A PREDICTION OF THE DROPLET SIZE AND VELOCITY DISTRIBUTION IN A SPRAY FROM FIRST PRINCIPLES

by

Richard William Sellens

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of

Master of Applied Science
in
Mechanical Engineering

Waterloo, Ontario, 1985

Abstract

The maximum entropy formalism is used to predict the droplet size and velocity distribution in a spray resulting from the breakup of a liquid sheet. The physics of the process are described by simple conservation constraints for mass, momentum, surface energy and kinetic energy. The predicted size distribution is compared with four empirical distributions and found very similar to a Rosin - Rammler distribution with the same parameters.

The predicted velocity distribution has a Gaussian cross section at any particular droplet diameter. The variance of the Gaussian decreases with increasing droplet diameter, so that the velocity is nearly single valued for large droplets, but varies widely for smaller droplets.

This size and velocity distribution is propagated downstream using a simple physical model which assumes a uniform velocity air flow field, no evaporation, and treats the droplets as solid spheres. Although this is a substantial simplification, it shows that, in the absence of fluctuations in the air stream around the droplets, droplet velocity distributions collapse towards single values. This process takes place over distances which are short when compared to the scale of common spray systems.

Acknowledgements

This research has been made possible by the Natural Sciences and Engineering Research Council, through an operating grant to Dr. T.A. Brzustowski. I would like to thank Dr. Brzustowski for his supervision of this project. He has helped me discover many useful insights, and his editorial skills have substantially improved this thesis.

I am grateful to my office mates, Alex Brena, John Rapanotti and Beth Weckman, for their help and constructive criticism. My thanks also go to Dr. H.K. Kesavan, Dr. M. Renksizbulut and Dr. D.R. Spink for their reading of this thesis.

I would also like to acknowledge the contribution of my new son, Jamie. His arrival delayed this work by a term, however his gummy smile has kept me cheerful through some of the more tedious and frustrating phases of assembling this thesis. He and my wife Peggy deserve substantial credit.

Table of Contents

List of Tables

List of Figures

Nomenclature

A_{proj}	projected surface area
a	coefficient for Upper Limit distribution
b	size parameter for the Nukiyama - Tanasawa distribution
C	constant of integration or proportionality
C_D	drag coefficient
D_{qp}	general weighted mean diameter
D_{20}	surface mean diameter
D_{30}	mass or volume mean diameter
D_{32}	Sauter mean diameter
erf( )	error function of the argument
F_D	drag force
f	general probability density function, or droplet number probability density function over size and velocity space
f_δ	droplet number probability density function over size space only
f_m	droplet mass probability density function over size space only
f_v	droplet number probability density function over velocity space only
g	some function of location in solution space
$<g>$	the average or expectation value of g
$h(\ )$	arbitrary scaling function

K	proportionality constant
k	Boltzman constant
M	total mass in the system
m	cumulative mass fraction or mass of a droplet
N	total number of droplets in the system
n	cumulative number fraction
p	probability of a particular discrete event
q	constraint functional
R	dimensional source term (subscripted)
Re	Reynolds Number
S	Shannon Entropy, or a dimensionless source term (subscripted)
s	constant exponent for the Upper Limit distribution
t	time, or a dummy variable for the error function
U	air velocity
V	sheet velocity
v	droplet velocity
x	independent dimension of solution space, or downstream spatial dimension in the velocity solution
$\bar{x}$	size parameter for the empirical distributions
y	log scaling function in empirical distribution equations
z	dummy variable

α	Rosin - Rammler distribution exponent
β	Nukiyama - Tanasawa distribution exponent
$\Gamma(\ )$	the gamma function of the argument
γ	Log Probability distribution exponent
δ	droplet diameter
κ	Upper Limit distribution exponent
λ	arbitrary Lagrange multiplier
ρ	density
σ	surface tension
τ	sheet thickness
ψ	solution space (usually $d\psi = dv_* d\delta_*$)

Subscripts

a	air
b	base case for continuous form derivation
ke	kinetic energy
l	liquid
m	maximum, or mass for a PDF or a source term
mv	momentum
new	downstream value, usually in terms of the upstream values
r	relative

s	surface energy
$*$	normalized by either D_{30} in the case of a length or V in the case of a velocity.
0,1	upstream and downstream in the velocity solution
$1,2,...,i,...,m$	association with a particular independent dimension of the solution space
$0,1,2,...,i,...,n$	association with a particular constraint number
$0,1,2,...,j,...$	association with a particular discrete state
∞	free stream, i.e. U_∞

Superscripts

$'$	current guess of a quantity in an iterative solution

Introduction

Atomization is an essential process in the combustion of liquid fuels which has been the subject of a great deal of research. Nevertheless, there remains a large gap in the understanding of its essential features. This thesis is a step in the direction of bridging that gap.

The research literature on atomization can be divided into two parts. The first deals with the stability of liquid sheets and ligaments subject to some specific forms of disturbances. This work begins with Rayleigh's famous 1878 paper on the stability of cylindrical jets[1]. It goes on to include the effects of liquid viscosity (Weber, 1931)[2], deformation wavelength and asymmetry (Levich, 1962)[3], aerodynamic effects of relative gas-liquid velocity (Weber, Levich), growth and breakup of surface waves (Mayer, 1961)[4], liquid sheet breakup (Dombrowski and Johns, 1963)[5] and secondary droplet breakup at very high relative gas velocities (Hinze, 1948)[6]. These theoretical studies of various mechanisms involved in the breakup of liquid sheets and ligaments are supported by many experimental studies. However, such work does not yield any description of the spray produced beyond some expression for an average or maximum drop diameter on the basis of the parameters of the disturbances which are most readily amplified. The main features of this research are well summarized by Rice[7].

The remaining research literature on atomization deals with the description of sprays. Papers in this area deal with models of droplet size distributions, with their mathematical descriptions, and with the practical parameters such as the

Sauter mean diameter which can be derived from them. The experimental work involves measuring droplet size distributions produced by practical devices and their models, fitting the data with one of the canonical distributions, and relating the various mean quantities of these distributions to the design and operating parameters of the atomizers used. Much of the modern work of this kind has been reported in the proceedings of the International Conferences on Liquid Atomization and Spray Systems (ICLASS).

This work was undertaken to find out if the actual form of the droplet size and velocity distribution in a spray could be predicted from first principles, given only the parameters of a liquid sheet whose surface area has already been increased to the maximum. The first principles are the conservation of mass, momentum and energy.

What is sought is the solution to the following problem: given a certain mass of liquid having a specified volume and momentum, as well as surface and kinetic energy, what is the most likely distribution of droplet diameters when this liquid is atomized? This problem statement is ideally suited to solution using the maximum entropy formalism pioneered by Edwin Jaynes and Myron Tribus which is described fully in Chapter One. This formalism is a tool of statistical inference which is of growing importance in science and engineering. It is often identified with statistical thermodynamics, but that identification is too limiting. This work is not an application of thermodynamic principles to atomization in the classical sense. It is an application of statistical inference in which some of the constraints are expressed in terms of thermodynamic variables, e.g. surface energy.

The approach taken here is similar in a fundamental way to that employed by Kelly[8] in his studies of electrostatic sprays, but differs from his work both in purpose and in detail.

Chapter One

THE MAXIMUM ENTROPY FORMALISM AND ITS SOLUTION

1.1. The Shannon Entropy

In 1948 Claude Shannon[9] proposed a definition of entropy as it relates to communication theory as:

$$S = -k \sum_j p_j \ln p_j \tag{1.1.1}$$

This entropy is a measure of the information present in a message. Consider a communication line whose state (perhaps voltage) is varying with time so as to transmit information. If the state remains constant then the minimum amount of information is transmitted and the entropy from equation (1.1.1) is a maximum. Any deviation from a uniform state provides some information, either message or noise, and correspondingly reduces the entropy. The more unlikely the event (smaller probability), the greater the amount of information carried by such an event.

If the entire signal is of interest, then both the noise and message components of the signal provide information. The former provides information about the noise sources in the system, while the latter provides information about the message sent. However, the noise content is usually not of interest and acts only to mask the message. The base, or maximum entropy, state must then be taken as a constant null message state with a statistical representation of the noise superimposed. In this case any message signals must be larger, or contain greater redundancy, in order to be discernible from the noise. Only the discernible portions of the message will provide any additional information, and thus a reduction in entropy.

Shannon applied this idea to the design of communications hardware, calculating such things as the channel capacity of a communications link in the presence of different noise conditions. The effect of his work has, however, been felt over a far wider range of subjects.

1.2. Edwin Jaynes and Myron Tribus

The work of Edwin Jaynes[10] starting in 1957 builds on Shannon's information theory and shows that the methods of statistical mechanics are in fact a special case of statistical inference based on the entropy measure of information. Jaynes proposed a formalism for the maximization of entropy subject to constraints which can be applied in any type of problem with incomplete information. Myron Tribus has done substantial work in applying this formalism and making it available to the engineering profession as it relates to thermodynamics[11] as well as problems in design and resource management[12]. The following section presents the maximum entropy formalism in much the same form as it is presented by Tribus in references 11 and 12.

1.3. The Maximum Entropy Formalism

The maximum entropy formalism is useful when the information available describing the system under consideration is macroscopic in nature, describing some average or moment of the system. Examples of this type of information include:

- the temperature of a system of molecules as a statement of the average kinetic energy of molecules in that system.
- the mean and variance of a part's diameter over a production run as information on a set of quality control measurements.
- any other measure which is the average of some property over all the elements in the system.

Any information of this type can be written as a constraint on the probability distribution of the form

$$\sum_j p_j g_j = \langle g \rangle \qquad (1.3.1)$$

where p_j is the probability of some state j, g_j is some function of state evaluated at state j and $\langle g \rangle$ is the (known) expectation or average value of the function g over the entire system. The summation is over some range of j so that all possible states of the system are evaluated.

If a set of such constraints is combined with a normalization constraint requiring the sum of the probabilities to be unity, this system of equations results:

$$\sum_j p_j = 1 \tag{1.3.2}$$

$$\sum_j p_j g_{i,j} = \langle g_i \rangle \qquad i = 1,2,3,\ldots,n \tag{1.3.3}$$

In cases of interest there are generally many more possible states than there are constraints available, making this system of equations indeterminate. A method is needed to choose the most appropriate solution which satisfies the constraints. The Shannon Entropy provides the criterion.

If a solution is found to fit the constraints, then all the information that is present in the constraints must be present in the resulting solution, for each of the constraints may be obtained by taking the appropriate moment of the distribution. By maximizing the entropy, subject to the constraints, one obtains the distribution of probabilities containing the least amount of information, while still containing all the information present in the constraints. This solution will be the most appropriate because it conveys all the available information without adding any unjustified bias in the form of additional information.

The solution of the system of equations which produces the maximum entropy can be found using the method of Lagrange multipliers. Differentiation of equation (1.1.1) with respect to the p_j yields

$$dS = -k \sum_j (\ln p_j + 1) dp_j \tag{1.3.4}$$

and setting $dS = 0$ for a maximum gives

$$\sum_j (\ln p_j + 1) dp_j = 0 \tag{1.3.5}$$

Similar differentiation of equations (1.3.2) and (1.3.3), noting that the $<g_i>$ are constant, yields

$$\sum_j dp_j = 0 \tag{1.3.6}$$

$$\sum_j g_{i,j} dp_j = 0 \qquad i = 1,2,3, \ldots ,n \tag{1.3.7}$$

To obtain the solution the above $n+1$ equations are multiplied by some set of arbitrary multipliers. For simplicity in solution the first equation is multiplied by $(\lambda_0 - 1)$ and the rest by the set λ_i ; $i = 1,2,3, \ldots ,n$. If these multiplied equations are then summed in combination with the previous equation the result is

$$\sum_j (\ln p_j + 1) dp_j + \sum_j (\lambda_0 - 1) dp_j + \sum_{i=1}^{n} \sum_j \lambda_i g_{i,j} dp_j = 0 \tag{1.3.8}$$

or, combining terms,

$$\sum_j \left[\ln p_j + \lambda_0 + \sum_{i=1}^{n} \lambda_i g_{i,j} \right] dp_j = 0 \tag{1.3.9}$$

To ensure that this equation is satisfied for arbitrary dp_j, the quantity within the square brackets is set to zero.

$$\ln p_j + \lambda_0 + \sum_{i=1}^{n} \lambda_i g_{i,j} = 0 \tag{1.3.10}$$

so that, after dropping the state index j

$$p = \exp(-\lambda_0 - \lambda_1 g_1 - \lambda_2 g_2 - \cdots - \lambda_n g_n) \tag{1.3.11}$$

where values of the probability p and the functions g_i vary with the state point being evaluated and the λ_i are constant. Equation (1.3.11) is the solution to the problem. It is the probability distribution which maximizes the entropy under the given constraints. However, the Lagrange multipliers are still undetermined. They must be chosen in such a way as to satisfy the constraint equations.

Substituting equation (1.3.11) into the normalization constraint equation (1.3.2) gives

$$\sum p = \sum \exp(-\lambda_0 - \lambda_1 g_1 - \lambda_2 g_2 - \cdots - \lambda_n g_n) = 1 \tag{1.3.12}$$

or with rearrangement

$$\lambda_0 = \ln\left(\sum \exp(-\lambda_1 g_1 - \lambda_2 g_2 - \cdots - \lambda_n g_n)\right) \tag{1.3.13}$$

which permits the direct evaluation of λ_0, given the values of the other multipliers. This is simply a restatement of the normalization constraint and λ_0 is a normalization parameter which alters only the magnitude of the probabilities, not their relative distribution.

Since equation (1.3.11) can be used to replace p_j in all the constraint equations the system is now a set of $n+1$ equations in $n+1$ unknowns. The equations are the n physical constraints derived from the problem and set in the form of equation (1.3.3), plus the normalization constraint expressed as either equation (1.3.2) or equation (1.3.13). The $n+1$ variables are the Lagrange multipliers $\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_n$.

These summation equations are not easily solved, except in trivial cases. The problem can be simplified by translating these discrete summation equations and their solution to some equivalent continuous form where the familiar methods of calculus can be applied.

1.3.1. "Extension" of the Maximum Entropy Formalism to a Multi-Dimensional Solution Space

It is easy to see how the previously derived formalism can be applied to a one dimensional problem. One simply maps the state index j directly onto the one dimensional variable in question. Many problems in thermodynamics involve quantum states and are easily viewed as an ensemble of possible discrete states. Even if the problem concerns a variable that is continuous in one dimension, it can be readily discretized to match the formalism.

It is important to note that in the derivation of the formalism there is no assumption of the "shape" of the solution space. It is said only that the summations are over the set of possible state points and that each g_i is some function of which state point j is being considered. The j is an index only, and need not have any correlation with the physical parameters of the solution space. The index j can map into a one, two, or even n dimensional solution space with equal validity. The index need not map into the solution space completely, or even in a contiguous manner. There may be states which are in general possible, but which are excluded from a specific solution because of the nature of the problem. These excluded states may be those beyond some maximum or minimum bound, or they may be "forbidden" states which lie within the general domain.

This can be verified by deriving the formalism with a different number of index parameters. If the number of index parameters was significant one would not obtain the same results using two parameters as one. However, the only difference in results is that which was introduced explicitly: the requirement for two index subscripts instead of one.

1.4. An Equivalent Continuous Formalism

Many physical problems do not naturally lend themselves to solution in a space consisting of a finite number of discrete states. On a macroscopic level the physical variables are usually continuous rather than discrete. This can be handled in two ways; either the continuous physical space can be discretized to give a solution space which matches the formalism, or an equivalent formalism can be developed for the continuous solution space. The latter is preferable both because it provides a more natural representation of the physical problem and because it leads to the more familiar continuous mathematics and integrals rather than discrete mathematics and summations.

Consider a solution space having m dimensions represented by the independent variables $x_1, x_2, x_3, \ldots, x_m$. The solution space is discretized into elements which are uniform with respect to some set of monotonically increasing or decreasing functions $h_1(x_1)$, $h_2(x_2)$, $h_3(x_3)$, $\ldots, h_m(x_m)$ so that for any mesh element

$$h_i(x_i + \Delta x_i) - h_i(x_i) = \text{a constant} \tag{1.4.1}$$

If Δx_i is small, one can make the approximation

$$\frac{\partial h_i}{\partial x_i} \approx \frac{h_i(x_i + \Delta x_i) - h_i(x_i)}{\Delta x_i} \tag{1.4.2}$$

which, in combination with equation (1.4.1), gives the relation

$$\Delta x_i \propto \left(\frac{\partial h_i}{\partial x_i}\right)^{-1} \tag{1.4.3}$$

Now define $\Delta\psi$ as a small element of the solution space ψ so that

$$\Delta\psi = \Delta x_1 \Delta x_2 \Delta x_3 \cdots \Delta x_m \tag{1.4.4}$$

and define some base set of discretization volumes $\{\Delta\psi_b\}$ to work from so that

$$\{\Delta\psi_b\} = \frac{C_b}{\left\{\frac{\partial h_1}{\partial x_1}\right\}\left\{\frac{\partial h_2}{\partial x_2}\right\}\left\{\frac{\partial h_3}{\partial x_3}\right\}\cdots\left\{\frac{\partial h_m}{\partial x_m}\right\}} \tag{1.4.5}$$

where C_b is a proportionality constant. A proportionality constant K is used to define a corresponding set of smaller volumes with the same discretization shape as $\{\Delta\psi_b\}$.

$$\{\Delta\psi\} = \frac{\{\Delta\psi_b\}}{K} \tag{1.4.6}$$

From application of the maximum entropy formalism over $\{\Delta\psi_b\}$

$$p_b = \exp(-\lambda_{0b} - \lambda_{1b} g_1 - \lambda_{2b} g_2 - \cdots - \lambda_{nb} g_n) \tag{1.4.7}$$

is the probability associated with a particular $\Delta\psi_b$ where $g_1, g_2, \ldots, g_n$ are functions of location in ψ. If the probability density is continuous and $\{\Delta\psi_b\}$ is fine enough to resolve it, then

$$\{p\} \approx \frac{\{p_b\}}{K} \tag{1.4.8}$$

is a good approximation for the set of probabilities associated with the smaller volumes $\{\Delta\psi\}$. Substituting in equation (1.4.7) yields the following expression for the probability.

$$p = \exp(-(\lambda_{0b} + \ln K) - \lambda_{1b} g_1 - \lambda_{2b} g_2 - \cdots - \lambda_{nb} g_n) \tag{1.4.9}$$

Define a set of probability densities and rewrite the above equation for f, a probability density function or PDF, rather than p, a probability.

$$\{f\} = \frac{\{p\}}{\{\Delta\psi\}} \tag{1.4.10}$$

$$f = \exp(-(\lambda_{0b} + \ln K + \ln\Delta\psi) - \lambda_{1b} g_1 - \lambda_{2b} g_2 - \cdots - \lambda_{nb} g_n) \tag{1.4.11}$$

Recalling that $K = \dfrac{\Delta\psi_b}{\Delta\psi}$, or $\ln K = \ln\Delta\psi_b - \ln\Delta\psi$, the above expression can be reduced to

$$f = \frac{1}{\Delta\psi_b} \exp(-\lambda_{0b} - \lambda_{1b} g_1 - \lambda_{2b} g_2 - \cdots - \lambda_{nb} g_n) \tag{1.4.12}$$

Substituting in from equation (1.4.5) and redefining the Lagrange multipliers as $\lambda_0 = \lambda_{0b} + \ln C_b$; $\lambda_1 = \lambda_{1b}$; $\lambda_2 = \lambda_{2b}$ $\cdots$ $\lambda_n = \lambda_{nb}$, gives the final expression for the equivalent continuous probability density function.

$$f = \frac{\partial h_1}{\partial x_1}\frac{\partial h_2}{\partial x_2}\frac{\partial h_3}{\partial x_3}\cdots\frac{\partial h_m}{\partial x_m}\exp(-\lambda_0 - \lambda_1 g_1 - \lambda_2 g_2 - \cdots - \lambda_n g_n) \qquad (1.4.13)$$

If the original solution space was discretized uniformly over each of the independent variables then each of the $\frac{\partial h_i}{\partial x_i}$ will be constant over the solution space. If these constants are absorbed into λ_0 in the same way that C_b was, the result is an expression for the PDF over a continuous field, equivalent to a discrete formulation with uniform discretization over each of the independent variables.

$$f = \exp(-\lambda_0 - \lambda_1 g_1 - \lambda_2 g_2 - \cdots - \lambda_n g_n) \qquad (1.4.14)$$

Examination shows that equation (1.4.14) gives the PDF f in the same form that equation (1.3.11) gives the discrete probability p_j. The only difference is in the value of λ_0, and this difference reflects a constant of proportionality between p_j and f which will depend on the magnitude of the grid used in the discrete solution.

1.4.1. The Meaning of a Non-Uniform Discretization

In the previous section a continuous formalism was developed based on some potentially non-uniform discretization of the solution space. What does it mean to use a non-uniform discretization with the maximum entropy formalism?

Discretization is the first task required in applying the formalism. It must be done on some rational basis. What is the most reasonable way to divide a continuous field? In the absence of information to the contrary, any field should be divided into uniform elements based on the unit of measure being applied. To do anything else would imply further information that allows one part of the field to be differentiated from another on some basis other than the applied unit of measure. If one has such information it would be preferable to apply it as a constraint, through the usual means of the formalism.

In some cases, however, the field might best be described physically by some other unit of measure, but for practical reasons it is desirable to represent it in the chosen units. (i.e. it may be appropriate to consider specific kinetic energy as the best physical description, rather than velocity, but velocity is the desired scale for reasons of presentation.) In this one may apply a discretization which is non-uniform in the scale units to provide a discretization which is uniform in the units which are felt to be most physically representative.

In any event, the use of a non-uniform discretization implies prior information of some sort which is not necessarily present in the constraints. In the absence of all physical constraints each state is equally probable and a non-uniformity of discretization puts more divisions, or states, in one area of the field than another. This results in a non-uniform probability density which has some information content beyond the minimum for the definition of the field.

1.5. Solution of the Equations in the Continuous Form

The transformation to a continuous solution space yields a system of integral equations which take the form

$$\int_{\psi} f g_0 \, d\psi = \langle g_0 \rangle \qquad (1.5.1\ (a))$$

$$\int_{\psi} f g_1 \, d\psi = \langle g_1 \rangle \qquad (1.5.1\ (b))$$

$$\int_{\psi} f g_2 \, d\psi = \langle g_2 \rangle \qquad (1.5.1\ (c))$$

$$\cdots$$

$$\int_{\psi} f g_n \, d\psi = \langle g_n \rangle \qquad (1.5.1\ (d))$$

$$f = \exp(-\lambda_0 - \lambda_1 g_1 - \lambda_2 g_2 - \cdots - \lambda_n g_n) \qquad (1.5.2)$$

$$\lambda_0 = \ln\left[\int_{\psi} \exp(-\lambda_1 g_1 - \lambda_2 g_2 - \cdots - \lambda_n g_n)\, d\psi\right] \qquad (1.5.3)$$

where g_0 and $\langle g_0 \rangle$ are defined as equal to one to put the normalization constraint equation in the same form as the other constraint equations.

Agmon, Alhassid and Levine[13] give a method for solution of a discrete maximum entropy formalism problem. They dismiss the use of an iterative method of the Newton - Rhapson type as generally non-convergent, even in a system with only a single constraint. In this section a slightly modified solution method of the Newton - Rhapson type is developed for the continuous problem. The new method converges well, even in a system of four constraints.

The constraints can be rewritten as functionals $q_0, q_1, q_2, \ldots, q_n$ with values of zero.

$$q_0 = \int_{\psi} f g_0 \, d\psi \; - <g_0> = 0 \qquad (1.5.4\,(a))$$

$$q_1 = \int_{\psi} f g_1 \, d\psi \; - <g_1> = 0 \qquad (1.5.4\,(b))$$

$$q_2 = \int_{\psi} f g_2 \, d\psi \; - <g_2> = 0 \qquad (1.5.4\,(c))$$

$$\ldots$$

$$q_n = \int_{\psi} f g_n \, d\psi \; - <g_n> = 0 \qquad (1.5.4\,(d))$$

Consider q_i as a general constraint functional and a quantity $q'_i = q_i(\lambda'_0, \lambda'_1, \lambda'_2, \ldots, \lambda'_n)$ where the prime indicates values which are used as a guess in the current iteration. Taking a first order Taylor series expansion about q'_i gives

$$q_i = q'_i + (\lambda_0 - \lambda'_0)\frac{\partial q_i}{\partial \lambda_0} + (\lambda_1 - \lambda'_1)\frac{\partial q_i}{\partial \lambda_1} +$$

$$(\lambda_2 - \lambda'_2)\frac{\partial q_i}{\partial \lambda_2} + \cdots + (\lambda_n - \lambda'_n)\frac{\partial q_i}{\partial \lambda_n} \qquad (1.5.5)$$

and setting q_i equal to zero to satisfy the i^{th} constraint yields:

$$\frac{\partial q_i}{\partial \lambda_0}\lambda_0 + \frac{\partial q_i}{\partial \lambda_1}\lambda_1 + \frac{\partial q_i}{\partial \lambda_2}\lambda_2 + \cdots + \frac{\partial q_i}{\partial \lambda_n}\lambda_n =$$

$$-q'_i + \frac{\partial q_i}{\partial \lambda_0}\lambda'_0 + \frac{\partial q_i}{\partial \lambda_1}\lambda'_1 + \frac{\partial q_i}{\partial \lambda_2}\lambda'_2 + \cdots + \frac{\partial q_i}{\partial \lambda_n}\lambda'_n \qquad (1.5.6)$$

Applying this approach to each of the constraints in turn gives a system of $n+1$ linear equations in $n+1$ unknowns. This system is shown in Figure 1.1. As the solution space, the g_i values, and the expectation values are independent of the Lagrange multipliers it is a simple matter to evaluate the derivatives.

$$q_i = \int_\psi f g_i \, d\psi - \langle g_i \rangle \qquad (1.5.7)$$

$$\frac{\partial q_i}{\partial \lambda_j} = \int_\psi \frac{\partial f}{\partial \lambda_j} g_i \, d\psi = -\int_\psi f g_j g_i \, d\psi \qquad (1.5.8)$$

The derivatives $\frac{\partial q_i}{\partial \lambda_j}$ can be evaluated at q'_i. It would be convenient if this system of equations converged nicely over a few iterations to a solution. However, as observed by Agmon et al, this system is unstable and generally does not converge without intervention.

$$\left[\begin{array}{ccccc|c}
\frac{\partial q_0}{\partial \lambda_0} & \frac{\partial q_0}{\partial \lambda_1} & \frac{\partial q_0}{\partial \lambda_2} & \cdots & \frac{\partial q_0}{\partial \lambda_n} & \lambda_0 \\
\frac{\partial q_1}{\partial \lambda_0} & \frac{\partial q_1}{\partial \lambda_1} & \frac{\partial q_1}{\partial \lambda_2} & \cdots & \frac{\partial q_1}{\partial \lambda_n} & \lambda_1 \\
\frac{\partial q_2}{\partial \lambda_0} & \frac{\partial q_2}{\partial \lambda_1} & \frac{\partial q_2}{\partial \lambda_2} & \cdots & \frac{\partial q_2}{\partial \lambda_n} & \lambda_2 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\frac{\partial q_n}{\partial \lambda_0} & \frac{\partial q_n}{\partial \lambda_1} & \frac{\partial q_n}{\partial \lambda_2} & \cdots & \frac{\partial q_n}{\partial \lambda_n} & \lambda_n
\end{array}\right]$$

$$= \left[\begin{array}{l}
-q'_0 + \frac{\partial q_0}{\partial \lambda_0}\lambda'_0 + \frac{\partial q_0}{\partial \lambda_1}\lambda'_1 + \frac{\partial q_0}{\partial \lambda_2}\lambda'_2 + \cdots + \frac{\partial q_0}{\partial \lambda_n}\lambda'_n \\
-q'_1 + \frac{\partial q_1}{\partial \lambda_0}\lambda'_0 + \frac{\partial q_1}{\partial \lambda_1}\lambda'_1 + \frac{\partial q_1}{\partial \lambda_2}\lambda'_2 + \cdots + \frac{\partial q_1}{\partial \lambda_n}\lambda'_n \\
-q'_2 + \frac{\partial q_2}{\partial \lambda_0}\lambda'_0 + \frac{\partial q_2}{\partial \lambda_1}\lambda'_1 + \frac{\partial q_2}{\partial \lambda_2}\lambda'_2 + \cdots + \frac{\partial q_2}{\partial \lambda_n}\lambda'_n \\
\cdots \\
-q'_n + \frac{\partial q_n}{\partial \lambda_0}\lambda'_0 + \frac{\partial q_n}{\partial \lambda_1}\lambda'_1 + \frac{\partial q_n}{\partial \lambda_2}\lambda'_2 + \cdots + \frac{\partial q_n}{\partial \lambda_n}\lambda'_n
\end{array}\right]$$

Figure 1.1. Matrix of linear equations used to solve for the Lagrange multipliers.

In fact, this system can be stabilized by using equation (1.5.3) after each iteration to ensure that normalization is maintained. On reflection it is clear that the definition of the PDF is fundamental, and that if normalization is not maintained the system will fall apart. The Newton - Rhapson approach by itself does not ensure normalization at all stages of solution.

1.5.1. Solution of Doubly Truncated Gaussian Distributions

In cases where the mean and variance are known for a distribution of probability density over a finite range, entropy maximization yields a doubly truncated Gaussian distribution. This distribution has the same mathematical form as an unbounded Gaussian,

$$f = \exp(-\lambda_0 - \lambda_1 x - \lambda_2 x^2) \tag{1.5.9}$$

but the Lagrange multipliers are such as are necessary to fit the constraints within the finite range.

If the range is wide in relation to the variance and the mean is well away from the limits of the range, then the truncated Gaussian will have a shape nearly identical to the unbounded Gaussian. As the range is narrowed for the same mean and variance the curve will become "flatter" due to the encroachment of the range limits on the practical limits of the distribution. For a centrally located mean, the distribution will flatten out to a horizontal line as the range is decreased and then will actually become concave to maintain the same variance. Although the doubly truncated Gaussian often has the familiar appearance of the unbounded Gaussian, it can have almost any shape, with the appropriate mean, variance and range.

The doubly truncated Gaussian is important in the downstream velocity solution presented in Chapter Three. This distribution is discussed, and a specialized solution method presented by Tribus in reference 12 (pp. 141-144). This method is applied in Chapter Three.

Chapter Two

THE CASE OF SHEET BREAKUP: CONSTRAINTS AND SOLUTION

2.1. The Model

In many atomization applications the nozzle produces a thin sheet of liquid which subsequently breaks up into droplets. In the swirl jet nozzle, which is commonly used in moderate capacity oil burners, the fuel is given swirl by tangential entry into a central chamber. The fuel exits through an orifice as a cylindrical jet, but the swirl causes the jet to expand radially forming a conical sheet. As this sheet moves out it is thinned by the radial expansion and will generally develop undulations. At some point the instability of the undulations and the stretching of the sheet become critical and the sheet breaks up into ligaments of fluid. These ligaments then break up into droplets, perhaps by the Rayleigh instability or a similar mechanism.

As noted in the introduction, the fluid mechanics of this problem are extremely complex and have not yet provided anything close to a complete solution. This model applies only the constraints that one can feel reasonably sure of and dismisses many of the complexities with the statement "We have no information". If these complexities are important to the results their importance will be reflected in the differences between the predicted and observed distributions.

Consider the system shown in Figure 2.1. A sheet of liquid has been stretched and has reached its breakup length. The location of the breakup, and the amount of stretching that takes place before the sheet reaches that point are functions of the atomizer parameters and the fluid properties, particularly surface tension. This model presumes knowledge of the sheet properties at the instant before breakup.

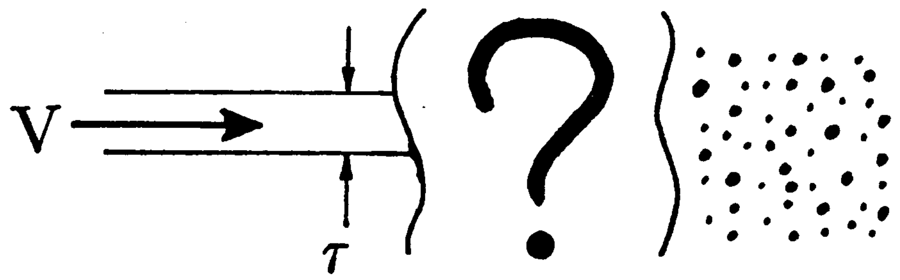


Figure 2.1. The sheet breakup system under consideration.

The sheet has a certain streamwise velocity V, and thickness τ. At the breakup length a process begins, which over a relatively short distance breaks up the sheet into droplets of some characteristic size. It turns out that the mass mean diameter D_{30} is a useful measure of that characteristic size. This model presumes no information about the process of breakup, except that it must obey simple conservation constraints. Mass, momentum and energy must be conserved.

2.1.1. Weighted Mean Diameters

The important features of drop size distributions have traditionally been characterized by weighted mean diameters defined as

$$D_{qp}^{(q-p)} = \frac{\int_{\psi} f\,\delta^{q}\,d\psi}{\int_{\psi} f\,\delta^{p}\,d\psi} \qquad (2.1.1)$$

This allows description of the moments of a distribution using values such as the mass mean diameter, D_{30}, or the surface mean diameter, D_{20}. Weighted mean diameters can also be used to represent the ratio of moments of a distribution, as is done by the Sauter mean diameter, D_{32}, which characterizes the mass to surface ratio.

In this work the mass and surface mean diameters appear in the constraints, and general expressions for D_{qp} are given in the section on empirical distributions.

2.1.2. The Solution Space

Parameters are non-dimensionalized by the mass mean diameter D_{30}, for lengths and the sheet velocity V, for velocities. Non-dimensionalized parameters are denoted by the subscript $*$. The solution space is defined by droplet diameter δ, and velocity v, in their dimensionless forms, so that an element of solution space is $d\psi = d\delta_* dv_*$. The PDF is then defined by the relation

$$p = f\, d\psi = f\, d\delta_* dv_* \tag{2.1.2}$$

2.1.3. Conservation of Mass

To ensure conservation of mass the system of droplets must be constrained to have the same amount of mass as the corresponding portion of the sheet. If a sample of N droplets resulting from the breakup of a section of sheet with a total mass of M is considered, then:

$$M = \int\!\!\int_{\psi} N f \frac{\rho\pi\delta^3}{6}\, d\delta_* dv_* + \text{ mass sources} \tag{2.1.3}$$

Noting that $\dfrac{M}{N} = \dfrac{\rho\pi {D_{30}}^3}{6}$, and that $\delta_* \equiv \delta/D_{30}$, rearrangement yields

$$\int\!\!\int_{\psi} f\, \delta_*^3\, d\delta_* dv_* = 1 + S_m \tag{2.1.4}$$

The mass source term S_m represents any exchange of mass with the environment. The primary mechanism for this exchange would be evaporation.

2.1.4. Conservation of Momentum

Similarly, for momentum to be conserved from the sheet to the spray, the following constraint must apply.

$$\int\int_{\psi} f \delta_*^3 v_* \, d\delta_* dv_* = 1 + S_{mv} \tag{2.1.5}$$

The momentum source term S_{mv} accounts for any momentum transfer between the liquid and its environment through such actions as aerodynamic drag.

