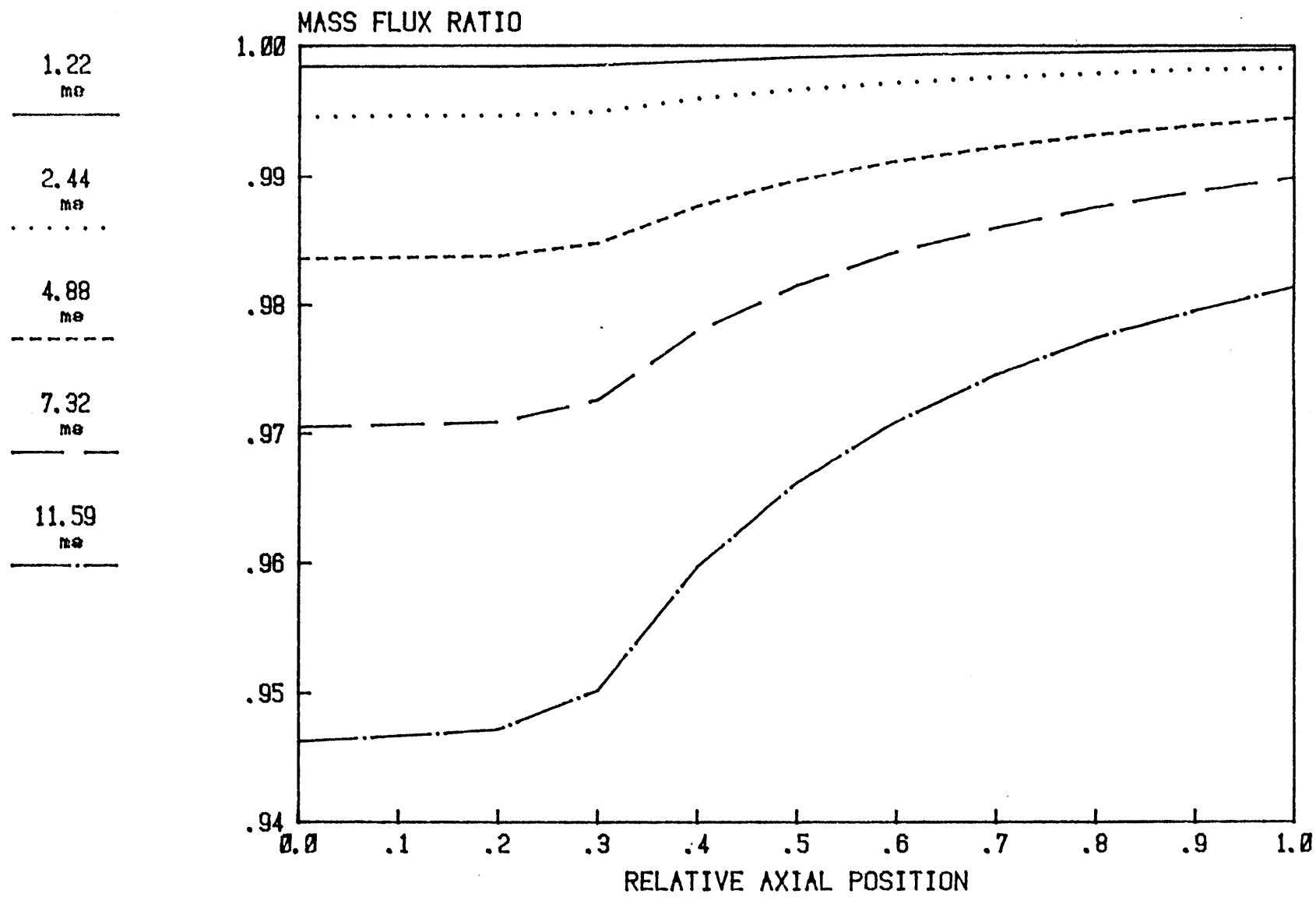


Fig. 7. Momentum Integral Model (BWR Pressure Drop Decrease Transient)

MOMENTUM INTEGRAL MODEL

PWR Heat Flux Increase Trans.

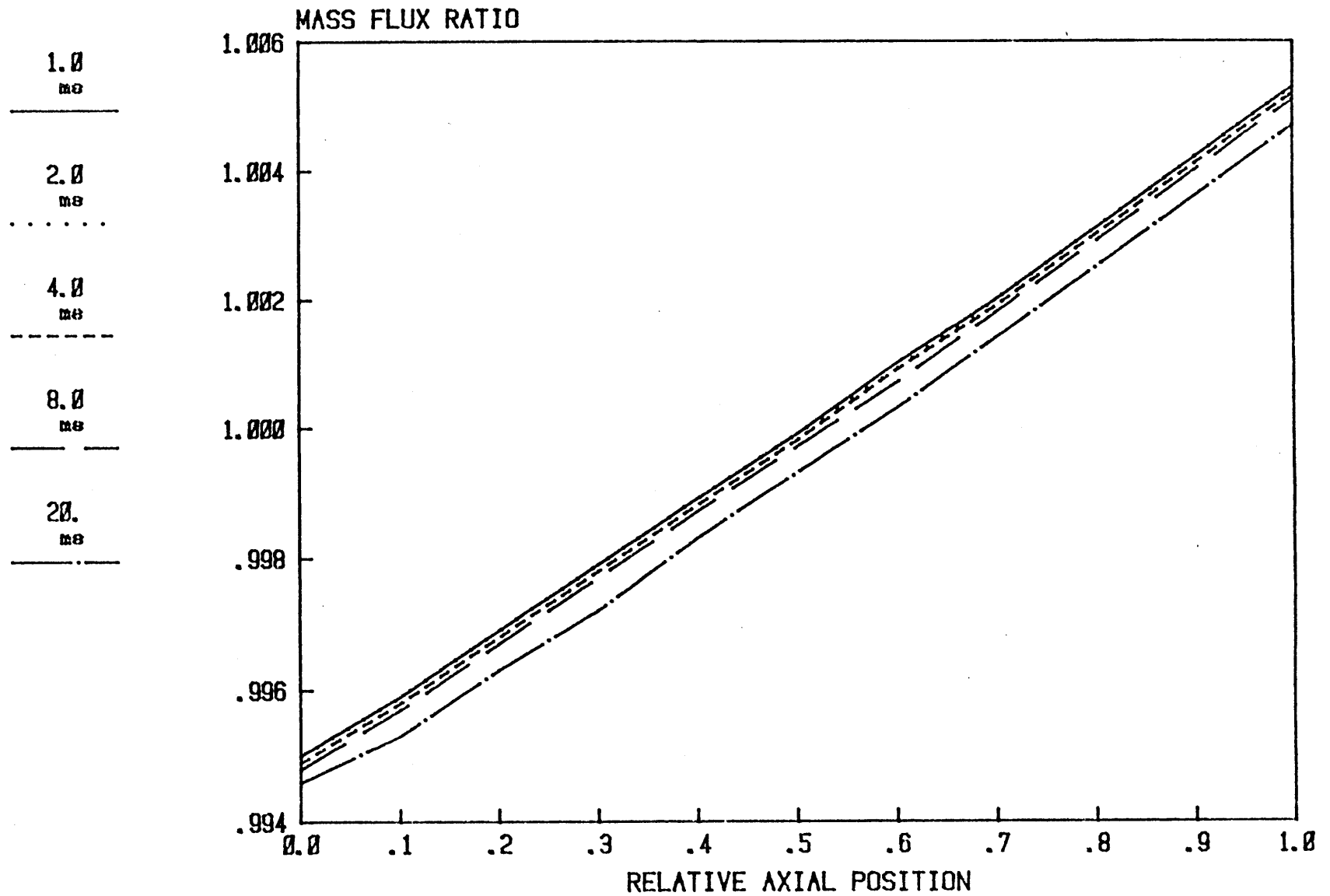


Fig. 8. Momentum Integral Model (PWR Heat Flux Increase Transient)

MOMENTUM INTEGRAL MODEL

BWR Heat Flux Increase Tran.

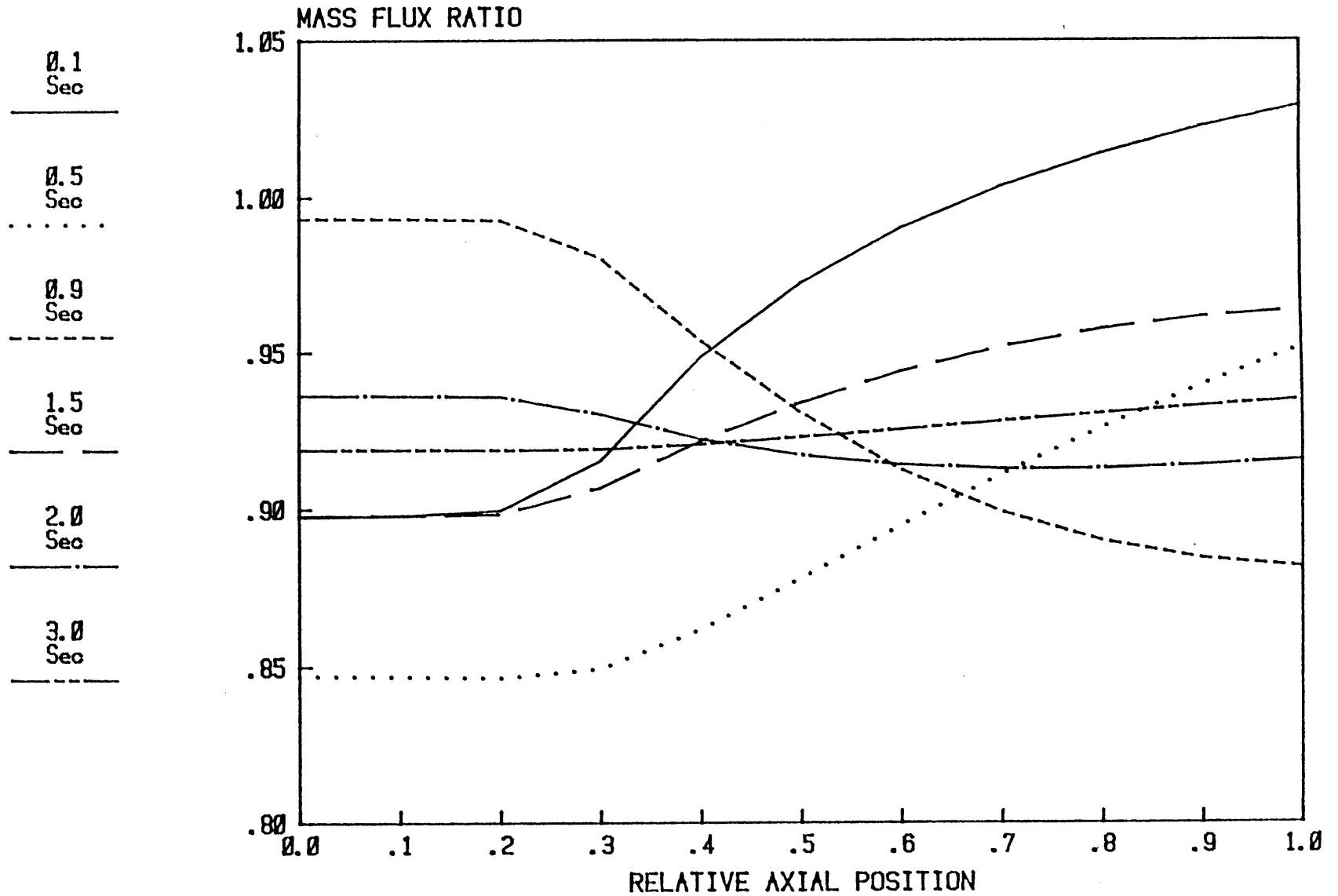


Fig. 9. Momentum Integral Model (BWR Heat Flux Increase Transient)

MOMENTUM INTEGRAL MODEL

PWR Pressure Drop Decrease Tran

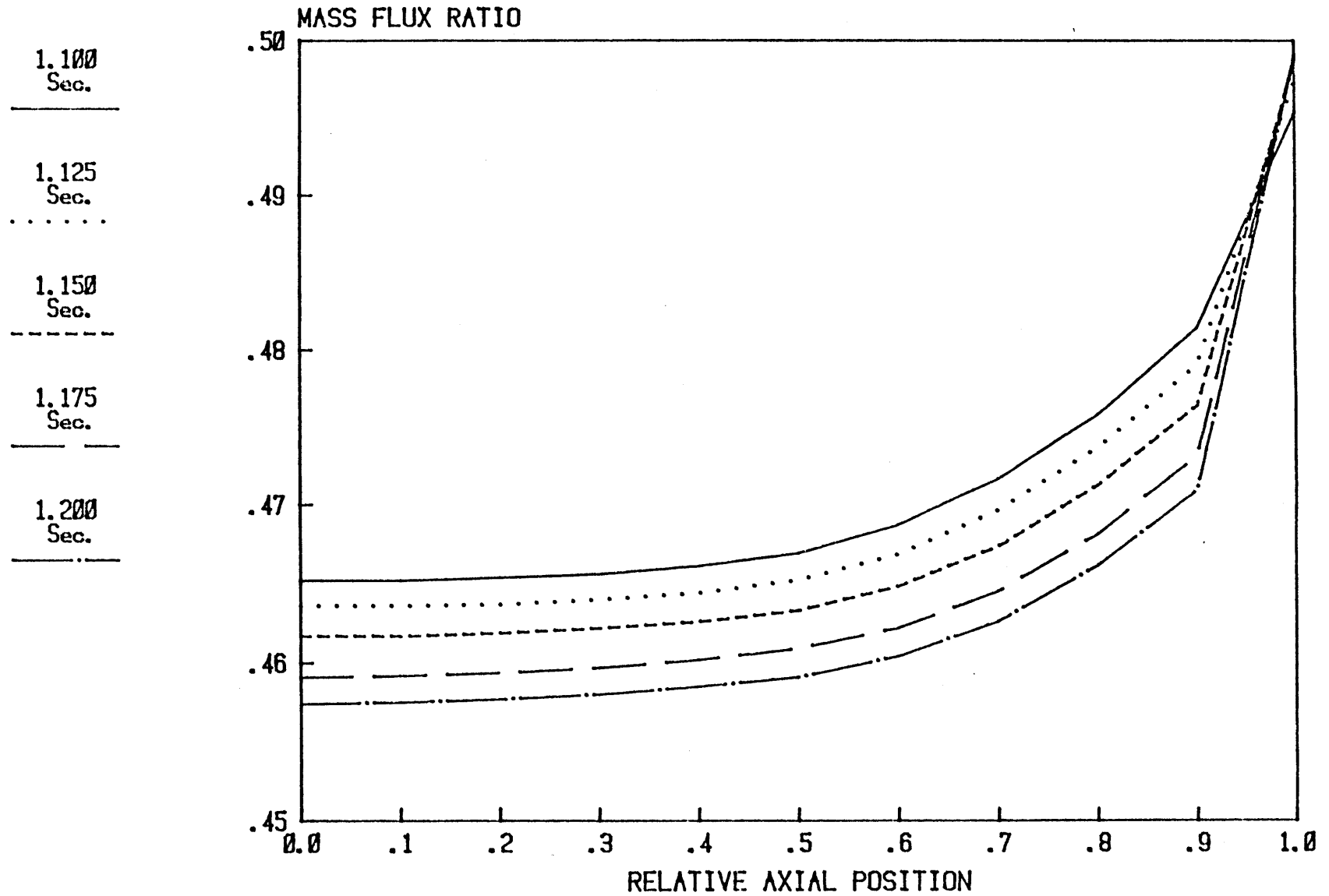


Fig. 10. Momentum Integral Model (PWR Pressure Drop Decrease Transient - Long Term)

SING-VEL. CHANL-INTE. MOMEN-INTE

PWR Pressure Drop Decrease Tran.

