Naval Medical Research Institute
8901 Wisconsin Avenue
Bethesda, MD 20889-5607

NMRI 94-36 August 1994

A MODEL OF BUBBLE EVOLUTION DURING DECOMPRESSION BASED ON A MONTE CARLO SIMULATION OF INERT GAS DIFFUSION

R. Ball
J. Himm
L. D. Homer
E. D. Thalmann

Naval Medical Research
and Development Command
Bethesda, Maryland 20889-5606

Department of the Navy
Naval Medical Command
Washington, DC 20372-5210

NOTICES

The opinions and assertions contained herein are the private ones of the writer and are not to be construed as official or reflecting the views of the naval service at large.

When U. S. Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from the Naval Medical Research Institute. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, Virginia 22161

Federal Government agencies and their contractors registered with the Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
Cameron Station
Alexandria, Virginia 22304-6145

TECHNICAL REVIEW AND APPROVAL

NMRI 94-36

The experiments reported herein were conducted according to the principles set forth in the current edition of the "Guide for the Care and Use of Laboratory Animals," Institute of Laboratory Animal Resources, National Research Council.

This technical report has been reviewed by the NMRI scientific and public affairs staff and is approved for publication. It is releasable to the National Technical Information Service where it will be available to the general public, including foreign nations.

ROBERT G. WALTER
CAPT, DC, USN
Commanding Officer
Naval Medical Research Institute

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1994	TECHNICAL 6/92-6/93

4. TITLE AND SUBTITLE

A model of bubble evolution during decompression based on a Monte Carlo simulation of inert gas diffusion.

5. FUNDING NUMBERS

PE - 62233N
PR - MM33P30
TA - .004
WU - 1050

6. AUTHOR(S)

Ball, R., J. Himm, L.D. Homer, and E.D. Thalmann

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Medical Research Institute
Commanding Officer
8901 Wisconsin Avenue
Bethesda, Maryland 20889-5607

8. PERFORMING ORGANIZATION REPORT NUMBER

NMRI 94-36

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Medical Research and Development Command
National Naval Medical Center
Building 1, Tower 12
8901 Wisconsin Avenue
Bethesda, Maryland 20889-5606

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

DN249500

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Previously, a Monte Carlo simulation of inert gas diffusion in a capillary bed had been developed at this laboratory to explore the effect of tissue heterogeneity and microvascular architecture on gas exchange under normobaric conditions. Because we needed a method of looking at gas phase dynamics during decompression in this environment, the Monte Carlo method was extended to simulate bubble growth and dissolution during decompression. The essence of our approach involves the placement of inert gas particles in a bubble-liquid module and simulating diffusion with random displacements within the module for a short, fixed time period. At the end of the time period the distribution of the particles is used to calculate the number of moles of gas inside the bubble. The new bubble volume is then calculated from the ideal gas law. We developed methods to speed up the simulation by computing distributions of displacements following many random steps so that the simulation of many steps might be made with a simple calculation. In addition, we can calculate the amount of time a particle will stay inside the bubble based on the solubility of the inert gas. We demonstrate that a bubble evolves to the expected equilibrium size and the time course of the evolution compares favorably with that predicted by a partial differential equation model. A Monte Carlo approach is successful in simulating bubble evolution during decompression and is potentially suitable for studying the influence of tissue micro-architecture on gas phase dynamics.

14. SUBJECT TERMS	15. NUMBER OF PAGES
decompression, bubble, model, monte carlo, stochastic, diffusion, markov process, diving, gas exchange, kinetics, dynamics, supercomputer, parallel processor, partial differential equation, probability distribution, simulation	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

TABLE OF CONTENTS

page

LIST OF APPENDICES

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

This work was supported by NMRDC work unit No. 62233N MM33P30.004-1050. We are grateful to Paul Massell for his mathematical insights, as well as Susan Mannix for her editorial assistance.

Computer time provided by the National Cancer Institute's biomedical supercomputer center, Frederick, MD; Uniformed Services University of the Health Sciences, Bethesda, MD; the Naval Health Research Center, San Diego, CA; and by several departments of the Naval Medical Research Institute.

The opinions expressed in this paper are those of the authors and do not reflect the official policy or position of the Department of Navy, Department of Defense, or the U.S. Government.

BACKGROUND

In developing a probabilistic decompression model it was found that asymmetrical gas kinetics proved to fit the available human exposure data much better than symmetrical exponential kinetics (1-3). This asymmetrical model used exponential kinetics for gas uptake, but when tissue gas tension exceeded ambient pressure by a specified amount the kinetics became linear, dramatically slowing the rate of gas elimination. This so-called linear exponential (LE) model was initially developed for computing fixed oxygen partial pressure decompression tables (4,5), but had been extended to air in some exploratory trials (6).

The success of the LE model led us to seek a physiologic rationale that might explain its performance. One rationale was that the change in kinetics described slowing of gas elimination as a result of gas phase formation. This hypothesis is supported qualitatively by observations that the slowing of inert gas elimination during decompression (7-9). Could the slow pace of gas elimination implied by the success of the LE model be confirmed by a physiologically and physically plausible approximation to gas elimination in the presence of bubbles?

To answer this question we wanted to model gas phase growth and dissolution in heterogeneous architectures representative of actual tissue. Previously, a Monte Carlo simulation of diffusion had been developed in a tissue model to explore the effect of tissue heterogeneity and microvascular architecture on gas exchange (with no gas phase present) under normobaric conditions (10-12). As a first step towards modeling gas phase dynamics in that environment, we have developed a model of bubble evolution during decompression in a homogeneous and uniformly perfused tissue by extending the Monte Carlo methods used in

previous simulations. Using a homogenous and uniformly perfused tissue allows for comparisons using well established methods of modeling bubble evolution (13). However, in principle the Monte Carlo approach should allow modeling in architectures of any complexity. This report presents the methods employed and their derivations, results of tests examining the model's ability to predict bubble radii under equilibrium conditions, a comparison of the time course of bubble evolution predicted by the Monte Carlo model with that predicted from the numerical solution of a partial differential equation model, and the model's performance at different levels of precision. Application of the model in reports to follow will help us see if gas elimination can be delayed as much as predicted by the LE model.

METHODS

Overview

This section provides a brief description of the bubble-liquid module, the Monte Carlo method, initial conditions, and the simulation structure. Details regarding the derivation and implementation of the techniques described are presented in the sections that follow.

As a first approximation of a physiological model of bubble evolution in tissue, we developed a model of a single spherical bubble evolving in a homogenous liquid of fixed and finite volume. The volume of the liquid surrounding a single bubble can be interpreted to be the inverse of the bubble density (number of bubbles per volume of liquid). A tissue might consist of many such bubbles, each associated with its own liquid volume as shown in Figure 1. The system is open to mass transfer.

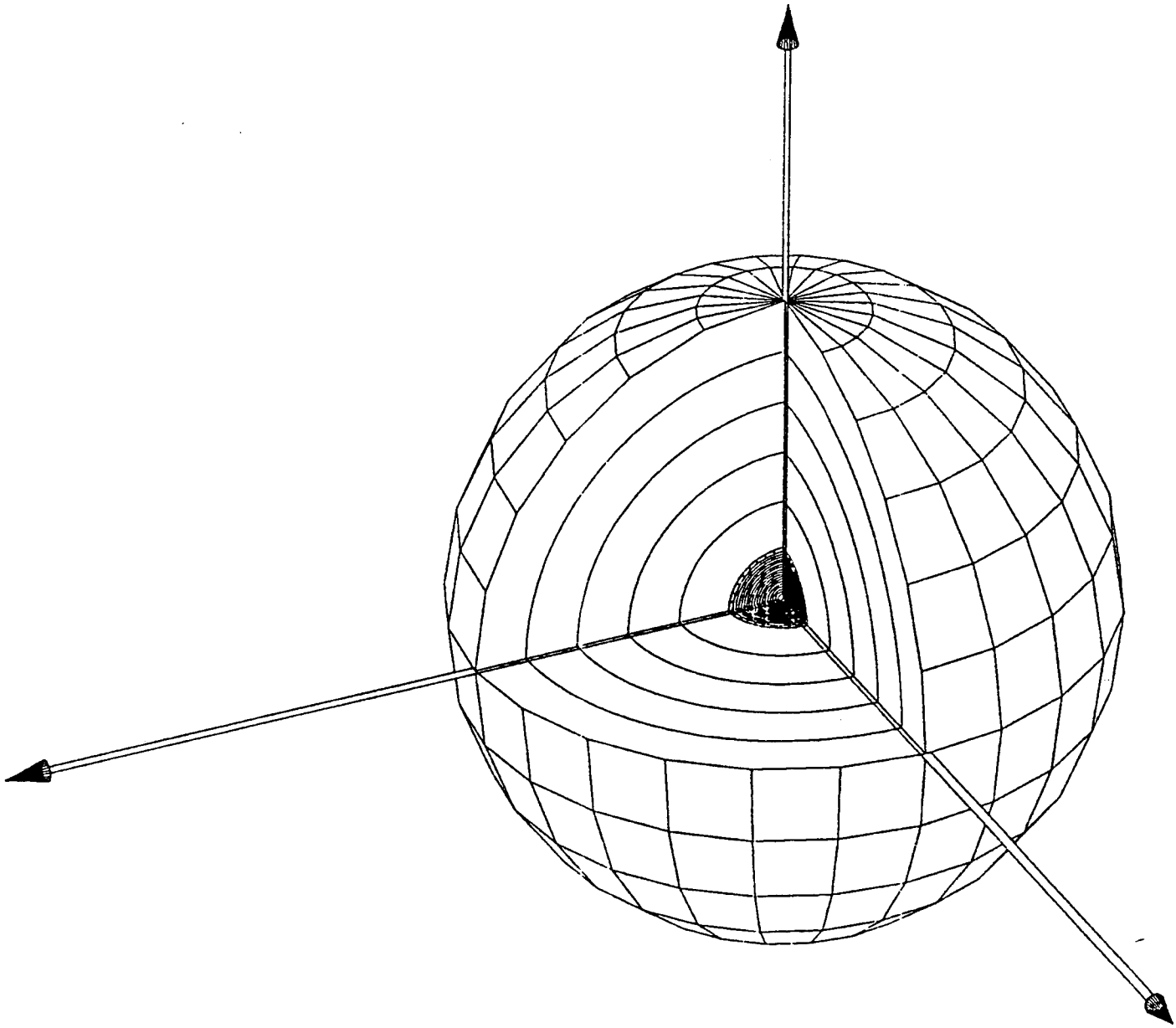

FIGURE 1 - Schematic diagram of a single bubble-liquid module. The shaded central region represents the bubble. The concentric spheres surrounding the bubble are the shells used to simulate the concentration gradient in the liquid. The number and size of the shells and the size of the bubble vary according to the parameters of the model.

The traditional approach to modeling the temporal and spatial evolution of inert gas in tissues using a system of partial differential equations becomes intractable when the architecture of the tissue is complex. An alternative to this type of deterministic modeling is a Monte Carlo simulation (10-14). The essence of this approach involves the placement of inert gas particles in a bubble-liquid module and allowing them to take random steps to simulate diffusion for a short period of time. At the end of the time period the distribution of the particles is used to calculate the number of moles of gas in the bubble and liquid. The new bubble volume is then calculated from the ideal gas law. This process is repeated many times to obtain the time course of bubble evolution.

We start our simulation with the liquid saturated with inert gas at one ambient pressure and then make a step decrease in the pressure. Since there is little information about the initial formation of bubbles (15,16), we simply assume that the gas phase initially has a radius close to the critical radius computed by Ward and Tikuisis (17,18). We then place the bubble centered in the liquid region.

We obtain the value of the critical radius from calculations of equilibrium radii for a closed system with a finite gas supply carried out by Ward and Tikuisis; the first is an unstable radius, referred to as the critical radius r_c. Gas phases smaller than r_c will shrink and disappear, those larger will grow until the second stable equilibrium radius, r_e is reached. Bubbles with radii larger than r_e will shrink until r_e is reached. The quartic polynomial from which r_c and r_e are obtained is given in Appendix A. The specific initial condition values we used in our system test are given in the section entitled "Model Testing" and Table 1.