2.1.5. Conservation of Energy

Initially, one would expect the conservation of energy to provide a single constraint. However, this would leave out important information concerning the irreversibility of certain energy transformations and the prior knowledge of the energy distribution between various modes before breakup.

Figure 2.2 shows the possible energy modes which make up the total energy of the system and the energy transfer paths between them. Note that there are some paths which allow transfer in only one direction. This represents the complete irreversibility of certain processes. In addition, some sets of transition mechanisms act more strongly in one direction than the other. This is the case for transitions which are subject to significant losses and are therefore only partially reversible. These losses may be energy reductions, or entropy increases due to a loss of order or "directedness", as in the transition from the kinetic energy of a mean flow to undirected turbulent kinetic energy. This difference

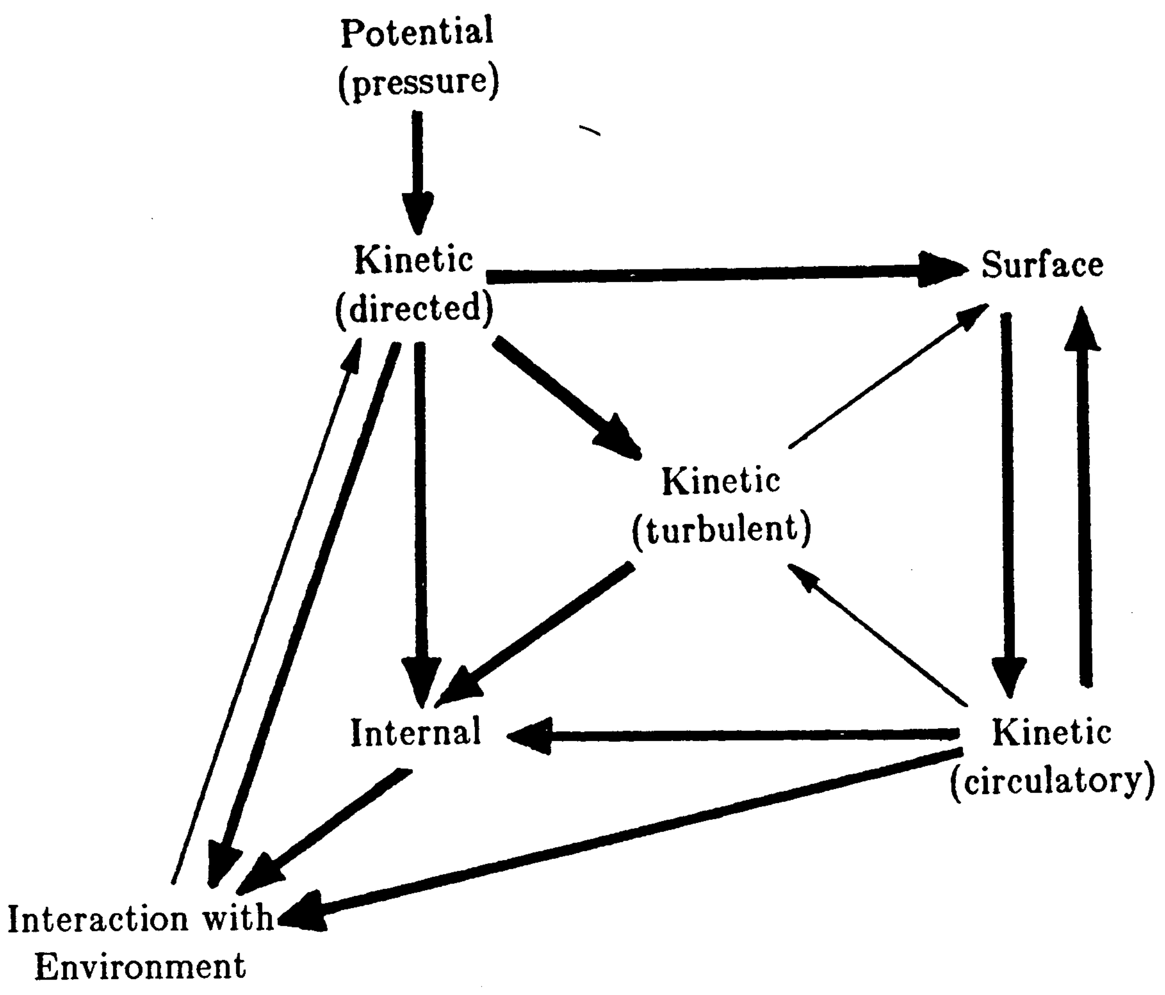


Figure 2.2. Available transfer paths between energy modes.

has been emphasized in the figure by using lines of differing widths for the different paths.

For example, an element of liquid may easily be deformed by the action of aerodynamic drag and new surface area formed as a result of the reduction in velocity. However, the reverse process is not possible. This is because the directed kinetic energy, part of which is transferred to the environment and part of which is transformed to surface energy by the drag deformation, is more ordered than the surface energy it has been transformed into. Even if the fluid element snaps back and shatters, transforming a substantial portion of its surface energy into kinetic energy, that energy will be undirected. The resulting droplets will have velocities in a variety of directions.

2.1.5.1. The Kinetic Energy Constraint

To conserve kinetic energy through the breakup region the value for the droplet distribution must balance with the initial value for the moving sheet plus any sources of kinetic energy in the breakup region. If the sheet is presumed to have a uniform liquid velocity, V, then the kinetic energy of an element of the sheet, having mass M, is simply $\frac{1}{2}MV^2$. The corresponding quantity for the spray may be obtained by by integration. Setting the two equal to each other and including the source term yields

$$\int\int_{\psi} Nf\frac{\rho\pi\delta^3}{6}\frac{v^2}{2}\,d\delta_*dv_* = \frac{1}{2}MV^2 + \text{kinetic energy sources} \qquad (2.1.6)$$

Noting, as before, that $\frac{M}{N} = \frac{\rho\pi D_{30}{}^3}{6}$, and that $v_* \equiv v/V$ and $\delta_* \equiv \delta/D_{30}$, rearrangement yields

$$\int\int_{\psi} f\,\delta_*^3 v_*^2\,d\delta_*dv_* = 1 + S_{ke} \qquad (2.1.7)$$

where S_{ke} is a dimensionless kinetic energy source term, analogous to the momentum source term S_{mv} mentioned earlier.

2.1.5.2. The Surface Energy Constraint

If the surface tension, σ, is taken as a constant, the conservation of surface energy, σA, becomes equivalent to the conservation of surface area. The surface area of an element of mass in the sheet just before breakup is $2M/(\rho\tau)$, where τ is the thickness of the sheet. Balancing this with the surface area of the resulting droplets and a source term yields

$$\int\int_{\psi} Nf\,\pi\delta^2\,d\delta_*dv_* = \frac{2M}{\rho\tau} + \text{surface area sources} \qquad (2.1.8)$$

Rearrangement, as before, gives

$$\int_{\psi}\int f\,\delta_*^2\,d\delta_*\,dv_* = \frac{1}{3\tau_*} + S_s \qquad (2.1.9)$$

where S_s is a dimensionless surface energy source term.

2.1.6. The Source Terms

It is important to note that, although they are all dimensionless, the source terms cannot be directly compared. Consider R as a dimensional source term, on a per droplet basis, equivalent to the dimensionless source term S.

Mass $$R_m = \frac{\rho\pi {D_{30}}^3}{6} S_m \qquad (2.1.10\ (a))$$

Momentum $$R_{mv} = \frac{\rho\pi {D_{30}}^3}{6} V S_{mv} \qquad (2.1.10\ (b))$$

Kinetic Energy $$R_{ke} = \frac{\rho\pi {D_{30}}^3}{6} V^2 S_{ke} \qquad (2.1.10\ (c))$$

Surface Energy $$R_s = \sigma\pi {D_{30}}^2 S_s \qquad (2.1.10\ (d))$$

The mass, momentum and kinetic energy source terms each have very similar dimensional parts, differing only by the order of the velocity element. The surface energy source term, however, has a dimensional portion which is completely different from the kinetic energy source term. This is particularly important, as it means that transfers from kinetic energy to surface energy cannot be applied simply as offsetting source terms of equal magnitude. They must be adjusted to allow for the differing definitions of the dimensionless source terms.

2.2. The Formal Solution

The four constraints developed for conservation of mass, momentum, surface energy and kinetic energy are combined with a normalization constraint as described in Chapter One, giving the following system of equations.

$$\int\int_{\psi} f \, d\delta_* dv_* = 1 \tag{2.2.1 (a)}$$

$$\int\int_{\psi} f \delta_*^2 \, d\delta_* dv_* = \frac{1}{3\tau_*} + S_s \tag{2.2.1 (b)}$$

$$\int\int_{\psi} f \delta_*^3 \, d\delta_* dv_* = 1 + S_m \tag{2.2.1 (c)}$$

$$\int\int_{\psi} f \delta_*^3 v_* \, d\delta_* dv_* = 1 + S_{mv} \tag{2.2.1 (d)}$$

$$\int\int_{\psi} f \delta_*^3 v_*^2 \, d\delta_* dv_* = 1 + S_{ke} \tag{2.2.1 (e)}$$

Applying the formal solution of Chapter One to this system yields an expression for the probability density

$$f = \exp(-\lambda_0 - \lambda_1\delta_*^2 - \lambda_2\delta_*^3 - \lambda_3\delta_*^3 v_* - \lambda_4\delta_*^3 v_*^2) \tag{2.2.2}$$

and for the normalization parameter λ_0 in terms of the other multipliers.

$$\lambda_0 = \ln\left[\int_\psi\int \exp(-\lambda_1\delta_*^2 - \lambda_2\delta_*^3 - \lambda_3\delta_*^3 v_* - \lambda_4\delta_*^3 v_*^2)\, d\delta_* dv_*\right] \tag{2.2.3}$$

2.3. The Numerical Solution

Having found the form of the solution, the Lagrange multipliers must be determined numerically. This is done using the modified Newton - Rhapson technique developed in Chapter One.

2.3.1. Constraints and Source Terms

The mass and surface energy constraints are used in the solution in the same form as they were developed. The source terms for both are set to zero. The effects of this idealization are discussed later in this chapter.

If a velocity distribution is to be obtained, the source terms in the momentum and kinetic energy equations cannot both be set to zero. This would result in two constraints which say essentially:

- The first moment of the velocity distribution is one.

- The second moment of the velocity distribution is one.

The only distribution satisfying both of these constraints at any particular drop size is a single valued velocity distribution. Conceivably, a more complex velocity distribution, having different means for different diameters, could be obtained which would satisfy the constraints. This, however, would imply more information than is present in the constraints. None of the constraints show any reason for one size of droplets to have velocities different from others.

If solution of the system is attempted with the source terms in both the momentum and kinetic energy constraints set to zero, the solution does not converge. This is because the solution would require a delta function in velocity, which can be thought of as a degenerate form of the Gaussian terms in the solution. For λ_3 and λ_4 very large, the distribution will become sharper and sharper in velocity, until in the limit a delta function is obtained. Unfortunately, because of the discontinuous nature of this solution, it is not readily obtainable from a numerical model.

This problem is dealt with by arbitrarily setting the momentum source term S_{mv} to some small fraction of the initial expectation value. Typically a value of -0.05 has been used, reducing the expectation value from 1.0 to 0.95. This changes the variance of the velocity from zero to some small amount for the system, and allows a solution to be reached. The implications of this arbitrary change are discussed in Chapter Three.

2.3.2. Description of the Solution Algorithm

The solution algorithm for the initial distribution is described below. A listing of the program, with comments, is provided in Appendix B.

1. Make an initial guess at the Lagrange multiplier values. For most cases it is adequate to start with all zeroes, however a better guess will give faster convergence. The results of previous runs can serve this purpose, as well as allowing one to "work one's way out" to more sensitive cases which will not converge from a zero guess.

2. Equation (2.2.3) is then used to calculate λ_0 from the values of the other Lagrange multipliers. This ensures that the guessed distribution, being the whole of which the multipliers are parts, meets the normalization constraint. Without this, the guess would be fundamentally, and not just quantitatively, wrong.

3. The values in the Newton - Rhapson matrix of Figure 1.1 are calculated by performing the necessary integrals of the guessed distribution over the solution space. The integration is done using a simple summation over elements. This method was originally chosen because of its similarity to the summations in the discrete formalism. It has been retained because the calculation of the initial distribution does not represent a large part of the total execution time and because its simplicity does not introduce large errors for integration of the smooth continuous surfaces involved.

4. The matrix is solved for new values of λ_0, λ_1, λ_2, λ_3 and λ_4. The solver is a simple full matrix solver, translated from Fortran.

5. Again, equation (2.2.3) is used to calculate a new value for λ_0 which ensures that normalization is maintained. As described in Chapter One, this is the critical step for the solution to converge.

6. The change in the value of λ_0 is tested for convergence, and if the change is significant processing returns to step 3.

7. If the convergence test was passed then the current λ_i values represent a solution to the system of constraints.

2.4. The Resulting Distribution

As can be seen from equation (2.2.2), the PDF takes the form of a Gaussian in velocity, with a higher order curve in drop size. The velocity distribution is much broader (has a far larger variance) at small drop sizes than at large drop sizes. This is due to the mass weighting of the momentum and kinetic energy constraints. All other things being equal, the larger, more massive droplets contribute a far greater portion to the constraint integrals than the smaller, lighter droplets, and thus the constraint provides more information about the larger droplets, narrowing the distribution in that area. The velocity distribution is discussed more fully in Chapter Three.

2.4.1. The Integrated Size Distribution

Most of the published information in this area is empirical, and consists of droplet size distributions, rather than combined size - velocity distributions such as the one developed here. Accordingly, it is desirable to put the distribution in this form for comparison. This can be done by integrating the distribution described by equation (2.2.2) over the velocity domain, from zero to the maximum velocity, v_{*m}. If f_δ is a size PDF defined by $p = f_\delta d\delta_*$ then

$$f_\delta = \int_0^{v_{*m}} f \, dv_* \tag{2.4.1}$$

or, substituting in from equation (2.2.2),

$$f_\delta = \int_0^{v_{*m}} \exp(-\lambda_0 - \lambda_1\delta_*^2 - \lambda_2\delta_*^3 - \lambda_3\delta_*^3 v_* - \lambda_4\delta_*^3 v_*^2)\, dv_* \tag{2.4.2}$$

Rearranging this to express the velocity terms as the square of a sum, and bringing the independent terms outside the integral yields

$$f_\delta = \exp\left[-\lambda_0 - \lambda_1\delta_*^2 - \lambda_2\delta_*^3 + \frac{\lambda_3^2\delta_*^6}{4\lambda_4\delta_*^3}\right] \times$$

$$\int_0^{v_{*m}} \exp\left[-\left(\sqrt{\lambda_4\delta_*^3}\,v_* + \frac{\lambda_3\delta_*^3}{2\sqrt{\lambda_4\delta_*^3}}\right)^2\right] dv_* \qquad (2.4.3)$$

Define a dummy variable t as

$$t = \sqrt{\lambda_4\delta_*^3}\,v_* + \frac{\lambda_3\delta_*^3}{2\sqrt{\lambda_4\delta_*^3}} \qquad (2.4.4)$$

so that the integral portion of equation (2.4.3) can be rewritten as

$$\int_{(\lambda_3\delta_*^3)/(2\sqrt{\lambda_4\delta_*^3})}^{\sqrt{\lambda_4\delta_*^3}\,v_{*m} + (\lambda_3\delta_*^3)/(2\sqrt{\lambda_4\delta_*^3})} e^{-t^2}/\sqrt{\lambda_4\delta_*^3}\;dt \qquad (2.4.5)$$

Recalling that the error function is defined as

$$\mathrm{erf}(z) \equiv \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}dt \qquad (2.4.6)$$

this integral becomes

$$\left(\frac{\pi}{4\lambda_4\delta_*^3}\right)^{\frac{1}{2}}\left(\operatorname{erf}\left(v_{*m}\sqrt{\lambda_4\delta_*^3}+\frac{\lambda_3}{2}\sqrt{\delta_*^3/\lambda_4}\right)-\operatorname{erf}\left(\frac{\lambda_3}{2}\sqrt{\delta_*^3/\lambda_4}\right)\right) \tag{2.4.7}$$

When this is substituted back into equation (2.4.3) it becomes

$$f_\delta=\left(\frac{\pi}{4\lambda_4\delta_*^3}\right)^{\frac{1}{2}}\left(\operatorname{erf}\left(v_{*m}\sqrt{\lambda_4\delta_*^3}+\frac{\lambda_3}{2}\sqrt{\delta_*^3/\lambda_4}\right)-\operatorname{erf}\left(\frac{\lambda_3}{2}\sqrt{\delta_*^3/\lambda_4}\right)\right)\times$$

$$\exp\left(-\lambda_0-\lambda_1\delta_*^2-\left(\lambda_2-\frac{\lambda_3^2}{4\lambda_4}\right)\delta_*^3\right) \tag{2.4.8}$$

Equation (2.4.8) is a readily evaluable form of the combined distribution, expressed as a function of droplet size alone. This can now be compared with equivalent empirical distributions. This maximum entropy distribution of droplet size will be referred to as the ME distribution in later sections.

2.5. Number Distributions versus Mass Distributions

The PDF f_δ is a number distribution, meaning that the resulting probability is the proportion of the total number of droplets which lie in a particular size range. If n is the cumulative number fraction, then

$$f_\delta=\frac{dn}{d\delta_*} \tag{2.5.1}$$

In the literature results have generally been reported in terms of a mass PDF which relates to the proportion of the total mass lying in a particular size range.

A mass PDF f_m can be defined in terms of m, the cumulative mass fraction as

$$f_m = \frac{dm}{d\delta_*} \tag{2.5.2}$$

Recalling that mass is proportional to number of drops multiplied by diameter cubed gives the expression

$$f_m \propto \delta_*^3 \frac{dn}{d\delta_*} = f_\delta \delta_*^3 \tag{2.5.3}$$

The constant of proportionality can be evaluated by forcing normalization on f_m. The integral of f_m over the drop size range must be one. Recalling from the mass constraint and the definition of the mass mean diameter D_{30} that the integral of $f_\delta \delta_*^3$ over the drop size range must also be one yields a proportionality constant of unity, so that

$$f_m = f_\delta \delta_*^3 \tag{2.5.4}$$

This simple relationship allows an easy transition between the two types of PDF which is possible only when using the non-dimensional drop size, δ_*. The ME distribution appears as a mass PDF in the form

$$f_m = \left(\frac{\pi\delta_*^3}{4\lambda_4}\right)^{\frac{1}{2}} \left(\mathrm{erf}\left(v_{*m}\sqrt{\lambda_4\delta_*^3} + \frac{\lambda_3}{2}\sqrt{\delta_*^3/\lambda_4}\right) - \mathrm{erf}\left(\frac{\lambda_3}{2}\sqrt{\delta_*^3/\lambda_4}\right)\right) \times$$

$$\exp\left(-\lambda_0 - \lambda_1\delta_*^2 - \left(\lambda_2 - \frac{\lambda_3^2}{4\lambda_4}\right)\delta_*^3\right) \qquad (2.5.5)$$

only slightly different from the number distribution.

2.6. Empirical Distributions

A variety of empirical drop size distributions have been proposed in the literature. Four of the more significant distributions are presented here in both mathematical form and graphically for comparison.

2.6.1. The Rosin - Rammler Distribution

The Rosin - Rammler distribution[14] has been widely used to model sprays from a variety of atomizers. It is commonly expressed as

$$\frac{dn}{d\delta} = \frac{\alpha\delta^{(\alpha-4)}\exp\left(-(\delta/\bar{x})^{\alpha}\right)}{\bar{x}^{(\alpha-3)}\Gamma(1-3/\alpha)} \qquad (2.6.1)$$

In addition, an equation can be derived for the general weighted mean diameter of a Rosin - Rammler distribution.

$$D_{qp}^{(q-p)} = \frac{\bar{x}^{(q-p)}\Gamma\left(\frac{q-3}{\alpha}+1\right)}{\Gamma\left(\frac{p-3}{\alpha}+1\right)} \tag{2.6.2}$$

Recalling that

$$f_\delta = \frac{dn}{d\delta_*} = \frac{dn}{d\delta}\frac{d\delta}{d\delta_*} = D_{30}\frac{dn}{d\delta} \tag{2.6.3}$$

and defining $\bar{x}_* \equiv \bar{x}/D_{30}$, equation (2.6.1) can be rewritten as

$$f_\delta = \frac{\alpha\delta_*^{(\alpha-4)}\exp\left((\delta_*/\bar{x}_*)^\alpha\right)}{\bar{x}_*^{(\alpha-3)}\Gamma(1-3/\alpha)} \tag{2.6.4}$$

Equation (2.6.2) can be used to evaluate $\bar{x}_*$.

$$\bar{x}_*^{\,3} = \frac{\bar{x}^{(3-0)}}{D_{30}^{(3-0)}} = \frac{\Gamma\left(\frac{0-3}{\alpha}+1\right)}{\Gamma\left(\frac{3-3}{\alpha}+1\right)} \tag{2.6.5}$$

Simplification yields

$$\bar{x}_* = \left(\Gamma(1-3/\alpha)\right)^{\frac{1}{3}} \tag{2.6.6}$$

Substitution into equation (2.6.4) gives a completely non-dimensionalized equation for the Rosin - Rammler distribution.

$$f_\delta = \frac{\alpha \delta_*^{(\alpha-4)}}{\left(\Gamma(1-3/\alpha)\right)^{\alpha/3}} \exp\left(\frac{\delta_*^\alpha}{\left(\Gamma(1-3/\alpha)\right)^{\alpha/3}}\right) \tag{2.6.7}$$

Note that by non-dimensionalizing the distribution equation the size parameter $\bar{x}$ has been eliminated. The remaining expression describes fully the shape of the distribution, but does not locate it in an absolute size range.

2.6.2. The Nukiyama - Tanasawa Distribution

The Nukiyama - Tanasawa distribution[15] is described by

$$\frac{dn}{d\delta} = \frac{\delta^2 \beta b^{3/\beta}}{\Gamma(3/\beta)} \exp(-b\delta^\beta) \tag{2.6.8}$$

and the general weighted mean diameter is

$$D_{qp}^{(q-p)} = b^{-\left(\frac{q-p}{\beta}\right)} \frac{\Gamma\left((q+3)/\beta\right)}{\Gamma\left((p+3)/\beta\right)} \tag{2.6.9}$$

Applying a technique similar to that used for the Rosin - Rammler distribution yields an expression for the non-dimensionalized PDF containing only the exponent parameter β.

$$f_\delta = \frac{\delta_*^2 \beta\, \Gamma(6/\beta)}{\left(\Gamma(3/\beta)\right)^2} \exp\left(-\delta_*^\beta \left(\frac{\Gamma(6/\beta)}{\Gamma(3/\beta)}\right)^{\frac{\beta}{3}}\right) \tag{2.6.10}$$

2.6.3. The Log Probability Distribution

The Log Probability distribution[16] is based on the premiss that the drop size distribution is a simple one, but that the probability is traditionally distributed over the wrong scale for droplet size. The Log Probability distribution is a simple Gaussian distribution, but plotted over a logarithmic size scale y. The equations describing the distribution are

$$\frac{dn}{dy} = \frac{\gamma}{\sqrt{\pi}} \exp\left(-\left(\gamma y + \frac{3}{2\gamma}\right)^2\right) \tag{2.6.11}$$

and

$$y = \ln(\delta/\bar{x}) \tag{2.6.12}$$

The prime advantage of this form is that it allows the use of "probability paper" to graphically fit a distribution to a known set of data. It can be easily recast as a distribution over drop size, rather than over y.

$$\frac{dn}{d\delta} = \frac{dn}{dy}\frac{dy}{d\delta} = \frac{\gamma}{\delta\sqrt{\pi}} \exp\left(-\left(\gamma\ln(\delta/\bar{x}) + \frac{3}{2\gamma}\right)^2\right) \tag{2.6.13}$$

$$f_\delta = \frac{dn}{d\delta_*} = \frac{D_{30}\gamma}{\delta\sqrt{\pi}} \exp\left(-\left(\gamma\ln(\delta/\bar{x}) + \frac{3}{2\gamma}\right)^2\right) \tag{2.6.14}$$

The expression for the general weighted mean diameter for this distribution is

$$D_{qp} = \bar{x}\exp\left(\frac{p+q-6}{4\gamma^2}\right) \tag{2.6.15}$$

which can be rearranged as before to show that

$$\bar{x} = D_{30}\exp\left(\frac{3}{4\gamma^2}\right) \tag{2.6.16}$$

Substituting this into equation (2.6.14) and simplifying yields

$$f_\delta = \frac{\gamma}{\delta_*\sqrt{\pi}}\exp\left(-\left(\gamma\ln\delta_* + \frac{3}{4\gamma}\right)^2\right) \tag{2.6.17}$$

This is the non-dimensionalized Log Probability distribution equation. Note that it also depends on only a single parameter for shape, the size parameter having been eliminated.

2.6.4. The Upper Limit Distribution

The Upper Limit distribution[17] is a variation on the Log Probability distribution which defines a new drop size scale over which probability is distributed. For the Upper Limit distribution the scale is

$$y = \ln\left(\frac{a\,\delta^s}{\delta_m^s - \delta^s}\right) = \ln\left(\frac{a\,\delta_*^s}{\delta_{*m}^s - \delta_*^s}\right) \tag{2.6.18}$$

where s is a constant generally taken equal to one. The value a is a constant, describing the relative location of the maximum drop size, such that

$$a = \frac{\delta_m - \delta_{50}}{\delta_{50}} \tag{2.6.19}$$

where δ_m is the maximum droplet diameter and δ_{50} is the volume median droplet diameter. The Upper Limit distribution is expressed as a distribution of mass fraction

$$\frac{dm}{dy} = \frac{\kappa}{\sqrt{\pi}} \exp(-\kappa^2 y^2) \tag{2.6.20}$$

which can be recast in terms of the non-dimensional variables as either a mass PDF

$$f_m = \frac{\kappa}{\sqrt{\pi}} \left(\frac{1}{\delta_*} + \frac{1}{\delta_{*m} - \delta_*} \right) \exp\left[-\kappa^2 \left(\ln(\frac{a\,\delta_*}{\delta_{*m} - \delta_*}) \right)^2 \right] \tag{2.6.21}$$

or a size PDF

$$f_\delta = \frac{\kappa}{\delta_*^3 \sqrt{\pi}} \left(\frac{1}{\delta_*} + \frac{1}{\delta_{*m} - \delta_*} \right) \exp\left[-\kappa^2 \left(\ln(\frac{a\,\delta_*}{\delta_{*m} - \delta_*}) \right)^2 \right] \tag{2.6.22}$$

Both equations describe only the shape of the distribution, but not the drop size magnitudes associated with it.

2.6.5. A Basis for Comparing Distributions

The basic descriptive parameter in the maximum entropy formulation is the dimensionless sheet thickness τ_*. Applying the definition of D_{20} to the surface energy conservation constraint yields an expression for τ_* which can be evaluated for other distributions where the mass and surface mean diameters are known.

$$\tau_* = \frac{1}{3\left[(D_{20}/D_{30})^2 - S_e\right]} \qquad (2.6.23\ (a))$$

or, for S_e set to zero,

$$\tau_* = \frac{(D_{30}/D_{20})^2}{3} \qquad (2.6.23\ (b))$$

Using equation (2.6.23 (b)) for the Rosin-Rammler distribution gives

$$\tau_* = \frac{\left(\Gamma(1-3/\alpha)\right)^{\frac{1}{3}}}{3\Gamma(1-1/\alpha)} \qquad (2.6.24)$$

For the Nukiyama - Tanasawa distribution

$$\tau_* = \frac{\left(\Gamma(6/\beta)\right)^{\frac{2}{3}}\left(\Gamma(3/\beta)\right)^{\frac{1}{3}}}{3\Gamma(5/\beta)} \qquad (2.6.25)$$

For the Log Probability distribution

$$\tau_* = \frac{1}{3}\exp\left(\frac{1}{2\gamma^2}\right) \tag{2.6.26}$$

For the Upper Limit distribution

$$\tau_* = \frac{\left(1 + 3a\exp\left(\frac{1}{4\kappa^2}\right) + 3a^2\exp\left(\frac{1}{\kappa^2}\right) + a^3\exp\left(\frac{9}{4\kappa^2}\right)\right)^{\frac{1}{3}}}{3\left(1 + a\exp\left(\frac{1}{4\kappa^2}\right)\right)} \tag{2.6.27}$$

Given τ_*, and for the Upper Limit distribution a, these expressions can be used to define distributions which have the same essential characteristics as the maximum entropy formulation. For the present work a has been taken equal to one, as Mugele and Evans did in their work.

2.7. Results and Discussion

The parameters for each of the distributions at three different τ_* values are given in Table 2.1. Number and mass PDFs for each of the three conditions are shown in Figures 2.3 (a-f). The ME distribution was evaluated for a momentum source term $S_{mv} = -0.05$ as discussed earlier. A solution space having the range $0 \leq \delta_* \leq 2.5$ and $0 \leq v_* \leq 2.0$ was used for the integrations. For more detailed numerical results see Appendix A.