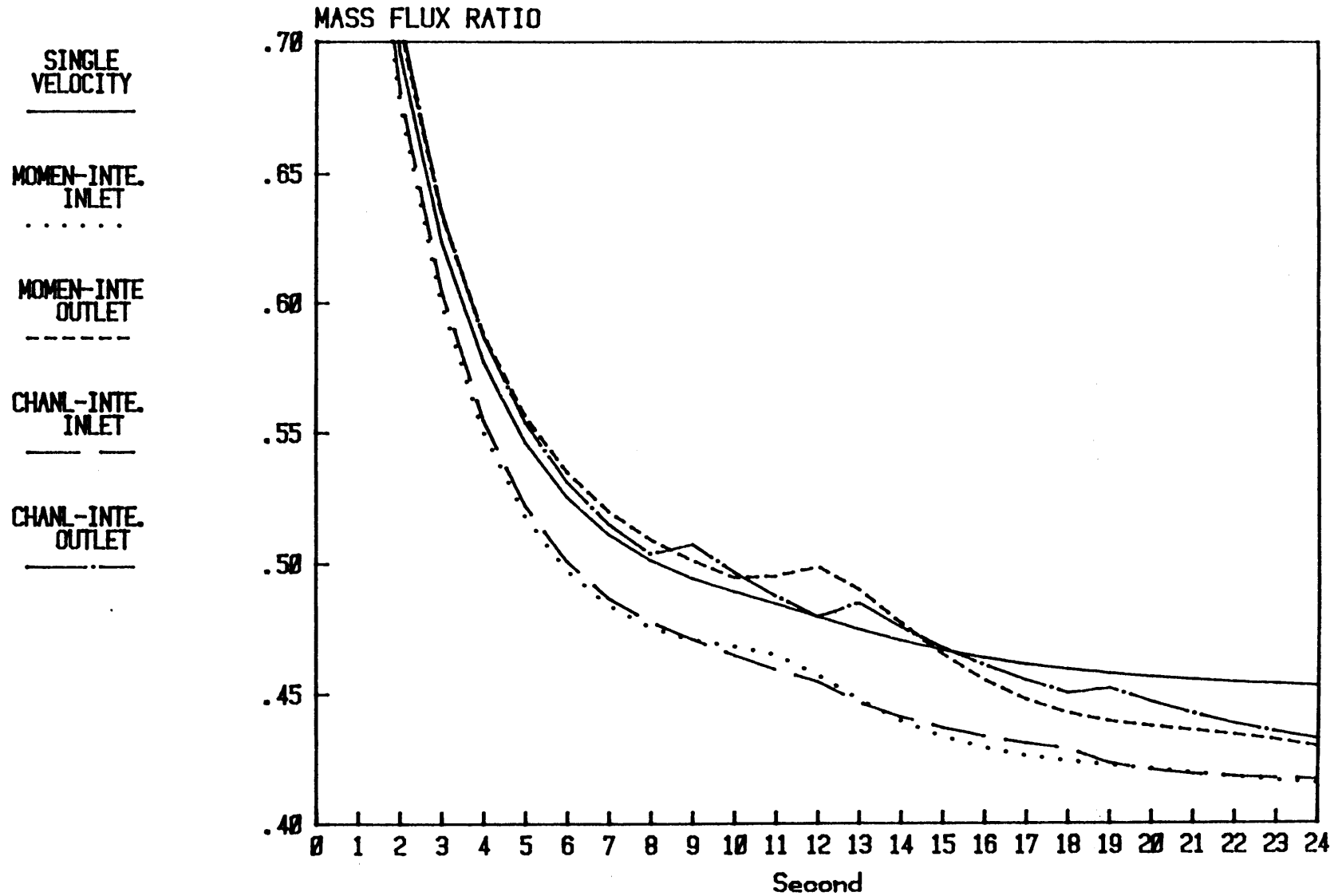


Fig. 11. Comparison between Single Mass Velocity Models and Others (PWR Pressure Drop Decrease Transient)

CHANNEL INTEGRAL MODEL

PWR Pressure Drop Decrease Tran

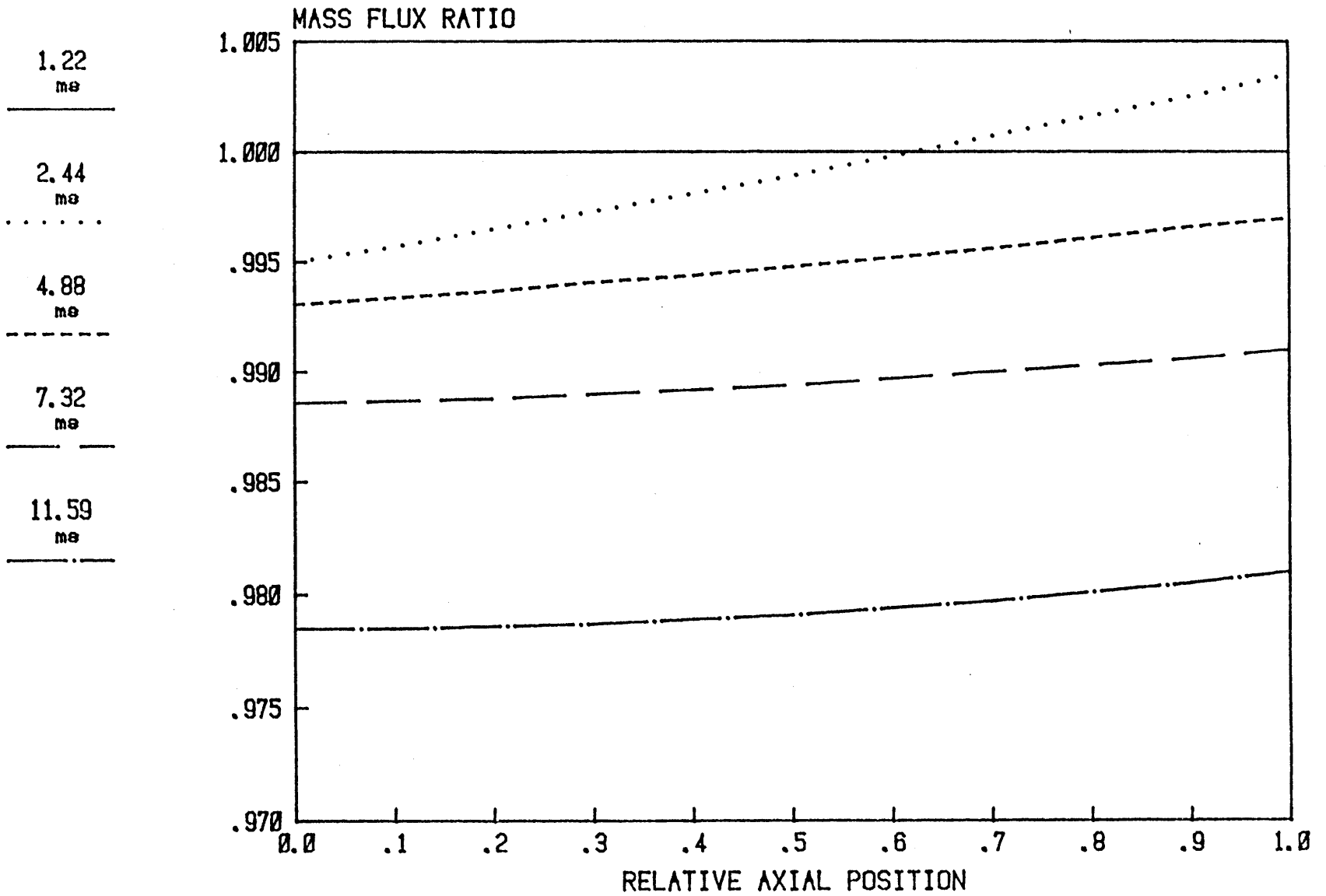


Fig. 12. Channel Integral (PWR Pressure Drop Decrease Transient)

CHANNEL INTEGRAL MODEL

BWR Pressure Drop Decrease Tran

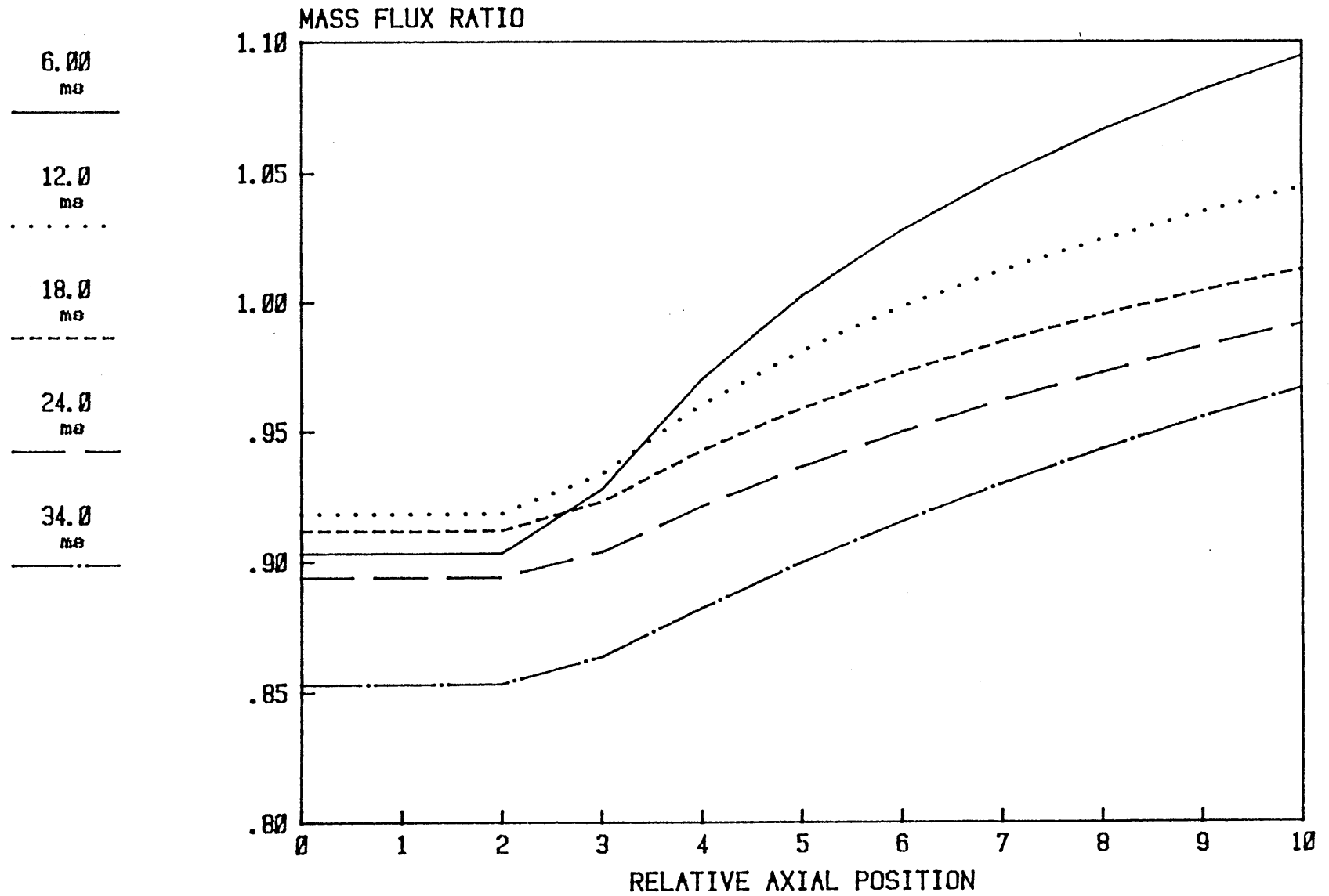


Fig. 13. Channel Integral Model (BWR Pressure Drop Decrease Transient)

CHANNEL INTEGRAL MODEL

PWR Heat Flux Increase Trans.

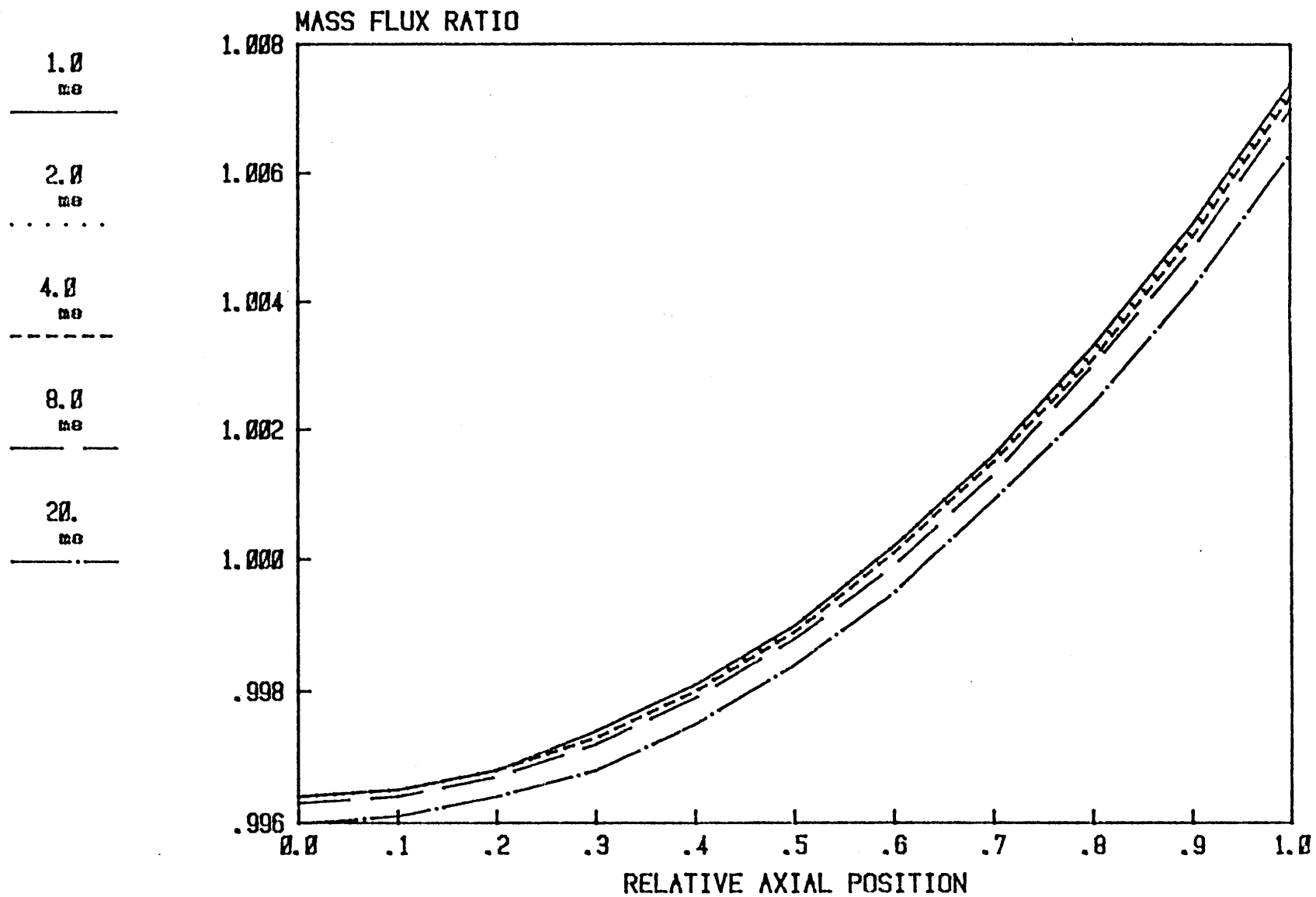


Fig. 14. Channel Integral Model (PWR Heat Flux Increase Transient)