TABLE 1: Model Parameters by Equation Number

Eq No.	Parameter	Reference
(1)	P_{tisO2} = 39.76 mmHg	27
	P_{tisCO2} = 44.26 mmHg	
(2)	V_{liq} = 0.000125 cm^3, 0.0005 cm^3 PDE	see text
	C_{liq} = 0.055140 moles/cm^3	1/molecular wt
	P_{amb0} = 1520 mmHg, (2 ATA)	see text
	P_{vap} = vapor pressure of water at 37 °C= 46.52 mmHg	27
	α = Ostwald solubility N_2 = 0.0143	28
	R_g = 62358 (cm^3 mmHg)/(mole K)	36
	T = 310.1 K, (37 °C)	
(3)	P_{amb} = 760 mmHg, (1 ATA)	see text
	γ = surface tension = 0.037515 mmHg-cm, (50 dynes/cm)	15
	r_{bub} = 0.00065 cm, 0.00092 cm PDE	see text
	post-bubble formation boundary condition = uniform from all shells	
(23)	D_{N2} = 0.5 X 10^{-5} cm^2/sec, 0.75 X 10^{-5} cm^2/sec PDE	see text
(26)	grid size = 0.00025 cm	see text
	[derived: diffusion step size, (a), = 0.00043 cm]	
(27)	n_{cycle} = 50	see text
	[derived: t_{cycle} = 0.3125 seconds]	
	[derived: t_{cycle} = 0.2100 seconds] PDE	
(57)	τ = 2.5 minutes, 10 minutes PDE	see text
(62)	Fi_{O2} = 0.21, (air)	

The amount of gas required to fill this initial bubble is taken from the liquid phase. This can be done either by removing gas uniformly from within the liquid, leaving no initial gas gradient or by removing gas non-uniformly, yielding an initial gradient. In either case we keep track of the gas concentration with a series of concentric shells constructed around the bubble. The shells expand or contract with the expansion or contraction of the bubble. The fraction of total gas in the module that is present in the liquid is the same whether or not an initial gradient is present. The details of these procedures are explained in the sections entitled "Calculation of Mole Fraction in the Bubble and Liquid Shells" and "Simulation of the Gas Gradient in the Liquid."

After the initial placement of a bubble in the module and the establishment of the gas gradient in the liquid, the simulation begins a repeating cycle of particle placement, random walks, gas wash-out, and adjustment of the bubble size and gas gradient in the liquid. We refer to each repetition of this sequence of events as a bubble growth cycle, shown schematically in Figure 2. The simulation consists of many bubble growth cycles. During each cycle, bubble radius is kept constant and no gas washout occurs while the gas particles diffuse throughout the module.

Simulating diffusion using a Monte Carlo process involves many particles, representing idealized gas molecules, being placed one at a time within the bubble-liquid module. For each particle the location of the placement is randomly selected to be in either the bubble or the liquid, based on the fraction of total gas in each region. The derivation of this placement procedure is contained in the section entitled "Placement of Particles in the Bubble-liquid Module" and Appendix B.

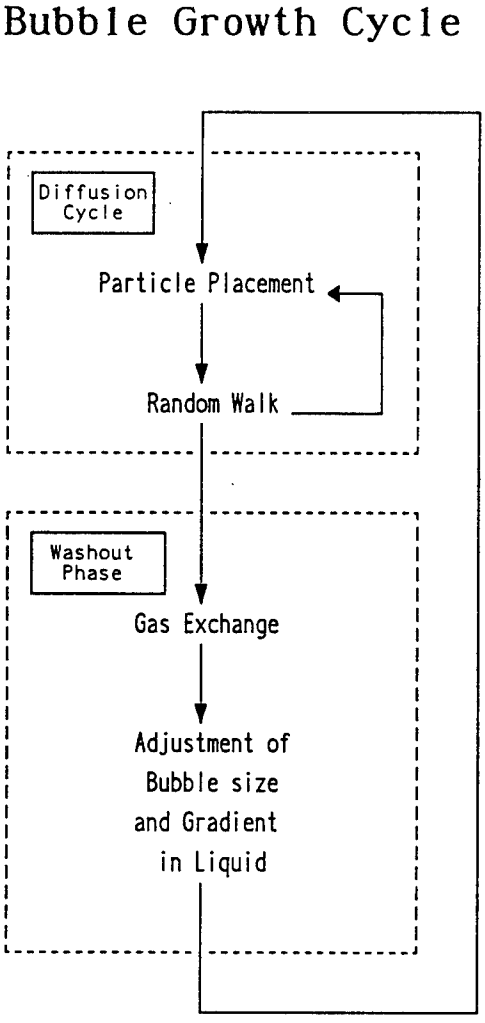

FIGURE 2 - Bubble growth cycle structure.

After placement, each particle walks randomly through the module for a length of time specified as t_{cycle}, the bubble growth cycle time. At the completion of the random walk, the particle's new position (i.e., which shell the particle is in) is recorded. Then another particle is placed and walked through the module. This is repeated for all of the particles (at least 2 x 10^8 particles/(ml of liquid in the module)). The details of this procedure are in the section entitled "Random Walk Procedure."

For purposes of the random walk procedure, gas particles in the bubble and liquid phase are treated differently. If the gas particles are in the bubble, then they are assumed to be uniformly distributed and transition from the bubble to the liquid can only occur if the particle begins its step from within a shell just inside the bubble-liquid boundary. The probability of crossing from the bubble to the liquid in a particular step is the product of three probabilities: the probability of being in the transition region, the probability of striking the bubble-liquid interface, and the probability of crossing the interface if struck.

If in the liquid, the particles are allowed to cross from the liquid to bubble if the bubble-liquid boundary is struck. If a particle strikes the outer liquid boundary during a step it is reflected back into the module. This simulates a zero pressure gradient with no mass transfer across the outer tissue boundary. Details are in the section entitled "Derivation of the Transition Probability."

After all particles have been placed and undergone the random walk, there will be a new particle distribution. The number of particles in the gas phase and each individual shell is divided by the respective volumes of the gas phase or shell, to get the mean concentration in the gas phase and in each shell. Next, the concentration in each shell is adjusted to

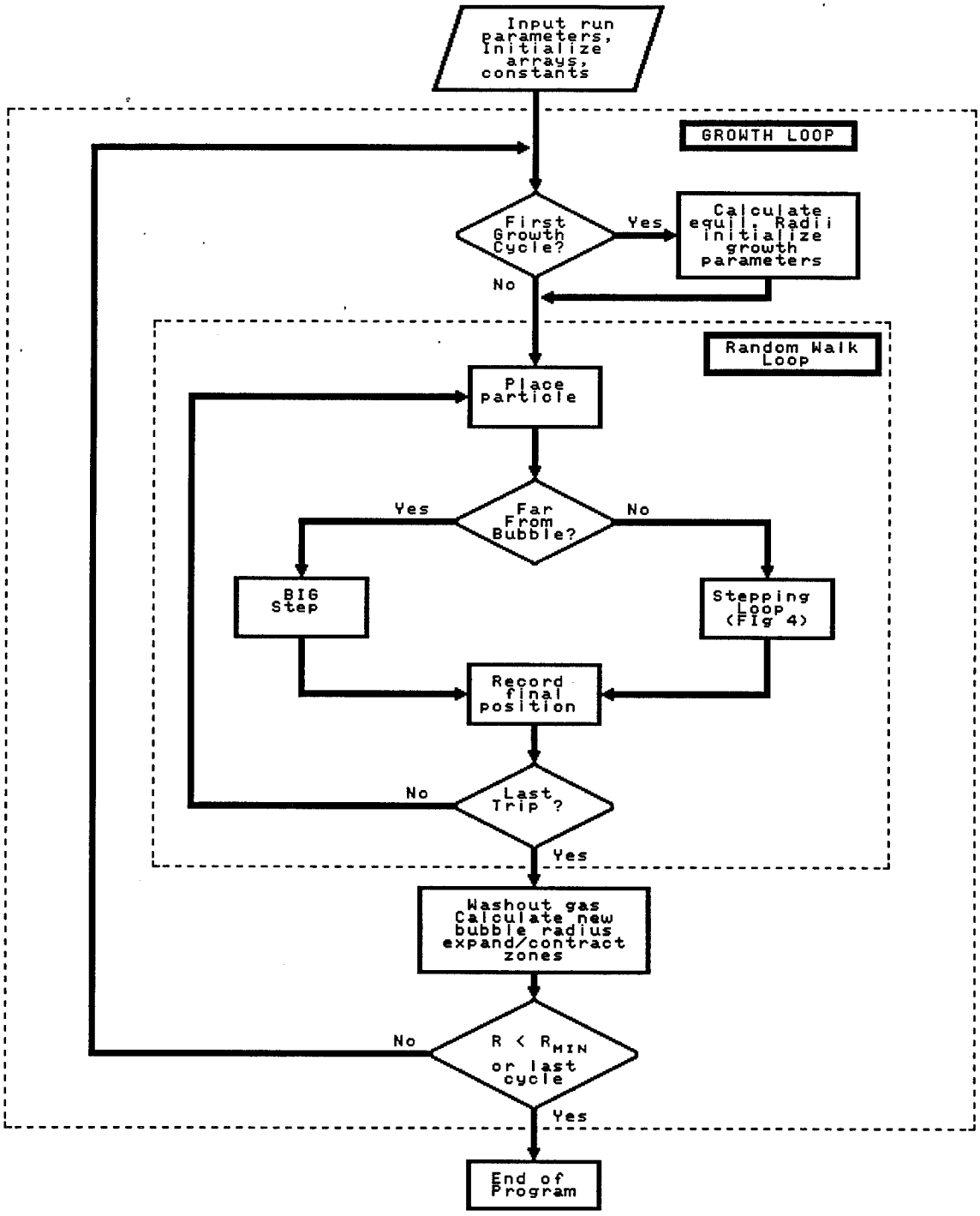

FIGURE 3 - Bubble simulation flow chart. r_{min} is equal to the diffusion step size or critical radius, whichever is larger.