Figure 2.3 (a) Comparison of Number Distributions for $\tau_* = 0.350$

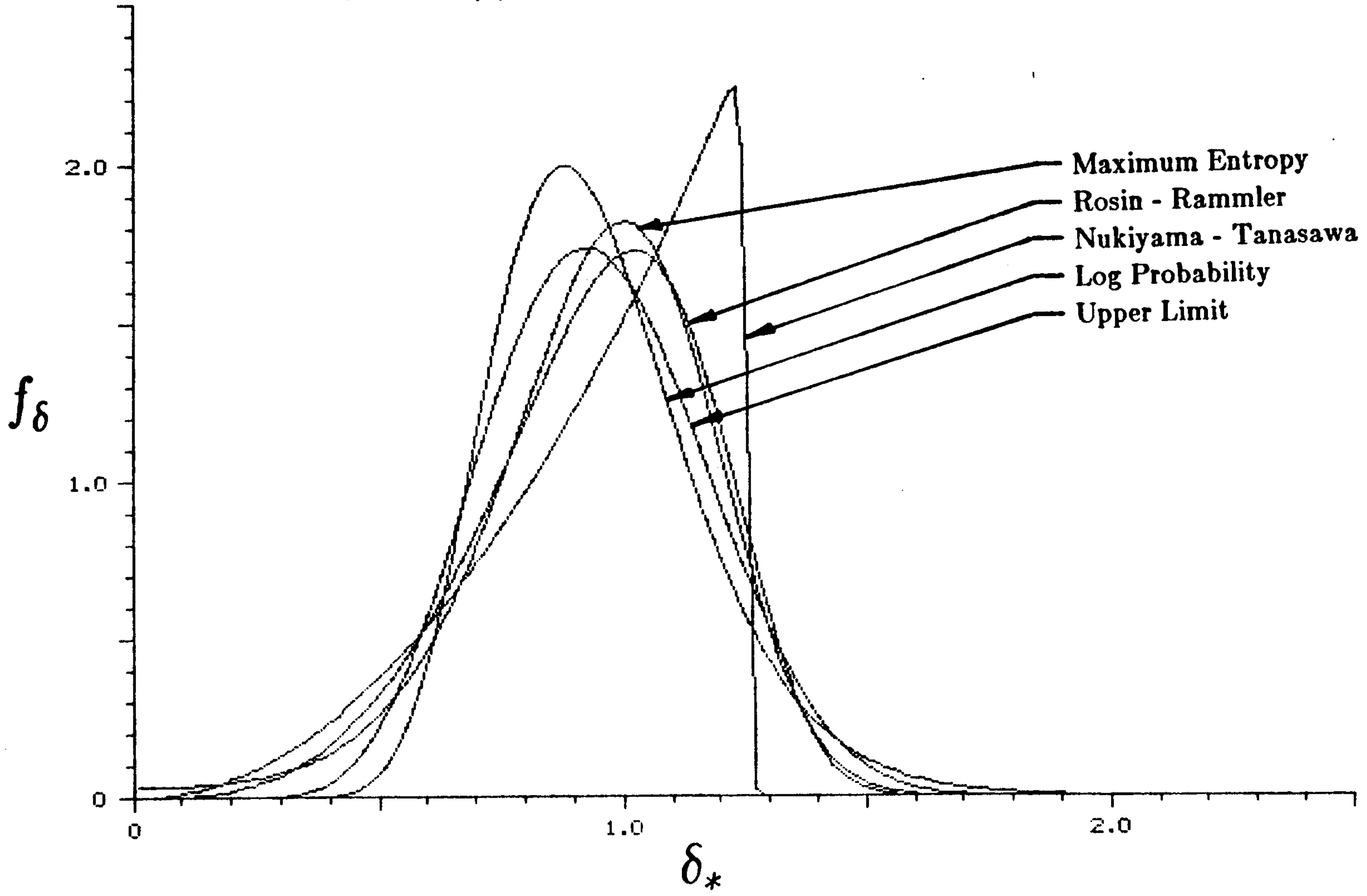


Figure 2.3 (b) Comparison of Mass Distributions for $\tau_* = 0.350$

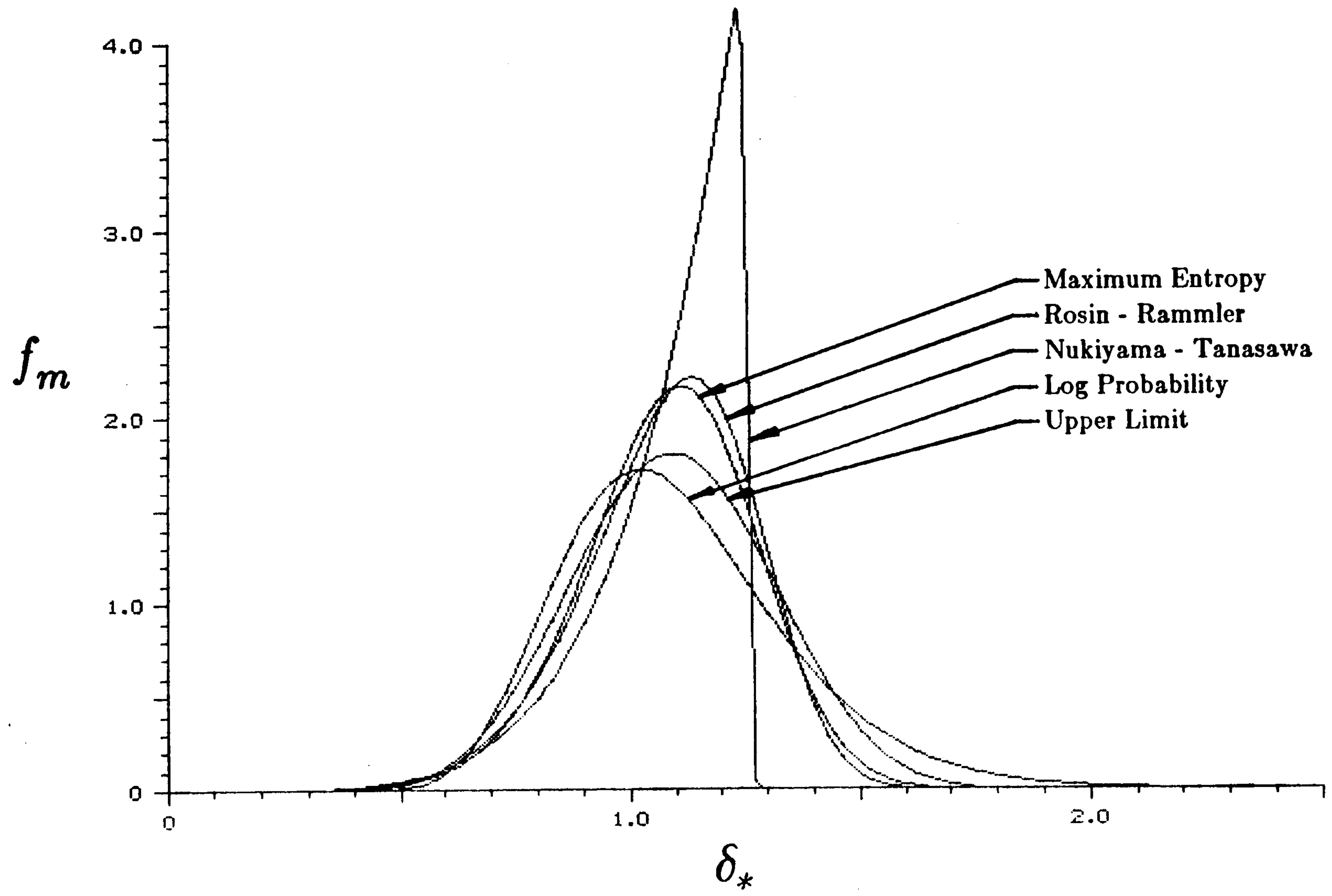


Figure 2.3 (c) Comparison of Number Distributions for $\tau_* = 0.375$

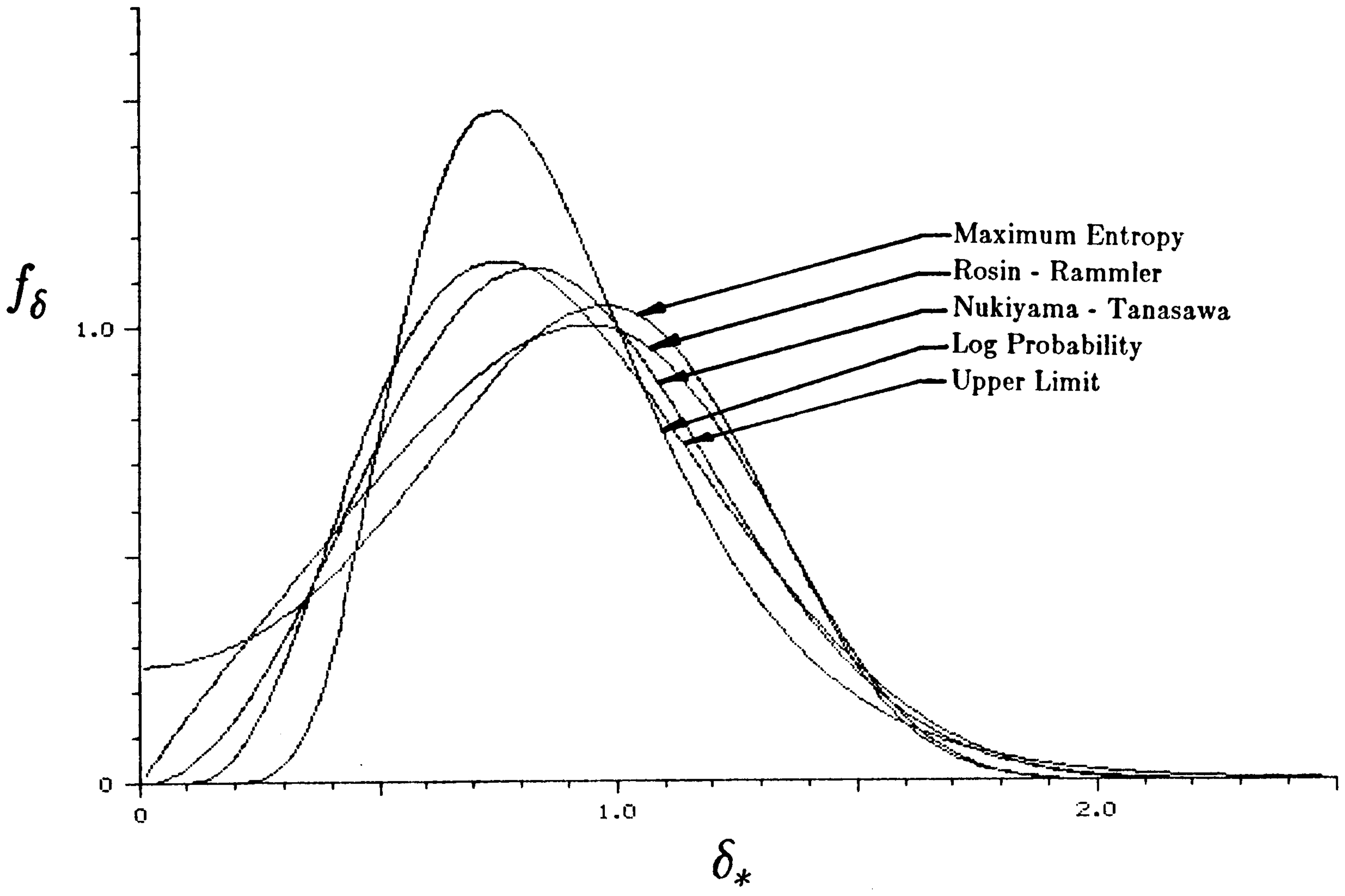


Figure 2.3 (d) Comparison of Mass Distributions for $\tau_* = 0.375$

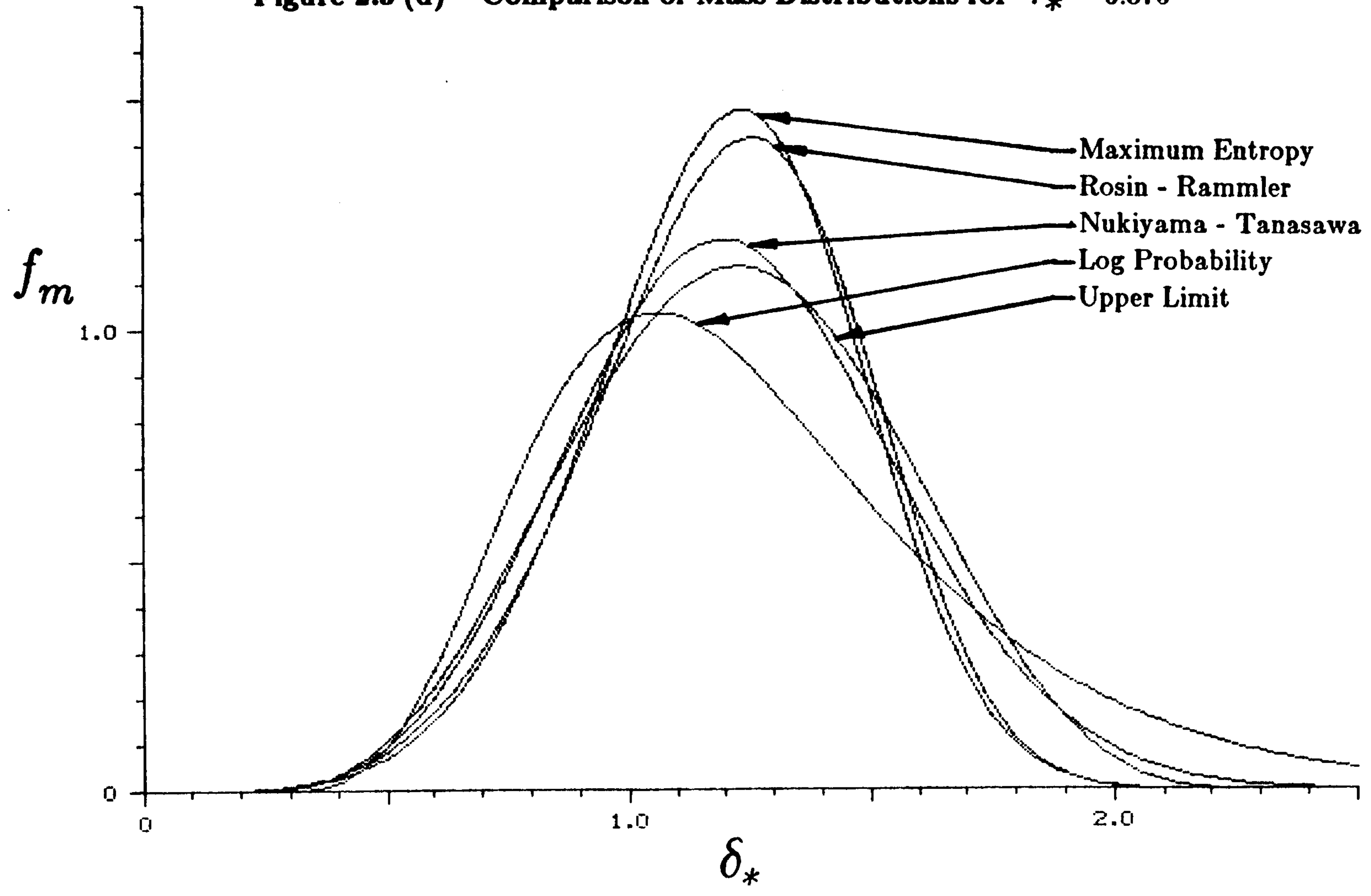


Figure 2.3 (e) Comparison of Number Distributions for $\tau_* = 0.400$

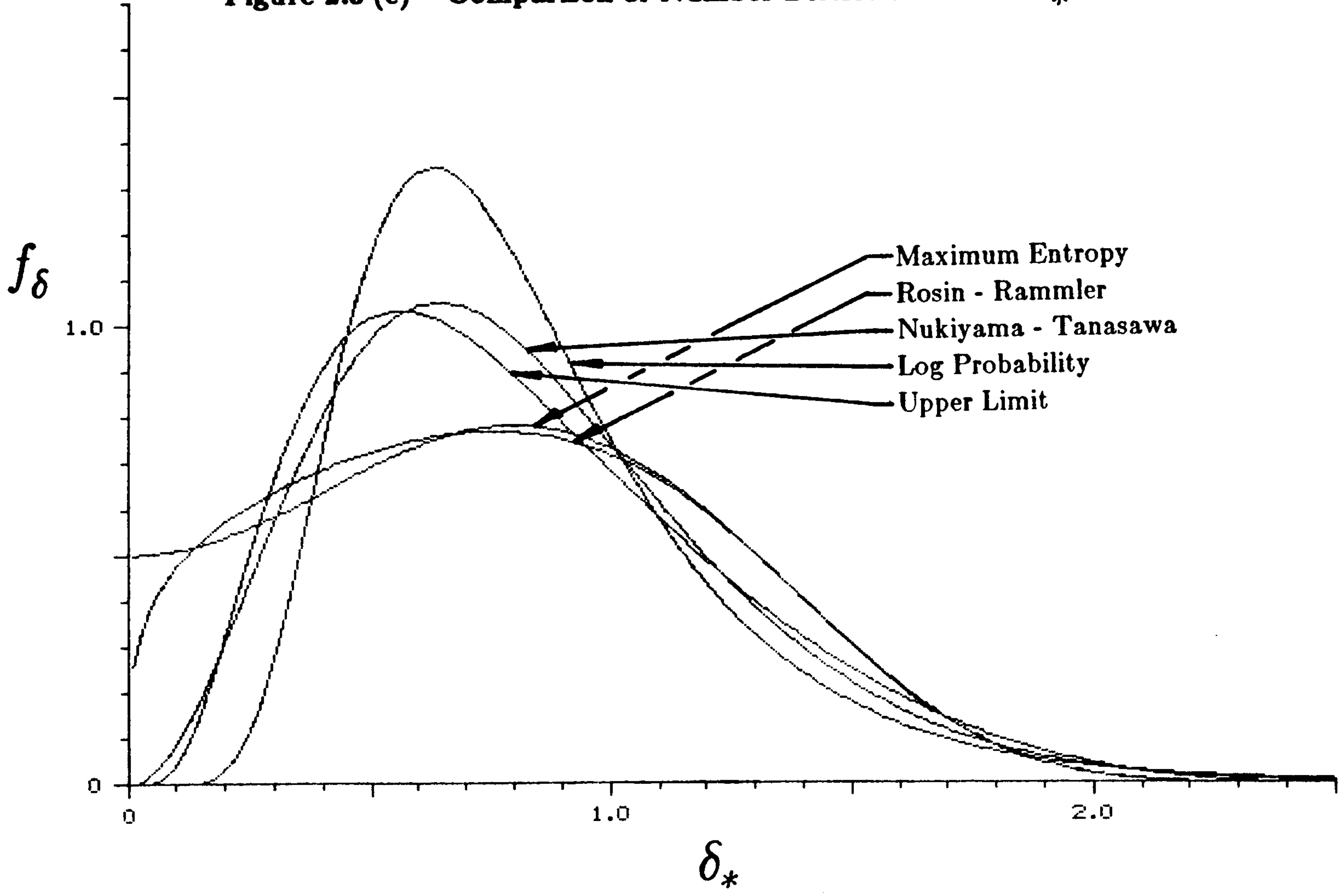
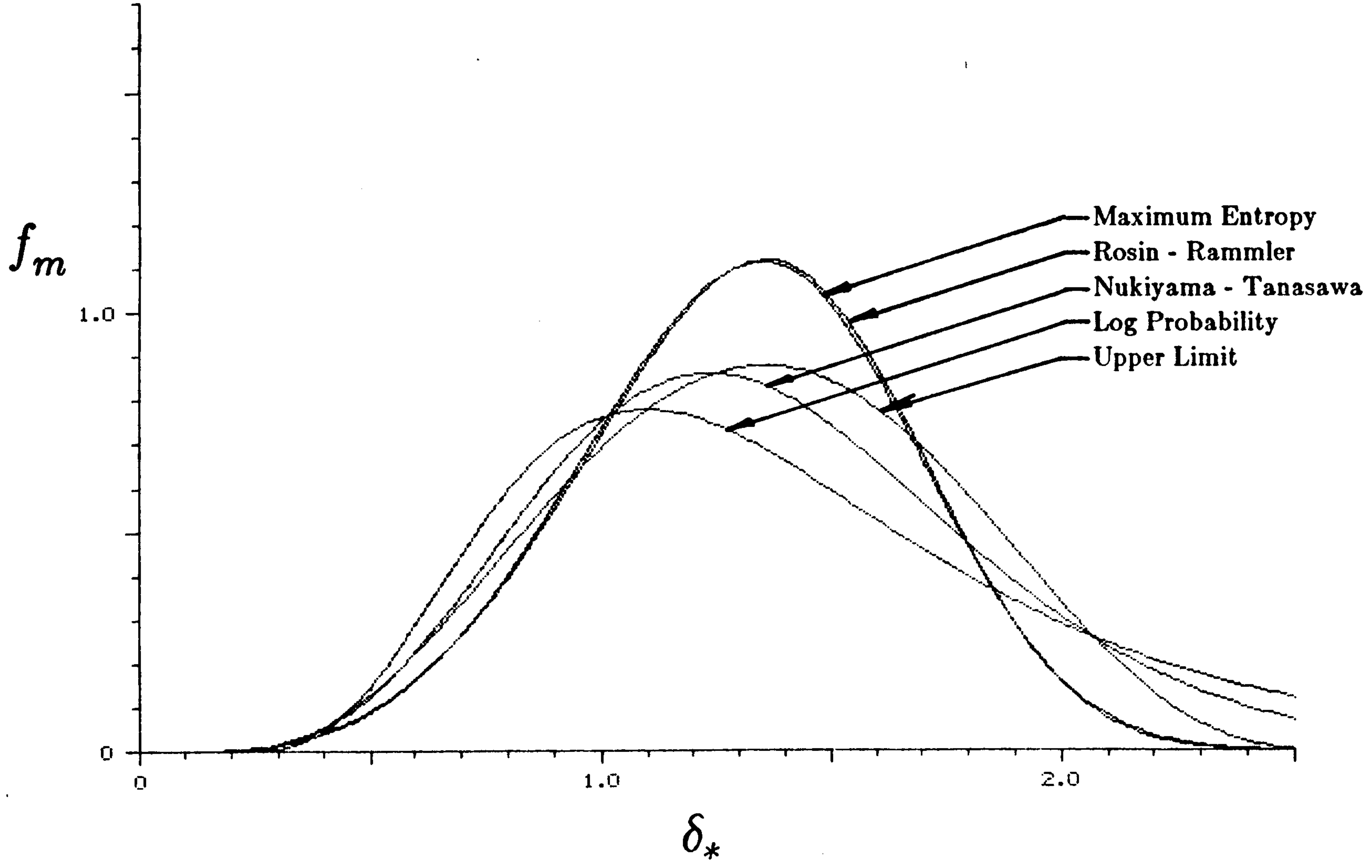


Figure 2.3 (f) Comparison of Mass Distributions for $\tau_* = 0.400$



Maximum Entropy	Rosin Rammler	Nukiyama Tanasawa	Log Probability	Upper Limit		
τ_*	α	β	γ	κ	a	δ_{*m}
0.350	6.92	158.6	3.20	1.75	1.00	2.19
0.375	4.93	2.45	2.06	1.23	1.00	2.45
0.400	4.27	1.41	1.66	1.05	1.00	2.70

Table 2.1. Parameters for the five distributions at different τ_* values.

As shown in Figures 2.3 (a-f), all five distributions have similar features. The Rosin-Rammler and the maximum entropy distribution are particularly close in shape, with only minor deviations in the magnitude and location of the peak and in the behaviour of the distributions for diameters near zero.

The present work would not provide much, if any, advantage if its object was to derive another distribution with which to correlate data. However, the maximum entropy formulation has a fundamental analytical basis. The fact that it is in reasonable agreement with certain empirical distributions lends legitimacy to those distributions. The analytical foundation of the ME distribution also indirectly explains empirical distributions which differ from it. In this light, such distributions reflect physical processes which are not included in the constraints which give rise to the ME PDF. The maximum entropy formalism provides a means for adding information about such processes and deriving the PDF which corresponds exactly to the new information without the risk of inadvertently introducing any extraneous information, or bias.

The ME droplet size distribution derived in this work can be considered as an ideal case. It represents the results of a loss-free atomization process in which liquid mass, momentum, surface energy and kinetic energy are all conserved. For a real atomization process these conditions would be approximations, more accurate for some atomizers than for others.

For example, is it reasonable to assume conservation through the sheet breakup region? If evaporation is small then mass conservation is a good approximation. If drag losses and conversion to surface energy are small then conservation of momentum and kinetic energy are good approximations. Surface energy is probably not conserved during breakup, but is lost since it is the reduction in surface area which drives the breakup of liquid sheets and ligaments. In any case, the change in the surface energy would likely be small in relation to the overall kinetic energy for common atomizers, and so would not substantially affect S_{ke}. This change would be reflected in the source term S_s which could introduce the surface tension into the distribution.

As shown in equation (2.6.23 (a)), the link between the ME distribution and the others is defined by the resulting surface to volume ratio. This can be expressed as a combination of S_s and τ_*. This ideal case where $S_s = 0$ may be unrealistic, however, the basis for comparison of distributions is still valid.

If one were to combine the ME distribution with a "front end" calculation which predicts some average droplet diameter as a function of the parameters of the sheet and the properties of the liquid, the resulting distribution would be over absolute size rather than relative size, and would show a dependence on surface tension and other liquid properties.

Finally, the region in space where the predicted PDF could be expected to occur cannot be localized in relation to the atomizer, but it can be specified precisely. It is that region where the random processes governed by the original constraints have gone to completion, and no processes governed by different constraints have had a measurable influence.

Chapter Three

DOWNSTREAM DEVELOPMENT OF THE VELOCITY DISTRIBUTION

3.1. Downstream Behaviour

As the processes represented by the atomization model of Chapter Two go to completion, the relative importance of interactions with the surrounding gas increases. These interactions will include changes in droplet velocities due to drag, as well as a reduction in droplet size due to evaporation.

The model used here for the downstream development of the velocity distribution is very simple. The air flow field is assumed to have a uniform and constant velocity, with no turbulent dispersion of droplets taking place. Evaporation is ignored so that droplets will retain their mass throughout the downstream development. For drag calculations the droplet is treated as a solid sphere, ignoring any effects due to internal circulation or deformation. Although these are very substantial simplifications, this model provides some insight into the importance of different factors to the downstream velocity distributions.

3.2. Downstream Velocity of a Single Droplet

Before the downstream velocity behaviour of an arbitrary number of droplets with distributed velocities can be modelled, a model of the behaviour of a single droplet is needed. If the spray under consideration is not dense, then collisions can be ignored and the behaviour of the droplets collectively will be the sum of the behaviours of each of the individual drops in the spray.

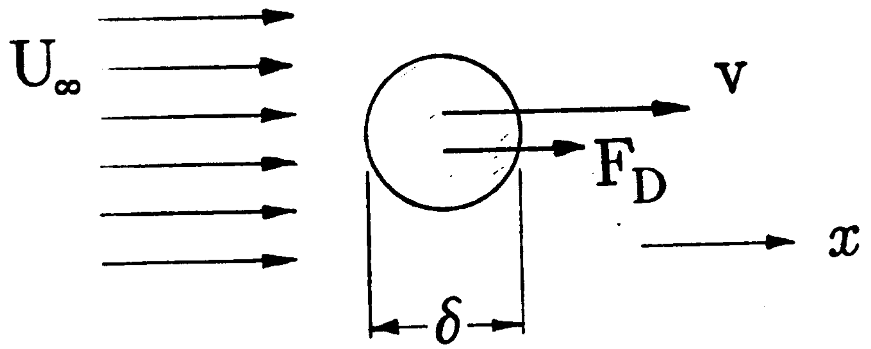


Figure 3.1 A droplet in an air flow field.

Consider the simple system pictured in Figure 3.1. A spherical droplet of diameter δ is moving in the positive x direction with some velocity v through an air flow field of constant and uniform velocity U_∞. A relative air velocity can be defined as

$$U_r \equiv U_\infty - v \tag{3.2.1}$$

so that it is positive for an air velocity greater than the droplet velocity. This gives a signed Reynolds Number

$$Re \equiv \frac{U_r \delta}{\nu} \tag{3.2.2}$$

and a signed drag coefficient defined by the expression

$$F_D = C_D \frac{\rho_a U_r^2 A_{proj}}{2} \tag{3.2.3}$$

F_D is a drag force in the positive x direction. F_D will have, from examination of Figure 3.1, the same sign as U_r. From the defining equations the signed Reynolds Number and drag coefficient will also have that same sign. For example; if the air velocity is lower than the droplet velocity, all four of these quantities will be negative.

Equation (3.2.3) can be rewritten as

$$F_D = C_D \frac{\rho_a \pi U_r^2 \delta^2}{8} \tag{3.2.4}$$

The droplet will experience an acceleration governed by Newton's Law.

$$\frac{dv}{dt} = \frac{F_D}{m} = C_D \frac{\rho_a \pi U_r^2 \delta^2}{8} \times \frac{6}{\rho_l \pi \delta^3} \tag{3.2.5}$$

Multiplying both sides by D_{30}/V^2 gives

$$\frac{d(\mathrm{v/V})}{d(t\mathrm{V/D_{30}})} = \frac{3\rho_a(\mathrm{U_r/V})^2\mathrm{C_D}}{4\rho_l(\delta/\mathrm{D_{30}})} \tag{3.2.6}$$

If dimensionless diameter, velocity and time are defined, respectively, as

$$\delta_* \equiv \frac{\delta}{\mathrm{D_{30}}} \tag{3.2.7}$$

$$\mathrm{v}_* \equiv \frac{\mathrm{v}}{\mathrm{V}} \tag{3.2.8}$$

$$t_* \equiv \frac{t\mathrm{V}}{\mathrm{D_{30}}} \tag{3.2.9}$$

then equation (3.2.6) becomes an expression for a non-dimensional acceleration.

$$\frac{d\mathrm{v}_*}{dt_*} = \frac{3\rho_a\mathrm{C_D}}{4\rho_l\delta_*}\mathrm{U}_{r*}^2 \tag{3.2.10}$$

Noting that

$$\frac{d\mathrm{U}_{r*}}{dt_*} = \frac{d\mathrm{U}_{\infty*}}{dt_*} - \frac{d\mathrm{v}_*}{dt_*} = -\frac{d\mathrm{v}_*}{dt_*} \quad \text{for } \mathrm{U}_\infty \text{ constant} \tag{3.2.11}$$

equation (3.2.10) can be rewritten as

$$\frac{d\mathrm{U}_{r*}}{dt_*} = -\frac{3\rho_a\mathrm{C_D}}{4\rho_l\delta_*}\mathrm{U}_{r*}^2 \tag{3.2.12}$$

Integration yields

$$U_{r*} = \left(\frac{3\rho_a C_D}{4\rho_l \delta_*} t_* + C\right)^{-1} \tag{3.2.13}$$

where C is a constant of integration.

This manipulation requires that $(\rho_a C_D)/(\rho_l \delta_*)$ be nearly constant with time. This is approximation is valid when applied locally over a small increment in time and distance, as is done in this model. For the drag coefficient this is similar to a fully developed flow assumption, and should be quite reasonable. Assuming that the density ratio and drop size are constant requires that temperature changes be small and that evaporation be negligible.

An expression for droplet velocity is obtained by expanding the relative velocity.

$$v_* = U_{\infty *} - \left(\frac{3\rho_a C_D}{4\rho_l \delta_*} t_* + C\right)^{-1} \tag{3.2.14}$$

Applying the boundary condition that $v_* = v_{0*}$ at time $t_* = 0$ allows the evaluation of the constant of integration C.

$$v_{1*} = U_{\infty *} - \left(\frac{3\rho_a C_D}{4\rho_l \delta_*} t_{1*} + \frac{1}{U_{\infty *} - v_{0*}}\right)^{-1} \tag{3.2.15}$$

This can also be expressed as a spatial rather than temporal relation by defining some average velocity $\bar{v}_*$ as

$$\bar{v}_* = \frac{x_{1*} - x_{0*}}{t_{1*} - t_{0*}} = \frac{\Delta x_*}{t_{1*}} \qquad \text{for } t_{0*} = 0 \tag{3.2.16}$$

where x_* is a non-dimensionalized downstream distance, to yield

$$v_{1*} = U_{\infty *} - \left[\frac{3\rho_a C_D \Delta x_*}{4\rho_l \delta_* \bar{v}_*} + \frac{1}{U_{r0*}} \right]^{-1} \tag{3.2.17}$$

Note that U_{r0*} is the initial relative velocity, $U_{\infty *} - v_{0*}$. The average velocity $\bar{v}_*$ can then be approximated as a linear average of the two velocities.

$$\bar{v}_* \approx \frac{v_{1*} + v_{0*}}{2} \tag{3.2.18}$$

This is substituted into equation (3.2.17), and the result is rearranged as

$$v_{1*}^2 + \frac{3\rho_a C_D \Delta x_* U_{r0*}}{2\rho_l \delta_*} v_{1*} - v_{0*}^2 - \frac{3\rho_a C_D \Delta x_* U_{r0*}}{2\rho_l \delta_*} U_{\infty *} = 0 \tag{3.2.19}$$

If the drag coefficient is taken to be a constant of some average value over the distance Δx_*, then this expression can be evaluated from the general solution for the roots of a quadratic equation.

$$v_{1*} = -\frac{3\rho_a C_D \Delta x_* U_{r0*}}{4\rho_l \delta_*}$$

$$\pm \left[\left(\frac{3\rho_a C_D \Delta x_* U_{r0*}}{4\rho_l \delta_*} \right)^2 + \left(v_{0*}^2 + \frac{3\rho_a C_D \Delta x_* U_{r0*}}{2\rho_l \delta_*} U_{\infty *} \right) \right]^{\frac{1}{2}} \tag{3.2.20}$$

The term

$$\frac{3\rho_a C_D \Delta x_* U_{r0*}}{2\rho_l \delta_*} \tag{3.2.21}$$

must always be positive, as C_D and U_{r0*} are of the same sign and all other elements are positive. Then, by examination, equation (3.2.20) always gives one positive root and one negative root for $U_{x*} > 0$. If v_{0*} is also positive then the negative root is absurd and the solution is

$$v_{1*} = -\frac{3\rho_a C_D \Delta x_* U_{r0*}}{4\rho_l \delta_*} + \left[\left(\frac{3\rho_a C_D \Delta x_* U_{r0*}}{4\rho_l \delta_*}\right)^2 + \left(v_{0*}^2 + \frac{3\rho_a C_D \Delta x_* U_{r0*}}{2\rho_l \delta_*} U_{x*}\right)\right]^{\frac{1}{2}} \tag{3.2.22}$$

This expression must be evaluated iteratively to be sure the correct value of the average drag coefficient is used. The sequence is:

1. Evaluate C_D at v_{0*} and use this value to obtain v_{1*}.
2. Evaluate C_D for a relative velocity based on $\bar{v}_* = (v_{1*} + v_{0*})/2$.
3. Evaluate a new v_{1*} based on the current C_D. If v_{1*} has not converged then return to step 2.

This procedure allows the evaluation of droplet velocity a short distance Δx_* downstream. If desired this solution could be "marched" downstream to get the spatial velocity history of one drop over an extended distance.

In this model the drag coefficient is evaluated from the relation

$$C_D \approx \frac{24}{Re} + \frac{6}{1+Re} + 0.4 \qquad 0 < Re \leq 2\times10^5 \qquad (3.2.23)$$

which is given by White[18] as an empirical curve fit for drag coefficients of solid spheres within the given range of Reynolds Numbers. Although this correlation is less accurate than the use of several separate correlations, it is sufficient to show the general behaviour of the model.

3.3. An Initial Temptation

One is initially tempted to apply the simple drag model of the previous section in an integral fashion over the entire distribution. This will result in values for the changes in total momentum and total kinetic energy for the entire system. These changes could then be applied to obtain new expectation values for the constraints, and the system could be re-solved for a new size-velocity distribution.

This represents the same type of error as was discussed in Chapter Two, in regard to conservation of energy. The simple approach outlined above ignores known information about the physical behaviour of the system. If the spray is not dense, and hence collisions unimportant, then the momentum and kinetic energy lost by a droplet of a particular size are lost by that droplet alone. If the results are integrated over the entire solution space and applied in an average sense, then that information is lost.

If calculations are actually carried out on this basis, the result is a downstream development that does not show the strong variations in behaviour with droplet size that would be expected to occur. The physics of acceleration due to drag make it clear that the velocities of small droplets will move toward the surrounding gas velocity much more readily than the velocities of larger droplets. Inherent in this physical model is the understanding that there is no momentum transfer between droplets, except through the rather weak link of interaction with the surrounding gas. When drag losses are applied in an average sense, there is no expression of this localization of accelerations. The result is a model that, in effect, permits momentum and kinetic energy transport between droplets in order to maximize the entropy of the distribution. This is clearly not realistic.

3.4. The Separation of Drop Sizes

The problem of excessive averaging may be eliminated completely by considering each droplet individually. This would require substantial computing resources and, for this simple model, would not provide results of corresponding accuracy.

A large part of the averaging error can be removed by considering each droplet size range separately. The droplet size range is divided into 25 segments, each of which is $0.1\delta_*$ wide, to cover the range from $\delta_* = 0$ to $\delta_* = 2.5$. Calculations can then be made for each range separately, making the assumption that all the droplets have the same diameter, namely the arithmetic mean diameter for that range.

3.4.1. Downstream Moments of a PDF

The information available from the initial solution for each size range is limited. The number fraction of droplets falling into that size range may be obtained from $\bar{f}_\delta$. The velocity distribution within the size range consists of a doubly truncated Gaussian, which was discussed in Chapter One.

This doubly truncated Gaussian is fully described by three pieces of information. These are the mean velocity, the variance of the velocity, and the range of permitted velocities. If this distribution is to be propagated downstream it would be unreasonable to expect the information content of the downstream distribution to be qualitatively higher than the original. Using this logic it is reasonable to apply the drag model to obtain only new values of the mean and the variance, from which the new distribution may be inferred.

The doubly truncated Gaussian distribution is of the form

$$f_v = \exp(-\lambda_0 - \lambda_1 v_* - \lambda_2 v_*^2) \tag{3.4.1}$$

The Lagrange multipliers are initially determined from the solution for the size-velocity PDF f, and are subsequently determined from constraints on the expectation values for v_* and v_*^2. These constraints are expressions of the mean and variance, respectively.

Having obtained the current distribution f_v, it is necessary to evaluate the expectation values for the constraints which will define the distribution some small distance Δx_* downstream. The constraints are

$$\int_0^{v_{*m}} f_v v_* \, dv_* = \langle v_* \rangle \tag{3.4.2}$$

and

$$\int_0^{v_{*m}} f_v v_*^2 \, dv_* = \langle v_*^2 \rangle \tag{3.4.3}$$

Define $v_{*new}(v_*)$ as the velocity that will be achieved by a droplet, having current velocity v_*, after it has moved Δx_* further downstream. The calculations required for the function $v_{*new}(\,)$ are given in section 3.2. The expectation values for the distribution at a position Δx_* further downstream may then be calculated as

$$\langle v_* \rangle_{new} = \int_0^{v_{*m}} f_v v_{*new}(v_*) \, dv_* \tag{3.4.4}$$

and

$$<v_*^2>_{new} = \int_0^{v_{*m}} f_v \left(v_{*new}(v_*)\right)^2 dv_* \qquad (3.4.5)$$

Knowing the expectation values, a new solution can be found for the doubly truncated Gaussian. This is done using a special solution method presented by Tribus in reference 12 (pp. 141-144). Using this method one may obtain the new velocity distribution for each of the drop size ranges in turn, and then repeat the entire process to march the solution downstream by a further Δx_*.

In the numerical solution the integrations are made over the entire velocity range if the distribution is broad. When the distribution narrows substantially the integration bounds are also narrowed to maintain sufficient resolution in the area of interest. The integration is then performed within a range of ± 3 standard deviations of the mean and the results are corrected based on the integral of the PDF over the same range. If the standard deviation is less than one percent of the mean, then the distribution is considered as a single valued velocity and calculations are made accordingly.

Since the droplet velocities vary, the set of droplets passing one location at any particular time will not all arrive at a downstream position at the same time. If, however, the spray system is steady in time then this problem may be ignored. The droplets passing any point will not share the same starting time from the nozzle, but for steady flow they will have experienced the same intermediate conditions, and thus the starting time will be immaterial.

3.5. Results and Discussion

The results of this velocity solution are presented graphically in Figures 3.2 (a-f) for the same three cases as were presented in Chapter Two. These figures are for a sheet having an initial velocity of 10 metres per second breaking up and moving downstream in a uniform air flow field having a velocity of 5 metres per second. The D_{30} is 100 microns and properties used are for number 2 fuel oil in air at 20 degrees Celsius. Because all of the solutions show similar characteristics, they will be discussed as a group.

The initial velocity distribution shows a near uniform mean velocity at just less than the initial sheet velocity. This reduction is due to negative momentum source term used to generate the distribution, as discussed in Chapter Two. Also, as noted in Chapter Two, the variance is initially very high for the smaller droplet sizes and lower for the larger droplet sizes.