CHANNEL INTEGRAL MODEL

PWR Pressure Drop Decrease Tran

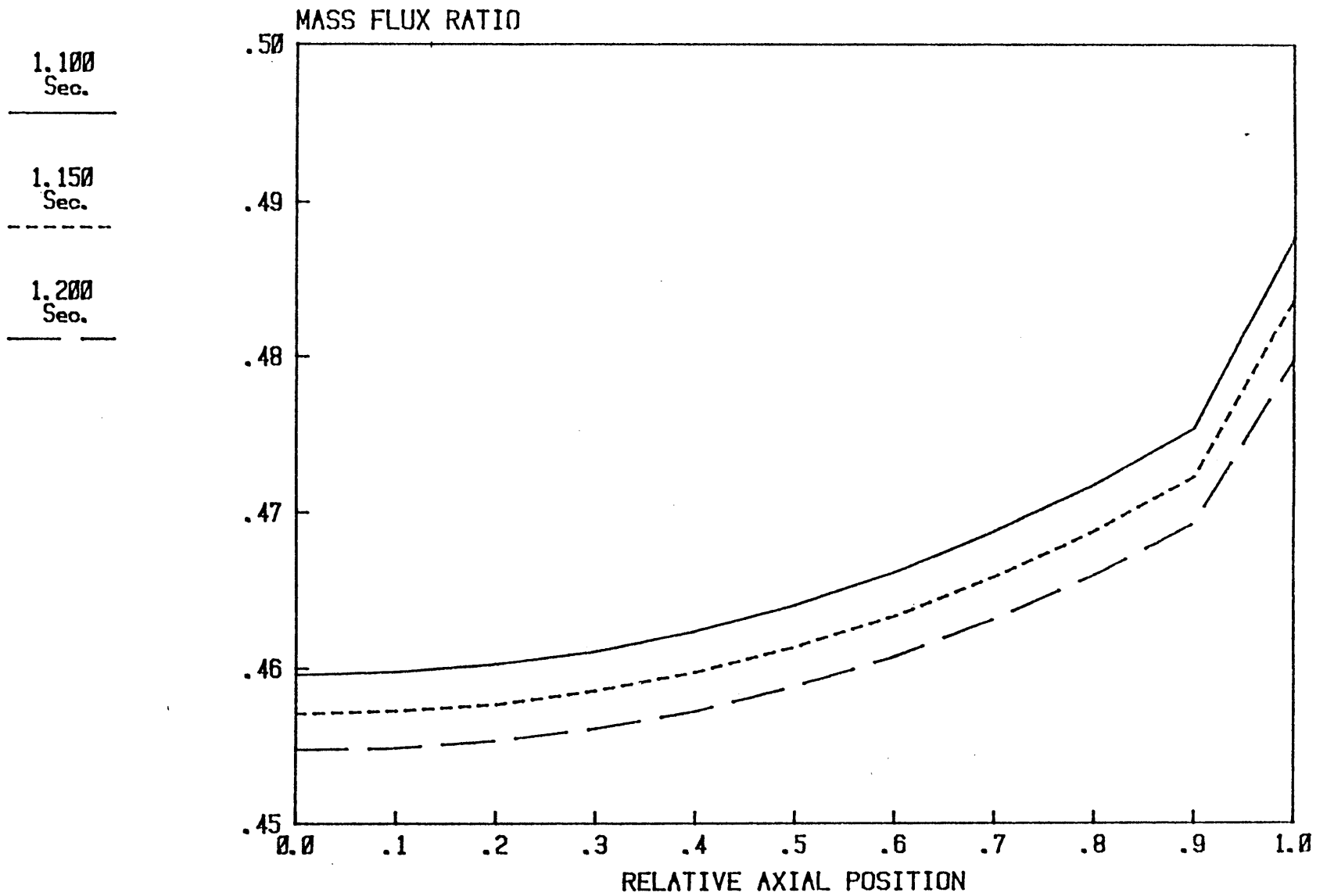


Fig. 15. Channel Integral Model (PWR Pressure Drop Decrease Transient--Long Term)

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 (m²)
 (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) i ϕ (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 \neq 1

- (1) $po(i)$ pressure for ith node (Pa)
- (2) $go(i)$ mass velocity for ith node ($kg/m^2 \text{ sec}$)
- (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


```

1 c solutions of transient balance equations for single heated channel 1
2 c 2
3 c ** SECTIONALIZED COMPRESSIBLE FLOW MODEL ** 3
4 c 4
5 c june 1983 5
6 c 6
7 dimension title(20) 7
8 common /param/hn(21),ho(21),gn(21),go(21),dn(21),do(21),pn(21), 8
9 1 po(21),x(21),un(21),uo(21),uin,uio,din,dio 9
10 common /sonvel/ csq(21) 10
11 common /data1/ chan1,area,equd,nosn,nots,dz,dt 11
12 read (5,1000)(title(i),i=1,20) 12
13 1000 format (20a4) 13
14 read (5,1010)chan1,area,equd 14
15 read(5,1050)is,lsb,lo,npp 15
16 read(5,1030)time,nots,nosn 16
17 1050 format(4i5) 17
18 1010 format(3e10.5) 18
19 read(5,1020)gilt0,hilt0,pilt0,po1t0,powe0 19
20 1020 format(5e10.5) 20
21 1030 format(e10.6,2i5) 21
22 dz=chan1/(nosn-1) 22
23 dt=time/nots 23
24 if(is.eq.1)go to 130 24
25 do 10 i=1,nosn 25
26 read (5,1040)po(i),go(i),ho(i) 26
27 10 continue 27
28 1040 format(3e10.5) 28
29 go to 140 29
30 c 30
31 c 31
32 c calculate the steady state density distribution 32
33 c 33
34 c 34
35 c the initial condition is calculated by sub. init 35
36 c 36
37 130 call init(isb,gilt0,hilt0,pilt0,po1t0,powe0, 37
38 1 po,ho,do,go,x) 38
39 140 continue 39
40 nosn1=nosn-1 40
41 do 90 i=1,nosn1 41
42 ha=(ho(i)+ho(i+1))/2 42
43 pa=(po(i)+po(i+1))/2. 43
44 call densi(pa,ha,do(i),x(i)) 44
45 uo(i)=go(i)/do(i) 45
46 90 continue 46
47 call densi(po(nosn),ho(nosn),do(nosn),x(nosn)) 47
48 call densi(po(1),ho(1),dio,xa) 48
49 uio=gilt0/dio 49
50 uo(nosn)=go(nosn)/do(nosn) 50
51 write(6,2000) 51
52 2000 format (//," transient solutions of single heated channel by", 52
53 1 " sectionalized compressible fluid model ") 53
54 write(6,2010)(title(i),i=1,20) 54
55 2010 format (/,20a4) 55
56 write(6,2020) 56
57 2020 format(/," channel geometry ") 57
58 write (6,2030)chan1,area,equd 58
59 2030 format(1x," channel length=",f6.3," m", 59

```

```

60      1/," flow area = ",e13.6," m**2",
61      1      /," equi diame = ",f6.3," m")
62      write(6,2040)
63      2040 format(/," operating condition ")
64      hiltw=hilt0/1000
65      piltw=pilt0/1.e6
66      poltw=pol0/1.e6
67      powew=powe0/1000
68      write (6,2050)gilt0,hiltw,piltw,poltw,powew
69      2050 format(1x," inlet mass flux = ",f8.3," kg/m**2.sec",
70      1      /," inlet enthalpy = ",f8.3," kj/kg",
71      1      /," inlet pressure = ",f7.4," Mpa",
72      1      /," outlet pressure = ",f7.4," Mpa",
73      1      /," power = ",f7.4," kw/m")
74      write(6,2300)time,dt
75      2300 format(/," total transient time = ",e13.6,
76      1 /," time step size ",e13.6)
77      write(6,2110)
78      write(6,2070)
79      do 150 i=1,nosn
80      pw=po(i)/1.e6
81      hw=ho(i)/1000
82      write(6,3080)i,pw,go(i),hw,do(i),uo(i),x(i)
83      150 continue
84      2110 format(/," initial conditions for transient calculation ")
85      c
86      np=0
87      do 30 i=1,nots
88      9000 format(1x," the time step ",i6)
89      np=np+1
90      ttime=dt*i
91      time1=ttime-dt
92      tpolt=pol0(time1)
93      qipt=power(powe0,time1)
94      hilt=hilt0
95      call calc(tpolt,hilt,qipt,ieror)
96      if(ieror.eq.1) go to 71
97      dtdz=dt/dz
98      c
99      c determine the inlet velocity or inlet pressure
100     c
101     if(io.eq.1) go to 40
102     c
103     c determine the inlet pressure for a flow reduced transient
104     c
105     tgilt=gilt(gilt0,ttime)
106     call densi(po(1),hn(1),din,xa)
107     ha1=(3*ho(1)+ho(2))/4
108     pa=(3*po(1)+po(2))/4
109     call densi(pa,ha1,da,xa1)
110     call rph(pa,ha1,da,xa1,rp1,rha1)
111     csqa1=1/(rp1+rha1/da)
112     if(csqa1.le.0.0)write(0,8888)csqa1
113     8888 format(1x," csqa1",1x,e13.6)
114     uin=tgilt/din
115     uan=(uin+un(1))/2.
116     ga1=da*uan
117     call fric1(xa,pa,ga1,da,foda,ieror)
118     if(ieror.eq.1) go to 71
119     pa=(3*po(2)+po(1))/4.

```

120	ha2=(3*ho(2)+ho(1))/4.	120
121	call densi(pa,ha2,daa,xaa)	121
122	call rph(pa,ha2,daa,xaa,rpa2,rha2)	122
123	csqa2=1./(rpa2+rha2/daa)	123
124	csq1=((csqa1**.5+csqa2**.5)/2.)**2.	124
125	pa=(po(1)+po(2))/2.	125
126	call densi(pa,ha1,da1,xa1)	126
127	call densi(pa,ha2,da2,xa2)	127
128	rh1=(da2-da1)/(ha2-ha1)	128
129	daaa=(daa+da)/2.	129
130	c1=(daaa*csq1)*(un(1)+un(2)-2*uin)*dtdz/2.	130
131	c2=rh1/(daaa/csq1)*(qipt/area+foda*da**2*uan**3/(131
132	1 2*equd))*dt	132
133	c3=(uin+un(1))*(po(2)-po(1))/2.*dtdz	133
134	pn(1)=po(2)+po(1)-pn(2)-2*(c1+c2+c3)	134
135	go to 60	135
136	c	136
137	c determine the inlet velocity for pressure transient	137
138	c	138
139	40 pn(1)=pilt(pilt0,polt0,ttime)	139
140	call densi(pn(1),hn(1),din,xa)	140
141	ha1=(3*ho(1)+ho(2))/4	141
142	pa=(3*po(1)+po(2))/4	142
143	ua=(uio+uo(1))/2	143
144	call densi(pa,ha1,da,xa)	144
145	call rph(pa,ha1,da,xa,rpa1,rha1)	145
146	csqa1=1./(rpa1+rha1/da)	146
147	ga=ua*da	147
148	call fric1(xa,pa,ga,da,foda,ieror)	148
149	if(ieror.eq.1)go to 71	149
150	pa=(3*po(2)+po(1))/4.	150
151	ha2=(3*ho(2)+ho(1))/4.	151
152	call densi(pa,ha2,daa,xaa)	152
153	call rph(pa,ha2,daa,xaa,rpa2,rha2)	153
154	csqa2=1./(rpa2+rha2/daa)	154
155	if(csqa2.le.0.0) write(0,8887)csqa2	155
156	8887 format(1x," csqa=",e13.6)	156
157	csqa=((csqa1**.5+csqa2**.5)/2.)**2.	157
158	pa=(po(2)+po(1))/2.	158
159	call densi(pa,ha1,da1,xa1)	159
160	call densi(pa,ha2,da2,xa2)	160
161	rha=(da2-da1)/(ha2-ha1)	161
162	daaa=(da+daa)/2.	162
163	c1=(foda*da*ua**2/(2*equd)+9.8)*dt*2	163
164	c2=(po(2)-po(1))/da*dtdz*2.	164
165	c3=(uo(1)+uio)*(uo(1)-uio)*dtdz*2	165
166	uin=uo(1)+uio-un(1)-c1-c2-c3	166
167	udif=0.1	167
168	int=0	168
169	itest=0	169
170	it1=0	170
171	d1=0.	171
172	uio1=uin	172
173	uin1=uin	173
174	61 int=int+1	174
175	if(itest.eq.0)uin=uin1	175
176	c	176
177	c iteration over the pressure boundary	177
178	c	178
179	uan=(uin+un(1))/2.	179

180	c1=(daaa*csqa)*(un(1)+un(2)-2*uin)*dtdz/2.	180
181	c2=rha/(daaa/csqa)*(qipt/area+foda*da**2.*uan**3./(2*equd))*dt	181
182	c3=(uin+un(1))*(po(2)-po(1))/2.*dtdz	182
183	pn1=po(2)+po(1)-pn(2)-2*(c1+c2+c3)	183
184	d2=pn1-pn(1)	184
185	if(abs(d2).le.1.0)go to 62	185
186	if(itest.eq.1)go to 66	186
187	if(d1.eq.0.0)go to 63	187
188	if((d1*d2).le.0.0)go to 64	188
189	63 uio1=uin1	189
190	if(it1.eq.1)go to 67	190
191	if(d2.gt.0.0)uin1=uin1-udif	191
192	if(d2.lt.0.0)uin1=uin1+udif	192
193	if(d1.eq.0.0)go to 65	193
194	if(abs(d2).le.abs(d1))go to 65	194
195	it1=1	195
196	67 if(d2.gt.0.0)uin1=uin1+udif	196
197	if(d2.lt.0.0)uin1=uin1-udif	197
198	go to 65	198
199	64 itest=1	199
200	if(d1.gt.0.0)go to 68	200
201	uh=uin1	201
202	u1=uio1	202
203	dh=abs(d2)	203
204	d1=abs(d1)	204
205	go to 69	205
206	68 uh=uio1	206
207	u1=uin1	207
208	dh=abs(d1)	208
209	d1=abs(d2)	209
210	69 uin=(u1*dh+uh*d1)/(d1+dh)	210
211	go to 65	211
212	66 if(d2.gt.0.0)go to 59	212
213	u1=uin	213
214	d1=abs(d2)	214
215	go to 69	215
216	59 uh=uin	216
217	dh=abs(d2)	217
218	go to 69	218
219	65 if(int.gt.50)go to 70	219
220	d1=d2	220
221	go to 61	221
222	62 continue	222
223	tgilt=uin*din	223
224	60 predi=pn(1)-pn(nosn)	224
225	c	225
226	do 260 i1=1,nosn1	226
227	ha=(hn(i1)+hn(i1+1))/2	227
228	pa=(pn(i1)+pn(i1+1))/2.	228
229	call densi(pa,ha,dn(i1),x(i1))	229
230	gn(i1)=dn(i1)*un(i1)	230
231	260 continue	231
232	call densi(pn(nosn),hn(nosn),dn(nosn),x(nosn))	232
233	gn(nosn)=un(nosn)*dn(nosn)	233
234	do 110 i1=1,nosn	234
235	po(i1)=pn(i1)	235
236	ho(i1)=hn(i1)	236
237	go(i1)=gn(i1)	237
238	do(i1)=dn(i1)	238
239	uo(i1)=un(i1)	239

240	110	continue	240
241		uio=uin	241
242		dio=din	242
243		if(i.eq.20000)npp=100	243
244		if(i.eq.20000)np=0	244
245		if(np.ne.npp)go to 31	245
246	71	continue	246
247	c	write(0,9000)i	247
248		np=0	248
249		predw=predi/1000	249
250		qiptw=qipt/1000	250
251		write(6,2060)ttime,predw,qiptw,uin,tgilt	251
252		write(6,2070)	252
253		do 120 ii=1,nosn	253
254		pw=pn(ii)/1.e6	254
255		hw=hn(ii)/1000	255
256		gr=gn(ii)/gilt0	256
257		sonic=csq(ii)**0.5	257
258		write(6,2080)ii,pw,gn(ii),hw,dn(ii),un(ii),x(ii),gr,sonic	258
259	120	continue	259
260		if(ierror.eq.1) go to 160	260
261	2060	format(//," transient time=",e13.6," sec ",	261
262	1	/," pressure drop = ",f8.4," kpa",	262
263	1	/," power = ",f8.4," kw/m",	263
264	1	/," inlet velocity = ",f7.4," m/sec",	264
265	1	/," inlet mass fux = ",f8.3," kg/m**2.sec")	265
266	2070	format(/" pressure ",	266
267	1	" mass flux enthalpy density",	267
268	1	" velocity quality ",	268
269	1	/,7x, " Mpa kg/m**2.sec kj/kg kg/m**3 m/sec. ")	269
270	2080	format(1x,i2,3x,f7.4,6x,f7.2,3x,f8.3,3x,	270
271	1	f7.3,3x,f7.4,4x,f6.3,3x,f7.4,4x,f7.2)	271
272	3080	format(1x,i2,3x,f7.4,6x,f7.2,3x,f8.3,3x,	272
273	1	f7.3,3x,f7.4,4x,f6.3)	273
274	31	continue	274
275	30	continue	275
276		go to 160	276
277	c		277
278	70	write(6,2100)ttime	278
279	2100	format(1x," iteration not converge at time eq. ",e13.6)	279
280	160	continue	280
281	c		281
282		end	282