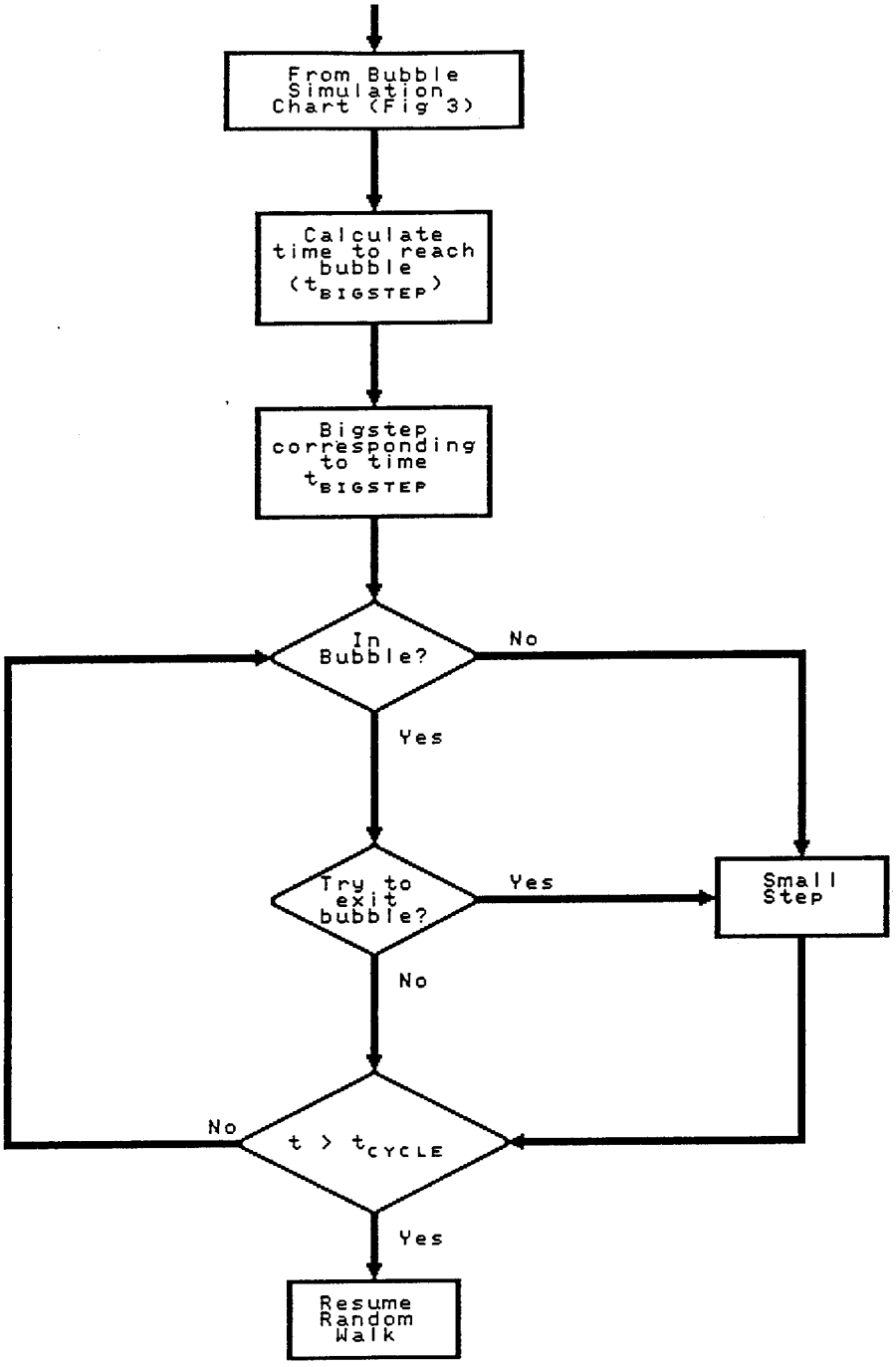

FIGURE 4 - Stepping loop flow chart. This algorithm is followed whenever the particle has a chance of entering the bubble during a bubble growth cycle of length t_{cycle}.

account for the wash-in or wash-out that occurred during the cycle time. Thus, the gas concentration gradient in the liquid is established for the beginning of the next cycle. The bubble radius is adjusted according to the number of moles in the gas phase assuming the ideal gas law as explained in the section entitled "Calculation of the New Bubble Radius." Once this new radius is established the thickness of the shells is adjusted. The new shell radii and the mean gas concentration in each of the shells are the initial conditions for the next cycle. Details of this process are given in the section entitled "Simulation of the Gas Gradient in the Liquid."

In summary, we begin with a liquid saturated with inert gas at some ambient pressure, reduce the ambient pressure, and insert a spherical bubble with a radius just above the critical radius for the system. For a short time interval we fix the bubble size while gas in the bubble-liquid module is redistributed using a Monte Carlo simulation of diffusion. At the end of this time interval, the bubble size and gas gradient in the liquid are adjusted based on this new particle distribution. The next cycle then begins, and this process continues indefinitely. Figures 3 and 4 present flow charts that summarize the model's logical structure. Details of the derivation and implementation of the method follow.

Calculation of Mole Fraction in the Bubble and Liquid Shells

The number of moles of inert gas dissolved in the liquid at the start of the simulation before a bubble forms (n_{tot}) is calculated from the saturated volume of inert gas at the ambient pressure just before the pressure reduction. In living organisms the metabolic gases oxygen and carbon dioxide are present and must be taken into account. We first define the partial pressure from metabolic gases (P_{mg}) to be,

$$P_{mg} = P_{tis_{O_2}} + P_{tis_{CO_2}} \quad (1)$$

where P_{tisO2} = partial pressure of tissue oxygen (mmHg)
P_{tisCO2} = partial pressure of tissue carbon dioxide (mmHg)

For simulations not involving metabolic gases p_{mg} is assumed to be 0. n_{tot} is then calculated as,

$$n_{tot} = \frac{V_{liq} \cdot C_{liq} \cdot P_{liq}}{K_H} \quad (2)$$

where,

V_{liq} = volume of liquid, (cm^3)
C_{liq} = molar concentration of the solvent, ($moles/cm^3$)
P_{liq} = pressure of inert gas in the liquid, (mmHg)
= $(P_{amb0}-P_{vap}-P_{mg})$
P_{amb0} = ambient pressure before pressure reduction, (mmHg)
P_{vap} = vapor pressure of water, (mmHg) (constant)
K_H = Henry's coefficient = $P_H/([(R_g \cdot T/(V_L \cdot \alpha \cdot P_H))+1]^{-1})$, (mmHg)(35)
V_L = molar volume of solvent ($1/C_{liq}$), (cm^3/mole)
α = Ostwald solubility coefficient
P_H = 760 mmHg
R_g = Universal gas constant in (cm^3 mmHg)/(mole K)
T = temperature in Kelvin

Once a bubble is placed in the liquid, the liquid is assumed to remain at constant volume as the bubble expands. All the gas in the bubble comes from the liquid phase, keeping the total amount of gas in the module constant during a bubble growth cycle. Assuming an ideal gas and negligible liquid elastic forces, the number of moles of inert gas in the bubble (n_{bub}) is determined by,

$$n_{bub} = \frac{P_{bub} \cdot V_{bub}}{R_g \cdot T} \tag{3}$$

where,

P_{bub} = inert gas pressure in the bubble = P_{amb}-P_{vap}-P_{mg}+P_γ, (mmHg)
P_{amb} = ambient pressure after pressure reduction, (mmHg)
P_γ = surface tension pressure = $(2 \cdot \gamma)/r_{bub}$, (mmHg)
γ = surface tension, (mmHg-cm)
[γ (mmHg-cm) = γ (dynes/cm) / 1332.8 dynes/(mmHg·cm^2)]
r_{bub}= bubble radius, (cm)
V_{bub} = bubble volume, (cm^3). (V_{bub} is the volume of a bubble of radius r_{bub} selected to be just above the critical radius for a pressure reduction from P_{amb0} to P_{amb}.)

Because placement of particles in the module for the Monte Carlo simulation of diffusion depends on the relative amount of gas in each region of the bubble-liquid module, it is necessary to calculate the mole fraction of gas in each region. The initial mole fraction in the bubble is simply,

$$x_{bub_0} = \frac{n_{bub}}{n_{tot}} \tag{4}$$

and the initial mole fraction in the liquid is,

$$x_{liq_0} = 1 - \frac{n_{bub}}{n_{tot}} \tag{5}$$

The initial mole fraction in each shell, $x_{shelli0}$, is determined from the equation,

$$x_{shell_{i_0}} = \frac{n_{shell_i}}{n_{tot}} \tag{6}$$

At the start of the simulation, the number of moles of gas in each shell, n_{shelli}, is determined by the selection of the gas gradient in the liquid. We have provided for one of two alternative gas distributions: the gas for the bubble can be taken uniformly from all of the shells in the liquid so that there is no initial gradient in the liquid, or the inner-most shells can be depleted of the gas that makes up the bubble so that partial pressure equilibrium across the boundary is maintained. (These two choices represent the largest and smallest gradients for gas diffusion. We may then evaluate the effect of these two extreme post-bubble formation boundary conditions on bubble evolution.) In the uniform distribution case, the number of moles in each shell is the same and is simply,

$$n_{shell_i} = \frac{n_{tot}}{number\ of\ shells} \tag{7}$$

In the other case, the procedure for calculating the number of moles in each shell is more complex. Gas is taken from the innermost shell until the partial pressure is equal to the calculated partial pressure for inert gas in the bubble. If additional gas is required to fill the bubble, the same procedure is applied to succeeding shells until the total gas needed has been taken. Appendix D4 contains the FORTRAN code that implements this procedure.

After the first bubble growth cycle, the particles will have a different distribution than before their random walks simulating diffusion began. This new distribution forms the basis for placement to begin the next bubble growth cycle. Consequently, it is necessary to calculate the mole fraction in each region of the module at the end of the cycle. In this case,

the mole fraction in each region is determined by the fraction of particles that are in the region at the end of a particular bubble growth cycle, which is given by,

$$x_{bub} = \frac{number\ of\ particles\ in\ the\ bubble}{total\ number\ of\ particles\ in\ the\ module} \tag{8}$$

The mole fraction in the liquid is,

$$x_{liq} = \frac{number\ of\ particles\ in\ the\ liquid}{total\ number\ of\ particles\ in\ the\ module} = 1 - x_{bub} \tag{9}$$

and the mole fraction in each shell is,

$$x_{shell_i} = \frac{number\ of\ particles\ in\ shell_i}{total\ number\ of\ particles\ in\ the\ module} \tag{10}$$

Placement of Particles in the Bubble-liquid Module

In order to obtain the correct distribution of particles after they are subject to diffusion and wash-out, the placement of particles at the beginning of each bubble growth cycle cannot be done haphazardly, but must follow certain rules.

First, a random number between 0 and 1 is compared with the mole fraction in the bubble. If the random number is less than the mole fraction the particle is assigned to the

bubble, otherwise it is assigned to the liquid. The liquid is divided into concentric shells of equal width, which are used to simulate the concentration gradient of inert gas in the liquid. Within the liquid, the probability of placement in each shell is proportional to the mole fraction of gas in the shell. In order to weight the particle placement by the mole fraction in each shell, a random number between 0 and 1 is compared with the serially accumulating mole fraction beginning from the inner most shell. At the point the random number is less than that value, that shell is selected for particle placement. The smooth gradient in the liquid is approximated by the series of shells with different mean concentrations in each shell, forming a step gradient. However, the concentration is uniform within each shell, implying that the probability of a particle being placed in any two regions of equal volume within a given shell must be the same. Within a shell, our spherical coordinate system requires that more particles be placed farther from the bubble because a spherical sub-shell far from the bubble will occupy more volume than one of equal width nearer the bubble.

We use the spherical coordinates ρ, θ, and ϕ to place the particles within a shell, where ρ is the radial coordinate, θ the azimuthal angle between 0 and π, and ϕ orthogonal to θ, between 0 and 2π. In order to ensure the proper values for these variables, we must transform the uniformly distributed random numbers (u) between 0 and 1 generated by the computer. We do this using the transformations h_ρ, h_θ, and h_ϕ derived in Appendix B (19).

$$\rho = h_\rho(u) = Ru^{\frac{1}{3}} \tag{11}$$

where R is the radius of the sphere. However, since placement is to occur in a shell and not

an entire sphere, ρ must be adjusted accordingly,

$$\rho = (r_{i-1}^3 + u \cdot (r_i^3 - r_{i-1}^3))^{\frac{1}{3}} \quad (12)$$

with ρ between the outer (r_i) and inner (r_{i-1}) shell radii.

$$\theta = h_\theta(u) = \arccos(1-2u) \quad (13)$$

and

$$\phi = h_\phi(u) = 2\pi \cdot u \quad (14)$$

These functions map the uniformly distributed random variable u between 0 and 1 into the values ρ, θ and ϕ which are uniformly distributed by volume within a shell. These coordinates are translated to cartesian coordinates,

$$x = \rho \cdot \sin(\theta) \cdot \cos(\phi) \quad (15)$$

$$y = \rho \cdot \sin(\theta) \cdot \sin(\phi) \quad (16)$$

$$z = \rho \cdot \cos(\theta) \quad (17)$$

Random-walk Procedure

The simplest way of simulating diffusion with a Monte-Carlo process is to translate from the spherical coordinate system of Figure 1 into a cartesian coordinate system using Equations 15-17. A random number generator would be used to specify positive or negative movement in the X, Y, and Z directions (10-12), with the length of the movement in each

dimension being equal to a cartesian grid unit. The particle would end up on one of the 8 corners of the cube with sides 2 grid units in length centered on the original position. Thus the effective distance traveled, which we refer to as the diffusion step size (a), would be $\sqrt{3}$·(length of the grid unit). This process would be repeated for a predetermined number of steps and would result in the generation of the particle's path through the module.