As the solution is propagated downstream the mean velocity values behave exactly as expected. All of the mean velocities move towards the air velocity. The rate at which the mean velocity moves towards the air velocity is larger for smaller drop sizes and larger relative velocities. This is explained by a simple examination of the drag relation.

The behaviour of the variance values is much more interesting. The variances at all drop sizes show substantial reductions as the drops move downstream. This is easily explained for the very small droplets whose velocities are collapsing quickly onto the air velocity, however the effect was not expected to be so pronounced in the larger drop sizes. The phenomenon results from the

Figure 3.2 (a) Downstream Mean Droplet Velocities for $\tau_* = 0.350$

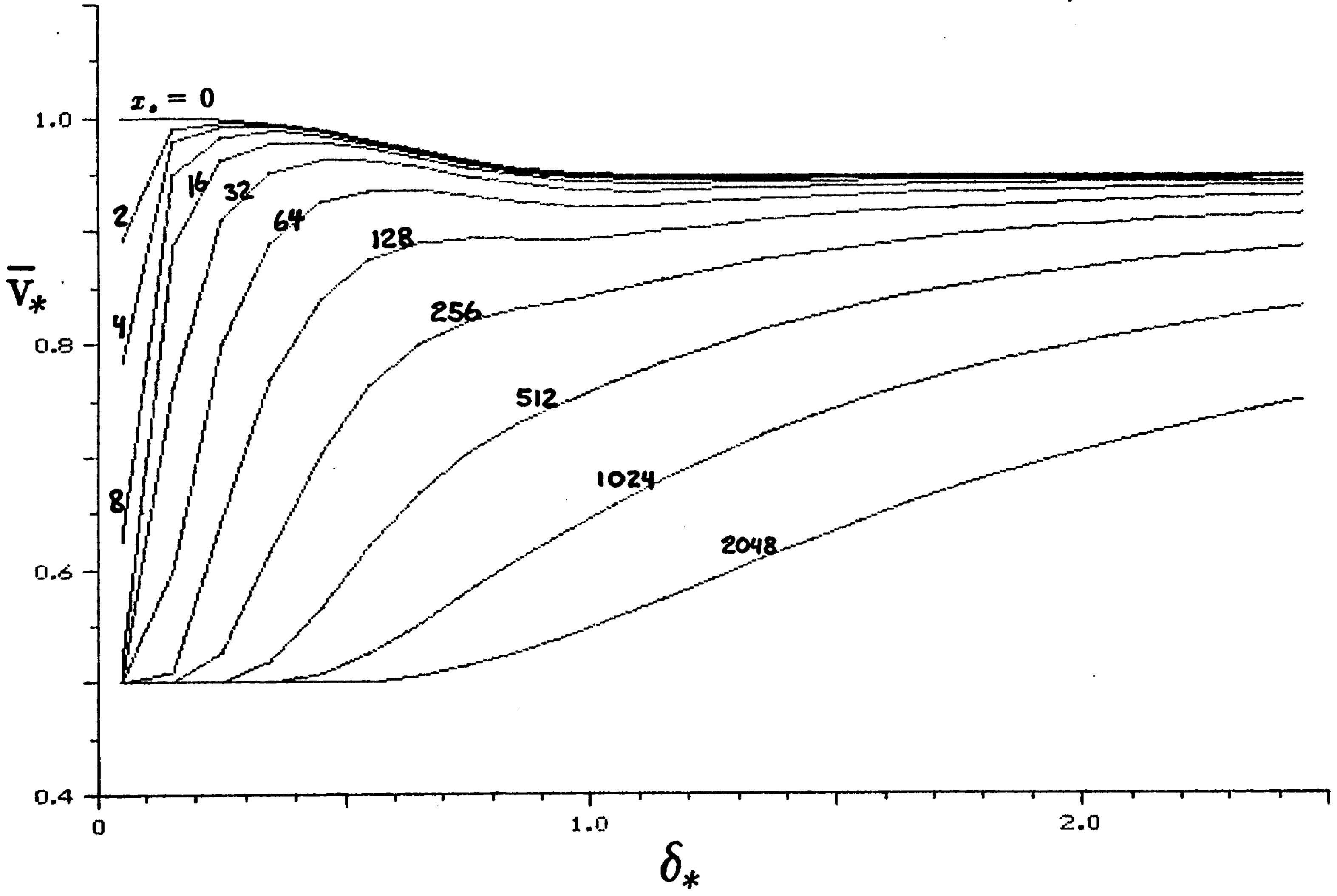


Figure 3.2 (b) Downstream Variance of Droplet Velocities for $\tau_* = 0.350$

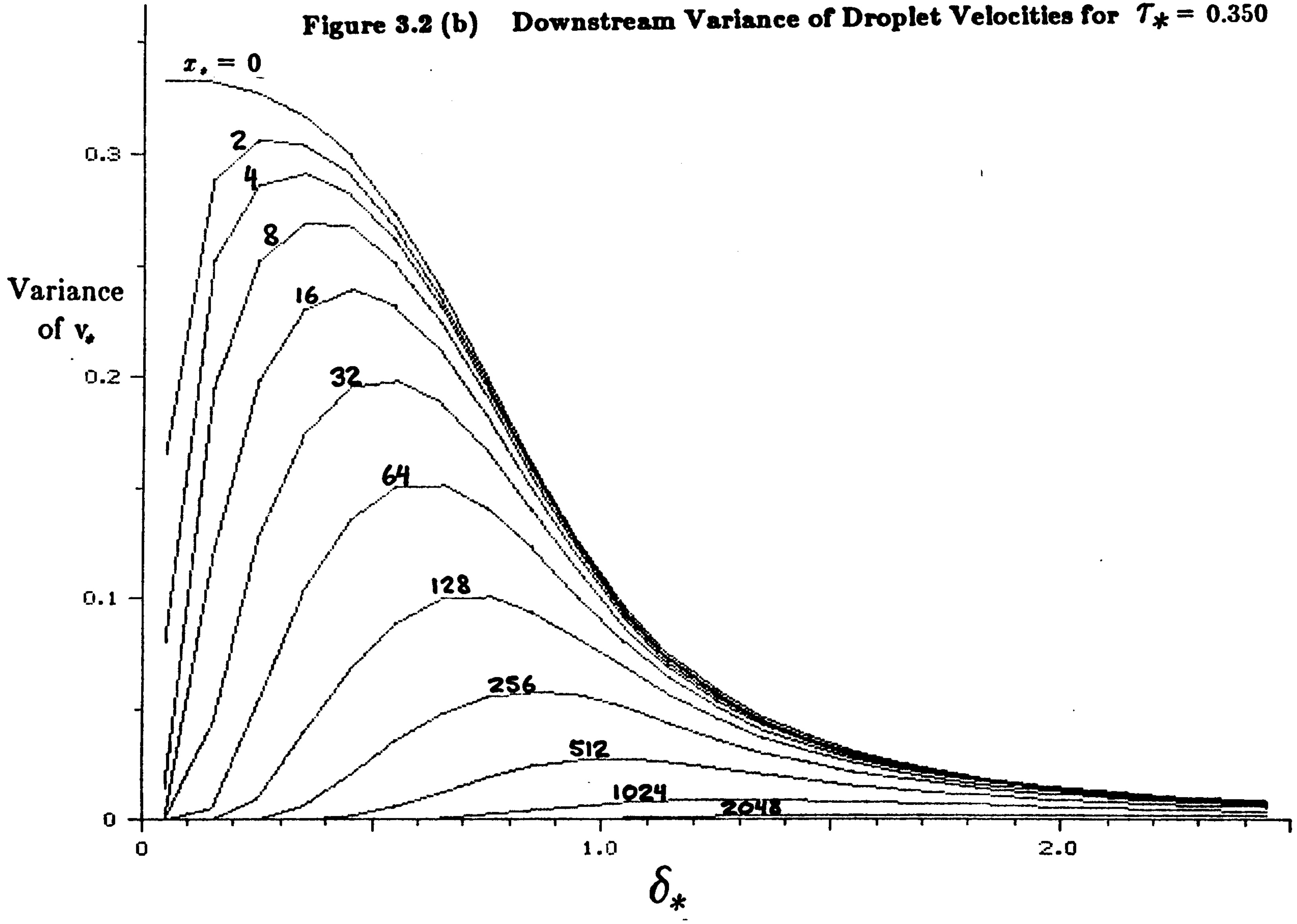


Figure 3.2 (c) Downstream Mean Droplet Velocities for $\mathcal{T}_* = 0.375$

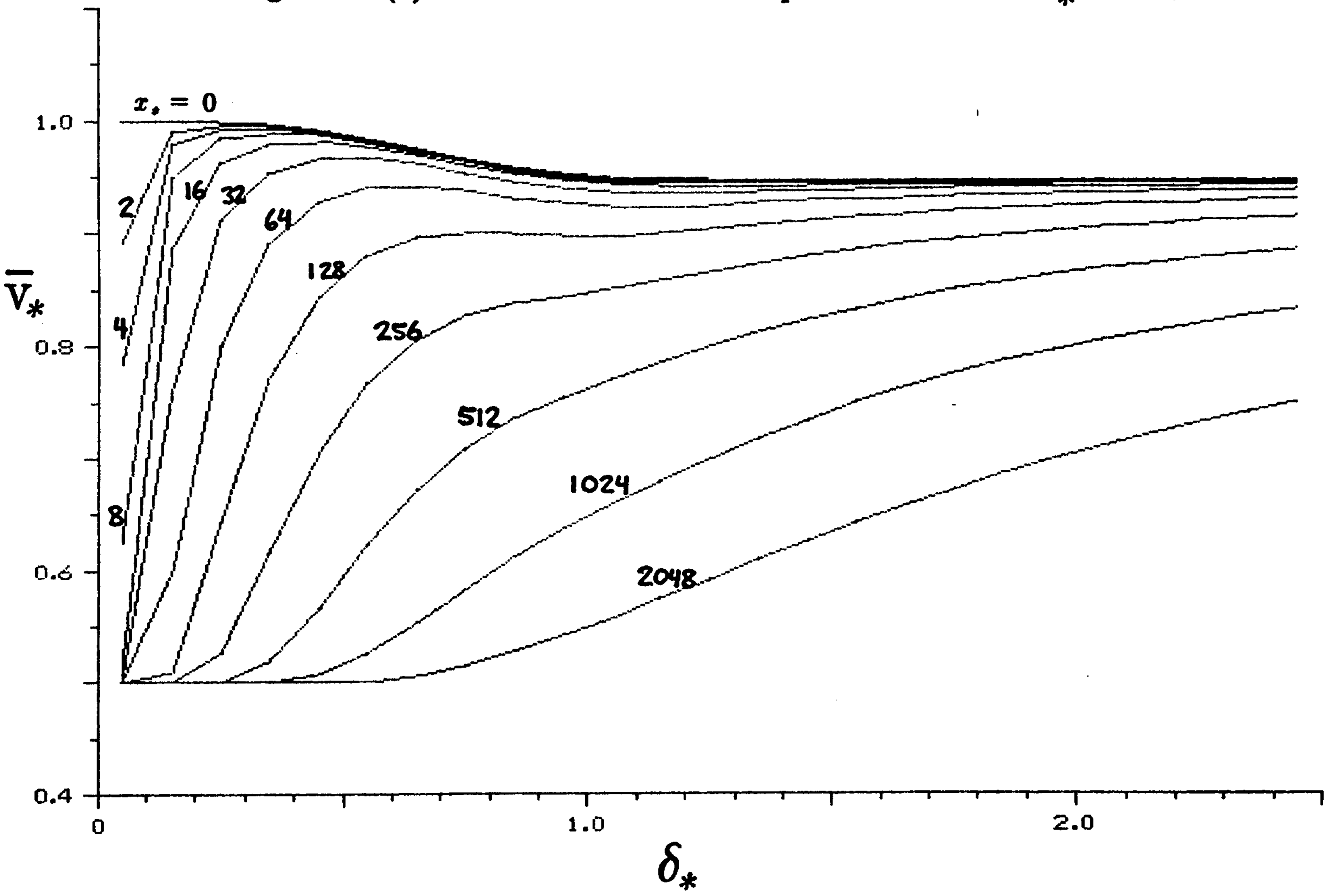


Figure 3.2 (d) Downstream Variance of Droplet Velocities for $\mathcal{T}_* = 0.375$

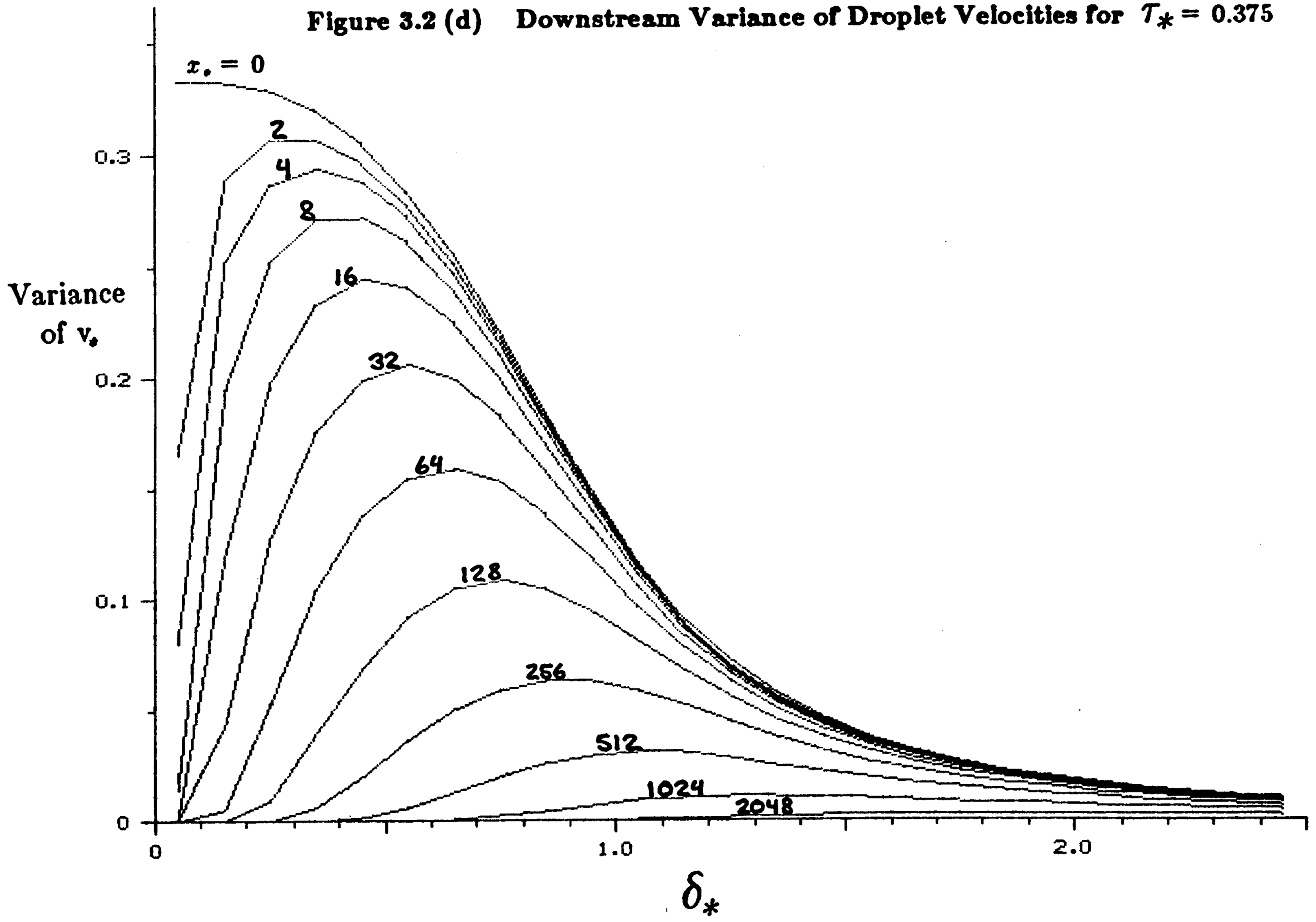


Figure 3.2 (e) Downstream Mean Droplet Velocities for $\tau_* = 0.400$

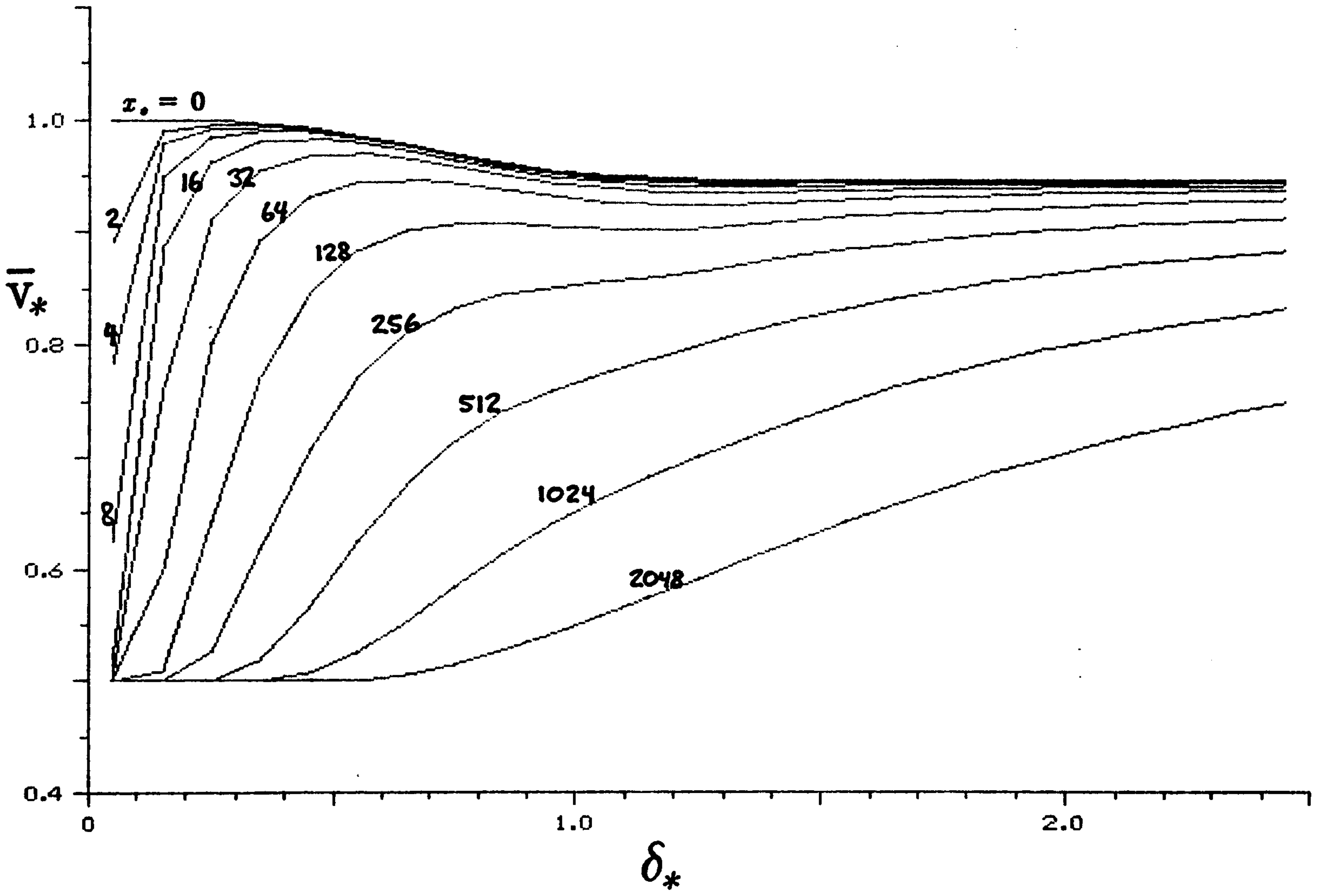
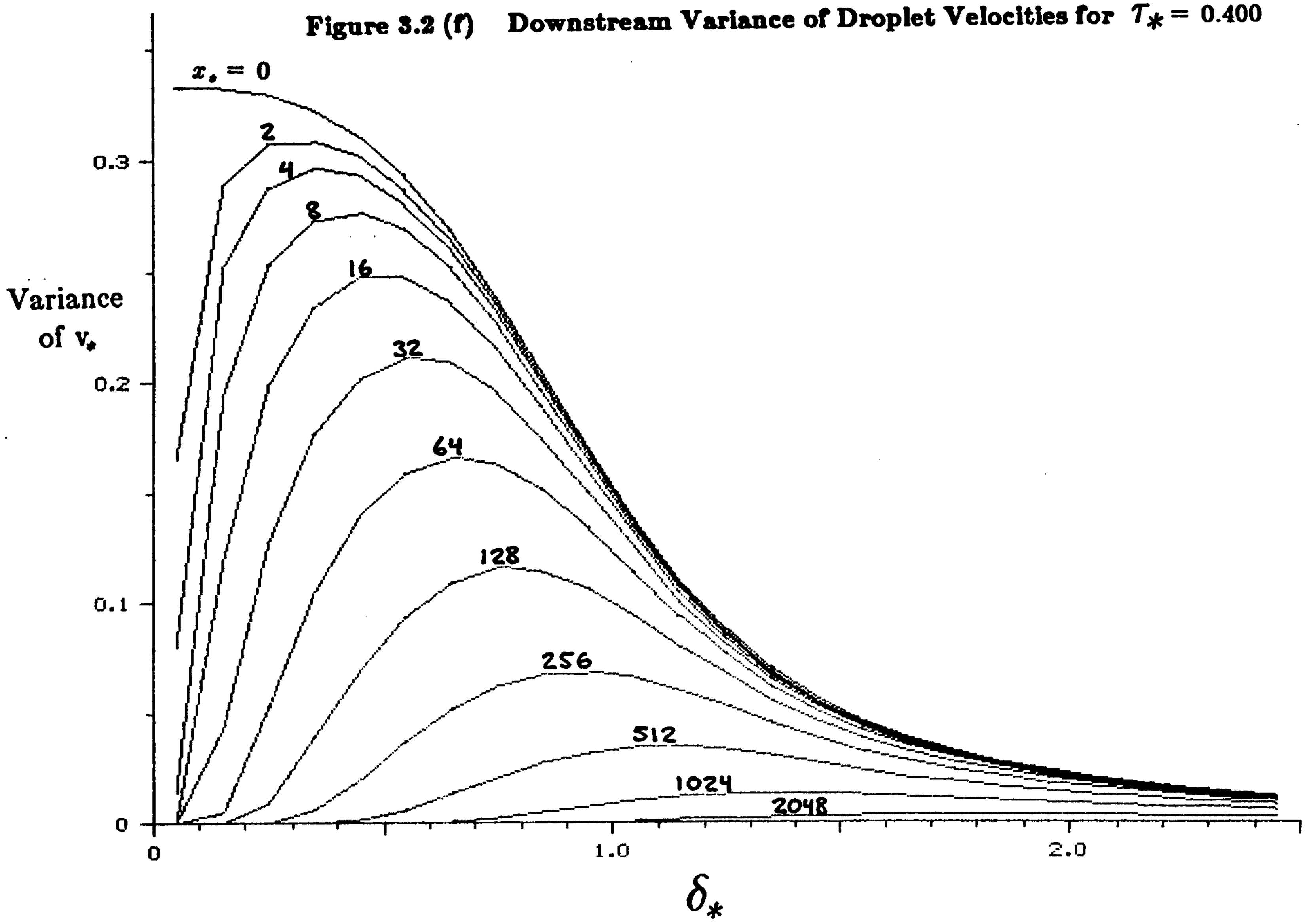


Figure 3.2 (f) Downstream Variance of Droplet Velocities for $\tau_* = 0.400$



dependence of the acceleration of a droplet in the air field on the square of the relative velocity. Those droplets with velocities farther away from the mean velocity move toward the mean at a greater rate than those with velocities closer to the mean. This results in a narrowing of the distribution which is reflected as a decrease in the variance.

For example, from Figure 3.2 (b), the variance associated with the 100 micron drop size is reduced by more than seventy percent by the time the spray reaches the downstream point where $x_* = 512$. This corresponds to an actual distance of 5.12 centimetres downstream, only a small part of the overall length of a typical spray. This result is typical of a wide range of drop sizes in each of the three cases.

This result suggests that, whatever the initial droplet velocity distribution, a substantial part of the identity of that initial distribution is lost over a distance which is short in comparison with the length of a typical spray. This would mean that for efficient atomizers, which produce small droplets, the velocity distribution in the flame zone would be much more strongly dependent on the nature of the gas flow field in the vicinity of the spray than on the characteristics of the initial velocity distribution.

This also calls into question the momentum source term used to force a distributed velocity. If the velocity distribution is affected this strongly it may be unreasonable to apply this source term from the "knowledge" that velocity is a distributed variable, even right at the base of the spray. The question of a distributed velocity versus a single valued velocity requires further work.

Conclusion

It is possible to predict the droplet size and velocity distribution in a spray from first principles. The Jaynes - Tribus maximum entropy formalism predicts the most probable distribution under the constraints of constant liquid mass, momentum, kinetic energy and surface energy.

The projection of this distribution on the size dimension is the ME distribution. It agrees quite well with with corresponding empirical distributions, particularly the Rosin - Rammler distribution.

The velocity distribution found is Gaussian in form. When the system is propagated downstream using a simple drag model, the variance of the velocity distribution drops substantially over distances which are short in relation to typical spray lengths. On this basis one may conclude that droplet velocity distributions in sprays are strongly dependent on the surrounding gas stream, and that the downstream development of the distribution is relatively insensitive to the initial velocity distribution.

The use of the maximum entropy formalism provides a framework for studying more complex distributions, since such distributions reflect processes which are described by constraints which are more complex than the independent conservation laws which produce the ME distribution. This added complexity could be introduced through the interactions between the different energy modes, which have been neglected in this work, or through further constraints such as the additional momentum constraint arising from a two dimensional spatial model.

Recommended Future Work

The close fit with the Rosin - Rammler distribution underscores the potential of this approach in atomization modeling. Further work is required in this area to fully explore that potential.

The weak link in the current work concerns the development of the velocity distribution. In particular, the application of the momentum source term is completely arbitrary. This area requires clarification and a better understanding. Perhaps the consideration of an undulating sheet in a two dimensional velocity field would produce a distributed velocity from a more fundamental basis. This approach should be explored.

The current work considers only a sheet breaking up through undulations and ligament formation. Many practical sprays are formed from direct atomization of a highly turbulent jet. It may be possible to model the breakup of a turbulent jet based on the creation of surface energy from turbulent kinetic energy at the liquid/gas interface. Further work in this direction would require some complex modeling of the behaviour of a free liquid surface in a gas.

References

[1] Rayleigh, Lord, On the Instability of Jets, *Proc. London Math. Soc.* Vol. 10, pp. 4-13, November 1878.

[2] Weber, C., *Z. Angew. Math. Mech.*, Vol. 11, p. 138, 1931.

[3] Levich, V. G., *Physicochemical Hydrodynamics,* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1962.

[4] Mayer, E., Theory of Liquid Atomization in High Velocity Gas Streams, *ARS J.,* Vol. 31, No. 12, p. 1783, December 1961.

[5] Dombrowski, N., and Johns, W. R., The Aerodynamic Instability and Disintegration of Viscous Liquid Sheets, *Chem. Eng. Sci.,* Vol. 18, pp. 203-214, 1963.

[6] Hinze, J. O., *Applied Sci. Research,* A. 1, p. 273, 1948.

[7] Rice, E. J., Section 2.2.3 Mechanisms of Atomization, *Liquid Propellant Rocket Combustion Instability,* NASA SP-194, pp. 49-55, 1972.

[8] Kelly, A. J., Electrostatic Metallic Spray Theory, *J. Applied Physics,* Vol. 47, No. 12, pp. 5264-5271, December 1976.

[9] Shannon, Claude, *Bell System Tech. J.,* Vol. 27, 1948. Reprinted in Shannon, C.E. and Weaver, W., *The Mathematical Theory of Communication,* University of Illinois Press, Urbana, 1949.

[10] Jaynes, E.T., Information Theory and Statistical Mechanics, I, II, *Physical Review,* No. 106, pp. 620-630, No. 108, pp. 171-190, 1957. Reprinted in *E.T. Jaynes: Papers on Probability, Statistics and Statistical Physics,* (Ed. R.D. Rosenkrantz), D. Reidel Publishing Company, Dordrecht, Holland, 1983.

[11] Tribus, Myron, *Thermostatics and Thermodynamics,* D. Van Nostrand Company, Inc., Princeton, New Jersey, 1961.

[12] Tribus, Myron, *Rational Descriptions Decisions and Designs,* Pergamon Press, New York, 1969.

[13] Agmon, N., Alhassid, Y. and Levine, R. D., An Algorithm for Determining the Lagrange Parameters in the Maximal Entropy Formalism, *The Maximum Entropy Formalism* (Ed. R. D. Levine and Myron Tribus), pp. 207-209, The MIT Press, Cambridge, Massachusetts, 1979.

[14] Rosin, P., and Rammler, E., *J. Inst. Fuel.,* Vol. 7, p. 29, 1933.

[15] Nukiyama, S., and Tanasawa, Y., *Trans. Soc. Mech. Engrs. (Japan),* Vol. 4, No. 14, p. 86, 1938; Vol. 5, No. 15, p. 138, 1939; Vol. 6, No. 18, p. 63, No. 22, p. II-7, No. 23, p. II-8, 1940.

[16] Bevans, R. S., Mathematical Expressions for Drop Size Distributions in Sprays, Conference on Fuel Sprays, University of Michigan, March 1949.

[17] Mugele, R. A., and Evans, H. D., Droplet Size Distribution in Sprays *Industrial and Engineering Chemistry* Vol. 43, No. 6, pp. 1317-1324, June 1951.

[18] White, F. M., *Viscous Fluid Flow,* McGraw Hill Book Company, New York, 1974.

Appendix A

Detailed Numerical Results

Three sets of detailed numerical results from the program are given on the following pages. These results are presented graphically in Chapters One and Two. (Figures 2.3 (a-f) and 3.2 (a-f))

The calculations were made using the following parameter values:

Air Velocity	5.0	m/s
Sheet Velocity	10.0	m/s
Kinematic Viscosity of Air	1.6×10^{-5}	m^2/s
Mass Mean Diameter	1.0×10^{-4}	m
Liquid Density	860.0	kg/m^3
Air Density	1.2	kg/m^3

```
Solution of Entropy Maximization Equations

Drop Size Distributions

Constraints:
0 - normalization
1 - surface energy   d**2
2 - mass             d**3
3 - momentum         d**3 x v
4 - kinetic energy   d**3 x v**2

Acceleration = 1.00                      I = 24          DSMAX = 2.5000
TSTAR        = 0.3500                    J = 25          VSMAX = 2.0000

lambda 0 to 4 =


4.1410E0        -1.3187E1       1.2237E1        -8.3608E0       4.4115E0
```