283	subroutine init(is1,gilt0,hilt0,pilt0,po1t0,powe0,	1
284	1 p,h,d,g,x)	2
285	dimension h(21),d(21),p(21),g(21),x(21)	3
286	common /data1/ chan1,area,equd,nosn,nots,dz,dt	4
287	call densi(pilt0,hilt0,d(1),x(1))	5
288	h(1)=hilt0	6
289	p(1)=pilt0	7
290	g(1)=gilt0	8
291	i=1	9
292	n1=1	10
293	90 do 10 i=2,nosn	11
294	h1=h(i-1)+powe0*dz/(area*gilt0)	12
295	p1=p(i-1)	13
296	call densi(p1,h1,d(i),x(i))	14
297	60 ha=(h1+h(i-1))/2.	15
298	pa=(p2+p(i-1))/2.	16
299	call densi(pa,ha,da,x1)	17
300	call fric1(x1,pa,gilt0,da,fod,ieror)	18
301	p2=p(i-1)-fod*gilt0**2.*dz/(2*equd)	19
302	1 -da*9.80*dz-gilt0**2.*(1/d(i)-1/d(i-1))	20
303	call densi(p2,h1,d(i),x(i))	21
304	pa=(p2+p(i-1))/2.	22
305	call densi(pa,ha,da,x1)	23
306	call fric1(x1,p2,gilt0,da,fod,ieror)	24
307	31 h1=h(i-1)+powe0*dz/(area*gilt0)+(p2-p(i-1)+	25
308	1 (fod*gilt0**2*dz)/(2*equd))/da	26
309	if(abs((p2-p1)/p2).le.1.0e-8)go to 40	27
310	n=n+1	28
311	if(n.gt.50)goto 50	29
312	p1=p2	30
313	goto 60	31
314	40 p(i)=p2	32
315	h(i)=h1	33
316	g(i)=gilt0	34
317	10 continue	35
318	if(is1.eq.1)po1t0=p(nosn)	36
319	if(abs((p(nosn)-po1t0)/po1t0).le.1.0e-4)return	37
320	if(n1.gt.50) go to 80	38
321	n1=n1+1	39
322	gilt0=gilt0*((p(nosn)-p(1))/(po1t0-pilt0))*0.5	40
323	goto 90	41
324	80 write(6,2110)	42
325	2110 format(1x," iteration not converge for pressure b.c.")	43
326	return	44
327	50 write(6,2090) i	45
328	2090 format(1x,"iteration not converge for spatial node ",i2)	46
329	return	47
330	end	48

```
331 function power(powe0,time1)
332 power=powe0
333 return
334 end
```

```
1
2
3
4
```

```
335 function polt(polto,time1)
336 polt=polto
337 return
338 end
```

```
1
2
3
4
```



```
339 function gilt(gilt0,time1)
340   gilt=gilt0
341   return
342 end
```

```
1
2
3
4
```

```

343      subroutine fric1(x1,p,g,d,fod,ieror)                1
344      common /data1/ chan1,area,equd,nosn,nots,dz,dt      2
345      data conv1,conv2,vics/737.4643,1.4504e-4,91.7e-6/    3
346      data vics1/19.73e-6/                                4
347 c
348      ieror=0                                             5
349      if(g.lt.0.0)ieror=1                                6
350      if(g.lt.0.0)go to 60                                7
351 c
352 c use McAdames correlation to calculate the friction factor 10
353      thm=1.0                                             11
354      dpc=d                                               12
355      if(x1.eq.-1.0)go to 10                              13
356      if(x1.eq.1.)go to 20                               14
357      call satt(p,h1s,hvs)                                15
358      call lden(p,h1s,ro1)                                16
359      call vden(p,hvs,rov)                                17
360      gtest=g*conv1/1.0e6                                 18
361      if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest  19
362      if(gtest.gt.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest 20
363      if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest  21
364      if(gtest.gt.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest 22
365      thm=c1*(1.2*(ro1/rov-1)*x1**0.824)+1.0             23
366      dpc=ro1                                             24
367      fric=0.184*(g*equd/vics)**(-0.2)                   25
368      go to 21                                            26
369      fric=0.184*(g*equd/vics1)**(-0.2)                  27
370      fod=thm*fric/dpc                                    28
371      continue                                           29
372      return                                              30
373      end                                                  31

```

```

374      subroutine calc(tpolt,hilt,qipt,ieror)          1
375      common /sonvel/csq(21)                        2
376      common /param/ hn(21),ho(21),gn(21),go(21),dn(21),do(21),pn(21)  3
377      1 ,po(21),x(21),un(21),uo(21),uin,uio,din,dio  4
378      common /data1/ chan1,area,equd,nosn,nots,dz,dt  5
379      dimension fod(21),rp(21),rh(21)              6
380      nosn1=nosn-1                                  7
381      nosn2=nosn-2                                  8
382      pa=po(1)/2.+po(2)/2.                          9
383      do 10 i=1,nosn1                                10
384      ha=(ho(i)+ho(i+1))/2                          11
385      pa=(po(i)+po(i+1))/2.                          12
386      call rph(pa,ha,do(i),x(i),rp(i),rh(i))        13
387      csq(i)=1/(rp(i)+rh(i)/do(i))                  14
388      if(csq(i).le.0.0)write(0,9999)i,csq(i)        15
389 9999 format(1x,"i=",i4,"csq(i)=",e13.6)          16
390      call fric1(x(i),po(i),go(i),do(i),fod(i),ieror) 17
391      if(ieror.eq.1)goto 100                        18
392 10 continue                                       19
393      do 11 i=1,nosn2                                20
394      ha1=(ho(i)+ho(i+1))/2.                        21
395      ha2=(ho(i+1)+ho(i+2))/2.                    22
396      pa1=(po(i)+po(i+1))/2.                      23
397      pa2=(po(i+1)+po(i+2))/2.                    24
398      call densi(po(i+1),ha1,da1,xa1)              25
399      call densi(po(i+1),ha2,da2,xa2)              26
400      call densi(pa1,ho(i+1),da3,xa3)              27
401      call densi(pa2,ho(i+1),da4,xa4)              28
402      rp(i)=(da3-da4)/(pa1-pa2)                    29
403      rh(i)=(da2-da1)/(ha2-ha1)                    30
404 11 continue                                       31
405      pa=(3*po(nosn)+po(nosn1))/4.                 32
406      ha1=(ho(nosn1)+ho(nosn))/2.                  33
407      call densi(pa,ha1,da1,xa1)                   34
408      call densi(pa,ho(nosn),da2,xa2)              35
409      rh(nosn1)=(da2-da1)/(ho(nosn)-ha1)           36
410      ha=(3*ho(nosn)+ho(nosn1))/4.                 37
411      pa1=(po(nosn1)+po(nosn))/2.                  38
412      call densi(pa1,ha,da3,xa3)                   39
413      call densi(po(nosn),ha,da4,xa4)              40
414      rp(nosn1)=(da4-da3)/(po(nosn)-pa1)           41
415      hn(1)=hilt                                    42
416      dtdz=dt/dz                                    43
417      c1=uo(1)*(uo(1)-uio)*2*dtdz                  44
418      c2=(po(2)-po(1))/do(1)*dtdz                  45
419      c3=fod(1)*do(1)*uo(1)**2.*dt/(2*equd)+9.8*dt 46
420      un(1)=uo(1)-c1-c2-c3                          47
421      do 40 i=2,nosn1                                48
422      c1=uo(i)*(uo(i)-uo(i-1))*dtdz                49
423      c2=(po(i+1)-po(i))/do(i)*dtdz                50
424      c3=fod(i)*do(i)*uo(i)**2.*dt/(2*equd)+9.8*dt 51
425      un(i)=uo(i)-c1-c2-c3                          52
426      go1=un(i)*do(i)                                53
427      call fric1(x(i),po(i),go1,do(i),fod(i),ieror) 54
428      if(ieror.eq.1) go to 100                      55
429 40 continue                                       56
430      do 50 i=2,nosn1                                57
431      da=(do(i-1)+do(i))/2.                          58
432      csqa=((csq(i-1)**.5+csq(i)**.5)/2.)*2.         59

```

433	rha=rh(i-1)	60
434	c1=un(i-1)*(po(i)-po(i-1))*dtdz	61
435	c2=(da*csqa)*(un(i)-un(i-1))*dtdz	62
436	c3=rha/(da/csqa)*(qipt/area+fod(i-1)*	63
437	1 do(i-1)**2.*un(i-1)**3./(2*equd))*dt	64
438	pn(i)=po(i)-c1-c2-c3	65
439	50 continue	66
440	pn(nosn)=tpo1t	67
441	ua=un(nosn1)	68
442	call rph(po(nosn),ho(nosn),do(nosn),x(nosn),rp(nosn),rh(nosn))	69
443	csq(nosn)=1./(rp(nosn)+rh(nosn)/do(nosn))	70
444	if(csq(nosn).le.0.0)write(0,8886)csq(nosn)	71
445	8886 format(1x," csqa11 ",e13.6)	72
446	uan=un(nosn1)	73
447	csqa=((csq(nosn1)**.5+csq(nosn1)**.5)/2.)**2.	74
448	rha=rh(nosn1)	75
449	daa=(do(nosn1)+do(nosn))/2.	76
450	c1=(3*pn(nosn)+pn(nosn1)-3*po(nosn)-po(nosn1))/(8*dtdz)	77
451	c2=un(nosn1)*(po(nosn)-po(nosn1))	78
452	c3=(c1+c2)/(daa*csqa)	79
453	c4=rha*dz/daa**2.*(qipt/area+fod(nosn1)*do(nosn1)**2.	80
454	1 *un(nosn1)**3./(2*equd))/2.	81
455	un(nosn)=un(nosn1)-c3-c4	82
456	do 130 i=2,nosn1	83
457	csqa=((csq(i-1)**.5+csq(i)**.5)/2.)**2.	84
458	da=(do(i-1)+do(i))/2.	85
459	rpa=rp(i-1)	86
460	c1=un(i-1)*(ho(i)-ho(i-1))*dtdz	87
461	c2=csqa*(un(i)-un(i-1))*dtdz	88
462	c3=rpa/(da/csqa)*(qipt/area+fod(i-1)	89
463	1 *do(i-1)**2.*un(i-1)**3./(2*equd))*dt	90
464	hn(i)=ho(i)+c3-c1-c2	91
465	130 continue	92
466	csqa=((csq(nosn)**.5+csq(nosn1)**.5)/2.)**2.	93
467	rpa=rp(nosn1)	94
468	daa=(do(nosn1)+do(nosn))/2.	95
469	c1=un(nosn1)*(ho(nosn)-ho(nosn1))*dtdz	96
470	c2=csqa*(un(nosn)-un(nosn1))	97
471	1 *2*dtdz	98
472	c3=rpa/(daa/csqa)*(qipt/area+fod(nosn1)	99
473	1 *do(nosn1)**2.*un(nosn1)**3./(2*equd))*dt	100
474	hn(nosn)=ho(nosn)+c3-c1-c2	101
475	100 continue	102
476	return	103
477	end	104

478	subroutine densi(p,h,d,x)	1
479	c	2
480	call satt(p,h1s,hvs)	3
481	if(h.le.h1s) go to 10	4
482	if (h.gt.hvs) go to 20	5
483	call lden(p,h1s,ro1)	6
484	call vden(p,hvs,rov)	7
485	x=(h-h1s)/(hvs-h1s)	8
486	sv1=1./ro1	9
487	svv=1./rov	10
488	sv=sv1*(1-x)+svv*x	11
489	d=1./sv	12
490	return	13
491	10 x=-1.	14
492	call lden(p,h,ro1)	15
493	d=ro1	16
494	return	17
495	20 x=1.	18
496	call vden(p,h,rov)	19
497	d=rov	20
498	return	21
499	end	22

500	subroutine lden(p,h,ro1)	1
501	data f11,f12/999.65,4.9737e-7/	2
502	data f21,f22/-2.5847e-10,6.1767e-19/	3
503	data f31,f32/1.2696e-22,-4.9223e-31/	4
504 c		5
505	data f41,f42/1488.64,1.3389e-6/	6
506.	data f51,f52/1.4695e9,8.85736/	7
507	data f61,f62/3.20372e6,1.20483e-2/	8
508 c		9
509	if(h.gt.6.513e5) go to 10	10
510	f1=f11+f12*p	11
511	f2=f21+f22*p	12
512	f3=f31+f32*p	13
513	ro1=f1+h*h*(f2+f3*h*h)	14
514	go to 20	15
515	10 f4=f41+f42*p	16
516	f5=f51+f52*p	17
517	f6=f61+f62*p	18
518	ro1=f4+f5/(h-f6)	19
519	20 return	20
520	end	21

```
521      subroutine vden(p,h,rov)                                1
522 c                                                                 2
523      data g00,g01,g02,g10,g11,g12/-5.1026e-5,1.1208e-10,    3
524      1      -4.4506e5,-1.6893e-10,-3.3980e-17,2.3058e-1/    4
525 c                                                                 5
526      g0p=g00+g01*p+g02/p                                    6
527      g1p=g10+g11*p+g12/p                                    7
528      rov=1./(g0p+g1p*h)                                     8
529      return                                                  9
530      end                                                       10
```

```

531      subroutine satt(p,hls,hvs)                                1
532      data tsc1,tsc2, tsexp /9.0395, 255.2, 0.223/           2
533      data cps1,cps2, cpsexp /9.5875e2, .00132334, -0.8566/  3
534 c    for hls and hvs                                        4
535      data h10,h11,h12,h13,h14,h15/5.7474718e5,2.0920624e-1,  5
536      1 -2.8051070e-8,2.3809828e-15,-1.0042660e-22,1.6586960e-30/ 6
537      data hv0,hv1,hv2,hv3,hv4/2.7396234e6,3.758844e-2,      7
538      1 -7.1639909e-9,4.2002319e-16,-9.8507521e-24 /        8
539 c                                                    9
540      hvs = hv0 + p*(hv1 + p*(hv2 + p*(hv3 + p*hv4)))        10
541      hls = h10 + p*(h11 + p*(h12 + p*(h13 + p*(h14 + p*h15)))) 11
542      return                                                  12
543      end                                                       13

```



```

544      subroutine rph(p,h,d,x,rp,rh)
545 c
546      data h111,h122,h133,h144,h155/2.0920624e-1,-5.610214e-8,
547 1 7.142948e-15,-4.0170640e-22,8.293480e-30/
548 c
549      data hv11,hv22,hv33,hv44/3.758844e-2,-14.33279818e-9,
550 1 12.6006957e-16,-39.4030084e-24/
551 c
552      data f11,f12/999.65,4.9737e-7/
553      data f21,f22/-2.5847e-10,6.1767e-19/
554      data f31,f32/1.2696e-22,-4.9223e-31/
555 c
556      data f41,f42/1488.64,1.3389e-6/
557      data f51,f52/1.4695e9,8.85736/
558      data f61,f62/3.20372e6,1.20483e-2/
559 c
560      data g00,g01,g02,g10,g11,g12/-5.1026e-5,1.1208e-10,
561 1 -4.4506e5,-1.6893e-10,-3.3980e-17,2.3058e-1/
562 c
563      if(x.eq.-1.)go to 10
564      call satt(p,h1s,hvs)
565      call lden(p,h1s,rol)
566      call vden(p,hvs,rov)
567      dh1dp=h111+p*(h122+p*(h133+p*(h144+p*h155)))
568      dhvdp=hv11+p*(hv22+p*(hv33+p*hv44))
569      dvvdp=(g01-g02/p**2.)+(g11-g12/p**2.)*hvs
570      if(h1s.gt.6.513e5) go to 20
571      f2=f21+f22*p
572      f3=f31+f32*p
573      dv1dp=-(f12+h1s*h1s*(f22+f32*h1s*h1s))/rol**2.
574      go to 30
575 20 f5=f51+f52*p
576      f6=f61+f62*p
577      dv1dp=-(f42+f52/(h1s-f6)+f5*f62/(h1s-f6)**2.)/rol**2.
578 30 continue
579      dxdp=-dh1dp/(hvs-h1s)-(dhvdp-dh1dp)*(h-h1s)/(hvs-h1s)**2.
580      dvdp=(1-x)*dv1dp+x*dvvdp+(1./rov-1./rol)*dxdp
581      dvdh=(1./rov-1./rol)/(hvs-h1s)
582      rp=-d**2.*dvdp
583      rh=-d**2.*dvdh
584      return
585 10 if(h.gt.6.513e5)go to 40
586      f2=f21+f22*p
587      f3=f31+f32*p
588      rp=(f12+h*h*(f22+f32*h*h))
589      rh=(2*h*f2+4*h*h*h*f3)
590      go to 50
591 40 f5=f51+f52*p
592      f6=f61+f62*p
593      rh=-f5/(h-f6)**2.
594      rp=(f42+f52/(h-f6)+f5*f62/(h-f6)**2.)
595 50 continue
596      return
597      end

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```