Unfortunately, this method requires large amounts of computer execution time. We took advantage of the fact that the region outside the bubble is isotropic to diffusion by calculating the distribution of distances a particle will travel in a relatively large amount of time under the influence of diffusion alone. We refer to this as the Bigstep distribution. This distribution is computed as follows. If an amount of substance M is deposited at a point in an infinite volume, then the concentration C at a distance r from the point source under the influence of diffusion alone is given by Crank (20) as,

$$C=\frac{M}{(4\pi Dt)^{3/2}}\cdot e^{\frac{-r^2}{4Dt}} \qquad (18)$$

By dividing both sides of this equation by M we obtain the probability density function (PDF) of the distance r, a particle will travel in time t from a point in the volume,

$$f(r,t)=\frac{1}{(4\pi Dt)^{3/2}}\cdot e^{\frac{-r^2}{4Dt}} \qquad (19)$$

The cumulative distribution function (CDF) can then be obtained by integrating the PDF over the volume,

$$F(r,t)=\frac{1}{(4\pi Dt)^{3/2}}\int_0^r\int_0^{2\pi}\int_0^{\pi} e^{\frac{-r^2}{4Dt}}\cdot r^2\cdot\sin\theta\, d\theta\, d\phi\, dr \qquad (20)$$

which reduces to,

$$F(r,t)=\frac{4\pi}{(4\pi Dt)^{3/2}}\int_0^r r^2\cdot e^{\frac{-r^2}{4Dt}}dr \qquad (21)$$

If we let $z=\frac{r}{\sqrt{2Dt}}$ this becomes,

$$F(z)=\sqrt{\frac{2}{\pi}}\cdot\int_0^z z^2\cdot e^{-\frac{z^2}{2}}dz \qquad (22)$$

Since a closed form solution of this integral cannot be obtained, we integrate it numerically to obtain F(z), which is the probability a particle will travel the normalized distance z in time t. We set the upper bound of the CDF integral at 5 normalized distance units ($5\cdot\sqrt{2Dt}$) because 99.99% of the particles will travel less than or equal to this distance in time t. Graphs of the PDF and numerically obtained CDF are shown in Figure 5.

By applying the inverse transformation method (13) to the CDF by randomly generating a number between 0 and 1, we can select the normalized distance z a particle travels. We do this by discretizing F(z) into increments of 0.0001 and assign to each the value of z needed to obtain it. These z values are stored in a 10,000-element array. We

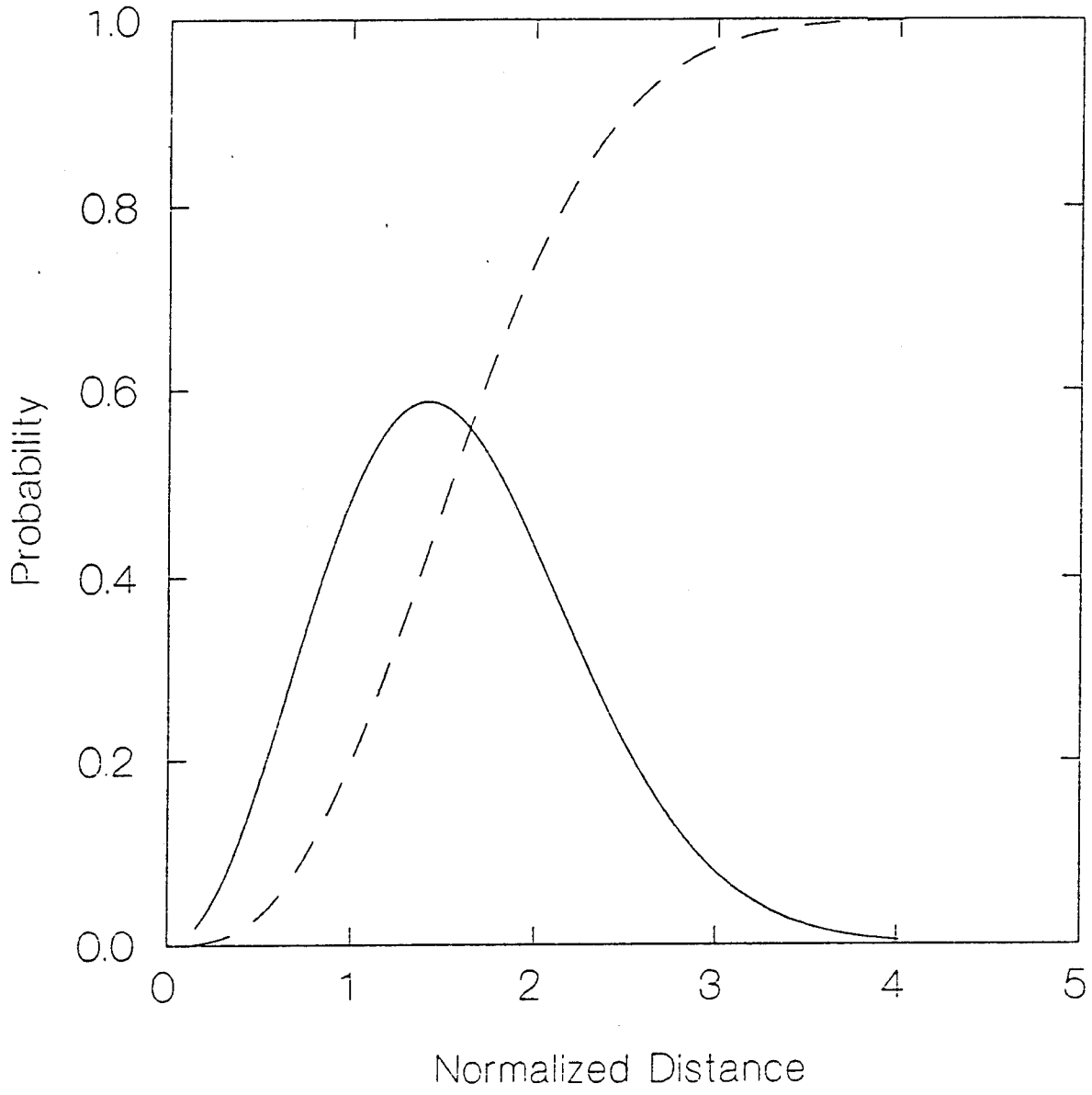

FIGURE 5 - Probability and cumulative distribution functions of normalized diffusion distances in three dimensions. The solid line represents the PDF and the dashed line represents the CDF.

then randomly generate a value for F(z) and go to the array to find the associated value of z.

Since $z=\frac{r}{\sqrt{2Dt}}$, we must either select a value for r and compute t, or conversely specify t and compute r. We elect to keep the diffusion time increment, t, constant and compute the distance traveled as,

$$r = z\sqrt{2Dt} \tag{23}$$

The value we choose for t is limited by the architecture of the module. One constraint placed on the procedure is that we do not want the particle to cross the liquid-gas boundary during a bigstep. Once a particle enters the bubble it must be given the opportunity to exit, diffuse in the liquid and reenter the bubble, perhaps on several occasions. If we allowed the particle to enter the bubble on a bigstep this opportunity would not be provided and we would end up with the wrong distribution of particles between the phases of the module.

We apply this constraint by first requiring that the bigstep only be allowed for particles outside of the first shell. Secondly, we require that the probability of the particle reaching the bubble boundary during a bigstep be very small (<0.0001). Since the upper limit of the integral in Equation 22 is 5 normalized distance units, the probability of attaining a distance of greater than or equal to 5 normalized distance units is less than 0.0001. If the particle is a distance r_0 from the boundary, then as long as $\frac{r_0}{\sqrt{2Dt}} \geq 5$ the particle has a

very small chance (<0.0001) of reaching the bubble boundary during time t.

We define the largest distance a particle can travel in a straight line on a single diffusion step to be the mean squared distance. The mean squared distance (MSD) a particle travels under the influence of diffusion in three dimensions is (20),

$$MSD = 6Dt \tag{24}$$

where D is the diffusion coefficient and t is the time allowed for diffusion. At the beginning of the simulation, we assign a value for the mean squared distance on a single diffusion step to be a^2 where,

$$a^2 = 6Dt_{step} \tag{25}$$

We solve Equation 25 for t_{step}, to get the time to travel the distance (a) that corresponds to a single diffusion step,

$$t_{step} = \frac{a^2}{6D} \tag{26}$$

We also assign a value to the number of single steps each particle will undergo during each bubble growth cycle, n_{cycle}, and compute the time for each bubble growth cycle, t_{cycle}, as

$$t_{cycle} = n_{cycle} \cdot t_{step} \tag{27}$$

The time t_{cycle} is the longest time allowed for a bigstep, so if $\frac{r_0}{\sqrt{2Dt_{cycle}}} \geq 5$ then there is

almost no chance the particle will reach the boundary during time t_{cycle}. In this case the distance r_T actually traveled during the time t_{cycle} can be calculated directly from Equation 23 as $r_T = z\sqrt{2Dt_{cycle}}$.

The procedure can be illustrated with an example. If the computer generated random number is 0.12145673, we multiply it by 10,000 and round to obtain the array index, 1,215. The associated z value of 0.823 (obtained from a table look-up) combined with D of $0.5X10^{-5}$ cm^2/sec and t equal to t_{cycle} of 0.3125 sec in Equation 23, gives the actual distance traveled of 0.00173 cm (17.3 microns).

If $\frac{r_0}{\sqrt{2Dt_{cycle}}} < 5$ then the time allowed for bigstep is reduced as follows. First, the number of single steps of length a in a direct path to the boundary is calculated as,

$$n_{step} = \frac{r_0}{a} \tag{28}$$

then

$$t_{bigstep} = n_{step} \cdot t_{step} \tag{29}$$

and

$$r_T = z\sqrt{2Dt_{bigstep}} \tag{30}$$

The probability of all of the random walk steps occurring in the same direction is small, so by computing the distance traveled in time $t_{bigstep}$ using the Bigstep distribution, the particle will have almost no chance of stepping distance r_0 and entering the bubble.

Although this is the method we have used in the current version of our program, a more consistent approach would be to calculate $t_{bigstep}$ from Equation 23 after transforming the radial distance to the bubble boundary into 5 normalized distance units.

$$t_{bigstep}=(\frac{r_0}{5})^2\cdot(\frac{1}{2D}) \tag{31}$$

Then by randomly generating z as previously described, and using $t_{bigstep}$ from Equation 31, the distance traveled can be calculated using Equation 30.

This approach guarantees that the particle will have less than a 0.01% chance of entering the bubble on a bigstep.

The direction of the step is obtained by randomly generating two orthogonal angles, θ and ϕ using Equations 13 and 14. Thus, a particle will be placed randomly on the surface of a sphere centered at its initial location with a radius obtained randomly from the Bigstep distribution.

For those particles with $t_{bigstep} < t_{cycle}$, the particles must be given the opportunity to take individual diffusion steps as described in the beginning of this section to complete the bubble growth cycle. The time remaining in the bubble growth cycle after taking the bigstep is obtained and the number of single diffusion steps is calculated as,

$$n_{step} = \frac{t_{cycle} - t_{bigstep}}{t_{step}} \quad (32)$$

If the particle has crossed the bubble boundary and entered the gas phase, a different stepping procedure is invoked. We can calculate the probability the particle will be available to exit from the bubble during a single step, which is equal to the product of three individual probabilities: the probability of being in the transition region, the probability of striking the bubble-liquid interface from the transition region, and the probability of crossing the interface if struck. If a random number generated by the computer is less than the product of these probabilities, the particle is placed in the transition region uniformly by volume as described in the section on placement of particles. It is then allowed to take a single step to the surface of a sphere of radius (a), centered at its location in the transition region. If this step does not take the particle outside the boundary, it is "returned" to the well-mixed bubble region and given another opportunity to be available for exiting. This cycle is repeated until the particle steps out of the bubble or the number of steps allowed in a bubble growth cycle is reached. The derivation of this procedure is described in the section "Derivation of the Bubble Transition Probability."

Any particle that reaches the outer boundary of the liquid module is reflected inward to simulate a particle entering from an adjacent bubble-liquid module. A slight error is introduced because packed spheres are not space filling, and particles can never enter this intersphere region.