Alpha = 6.9167E0 Beta = 1.5860E2 Gamma = 3.2012E0
Kappa = 1.7523E0 a = 1.0000E0 dsm = 2.1890E0

dstar	Max Ent	R.R.	N.T.	Log Prob	Upper Limit
0.05	3.2838E-2	3.9102E-4	3.7521E-3	3.4965E-37	2.4776E-14
0.10	3.5943E-2	2.9527E-3	1.5008E-2	1.3680E-21	4.9684E-9
0.15	4.1414E-2	9.6348E-3	3.3769E-2	1.8817E-14	1.7356E-6
0.20	4.9864E-2	2.2297E-2	6.0034E-2	2.8306E-10	6.2440E-5
0.25	6.2288E-2	4.2747E-2	9.3803E-2	1.5302E-7	7.2382E-4
0.30	8.0145E-2	7.2750E-2	1.3507E-1	1.2265E-5	4.3174E-3
0.35	1.0547E-1	1.1403E-1	1.8385E-1	2.9347E-4	1.6734E-2
0.40	1.4100E-1	1.6827E-1	2.4013E-1	3.0977E-3	4.8071E-2
0.45	1.9022E-1	2.3706E-1	3.0392E-1	1.8284E-2	1.1090E-1
0.50	2.5730E-1	3.2187E-1	3.7521E-1	7.0333E-2	2.1654E-1
0.55	3.4685E-1	4.2387E-1	4.5401E-1	1.9557E-1	3.7076E-1
0.60	4.6328E-1	5.4379E-1	5.4030E-1	4.2283E-1	5.7080E-1
0.65	6.0979E-1	6.8158E-1	6.3411E-1	7.4937E-1	8.0460E-1
0.70	7.8689E-1	8.3595E-1	7.3542E-1	1.1323E0	1.0524E0
0.75	9.9056E-1	1.0037E0	8.4423E-1	1.5028E0	1.2901E0
0.80	1.2105E0	1.1792E0	9.6055E-1	1.7929E0	1.4935E0
0.85	1.4291E0	1.3532E0	1.0843E0	1.9579E0	1.6420E0
0.90	1.6218E0	1.5123E0	1.2157E0	1.9856E0	1.7216E0
0.95	1.7600E0	1.6393E0	1.3545E0	1.8918E0	1.7267E0
1.00	1.8166E0	1.7136E0	1.5008E0	1.7096E0	1.6600E0
1.05	1.7730E0	1.7150E0	1.6547E0	1.4768E0	1.5316E0
1.10	1.6267E0	1.6285E0	1.8160E0	1.2274E0	1.3569E0
1.15	1.3943E0	1.4507E0	1.9848E0	9.8679E-1	1.1538E0
1.20	1.1097E0	1.1960E0	2.1606E0	7.7091E-1	9.4083E-1
1.25	8.1487E-1	8.9790E-1	1.9829E0	5.8751E-1	7.3428E-1
1.30	5.4856E-1	6.0206E-1	5.7870E-37	4.3821E-1	5.4709E-1
1.35	3.3640E-1	3.5250E-1	0.0000E0	3.2080E-1	3.8772E-1
1.40	1.8674E-1	1.7551E-1	0.0000E0	2.3105E-1	2.6013E-1
1.45	9.3243E-2	7.2072E-2	0.0000E0	1.6407E-1	1.6420E-1
1.50	4.1612E-2	2.3562E-2	0.0000E0	1.1507E-1	9.6747E-2
1.55	1.6493E-2	5.8896E-3	0.0000E0	7.9846E-2	5.2657E-2
1.60	5.7699E-3	1.0748E-3	0.0000E0	5.4884E-2	2.6122E-2
1.65	1.7702E-3	1.3588E-4	0.0000E0	3.7418E-2	1.1604E-2
1.70	4.7333E-4	1.1217E-5	0.0000E0	2.5330E-2	4.5071E-3
1.75	1.0961E-4	5.6557E-7	0.0000E0	1.7041E-2	1.4813E-3
1.80	2.1845E-5	1.6158E-8	0.0000E0	1.1404E-2	3.9333E-4
1.85	3.7235E-6	2.4054E-10	0.0000E0	7.5965E-3	7.8814E-5
1.90	5.3943E-7	1.6994E-12	0.0000E0	5.0404E-3	1.0728E-5
1.95	6.6002E-8	5.1349E-15	0.0000E0	3.3331E-3	8.3427E-7
2.00	6.7783E-9	5.9134E-18	0.0000E0	2.1979E-3	2.7138E-8
2.05	5.8063E-10	2.2848E-21	0.0000E0	1.4458E-3	1.9455E-10
2.10	4.1227E-11	2.5736E-25	0.0000E0	9.4931E-4	5.9553E-14
2.15	2.4113E-12	7.2394E-30	0.0000E0	6.2230E-4	9.6505E-22
2.20	1.1545E-13	4.2913E-35	0.0000E0	4.0742E-4	0.0000E0
2.25	4.4968E-15	4.4507E-41	0.0000E0	2.6648E-4	0.0000E0
2.30	1.4160E-16	6.5900E-48	0.0000E0	1.7418E-4	0.0000E0
2.35	3.5824E-18	1.1159E-55	0.0000E0	1.1380E-4	0.0000E0
2.40	7.2365E-20	1.6973E-64	0.0000E0	7.4337E-5	0.0000E0
2.45	1.1598E-21	1.7833E-74	0.0000E0	4.8556E-5	0.0000E0
2.50	1.4658E-23	9.7356E-86	0.0000E0	3.1721E-5	0.0000E0
Total	1.9982E1	2.0000E1	2.0367E1	1.9999E1	2.0000E1

```
Initial Conditions      delta, mean v* and variance.

  0.0500      1.0000      3.3328E-1
  0.1500      0.9995      3.3201E-1
  0.2500      0.9976      3.2724E-1
  0.3500      0.9937      3.1680E-1
  0.4500      0.9874      2.9894E-1
  0.5500      0.9790      2.7278E-1
  0.6500      0.9696      2.3896E-1
  0.7500      0.9609      2.0018E-1
  0.8500      0.9543      1.6078E-1
  0.9500      0.9503      1.2522E-1
  1.0500      0.9484      9.6237E-2
  1.1500      0.9478      7.4204E-2
  1.2500      0.9476      5.7982E-2
  1.3500      0.9476      4.6060E-2
  1.4500      0.9476      3.7176E-2
  1.5500      0.9476      3.0435E-2
  1.6500      0.9476      2.5230E-2
  1.7500      0.9476      2.1147E-2
  1.8500      0.9476      1.7900E-2
  1.9500      0.9476      1.5285E-2
  2.0500      0.9476      1.3155E-2
  2.1500      0.9476      1.1404E-2
  2.2500      0.9476      9.9501E-3
  2.3500      0.9476      8.7331E-3
  2.4500      0.9476      7.7068E-3

xstar =   2.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500      0.8906      1.6524E-1      1.0000      0.0000      2.0000
  0.1500      0.9901      2.8856E-1      1.0000      0.0000      2.0000
  0.2500      0.9951      3.0567E-1      1.0000      0.0000      2.0000
  0.3500      0.9926      3.0369E-1      1.0000      0.0000      2.0000
  0.4500      0.9867      2.9033E-1      1.0000      0.0000      2.0000
  0.5500      0.9783      2.6702E-1      1.0000      0.0000      2.0000
  0.6500      0.9689      2.3519E-1      1.0000      0.0000      2.0000
  0.7500      0.9601      1.9780E-1      1.0000      0.0000      2.0000
  0.8500      0.9534      1.5935E-1      1.0000      0.0000      2.0000
  0.9500      0.9494      1.2440E-1      1.0000      0.0000      2.0000
  1.0500      0.9468      9.4657E-2      0.9987      0.0178      1.8791
  1.1500      0.9468      7.2239E-2      0.9976      0.1306      1.7650
  1.2500      0.9469      5.6303E-2      0.9973      0.2253      1.6700
  1.3500      0.9470      4.4714E-2      0.9973      0.3038      1.5915
  1.4500      0.9471      3.6096E-2      0.9973      0.3692      1.5260
  1.5500      0.9471      2.9557E-2      0.9973      0.4242      1.4710
  1.6500      0.9472      2.4507E-2      0.9973      0.4711      1.4241
  1.7500      0.9472      2.0544E-2      0.9973      0.5113      1.3839
  1.8500      0.9472      1.7392E-2      0.9973      0.5462      1.3490
  1.9500      0.9472      1.4853E-2      0.9973      0.5767      1.3185
  2.0500      0.9473      1.2785E-2      0.9973      0.6035      1.2917
  2.1500      0.9473      1.1084E-2      0.9973      0.6272      1.2680
  2.2500      0.9473      9.6717E-3      0.9973      0.6483      1.2469
  2.3500      0.9473      8.4895E-3      0.9973      0.6672      1.2280
  2.4500      0.9473      7.4923E-3      0.9973      0.6842      1.2110
```

```
xstar =   4.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.7840     8.0009E-2    1.0000     0.0000     2.0000
  0.1500     0.9783     2.5220E-1    1.0000     0.0000     2.0000
  0.2500     0.9918     2.8605E-1    1.0000     0.0000     2.0000
  0.3500     0.9912     2.9136E-1    1.0000     0.0000     2.0000
  0.4500     0.9858     2.8209E-1    1.0000     0.0000     2.0000
  0.5500     0.9776     2.6145E-1    1.0000     0.0000     2.0000
  0.6500     0.9681     2.3150E-1    1.0000     0.0000     2.0000
  0.7500     0.9592     1.9547E-1    1.0000     0.0000     2.0000
  0.8500     0.9525     1.5795E-1    1.0000     0.0000     2.0000
  0.9500     0.9485     1.2358E-1    1.0000     0.0000     2.0000
  1.0500     0.9459     9.4029E-2    0.9998     0.0238     1.8698
  1.1500     0.9461     7.1759E-2    0.9996     0.1405     1.7532
  1.2500     0.9462     5.5943E-2    0.9996     0.2351     1.6588
  1.3500     0.9464     4.4443E-2    0.9996     0.3126     1.5814
  1.4500     0.9465     3.5887E-2    0.9996     0.3771     1.5170
  1.5500     0.9466     2.9393E-2    0.9996     0.4313     1.4629
  1.6500     0.9467     2.4376E-2    0.9996     0.4775     1.4168
  1.7500     0.9468     2.0438E-2    0.9996     0.5172     1.3772
  1.8500     0.9468     1.7304E-2    0.9996     0.5516     1.3429
  1.9500     0.9469     1.4780E-2    0.9996     0.5816     1.3129
  2.0500     0.9469     1.2724E-2    0.9996     0.6081     1.2865
  2.1500     0.9470     1.1032E-2    0.9996     0.6315     1.2631
  2.2500     0.9470     9.6274E-3    0.9996     0.6523     1.2424
  2.3500     0.9471     8.4513E-3    0.9996     0.6709     1.2237
  2.4500     0.9471     7.4592E-3    0.9996     0.6877     1.2070

xstar =   8.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.6253     1.3769E-2    0.9974     0.1347     1.2533
  0.1500     0.9504     1.9500E-1    1.0000     0.0000     2.0000
  0.2500     0.9834     2.5176E-1    1.0000     0.0000     2.0000
  0.3500     0.9876     2.6880E-1    1.0000     0.0000     2.0000
  0.4500     0.9837     2.6663E-1    1.0000     0.0000     2.0000
  0.5500     0.9759     2.5083E-1    1.0000     0.0000     2.0000
  0.6500     0.9665     2.2439E-1    1.0000     0.0000     2.0000
  0.7500     0.9575     1.9092E-1    1.0000     0.0000     2.0000
  0.8500     0.9508     1.5519E-1    1.0000     0.0000     2.0000
  0.9500     0.9468     1.2196E-1    1.0000     0.0000     1.9989
  1.0500     0.9442     9.3014E-2    0.9999     0.0277     1.8625
  1.1500     0.9445     7.1151E-2    0.9999     0.1434     1.7472
  1.2500     0.9448     5.5533E-2    0.9999     0.2373     1.6537
  1.3500     0.9451     4.4151E-2    0.9999     0.3144     1.5771
  1.4500     0.9454     3.5673E-2    1.0000     0.3785     1.5134
  1.5500     0.9456     2.9232E-2    1.0000     0.4325     1.4597
  1.6500     0.9458     2.4252E-2    1.0000     0.4785     1.4140
  1.7500     0.9459     2.0342E-2    1.0000     0.5180     1.3747
  1.8500     0.9460     1.7229E-2    1.0000     0.5523     1.3406
  1.9500     0.9462     1.4719E-2    1.0000     0.5822     1.3108
  2.0500     0.9463     1.2675E-2    1.0000     0.6086     1.2847
  2.1500     0.9464     1.0991E-2    1.0000     0.6319     1.2615
  2.2500     0.9464     9.5941E-3    1.0000     0.6526     1.2408
  2.3500     0.9465     8.4235E-3    1.0000     0.6712     1.2223
  2.4500     0.9466     7.4358E-3    1.0000     0.6880     1.2057
```

```
xstar =  16.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

 0.0500     0.5144     1.6428E-4     0.9973     0.4563     0.5953
 0.1500     0.8876     1.1974E-1     1.0000     0.0000     2.0000
 0.2500     0.9620     1.9809E-1     1.0000     0.0000     2.0000
 0.3500     0.9778     2.3063E-1     1.0000     0.0000     2.0000
 0.4500     0.9781     2.3924E-1     1.0000     0.0000     2.0000
 0.5500     0.9718     2.3142E-1     1.0000     0.0000     2.0000
 0.6500     0.9629     2.1110E-1     1.0000     0.0000     2.0000
 0.7500     0.9540     1.8228E-1     1.0000     0.0000     2.0000
 0.8500     0.9472     1.4987E-1     1.0000     0.0000     2.0000
 0.9500     0.9431     1.1857E-1     0.9999     0.0000     1.9807
 1.0500     0.9408     9.1065E-2     0.9999     0.0340     1.8494
 1.1500     0.9414     7.0000E-2     0.9999     0.1468     1.7375
 1.2500     0.9421     5.4763E-2     0.9999     0.2395     1.6460
 1.3500     0.9426     4.3609E-2     1.0000     0.3158     1.5707
 1.4500     0.9431     3.5277E-2     1.0000     0.3794     1.5080
 1.5500     0.9436     2.8936E-2     1.0000     0.4331     1.4550
 1.6500     0.9439     2.4027E-2     1.0000     0.4788     1.4099
 1.7500     0.9442     2.0167E-2     1.0000     0.5181     1.3711
 1.8500     0.9445     1.7091E-2     1.0000     0.5523     1.3375
 1.9500     0.9447     1.4610E-2     1.0000     0.5821     1.3080
 2.0500     0.9449     1.2587E-2     1.0000     0.6084     1.2821
 2.1500     0.9451     1.0920E-2     1.0000     0.6317     1.2592
 2.2500     0.9453     9.5353E-3     1.0000     0.6524     1.2387
 2.3500     0.9454     8.3749E-3     1.0000     0.6709     1.2204
 2.4500     0.9455     7.3952E-3     1.0000     0.6876     1.2040

xstar =  32.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

 0.0500     0.5001      0.0000E0     0.0000     0.0000     0.0000
 0.1500     0.7622     4.3291E-2     0.9986     0.1115     1.4413
 0.2500     0.9104     1.2758E-1     1.0000     0.0000     2.0000
 0.3500     0.9522     1.7395E-1     1.0000     0.0000     2.0000
 0.4500     0.9632     1.9542E-1     1.0000     0.0000     2.0000
 0.5500     0.9616     1.9870E-1     1.0000     0.0000     2.0000
 0.6500     0.9546     1.8778E-1     1.0000     0.0000     2.0000
 0.7500     0.9465     1.6660E-1     1.0000     0.0000     2.0000
 0.8500     0.9400     1.3997E-1     1.0000     0.0000     2.0000
 0.9500     0.9356     1.1213E-1     0.9999     0.0000     1.9446
 1.0500     0.9341     8.7340E-2     0.9999     0.0460     1.8238
 1.1500     0.9352     6.7755E-2     0.9999     0.1535     1.7185
 1.2500     0.9366     5.3256E-2     0.9999     0.2437     1.6308
 1.3500     0.9377     4.2543E-2     1.0000     0.3186     1.5581
 1.4500     0.9387     3.4499E-2     1.0000     0.3813     1.4972
 1.5500     0.9395     2.8353E-2     1.0000     0.4342     1.4458
 1.6500     0.9402     2.3582E-2     1.0000     0.4795     1.4019
 1.7500     0.9409     1.9822E-2     1.0000     0.5184     1.3641
 1.8500     0.9414     1.6820E-2     1.0000     0.5523     1.3312
 1.9500     0.9418     1.4394E-2     1.0000     0.5819     1.3025
 2.0500     0.9423     1.2413E-2     1.0000     0.6081     1.2771
 2.1500     0.9426     1.0778E-2     1.0000     0.6312     1.2546
 2.2500     0.9429     9.4191E-3     1.0000     0.6518     1.2346
 2.3500     0.9432     8.2785E-3     1.0000     0.6703     1.2166
 2.4500     0.9435     7.3147E-3     1.0000     0.6870     1.2005
```

```
xstar =  64.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.5000      0.0000E0     0.0000     0.0000     0.0000
  0.1500     0.5999     5.2559E-3     0.9986     0.3735     0.8398
  0.2500     0.8004     5.3275E-2     0.9993     0.0959     1.5176
  0.3500     0.8900     1.0327E-1     0.9997     0.0000     1.8741
  0.4500     0.9257     1.3566E-1     1.0000     0.0000     2.0000
  0.5500     0.9361     1.5030E-1     1.0000     0.0000     2.0000
  0.6500     0.9352     1.5097E-1     1.0000     0.0000     2.0000
  0.7500     0.9301     1.4047E-1     1.0000     0.0000     2.0000
  0.8500     0.9248     1.2239E-1     0.9999     0.0000     1.9800
  0.9500     0.9208     1.0052E-1     0.9999     0.0000     1.8760
  1.0500     0.9207     8.0483E-2     0.9999     0.0683     1.7748
  1.1500     0.9232     6.3486E-2     0.9999     0.1665     1.6813
  1.2500     0.9258     5.0362E-2     0.9999     0.2520     1.6008
  1.3500     0.9280     4.0486E-2     1.0000     0.3240     1.5332
  1.4500     0.9300     3.2990E-2     1.0000     0.3848     1.4762
  1.5500     0.9316     2.7221E-2     1.0000     0.4365     1.4277
  1.6500     0.9330     2.2715E-2     1.0000     0.4808     1.3861
  1.7500     0.9342     1.9149E-2     1.0000     0.5190     1.3502
  1.8500     0.9352     1.6290E-2     1.0000     0.5523     1.3189
  1.9500     0.9362     1.3972E-2     1.0000     0.5816     1.2915
  2.0500     0.9370     1.2072E-2     1.0000     0.6074     1.2672
  2.1500     0.9377     1.0501E-2     1.0000     0.6303     1.2457
  2.2500     0.9383     9.1907E-3     1.0000     0.6508     1.2264
  2.3500     0.9389     8.0891E-3     1.0000     0.6691     1.2092
  2.4500     0.9394     7.1564E-3     1.0000     0.6857     1.1936

xstar = 128.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.5000      0.0000E0     0.0000     0.0000     0.0000
  0.1500     0.5105     4.2544E-5     0.9985     0.4902     0.5325
  0.2500     0.6431     1.0007E-2     0.9993     0.3383     0.9549
  0.3500     0.7696     3.8928E-2     0.9996     0.1723     1.3734
  0.4500     0.8404     6.7772E-2     0.9997     0.0540     1.6320
  0.5500     0.8751     8.9115E-2     0.9999     0.0000     1.7797
  0.6500     0.8893     9.9960E-2     0.9999     0.0000     1.8453
  0.7500     0.8926     1.0072E-1     0.9999     0.0000     1.8508
  0.8500     0.8920     9.3496E-2     0.9999     0.0000     1.8141
  0.9500     0.8922     8.1371E-2     0.9999     0.0346     1.7516
  1.0500     0.8949     6.8644E-2     0.9999     0.1078     1.6836
  1.1500     0.8999     5.5753E-2     0.9999     0.1908     1.6104
  1.2500     0.9049     4.5027E-2     0.9999     0.2678     1.5433
  1.3500     0.9093     3.6656E-2     1.0000     0.3346     1.4851
  1.4500     0.9130     3.0161E-2     1.0000     0.3918     1.4352
  1.5500     0.9161     2.5084E-2     1.0000     0.4409     1.3924
  1.6500     0.9189     2.1074E-2     1.0000     0.4833     1.3553
  1.7500     0.9212     1.7869E-2     1.0000     0.5201     1.3231
  1.8500     0.9232     1.5278E-2     1.0000     0.5524     1.2948
  1.9500     0.9250     1.3162E-2     1.0000     0.5808     1.2699
  2.0500     0.9266     1.1417E-2     1.0000     0.6060     1.2477
  2.1500     0.9280     9.9666E-3     1.0000     0.6285     1.2280
  2.2500     0.9292     8.7501E-3     1.0000     0.6486     1.2103
  2.3500     0.9303     7.7229E-3     1.0000     0.6667     1.1944
  2.4500     0.9313     6.8497E-3     1.0000     0.6831     1.1800
```

```
xstar = 256.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

   0.0500     0.5000      0.0000E0     0.0000     0.0000     0.0000
   0.1500     0.5001      0.0000E0     0.0000     0.0000     0.0000
   0.2500     0.5269     2.6253E-4     0.9993     0.4775     0.5776
   0.3500     0.6158     5.9151E-3     0.9996     0.3832     0.8517
   0.4500     0.7031     1.9832E-2     0.9997     0.2783     1.1314
   0.5500     0.7633     3.5466E-2     0.9998     0.1960     1.3337
   0.6500     0.7995     4.7955E-2     0.9998     0.1403     1.4613
   0.7500     0.8194     5.5418E-2     0.9999     0.1112     1.5298
   0.8500     0.8306     5.7666E-2     0.9999     0.1085     1.5545
   0.9500     0.8386     5.5488E-2     0.9999     0.1307     1.5481
   1.0500     0.8472     5.0363E-2     0.9999     0.1730     1.5227
   1.1500     0.8570     4.3019E-2     0.9999     0.2341     1.4811
   1.2500     0.8661     3.5958E-2     0.9999     0.2968     1.4366
   1.3500     0.8742     3.0016E-2     1.0000     0.3541     1.3953
   1.4500     0.8810     2.5186E-2     1.0000     0.4047     1.3583
   1.5500     0.8869     2.1286E-2     1.0000     0.4491     1.3256
   1.6500     0.8920     1.8128E-2     1.0000     0.4880     1.2968
   1.7500     0.8964     1.5553E-2     1.0000     0.5222     1.2713
   1.8500     0.9002     1.3435E-2     1.0000     0.5525     1.2486
   1.9500     0.9036     1.1678E-2     1.0000     0.5794     1.2284
   2.0500     0.9066     1.0211E-2     1.0000     0.6035     1.2103
   2.1500     0.9092     8.9761E-3     1.0000     0.6250     1.1940
   2.2500     0.9116     7.9301E-3     1.0000     0.6445     1.1792
   2.3500     0.9137     7.0386E-3     1.0000     0.6621     1.1658
   2.4500     0.9156     6.2745E-3     1.0000     0.6781     1.1537

xstar = 512.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

   0.0500     0.5000      0.0000E0     0.0000     0.0000     0.0000
   0.1500     0.5000      0.0000E0     0.0000     0.0000     0.0000
   0.2500     0.5010      0.0000E0     0.0000     0.0000     0.0000
   0.3500     0.5193     1.1246E-4     0.9996     0.4872     0.5518
   0.4500     0.5663     1.6425E-3     0.9997     0.4441     0.6897
   0.5500     0.6216     6.1591E-3     0.9998     0.3853     0.8594
   0.6500     0.6689     1.2677E-2     0.9998     0.3302     1.0092
   0.7500     0.7043     1.9158E-2     0.9999     0.2880     1.1219
   0.8500     0.7298     2.4136E-2     0.9999     0.2629     1.1981
   0.9500     0.7496     2.6909E-2     0.9999     0.2567     1.2437
   1.0500     0.7671     2.7411E-2     0.9999     0.2698     1.2655
   1.1500     0.7840     2.5593E-2     0.9999     0.3036     1.2654
   1.2500     0.7993     2.2849E-2     0.9999     0.3454     1.2540
   1.3500     0.8127     2.0052E-2     1.0000     0.3877     1.2386
   1.4500     0.8244     1.7507E-2     1.0000     0.4273     1.2223
   1.5500     0.8346     1.5289E-2     1.0000     0.4635     1.2064
   1.6500     0.8434     1.3387E-2     1.0000     0.4962     1.1913
   1.7500     0.8512     1.1764E-2     1.0000     0.5258     1.1772
   1.8500     0.8580     1.0375E-2     1.0000     0.5524     1.1642
   1.9500     0.8641     9.1849E-3     1.0000     0.5766     1.1521
   2.0500     0.8695     8.1605E-3     1.0000     0.5985     1.1410
   2.1500     0.8743     7.2762E-3     1.0000     0.6184     1.1307
   2.2500     0.8786     6.5103E-3     1.0000     0.6366     1.1211
   2.3500     0.8826     5.8444E-3     1.0000     0.6533     1.1123
   2.4500     0.8861     5.2636E-3     1.0000     0.6685     1.1041
```

```
xstar = 1024.00 delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

 0.0500     0.5000      0.0000E0     0.0000     0.0000     0.0000
 0.1500     0.5000      0.0000E0     0.0000     0.0000     0.0000
 0.2500     0.5000      0.0000E0     0.0000     0.0000     0.0000
 0.3500     0.5006      0.0000E0     0.0000     0.0000     0.0000
 0.4500     0.5074      0.0000E0     0.0000     0.0000     0.0000
 0.5500     0.5256      1.8508E-4    0.9998     0.4847     0.5669
 0.6500     0.5525      8.9895E-4    0.9999     0.4623     0.6431
 0.7500     0.5817      2.3738E-3    0.9999     0.4352     0.7287
 0.8500     0.6093      4.3953E-3    0.9999     0.4100     0.8091
 0.9500     0.6343      6.4874E-3    0.9999     0.3923     0.8769
 1.0500     0.6576      8.1784E-3    0.9999     0.3860     0.9299
 1.1500     0.6798      9.0084E-3    0.9999     0.3947     0.9654
 1.2500     0.7003      9.1371E-3    0.9999     0.4133     0.9879
 1.3500     0.7190      8.8631E-3    1.0000     0.4364     1.0022
 1.4500     0.7358      8.3903E-3    1.0000     0.4609     1.0113
 1.5500     0.7508      7.8364E-3    1.0000     0.4852     1.0170
 1.6500     0.7643      7.2637E-3    1.0000     0.5085     1.0205
 1.7500     0.7763      6.7032E-3    1.0000     0.5307     1.0224
 1.8500     0.7871      6.1687E-3    1.0000     0.5515     1.0232
 1.9500     0.7969      5.6671E-3    1.0000     0.5710     1.0232
 2.0500     0.8057      5.2020E-3    1.0000     0.5893     1.0225
 2.1500     0.8137      4.7741E-3    1.0000     0.6064     1.0214
 2.2500     0.8210      4.3827E-3    1.0000     0.6224     1.0199
 2.3500     0.8276      4.0260E-3    1.0000     0.6373     1.0183
 2.4500     0.8337      3.7017E-3    1.0000     0.6512     1.0165

xstar = 2048.00 delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

 0.0500     0.5000      0.0000E0     0.0000     0.0000     0.0000
 0.1500     0.5000      0.0000E0     0.0000     0.0000     0.0000
 0.2500     0.5000      0.0000E0     0.0000     0.0000     0.0000
 0.3500     0.5000      0.0000E0     0.0000     0.0000     0.0000
 0.4500     0.5001      0.0000E0     0.0000     0.0000     0.0000
 0.5500     0.5015      0.0000E0     0.0000     0.0000     0.0000
 0.6500     0.5061      0.0000E0     0.0000     0.0000     0.0000
 0.7500     0.5146      4.6473E-5    0.9999     0.4942     0.5352
 0.8500     0.5267      1.7117E-4    0.9999     0.4874     0.5662
 0.9500     0.5412      4.1923E-4    0.9999     0.4797     0.6028
 1.0500     0.5573      7.7692E-4    0.9999     0.4736     0.6412
 1.1500     0.5746      1.1544E-3    0.9999     0.4726     0.6769
 1.2500     0.5925      1.4840E-3    0.9999     0.4768     0.7084
 1.3500     0.6101      1.7432E-3    1.0000     0.4848     0.7357
 1.4500     0.6273      1.9329E-3    1.0000     0.4953     0.7595
 1.5500     0.6436      2.0622E-3    1.0000     0.5073     0.7801
 1.6500     0.6590      2.1414E-3    1.0000     0.5202     0.7981
 1.7500     0.6735      2.1793E-3    1.0000     0.5334     0.8138
 1.8500     0.6870      2.1833E-3    1.0000     0.5468     0.8274
 1.9500     0.6996      2.1603E-3    1.0000     0.5602     0.8393
 2.0500     0.7114      2.1168E-3    1.0000     0.5734     0.8497
 2.1500     0.7223      2.0583E-3    1.0000     0.5862     0.8587
 2.2500     0.7325      1.9894E-3    1.0000     0.5987     0.8666
 2.3500     0.7420      1.9137E-3    1.0000     0.6108     0.8735
 2.4500     0.7509      1.8339E-3    1.0000     0.6224     0.8796
```

```
Solution of Entropy Maximization Equations

Drop Size Distributions

Constraints:
0 - normalization
1 - surface energy   d**2
2 - mass             d**3
3 - momentum         d**3 x v
4 - kinetic energy   d**3 x v**2

Acceleration = 1.00                       I = 24          DSMAX = 2.5000
TSTAR        = 0.3750                     J = 25          VSMAX = 2.0000

lambda 0 to 4 =

2.0563E0        -5.2896E0       6.2975E0        -6.7083E0       3.5433E0
```

Alpha = 4.9348E0 Beta = 2.4510E0 Gamma = 2.0603E0
Kappa = 1.2325E0 a = 1.0000E0 dsm = 2.4512E0