```
598     function pilt(pilt0,po1t0,time1)           1
599     cn=400.*time1                               2
600     if(cn.gt.79.99)go to 10                     3
601     pilt=(pilt0+po1t0)/2.+(pilt0-po1t0)/2.*(exp(-400*time1)) 4
602     go to 20                                     5
603     10 pilt=(pilt0+po1t0)/2.                     6
604     20 continue                                 7
605     return                                       8
606     end                                         9
```

B. Momentum Integral Model

```

1 c  solution of transient balance equations for single heated channel      1
2 c                                                                              2
3 c  ***  MOMENTUM INTEGRAL MODEL  ***                                       3
4 c                                                                              4
5 c      june 1983                                                            5
6 c                                                                              6
7      dimension hn(21),ho(21),gn(21),go(21),dn(21),do(21),                7
8      1  x(21),  title(20),ddh(21)                                          8
9      common /data1/  chan1,area,equd,nosn,nots,dz,dt                       9
10     read (5,1000)(title(i),i=1,20)                                        10
11 1000 format (20a4)                                                       11
12     read (5,1010)chan1,area,equd                                         12
13     read(5,1050)is,lsb,io,npp                                             13
14     read(5,1030)time,nots,nosn                                           14
15 1050 format(4i5)                                                         15
16     nosn1=nosn-1                                                         16
17     nosn2=nosn-2                                                         17
18 1010 format(3e10.5)                                                      18
19     read(5,1020)gilt0,hilt0,pilt0,po1t0,powe0                            19
20 1020 format(5e10.5)                                                      20
21 1030 format(e10.6,2i5)                                                  21
22     dz=chan1/(nosn-1)                                                   22
23     dt=time/nots                                                         23
24     if(is.eq.1)go to 130                                                 24
25     do 10 i=1,nosn                                                       25
26     read (5,1040)go(i),ho(i)                                           26
27     10 continue                                                         27
28 1040 format(2e10.5)                                                      28
29 c                                                                              29
30 c                                                                              30
31 c  calculate the steady state density distribution                         31
32 c                                                                              32
33     p=(pilt0+po1t0)/2                                                   33
34     go to 140                                                            34
35 c                                                                              35
36 c  the initial condition is calculated by sub. init                       36
37 c                                                                              37
38 130 call init(isb,gilt0,hilt0,pilt0,po1t0,powe0,                         38
39     1  p,ho,do,go,x)                                                    39
40 c                                                                              40
41 140 write(6,2000)                                                        41
42 c                                                                              42
43     call densi(p,hilt0,din,xin)                                          43
44     do 141 i=1,nosn1                                                     44
45     hn(i)=(ho(i)+ho(i+1))/2.                                           45
46 141 continue                                                             46
47     hn(nosn)=ho(nosn)                                                  47
48     do 142 i=1,nosn                                                     48
49     ho(i)=hn(i)                                                         49
50     call densi(p,ho(i),do(i),x(i))                                       50
51 142 continue                                                             51
52 c  calculate the average of the initial mass flux                         52
53 c                                                                              53
54     sum=0                                                                54
55     do 30 i=2,nosn                                                       55
56     sum=sum+(go(i-1)+go(i))/2.                                          56
57 30 continue                                                             57
58     gavo=sum/(nosn-1)                                                  58
59     dz=chan1/(nosn-1)                                                  59

```

```

60      dt=time/notes
61 c
62 c calculate the friction term of the integral momentum equation
63 c
64      call intrgl(p,go,ho,do,dln,x,fric)
65 2000 format (//," transient solutions of single heated channel by "
66      1"momentum integral model")
67      write(6,2010)(title(i),i=1,20)
68 2010 format (/ ,20a4)
69      write(6,2020)
70 2020 format(/," channel geometry ")
71      write (6,2030)chan1,area,eqd
72 2030 format(1x," channel length=",f6.3," m",
73      1/," flow area = ",e13.6," m**2",
74      1 /," equi diame = ",f6.3," m")
75      write(6,2040)
76 2040 format(/," operating condition ")
77      hiltw=hilt0/1000
78      piltw=pilt0/1.e6
79      poltw=polt0/1.e6
80      powew=powe0/1000
81      write (6,2050)gilt0,hiltw,piltw,poltw,powew
82 2050 format(1x," inlet mass flux =",f8.3," kg/m**2.sec",
83      1 /," inlet enthalpy = ",f9.4," kj/kg",
84      1 /," inlet pressure = ",f8.4," Mpa",
85      1 /," outlet pressure = ",f8.4," Mpa",
86      1 /," power = ",f8.4," kw/m")
87      write(6,3000)time,dt
88 3000 format(/," total transient time = ",e13.6," sec",
89      1 /," time step size = ",e13.6," sec")
90      write(6,3010)
91 3010 format(/," initial condition for transient calculation ")
92      write(6,2070)
93      do 240 i=1,nosn
94      hw=ho(i)/1000
95      write(6,2080)i,go(i),hw,do(i),x(i)
96 240 continue
97 c
98      np=0
99      do 40 i=1,notes
100      np=np+1
101 9000 format(1x," time step ",i5)
102      ttime=dt*i
103      time1=ttime-dt
104      qipt=power(powe0,time1)
105 c
106 c the power should be evaluate at time=i
107 c
108 c calculate the enthalpy of the new time stop i.e. i+1
109 c
110      hin=hilt0
111      c1=dt/(dz*do(1))
112      c2=(ho(2)-ho(1))/2.*go(2)-(hin-ho(1))*go(1)
113      c3=qi*pt*dt/(area*do(1))
114      hn(1)=ho(1)-c1*c2+c3
115      do 50 i1=2,nosn2
116      c1=dt/(dz*do(i1))
117      c2=(ho(i1+1)-ho(i1))/2.*go(i1+1)-(ho(i1-1)-ho(i1))/2.*go(i1)
118      c3=qi*pt*dt/(area*do(i1))
119      hn(i1)=ho(i1)-c1*c2+c3

```

120	50 continue	120
121	c1=dt/(dz*do(nosn1))	121
122	c2=(ho(nosn)-ho(nosn1))*go(nosn)-(ho(nosn2)-ho(nosn1))/2.	122
123	1 *go(nosn1)	123
124	c3=qipt*dt/(area*do(nosn1))	124
125	hn(nosn1)=ho(nosn1)-c1*c2+c3	125
126	hoh=(ho(nosn)+ho(nosn1))/2.	126
127	call densi(p,hoh,da,xa)	127
128	c1=2*dt/(da*dz)	128
129	c2=ho(nosn)*go(nosn)-ho(nosn1)*(go(nosn)+go(nosn1))/2.	129
130	1 -hoh*(go(nosn)-go(nosn1))/2.	130
131	c3=qipt*dt/(area*da)	131
132	hn(nosn)=-hn(nosn1)+2*hoh-2*c1*c2+2*c3	132
133	c	133
134	c calculate the new density distribution from the equation of the state	134
135	call densi(p,hin,din,xin)	135
136	c	136
137	do 60 ii=1,nosn	137
138	call densi(p,hn(ii),dn(ii),x(ii))	138
139	60 continue	139
140	c	140
141	c the mass flux distribution at new time step	141
142	c	142
143	if(io.eq.1)goto 70	143
144	c io.eq.2 flow reduced transient	144
145	tgilt=gilt(gilt0,ttime)	145
146	tpilt=pilt0	146
147	ddh(1)=(dn(1)-din)/(hn(1)-hin)	147
148	do 62 ii=2,nosn	148
149	ddh(ii)=(dn(ii)-dn(ii-1))/(hn(ii)-hn(ii-1))	149
150	62 continue	150
151	call masscal(p,qipt,dn,hn,hin,ddh,tgilt,gn)	151
152	c	152
153	c calculate the average of the mass flux	153
154	c	154
155	sum=0	155
156	do 80 ii=2,nosn	156
157	sum=sum+(gn(ii-1)+gn(ii))/2.	157
158	80 continue	158
159	gavn=sum/(nosn-1)	159
160	predi=(gavn-gavo)/dt+fric	160
161	tpolt=tpilt+predi	161
162	goto 90	162
163	c io.eq.1 pressure transient	163
164	70 continue	164
165	tpilt=pilt(pilt0,polt0,time1)	165
166	tpolt=polt(polt0,time1)	166
167	predi=tpilt-tpolt	167
168	gavn=gavo+dt/chan1*(predi-fric)	168
169	c	169
170	c calculate the inlet mass flux	170
171	c	171
172	ddh(1)=(dn(1)-din)/(hn(1)-hin)	172
173	do 71 ii=2,nosn	173
174	ddh(ii)=(dn(ii)-dn(ii-1))/(hn(ii)-hn(ii-1))	174
175	71 continue	175
176	sum1=0	176
177	sum2=0	177
178	qipt=power(powe0,ttime)	178
179	gamm0=1	179

```

180      del t0=0
181      c1=ddh(1)/dn(1)
182      alpha=(1+(hn(1)-hin)*c1)/(1+(hn(1)-hn(2))/2.*c1)
183      beta=-dz*c1*qipt/(area*(1+(hn(1)-hn(2))/2.*c1))
184      gamm1=gamm0*alpha
185      del t1=del t0*alpha+beta
186      sum1=sum1+(gamm1+gamm0)/2.
187      sum2=sum2+(del t0+del t1)/2.
188      gamm0=gamm1
189      del t0=del t1
190      do 100 i1=3,nosn1
191      c1=ddh(i1-1)/(dn(i1-1)*2)
192      alpha=(1+(hn(i1-1)-hn(i1-2))*c1)/(1+(hn(i1-1)-hn(i1))*c1)
193      beta=-2*dz*c1*qipt/(area*(1+(hn(i1-1)-hn(i1))*c1))
194      gamm1=alpha*gamm0
195      del t1=alpha*del t0+beta
196      sum1=sum1+(gamm1+gamm0)/2.
197      sum2=sum2+(del t1+del t0)/2.
198      gamm0=gamm1
199      del t0=del t1
200 100 continue
201      c1=ddh(nosn1)/dn(nosn1)
202      alpha=(1+(hn(nosn1)-hn(nosn2))/2.*c1)/(1+(hn(nosn1)-hn(nosn
203      1 )*c1))
204      beta=-dz*c1*qipt/(area*(1+(hn(nosn1)-hn(nosn))*c1))
205      gamm1=alpha*gamm0
206      del t1=del t0*alpha+beta
207      sum1=sum1+(gamm1+gamm0)/2.
208      sum2=sum2+(del t0+del t1)/2.
209      ave1=sum1/(nosn-1)
210      ave2=sum2/(nosn-1)
211      tgilt =(gavn-ave2)/ave1
212 9900 format(1x,"gavn,ave2,ave1",3(2x,e10.5),2x,15)
213      call masscal(p,qipt,dn,hn,hin,ddh,tgilt,gn)
214 c
215 c calculate the friction term of the integral momemtum equation
216 c
217 90 continue
218 do 91 i1=1,nosn
219 if(gn(i1).lt.0.0)go to 45
220 91 continue
221 call intrgl(p,gn,hn,dn,din,x,fric)
222 do 110 i1=1,nosn
223 ho(i1)=hn(i1)
224 go(i1)=gn(i1)
225 do(i1)=dn(i1)
226 110 continue
227 p=(tpilt+tpolt)/2.
228 gavo=gavn
229 if(np.ne.npp)go to 41
230 if(gn(1).lt.0.0)write(0,9999)gn(1)
231 9999 format(1x," something wrong gn(1)=",e13.6)
232 write(0,9000)i
233 np=0
234 predw=predi/1000
235 qiptw=qipt/1000
236 write(6,2060)ttime,predw,qiptw,gavn
237 write(6,2070)
238 do 120 i1=1,nosn
239 hw=hn(i1)/1000

```

240	gr=gn(i1)/gilt0	240
241	write(6,2080)i1,gn(i1),hw,dn(i1),x(i1),gr	241
242	120 continue	242
243	2060 format(//," transient time=",e13.6," sec ",	243
244	1 /," pressure drop = ",f8.4," kpa",	244
245	1 /," power = ",f8.4," kw/m",	245
246	1 /," average mass flux =",f9.4," kg/m**2.sec")	246
247	2070 format(/," mass flux enthalpy density quality",	247
248	1 /," (kg/m**2.sec) (kj/kg) (kg/m**3)"	248
249	2080 format(1x,i3.2x,f8.3,6x,f8.3,4x,f8.3,2x,f6.3,3x,f7.4)	249
250	41 continue	250
251	40 continue	251
252	go to 46	252
253		253
254	45 write(0,5000)i,i1,gn(i1)	254
255	write(6,5000)i,i1,gn(i1)	255
256	5000 format(1x," flow rate is negative at time step no"	256
257	1 .1x,i5./," at node ",i4," gn=",e13.6)	257
258	46 continue	258
259	stop	259
260	end	260
261	c	261
262	c *****	262
263	c	263
264	c	264
265	c *****	265
266	c	266