Simulation of the Gas Gradient in the Liquid

In order to simulate a concentration gradient within the liquid, "shells" consisting of

concentric spheres are placed around the bubble. At the start of each bubble growth cycle the width of each shell is $k\cdot\sqrt{2Dt_{cycle}}$,where D is the diffusion coefficient, t_{cycle} is cycle time, and k is a constant between 0 and 1. The root mean square distance the particle will travel in t_{cycle} is $\sqrt{6Dt_{cycle}}$. Since a particle will travel this average distance in t_{cycle}, the assumption that the shells are well mixed is consistent with a maximum shell width of $\sqrt{2Dt_{cycle}}$. This choice for shell width is also convenient, since it is proportional to the normalized distance unit of the Bigstep distribution. For greater resolution of the gradient, k can be selected smaller than 1.

After the random walk portion of the growth cycle the bubble will change size. This will result in a change in the thickness of the shells surrounding the bubble. For example, if the bubble grows the inner and outer radii of each shell increase, which makes the shell thickness become smaller since the shell volume is held constant during expansion. For a growing bubble the shell width would eventually become very small. In order to adjust for this, a new set of shells of width $k\cdot\sqrt{2Dt_{cycle}}$ is created at the beginning of each cycle. These new shells will overlap the expanded or contracted shells of the previous cycle as shown in Figure 6. While they are the same thickness as the previous set of shells before the bubble changed size, they will not have the same volume as the expanded or contracted shells of the previous cycle. Particles are placed using the expanded/contracted shells of the previous cycle. The new set of shells is used to record the final position of the particles at the end of their random walks.

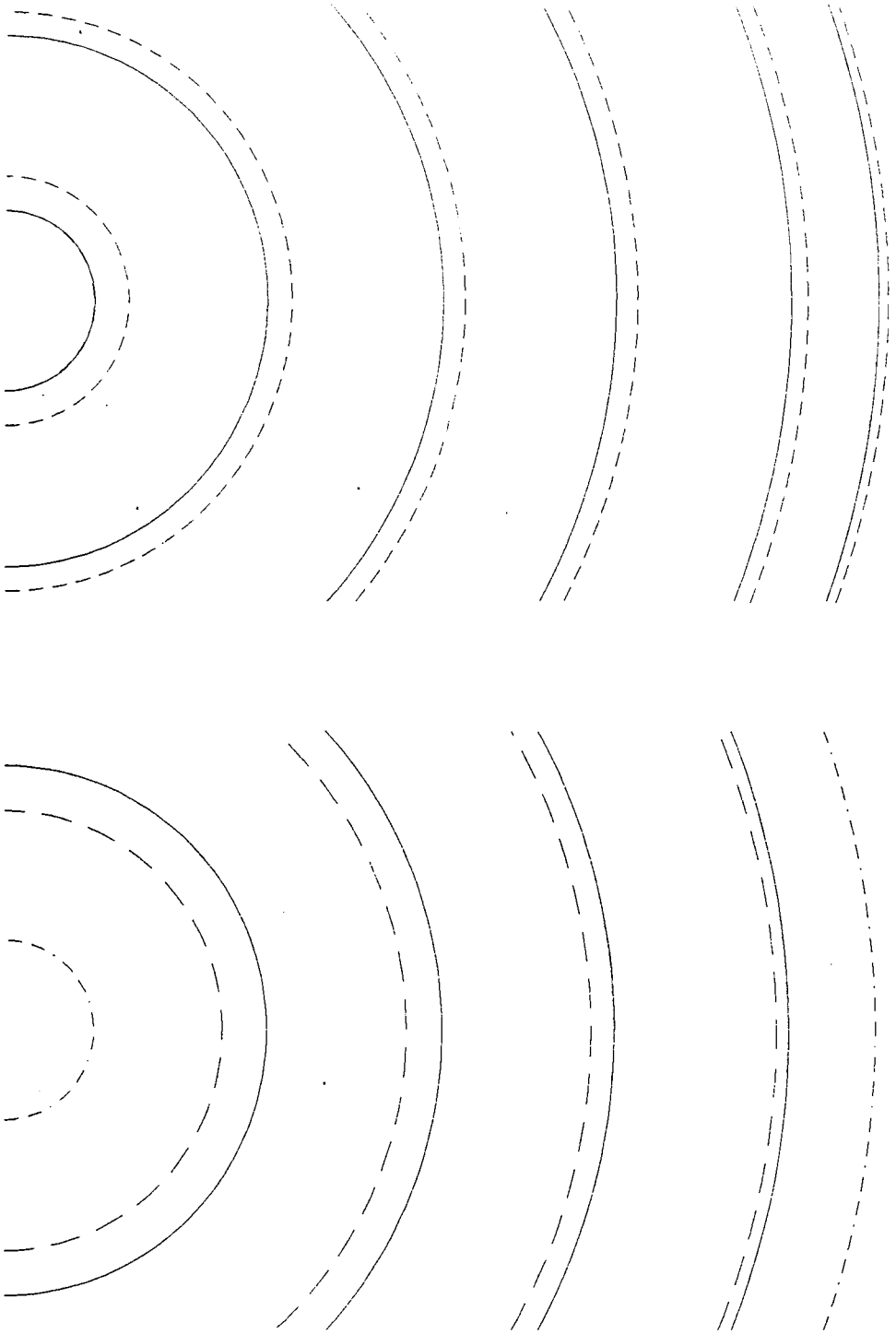

FIGURE 6 - Schematic diagram of cross-sectional view of changes in shells during bubble expansion. In the first frame, the dashed lines represent the location of the shell boundaries prior to expansion. After the bubble expands, the shell boundaries also expand to the solid lines so that shell volumes are the same. In the second frame, a new set set of shells, all with the same width, are overlayed on the expanded shells of frame 1. Placement of particles in these new shells is weighted by the expanded shells.

Derivation of the Bubble Transition Probability

In our simulation the gas phase is assumed homogeneous and a particle in the gas phase has an equal probability of being anywhere in the bubble. If a particle initially in the liquid phase encounters the bubble boundary, it is allowed to cross the boundary. Since a single diffusion step is the largest straight line distance a particle is allowed to travel (representing the Mean Squared Distance), a particle is not allowed to enter and again exit the bubble during a single diffusion step.

If a particle is placed in the bubble, we then compute a probability that it will exit on a particular step. To compute this probability we first define an inner transition region within the bubble which is a spherical shell of thickness (a), whose outer boundary is the gas-liquid interface. Only particles placed in this inner transition region are allowed to exit the bubble (with a certain probability), particles not in the transition region cannot exit the bubble.

The probability of exiting the bubble is the product of three individual probabilities: the probability of being in the inner transition region, the probability of striking the bubble-liquid interface from the transition region and the probability of crossing the interface if struck.

The probability of being in the inner transition region is simply the ratio of the inner transition region volume to the bubble volume,

$$p_{trans} = \frac{R^3-(R-a)^3}{R^3} \tag{33}$$

Once in the inner transition region the probability of striking the interface is derived as follows. The particle can step from point P in the inner transition region to any point a

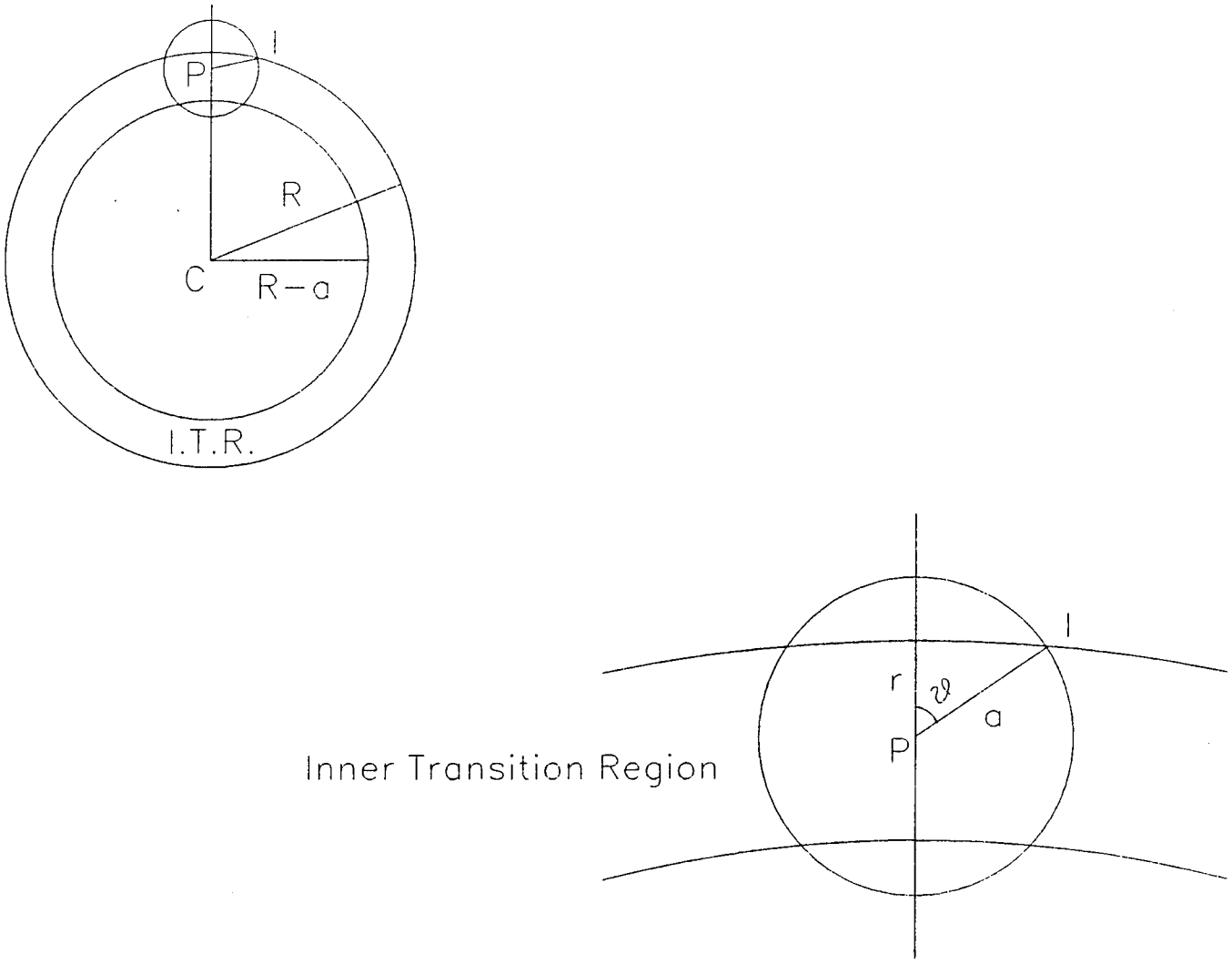

FIGURE 7 - Schematic diagram of transition region inside the bubble. The upper part of this illustration represents a cross section through the bubble centered at point C with radius R. The transition region (T.R.) extends a distance a inward from the bubble boundary. The lower part of the illustration is a cross section through the transition sphere of radius a centered at point P in the transition region. The point I is as the intersection of the transition sphere and the bubble cross section. r is the distance of point P from the surface of the bubble.

distance (a) away, as illustrated in Figure 7. The probability of striking the interface (P_{int}) is the fractional surface area of the sphere of radius (a) that lies outside of the bubble. The fraction of this small sphere outside of the bubble is given by,

$$P_{int}(\theta)=\frac{(1-\cos(\theta))}{2} \tag{34}$$

where θ is the angle between line CP and line PI, and I is a point of intersection of the two spheres. Using the law of cosines, we obtain the following expression for $\cos(\theta)$,

$$\cos(\theta)=\frac{2rR-r^2-a^2}{2a(R-r)} \tag{35}$$

where r is the distance of point P from the surface of the bubble. The probability that a particle strikes the bubble interface from point P is then,

$$P_{int}(r)=\frac{r^2-2(a+R)r+a^2+2aR}{4a(R-r)} \tag{36}$$

To get the probability of striking the interface from anywhere in the transition region, we need to integrate this expression over the entire region, weighted appropriately.

$$P_{int}=\frac{\int_0^a P_{int}(r)(4\pi(R-r)^2)dr}{\int_0^a (4\pi(R-r)^2)dr} \tag{37}$$

Substituting in the expression for $P_{int}(r)$, and integrating yields

$$P_{int} = \frac{a(12R^2-a^2)}{16(R^3-(R-a)^3)} \tag{38}$$

If the particle strikes the interface, the probability of crossing the interface is simply the Ostwald solubility coefficient,

$$p_{cross} = \alpha \tag{39}$$

The individual probabilities can be combined to obtain the probability of exiting the bubble on a single diffusion step p_{exit},

$$p_{exit} = p_{cross} \cdot p_{trans} \cdot p_{int} \tag{40}$$

which reduces to,

$$p_{exit} = \alpha \frac{a(12R^2-a^2)}{16R^3} \tag{41}$$

Verification of the Transition Algorithm - Because the transition algorithm extends methods originally developed assuming diffusivity was approximately the same in high and low solubility regions, we felt obligated to present an alternative argument in support of using this method in our simulation. The second approach to calculating the probability of leaving a gas bubble begins by defining several probabilities.