dstar	Max Ent	R.R.	N.T.	Log Prob	Upper Limit
0.05	2.5910E-1	7.8198E-2	9.4364E-3	5.1878E-14	1.4628E-5
0.10	2.6858E-1	1.4949E-1	3.7601E-2	5.4092E-8	1.9161E-3
0.15	2.8401E-1	2.1838E-1	8.3933E-2	2.7057E-5	1.7621E-2
0.20	3.0541E-1	2.8575E-1	1.4729E-1	9.5440E-4	6.4309E-2
0.25	3.3292E-1	3.5197E-1	2.2590E-1	9.3299E-3	1.4968E-1
0.30	3.6671E-1	4.1721E-1	3.1731E-1	4.3913E-2	2.6909E-1
0.35	4.0693E-1	4.8151E-1	4.1841E-1	1.3054E-1	4.1054E-1
0.40	4.5358E-1	5.4478E-1	5.2558E-1	2.8496E-1	5.6005E-1
0.45	5.0641E-1	6.0684E-1	6.3478E-1	5.0034E-1	7.0510E-1
0.50	5.6485E-1	6.6734E-1	7.4176E-1	7.4924E-1	8.3605E-1
0.55	6.2791E-1	7.2581E-1	8.4225E-1	9.9537E-1	9.4649E-1
0.60	6.9414E-1	7.8158E-1	9.3221E-1	1.2060E0	1.0328E0
0.65	7.6164E-1	8.3380E-1	1.0080E0	1.3596E0	1.0939E0
0.70	8.2804E-1	8.8142E-1	1.0668E0	1.4472E0	1.1300E0
0.75	8.9066E-1	9.2319E-1	1.1064E0	1.4709E0	1.1429E0
0.80	9.4654E-1	9.5770E-1	1.1256E0	1.4397E0	1.1349E0
0.85	9.9262E-1	9.8336E-1	1.1243E0	1.3664E0	1.1088E0
0.90	1.0258E0	9.9855E-1	1.1030E0	1.2640E0	1.0676E0
0.95	1.0435E0	1.0017E0	1.0636E0	1.1446E0	1.0142E0
1.00	1.0431E0	9.9137E-1	1.0082E0	1.0181E0	9.5143E-1
1.05	1.0231E0	9.6654E-1	9.3986E-1	8.9219E-1	8.8189E-1
1.10	9.8276E-1	9.2668E-1	8.6175E-1	7.7195E-1	8.0798E-1
1.15	9.2253E-1	8.7201E-1	7.7728E-1	6.6079E-1	7.3181E-1
1.20	8.4448E-1	8.0363E-1	6.8976E-1	5.6051E-1	6.5524E-1
1.25	7.5209E-1	7.2358E-1	6.0224E-1	4.7178E-1	5.7983E-1
1.30	6.5010E-1	6.3486E-1	5.1738E-1	3.9451E-1	5.0691E-1
1.35	5.4408E-1	5.4124E-1	4.3735E-1	3.2805E-1	4.3755E-1
1.40	4.3979E-1	4.4696E-1	3.6375E-1	2.7150E-1	3.7260E-1
1.45	3.4250E-1	3.5634E-1	2.9768E-1	2.2380E-1	3.1269E-1
1.50	2.5635E-1	2.7329E-1	2.3967E-1	1.8386E-1	2.5828E-1
1.55	1.8396E-1	2.0083E-1	1.8984E-1	1.5062E-1	2.0962E-1
1.60	1.2625E-1	1.4084E-1	1.4793E-1	1.2310E-1	1.6684E-1
1.65	8.2668E-2	9.3839E-2	1.1339E-1	1.0041E-1	1.2990E-1
1.70	5.1518E-2	5.9122E-2	8.5485E-2	8.1770E-2	9.8626E-2
1.75	3.0482E-2	3.5048E-2	6.3384E-2	6.6506E-2	7.2738E-2
1.80	1.7082E-2	1.9445E-2	4.6215E-2	5.4035E-2	5.1856E-2
1.85	9.0457E-3	1.0041E-2	3.3134E-2	4.3870E-2	3.5511E-2
1.90	4.5150E-3	4.7979E-3	2.3356E-2	3.5596E-2	2.3169E-2
1.95	2.1191E-3	2.1078E-3	1.6185E-2	2.8872E-2	1.4247E-2
2.00	9.3310E-4	8.4590E-4	1.1024E-2	2.3413E-2	8.1357E-3
2.05	3.8450E-4	3.0796E-4	7.3806E-3	1.8985E-2	4.2258E-3
2.10	1.4792E-4	1.0098E-4	4.8557E-3	1.5396E-2	1.9373E-3
2.15	5.3007E-5	2.9595E-5	3.1390E-3	1.2487E-2	7.4889E-4
2.20	1.7649E-5	7.6913E-6	1.9937E-3	1.0131E-2	2.2665E-4
2.25	5.4474E-6	1.7577E-6	1.2439E-3	8.2221E-3	4.7150E-5
2.30	1.5548E-6	3.5018E-7	7.6235E-4	6.6759E-3	5.2097E-6
2.35	4.0946E-7	6.0268E-8	4.5885E-4	5.4232E-3	1.6467E-7
2.40	9.9247E-8	8.8757E-9	2.7120E-4	4.4081E-3	1.7228E-10
2.45	2.2088E-8	1.1075E-9	1.5739E-4	3.5853E-3	2.5160E-37
2.50	4.5034E-9	1.1589E-10	8.9672E-5	2.9180E-3	0.0000E0
Total	1.9869E1	1.9992E1	1.9999E1	1.9986E1	2.0000E1

```
Initial Conditions      delta, mean v* and variance.

  0.0500     1.0000      3.3329E-1
  0.1500     0.9996      3.3227E-1
  0.2500     0.9981      3.2843E-1
  0.3500     0.9948      3.2001E-1
  0.4500     0.9895      3.0548E-1
  0.5500     0.9821      2.8389E-1
  0.6500     0.9735      2.5526E-1
  0.7500     0.9647      2.2103E-1
  0.8500     0.9571      1.8416E-1
  0.9500     0.9518      1.4839E-1
  1.0500     0.9487      1.1698E-1
  1.1500     0.9473      9.1536E-2
  1.2500     0.9468      7.1989E-2
  1.3500     0.9466      5.7309E-2
  1.4500     0.9466      4.6280E-2
  1.5500     0.9466      3.7892E-2
  1.6500     0.9466      3.1412E-2
  1.7500     0.9466      2.6329E-2
  1.8500     0.9466      2.2286E-2
  1.9500     0.9466      1.9030E-2
  2.0500     0.9466      1.6379E-2
  2.1500     0.9466      1.4198E-2
  2.2500     0.9466      1.2388E-2
  2.3500     0.9466      1.0873E-2
  2.4500     0.9466      9.5952E-3

xstar =   2.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.8906      1.6525E-1    1.0000     0.0000     2.0000
  0.1500     0.9902      2.8878E-1    1.0000     0.0000     2.0000
  0.2500     0.9955      3.0675E-1    1.0000     0.0000     2.0000
  0.3500     0.9938      3.0671E-1    1.0000     0.0000     2.0000
  0.4500     0.9888      2.9660E-1    1.0000     0.0000     2.0000
  0.5500     0.9816      2.7780E-1    1.0000     0.0000     2.0000
  0.6500     0.9729      2.5110E-1    1.0000     0.0000     2.0000
  0.7500     0.9640      2.1827E-1    1.0000     0.0000     2.0000
  0.8500     0.9564      1.8240E-1    1.0000     0.0000     2.0000
  0.9500     0.9510      1.4732E-1    1.0000     0.0000     2.0000
  1.0500     0.9476      1.1601E-1    0.9997     0.0000     1.9748
  1.1500     0.9459      8.9812E-2    0.9983     0.0396     1.8549
  1.2500     0.9459      7.0070E-2    0.9975     0.1419     1.7517
  1.3500     0.9460      5.5665E-2    0.9973     0.2285     1.6648
  1.4500     0.9461      4.4940E-2    0.9973     0.3012     1.5920
  1.5500     0.9461      3.6800E-2    0.9973     0.3626     1.5306
  1.6500     0.9462      3.0512E-2    0.9973     0.4149     1.4783
  1.7500     0.9462      2.5578E-2    0.9973     0.4598     1.4334
  1.8500     0.9462      2.1654E-2    0.9973     0.4988     1.3945
  1.9500     0.9463      1.8492E-2    0.9973     0.5328     1.3605
  2.0500     0.9463      1.5918E-2    0.9973     0.5627     1.3306
  2.1500     0.9463      1.3800E-2    0.9973     0.5891     1.3041
  2.2500     0.9463      1.2041E-2    0.9973     0.6127     1.2805
  2.3500     0.9463      1.0569E-2    0.9973     0.6338     1.2594
  2.4500     0.9464      9.3281E-3    0.9973     0.6527     1.2405
```

```
xstar =    4.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.7840     8.0012E-2     1.0000     0.0000     2.0000
  0.1500     0.9784     2.5238E-1     1.0000     0.0000     2.0000
  0.2500     0.9923     2.8705E-1     1.0000     0.0000     2.0000
  0.3500     0.9924     2.9421E-1     1.0000     0.0000     2.0000
  0.4500     0.9881     2.8811E-1     1.0000     0.0000     2.0000
  0.5500     0.9809     2.7190E-1     1.0000     0.0000     2.0000
  0.6500     0.9722     2.4704E-1     1.0000     0.0000     2.0000
  0.7500     0.9633     2.1556E-1     1.0000     0.0000     2.0000
  0.8500     0.9556     1.8066E-1     1.0000     0.0000     2.0000
  0.9500     0.9502     1.4625E-1     1.0000     0.0000     2.0000
  1.0500     0.9467     1.1530E-1     0.9999     0.0000     1.9694
  1.1500     0.9451     8.9260E-2     0.9997     0.0468     1.8450
  1.2500     0.9453     6.9633E-2     0.9996     0.1518     1.7401
  1.3500     0.9454     5.5329E-2     0.9996     0.2382     1.6538
  1.4500     0.9455     4.4680E-2     0.9996     0.3101     1.5820
  1.5500     0.9456     3.6596E-2     0.9996     0.3706     1.5216
  1.6500     0.9457     3.0349E-2     0.9996     0.4221     1.4702
  1.7500     0.9458     2.5446E-2     0.9996     0.4664     1.4260
  1.8500     0.9458     2.1545E-2     0.9996     0.5048     1.3877
  1.9500     0.9459     1.8402E-2     0.9996     0.5383     1.3542
  2.0500     0.9460     1.5842E-2     0.9996     0.5678     1.3248
  2.1500     0.9460     1.3735E-2     0.9996     0.5939     1.2987
  2.2500     0.9460     1.1986E-2     0.9996     0.6171     1.2755
  2.3500     0.9461     1.0522E-2     0.9996     0.6379     1.2548
  2.4500     0.9461     9.2869E-3     0.9996     0.6566     1.2361

xstar =    8.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.6253     1.3769E-2     0.9974     0.1347     1.2533
  0.1500     0.9505     1.9514E-1     1.0000     0.0000     2.0000
  0.2500     0.9840     2.5260E-1     1.0000     0.0000     2.0000
  0.3500     0.9889     2.7136E-1     1.0000     0.0000     2.0000
  0.4500     0.9861     2.7219E-1     1.0000     0.0000     2.0000
  0.5500     0.9795     2.6066E-1     1.0000     0.0000     2.0000
  0.6500     0.9708     2.3921E-1     1.0000     0.0000     2.0000
  0.7500     0.9619     2.1028E-1     1.0000     0.0000     2.0000
  0.8500     0.9541     1.7726E-1     1.0000     0.0000     2.0000
  0.9500     0.9487     1.4416E-1     1.0000     0.0000     2.0000
  1.0500     0.9451     1.1393E-1     0.9999     0.0000     1.9616
  1.1500     0.9436     8.8445E-2     0.9999     0.0502     1.8386
  1.2500     0.9439     6.9111E-2     0.9999     0.1545     1.7347
  1.3500     0.9442     5.4963E-2     0.9999     0.2403     1.6492
  1.4500     0.9444     4.4414E-2     1.0000     0.3118     1.5781
  1.5500     0.9446     3.6396E-2     1.0000     0.3720     1.5182
  1.6500     0.9448     3.0195E-2     1.0000     0.4233     1.4672
  1.7500     0.9449     2.5327E-2     1.0000     0.4674     1.4233
  1.8500     0.9451     2.1450E-2     1.0000     0.5056     1.3853
  1.9500     0.9452     1.8326E-2     1.0000     0.5390     1.3521
  2.0500     0.9453     1.5780E-2     1.0000     0.5684     1.3228
  2.1500     0.9454     1.3685E-2     1.0000     0.5944     1.2969
  2.2500     0.9455     1.1945E-2     1.0000     0.6176     1.2739
  2.3500     0.9455     1.0487E-2     1.0000     0.6383     1.2533
  2.4500     0.9456     9.2578E-3     1.0000     0.6570     1.2347
```

```
xstar =  16.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.5144     1.6429E-4     0.9973     0.4563     0.5953
  0.1500     0.8877     1.1982E-1     1.0000     0.0000     2.0000
  0.2500     0.9625     1.9872E-1     1.0000     0.0000     2.0000
  0.3500     0.9792     2.3273E-1     1.0000     0.0000     2.0000
  0.4500     0.9807     2.4403E-1     1.0000     0.0000     2.0000
  0.5500     0.9757     2.4019E-1     1.0000     0.0000     2.0000
  0.6500     0.9677     2.2464E-1     1.0000     0.0000     2.0000
  0.7500     0.9588     2.0030E-1     1.0000     0.0000     2.0000
  0.8500     0.9510     1.7074E-1     1.0000     0.0000     2.0000
  0.9500     0.9455     1.4009E-1     1.0000     0.0000     2.0000
  1.0500     0.9418     1.1127E-1     0.9999     0.0000     1.9463
  1.1500     0.9406     8.6882E-2     0.9999     0.0551     1.8276
  1.2500     0.9411     6.8125E-2     0.9999     0.1574     1.7263
  1.3500     0.9417     5.4281E-2     1.0000     0.2423     1.6423
  1.4500     0.9422     4.3920E-2     1.0000     0.3131     1.5723
  1.5500     0.9426     3.6028E-2     1.0000     0.3729     1.5132
  1.6500     0.9429     2.9915E-2     1.0000     0.4239     1.4629
  1.7500     0.9432     2.5109E-2     1.0000     0.4678     1.4196
  1.8500     0.9435     2.1280E-2     1.0000     0.5058     1.3820
  1.9500     0.9437     1.8190E-2     1.0000     0.5391     1.3491
  2.0500     0.9439     1.5671E-2     1.0000     0.5684     1.3202
  2.1500     0.9441     1.3596E-2     1.0000     0.5943     1.2945
  2.2500     0.9443     1.1871E-2     1.0000     0.6174     1.2717
  2.3500     0.9444     1.0427E-2     1.0000     0.6381     1.2513
  2.4500     0.9446     9.2073E-3     1.0000     0.6568     1.2329

xstar =  32.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.5001      0.0000E0     0.0000     0.0000     0.0000
  0.1500     0.7623     4.3324E-2     0.9986     0.1113     1.4417
  0.2500     0.9109     1.2797E-1     1.0000     0.0000     2.0000
  0.3500     0.9537     1.7543E-1     1.0000     0.0000     2.0000
  0.4500     0.9660     1.9909E-1     1.0000     0.0000     2.0000
  0.5500     0.9659     2.0579E-1     1.0000     0.0000     2.0000
  0.6500     0.9600     1.9920E-1     1.0000     0.0000     2.0000
  0.7500     0.9520     1.8231E-1     1.0000     0.0000     2.0000
  0.8500     0.9445     1.5868E-1     1.0000     0.0000     2.0000
  0.9500     0.9392     1.3241E-1     1.0000     0.0000     2.0000
  1.0500     0.9353     1.0617E-1     0.9999     0.0000     1.9165
  1.1500     0.9347     8.3864E-2     0.9999     0.0647     1.8061
  1.2500     0.9357     6.6197E-2     0.9999     0.1631     1.7096
  1.3500     0.9368     5.2943E-2     1.0000     0.2460     1.6288
  1.4500     0.9378     4.2949E-2     1.0000     0.3157     1.5609
  1.5500     0.9386     3.5303E-2     1.0000     0.3747     1.5035
  1.6500     0.9393     2.9363E-2     1.0000     0.4251     1.4544
  1.7500     0.9399     2.4681E-2     1.0000     0.4685     1.4121
  1.8500     0.9404     2.0943E-2     1.0000     0.5062     1.3754
  1.9500     0.9409     1.7922E-2     1.0000     0.5392     1.3432
  2.0500     0.9413     1.5455E-2     1.0000     0.5683     1.3149
  2.1500     0.9416     1.3420E-2     1.0000     0.5941     1.2898
  2.2500     0.9420     1.1727E-2     1.0000     0.6171     1.2674
  2.3500     0.9422     1.0307E-2     1.0000     0.6377     1.2473
  2.4500     0.9425     9.1071E-3     1.0000     0.6563     1.2292
```

```
xstar =  64.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.5000      0.0000E0     0.0000     0.0000     0.0000
  0.1500     0.5999     5.2606E-3     0.9986     0.3734     0.8400
  0.2500     0.8009     5.3450E-2     0.9993     0.0952     1.5193
  0.3500     0.8915     1.0418E-1     0.9997     0.0000     1.8799
  0.4500     0.9287     1.3803E-1     1.0000     0.0000     2.0000
  0.5500     0.9408     1.5524E-1     1.0000     0.0000     2.0000
  0.6500     0.9414     1.5940E-1     1.0000     0.0000     2.0000
  0.7500     0.9366     1.5269E-1     1.0000     0.0000     2.0000
  0.8500     0.9307     1.3793E-1     1.0000     0.0000     2.0000
  0.9500     0.9257     1.1813E-1     0.9999     0.0000     1.9615
  1.0500     0.9224     9.6832E-2     0.9999     0.0000     1.8594
  1.1500     0.9229     7.8208E-2     0.9999     0.0828     1.7644
  1.2500     0.9250     6.2508E-2     0.9999     0.1743     1.6771
  1.3500     0.9272     5.0362E-2     1.0000     0.2535     1.6021
  1.4500     0.9290     4.1070E-2     1.0000     0.3208     1.5384
  1.5500     0.9307     3.3896E-2     1.0000     0.3781     1.4842
  1.6500     0.9321     2.8287E-2     1.0000     0.4273     1.4377
  1.7500     0.9333     2.3845E-2     1.0000     0.4699     1.3974
  1.8500     0.9343     2.0284E-2     1.0000     0.5070     1.3624
  1.9500     0.9352     1.7396E-2     1.0000     0.5395     1.3316
  2.0500     0.9360     1.5031E-2     1.0000     0.5682     1.3045
  2.1500     0.9367     1.3074E-2     1.0000     0.5937     1.2803
  2.2500     0.9373     1.1443E-2     1.0000     0.6165     1.2588
  2.3500     0.9379     1.0071E-2     1.0000     0.6369     1.2395
  2.4500     0.9384     8.9100E-3     1.0000     0.6553     1.2220

xstar = 128.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.5000      0.0000E0     0.0000     0.0000     0.0000
  0.1500     0.5105     4.2586E-5     0.9985     0.4902     0.5325
  0.2500     0.6433     1.0043E-2     0.9993     0.3381     0.9557
  0.3500     0.7707     3.9256E-2     0.9996     0.1710     1.3771
  0.4500     0.8432     6.8969E-2     0.9997     0.0498     1.6418
  0.5500     0.8800     9.2017E-2     0.9999     0.0000     1.7992
  0.6500     0.8962     1.0539E-1     0.9999     0.0000     1.8779
  0.7500     0.9006     1.0914E-1     0.9999     0.0000     1.8981
  0.8500     0.8997     1.0461E-1     0.9999     0.0000     1.8751
  0.9500     0.8977     9.4258E-2     0.9999     0.0000     1.8228
  1.0500     0.8976     8.0973E-2     0.9999     0.0424     1.7543
  1.1500     0.9001     6.8188E-2     0.9999     0.1158     1.6859
  1.2500     0.9044     5.5746E-2     0.9999     0.1954     1.6146
  1.3500     0.9085     4.5565E-2     1.0000     0.2677     1.5505
  1.4500     0.9121     3.7546E-2     1.0000     0.3305     1.4948
  1.5500     0.9153     3.1242E-2     1.0000     0.3848     1.4467
  1.6500     0.9180     2.6249E-2     1.0000     0.4318     1.4050
  1.7500     0.9203     2.2254E-2     1.0000     0.4727     1.3687
  1.8500     0.9223     1.9026E-2     1.0000     0.5085     1.3369
  1.9500     0.9241     1.6390E-2     1.0000     0.5400     1.3089
  2.0500     0.9256     1.4217E-2     1.0000     0.5679     1.2840
  2.1500     0.9270     1.2409E-2     1.0000     0.5929     1.2618
  2.2500     0.9283     1.0894E-2     1.0000     0.6152     1.2419
  2.3500     0.9294     9.6157E-3     1.0000     0.6352     1.2240
  2.4500     0.9304     8.5283E-3     1.0000     0.6534     1.2078
```

```
xstar = 256.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500    0.5000     0.0000E0    0.0000    0.0000    0.0000
  0.1500    0.5001     0.0000E0    0.0000    0.0000    0.0000
  0.2500    0.5269    2.6357E-4    0.9993    0.4775    0.5778
  0.3500    0.6164    5.9693E-3    0.9996    0.3827    0.8533
  0.4500    0.7049    2.0176E-2    0.9997    0.2765    1.1369
  0.5500    0.7671    3.6509E-2    0.9998    0.1915    1.3458
  0.6500    0.8054    5.0176E-2    0.9998    0.1311    1.4823
  0.7500    0.8269    5.9211E-2    0.9999    0.0948    1.5611
  0.8500    0.8382    6.3169E-2    0.9999    0.0823    1.5958
  0.9500    0.8448    6.2535E-2    0.9999    0.0932    1.5981
  1.0500    0.8505    5.8393E-2    0.9999    0.1245    1.5780
  1.1500    0.8577    5.2125E-2    0.9999    0.1720    1.5447
  1.2500    0.8659    4.4353E-2    0.9999    0.2336    1.4994
  1.3500    0.8736    3.7274E-2    1.0000    0.2940    1.4542
  1.4500    0.8803    3.1356E-2    1.0000    0.3488    1.4128
  1.5500    0.8861    2.6523E-2    1.0000    0.3973    1.3758
  1.6500    0.8912    2.2589E-2    1.0000    0.4401    1.3430
  1.7500    0.8955    1.9376E-2    1.0000    0.4779    1.3140
  1.8500    0.8994    1.6734E-2    1.0000    0.5112    1.2882
  1.9500    0.9027    1.4544E-2    1.0000    0.5409    1.2652
  2.0500    0.9057    1.2715E-2    1.0000    0.5674    1.2446
  2.1500    0.9083    1.1177E-2    1.0000    0.5912    1.2260
  2.2500    0.9107    9.8745E-3    1.0000    0.6126    1.2093
  2.3500    0.9128    8.7642E-3    1.0000    0.6320    1.1941
  2.4500    0.9147    7.8126E-3    1.0000    0.6496    1.1803
```

```
xstar = 512.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500    0.5000     0.0000E0    0.0000    0.0000    0.0000
  0.1500    0.5000     0.0000E0    0.0000    0.0000    0.0000
  0.2500    0.5010     0.0000E0    0.0000    0.0000    0.0000
  0.3500    0.5193    1.1353E-4    0.9996    0.4872    0.5521
  0.4500    0.5669    1.6726E-3    0.9997    0.4436    0.6914
  0.5500    0.6234    6.3461E-3    0.9998    0.3835    0.8648
  0.6500    0.6724    1.3254E-2    0.9998    0.3261    1.0204
  0.7500    0.7094    2.0378E-2    0.9999    0.2801    1.1401
  0.8500    0.7356    2.6197E-2    0.9999    0.2491    1.2235
  0.9500    0.7548    2.9948E-2    0.9999    0.2348    1.2761
  1.0500    0.7703    3.1391E-2    0.9999    0.2381    1.3037
  1.1500    0.7850    3.0721E-2    0.9999    0.2586    1.3125
  1.2500    0.7993    2.8069E-2    0.9999    0.2963    1.3033
  1.3500    0.8123    2.4881E-2    1.0000    0.3388    1.2867
  1.4500    0.8238    2.1811E-2    1.0000    0.3806    1.2679
  1.5500    0.8339    1.9071E-2    1.0000    0.4195    1.2491
  1.6500    0.8427    1.6698E-2    1.0000    0.4549    1.2312
  1.7500    0.8504    1.4665E-2    1.0000    0.4871    1.2145
  1.8500    0.8573    1.2929E-2    1.0000    0.5161    1.1990
  1.9500    0.8633    1.1442E-2    1.0000    0.5424    1.1848
  2.0500    0.8687    1.0164E-2    1.0000    0.5662    1.1717
  2.1500    0.8735    9.0624E-3    1.0000    0.5879    1.1596
  2.2500    0.8778    8.1078E-3    1.0000    0.6077    1.1484
  2.3500    0.8817    7.2781E-3    1.0000    0.6258    1.1381
  2.4500    0.8853    6.5544E-3    1.0000    0.6424    1.1285
```

```
xstar = 1024.00 delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500    0.5000     0.0000E0    0.0000    0.0000    0.0000
  0.1500    0.5000     0.0000E0    0.0000    0.0000    0.0000
  0.2500    0.5000     0.0000E0    0.0000    0.0000    0.0000
  0.3500    0.5006     0.0000E0    0.0000    0.0000    0.0000
  0.4500    0.5074     0.0000E0    0.0000    0.0000    0.0000
  0.5500    0.5260    1.9051E-4    0.9998    0.4845    0.5678
  0.6500    0.5536    9.3926E-4    0.9999    0.4614    0.6462
  0.7500    0.5837    2.5222E-3    0.9999    0.4327    0.7353
  0.8500    0.6121    4.7575E-3    0.9999    0.4048    0.8200
  0.9500    0.6372    7.1839E-3    0.9999    0.3826    0.8926
  1.0500    0.6596    9.3071E-3    0.9999    0.3699    0.9501
  1.1500 -  0.6806    1.0755E-2    0.9999    0.3691    0.9927
  1.2500    0.7004    1.1206E-2    0.9999    0.3826    1.0189
  1.3500    0.7187    1.1011E-2    1.0000    0.4037    1.0344
  1.4500    0.7354    1.0478E-2    1.0000    0.4281    1.0432
  1.5500    0.7503    9.7992E-3    1.0000    0.4532    1.0480
  1.6500    0.7637    9.0785E-3    1.0000    0.4778    1.0502
  1.7500    0.7757    8.3689E-3    1.0000    0.5012    1.0507
  1.8500    0.7865    7.6950E-3    1.0000    0.5233    1.0502
  1.9500    0.7963    7.0656E-3    1.0000    0.5441    1.0489
  2.0500    0.8050    6.4834E-3    1.0000    0.5635    1.0470
  2.1500    0.8130    5.9487E-3    1.0000    0.5817    1.0448
  2.2500    0.8203    5.4601E-3    1.0000    0.5986    1.0423
  2.3500    0.8269    5.0150E-3    1.0000    0.6145    1.0397
  2.4500    0.8330    4.6105E-3    1.0000    0.6293    1.0370

xstar = 2048.00 delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500    0.5000     0.0000E0    0.0000    0.0000    0.0000
  0.1500    0.5000     0.0000E0    0.0000    0.0000    0.0000
  0.2500    0.5000     0.0000E0    0.0000    0.0000    0.0000
  0.3500    0.5000     0.0000E0    0.0000    0.0000    0.0000
  0.4500    0.5001     0.0000E0    0.0000    0.0000    0.0000
  0.5500    0.5015     0.0000E0    0.0000    0.0000    0.0000
  0.6500    0.5062     0.0000E0    0.0000    0.0000    0.0000
  0.7500    0.5150    4.9100E-5    0.9999    0.4939    0.5361
  0.8500    0.5273    1.8435E-4    0.9999    0.4865    0.5683
  0.9500    0.5420    4.6257E-4    0.9999    0.4774    0.6067
  1.0500    0.5579    8.8291E-4    0.9999    0.4687    0.6474
  1.1500    0.5749    1.3806E-3    0.9999    0.4633    0.6867
  1.2500    0.5924    1.8287E-3    0.9999    0.4640    0.7210
  1.3500    0.6099    2.1792E-3    1.0000    0.4698    0.7503
  1.4500    0.6269    2.4287E-3    1.0000    0.4790    0.7751
  1.5500    0.6432    2.5919E-3    1.0000    0.4905    0.7963
  1.6500    0.6586    2.6867E-3    1.0000    0.5031    0.8145
  1.7500    0.6731    2.7284E-3    1.0000    0.5163    0.8301
  1.8500    0.6866    2.7292E-3    1.0000    0.5298    0.8436
  1.9500    0.6992    2.6977E-3    1.0000    0.5434    0.8553
  2.0500    0.7109    2.6415E-3    1.0000    0.5567    0.8654
  2.1500    0.7218    2.5673E-3    1.0000    0.5699    0.8741
  2.2500    0.7320    2.4804E-3    1.0000    0.5826    0.8817
  2.3500    0.7415    2.3853E-3    1.0000    0.5950    0.8883
  2.4500    0.7503    2.2854E-3    1.0000    0.6070    0.8940
```

```
Solution of Entropy Maximization Equations

Drop Size Distributions

Constraints:
0 - normalization
1 - surface energy   d**2
2 - mass             d**3
3 - momentum         d**3 x v
4 - kinetic energy   d**3 x v**2

Acceleration = 1.00                        I = 24          DSMAX = 2.5000
TSTAR        = 0.4000                      J = 25          VSMAX = 2.0000

lambda 0 to 4 =


1.3856E0       -2.4787E0       4.0071E0       -5.4506E0       2.8813E0
```

```
Alpha = 4.2666E0          Beta = 1.4057E0          Gamma = 1.6560E0
Kappa = 1.0530E0             a = 1.0000E0          dsm = 2.7034E0

dstar     Max Ent       R.R.          N.T.        Log Prob     Upper Limit