```

267      subroutine intrgl(p,g,h,d,din,xa,fric)                1
268      dimension h(21),g(21),d(21),xa(21)                  2
269      common /data1/ chan1,area,equd,nosn,nots,dz,dt       3
270      data conv1,conv2,vics,g1/737.4643,1.4504e-4,91.7e-6,9.8/ 4
271      data vics1/19.73e-6/                                  5
272 c                                                    6
273      fric=0                                                7
274      nosn1=nosn-1                                          8
275 c acceleration pressure drop                             9
276 c                                                    10
277      fric=g(nosn)**2/d(nosn)-g(1)**2/din                 11
278 c                                                    12
279 c thm: two phase pressure drop multiplier                 13
280 c                                                    14
281      thm=1                                                15
282 c                                                    16
283 c get the saturation liquid and vapor density rho1,rov  17
284 c                                                    18
285      do 10 i=1,nosn1                                       19
286      ga=(g(i+1)+g(i))/2.                                   20
287 c                                                    21
288 c elavation pressure drop                                22
289 c                                                    23
290      fric=fric+d(i)*g1*dz                                  24
291      dpc=d(i)                                             25
292 c                                                    26
293 c if x>0 calculate the two phase multiplier              27
294 c eq(5.209) p.230 Lahey and Moody was used              28
295 c                                                    29
296      if(xa(i).eq.-1.) goto 20                               30
297      if(xa(i).eq.1.) go to 30                               31
298      call satt(p,hls,hvs)                                   32
299      call lden(p,hls,rho1)                                  33
300      call vden(p,hvs,rov)                                   34
301      dpc=rho1                                              35
302      gtest=ga*conv1/1.0e6                                   36
303      if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest    37
304      1 -0.000714*p*conv2*gtest                             38
305      if(gtest.ge.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest  39
306      1 +0.00028*p/gtest*conv2                             40
307      thm=c1*(1.2*(rho1/rov-1)*xa(i)**0.824)+1.0          41
308 c                                                    42
309 c use Blasius for friction factor                         43
310 c                                                    44
311      20 fact=0.184*(ga*equd/vics)**(-0.20)                 45
312      go to 40                                              46
313      30 fact=0.184*(ga*equd/vics1)**-.20                   47
314      40 fric=fric+thm*fact*ga**2/(2*equd*dpc)*dz         48
315      10 continue                                          49
316      return                                              50
317      end                                                  51
318 c                                                    52
319 c *****                                                53
320 c                                                    54

```

321	subroutine init(is1,gilt0,hilt0,pilt0,polt0,powe0,	1
322	1 p1,h,d,g,x)	2
323	dimension h(21),d(21),p(21),g(21),x(21)	3
324	common /data1/ chan1,area,equd,nosn,nots,dz,dt	4
325	call densi(pilt0,hilt0,d(1),x(1))	5
326	h(1)=hilt0	6
327	p(1)=pilt0	7
328	g(1)=gilt0	8
329	i=1	9
330	n1=1	10
331	90 do 10 i=2,nosn	11
332	h1=h(i-1)+powe0*dz/(area*gilt0)	12
333	p1=p(i-1)	13
334	call densi(p1,h1,d(i),x(i))	14
335	60 ha=(h1+h(i-1))/2.	15
336	call densi(p1,ha,da,x1)	16
337	call fric1(x1,p1,gilt0,da,fod)	17
338	p2=p(i-1)-fod*gilt0**2.*dz/(2*equd)	18
339	1 -da*9.80*dz-gilt0**2.*(1/d(i)-1/d(i-1))	19
340	call densi(p2,h1,d(i),x(i))	20
341	call densi(p2,ha,da,x1)	21
342	call fric1(x1,p2,gilt0,da,fod)	22
343	31 h1=h(i-1)+powe0*dz/(area*gilt0)+(p2-p(i-1)+	23
344	1 (fod*gilt0**2*dz)/(2*equd))/da	24
345	if(abs((p2-p1)/p2).le.1.0e-8)go to 40	25
346	n=n+1	26
347	if(n.gt.50)goto 50	27
348	p1=p2	28
349	goto 60	29
350	40 p(i)=p2	30
351	h(i)=h1	31
352	g(i)=gilt0	32
353	10 continue	33
354	if(is1.eq.1)polt0=p(nosn)	34
355	if(abs((p(nosn)-polt0)/polt0).le.1.0e-4)go to 160	35
356	if(n1.gt.50) go to 80	36
357	n1=n1+1	37
358	gilt0=gilt0*((p(nosn)-p(1))/(polt0-pilt0))*0.5	38
359	goto 90	39
360	80 write(6,2110)	40
361	2110 format(1x," iteration not converge for pressure b.c.")	41
362	stop	42
363	50 write(6,2090)i	43
364	2090 format(1x,"iteration not converge for spatial node ",i2)	44
365	stop	45
366	160 p1=(polt0+pilt0)/2.	46
367	return	47
368	end	48

369	subroutine fric1(x1,p,g,d,fod)	1
370	common /data1/ chan1,area,equd,nosn,nots,dz,dt	2
371	data conv1,conv2,vics/737.4643,1.4504e-4,91.7e-6/	3
372		4
373	c use McAdames correlation to calculate the friction factor	5
374		6
375	fric=0.184*(g*equd/vics)**(-0.2)	7
376	if(x1.eq.-1.0)go to 10	8
377	call satt(p,h1s,hvs)	9
378	call lden(p,h1s,rol)	10
379	call vden(p,hvs,rov)	11
380	gtest=g*conv1/1.0e6	12
381	if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest	13
382	1 -0.000714*p*conv2*gtest	14
383	if(gtest.gt.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest	15
384	1+0.00028*p/gtest*conv2	16
385	thm=c1*(1.2*(rol/rov-1)*x1**0.824)+1.0	17
386	fod=thm*fric/rol	18
387	go to 21	19
388	10 fod=fric/d	20
389	21 continue	21
390	return	22
391	end	23

```
392 function power(powe0,time1)
393 power=powe0
394 return
395 end
```

```
1
2
3
4
```

```
396      function pilt(pilt0,polt0,time1)           1
397      cn=400.*time1                             2
398      if(cn.gt.79.99)go to 10                   3
399      pilt=(pilt0+pol1t0)/2.+(pilt0-pol1t0)/2.*(exp(-400*time1)) 4
400      go to 20                                   5
401  10 pilt=(pilt0+pol1t0)/2.                     6
402  20 continue                                   7
403      return                                     8
404      end                                       9
```

```
405 function polt(polto,time1)
406 polt=polto
407 return
408 end
```

```
1
2
3
4
```

```
409     function gilt(gilt0,time1)
410     gilt=gilt0
411     return
412     end
413 c
```

```
1
2
3
4
5
```

414	subroutine densi(p,h,d,x)	1
415	c	2
416	call satt(p,h1s,hvs)	3
417	if(h.le.h1s) go to 10	4
418	if(h.gt.hvs) go to 20	5
419	call lden(p,h1s,ro1)	6
420	call vden(p,hvs,rov)	7
421	x=(h-h1s)/(hvs-h1s)	8
422	sv1=1./ro1	9
423	svv=1./rov	10
424	sv=sv1*(1-x)+svv*x	11
425	d=1./sv	12
426	return	13
427	10 x=-1.	14
428	call lden(p,h,ro1)	15
429	d=ro1	16
430	return	17
431	20 x=1.0	18
432	call vden(p,hvs,rov)	19
433	d=rov	20
434	return	21
435	end	22

436	subroutine lden(p,h,rol)	1
437	data f11,f12/999.65,4.9737e-7/	2
438	data f21,f22/-2.5847e-10,6.1767e-19/	3
439	data f31,f32/1.2696e-22,-4.9223e-31/	4
440 c		5
441	data f41,f42/1488.64,1.3389e-6/	6
442	data f51,f52/1.4695e9,8.85736/	7
443	data f61,f62/3.20372e6,1.20483e-2/	8
444 c		9
445	if(h.gt.6.513e5) go to 10	10
446	f1=f11+f12*p	11
447	f2=f21+f22*p	12
448	f3=f31+f32*p	13
449	rol=f1+h*h*(f2+f3*h*h)	14
450	go to 20	15
451	10 f4=f41+f42*p	16
452	f5=f51+f52*p	17
453	f6=f61+f62*p	18
454	rol=f4+f5/(h-f6)	19
455	20 return	20
456	end	21

```
457      subroutine vden(p,h,rov)                                1
458 c                                                                    2
459      data g00,g01,g02,g10,g11,g12/-5.1026e-5,1.1208e-10,    3
460 1      -4.4506e5,-1.6893e-10,-3.3980e-17,2.3058e-1/        4
461 c                                                                    5
462      g0p=g00+g01*p+g02/p                                     6
463      g1p=g10+g11*p+g12/p                                     7
464      rov=1./(g0p+g1p*h)                                       8
465      return                                                    9
466      end                                                         10
```

```

467      subroutine satt(p,hls,hvs)                                1
468      data tsc1,tsc2, tsexp /9.0395, 255.2, 0.223/           2
469      data cps1,cps2, cpsexp /9.5875e2, .00132334, -0.8566/   3
470 c      cps2 = -cpsexp * tcrinv                                4
471 c      for hls and hvs                                       5
472      data h10,h11,h12,h13,h14,h15/5.7474718e5,2.0920624e-1,   6
473      1 -2.8051070e-8,2.3809828e-15,-1.0042660e-22,1.6586960e-30/ 7
474      data hv0,hv1,hv2,hv3,hv4/2.7396234e6,3.758844e-2,      8
475      1 -7.1639909e-9,4.2002319e-16,-9.8507521e-24 /        9
476 c
477      tsat = tsc1* p**tsexp                                    10
478      hvs = hv0 + p*(hv1 + p*(hv2 + p*(hv3 + p*hv4)))        11
479      hls = h10 + p*(h11 + p*(h12 + p*(h13 + p*(h14 + p*h15)))) 12
480      pinv = 1./ p                                            13
481      dtsdp = tsat*tsexp*pinv                                 14
482      tsat = tsat + tsc2                                      15
483      return                                                  16
484      end                                                       17

```

18

485	subroutine masscal(p,qipt,dn,hn,hin,ddh,tgilt,gn)	1
486	dimension dn(21),hn(21),gn(21),ddh(21)	2
487	common /data1/ chan1,area,equd,nosn,nots,dz,dt	3
488	nosn1=nosn-1	4
489	nosn2=nosn-2	5
490	gn(1)=tgilt	6
491	c1=ddh(1)/dn(1)	7
492	beta=-dz*c1*qi pt/(area*(1+(hn(1)-hn(2))/2.*c1))	8
493	alpha=(1+(hn(1)-hin)*c1)/(1+(hn(1)-hn(2))/2.*c1)	9
494	gn(2)=gn(1)*alpha+beta	10
495	do 100 i1=3,nosn1	11
496	c1=ddh(i1-1)/(dn(i1-1)*2)	12
497	alpha=(1+(hn(i1-1)-hn(i1-2))*c1)/(1+(hn(i1-1)-hn(i1))*c1)	13
498	beta=-2*dz*c1*qi pt/(area*(1+(hn(i1-1)-hn(i1))*c1))	14
499	gn(i1)=gn(i1-1)*alpha+beta	15
500	100 continue	16
501	c1=ddh(nosn1)/dn(nosn1)	17
502	alpha=(1+(hn(nosn1)-hn(nosn2))/2.*c1)/(1+(hn(nosn1)-hn(nosn1	18
503	1)*c1)	19
504	beta=-dz*c1*qi pt/(area*(1+(hn(nosn1)-hn(nosn))*c1))	20
505	gn(nosn)=alpha*gn(nosn1)+beta	21
506	return	22
507	end	23

C. Single Velocity Model

1	c	solution of transient balance equations for single heated channel	1
2	c		2
3	c	*** SINGLE VELOCITY MODEL ***	3
4	c		4
5	c	june 1983	5
6	c		6
7		dimension hn(21),ho(21), dn(21),do(21),	7
8	1	x(21),xa(21),da(21), title(20)	8
9		common /data1/ chan1,area,equd,nosn,nots,dz,dt	9
10		read (5,1000)(title(i),i=1,20)	10
11	1000	format (20a4)	11
12		read (5,1010)chan1,area,equd	12
13		read(5,1050)is, isb, io,npp	13
14		read(5,1030)time,nots,nosn	14
15	1050	format(4i5)	15
16	1010	format(3e10.5)	16
17		read(5,1020)gilt0,hilt0,pilt0,polto,powe0	17
18	1020	format(5e10.5)	18
19	1030	format(e10.6,2i5)	19
20		dz=chan1/(nosn-1)	20
21		dt=time/nots	21
22		if(is.eq.1)go to 130	22
23		do 10 i=1,nosn	23
24		read (5,1040)ho(i)	24
25	10	continue	25
26	1040	format(e10.5)	26
27	c		27
28	c		28
29	c		29
30		p=(pilt0+polto)/2	30
31		go to 140	31
32	c		32
33	c	the initial condition is calculated by sub. init	33
34	c		34
35	130	call init(isb,gilt0,hilt0,pilt0,polto,powe0,	35
36	1	p,ho,do,x)	36
37	c		37
38	140	write(6,2000)	38
39	c		39
40	c	calculate the steady state density distribution	40
41	c		41
42		do 20 i=1,nosn	42
43		call densi(p,ho(i),do(i),x(i))	43
44	20	continue	44
45	c		45
46		do 141 i=2,nosn	46
47		hoh=(ho(i)+ho(i-1))/2.	47
48		call densi(p,hoh,da(i),xa(i))	48
49	141	continue	49
50		go=gilt0	50
51		call intrgl(p,go,ho,do,da,xa,fric)	51
52	2000	format (1h1," transient solutions of single heated channel "	52
53	1	," by single velocity model ")	53
54		write(6,2010)(title(i),i=1,20)	54
55	2010	format (/,20a4)	55
56		write(6,2020)	56
57	2020	format(/," channel geometry ")	57
58		write (6,2030)chan1,area,equd	58
59	2030	format(1x," channel length=",f6.3," m",	59