We define an outer transition region, also of thickness (a), in the liquid immediately adjacent to the bubble as shown in Figure 8. $P(t|g)$ is the probability of entering the outer transition region given that the particle is in the gas phase. $P(g|t)$ is the probability of entering the gas phase given that the particle is in the outer transition region. If P(t) is the probability of being in the outer transition region, and P(g) is the probability of being in the

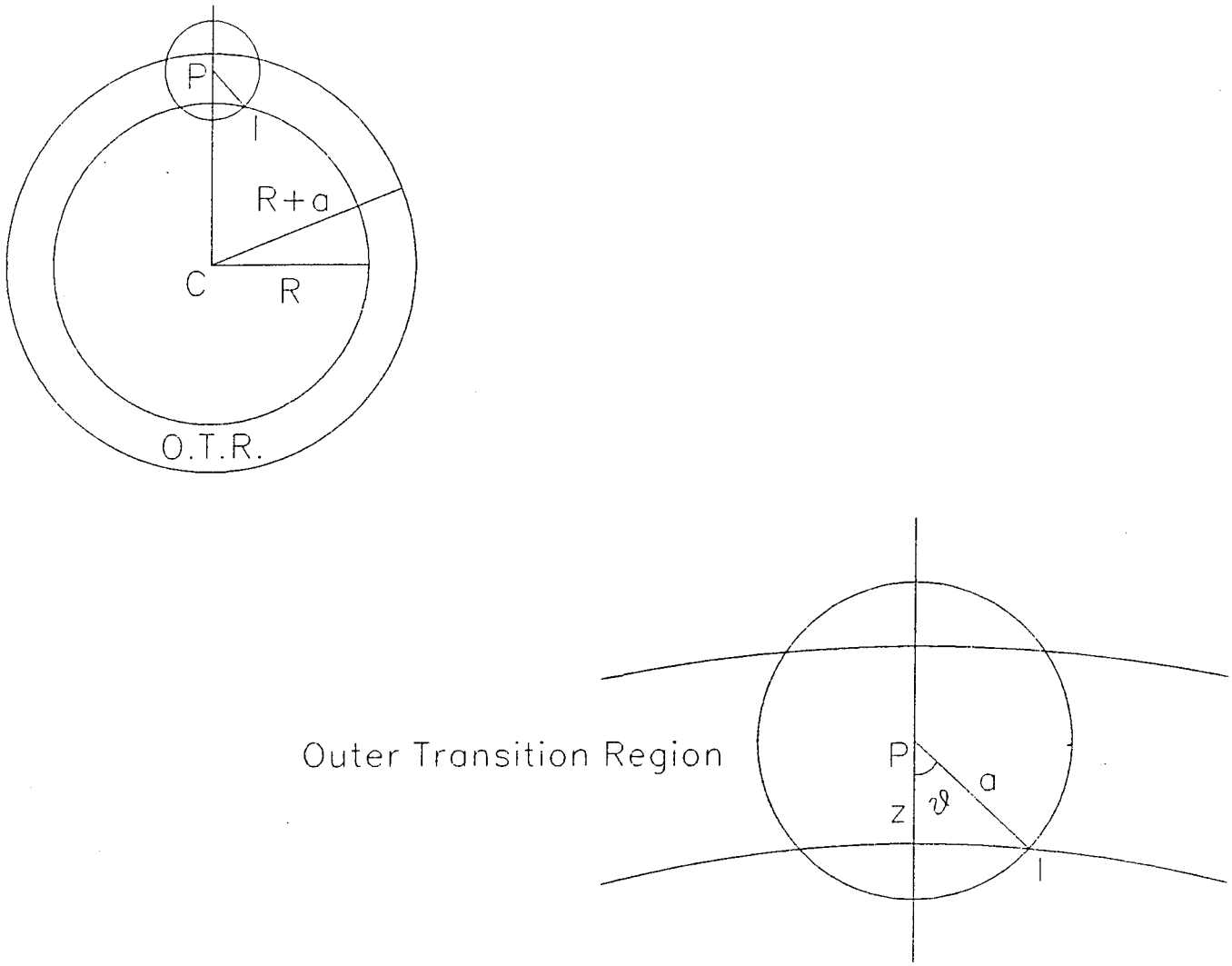

FIGURE 8 - Schematic diagram of transition region outside the bubble. The upper part of this illustration represents a cross section through the bubble centered at point C with radius R. The transition region (T.R.) extends a distance a outward from the bubble boundary. The lower part of the illustration is a cross section through the transition sphere of radius a centered at point P in the transition region. The point I is at the intersection of the transition sphere and the bubble cross section. z is the distance of point P from the surface of the bubble.

gas phase then, at equilibrium, particles must be transferring from liquid to gas at the same rate as from gas to liquid, so we have,

$$P(t|g)P(g) = P(g|t)P(t) \tag{42}$$

Also, at equilibrium, since the partial pressures of the gas must be equal on either side, the probabilities, P(i) of being in phase i with volume V_i are related to the Ostwald solubility coefficient α by,

$$\frac{P(g)}{P(t)} = \frac{V_g}{V_t \alpha} \tag{43}$$

These two equations may be solved for P(t|g) as a function of P(g|t), the volumes, and the solubility coefficient.

P(g|t) is a geometric factor that we calculate as follows. Imagine a particle within the outer transition region, a shell of thickness (a) surrounding the bubble of radius R, and located a distance $z \leq a$ from the gas phase boundary as in Figure 7. The next diffusion step of size (a) defines a sphere of radius (a). The probability that the particle will enter the gas phase is proportional to the surface area of this small sphere, which lies within the bubble. If the sphere of radius (a) intersects the bubble at an angle θ from the line normal to the surface of both spheres, then θ satisfies,

$$Cos(\theta) = \frac{a^2+2Rz+z^2}{2a(R+z)} \tag{44}$$

and P(g|t) is given by the fraction

$$P(g|t) = \frac{1-\cos\theta}{2} \tag{45}$$

Then P(t|g) can be obtained in terms of a, R, z, and α by substitution into the equation for P(t|g) above with the expression for P(g|t) and writing V_g in terms of R.

$$P(t|g) = \frac{\alpha(1-\cos\theta)((R+a)^3-R^3)}{2R^3} \tag{46}$$

Using the expression for $\cos\theta$, we obtain,

$$P(t|g) = \frac{\alpha[2a(R+z)-(a^2+2Rz+z^2)][(R+a)^3-R^3]}{4aR^3(R+z)} \tag{47}$$

Notice that as z tends to (a), P(t|g) tends to zero. We interpret this to mean that exits into the transition region do not occur with a frequency representing the volume, but are biased in closer to the bubble. The cumulative probability of exit is,

$$F(t|g) = \frac{\int_0^a P(t|g)4\pi(R+z)^2dz}{\int_0^a 4\pi(R+z)^2dz} \tag{48}$$

which integrates directly to give,

$$F(t|g) = \alpha\frac{a(12R^2-a^2)}{16R^3} \tag{49}$$

This is equivalent to the expression P_{exit} used in the transition algorithm.

Calculation of the New Bubble Radius

After completing a bubble growth cycle, the location of all particles placed during the cycle is known. The mole fraction in the bubble, x_{bub} is taken to be the ratio of the number

of particles ending in the bubble and the total number of particles placed during the cycle from Equation 8. The number of moles of gas in the bubble is then calculated as,

$$n_{bub} = x_{bub} \cdot n_{tot} \tag{50}$$

Assuming the ideal gas law and using the expression for bubble pressure derived in equation 3 gives,

$$(P_{amb} + \frac{2\gamma}{r_{bub}} - P_{vap} - P_{mg}) \cdot (\frac{4\pi}{3} r_{bub}^3) = n_{bub} RT \tag{51}$$

This reduces to the cubic polynomial,

$$\frac{4\pi}{3}(p_{amb} - P_{vap} - P_{mg}) r_{bub}^3 + \frac{8\pi}{3}\gamma r_{bub}^2 - n_{bub} RT = 0 \tag{52}$$

The real positive root of this polynomial is the new bubble radius. If the bubble radius becomes less than the step size or the critical radius, the bubble vanishes.

Derivation of Wash-out Probability

Our intention is to apply the model to study gas wash-out from animals and humans during decompression. Although this goal requires us to model heterogeneously distributed sources and sinks, we begin with tissue taken to be homogeneously perfused for simplicity.

This assumption implies that the amount of gas washed out per unit time from the tissue is inversely proportional to the tissue time constant, τ. Thus during a single diffusion step of time t_{step}, t_{step}/τ of the material will wash out. This allows us to define $p_{wash} = t_{step}/\tau$ as the probability of washing out on a single step so that we can calculate the wash-out probability after n_{cycle} steps as,

$$P_{cycle}=p_{wash}(1+(1-p_{wash})+(1-p_{wash})^2+\ldots+p_{wash}(1-p_{wash})^{n_{cycle}-1} \quad (53)$$

which can be expressed as,

$$P_{cycle}=p_{wash}\cdot\frac{1-(1-p_{wash})^{ncycle}}{1-(1-p_{wash})} \quad (54)$$

which reduces to,

$$P_{cycle}=1-(1-p_{wash})^{n_{cycle}} \quad (55)$$

Alternatively for ease of calculation, this can be expressed as an exponential since if e^{-pwash} is expanded as a Taylor series it is easy to see that for small values of p_{wash}, $(1-p_{wash})$ is approximately equal to e^{-pwash}. In fact, if $p_{wash}<0.075$ this approximation is accurate to within 4%. Since p_{wash} is generally smaller than this we can rewrite p_{cycle} as,

$$P_{cycle}=1-e^{-n_{cycle}\cdot p_{wash}} \quad (56)$$

Since $n_{cycle}\cdot p_{wash} = n_{cycle}\cdot t_{step}/\tau = t_{cycle}/\tau$, this can be written as,

$$P_{cycle}=1-e^{-\frac{t_{cycle}}{\tau}} \quad (57)$$

The inert gas concentration in tissue after wash-out (C_{adj}) can be obtained by recognizing that P_{cycle} is simply the fraction of the concentration above the asymptotic value (concentration when $t = \infty$) removed during wash-out, so that,

or

$$P_{cycle} = \frac{C_0 - C_{adj}}{C_0 - C_\infty} \tag{58}$$

$$C_{adj} = C_0 - P_{cycle}(C_0 - C_\infty) \tag{59}$$

where

C_0 = concentration before adjusting for wash-out
C_∞ = concentration when $t=\infty$.

and C_∞ is computed from the arterial inert gas tension, (P_{ainert}), which we assume to be equal to the alveolar inert gas tension, (P_{Ainert}) calculated as (22),

$$P_{A_{inert}} = P_{amb} - (P_{A_{O_2}} + P_{A_{CO_2}} + P_{vap}) \tag{60}$$

where,

$$P_{A_{O_2}} = F_{i_{O_2}}(P_{amb} - P_{vap}) - P_{A_{CO_2}} \tag{61}$$

Combining these two equations yields,

$$P_{A_{inert}} = (P_{amb} - P_{vap}) \cdot (1 - F_{i_{O_2}}) \tag{62}$$

where:
P_{ACO2} = Alveolar carbon dioxide partial pressure (mmHg)
P_{AO2} = Alveolar oxygen partial pressure (mmHg)
F_{iO2} = Inspired oxygen fraction

so that,

$$C_\infty = \frac{P_{A_{inert}}}{760} \cdot \frac{1}{22.4} \left(\frac{moles}{l \cdot ata}\right) \cdot \alpha \tag{63}$$

and

$$C_{adj} = C_0 - P_{cycle} \cdot (C_0 - \frac{P_{A_{inert}}}{760} \cdot \frac{1}{22.4} \cdot \alpha) \quad (64)$$

Model Testing

We tested our model by first verifying that the equilibrium conditions predicted by Tikuisis and Ward (17,18) were met for closed systems with one gas as calculated by the equations in Appendix A. Wash-out was then simulated without the presence of a bubble and compared with the expected single exponential curve. Next, we compared the solution with that of a partial differential equation model. Finally, we developed measures for summarizing the inherently variable Monte Carlo simulations and examined the model's performance at different levels of precision.