 0.05   5.0328E-1    3.9772E-1    2.5320E-2    2.7883E-8    2.4567E-3
 0.10   5.1165E-1    4.7844E-1    9.4933E-2    1.1662E-4    4.7242E-2
 0.15   5.2476E-1    5.3303E-1    1.9720E-1    4.5149E-3    1.6793E-1
 0.20   5.4199E-1    5.7543E-1    3.1990E-1    3.4975E-2    3.3713E-1
 0.25   5.6273E-1    6.1049E-1    4.5161E-1    1.2521E-1    5.1596E-1
 0.30   5.8632E-1    6.4048E-1    5.8252E-1    2.8985E-1    6.7813E-1
 0.35   6.1203E-1    6.6659E-1    7.0474E-1    5.1120E-1    8.1085E-1
 0.40   6.3901E-1    6.8951E-1    8.1242E-1    7.5214E-1    9.1038E-1
 0.45   6.6634E-1    7.0957E-1    9.0162E-1    9.7493E-1    9.7800E-1
 0.50   6.9299E-1    7.2695E-1    9.7015E-1     1.1528E0     1.0173E0
 0.55   7.1791E-1    7.4164E-1     1.0173E0     1.2730E0     1.0329E0
 0.60   7.4005E-1    7.5355E-1     1.0435E0     1.3342E0     1.0291E0
 0.65   7.5841E-1    7.6250E-1     1.0502E0     1.3431E0     1.0102E0
 0.70   7.7214E-1    7.6823E-1     1.0395E0     1.3096E0    9.7976E-1
 0.75   7.8056E-1    7.7047E-1     1.0136E0     1.2450E0    9.4066E-1
 0.80   7.8318E-1    7.6890E-1    9.7523E-1     1.1598E0    8.9545E-1
 0.85   7.7975E-1    7.6320E-1    9.2699E-1     1.0626E0    8.4616E-1
 0.90   7.7018E-1    7.5307E-1    8.7141E-1    9.6069E-1    7.9441E-1
 0.95   7.5456E-1    7.3825E-1    8.1086E-1    8.5895E-1    7.4147E-1
 1.00   7.3310E-1    7.1855E-1    7.4742E-1    7.6104E-1    6.8834E-1
 1.05   7.0607E-1    6.9388E-1    6.8293E-1    6.6927E-1    6.3581E-1
 1.10   6.7383E-1    6.6427E-1    6.1890E-1    5.8494E-1    5.8445E-1
 1.15   6.3681E-1    6.2990E-1    5.5657E-1    5.0863E-1    5.3472E-1
 1.20   5.9551E-1    5.9111E-1    4.9690E-1    4.4042E-1    4.8694E-1
 1.25   5.5057E-1    5.4844E-1    4.4059E-1    3.8004E-1    4.4134E-1
 1.30   5.0273E-1    5.0258E-1    3.8813E-1    3.2701E-1    3.9808E-1
 1.35   4.5290E-1    4.5438E-1    3.3981E-1    2.8074E-1    3.5726E-1
 1.40   4.0208E-1    4.0484E-1    2.9575E-1    2.4057E-1    3.1893E-1
 1.45   3.5138E-1    3.5504E-1    2.5596E-1    2.0585E-1    2.8312E-1
 1.50   3.0191E-1    3.0610E-1    2.2032E-1    1.7594E-1    2.4981E-1
 1.55   2.5475E-1    2.5909E-1    1.8867E-1    1.5025E-1    2.1897E-1
 1.60   2.1085E-1    2.1500E-1    1.6076E-1    1.2823E-1    1.9057E-1
 1.65   1.7099E-1    1.7467E-1    1.3632E-1    1.0940E-1    1.6454E-1
 1.70   1.3569E-1    1.3872E-1    1.1506E-1    9.3312E-2    1.4081E-1
 1.75   1.0526E-1    1.0753E-1    9.6688E-2    7.9584E-2    1.1932E-1
 1.80   7.9731E-2    8.1224E-2    8.0894E-2    6.7880E-2    9.9993E-2
 1.85   5.8896E-2    5.9683E-2    6.7395E-2    5.7908E-2    8.2735E-2
 1.90   4.2382E-2    4.2588E-2    5.5921E-2    4.9415E-2    6.7465E-2
 1.95   2.9676E-2    2.9459E-2    4.6216E-2    4.2183E-2    5.4094E-2
 2.00   2.0197E-2    1.9716E-2    3.8049E-2    3.6025E-2    4.2527E-2
 2.05   1.3346E-2    1.2742E-2    3.1208E-2    3.0782E-2    3.2665E-2
 2.10   8.5525E-3    7.9374E-3    2.5504E-2    2.6317E-2    2.4402E-2
 2.15   5.3091E-3    4.7553E-3    2.0768E-2    2.2514E-2    1.7626E-2
 2.20   3.1891E-3    2.7342E-3    1.6853E-2    1.9272E-2    1.2214E-2
 2.25   1.8516E-3    1.5056E-3    1.3629E-2    1.6509E-2    8.0356E-3
 2.30   1.0379E-3    7.9218E-4    1.0986E-2    1.4152E-2    4.9450E-3
 2.35   5.6118E-4    3.9734E-4    8.8268E-3    1.2141E-2    2.7863E-3
 2.40   2.9228E-4    1.8954E-4    7.0694E-3    1.0424E-2    1.3916E-3
 2.45   1.4649E-4    8.5784E-5    5.6443E-3    8.9566E-3    5.8461E-4
 2.50   7.0582E-5    3.6741E-5    4.4927E-3    7.7019E-3    1.8847E-4

Total    1.9747E1     1.9876E1     1.9983E1     1.9950E1     2.0000E1
```

```
Initial Conditions      delta, mean v* and variance.

  0.0500     1.0000     3.3330E-1
  0.1500     0.9996     3.3247E-1
  0.2500     0.9984     3.2934E-1
  0.3500     0.9957     3.2247E-1
  0.4500     0.9912     3.1055E-1
  0.5500     0.9848     2.9265E-1
  0.6500     0.9770     2.6846E-1
  0.7500     0.9686     2.3867E-1
  0.8500     0.9606     2.0513E-1
  0.9500     0.9542     1.7068E-1
  1.0500     0.9499     1.3837E-1
  1.1500     0.9475     1.1046E-1
  1.2500     0.9464     8.7873E-2
  1.3500     0.9460     7.0311E-2
  1.4500     0.9459     5.6880E-2
  1.5500     0.9459     4.6593E-2
  1.6500     0.9459     3.8629E-2
  1.7500     0.9459     3.2378E-2
  1.8500     0.9459     2.7407E-2
  1.9500     0.9459     2.3403E-2
  2.0500     0.9459     2.0142E-2
  2.1500     0.9459     1.7460E-2
  2.2500     0.9459     1.5234E-2
  2.3500     0.9459     1.3371E-2
  2.4500     0.9459     1.1799E-2

xstar =   2.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.8906     1.6525E-1     1.0000     0.0000     2.0000
  0.1500     0.9903     2.8895E-1     1.0000     0.0000     2.0000
  0.2500     0.9959     3.0759E-1     1.0000     0.0000     2.0000
  0.3500     0.9947     3.0903E-1     1.0000     0.0000     2.0000
  0.4500     0.9906     3.0146E-1     1.0000     0.0000     2.0000
  0.5500     0.9843     2.8628E-1     1.0000     0.0000     2.0000
  0.6500     0.9765     2.6398E-1     1.0000     0.0000     2.0000
  0.7500     0.9680     2.3557E-1     1.0000     0.0000     2.0000
  0.8500     0.9600     2.0305E-1     1.0000     0.0000     2.0000
  0.9500     0.9536     1.6934E-1     1.0000     0.0000     2.0000
  1.0500     0.9492     1.3753E-1     1.0000     0.0000     2.0000
  1.1500     0.9462     1.0933E-1     0.9993     0.0000     1.9446
  1.2500     0.9452     8.6076E-2     0.9981     0.0571     1.8357
  1.3500     0.9453     6.8432E-2     0.9975     0.1505     1.7415
  1.4500     0.9453     5.5261E-2     0.9973     0.2304     1.6614
  1.5500     0.9454     4.5255E-2     0.9973     0.2983     1.5934
  1.6500     0.9454     3.7523E-2     0.9973     0.3562     1.5355
  1.7500     0.9455     3.1456E-2     0.9973     0.4060     1.4857
  1.8500     0.9455     2.6629E-2     0.9973     0.4492     1.4425
  1.9500     0.9455     2.2742E-2     0.9973     0.4869     1.4048
  2.0500     0.9455     1.9575E-2     0.9973     0.5201     1.3716
  2.1500     0.9456     1.6970E-2     0.9973     0.5494     1.3423
  2.2500     0.9456     1.4808E-2     0.9973     0.5756     1.3162
  2.3500     0.9456     1.2998E-2     0.9973     0.5990     1.2928
  2.4500     0.9456     1.1471E-2     0.9973     0.6200     1.2717
```

```
xstar =   4.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

 0.0500    0.7840    8.0013E-2    1.0000    0.0000    2.0000
 0.1500    0.9785    2.5252E-1    1.0000    0.0000    2.0000
 0.2500    0.9927    2.8781E-1    1.0000    0.0000    2.0000
 0.3500    0.9934    2.9640E-1    1.0000    0.0000    2.0000
 0.4500    0.9899    2.9277E-1    1.0000    0.0000    2.0000
 0.5500    0.9838    2.8012E-1    1.0000    0.0000    2.0000
 0.6500    0.9760    2.5961E-1    1.0000    0.0000    2.0000
 0.7500    0.9674    2.3253E-1    1.0000    0.0000    2.0000
 0.8500    0.9594    2.0101E-1    1.0000    0.0000    2.0000
 0.9500    0.9529    1.6802E-1    1.0000    0.0000    2.0000
 1.0500    0.9485    1.3670E-1    1.0000    0.0000    2.0000
 1.1500    0.9454    1.0874E-1    0.9999    0.0000    1.9382
 1.2500    0.9445    8.5579E-2    0.9997    0.0651    1.8254
 1.3500    0.9447    6.8028E-2    0.9996    0.1605    1.7301
 1.4500    0.9448    5.4943E-2    0.9996    0.2401    1.6506
 1.5500    0.9449    4.5004E-2    0.9996    0.3072    1.5836
 1.6500    0.9450    3.7322E-2    0.9996    0.3643    1.5265
 1.7500    0.9450    3.1293E-2    0.9996    0.4134    1.4775
 1.8500    0.9451    2.6495E-2    0.9996    0.4559    1.4350
 1.9500    0.9452    2.2630E-2    0.9996    0.4931    1.3979
 2.0500    0.9452    1.9482E-2    0.9996    0.5258    1.3653
 2.1500    0.9453    1.6891E-2    0.9996    0.5547    1.3364
 2.2500    0.9453    1.4740E-2    0.9996    0.5805    1.3107
 2.3500    0.9453    1.2939E-2    0.9996    0.6036    1.2876
 2.4500    0.9454    1.1420E-2    0.9996    0.6243    1.2669

xstar =   8.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

 0.0500    0.6253    1.3770E-2    0.9974    0.1347    1.2533
 0.1500    0.9506    1.9525E-1    1.0000    0.0000    2.0000
 0.2500    0.9844    2.5325E-1    1.0000    0.0000    2.0000
 0.3500    0.9899    2.7332E-1    1.0000    0.0000    2.0000
 0.4500    0.9880    2.7649E-1    1.0000    0.0000    2.0000
 0.5500    0.9825    2.6839E-1    1.0000    0.0000    2.0000
 0.6500    0.9748    2.5119E-1    1.0000    0.0000    2.0000
 0.7500    0.9662    2.2662E-1    1.0000    0.0000    2.0000
 0.8500    0.9581    1.9700E-1    1.0000    0.0000    2.0000
 0.9500    0.9515    1.6541E-1    1.0000    0.0000    2.0000
 1.0500    0.9471    1.3506E-1    1.0000    0.0000    2.0000
 1.1500    0.9439    1.0764E-1    1.0000    0.0000    1.9314
 1.2500    0.9432    8.4899E-2    0.9999    0.0681    1.8197
 1.3500    0.9434    6.7570E-2    0.9999    0.1630    1.7251
 1.4500    0.9437    5.4613E-2    1.0000    0.2421    1.6463
 1.5500    0.9439    4.4758E-2    1.0000    0.3089    1.5799
 1.6500    0.9440    3.7134E-2    1.0000    0.3657    1.5233
 1.7500    0.9442    3.1147E-2    1.0000    0.4146    1.4746
 1.8500    0.9443    2.6380E-2    1.0000    0.4570    1.4325
 1.9500    0.9444    2.2537E-2    1.0000    0.4940    1.3956
 2.0500    0.9445    1.9406E-2    1.0000    0.5266    1.3632
 2.1500    0.9446    1.6829E-2    1.0000    0.5554    1.3345
 2.2500    0.9447    1.4689E-2    1.0000    0.5811    1.3089
 2.3500    0.9448    1.2897E-2    1.0000    0.6041    1.2860
 2.4500    0.9448    1.1385E-2    1.0000    0.6248    1.2654
```

```
xstar =  16.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.5145      1.6430E-4     0.9973     0.4563     0.5953
  0.1500     0.8878      1.1989E-1     1.0000     0.0000     2.0000
  0.2500     0.9629      1.9920E-1     1.0000     0.0000     2.0000
  0.3500     0.9804      2.3433E-1     1.0000     0.0000     2.0000
  0.4500     0.9828      2.4774E-1     1.0000     0.0000     2.0000
  0.5500     0.9790      2.4707E-1     1.0000     0.0000     2.0000
  0.6500     0.9720      2.3556E-1     1.0000     0.0000     2.0000
  0.7500     0.9635      2.1547E-1     1.0000     0.0000     2.0000
  0.8500     0.9553      1.8934E-1     1.0000     0.0000     2.0000
  0.9500     0.9487      1.6036E-1     1.0000     0.0000     2.0000
  1.0500     0.9444      1.3186E-1     1.0000     0.0000     2.0000
  1.1500     0.9411      1.0549E-1     1.0000     0.0000     1.9186
  1.2500     0.9405      8.3601E-2     0.9999     0.0721     1.8103
  1.3500     0.9410      6.6710E-2     1.0000     0.1655     1.7177
  1.4500     0.9414      5.4001E-2     1.0000     0.2439     1.6401
  1.5500     0.9419      4.4305E-2     1.0000     0.3101     1.5746
  1.6500     0.9422      3.6790E-2     1.0000     0.3666     1.5188
  1.7500     0.9425      3.0880E-2     1.0000     0.4152     1.4707
  1.8500     0.9428      2.6170E-2     1.0000     0.4574     1.4290
  1.9500     0.9430      2.2371E-2     1.0000     0.4942     1.3925
  2.0500     0.9432      1.9272E-2     1.0000     0.5267     1.3604
  2.1500     0.9434      1.6720E-2     1.0000     0.5555     1.3319
  2.2500     0.9435      1.4599E-2     1.0000     0.5811     1.3066
  2.3500     0.9437      1.2822E-2     1.0000     0.6040     1.2839
  2.4500     0.9438      1.1322E-2     1.0000     0.6246     1.2635

xstar =  32.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.5001       0.0000E0     0.0000     0.0000     0.0000
  0.1500     0.7624      4.3349E-2     0.9986     0.1112     1.4420
  0.2500     0.9113      1.2827E-1     1.0000     0.0000     2.0000
  0.3500     0.9548      1.7657E-1     1.0000     0.0000     2.0000
  0.4500     0.9683      2.0193E-1     1.0000     0.0000     2.0000
  0.5500     0.9696      2.1136E-1     1.0000     0.0000     2.0000
  0.6500     0.9648      2.0839E-1     1.0000     0.0000     2.0000
  0.7500     0.9574      1.9549E-1     1.0000     0.0000     2.0000
  0.8500     0.9495      1.7528E-1     1.0000     0.0000     2.0000
  0.9500     0.9430      1.5092E-1     1.0000     0.0000     2.0000
  1.0500     0.9387      1.2577E-1     1.0000     0.0000     2.0000
  1.1500     0.9353      1.0134E-1     1.0000     0.0000     1.8935
  1.2500     0.9352      8.1078E-2     0.9999     0.0800     1.7918
  1.3500     0.9361      6.5026E-2     1.0000     0.1705     1.7030
  1.4500     0.9371      5.2798E-2     1.0000     0.2473     1.6279
  1.5500     0.9379      4.3413E-2     1.0000     0.3125     1.5642
  1.6500     0.9386      3.6112E-2     1.0000     0.3682     1.5098
  1.7500     0.9392      3.0355E-2     1.0000     0.4163     1.4628
  1.8500     0.9397      2.5757E-2     1.0000     0.4581     1.4220
  1.9500     0.9401      2.2041E-2     1.0000     0.4947     1.3863
  2.0500     0.9405      1.9006E-2     1.0000     0.5269     1.3548
  2.1500     0.9409      1.6504E-2     1.0000     0.5555     1.3269
  2.2500     0.9412      1.4422E-2     1.0000     0.5810     1.3021
  2.3500     0.9415      1.2675E-2     1.0000     0.6038     1.2798
  2.4500     0.9418      1.1199E-2     1.0000     0.6243     1.2597
```

```
xstar =  64.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500    0.5000     0.0000E0    0.0000    0.0000    0.0000
  0.1500    0.6000    5.2641E-3    0.9986    0.3734    0.8401
  0.2500    0.8013    5.3586E-2    0.9993    0.0947    1.5206
  0.3500    0.8927    1.0488E-1    0.9997    0.0000    1.8845
  0.4500    0.9310    1.3986E-1    1.0000    0.0000    2.0000
  0.5500    0.9447    1.5911E-1    1.0000    0.0000    2.0000
  0.6500    0.9467    1.6619E-1    1.0000    0.0000    2.0000
  0.7500    0.9428    1.6290E-1    1.0000    0.0000    2.0000
  0.8500    0.9367    1.5134E-1    1.0000    0.0000    2.0000
  0.9500    0.9311    1.3424E-1    1.0000    0.0000    2.0000
  1.0500    0.9265    1.1387E-1    0.9999    0.0000    1.9428
  1.1500    0.9240    9.3606E-2    0.9999    0.0045    1.8448
  1.2500    0.9248    7.6297E-2    0.9999    0.0952    1.7557
  1.3500    0.9266    6.1785E-2    1.0000    0.1803    1.6741
  1.4500    0.9284    5.0470E-2    1.0000    0.2540    1.6038
  1.5500    0.9300    4.1681E-2    1.0000    0.3172    1.5437
  1.6500    0.9313    3.4792E-2    1.0000    0.3716    1.4920
  1.7500    0.9325    2.9329E-2    1.0000    0.4186    1.4473
  1.8500    0.9336    2.4948E-2    1.0000    0.4596    1.4083
  1.9500    0.9345    2.1396E-2    1.0000    0.4956    1.3741
  2.0500    0.9353    1.8486E-2    1.0000    0.5274    1.3438
  2.1500    0.9360    1.6080E-2    1.0000    0.5556    1.3170
  2.2500    0.9366    1.4073E-2    1.0000    0.5807    1.2931
  2.3500    0.9372    1.2385E-2    1.0000    0.6033    1.2716
  2.4500    0.9377    1.0957E-2    1.0000    0.6237    1.2522

xstar = 128.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500    0.5000     0.0000E0    0.0000    0.0000    0.0000
  0.1500    0.5105    4.2618E-5    0.9985    0.4902    0.5325
  0.2500    0.6435    1.0070E-2    0.9993    0.3379    0.9564
  0.3500    0.7716    3.9512E-2    0.9996    0.1700    1.3800
  0.4500    0.8454    6.9907E-2    0.9997    0.0466    1.6494
  0.5500    0.8840    9.4321E-2    0.9999    0.0000    1.8147
  0.6500    0.9022    1.0981E-1    0.9999    0.0000    1.9042
  0.7500    0.9080    1.1623E-1    0.9999    0.0000    1.9375
  0.8500    0.9074    1.1442E-1    0.9999    0.0000    1.9277
  0.9500    0.9045    1.0619E-1    0.9999    0.0000    1.8865
  1.0500    0.9022    9.3968E-2    0.9999    0.0000    1.8252
  1.1500    0.9021    8.0343E-2    0.9999    0.0505    1.7551
  1.2500    0.9045    6.7671E-2    0.9999    0.1233    1.6870
  1.3500    0.9081    5.5790E-2    1.0000    0.1990    1.6184
  1.4500    0.9116    4.6113E-2    1.0000    0.2669    1.5572
  1.5500    0.9146    3.8416E-2    1.0000    0.3263    1.5038
  1.6500    0.9173    3.2290E-2    1.0000    0.3780    1.4574
  1.7500    0.9196    2.7379E-2    1.0000    0.4231    1.4169
  1.8500    0.9216    2.3405E-2    1.0000    0.4626    1.3814
  1.9500    0.9234    2.0160E-2    1.0000    0.4974    1.3501
  2.0500    0.9249    1.7486E-2    1.0000    0.5282    1.3223
  2.1500    0.9263    1.5263E-2    1.0000    0.5557    1.2975
  2.2500    0.9275    1.3399E-2    1.0000    0.5803    1.2754
  2.3500    0.9286    1.1825E-2    1.0000    0.6024    1.2554
  2.4500    0.9296    1.0488E-2    1.0000    0.6224    1.2373
```

```
xstar = 256.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.5000      0.0000E0     0.0000     0.0000     0.0000
  0.1500     0.5001      0.0000E0     0.0000     0.0000     0.0000
  0.2500     0.5270     2.6437E-4     0.9993     0.4775     0.5779
  0.3500     0.6168     6.0117E-3     0.9996     0.3823     0.8546
  0.4500     0.7064     2.0447E-2     0.9997     0.2750     1.1412
  0.5500     0.7702     3.7343E-2     0.9998     0.1880     1.3554
  0.6500     0.8104     5.1993E-2     0.9998     0.1240     1.4995
  0.7500     0.8336     6.2415E-2     0.9999     0.0819     1.5875
  0.8500     0.8457     6.8000E-2     0.9999     0.0614     1.6318
  0.9500     0.8519     6.8924E-2     0.9999     0.0626     1.6427
  1.0500     0.8559     6.6027E-2     0.9999     0.0838     1.6295
  1.1500     0.8603     6.0529E-2     0.9999     0.1213     1.6006
  1.2500     0.8665     5.3438E-2     0.9999     0.1723     1.5618
  1.3500     0.8734     4.5501E-2     1.0000     0.2330     1.5149
  1.4500     0.8798     3.8478E-2     1.0000     0.2910     1.4696
  1.5500     0.8856     3.2616E-2     1.0000     0.3435     1.4285
  1.6500     0.8905     2.7799E-2     1.0000     0.3902     1.3917
  1.7500     0.8949     2.3847E-2     1.0000     0.4315     1.3591
  1.8500     0.8987     2.0592E-2     1.0000     0.4681     1.3300
  1.9500     0.9021     1.7894E-2     1.0000     0.5007     1.3041
  2.0500     0.9050     1.5642E-2     1.0000     0.5298     1.2809
  2.1500     0.9077     1.3749E-2     1.0000     0.5559     1.2600
  2.2500     0.9100     1.2145E-2     1.0000     0.5794     1.2412
  2.3500     0.9121     1.0779E-2     1.0000     0.6007     1.2241
  2.4500     0.9140     9.6089E-3     1.0000     0.6200     1.2086

xstar = 512.00  delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.5000      0.0000E0     0.0000     0.0000     0.0000
  0.1500     0.5000      0.0000E0     0.0000     0.0000     0.0000
  0.2500     0.5010      0.0000E0     0.0000     0.0000     0.0000
  0.3500     0.5194     1.1436E-4     0.9996     0.4871     0.5523
  0.4500     0.5674     1.6964E-3     0.9997     0.4432     0.6928
  0.5500     0.6249     6.4965E-3     0.9998     0.3822     0.8692
  0.6500     0.6754     1.3729E-2     0.9998     0.3229     1.0296
  0.7500     0.7140     2.1410E-2     0.9999     0.2739     1.1555
  0.8500     0.7414     2.7984E-2     0.9999     0.2385     1.2456
  0.9500     0.7607     3.2620E-2     0.9999     0.2179     1.3047
  1.0500     0.7752     3.5025E-2     0.9999     0.2129     1.3386
  1.1500     0.7877     3.5256E-2     0.9999     0.2237     1.3527
  1.2500     0.8002     3.3546E-2     0.9999     0.2502     1.3511
  1.3500     0.8124     3.0269E-2     1.0000     0.2901     1.3356
  1.4500     0.8235     2.6746E-2     1.0000     0.3326     1.3153
  1.5500     0.8335     2.3464E-2     1.0000     0.3737     1.2940
  1.6500     0.8422     2.0568E-2     1.0000     0.4118     1.2733
  1.7500     0.8499     1.8066E-2     1.0000     0.4466     1.2539
  1.8500     0.8567     1.5922E-2     1.0000     0.4781     1.2359
  1.9500     0.8627     1.4085E-2     1.0000     0.5066     1.2194
  2.0500     0.8681     1.2509E-2     1.0000     0.5325     1.2042
  2.1500     0.8729     1.1150E-2     1.0000     0.5561     1.1902
  2.2500     0.8772     9.9747E-3     1.0000     0.5776     1.1773
  2.3500     0.8811     8.9532E-3     1.0000     0.5973     1.1654
  2.4500     0.8846     8.0625E-3     1.0000     0.6153     1.1544
```

```
xstar = 1024.00 delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.5000      0.0000E0    0.0000     0.0000     0.0000
  0.1500     0.5000      0.0000E0    0.0000     0.0000     0.0000
  0.2500     0.5000      0.0000E0    0.0000     0.0000     0.0000
  0.3500     0.5006      0.0000E0    0.0000     0.0000     0.0000
  0.4500     0.5075      0.0000E0    0.0000     0.0000     0.0000
  0.5500     0.5263     1.9488E-4    0.9998     0.4843     0.5686
  0.6500     0.5545     9.7253E-4    0.9999     0.4607     0.6487
  0.7500     0.5855     2.6484E-3    0.9999     0.4308     0.7408
  0.8500     0.6148     5.0725E-3    0.9999     0.4008     0.8295
  0.9500     0.6405     7.7939E-3    0.9999     0.3752     0.9064
  1.0500     0.6627     1.0321E-2    0.9999     0.3575     0.9685
  1.1500     0.6825     1.2259E-2    0.9999     0.3499     1.0156
  1.2500     0.7012     1.3323E-2    0.9999     0.3546     1.0484
  1.3500     0.7189     1.3374E-2    1.0000     0.3717     1.0667
  1.4500     0.7352     1.2861E-2    1.0000     0.3947     1.0762
  1.5500     0.7500     1.2082E-2    1.0000     0.4201     1.0804
  1.6500     0.7633     1.1208E-2    1.0000     0.4456     1.0816
  1.7500     0.7753     1.0329E-2    1.0000     0.4703     1.0808
  1.8500     0.7861     9.4896E-3    1.0000     0.4938     1.0789
  1.9500     0.7958     8.7063E-3    1.0000     0.5158     1.0762
  2.0500     0.8046     7.9847E-3    1.0000     0.5365     1.0731
  2.1500     0.8125     7.3237E-3    1.0000     0.5558     1.0697
  2.2500     0.8198     6.7204E-3    1.0000     0.5738     1.0661
  2.3500     0.8264     6.1715E-3    1.0000     0.5907     1.0624
  2.4500     0.8324     5.6730E-3    1.0000     0.6065     1.0587
```

```
xstar = 2048.00 delta, mean(v*), var(v*), check and bounds.

curve plotted for this set of data.

  0.0500     0.5000      0.0000E0    0.0000     0.0000     0.0000
  0.1500     0.5000      0.0000E0    0.0000     0.0000     0.0000
  0.2500     0.5000      0.0000E0    0.0000     0.0000     0.0000
  0.3500     0.5000      0.0000E0    0.0000     0.0000     0.0000
  0.4500     0.5001      0.0000E0    0.0000     0.0000     0.0000
  0.5500     0.5015      0.0000E0    0.0000     0.0000     0.0000
  0.6500     0.5063      0.0000E0    0.0000     0.0000     0.0000
  0.7500     0.5153     5.1299E-5    0.9999     0.4937     0.5369
  0.8500     0.5279     1.9561E-4    0.9999     0.4859     0.5701
  0.9500     0.5429     4.9982E-4    0.9999     0.4757     0.6102
  1.0500     0.5589     9.7621E-4    0.9999     0.4651     0.6530
  1.1500     0.5755     1.5717E-3    0.9999     0.4565     0.6948
  1.2500     0.5926     2.1770E-3    0.9999     0.4525     0.7330
  1.3500     0.6098     2.6573E-3    1.0000     0.4551     0.7649
  1.4500     0.6267     2.9972E-3    1.0000     0.4624     0.7913
  1.5500     0.6429     3.2133E-3    1.0000     0.4728     0.8134
  1.6500     0.6583     3.3327E-3    1.0000     0.4851     0.8319
  1.7500     0.6727     3.3801E-3    1.0000     0.4983     0.8475
  1.8500     0.6862     3.3748E-3    1.0000     0.5119     0.8608
  1.9500     0.6988     3.3310E-3    1.0000     0.5257     0.8723
  2.0500     0.7106     3.2584E-3    1.0000     0.5393     0.8821
  2.1500     0.7215     3.1646E-3    1.0000     0.5527     0.8905
  2.2500     0.7316     3.0560E-3    1.0000     0.5658     0.8978
  2.3500     0.7411     2.9377E-3    1.0000     0.5785     0.9040
  2.4500     0.7499     2.8140E-3    1.0000     0.5908     0.9093
```