```

60      1/," flow area = ",e13.6," m**2",
61      1      /," equi diame = ",f6.3," m")
62      write(6,2040)
63      2040 format(/," operating condition ")
64      hiltw=hilt0/1000
65      piltw=pilt0/1000000
66      poltw=polt0/1.e6
67      powew=powe0/1000
68      write (6,2050)gilt0,hiltw,piltw,poltw,powew
69      2050 format(1x," inlet mass flux = ",f9.3," kg/m**2.sec",
70      1      /," inlet enthalpy = ",f9.3," kj/kg",
71      1      /," inlet pressure = ",f8.4," Mpa",
72      1      /," outlet pressure = ",f8.4," Mpa",
73      1      /," power = ",f8.4," kw/m")
74      write(6,3200)time,dt
75
76      3200 format(/," total transient= ",e13.6," sec",
77      1      /" time step size = ",e13.6," sec")
78      write(6,3000)
79      3000 format(/," initial condition for transient calculation ")
80      write(6,2070)
81      do 200 i=1,nosn
82      hw=ho(i)/1000
83      write(6,2080)i,hw,do(i),x(i)
84      200 continue
85 c
86      np=0
87      do 40 i=1 ,nots
88      6000 format(1x," time step ",i5)
89      np=np+1
90      ttime=dt*i
91      time1=ttime-dt
92      qipt=power(powe0,time1)
93 c
94 c the power should be evaluate at time=i
95 c
96 c calculate the enthalpy of the new time step i.e. i+1
97 c
98      hn(1)=hilt0
99      do 50 i1=2,nosn
100      alpha=go*dt/(da(i1)*dz)
101      beta=(2*dt*qipt/area)/(da(i1)*(1+alpha))
102      hn(i1)=ho(i1-1)-(hn(i1-1)-ho(i1))*((1-alpha)/(1+alpha))+beta
103      50 continue
104 c
105 c calculate the new density distribution from the equation of the state
106 c
107      do 60 i1=1,nosn
108      call densi(p,hn(i1),dn(i1),x(i1))
109      60 continue
110      do 61 i1=2,nosn
111      hnh=(hn(i1)+hn(i1-1))/2.
112      call densi(p,hnh,da(i1),xa(i1))
113      61 continue
114 c
115 c the mass flux distribution at new time step
116 c
117      if(io.eq.1)goto 70
118 c io.eq.2 flow reduced transient
119      gn=gilt(gilt0,ttime)

```

120	tpilt=pilt0	120
121	predi=(gn-go)/dt+fric	121
122	tpolt=tpilt+predi	122
123	goto 90	123
124	c 10.eq.1 pressure transient	124
125	70 continue	125
126	tpilt=pilt(pilt0,polt0,time1)	126
127	tpolt=polt(polt0,time1)	127
128	predi=tpilt-tpolt	128
129	gn=go+dt/chan1*(predi-fric)	129
130	c	130
131	c calculate the friction term of the integral momentum equation	131
132	c	132
133	90 continue	133
134	call intrgl(p,gn,hn,dn,da,xa,fric)	134
135	do 110 i1=1,nosn	135
136	ho(i1)=hn(i1)	136
137	do(i1)=dn(i1)	137
138	110 continue	138
139	go=gn	139
140	if(np.ne.npp)go to 40	140
141	c write(0,6000)i	141
142	np=0	142
143	predw=predi/1000	143
144	qiptw=qipt/1000	144
145	gnr=gn/gilt0	145
146	write(6,2060)ttime,predw,qiptw,gn,gnr	146
147	write(6,2070)	147
148	do 120 i1=1,nosn	148
149	hw=hn(i1)/1000	149
150	write(6,2080)i1,hw,dn(i1),x(i1)	150
151	120 continue	151
152	2060 format(//," transient time=",f8.4," sec ",	152
153	1 /," pressure drop = ",f8.4," kpa",	153
154	1 /," power = ",f8.4," kw/m",	154
155	1 /," mass flux = ",f9.3," kg/m**2.sec",3x,f8.4)	155
156		156
157	2070 format(/," enthalpy density ", " quality ",	157
158	1 /," (kj/kg) (kg/m**3) ")	158
159	2080 format(1x,i2,2x,f9.3,5x,f8.4,4x,f6.3)	159
160	41 continue	160
161	40 continue	161
162	end	162

163	subroutine intrg1(p,g,h,d,da,xa,fric)	1
164	dimension h(21),d(21),xa(21),da(21)	2
165	common /data1/ chan1,area,equd,nosn,nots,dz,dt	3
166	data conv1,conv2,vics,g1/737.4643,1.4504e-4,91.7e-6,9.8/	4
167	data vics1/19.73e-6/	5
168	c	6
169	fric=0	7
170	c acceleration pressure drop	8
171	c	9
172	fric=g**2/d(nosn)-g**2/d(1)	10
173	c	11
174	c thm: two phase pressure drop multiplier	12
175	c	13
176	thm=1	14
177	c	15
178	c get the saturation liquid and vapor density rol,rov	16
179	c	17
180	call satt(p,hls,hvs)	18
181	call lden(p,hls,rol)	19
182	call vden(p,hvs,rov)	20
183	do 10 i=2,nosn	21
184	c	22
185	c elavation pressure drop	23
186	c	24
187	fric=fric+da(i)*g1*dz	25
188	dpc=da(i)	26
189	c	27
190	c if x>0 calculate the two phase multiplier	28
191	c eq(5.209) p.230 Lahey and Moody was used	29
192	c	30
193	if(xa(i).eq.-1) goto 20	31
194	if(xa(i).eq.1.0) go to 30	32
195	dpc=rol	33
196	gtest=g*conv1/1.0e6	34
197	if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest	35
198	1 -0.000714*p*conv2*gtest	36
199	if(gtest.ge.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest	37
200	1 +0.00028*p/gtest*conv2	38
201	thm=c1*(1.2*(rol/rov-1)*xa(21)**0.824)+1.0	39
202	c	40
203	c	41
204	20 fact=0.184*(g*equd/vics)**(-0.20)	42
205	go to 40	43
206	30 fact=0.184*(g*equd/vics1)**(-0.20)	44
207	40 fric=fric+thm*fact*g**2/(2*equd*dpc)*dz	45
208	10 continue	46
209	return	47
210	end	48

211	subroutine init(is1,gilt0,hilt0,pilt0,po1t0,po2t0,	1
212	1 p1,h,d,x)	2
213	dimension h(21),d(21),p(21),g(21),x(21)	3
214	common /data1/ chan1,area,equd,nosn,nots,dz,dt	4
215	tsub=600	5
216	call densi(pilt0,hilt0,d(1),x(1))	6
217	h(1)=hilt0	7
218	p(1)=pilt0	8
219	g(1)=gilt0	9
220	i=1	10
221	n1=1	11
222	90 do 10 i=2,nosn	12
223	h1=h(i-1)+po2t0*dz/(area*gilt0)	13
224	p1=p(i-1)	14
225	call densi(p1,h1,d(i),x(i))	15
226	60 ha=(h1+h(i-1))/2.	16
227	call densi(p1,ha,da,x1)	17
228	call fric1(x1,p1,gilt0,da,fod)	18
229	p2=p(i-1)-fod*gilt0**2.*dz/(2*equd)	19
230	1 -da*9.80*dz-gilt0**2.*(1/d(i)-1/d(i-1))	20
231	call densi(p2,h1,d(i),x(i))	21
232	call densi(p2,ha,da,x1)	22
233	call fric1(x1,p2,gilt0,da,fod)	23
234	31 h1=h(i-1)+po2t0*dz/(area*gilt0)+(p2-p(i-1)+	24
235	1 (fod*gilt0**2*dz)/(2*equd))/da	25
236	if(abs((p2-p1)/p2).le.1.0e-8)go to 40	26
237	n=n+1	27
238	if(n.gt.50)goto 50	28
239	p1=p2	29
240	goto 60	30
241	40 p(i)=p2	31
242	h(i)=h1	32
243	g(i)=gilt0	33
244	10 continue	34
245	if(is1.eq.1)po1t0=p(nosn)	35
246	if(abs((p(nosn)-po1t0)/po1t0).le.1.0e-4)go to 160	36
247	if(n1.gt.50) go to 80	37
248	n1=n1+1	38
249	gilt0=gilt0*((p(nosn)-p(1))/(po1t0-pilt0))**0.5	39
250	goto 90	40
251	80 write(6,2110)	41
252	2110 format(1x," iteration not converge for pressure b.c.")	42
253	stop	43
254	50 write(6,2090)1	44
255	2090 format(1x,"iteration not converge for spatial node ",i2)	45
256	stop	46
257	160 p1=(po1t0+pilt0)/2.	47
258	return	48
259	end	49

260	subroutine fric1(x1,p,g,d,fod)	1
261	common /data1/ chan1,area,equd,nosn,nots,dz,dt	2
262	data conv1,conv2,vics/737.4643e0,1.4504e-4,91.7e-6/	3
263		4
264 c	use McAdames correlation to calculate the friction factor	5
265		6
266	fric=0.184*(g*equd/vics)**(-0.2)	7
267	if(x1.eq.-1.0)go to 10	8
268	call satt(p,hls,hvs)	9
269	call lden(p,hls,rol)	10
270	call vden(p,hvs,rov)	11
271	gtest=g*conv1/1.0e6	12
272	if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest	13
273	1 -0.000714*p*conv2*gtest	14
274	if(gtest.gt.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest	15
275	1+0.00028*p/gtest*conv2	16
276	thm=c1*(1.2*(rol/rov-1)*x1**0.824)+1.0	17
277	fod=thm*fric/rol	18
278	go to 21	19
279	10 fod=fric/d	20
280	21 continue	21
281	return	22
282	end	23

```
283 function power(powe0,time1)
284 power=powe0
285 return
286 end
```

```
1
2
3
4
```

```
287 function polt(polto,time1)
288 polt=polto
289 return
290 end
```

```
1
2
3
4
```

```
291 function gilt(gilt0,time1)
292 gilt=gilt0
293 return
294 end
295 c
```

```
1
2
3
4
5
```

296	subroutine densi(p,h,d,x)	1
297	c	2
298	call satt(p,h1s,hvs)	3
299	if(h.le.h1s) go to 10	4
300	if(h.gt.hvs) go to 20	5
301	call lden(p,h1s,ro1)	6
302	call vden(p,hvs,rov)	7
303	x=(h-h1s)/(hvs-h1s)	8
304	sv1=1./ro1	9
305	svv=1./rov	10
306	sv=sv1*(1-x)+svv*x	11
307	d=1./sv	12
308	return	13
309	10 x=-1.	14
310	call lden(p,h,ro1)	15
311	d=ro1	16
312	return	17
313	20 x=1.0	18
314	call vden(p,hvs,rov)	19
315	d=rov	20
316	return	21
317	end	22

318	subroutine lden(p,h,rol)	1
319	data f11,f12/999.65,4.9737e-7/	2
320	data f21,f22/-2.5847e-10,6.1767e-19/	3
321	data f31,f32/1.2696e-22,-4.9223e-31/	4
322	c	5
323	data f41,f42/1488.64,1.3389e-6/	6
324	data f51,f52/1.4695e9,8.85736/	7
325	data f61,f62/3.20372e6,1.20483e-2/	8
326	c	9
327	if(h.gt.6.513e5) go to 10	10
328	f1=f11+f12*p	11
329	f2=f21+f22*p	12
330	f3=f31+f32*p	13
331	rol=f1+h*h*(f2+f3*h*h)	14
332	go to 20	15
333	10 f4=f41+f42*p	16
334	f5=f51+f52*p	17
335	f6=f61+f62*p	18
336	rol=f4+f5/(h-f6)	19
337	20 return	20
338	end	21


```
339      subroutine vden(p,h,rov)                                1
340 c                                                                    2
341      data g00,g01,g02,g10,g11,g12/-5.1026e-5,1.1208e-10,      3
342      1      -4.4506e5,-1.6893e-10,-3.3980e-17,2.3058e-1/      4
343 c                                                                    5
344      g0p=g00+g01*p+g02/p                                       6
345      g1p=g10+g11*p+g12/p                                       7
346      rov=1./(g0p+g1p*h)                                         8
347      return                                                       9
348      end                                                           10
```

```

349      subroutine satt(p,hls,hvs)                                1
350      data tsc1,tsc2, tsexp /9.0395, 255.2, 0.223/           2
351      data cps1,cps2, cpsexp /9.5875e2, .00132334, -0.8566/   3
352 c      cps2 = -cpsexp * tcrinv                                4
353 c      for hls and hvs                                        5
354      data h10,h11,h12,h13,h14,h15/5.7474718e5,2.0920624e-1,   6
355      t      -2.8051070e-8,2.3809828e-15,-1.0042660e-22,1.6586960e-30/ 7
356      data hv0,hv1,hv2,hv3,hv4/2.7396234e6,3.758844e-2,      8
357      t      -7.1639909e-9,4.2002319e-16,-9.8507521e-24 /    9
358 c
359      tsat = tsc1* p**tsexp                                    10
360      hvs = hv0 + p*(hv1 + p*(hv2 + p*(hv3 + p*hv4)))        11
361      hls = h10 + p*(h11 + p*(h12 + p*(h13 + p*(h14 + p*h15)))) 12
362      pinv = 1./ p                                            13
363      dtstdp = tsat*tsexp*pinv                                14
364      tsat = tsat + tsc2                                      15
365      return                                                  16
366      end                                                       17

```

```
367     function pilt(pilt0,polt0,time1)           1
368     cn=400.*time1                             2
369     if(cn.gt.79.99)go to 10                   3
370     pilt=(pilt0+pol1t0)/2.+(pilt0-pol1t0)/2.*(exp(-400*time1)) 4
371     go to 20                                   5
372 10 pilt=(pilt0+pol1t0)/2.                   6
373 20 continue                                  7
374     return                                    8
375     end                                       9
```

D. Channel Integral Model

1 c	solution of transient balance equations for single heated channel	1
2 c		2
3 c	*** CHANNEL INTEGRAL MODEL ***	3
4 c		4
5 c	june 1983	5
6 c		6
7	dimension hn(21),ho(21),gn(21),go(21),dn(21),do(21),	7
8	1 xn(21),xo(21),hn1(21),dn1(21),gamma(21),beta(21),title(20)	8
9	common /data1/ chan1,area,equd,nosn,nots,dz,dt	9
10	read (5,1000)(title(i),i=1,20)	10
11	1000 format (20a4)	11
12	read (5,1010)chan1,area,equd	12
13	read(5,1050)is,isb,io,npp	13
14	read(5,1030)time,nots,nosn	14
15	1050 format(4i5)	15
16	1010 format(3e10.5)	16
17	read(5,1020)gilt0,hilt0,pilt0,polto,powe0	17
18	1020 format(5e10.5)	18
19	1030 format(e10.6,2i5)	19
20	dz=chan1/(nosn-1)	20
21	dt=time/nots	21
22	if(is.eq.1)go to 130	22
23	do 10 i=1,nosn	23
24	read (5,1040)go(i),ho(i)	24
25	10 continue	25
26	1040 format(2e10.5)	26
27 c		27
28 c		28
29	p=(pilt0+polto)/2.	29
30	go to 140	30
31 c		31
32 c	the initial condition is calculated by sub. init	32
33 c		33
34	130 call init(isb,gilt0,hilt0,pilt0,polto,powe0,	34
35	1 p,ho,do,go,xo)	35
36 c		36
37 c		37
38 c		38
39 c	calculate the steady state density distribution	39
40 c		40
41	p=(pilt0+polto)/2.	41
42	do 20 i=1,nosn	42
43	call densi(p,ho(i),do(i),xo(i))	43
44	20 continue	44
45 c		45
46	140 write(6,2000)	46
47 c		47
48 c	calculate the average of the initial mass flux	48
49 c		49
50	sum=0	50
51	do 30 i=2,nosn	51
52	sum=sum+(go(i-1)+go(i))/2.	52
53	30 continue	53
54	gavo=sum/(nosn-1)	54
55	nosn1=nosn-1	55
56	sum=ho(1)/2.	56
57	do 40 j=2,nosn1	57
58	sum=sum+ho(j)	58
59	40 continue	59

```

60      sum=sum+ho(nosn)/2
61      havo=sum/nosn1
62      deltho=ho(nosn)-hilt0
63      do 50 j=1,nosn
64      beta(j)=(ho(j)-hilt0)/(havo-hilt0)
65      50 continue
66      dz=chan1/(nosn-1)
67      dt=time/notes
68 c
69 c
70 2000 format (//," transient solutions of single heated channel by ",
71 1"channel integral model")
72 write(6,2010)(title(i),i=1,20)
73 2010 format (/,20a4)
74 write(6,2020)
75 2020 format(/," channel geometry ")
76 write (6,2030)chan1,area,eqd
77 2030 format(1x," channel length=",f6.3," m",
78 1/," flow area = ",e13.6," m**2",
79 1 /," equi diame = ",f6.3," m")
80 write(6,2040)
81 2040 format(/," operating condition ")
82 hiltw=hilt0/1000
83 piltw=pilt0/1.e6
84 poltw=polto/1.e6
85 powew=powe0/1000
86 write (6,2050)gilt0,hiltw,piltw,poltw,powew
87 2050 format(1x," inlet mass flux =",f8.3," kg/m**2.sec",
88 1 /," inlet enthalpy = ",f9.4," kj/kg",
89 1 /," inlet pressure = ",f8.4," Mpa",
90 1 /," outlet pressure = ",e13.6," Mpa",
91 1 /," power = ",f8.4," kw/m")
92 write(6,3000)time,dt
93 3000 format(/," total transient time = ",e13.6," sec",
94 1 /," time step size = ",e13.6," sec")
95 write(6,3010)
96 3010 format(/," initial condition for transient calculation ")
97 write(6,3200)
98 do 240 i=1,nosn
99 hw=ho(i)/1000
100 write(6,3300)i,go(i),hw,do(i),xo(i)
101 240 continue
102 havn= havo*1.0001
103 havn=havo+1.
104 do 60 j=1,nosn
105 hn(j)=hilt0+beta(j)*(havn-hilt0)
106 call densi(p,hn(j),dn(j),xn(j))
107 60 continue
108 call cst(havo,ho,do,havn,hn,dn,deltho,gamma,hilt0,cst1
109 1,cst2,cst3)
110 c
111 np=0
112 do 70 i=1,notes
113 n=1
114 np=np+1
115 ttime=dt*i
116 time1=ttime-dt
117 qipt=power(powe0,time1)
118 thilt=hilt0
119 if(io.eq.1)go to 80

```

120	c		120
121	c	flow reduce transient	121
122	c		122
123		tgilt=gilt(gilt0,time1)	123
124		const=cst1/(cst3*gamma(nosn)*chan1)	124
125		gavo=(tgilt-const*qipt*chan1/area)/(1-const*deltho)	125
126	80	havn=havo+dt/(cst3*chan1)*(qipt*chan1/area-gavo*deltho)	126
127		do 110 j=1,nosn	127
128		hn(j)=thilt+beta(j)*(havn-thilt)	128
129		call densi(p,hn(j),dn(j),xn(j))	129
130	110	continue	130
131	180	continue	131
132		call cst(havo,ho,do,havn,hn,dn,deltho,	132
133	1	gamma,thilt,cst1,cst2,cst3)	133
134		if(io.eq.1)go to 81	134
135		const=cst1/(cst3*gamma(nosn)*chan1)	135
136		gavo=(tgilt-const*qipt*chan1/area)/(1-const*deltho)	136
137	81	havn1=havo+dt/(cst3*chan1)*(qipt*chan1/area-gavo*deltho)	137
138		if(abs((havn-havn1)/havn1).le.1.0e-7) go to 90	138
139		n=n+1	139
140		if(n.gt.100)go to 100	140
141		do 112 j=1,nosn	141
142		hn1(j)=thilt+beta(j)*(havn1-thilt)	142
143		call densi(p,hn1(j),dn1(j),xn(j))	143
144	112	continue	144
145		havn=havn1	145
146		do 120 j=1,nosn	146
147		dn(j)=dn1(j)	147
148		hn(j)=hn1(j)	148
149	120	continue	149
150		go to 180	150
151	90	tgilt=gavo+cst1/(cst3*gamma(nosn)*chan1)*(qipt*chan1/area	151
152	1	-gavo*deltho)	152
153		gn(1)=tgilt	153
154		do 190 j=2,nosn	154
155		gn(j)=gn(1)+gamma(j)*(gavo-gn(1))	155
156	190	continue	156
157		if(io.eq.1)go to 150	157
158	c		158
159	c	flow reduce transient	159
160	c		160
161	c	for this transient the calculated pressure drop	161
162	c	is corresponding to ttime-2dt hence there is no meaning	162
163	c	to calculate it for first time step	163
164	c		164
165		if(1.eq.1)go to 151	165
166		time2=time1-dt	166
167		tpolt=polt(polt0,time2)	167
168		do 191 i1=1,nosn	168
169		if(go(i1).lt.0.0)go to 73	169
170	191	continue	170
171		call intrgl(p,go,ho,do,xo,fric)	171
172		predi=(gavo-gavoo)*chan1/dt+fric	172
173		tpilt=tpolt+predi	173
174		p=(tpilt+tpolt)/2.	174
175	151	gavoo=gavo	175
176		go to 160	176
177	150	do 192 i1=1,nosn	177
178		if(gn(i1).lt.0.0)go to 73	178
179	192	continue	179

180	call intrgl(p,gn,hn,dn,xn,fric)	180
181	tpolt=poltt(poltt0,time1)	181
182	tpilt=piltt(piltt0,polt0,time1)	182
183	predi=tpilt-tpolt	183
184	p=(tpolt+tpilt)/2.	184
185	gavn=gavo+dt/chanl*(predi-fric)	185
186	160 gavo=gavn	186
187	deltho=hn(nosn)-thilt	187
188	do 170 j=1,nosn	188
189	do(j)=dn(j)	189
190	ho(j)=hn(j)	190
191	xo(j)=xn(j)	191
192	170 continue	192
193	havo=havn	193
194	if(i.gt.1000)npp=10	194
195	if(np.ne.npp)go to 71	195
196	np=0	196
197	c write(0,9000)i	197
198	9000 format(1x," timme steps ",i5)	198
199	write(6,3600)ttime	199
200	3600 format(//," transient time = ",e13.6," sec")	200
201	havw=havn/1000	201
202	qiptw=qipt/1000	202
203	predw=predi/1000	203
204	write(6,3100)gavn,havw,predw,qiptw	204
205	3100 format(/," average mass flux = ",f8.3," kg/m**2.sec",	205
206	1 /, " average enthalpy = ",f8.3," kj/kg",	206
207	1 /, " pressure drop = ",f8.3," kpa",	207
208	1 /, " linear power = ",f8.3," kw/m")	208
209	write(6,3200)	209
210	3200 format(/," mass flux enthalpy density quality ",	210
211	1 /," kg/m**2.sec kj/kg kg/m**3 ")	211
212	do 200 j=1,nosn	212
213	gr=gn(j)/gilt0	213
214	hw=hn(j)/1000	214
215	write(6,3300)j,gn(j),hw,dn(j),xn(j),gr	215
216	3300 format(1x,i3,2x,f8.3,3x,f8.3,3x,f8.4,2x,f7.4,3x,f7.4)	216
217	200 continue	217
218	71 continue	218
219	70 continue	219
220	go to 72	220
221	73 write(0,3500)i,i1	221
222	write(6,3500)i,i1	222
223	3500 format(1x," flow rate is negative at time step ",i5,/,	223
224	1 " for node ",i4)	224
225	go to 72	225
226	100 write(6,3400)i	226
227	3400 format(1x," iteration not converenge for time step",2x,i2)	227
228	72 continue	228
229	end	229