For comparison of the time course of bubble evolution with that produced by a standard method, we solve the partial differential equation (PDE):

$$\frac{\partial P_{liq}}{\partial t} = D(\frac{\partial^2 P_{liq}}{\partial r^2} + \frac{2}{r} \frac{\partial P_{liq}}{\partial r}) \quad (65)$$

where P_{liq} is the pressure of inert gas in the tissue, D is the diffusion coefficient, t is time, and r the radial coordinate. The boundary at the bubble-liquid interface satisfies the condition that the inert gas partial pressure near the bubble in the liquid is the same as that in the bubble and also satisfies the condition that the flux into the bubble is dependent on the gradient in the liquid near the bubble so that,

$$\frac{dn_l}{dt} = -\alpha D A_b \frac{\partial P_{liq}}{\partial r}\Big|_{r=r_{bub}} \quad (66)$$

Where n_l is the number of moles in the tissue, α is the solubility, and A_{bub} is the surface area of the bubble of radius r_{bub}. The outer boundary is a reflection boundary. The internal bubble pressure is given by Equation 3. The real positive root of Equation 52 is the new bubble radius. Gas wash-out is assumed to be uniform throughout the liquid. It is calculated assuming the rate of gas elimination is proportional to the amount of gas present. We have developed a solution to this system using the Crank-Nicholson method.

Because Monte Carlo models are based on random samples from the distributions of interest, no two simulations will be the same. As a consequence it is necessary to do multiple runs and use summary measures for comparisons. To estimate the variability in our model predictions we did 10 runs at 3 different levels of particle density precision: 2×10^8 particles/cm^3, 4×10^8 particles/cm^3, and 8×10^8 particles/cm^3. (For comparison, water in equilibrium with nitrogen gas at 1 ATA and room temperature has approximately 5×10^{17} molecules of N_2/cm^3.) When summarizing the 10 runs at a given level of precision, five measures are of particular interest: the maximum radius achieved (r_{max}), time to maximum radius (t_{max}), time to bubble dissolution (t_{dis}), mean transit time (TT), and relative dispersion of the transit times (RD) (2325). RD is equal to the standard deviation of the TT divided by the mean TT. From the moment of the step change in P_{amb}, t_{max} and t_{dis} are measured. Mean TT and RD do not represent the parameters of a single transit time distribution, as they do under normobaric conditions. This is because the evolving bubble continuously changes

the transit time distribution of the system. Nevertheless, this approach was selected because these measures represent a simple means for quantifying the effect of bubble evolution on gas transit through tissue, which is readily comparable with prior work.

The transit time probability density function describes the distribution of the transit times of all the inert gas particles in the tissue. If we express the transit time probability density function f(t) as,

$$f(t) = \sum_i B_i e^{-t/\beta_i} \tag{67}$$

where the B_i and β_i constants, the wash-out function g(t) is given by,

$$g(t) = Qc_a(1 - \int_o^t f(x)dx) \tag{68}$$

which reduces to,

$$g(t) = Qc_a \sum_i \beta_i \beta_i e^{-t/\beta_i} \tag{69}$$

where Qc_a is a scaling factor attributable to the blood flow and wash-in concentration. Furthermore, we know that,

$$TT = \sum_i \beta_i \beta_i^2 \tag{70}$$

and

$$RD = \frac{\sqrt{[2(\sum_i \beta_i \beta_i^3) - TT^2]}}{TT} \tag{71}$$

In order to estimate TT and RD, we fit g(t) to the simulated wash-out curves produced by the model, then use the exponential parameters to calculate our estimates of the mean transit time and relative dispersion of transit times (26).

Model Parameters

The parameters for the model can be divided into three categories: (1) the physical variables, (2) the boundary values immediately after the bubble is formed, and (3) the simulation control inputs. The physical variables include the pressure profile, surface tension, bubble density, washout time constant, the diffusion coefficient, and the Ostwald solubility coefficient for the liquid. The immediate post-bubble formation boundary conditions can range from uniform gas distribution to depletion of the inner most shells of the inert gas in the bubble but maintaining partial pressure equilibrium across the boundary. The simulation control parameters, grid size, and bubble growth cycle time were selected so as to assure the stability of the solution. The simulation results were considered stable when changes in either variable did not substantially alter the bubble evolution curves.

The values of the parameters used for the test runs are summarized in Table 1 by equation number and reference. Those values marked with a ^{PDE}, were used in the Monte Carlo - PDE comparison. The remainder of the physiological parameters were the same for

all comparisons. For those parameters marked "see text" in the reference column, the following provides the rationale for the values selected.

The values selected for V_{liq} were chosen for computational efficiency, but as close as possible to the physiological range. Francis et al. (31,32) reported bubble densities in cross sections of dog spinal cords ranging from 0.007 $\pm$ 0.017 bubbles/mm^2 to 0.251 $\pm$ 0.487 bubbles/mm^2. This corresponds to an intrabubble distance of approximately 10 mm to 2 mm. If these values are assumed to also represent the vertical intrabubble distance and bubbles are assumed to be uniformly distributed, the bubble density would range from 1 bubble/cm^3 to 125 bubbles/cm^3. Since some cord sections had 1 bubble/mm^2 the maximum density may extend to 1000 bubbles/cm^3. Low-density systems require more computer execution time because of the larger liquid volume they represent. Densities in the physiological range are impractical given our current computer technology. Consequently, we arbitrarily selected 2000 bubbles/cm^3 and 8000 bubbles/cm^3 even though such densities might exist only in extremely severe decompression sickness. These represent tissue volumes of 0.0005 and 0.000125 cm^3, respectively.

The pressure step from P_{amb0} (2 ATA) to P_{amb} (1 ATA) was selected arbitrarily.

The value for the initial bubble size (r_{bub}) for the equilibrium test was 6.5 microns. Ten to twenty percent of the bubbles with this starting radius shrink below the step size because of random fluctuations before they are able to reach a size close to the equilibrium radius. We increased the starting radius to 9.2 microns for the Monte Carlo and PDE comparison tests to avoid this problem.

Diffusion coefficients for nitrogen range from 0.6 x 10^{-5} cm^2/sec to 2.2 x 10^{-5} cm^2/sec

for tissue with water content between 65% and 100%. Tendon and bone are approximately 65% water, muscle is approximately 75% water, and brain is approximately 85% water with corresponding diffusion coefficients of about 0.6 x 10^{-5}, 0.75 x 10^{-5}, and 1.5 x 10^{-5} cm^2/sec (29,30). The choice of diffusion coefficient affects computer execution speed, i.e., in that slower diffusion requires less computer execution time. We ran the equilibrium test at 0.5 x 10^{-5} cm^2/sec, but to remain in a physiological range, we used a diffusion coefficient of 0.75 x 10^{-5} cm^2/sec for the Monte Carlo - PDE comparison tests.

The grid size for all tests was 2.5 microns, which corresponds to a diffusion step size, (a), of 4.3 microns. Diffusion step size in the Monte Carlo model and the grid size in the PDE model were selected to assure that smaller values would not change the solution.

The number of steps in a bubble growth cycle was set at 50 for both the equilibrium and Monte Carlo - PDE comparison tests. Because the diffusion coefficients were different for the two tests while the grid size remained the same, the time for a single step as calculated by Equation 26 was different. This results in bubble growth cycle time of 0.3125 seconds for the equilibrium test, but 0.21 seconds for the Monte Carlo - PDE comparison test.

Longer tissue wash-out times require more computer execution time, so we selected tissue times that were relatively fast but close to the physiologic range. This resulted in our choice of 2.5 and 10 minutes for the tests.

RESULTS

The critical (r_c) and equilibrium (r_e) radii for our system with a volume of 0.000125 cm^3 (8,000 bubbles/cm^3) and 0.0005 cm^3 (2,000 bubbles/cm^3) without mass transfer were 1.4

microns and 70.4 microns, respectively. Figure 9 demonstrates the model's ability to reach the predicted equilibrium radius (r_e) when the system is closed to mass transfer. A bubble with a radius greater than r_e shrinks to r_e and a bubble smaller than r_e grows to it.

The simulated inert gas wash-out without a bubble precisely matched the expected single exponential curve predicted by a well-stirred model.

The Monte Carlo and PDE predictions are compared in Table 2 and an illustrative condition is presented in Figures 10 and 11. The four combinations of bubble density and wash-out time referred to as conditions I-IV are presented in Table 2. Condition IV was not calculated for the PDE model because of the computational time required. Mean transit time and relative dispersion for condition II and the PDE model were not calculated because of difficulties encountered in fitting the wash-out curves with Equation 69. Values for the Monte Carlo mean r_{max} compare favorably with the PDE r_{max} for conditions I-III. Because of the variability in the Monte Carlo method, the remainder of the measures are not as closely matched. However, almost all fall within one standard deviation of the mean. In general, the predictions of the Monte Carlo and PDE models are similar as is graphically illustrated for condition I in Figures 10 and 11. Figures 12 and 13 show the variability of the model predictions according to the number of particles in each simulation for bubble radius verses time; Figure 12 has 2×10^8 particles/cm^3 and Figure 11 has 8×10^8 particles/cm^3. Figures 14 and 15 are the corresponding graphs for gas wash-out. They show the number of moles of gas in the system over time. Each figure contains curves from 10 separate simulations using the same starting conditions. The expected single exponential curves for inert gas wash-out without the presence of a bubble are shown for comparison in the bottom of Figures 14 and