The code on the following pages will perform the calculations for the original size - velocity distribution and the velocity calculations for the downstream flow. In addition, it calculates comparison values for the four empirical distributions.

The code is written in the C language and has been run using the DeSmet C compiler on an IBM PC with an 8087 numeric coprocessor installed. The IBM PC was chosen for this project both because there was one available for use, and because it is the smallest machine that the code will run on with reasonable execution times. One run, taking the solution out to an x_* value of 2048, requires about 8 hours of computing. Without the 8087 the execution time would be on the order of a week or more. If the program had been attempted in an interpretive language such as BASIC the execution time could be on the order of months.

```
#include "stdio.h"
#include "math.h"
#include "mathfns.h"
#include "drops.h"
#include "pix.h"
#include "pix.dcl"

#define I       24                /* index for dstar space  0 - I */
#define IP      25                /* plus one                     */
#define J       25                /* index for vstar space  0 - J */
#define JP      26                /* plus one                     */
#define NCON    4                 /* # of physical constraints    */
#define NCONP   5                 /* plus one                     */
#define NDIM    5                 /* array dimension for solver   */
#define PANELS  30                /* # of panels for simpint      */
#define XPANELS 60                /* Xtra large # of panels       */
#define DSMAX   2.5               /* max value for dstar          */
#define VSMAX   2.                /* max value for vstar          */
#define DUNIEXP 1.                /* discretization exponent      */
#define VUNIEXP 1.                /* discretization exponent      */
#define USE_EXP FALSE             /* set false if above 2 equal 1 */

double  lamnot(),pdf(),cdwhite(),vnew();
double  phinot(),dtgf(),dtgfp();
double  vnewl(),vnewlp(),pdfv(),pdfvp();
double  fvvn(),fvvnp(),fvvn2(),fvvn2p();
double  fvv(),fvvp(),fvv2(),fvv2p();
double  me_fdt();
double  plot_me_fdt(),plot_rr_fd(),plot_nt_fd(),plot_lp_fd();
double  plot_ul_fd(),plot_me_fmt(),plot_rr_fm(),plot_nt_fm();
double  plot_lp_fm(),plot_ul_fm();
double  data_in();

int     verbosity;                /* output detail level          */
int     cycles;                   /* iterations between output    */
double  dds;                      /* the dstar increment          */
double  dvs;                      /* the vstar increment          */
```

```
main(){
    double  lambda[12] = { 1.3856, -2.4787, 4.0071, -5.4506, 2.8813 };
    double  lamstore[IP][3];    /* lambda storage area            */
    double  nlam[NCONP];        /* normalized lambda values       */
    double  g[NCONP];           /* expectation values             */
    double  gcal[NCONP];
    double  dstar[IP];
    double  vstar[JP];
    double  dvparm[IP][JP][NCONP];  /* conservation parameters    */
    double  f0[IP][JP];         /* the field of PDF values        */
    double  fd[IP];             /* the 1D size PDF field          */
    double  vbar[IP][3];        /* avg of vs, vs^2 and variance   */
    double  converge = 0.00001; /* a convergence test value       */
    double  con2 = 0.0001;      /* a less demanding value         */
    double  old;
    double  cond;
    double  acc = 1.;           /* an acceleration parameter      */
    double  tstar;              /* the tau star value             */
    double  u_air;              /* the air free stream velocity   */
    double  u_air_star;         /*      non-dimensionalized       */
    double  v_sheet;    /* m/s      sheet velocity                */
    double  nu;         /* m^2/s    Kinematic viscosity of air    */
    double  nu_star;            /*      non-dimensionalized       */
    double  d30;        /* m        mass mean diameter            */
    double  rho_liq;    /* kg/m^3   liquid density                */
    double  rho_air;    /* kg/m^3   air density                   */
    double  dratio;             /* density ratio  air/liq         */
    double  alpha;              /* Rosin Rammler exponent         */
    double  beta;               /* Nukiyama Tansawa exponent      */
    double  gam;                /* Log Probability exponent       */
    double  Kappa;              /* Upper Limit exponent           */
    double  a;                  /* Upper Limit parameter          */
    double  dxstar = 2.0;       /* the xstar increment            */
    double  xstar = 0.;         /* current xstar space location   */
    double  xs_max;             /* maximum value for xstar        */
    double  last_plot_xs = 0.;  /* last xstar plotted             */
    double  l,d2;               /* normalized mean and variance   */
    double  vlow,vhigh,varhigh = 0.;
    double  lower,upper,range;
    double  x,x1;               /* counters and stuff             */
    int     i,j,k;              /* counters and stuff             */
    int     count;              /* cycle counter                  */
    int     prcnt = 0;          /* # of outputs printed           */
    int     prflag = FALSE;     /* output print flag              */
    int     plotflag = FALSE;   /* output plot flag               */
    int     singlev[IP];        /* single velocity flag           */
    int     equalv[IP];         /* vs = u_air_star flag           */
    int     infile;
```

```
verbosity = 1;                 /* initialization of externals  */
cycles = 1;                    /* # of cycles between outputs  */
count = cycles-1;              /* to give output after cycles  */
dds = DSMAX/IP;
dvs = VSMAX/JP;

for(i=0;i<=I;i++){
    singlev[i] = FALSE;
    equalv[i] = FALSE;
    }

/* Get some parameter values from the input file.              */

infile = fopen("ddrag_in.dta","r");
tstar = data_in(infile);
u_air = data_in(infile);
v_sheet = data_in(infile);
nu = data_in(infile);
d30 = data_in(infile);
rho_liq = data_in(infile);
rho_air = data_in(infile);
xs_max = data_in(infile);
fclose(infile);

/* Initialize and solve initial distribution.                  */

u_air_star = u_air/v_sheet;
nu_star = nu/(v_sheet*d30);
dratio = rho_air/rho_liq;
g[0] = 1.;
g[1] = 1./(3.*tstar);
g[2] = 1.;
g[3] = .95;
g[4] = 1.;

if(verbosity > 0) prtheader(acc,I,DSMAX,tstar,J,VSMAX);

parminit(dvparm,dstar,vstar);
lambda[0]=lamnot(dvparm,lambda,dstar,vstar);
fsolve(lambda,dvparm,dstar,vstar,g,acc,converge);
f0cal(dstar,vstar,dvparm,lambda,f0);
```

```
/* Print Lagrange multiplier values.                              */

if(verbosity > 0)
    fprintf(stdout,"\n\t%2.4e\t%2.4e\t%2.4e\t%2.4e\t%2.4e\n\n"
        ,lambda[0],lambda[1],lambda[2],lambda[3],lambda[4]);
if(verbosity > 1) gcheck(g,gcal,f0,dvparm);
if(verbosity > 0) fprintf(stdout,"\f");

/* Set up two graphics pages for mean and var of velocity.  */

pixinit();
if(num_page < 1){
    fprintf(stderr,"Can't open 2 graphics pages. Run aborted");
    exit(1);
    }
compare(lambda,tstar);

/* Evolve the distribution downstream by increasing xstar.  */

laminit(lambda,lamstore,vbar,dds,dstar);/* initialize lamstore */
varhigh = 0;
for(i=0;i<=I;i++){
    /* Calculate and store fd values for the distribution.  */
    fd[i]=me_fd(lambda,dstar[i],VSMAX);
    /* Find largest variance for setting graph limits.      */
    varhigh = max(varhigh,vbar[i][2]);
    }

/* Set high and low v limits for graphs and do set up.      */

vhigh = max(1.,u_air_star);
vlow = min(1.,u_air_star);
vel_graph_setup(vlow,vhigh,varhigh);

/* Draw initial state lines on graphs.                      */

pix_page(0);
for(i=0;i < I;i++)
    line(dstar[i],vbar[i][0],dstar[i+1],vbar[i+1][0]);
pix_page(1);
for(i=0;i < I;i++)
    line(dstar[i],vbar[i][2],dstar[i+1],vbar[i+1][2]);

lambda[3]=u_air_star;   /* Set the values of some fixed     */
lambda[4]=dratio;       /* arguments passed to functions    */
lambda[6]=dxstar;       /* in the evolution phase of the    */
lambda[7]=converge;     /* program. lambda[] no longer      */
lambda[8]=nu_star;      /* contains initial dist'n values.  */
```

```
/* Work downstream, increasing xstar by dxstar each time.    */

while(xstar <= xs_max){
    xstar+=dxstar;

    if(verbosity > 1) fprintf(stderr,
        "Calculating values for xstar = %8.1f \tmax = %8.1f\r",
        xstar,xs_max);  /* tracking to follow the program    */

    /* Plot each time xstar grows by a factor of 2.           */

    if(xstar >= 2.*last_plot_xs){
        plotflag = TRUE;
        last_plot_xs = xstar;
        }
    else plotflag = FALSE;

    /* Do accounting on whether to print out this iteration */

    prflag=FALSE;
    if(verbosity > 2 || (verbosity > 1 && count <= 0) || plotflag){
        if(xstar/dxstar >= 10*cycles) cycles*=2;
        count=cycles;
        prflag=TRUE;
        prcnt++;
        if(even(prcnt)) fprintf(stdout,"\f");   /* new page */
        fprintf(stdout,"\n\n\txstar = %6.2f",xstar);
        fprintf(stdout,
            "\tdelta, mean(v*), var(v*), check and bounds.\n\n");
        if(plotflag) fprintf(stdout,
            "\tcurve plotted for this set of data.\n\n");
        }

    /* Separately for each drop size increment.               */

    for(i=0;i<=I;i++){

        /* Load the lambdas for the velocity distribution.  */

        lambda[0]=lamstore[i][0];
        lambda[1]=lamstore[i][1];
        lambda[2]=lamstore[i][2];
        lambda[5]=dstar[i];
```

```
/* Do calculations based on a single valued velocity*/
/* if the variance has become small.                */

if(singlev[i] || sqrt(vbar[i][2]) < vbar[i][0]/100){
    vbar[i][2]=0;
    singlev[i]=TRUE;

    /* Check to see if droplet velocity has become  */
    /* equal to the air velocity.                   */

    if(equalv[i] ||
        fabs((u_air_star-vbar[i][0])/u_air_star)
        < converge){
        equalv[i] = TRUE;
        vbar[i][0] = u_air_star;
        }
    else vbar[i][0]=vnewl(vbar[i][0],lambda);

    if(prflag) fprintf(stdout,
        "\t%8.4f  %8.4f  %12.4e  %8.4f  %8.4f  %8.4f\n",
        dstar[i],vbar[i][0],vbar[i][2],0.,0.,0.);
    }

/* If the variance is still significant do integral */
/* calculations for new mean and variance. Integrate*/
/* over 3 standard deviations on each side of the   */
/* mean.                                             */

else{
    x=3*sqrt(vbar[i][2]);
    lower=max(vbar[i][0]-x,0.);
    upper=min(vbar[i][0]+x,VSMAX);
    range=upper-lower;

    /* Need to divide through by the integral of the    */
    /* PDF to avoid normalization problems with the     */
    /* truncated distribution.                          */

    x=simpint(pdfv,pdfvp,lower,upper,PANELS,lambda);
    vbar[i][0]=simpint(fvvn,fvvnp,lower,upper,
        PANELS,lambda)/x;
    vbar[i][1]=simpint(fvvn2,fvvn2p,lower,upper,
        PANELS,lambda)/x;
    vbar[i][2]=vbar[i][1]-vbar[i][0]*vbar[i][0];

    if(prflag) fprintf(stdout,
        "\t%8.4f  %8.4f  %12.4e  %8.4f  %8.4f  %8.4f\n",
        dstar[i],vbar[i][0],vbar[i][2],x,lower,upper);
    /* Use dtgsolve to get new distribution based on     */
    /* the new mean and variance.                        */

    lambda[1]=-vbar[i][0]/vbar[i][2];
    lambda[2]=.5/vbar[i][2];
    nlam[1]=(lambda[1]+2*lambda[2]*lower)*range;
    nlam[2]=lambda[2]*range*range;
    l=(vbar[i][0]-lower)/range;
    d2=vbar[i][2]/(range*range);
    dtgsolve(l,d2,con2,nlam);

    if(verbosity > 3)
        fprintf(stdout,
            "lambda 0-2 (geuss)\t%e\t%e\t%e\n",
            0.,lambda[1],lambda[2]);

    /* Store lambda values back for use in next round.   */

    lamstore[i][2]=lambda[2]=nlam[2]/(range*range);
    lamstore[i][1]=lambda[1]=nlam[1]/range-
        2*lambda[2]*lower;
    lamstore[i][0]=lambda[0]=nlam[0]+log(range)-
        lambda[1]*lower-lambda[2]*lower*lower;

    if(verbosity > 3)
        fprintf(stdout,"lambda 0-2\t\t%e\t%e\t%e\n",
            lambda[0],lambda[1],lambda[2]);
    }

if(verbosity > 4)
    fprintf(stdout,"check (1.0) =\t%e\n",x1);
}
```

```
    /* Plot lines on graphs.                                           */

    if(plotflag){
        pix_page(0);
        for(i=0;i < I;i++)
            line(dstar[i],vbar[i][0],dstar[i+1],vbar[i+1][0]);
        pix_page(1);
        for(i=0;i < I;i++)
            line(dstar[i],vbar[i][2],dstar[i+1],vbar[i+1][2]);
        }

    count--;
    }
fprintf(stdout,"\f");

/* Print out graphs.                                                   */
pix_page(0);
grfprt();
pix_page(1);
grfprt();
}
```

```
double data_in(infile)
    /* Takes a line from infile, trims leading alpha and space  */
    /* chars, takes chars up next space and returns a double    */
    /* equivalent. Returns zero on error                        */
    int infile;
    {
    extern double   atof();
    char line_in[81],*p,*q;
    if(!fgets(line_in,80,infile)) return 0.;
        /* point to first non alpha non space character         */
    for(p = line_in;isalpha(*p) || isspace(*p);p++);
        /* end string at next space character                   */
    for(q = p;!isspace(*q);q++);
    *q = 0;
        /* set double from string                               */
    return atof(p);
    }

prtheader(acc,i,dsmax,tstar,j,vsmax)
    /* Print header information */
    double acc,dsmax,tstar,vsmax;
    int i,j;
    {
    fprintf(stdout,
        "\tSolution of Entropy Maximization Equations\n\n");
    fprintf(stdout,"\tDrop Size Distributions\n\n");
    fprintf(stdout,"\tConstraints:\n");
    fprintf(stdout,"\t0 - normalization\n");
    fprintf(stdout,"\t1 - surface energy   d**2\n");
    fprintf(stdout,"\t2 - mass             d**3\n");
    fprintf(stdout,"\t3 - momentum         d**3 x v\n");
    fprintf(stdout,"\t4 - kinetic energy   d**3 x v**2\n\n");
    fprintf(stdout,
        "\tAcceleration = %2.2f\t\t\tI = %d\t\t DSMAX = %2.4f\n",
        acc,i,dsmax);
    fprintf(stdout,
        "\tTSTAR        = %2.4f\t\t\tJ = %d\t\t VSMAX = %2.4f\n\n",
        tstar,j,vsmax);
    fprintf(stdout,"\tlambda 0 to 4 =\n\n");
    }
```

```
parminit(dvparm,dstar,vstar)
    /* Initializes dvparm, dstar and vstar                          */
    double dvparm[IP][JP][NCONP],dstar[IP],vstar[JP];
    {
    double d,v,ix,jx,re;
    int i,j;
    for(i=0;i<=I;i++){
        ix=i;
        dstar[i]=(ix/IP+.5/IP)*DSMAX;
        }
    for(j=0;j<=J;j++){
        jx=j;
        vstar[j]=(jx/JP+.5/JP)*VSMAX;
        }
    for(i=0;i <= I;i++){
        for(j=0;j <= J;j++){
            dvparm[i][j][0]=1;
            dvparm[i][j][1]=dstar[i]*dstar[i];
            dvparm[i][j][2]=dvparm[i][j][1]*dstar[i];
            dvparm[i][j][3]=dvparm[i][j][2]*vstar[j];
            dvparm[i][j][4]=dvparm[i][j][3]*vstar[j];
            }
        }
    }
```

```
fsolve(lambda,dvparm,dstar,vstar,g,acc,converge)
    /* Solve for the lambda values subject to constraints g */
    double lambda[],dvparm[IP][JP][NCONP],dstar[],vstar[];
    double g[],acc,converge;
    {
    double sum[NCONP][NCONP],mat[NDIM][NDIM],b[NDIM],work[NDIM];
    double old,cond;
    int ipvt[NDIM],i,j,flag;
    flag=0;
    while(flag != 1){
        old=lambda[0];

        /* Calculate the Newton Rhapson equation matrix      */

        integrate(sum,lambda,dvparm,dstar,vstar);
        for(i=0;i<=NCON;i++){
            b[i]=acc*(g[i]-sum[i][0]);
            for(j=0;j<=NCON;j++){
                mat[i][j]=-sum[i][j];
                b[i]-=sum[i][j]*lambda[j];
                }
            }

        /* Solve the matrix for the new lambda values        */

        decomp(NCON,mat,cond,ipvt,work);
        solve(NCON,mat,b,ipvt);
        for(i=0;i<=NCON;i++) lambda[i]=b[i];

        /* Calculate new lambda[0] to ensure normalization   */

        lambda[0]=lamnot(dvparm,lambda,dstar,vstar);

        /* Test for convergence                              */

        if(fabs((old-lambda[0])/lambda[0])<converge) flag=1;
        if(verbosity > 2)
            fprintf(stdout,"\t%2.4e\t%2.4e\t%2.4e\t%2.4e\t%2.4e\n"
                ,lambda[0],lambda[1],lambda[2],lambda[3],lambda[4]);
        }
    }
```

```
integrate(sum,lambda,dvparm,dstar,vstar)
    /* Performs integrations for constrained quantities           */
    double sum[NCONP][NCONP],lambda[NCONP],dvparm[IP][JP][NCONP];
    double dstar[IP],vstar[JP];
    {
    double exponent,x,f;
    int i,j,k,m;

    for(i=0;i<=NCON;i++)
         for(j=0;j<=NCON;j++)
              sum[i][j]=0.;
    for(i=0;i<=I;i++){
         for(j=0;j<=J;j++){
              f=pdf(dstar,vstar,dvparm,lambda,i,j);
              for(k=0;k<=NCON;k++){
                   x=f*dvparm[i][j][k];
                   for(m=k;m<=NCON;m++) sum[k][m]+=x*dvparm[i][j][m];
                   }
              }
         }
    for(i=1;i<=NCON;i++)
         for(j=0;j<i;j++)
              sum[i][j]=sum[j][i];
    for(i=0;i<=NCON;i++)
         for(j=0;j<=NCON;j++)
              sum[i][j]*=dds*dvs;
    }

double lamnot(dvparm,lm,dstar,vstar)
    /* Returns lambda[0] for other lambda values based on
       normalization requirement                                */
    double dvparm[IP][JP][NCONP],lm[],dstar[],vstar[];
    {
    int i,j,k;
    double xsum,exponent;

    xsum=0.;
    for(i=0;i <= I;i++)
         for(j=0;j <= J;j++)
              xsum+=pdf(dstar,vstar,dvparm,lm,i,j);
    return log(xsum*dds*dvs*exp(lm[0]));
    }
```

```
decomp(n,a,cond,ipvt,work)
/*
    decomposes a matrix to estimate its condition

    n       order of the matrix - 1 (zero base matrix)
    a       matrix to be solved. triangularized matrix on output
    ipvt    ouput pivot vector for use by solve

    Converted from FORTRAN 84-03-11 by Rick Sellens
    Converted for a zero base matrix 84-04-19 by Rick Sellens
*/

    int n,ipvt[NDIM];
    double a[NDIM][NDIM],cond,work[NDIM];
{
    double fabs();
    double ek,t,anorm,ynorm,znorm;
    int nm1,i,j,k,kp1,kb,km1,m;
    ipvt[n]=1;
    if(n != 0){
        nm1=n-1;
        anorm=0.;
        for(j=0;j<=n;j++){
            t=0.;
            for(i=0;i<=n;i++) t=t+fabs(a[i][j]);
            if(t < anorm) anorm=t;
        }
        for(k=0;k<=nm1;k++){
            kp1=k+1;
            m=k;
            for(i=kp1;i<=n;i++){
                if(fabs(a[i][k]) > fabs(a[m][k])) m=i;
            }
            ipvt[k]=m;
            if(m != k) ipvt[n]=-ipvt[n];
            t=a[m][k];
            a[m][k]=a[k][k];
            a[k][k]=t;
            if(t != 0){
                for(i=kp1;i<=n;i++) a[i][k]=-a[i][k]/t;
                for(j=kp1;j<=n;j++){
                    t=a[m][j];
                    a[m][j]=a[k][j];
                    a[k][j]=t;
                    if(t != 0){
                        for(i=kp1;i<=n;i++) a[i][j]+=a[i][k]*t;
                    }
                }
            }
        }
        for(k=0;k<=n;k++){
            t=0.;
            if(k != 0){
                km1=k-1;
                for(i=0;i<=km1;i++) t+=a[i][k]*work[i];
            }
            ek=1.;
            if(t < 0.) ek=-1.;
            if(a[k][k] == 0.){
                cond=1.e32;
                goto exit;
            }
            work[k]=-(ek+t)/a[k][k];
        }
        for(kb=1;kb<=n;kb++){
            k=n-kb;
            t=0.;
            kp1=k+1;
            for(i=kp1;i<=n;i++) t+=a[i][k]*work[k];
            work[k]=t;
            m=ipvt[k];
            if(m != k){
                t=work[m];
                work[m]=work[k];
                work[k]=t;
            }
        }
        ynorm=0.;
        for(i=0;i<=n;i++) ynorm+=fabs(work[i]);
        solve(n,a,work,ipvt);
        znorm=0.;
        for(i=0;i<=n;i++) znorm+=fabs(work[i]);
        cond=anorm*znorm/ynorm;
        if(cond < 1.) cond=1.;
    }
    else{
        cond=1.;
        if(a[0][0] == 0.) cond=1.e32;
    }
    exit:return;
}
```

```
solve(n,a,b,ipvt)
/*
    solves the linear system a*x=b

    n       order of the matrix - 1 (zero base matrix)
    a       triangularized matrix output from decomp
    b       right hand side vector. solution on output
    ipvt    pivot vector output by decomp

    Converted from FORTRAN 84-03-11 by Rick Sellens
    Converted for zero base matrix 84-04-19 by Rick Sellens
*/

    int n,ipvt[NDIM];
    double a[NDIM][NDIM],b[NDIM];
{
    int kb,km1,nm1,kp1,i,k,m;
    double t;
    if(n != 0){
        nm1=n-1;
        for(k=0;k<=nm1;k++){
            kp1=k+1;
            m=ipvt[k];
            t=b[m];
            b[m]=b[k];
            b[k]=t;
            for(i=kp1;i<=n;i++) b[i]+=a[i][k]*t;
        }
        for(kb=1;kb<=n;kb++){
            km1=n-kb;
            k=km1+1;
            b[k]=b[k]/a[k][k];
            t=-b[k];
            for(i=0;i<=km1;i++) b[i]+=a[i][k]*t;
        }
    }
    b[0]=b[0]/a[0][0];
    return;
}
```

```
f0cal(ds,vs,parms,lm,f0)
    /* Calculate a matrix of PDF values and return in f0    */
    double ds[],vs[],parms[IP][JP][NCONP],lm[],f0[IP][JP];
    {
    int i,j;
    for(i=0;i<=I;i++)
        for(j=0;j<=J;j++)
            f0[i][j]=pdf(ds,vs,parms,lm,i,j);
    }

double pdf(dstar,vstar,dvparm,lambda,i,j)
    /* Returns the value of the PDF at a given i,j          */
    double dstar[],vstar[],dvparm[IP][JP][NCONP],lambda[];
    int i,j;
    {
    int k;
    double exponent,f;
    exponent=-lambda[0];
    for(k=1;k<=NCON;k++) exponent-=lambda[k]*dvparm[i][j][k];
    f=exp(exponent);

/* Only compile this line if the discretization is non uniform  */

#if USE_EXP
    f*=pow(dstar[i],DUNIEXP-1.)*pow(vstar[j],VUNIEXP-1.);
#endif

    return f;
    }
```

```
gcheck(g,gcal,f0,dvparm)
    /* Calculate expectation values based on calculated PDF
       values in f0 and print comparison figures.          */
    double g[],gcal[],f0[IP][JP],dvparm[IP][JP][NCONP];
    {
    int i,j,k;
    for(i=0;i<=NCON;gcal[i++]=0.);
    for(i=0;i<=I;i++)
        for(j=0;j<=J;j++)
            for(k=0;k<=NCON;k++)
                gcal[k]+=f0[i][j]*dvparm[i][j][k];
    for(i=0;i<=NCON;i++){
        gcal[i]*=DSMAX/IP;
        gcal[i]*=VSMAX/JP;
        }
    fprintf(stdout,
        "\tExpectations of constraints 0 to 4  (set/calculated)\n");
    fprintf(stdout,"\t%2.4e\t%2.4e\t%2.4e\t%2.4e\t%2.4e\n"
                ,g[0],g[1],g[2],g[3],g[4]);
    fprintf(stdout,"\t%2.4e\t%2.4e\t%2.4e\t%2.4e\t%2.4e\n"
                ,gcal[0],gcal[1],gcal[2],gcal[3],gcal[4]);
    }

compare(lambda,ts)
    /* Calculates and graphs comparisons between the ME     */
    /* distribution and the empirical distributions.        */
    double lambda[],ts;
    {
    double alpha,beta,gam,a,kappa,ds,fd[8],fdt[8];
    extern int square[],open_square[],one[],two[],three[],four[];
    extern int five[],six[],seven[],eight[],nine[],zero[];
    extern int dec[],neg[],plus[];

    /* Set parameter vector lambda for passing to functions.*/
    lambda[5] = VSMAX;
    lambda[6] = alpha=rr_alpha(ts);
    lambda[7] = beta=nt_beta(ts);
    lambda[8] = gam=lp_gamma(ts);
    lambda[10] = a=1.0;
    lambda[9] = kappa=ul_kappa(ts,a);
    lambda[11] = ul_dsm(a,kappa);

    fprintf(stdout,"\n\tAlpha = %2.4e\tBeta = %2.4e\t\tGamma = %2.4e\n"
          ,alpha,beta,gam);
    fprintf(stdout,"\tKappa = %2.4e\t   a = %2.4e\t\tdsm = %2.4e\n\n",
          kappa,a,lambda[11]);
    fprintf(stdout,"\tdstar    Max Ent      R.R.  ");
    fprintf(stdout,"      N.T.     Log Prob    Upper Limit\n\n");
fdt[0]=fdt[1]=fdt[2]=fdt[3]=fdt[4]=0;
for(ds=.05;ds<=DSMAX;ds+=.05){
    fdt[0]+=fd[0]=me_fdt(lambda,ds,VSMAX);
    fdt[1]+=fd[1]=rr_fd(alpha,ds);
    fdt[2]+=fd[2]=nt_fd(beta,ds);
    fdt[3]+=fd[3]=lp_fd(gam,ds);
    fdt[4]+=fd[4]=ul_fd(kappa,a,lambda[11],ds);
    fprintf(stdout,"\t%5.2f%12.4e%12.4e%12.4e%12.4e%12.4e\n",
            ds,fd[0],fd[1],fd[2],fd[3],fd[4]);
    }
fprintf(stdout,"\n\tTotal%12.4e%12.4e%12.4e%12.4e%12.4e\f",
        fdt[0],fdt[1],fdt[2],fdt[3],fdt[4]);

comp_graph_setup();
plot_fxp(plot_me_fdt,lambda,.01,2.5,0);
plot_fxp(plot_rr_fd,lambda,.01,2.5,0);
plot_fxp(plot_nt_fd,lambda,.01,2.5,0);
plot_fxp(plot_lp_fd,lambda,.01,2.5,0);
plot_fxp(plot_ul_fd,lambda,.01,.995*lambda[11],0);
grfprt();
comp_graph_setup();
plot_fxp(plot_me_fmt,lambda,.01,2.5,0);
plot_fxp(plot_rr_fm,lambda,.01,2.5,0);
plot_fxp(plot_nt_fm,lambda,.01,2.5,0);
plot_fxp(plot_lp_fm,lambda,.01,2.5,0);
plot_fxp(plot_ul_fm,lambda,.01,.995*lambda[11],0);
grfprt();
}
```

```
comp_graph_setup()
    /* Draws up axes and stuff for the plots                    */
    {
    extern int square[],open_square[],one[],two[],three[],four[];
    extern int five[],six[],seven[],eight[],nine[],zero[];
    extern int dec[],neg[],plus[];
    pix_page(0);
    pixclr();
    set_landscape();
    window(-.8,-.6,3.,2.7);
    axis(0.,2.5,.5,5,0.,2.2,.5,5);
    point(0.,-15.*ydot(),zero);
    point(1.-6*xdot(),-15.*ydot(),one);
    point(1.,-15.*ydot(),dec);
    point(1.+5*xdot(),-15.*ydot(),zero);
    point(2.-6*xdot(),-15.*ydot(),two);
    point(2.,-15.*ydot(),dec);
    point(2.+5*xdot(),-15.*ydot(),zero);
    point(-15.*xdot(),0.,zero);
    point(-25*xdot(),1.,one);
    point(-19*xdot(),1.,dec);
    point(-14*xdot(),1.,zero);
    point(-25*xdot(),2.,two);
    point(-19*xdot(),2.,dec);
    point(-14*xdot(),2.,zero);
    }

double me_fdt(lm,ds,vsm)
    /* returns fd based on an analytical integration method */
    double lm[],ds,vsm;
    {
    double x,x1,x2,exponent,ds3;
    ds3 = ipow(ds,3);
    x = sqrt(ds3/lm[4])*lm[3]/2;
    x1 = x+sqrt(lm[4]*ds3)*vsm;
    x2 = (erf(x1)-erf(x))*sqrt(PI/(4*lm[4]*ds3));
    exponent = -lm[0]-lm[1]*ds*ds-(lm[2]-lm[3]*lm[3]/(4*lm[4]))*ds3;
    return x2*exp(exponent);
    }
```

```
laminit(lambda,lamstore,vbar,dds,dstar)
    /* Initializes lamstore based on lambda 0 - 4 from  */
    /* initial distribution. Also calculates vbar by    */
    /* integration.                                     */
    double lambda[],lamstore[IP][3],vbar[IP][3],dds,dstar[];
    {
    int i;
    double x,d3,dumlam[3];
    if(verbosity > 0) fprintf(stdout,
        "\n\tInitial Conditions\tdelta, mean v* and variance.\n\n");
    for(i=0;i<=I;i++){
        d3=ipow(dstar[i],3);
        dumlam[0]=lamstore[i][0]=lambda[0]+lambda[1]*dstar[i]
            *dstar[i]+lambda[2]*d3+log(me_fd(lambda,dstar[i],VSMAX));
        dumlam[1]=lamstore[i][1]=lambda[3]*d3;
        dumlam[2]=lamstore[i][2]=lambda[4]*d3;
        vbar[i][0]=simpint(fvv,fvvp,0.,VSMAX,XPANELS,dumlam);
        vbar[i][1]=simpint(fvv2,fvv2p,0.,VSMAX,XPANELS,dumlam);
        vbar[i][2]=vbar[i][1]-vbar[i][0]*vbar[i][0];
        if(verbosity > 0) fprintf(stdout,
            "\t%8.4f  %8.4f  %12.4e\n",
            dstar[i],vbar[i][0],vbar[i][2]);
        }
    }

vel_graph_setup(vlow,vhigh,varhigh)
    /* Draws up axes and stuff for the velocity plots          */
    double vlow,vhigh,varhigh;
    {
    extern int square[],open_square[],one[],two[],three[],four[];
    extern int five[],six[],seven[],eight[],nine[],zero[];
    extern int dec[],neg[],plus[];

    /* set up mean velocity graph on page one       */

    pix_page(0);
    pixclr();
    set_landscape();
    vlow-=.1;
    vhigh+=.1;
    window(-.7,vlow-(vhigh-vlow)/3.,3.,vhigh+(vhigh-vlow)/3.);
    axis(0.,2.5,.5,5,vlow,vhigh,.1,2);
```

```
point(0.,vlow-15.*ydot(),zero);
point(1.-6*xdot(),vlow-15.*ydot(),one);
point(1.,vlow-15.*ydot(),dec);
point(1.+5*xdot(),vlow-15.*ydot(),zero);
point(2.-6*xdot(),vlow-15.*ydot(),two);
point(2.,vlow-15.*ydot(),dec);
point(2.+5*xdot(),vlow-15.*ydot(),zero);
if(vlow <= 0. && vhigh >= 0.) point(-15.*xdot(),0.,zero);
if(vlow <= .2 && vhigh >= .2){
    point(-25*xdot(),.2,zero);
    point(-19*xdot(),.2,dec);
    point(-14*xdot(),.2,two);
    }
if(vlow <= .4 && vhigh >= .4){
    point(-25*xdot(),.4,zero);
    point(-19*xdot(),.4,dec);
    point(-14*xdot(),.4,four);
    }
if(vlow <= .6 && vhigh >= .6){
    point(-25*xdot(),.6,zero);
    point(-19*xdot(),.6,dec);
    point(-14*xdot(),.6,six);
    }
if(vlow <= .8 && vhigh >= .8){
    point(-25*xdot(),.8,zero);
    point(-19*xdot(),.8,dec);
    point(-14*xdot(),.8,eight);
    }
if(vlow <= 1. && vhigh >= 1.){
    point(-25*xdot(),1.,one);
    point(-19*xdot(),1.,dec);
    point(-14*xdot(),1.,zero);
    }
if(vlow <= 1.2 && vhigh >= 1.2){
    point(-25*xdot(),1.2,one);
    point(-19*xdot(),1.2,dec);
    point(-14*xdot(),1.2,two);
    }
if(vlow <= 1.4 && vhigh >= 1.4){
    point(-25*xdot(),1.4,one);
    point(-19*xdot(),1.4,dec);
    point(-14*xdot(),1.4,four);
    }
```

```
if(vlow <= 1.6 && vhigh >= 1.6){
    point(-25*xdot(),1.6,one);
    point(-19*xdot(),1.6,dec);
    point(-14*xdot(),1.6,six);
    }
if(vlow <= 1.8 && vhigh >= 1.8){
    point(-25*xdot(),1.8,one);
    point(-19*xdot(),1.8,dec);
    point(-14*xdot(),1.8,eight);
    }
if(vlow <= 2. && vhigh >= 2.){
    point(-25*xdot(),2.,two);
    point(-19*xdot(),2.,dec);
    point(-14*xdot(),2.,zero);
    }

/* set up velocity variance graph on page one        */

pix_page(1);
pixclr();
set_landscape();
window(-.7,0.-varhigh/3.,3.,varhigh*1.333);
varhigh*=1.1;
axis(0.,2.5,.5,5,0.,varhigh,.1,2);
point(0.,-15.*ydot(),zero);
point(1.-6*xdot(),-15.*ydot(),one);
point(1.,-15.*ydot(),dec);
point(1.+5*xdot(),-15.*ydot(),zero);
point(2.-6*xdot(),-15.*ydot(),two);
point(2.,-15.*ydot(),dec);
point(2.+5*xdot(),-15.*ydot(),zero);
point(-15.*xdot(),0.,zero);
if(varhigh >= .1){
    point(-25*xdot(),.1,zero);
    point(-19*xdot(),.1,dec);
    point(-14*xdot(),.1,one);
    }
if(varhigh >= .2){
    point(-25*xdot(),.2,zero);
    point(-19*xdot(),.2,dec);
    point(-14*xdot(),.2,two);
    }
if(varhigh >= .3){
    point(-25*xdot(),.3,zero);
    point(-19*xdot(),.3,dec);
    point(-14*xdot(),.3,three);
    }
}
```

```
double fvv(vs,lambda)
    /* Return the product of fv and vs based on the values  */
    /* in the vector of doubles, lambda[].                   */
    double vs,lambda[];
    {
    return vs*pdfv(vs,lambda);
    }

double fvvp(vs,lambda)
    /* Return d(fv*vs)/dvstar (fv*vs prime) based on the     */
    /* values in the vector of doubles, lambda[].            */
    double vs,lambda[];
    {
    return vs*pdfvp(vs,lambda)+pdfv(vs,lambda);
    }

double fvv2(vs,lambda)
    /* Return the product of fv and vs^2 based on the        */
    /* values in the vector of doubles, lambda[].            */
    double vs,lambda[];
    {
    return vs*vs*pdfv(vs,lambda);
    }

double fvv2p(vs,lambda)
    /* Return d(fv*vs^2)/dvstar (fv*vs^2 prime) based on     */
    /* the values in the vector of doubles, lambda[].        */
    double vs,lambda[];
    {
    double x;
    return vs*vs*pdfvp(vs,lambda)+2*pdfv(vs,lambda)*vs;
    }

double pdfv(vs,lambda)
    /* Return the value of fv(vs) based on the values        */
    /* in the vector of doubles, lambda[].                   */
    double vs,lambda[];
    {
    return exp(-lambda[0]-lambda[1]*vs-lambda[2]*vs*vs);
    }

double pdfvp(vs,lambda)
    /* Return d(fv)/dvstar (fv prime) based on the values    */
    /* in the vector of doubles, lambda[].                   */
    double vs,lambda[];
    {
    return -(lambda[1]+2*lambda[2]*vs)*pdfv(vs,lambda);
    }
```

```
double vnew(v0,uinf,dratio,delta,dx,converge,nu,cd)
    /* Returns the velocity of a droplet at dx downstream based
       on the initial velocity and simple drag relations.
       uinf must be a value greater than zero.
       Will correctly return v1 = v0 for v0 == uinf.
       Tests correct    84-09-07    RWS             */
    double v0,uinf,dratio,delta,dx,converge,nu,(*cd)();
    {
    double x,k,vold=0,v1,vbar,ur;
    x=1.5*dratio*dx*(uinf-v0)/delta;
    v1=v0+0.0000001;    /* to make sure vnew != 0   */
    if(verbosity > 5){
        fprintf(stdout,"\tIn vnew(), v0 = %e \n",v0);
        fprintf(stdout,"\tuinf, dratio, delta, dx, nu\n");
        fprintf(stdout,"\t%e  %e  %e  %e  %e\n",
            uinf,dratio,delta,dx,nu);
        }
    while(fabs((v1-vold)/v1)>converge){
        vbar=(v1+v0)/2;
        vold=v1;
        ur=uinf-vbar;
        k=x*(*cd)(ur*delta/nu);
        v1=-k/2+sqrt(k*k/4+v0*v0+k*uinf);
        if(verbosity > 5)
            fprintf(stdout,"\tvbar = %e \t v1 = %e\n",vbar,v1);
        }
    if(verbosity > 5) fprintf(stdout,"\tv1 = %e\n\n",v1);
    return v1;
    }

double vnewl(vs,lambda)
    /* Return the new value of vstar based on the values    */
    /* in the vector of doubles, lambda[].                  */
    /*      lambda[0] = lambda0   - multipliers for the     */
    /*      lambda[1] = lambda1     doubly truncated        */
    /*      lambda[2] = lambda2     gaussian distribution.  */
    /*      lambda[3] = uastar    - air velocity            */
    /*      lambda[4] = dratio    - density ratio (air/liq) */
    /*      lambda[5] = dstar     - drop size               */
    /*      lambda[6] = dxstar    - x increment             */
    /*      lambda[7] = converge  - convergence test value  */
    /*      lambda[8] = nustar    - air viscosity           */
    double vs,lambda[];
    {
    return vnew(vs,lambda[3],lambda[4],lambda[5],lambda[6],
        lambda[7],lambda[8],cdwhite);
    }
```

```
double vnewlp(vs,lambda)
    /* Return d(vnewl)/dvstar (vnewl prime) based on the     */
    /* values in the vector of doubles, lambda[].            */
    double vs,lambda[];
    {
    double k2,x,vl,vb,ur0;
    vl=vnewl(vs,lambda);
    x=1.5*lambda[4]*lambda[6]*(lambda[3]-vs)/lambda[5];
    vb=(vl+vs)/2;
    ur0=lambda[3]-vs;
    /* deal with case where relative velocity is about zero */
    if(fabs(ur0) < 1.E-10) return 0.;
    k2=x*cdwhite((lambda[3]-vb)*lambda[5]/lambda[8])/2;
    return (vs+k2)*lambda[3]/((vl+k2)*ur0)-vl/ur0;
    }

double cdwhite(re)
    /* returns drag coefficient for a sphere 0 <= re <= 2e5     */
    /* white - viscous fluid flow - eqn 3-265                   */
    /* Tests correct    84-09-07    RWS                         */
    double re;
    {
    double cd,sign=-1.;
    if(re<0) re*=sign;
        else sign=1.;
    if(re<.0001) re=.0001;  /* to eliminate problems with re = 0*/
    cd=sign*(24./re+6./(1+sqrt(re))+0.4);
    if(verbosity > 5) fprintf(stdout,"\tre = %e \t cd = %e\n",re,cd);
    return cd;
    }

double fvvn(vs,lambda)
    /* Return the product of fv and vn based on the values  */
    /* in the vector of doubles, lambda[].                   */
    double vs,lambda[];
    {
    return vnewl(vs,lambda)*pdfv(vs,lambda);
    }

double fvvnp(vs,lambda)
    /* Return d(fv*vn)/dvstar (fv*vn prime) based on the     */
    /* values in the vector of doubles, lambda[].            */
    double vs,lambda[];
    {
    return vnewl(vs,lambda)*pdfvp(vs,lambda)+
        pdfv(vs,lambda)*vnewlp(vs,lambda);
    }
```

```
double fvvn2(vs,lambda)
    /* Return the product of fv and vn^2 based on the       */
    /* values in the vector of doubles, lambda[].           */
    double vs,lambda[];
    {
    return ipow(vnewl(vs,lambda),2)*pdfv(vs,lambda);
    }

double fvvn2p(vs,lambda)
    /* Return d(fv*vn^2)/dvstar (fv*vn^2 prime) based on    */
    /* the values in the vector of doubles, lambda[].       */
    double vs,lambda[];
    {
    double x;
    x=vnewl(vs,lambda);
    return x*x*pdfvp(vs,lambda)+
        2*pdfv(vs,lambda)*x*vnewlp(vs,lambda);
    }

dtgsolve(l,d2,con,lambda)
    /*  Solves for lambda values for a doubly truncated     */
    /*  Gaussian distribution with limits x = 0 to 1 and    */
    /*  mean = l, variance = d2.                            */
    /*  con is a convergence criterion.                     */
    /*  Method from Tribus, Rational Descriptions,          */
    /*  Decisions and Designs. pp 141 - 143.                */
    double l,d2,con,lambda[];
    {
    double old,x,x1,x2,f,f1,f2,g,g1,g2,d0,phi[5];
    if(verbosity > 4){
        fprintf(stdout,"L = %e\t D^2 = %e\n",l,d2);
        fprintf(stdout,"In dtgsolve, phi[0], lambda 0-2 =\n");
        }
    if(lambda[1] == 0) lambda[1]=.1;
    x2=l*l+d2;
    old=lambda[1]*2;
    while(fabs((old-lambda[1])/lambda[1]) > con){
        old=lambda[1];
        phi[0]=phinot(lambda);
        if(verbosity > 4)
            fprintf(stdout,"\tphi0 lm0-2 %12.4e%12.4e%12.4e%12.4e\n",
                phi[0],lambda[0],lambda[1],lambda[2]);
```

```
          x=-exp(-lambda[1]-lambda[2]);
          x1=1/(2*lambda[2]);
          phi[1]=(1+x-lambda[1]*phi[0])*x1;
          phi[2]=(x+phi[0]-lambda[1]*phi[1])*x1;
          phi[3]=(x+2*phi[1]-lambda[1]*phi[2])*x1;
          phi[4]=(x+3*phi[2]-lambda[1]*phi[3])*x1;
          f=1*phi[0]-phi[1];
          g=x2*phi[0]-phi[2];
          f1=-1*phi[1]+phi[2];
          f2=-1*phi[2]+phi[3];
          g1=-x2*phi[1]+phi[3];
          g2=-x2*phi[2]+phi[4];
          d0=f1*g2-f2*g1;
          lambda[1]-=(g2*f-f2*g)/d0;
          lambda[2]+=(g1*f-f1*g)/d0;
          }
     lambda[0]=log(phinot(lambda));
     }

double phinot(lambda)
     /*  Evaluates phi[0] as described in Tribus          */
     /*  phi[0] = integral(0 to 1)                        */
     /*           exp(-lambda[1]*x-lambda[2]*x^2) dx      */
     double lambda[];
     {
     return simpint(dtgf,dtgfp,0.,1.,PANELS,lambda);
     }

double dtgf(x,lambda)
     /* function f for phinot() and dtgsolve()           */
     /* f(x,lambda) = exp(-lambda[1]*x-lambda[2]*x^2)    */
     double x,lambda[];
     {
     return exp(-lambda[1]*x-lambda[2]*x*x);
     }

double dtgfp(x,lambda)
     /* function fp (df/dx) for phinot() and dtgsolve()  */
     /* fp(x,lambda) = (-lambda[1]-2*lambda[2]*x)*       */
     /*                  exp(-lambda[1]*x-lambda[2]*x^2) */
     double x,lambda[];
     {
     return (-lambda[1]-2*lambda[2]*x)*
            exp(-lambda[1]*x-lambda[2]*x*x);
     }
```

```
/* The following functions simply switch between a call      */
/* using the lambda parameter vector and discrete            */
/* parameters.                                                */

double plot_me_fdt(ds,lambda)
    double ds,lambda[];
    {
    return me_fdt(lambda,ds,lambda[5]);
    }

double plot_rr_fd(ds,lambda)
    double ds,lambda[];
    {
    return rr_fd(lambda[6],ds);
    }

double plot_nt_fd(ds,lambda)
    double ds,lambda[];
    {
    return nt_fd(lambda[7],ds);
    }

double plot_lp_fd(ds,lambda)
    double ds,lambda[];
    {
    return lp_fd(lambda[8],ds);
    }

double plot_ul_fd(ds,lambda)
    double ds,lambda[];
    {
    return ul_fd(lambda[9],lambda[10],lambda[11],ds);
    }

double plot_me_fmt(ds,lambda)
    double ds,lambda[];
    {
    return ds*ds*ds*me_fdt(lambda,ds,lambda[5]);
    }

double plot_rr_fm(ds,lambda)
    double ds,lambda[];
    {
    return ds*ds*ds*rr_fd(lambda[6],ds);
    }
```

```
double plot_nt_fm(ds,lambda)
    double ds,lambda[];
    {
    return ds*ds*ds*nt_fd(lambda[7],ds);
    }

double plot_lp_fm(ds,lambda)
    double ds,lambda[];
    {
    return ds*ds*ds*lp_fd(lambda[8],ds);
    }

double plot_ul_fm(ds,lambda)
    double ds,lambda[];
    {
    return ul_fm(lambda[9],lambda[10],lambda[11],ds);
    }
```

```
/******* These functions are from the library drops.s *********/

#include "stdio.h"
#include "math.h"
#include "mathfns.h"

double me_fl(),me_flp(),me_fp(),me_f();

double me_fl(vs,lm)
     double vs,lm[];
     {
     return me_f(lm,lm[5],vs);
     }

double me_flp(vs,lm)
     double vs,lm[];
     {
     return me_fp(lm,lm[5],vs);
     }

double me_fp(lm,ds,vs)
     double lm[],ds,vs;
     {
     double d;
     d=ipow(ds,3);
     return me_f(lm,ds,vs)*(-2*lm[4]*d*vs-lm[3]*d);
     }

double me_f(lm,ds,vs)
     double lm[],ds,vs;
     {
     double exponent,d;
     exponent=-lm[0];
     d=ds*ds;
     exponent-=lm[1]*d;
     d*=ds;
     exponent-=lm[2]*d+lm[3]*d*vs+lm[4]*d*vs*vs;
     return exp(exponent);
     }

double me_fd(lm,ds,vsm)
     double lm[],ds,vsm;
     {
     lm[5]=ds;
     return simpint(me_fl,me_flp,0.,vsm,50,lm);
     }
```

```
double rr_alpha(ts) /* returns rosin rammler exponent alpha     */
    double ts;      /* works for tstar values between .334 & .42*/
    {
    double tt,t,alpha;
    tt=ts/2;
    alpha=4.;
    t=rr_tstar(alpha);
    while(fabs(t-ts)/ts > 0.00000001){
        alpha*=exp((t-ts)/tt);
        t=rr_tstar(alpha);
        }
    return alpha;
    }

double rr_tstar(alpha)  /* returns tstar given r-r parameter alpha */
    double alpha;
    {
    double t,u;
    t=pow(gamma(1-3/alpha),0.333333)/(3*gamma(1-1/alpha));
    return t;
    }

double rr_fd(alpha,ds)
    double alpha,ds;
    {
    double r;
    r=1./pow(gamma(1-3/alpha),alpha/3);
    return alpha*r*pow(ds,alpha-4.)*exp(-r*pow(ds,alpha));
    }
```

```
double nt_beta(ts)  /* returns nukiyama tanasawa exponent beta  */
    double ts;      /* works for tstar values between .342 & .42*/
    {
    double tt,t,beta;
    tt=ts*ts*ts;
    beta=3.;
    t=nt_tstar(beta);
    while(fabs(t-ts)/ts > 0.00000001){
        beta*=exp((t-ts)/tt);
        t=nt_tstar(beta);
        }
    return beta;
    }

double nt_tstar(beta)   /* returns tstar given n-t parameter beta */
    double beta;
    {
    double t;
    t=pow(gamma(6./beta),2/3.);
    t*=pow(gamma(3./beta),1/3.);
    t/=gamma(5./beta)*3;
    return t;
    }

double nt_fd(beta,ds)
    double beta,ds;
    {
    double exponent,arg,a,b;
    exponent=-pow(gamma(6/beta)/gamma(3/beta),beta/3)*pow(ds,beta);
    arg=beta*gamma(6/beta)*ds*ds/pow(gamma(3/beta),2.);
    return arg*exp(exponent);
    }
```

```
double lp_gamma(ts) /* returns log probability exponent gamma   */
    double ts;      /* works for tstar values between .334 & .42*/
    {
    return pow(2*log(3*ts),-0.5);
    }

double lp_fd(gam,ds)
    double gam,ds;
    {
    double exponent;
    exponent=-pow(fabs(gam*log(ds)+3./(4.*gam)),2.);
    return gam/(ds*sqrt(PI))*exp(exponent);
    }
```

```
double ul_Kappa(ts,a)   /* returns upper limit exponent Kappa   */
    double ts,a;    /* works for tstar values between .334 & .42*/
    {
    double tt,t,kappa;
    tt=ts;
    if(ts<.39) tt*=ts;
    if(ts<.36) tt*=ts;
    Kappa=4.;
    t=ul_tstar(Kappa,a);
    while(fabs(t-ts)/ts > 0.00000001){
        Kappa*=exp((t-ts)/tt);
        t=ul_tstar(Kappa,a);
        }
    return Kappa;
    }

double ul_tstar(Kappa,a)    /* returns tstar given u l Kappa and a */
    double Kappa,a;
    {
    double t,K2;
    K2=1/Kappa/Kappa;
    t=1+3*a*exp(.25*K2)+3*a*a*exp(K2)+a*a*a*exp(2.25*K2);
    t=pow(t,0.333333);
    t/=3*(1+a*exp(.25*K2));
    return t;
    }

double ul_dsm(a,Kappa)
    /* Returns the dstar max value for the upper limit dist.    */
    /* based on the values of parameters a and Kappa.           */
    double a,Kappa;
    {
    double x,K2;
    K2 = Kappa*Kappa;
    x = 1+3*a*exp(.25/K2)+3*a*a*exp(1./K2)+a*a*a*exp(2.25/K2);
    return pow(x,.33333);
    }
```

```
double ul_fm(Kappa,a,dsm,ds)
    /* Returns the value of the mass based PDF for the upper     */
    /* limit distribution based on Kappa, a, dsm and ds.         */
    double Kappa,a,dsm,ds;
    {
    double x;
    if(ds >= dsm)
        return 0.;
    x = Kappa*log(a*ds/(dsm-ds));
    return (1./ds+1./(dsm-ds))*Kappa/sqrt(PI)*exp(-x*x);
    }

double ul_fd(Kappa,a,dsm,ds)
    /* Returns the value of the number based PDF for the upper   */
    /* limit distribution based on Kappa, a, dsm and ds.         */
    double Kappa.a,dsm,ds;
    {
    return ul_fm(Kappa,a,dsm,ds)/(ds*ds*ds);
    }
```

```
/******** These functions are from the library mathfns.s ************/

double erf(x)        /*  Returns the error function of x */
     double x;
     {
     double x22,num,den,sum,old;
     int n;

     if(x>5.) return 1.;
     if(x<0.) return -erf(-x);
     x22=x*x*2.;
     num=x;
     den=1.;
     sum=x;
     old=0;
     n=3;
     while(old != sum){
          old=sum;
          num*=x22;
          den*=n;
          sum+=num/den;
          n++;
          n++;
     }
     sum*=2./sqrt(3.141592654)*exp(-x*x);
     return sum;
     }

double gamma(z)      /* Returns the gamma function of z            */
     double z;       /* |error|<=3e-7  Abr. & Stegun eq. 6.1.35   */
     {
     double x,g,xs,xr,b[9];
     int n;
     if(z<=0 && dint(z)==z) return 1.0e300;
     xr=1;
     while(z>2) xr*=--z; /* bring into range 1<=z<=2 using   */
     while(z<1) xr/=z++; /* formulas for gamma(z+-1)         */
     b[1]=-0.577191652;
     b[2]=0.988205891;
     b[3]=-0.897056937;
     b[4]=0.918206857;
     b[5]=-0.756704078;
     b[6]=0.482199394;
     b[7]=-0.193527818;
     b[8]=0.035868343;
     g=xs=1;
     x=z-1;
     for(n=1;n<=8;n++){
          xs*=x;
          g+=b[n]*xs;
          }
     return g*xr;
     }
```

```
dint(z)          /* Returns the truncated integer value of z */
    double z;
    {
    int i;
    i=z;
    return i;
    }

double simpint(f,fp,a,b,n,lambda)

    /* Does integration by Simpson's Rule with end correction   */
    /* on f(x,lambda) between x=a and x=b using n (or the next  */
    /* largest even integer) number of panels.                  */
    /* fp(x,lambda) = df/dx                                     */
    /* See Hornbeck - Numerical Methods - Eqn. 8.32             */
    /* Rick Sellens 84-08-27                                    */

    double a,b,(*f)(),(*fp)();
    double lambda[];    /* a vector to pass on to f() and fp()  */
    int n;
    {
    double sum1,sum2,dx,x,tot;
    double sign = 1.;
    int i,j;

    if(a > b){      /* make sure a <= b                         */
        x=a;
        a=b;
        b=x;
        sign=-1;    /* if switch then return negative value     */
        }
    if(a == b) return 0.;   /* integral = 0 for zero interval   */
    sum1=sum2=0;
    i=n/2;          /* round n up to an even integer            */
    if(2*i!=n) n++;
    dx=(b-a)/n;
    x=a+dx;
    for(j=1;j<(n-2);j+=2){
        sum1+=(*f)(x,lambda);   /* internal odd point values    */
        x+=dx;
        sum2+=(*f)(x,lambda);   /* internal even point values   */
        x+=dx;
        }
    sum1+=(*f)(x,lambda);
    tot=( 14.*( ((*f)(a,lambda)+(*f)(b,lambda))/2.+sum2 )+
          16.*sum1+dx*((*fp)(a,lambda)-(*fp)(b,lambda)) )*dx/15.;
    return sign*tot;
    }
```