```

230      subroutine intrgl(p,g,h,d,x,fric)                                1
231      dimension h(21),g(21),d(21),x(21)                              2
232      common /data1/  chan1,area,eqd,nosn,nots,dz,dt                  3
233      fric=0                                                            4
234      data conv1,conv2,vics,g1/737.4643,1.4504e-4,91.7e-6,9.8/      5
235      data vics1/19.73e-6/                                             6
236 c                                                                 7
237 c  acceleration pressure drop                                         8
238 c                                                                 9
239      fric=g(nosn)**2/d(nosn)-g(1)**2/d(1)                            10
240 c                                                                 11
241 c  thm:  two phase pressure drop multiplier                          12
242 c                                                                 13
243      thm=1                                                             14
244 c                                                                 15
245 c  get the saturation liquid and vapor density rho1,rov            16
246 c                                                                 17
247      call satt(p,hls,hvs)                                            18
248      call lden(p,hls,rho1)                                           19
249      call vden(p,hvs,rov)                                            20
250      do 10 i=2,nosn                                                  21
251      ha=(h(i-1)+h(i))/2.                                             22
252      call densi(p,ha,da,xa)                                           23
253      ga=(g(i-1)+g(i))/2.                                             24
254 c                                                                 25
255 c  elavation pressure drop                                           26
256 c                                                                 27
257      fric=fric+da*g1*dz                                              28
258 c                                                                 29
259 c  if x>0 calculate the two phase multiplier                          30
260 c  eq(5.209) p.230 Lahey and Moody was used                         31
261 c                                                                 32
262      if(xa.eq.-1) goto 20                                             33
263      if(xa.eq.1.) go to 30                                            34
264      da=rho1                                                          35
265      gtest=ga*conv1/1.0e6                                             36
266      if(gtest.le.0.7)ci=1.36+0.0005*p*conv2+0.1*gtest              37
267      1  -0.000714*p*conv2*gtest                                       38
268      if(gtest.ge.0.7)ci=1.26-0.0004*p*conv2+0.119/gtest           39
269      1  +0.00028*p/gtest*conv2                                        40
270      thm=ci*(1.2*(rho1/rov-1)*xa**0.824)+1.0                       41
271 c                                                                 42
272 c  use Blasius for friction factor                                     43
273 c                                                                 44
274      20 fact=0.184*(ga*eqd/vics)**(-0.20)                             45
275      go to 40                                                         46
276      30 fact=0.184*(ga*eqd/vics1)**(-0.2)                             47
277      40 fric=fric+thm*fact*ga**2/(2*eqd*da)*dz                       48
278      10 continue                                                    49
279      return                                                            50
280      end                                                                51
281 c                                                                 52
282 c *****                                                            53
283 c                                                                 54

```

284	subroutine init(is1,gilt0,hilt0,pilt0,polt0,powe0,	1
285	1 p1,h,d,g,x)	2
286	dimension h(21),d(21),p(21),g(21),x(21)	3
287	common /data1/ chan1,area,equd,nosn,nots,dz,dt	4
288	call densi(pilt0,hilt0,d(1),x(1))	5
289	h(1)=hilt0	6
290	p(1)=pilt0	7
291	g(1)=gilt0	8
292	i=1	9
293	n1=1	10
294	90 do 10 i=2,nosn	11
295	h1=h(i-1)+powe0*dz/(area*gilt0)	12
296	p1=p(i-1)	13
297	call densi(p1,h1,d(i),x(i))	14
298	60 ha=(h1+h(i-1))/2.	15
299	call densi(p1,ha,da,x1)	16
300	call fric1(x1,p1,gilt0,da,fod)	17
301	p2=p(i-1)-fod*gilt0**2.*dz/(2*equd)	18
302	1 -da*9.80*dz-gilt0**2.*(1/d(i)-1/d(i-1))	19
303	call densi(p2,h1,d(i),x(i))	20
304	call densi(p2,ha,da,x1)	21
305	call fric1(x1,p2,gilt0,da,fod)	22
306	31 h1=h(i-1)+powe0*dz/(area*gilt0)+(p2-p(i-1)+	23
307	1 (fod*gilt0**2*dz)/(2*equd))/da	24
308	if(abs((p2-p1)/p2).le.1.0e-8)go to 40	25
309	n=n+1	26
310	if(n.gt.50)goto 50	27
311	p1=p2	28
312	goto 60	29
313	40 p(i)=p2	30
314	h(i)=h1	31
315	g(i)=gilt0	32
316	10 continue	33
317	if(is1.eq.1)polt0=p(nosn)	34
318	if(abs((p(nosn)-polt0)/polt0).le.1.0e-4)go to 160	35
319	if(n1.gt.50) go to 80	36
320	n1=n1+1	37
321	gilt0=gilt0*((p(nosn)-p(1))/(polt0-pilt0))**0.5	38
322	goto 90	39
323	80 write(6,2110)	40
324	2110 format(1x," iteration not converge for pressure b.c.")	41
325	stop	42
326	50 write(6,2090) i	43
327	2090 format(1x,"iteration not converge for spatial node ",i2)	44
328	stop	45
329	160 p1=(polt0+pilt0)/2.	46
330	return	47
331	end	48

332	subroutine fric1(x1,p,g,d,fod)	1
333	common /data1/ chan1,area,equd,nosn,nots,dz,dt	2
334	data conv1,conv2,vics/737.4643,1.4504e-4,91.7e-6/	3
335		4
336	c use McAdames correlation to calculate the friction factor	5
337		6
338	fric=0.184*(g*equd/vics)**(-0.2)	7
339	call satt(p,hls,hvs)	8
340	call lden(p,hls,rol)	9
341	call vden(p,hvs,rov)	10
342	if(x1.eq.-1.0)go to 10	11
343	gtest=g*conv1/1.0e6	12
344	if(gtest.le.0.7)c1=1.36+0.0005*p*conv2+0.1*gtest	13
345	1 -0.000714*p*conv2*gtest	14
346	if(gtest.gt.0.7)c1=1.26-0.0004*p*conv2+0.119/gtest	15
347	1+0.00028*p/gtest*conv2	16
348	thm=c1*(1.2*(rol/rov-1)*x1**0.824)+1.0	17
349	fod=thm*fric/rol	18
350	go to 21	19
351	10 fod=fric/d	20
352	21 continue	21
353	return	22
354	end	23

355	subroutine cst(havo,ho,do,havn1,hn1,dn1,deltho,gamma,	1
356	1 hilt0,cst1,cst2,cst3)	2
357	dimension ho(21),hn1(21),do(21),dn1(21),b(21),gamma(21)	3
358	common/data1/ chan1,area,equd,nosn,nots,dz,dt	4
359	nosn1=nosn-1	5
360	smtot=do(1)/2.	6
361	do 30 j=2,nosn1	7
362	smtot=smtot+do(j)	8
363	30 continue	9
364	smtot=smtot+do(nosn)/2.	10
365	smtot=smtot*chan1/nosn1	11
366	etot=0	12
367	do 40 j=2,nosn	13
368	et=1/2.*(do(j-1)+do(j))*((ho(j-1)+ho(j))/2.-hilt0)	14
369	etot=etot+et	15
370	40 continue	16
371	etot=etot*chan1/nosn1	17
372	smtt1=1/2.*dn1(1)	18
373	do 60 j=2,nosn1	19
374	smtt1=smtt1+dn1(j)	20
375	60 continue	21
376	smtt1=smtt1+1/2.*dn1(nosn)	22
377	smtt1=smtt1*chan1/nosn1	23
378	etot1=0	24
379	do 70 j=2,nosn	25
380	et=1/2.*(dn1(j-1)+dn1(j))*((hn1(j)+hn1(j-1))/2.-hilt0)	26
381	etot1=etot1+et	27
382	70 continue	28
383	etot1=etot1*chan1/nosn1	29
384	cst1=(smtt1-smtot)/(havn1-havo)	30
385	cst2=(etot1-etot)/(havn1-havo)	31
386	b(1)=0	32
387	do 80 j=2,nosn	33
388	b(j)=dn1(j)-do(j)+dn1(j-1)-do(j-1)+b(j-1)	34
389	80 continue	35
390	btot=0	36
391	btot=1/2.*b(1)	37
392	do 90 j=2,nosn1	38
393	btot=btot+b(j)	39
394	90 continue	40
395	btot=btot+1/2.*b(nosn)	41
396	bave=btot/nosn1	42
397	do 100 j=1,nosn	43
398	gamma(j)=b(j)/bave	44
399	100 continue	45
400	cst3=(gamma(nosn)*cst2-cst1*(gamma(nosn)-1)*deltho)/	46
401	1(gamma(nosn)*chan1)	47
402	return	48
403	end	49
404	c	50

405	subroutine densi(p,h,d,x)	1
406	c	2
407	call satt(p,hls,hvs)	3
408	if(h.le.hls) go to 10	4
409	if(h.gt.hvs) go to 20	5
410	call lden(p,hls,rol)	6
411	call vden(p,hvs,rov)	7
412	x=(h-hls)/(hvs-hls)	8
413	svl=1./rol	9
414	svv=1./rov	10
415	sv=svl*(1-x)+svv*x	11
416	d=1./sv	12
417	return	13
418	10 x=-1.	14
419	call lden(p,h,rol)	15
420	d=rol	16
421	return	17
422	20 x=1.	18
423	call vden(p,h,rov)	19
424	d=rov	20
425	return	21
426	end	22

427	subroutine lden(p,h,ro1)	1
428	data f11,f12/999.65,4.9737e-7/	2
429	data f21,f22/-2.5847e-10,6.1767e-19/	3
430	data f31,f32/1.2696e-22,-4.9223e-31/	4
431	c	5
432	data f41,f42/1488.64,1.3389e-6/	6
433	data f51,f52/1.4695e9,8.85736/	7
434	data f61,f62/3.20372e6,1.20483e-2/	8
435	c	9
436	if(h.gt.6.513e5) go to 10	10
437	f1=f11+f12*p	11
438	f2=f21+f22*p	12
439	f3=f31+f32*p	13
440	ro1=f1+h*h*(f2+f3*h*h)	14
441	go to 20	15
442	10 f4=f41+f42*p	16
443	f5=f51+f52*p	17
444	f6=f61+f62*p	18
445	ro1=f4+f5/(h-f6)	19
446	20 return	20
447	end	21

```
448      subroutine vden(p,h,rov)                                1
449 c                                                                    2
450      data g00,g01,g02,g10,g11,g12/-5.1026e-5,1.1208e-10,      3
451      1 -4.4506e5,-1.6893e-10,-3.3980e-17,2.3058e-1/          4
452 c                                                                    5
453      g0p=g00+g01*p+g02/p                                       6
454      g1p=g10+g11*p+g12/p                                       7
455      rov=1./(g0p+g1p*h)                                           8
456      return                                                       9
457      end                                                           10
```

```

458      subroutine satt(p,hls,hvs)                                1
459      data tsc1,tsc2, tsexp /9.0395, 255.2, 0.223/           2
460      data cps1,cps2, cpsexp /9.5875e2, .00132334, -0.8566/  3
461 c      cps2 = -cpsexp * tcrinv                                4
462 c      for hls and hvs                                       5
463      data h10,h11,h12,h13,h14,h15/5.7474718e5,2.0920624e-1,  6
464      1 -2.8051070e-8,2.3809828e-15,-1.0042660e-22,1.6586960e-30/ 7
465      data hv0,hv1,hv2,hv3,hv4/2.7396234e6,3.758844e-2,      8
466      1 -7.1639909e-9,4.2002319e-16,-9.8507521e-24 /      9
467 c
468      tsat = tsc1* p**tsexp                                    10
469      hvs = hv0 + p*(hv1 + p*(hv2 + p*(hv3 + p*hv4)))         11
470      hls = h10 + p*(h11 + p*(h12 + p*(h13 + p*(h14 + p*h15)))) 12
471      pinv = 1./ p                                            13
472      dtstdp = tsat*tsexp*pinv                                14
473      tsat = tsat + tsc2                                       15
474      return                                                    16
475      end                                                        17

```



```
476 function power(powe0,time1)
477 power=powe0
478 return
479 end
```

```
1
2
3
4
```

```
480 function polt(polt0,time1)
481 polt=polto
482 return
483 end
```

```
1
2
3
4
```

```
484 function gilt(gilt0,time1)
485 gilt=gilt0
486 return
487 end
488 c
```

```
1
2
3
4
5
```

```
489     function pilt(pilt0,polt0,time1)           1
490     cn=100.*time1                             2
491     if(cn.gt.79.99)go to 10                   3
492     pilt=(pilt0+pol1t0)/2.+(pilt0-pol1t0)/2.*(exp(-100*time1)) 4
493     go to 20                                   5
494     10 pilt=(pilt0+pol1t0)/2.                 6
495     20 continue                               7
496     return                                    8
497     end                                       9
```