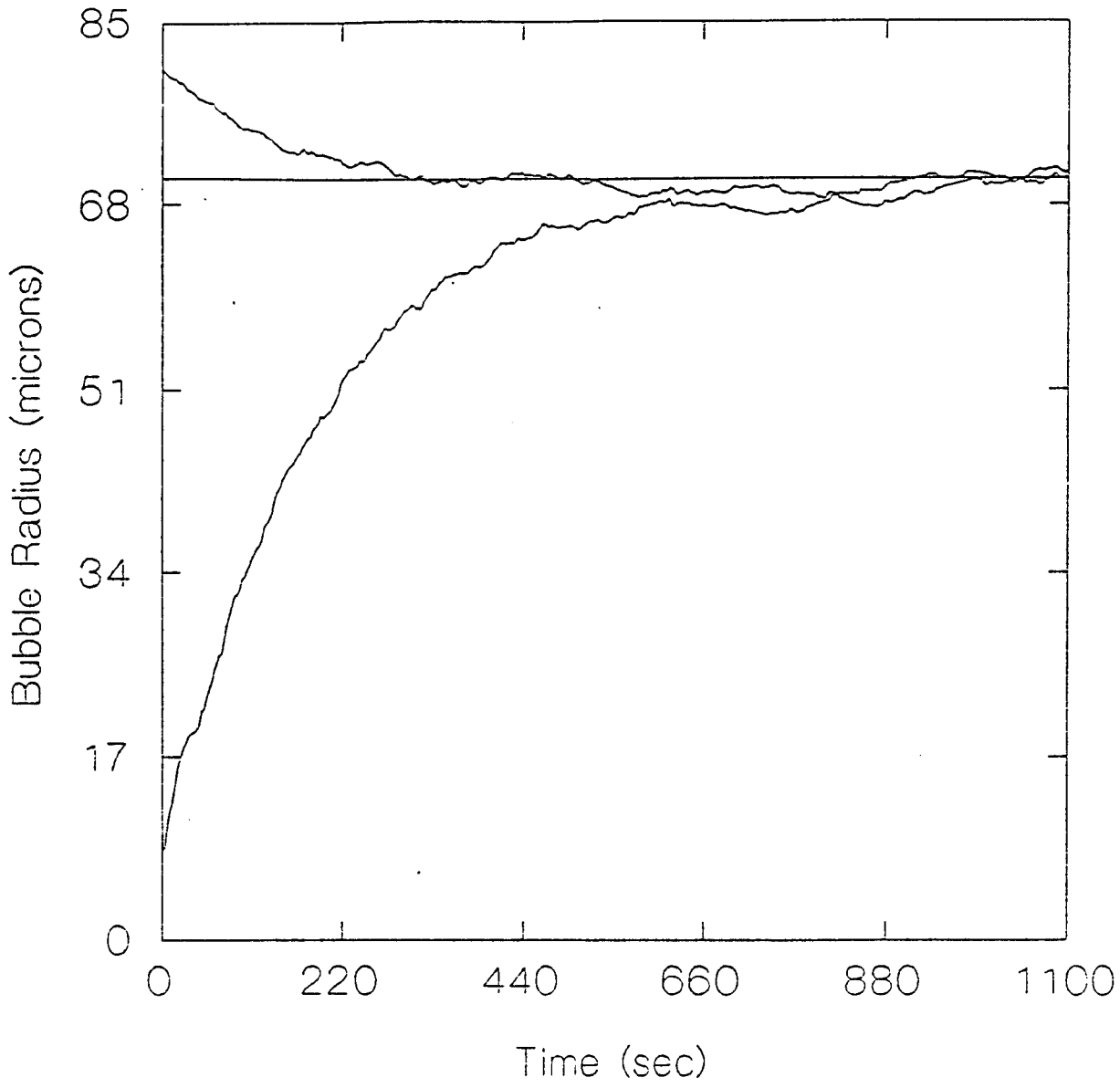

FIGURE 9 - Graph of simulation output for bubble evolution in a system closed to inert gas transport with initial bubble radii greater than and less than the expected equilibrium bubble radius. Tissue volume is 1.25×10^{-4} cm^3. Note that both simulations converge to the equilibrium radius predicted by the quartic polynomial in Appendix A (70.4 microns).

TABLE 2- Comparison of Monte Carlo and partial differential equation model predictions of the time course of bubble evolution

Condition	I	II	III	IV
Bubble Density (bubbles/cm^3)	8,000	8,000	2,000	2,000
Wash-out Time (sec)	150	600	150	600
critical radius (microns)	1.4	1.4	1.4	1.4
equilibrium radius (microns)	70.4	70.4	112.3	112.3
Monte Carlo Model				
number of runs	10	10	10	10
r_{max}[1] (microns)	43.0±2.7	61.7±2.1	42.8±5.3	80.3±1.5
t_{max}[1] (sec)	342.1±146.1	545.6±133.9	282.7±107.9	820.8±143.6
t_{dis}[1] (sec)	950.4±140.5	3799.6±349.4	771.3±203.3	4918.5±845.8
Mean Transit Time[1,5]	286.8±37.2[3]	2039.2±227.3[3]	177.1±17.4[3]	1403.0±164.7[4]
Relative Dispersion[1,5]	1.44±0.13[3]	1.16±0.04[3]	1.19±0.14[3]	1.0[4]
Partial Differential Equation Model				
r_{max} (microns)	40.5	59.5	43.2	*
t_{max} (sec)	236.0	504.3	277.2	*
t_{dis} (sec)	965.0	3930.0	1013.0	*
Mean Transit Time[2,3,5]	301.3±26.5	*	185.5±3.0	*
Relative Dispersion[2,3,5]	1.26±0.07	*	1.21±0.009	*

1. Mean ± s.d., coefficient of variation (s.d./mean)). 2. Estimated from 1 (1,2, and 1 runs respectively) and 2 exponential fits of simulated wash-out data. 3. Mean transit time for liquid without a bubble is 150 seconds with a relative dispersion of 1. 4. Estimated from 1-exponential fit of simulated wash-out data. 4. Estimated from 1-exponential fit of simulated wash-out data. 5. Mean transit time for liquid without a bubble is equal to the wash-out time. The relative dispersion is 1.Dissolution for the Monte Carlo model occurs when r_{bub} is 4.3 microns. * - not studied.

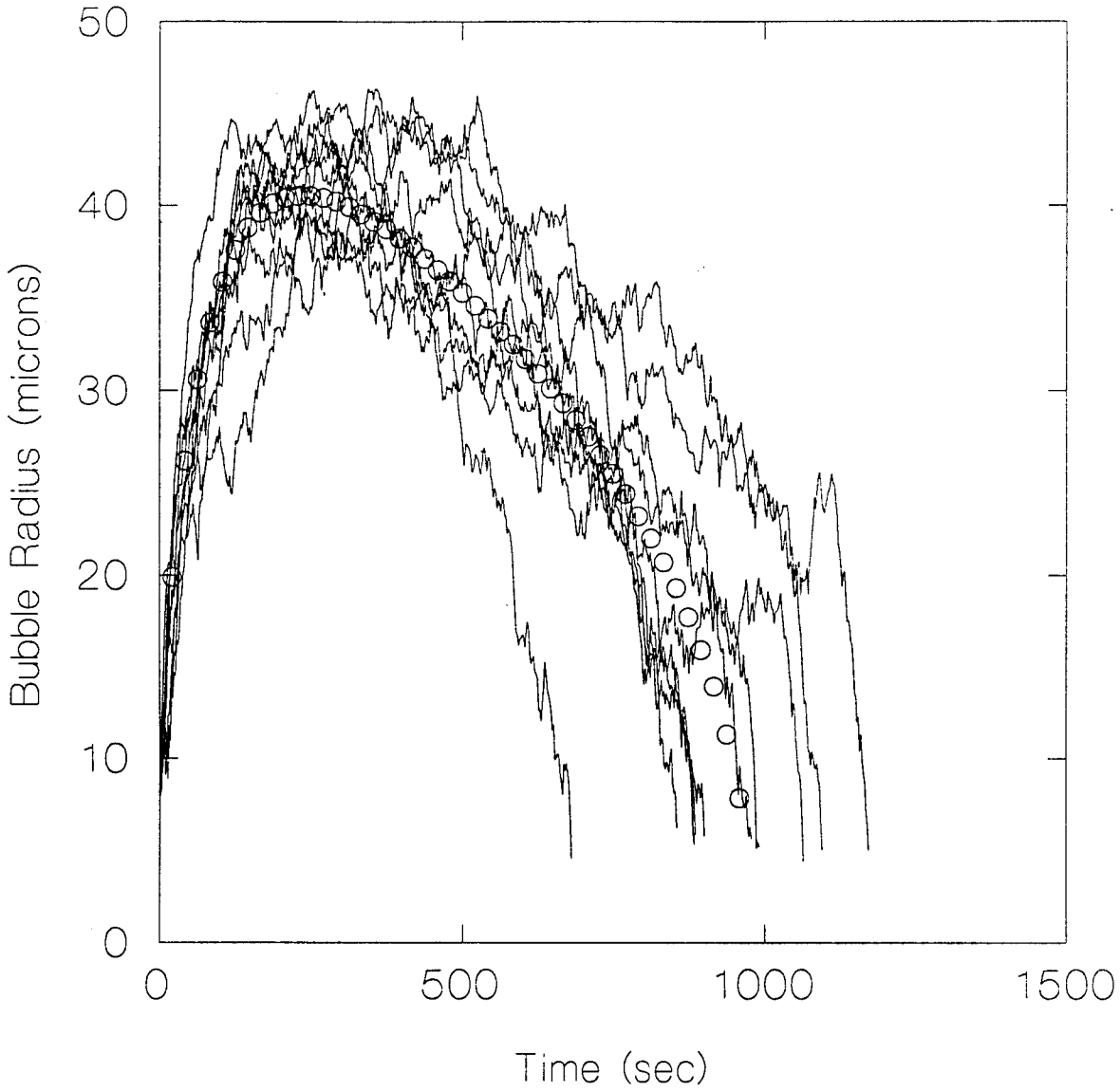

FIGURE 10 - Graph of Monte Carlo (▬) and PDE (O) predictions of bubble evolution time course for condition I.

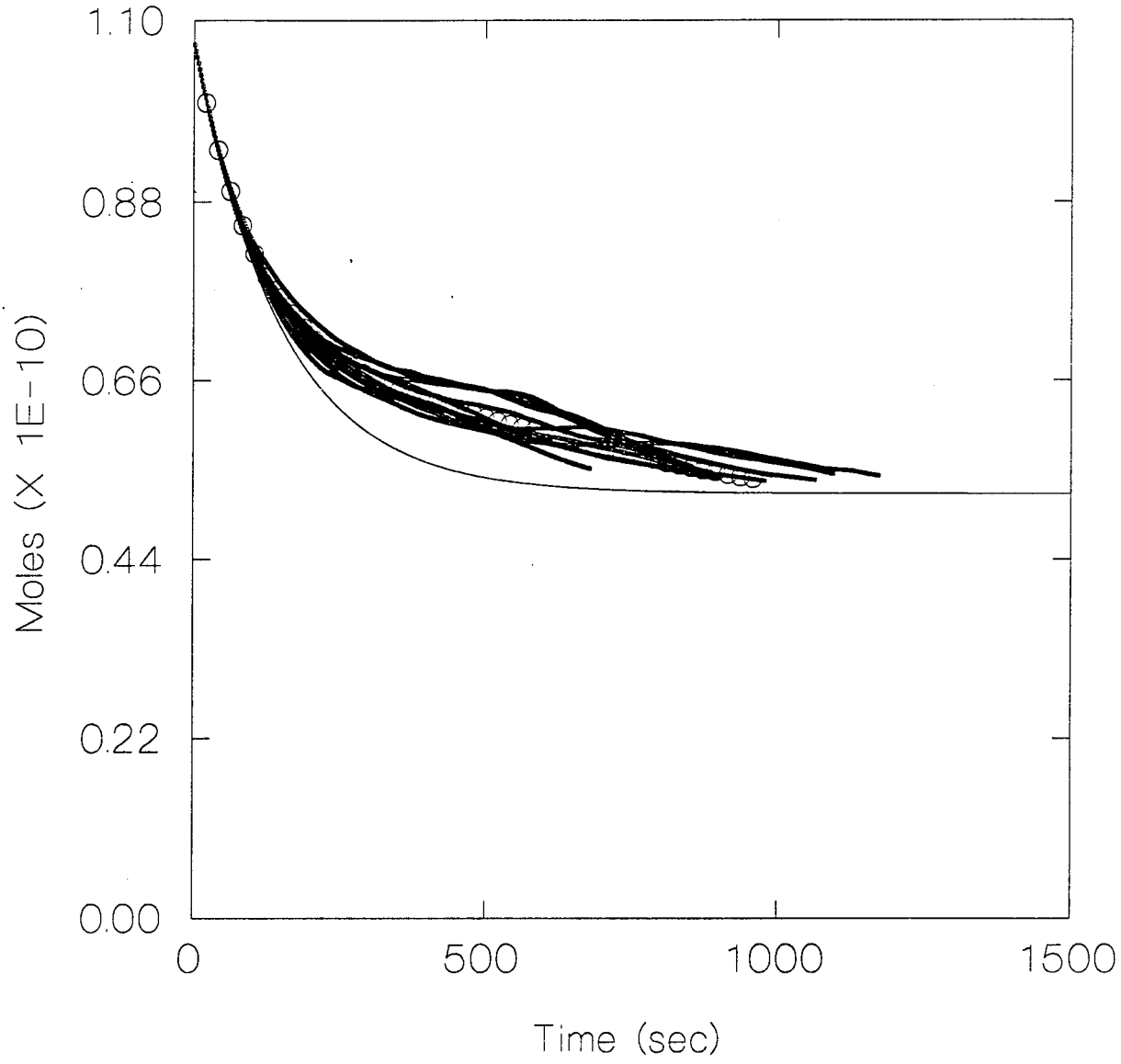

FIGURE 11 - Graph of Monte Carlo (▬) and PDE (O) predictions of gas wash-out versus time for condition I.

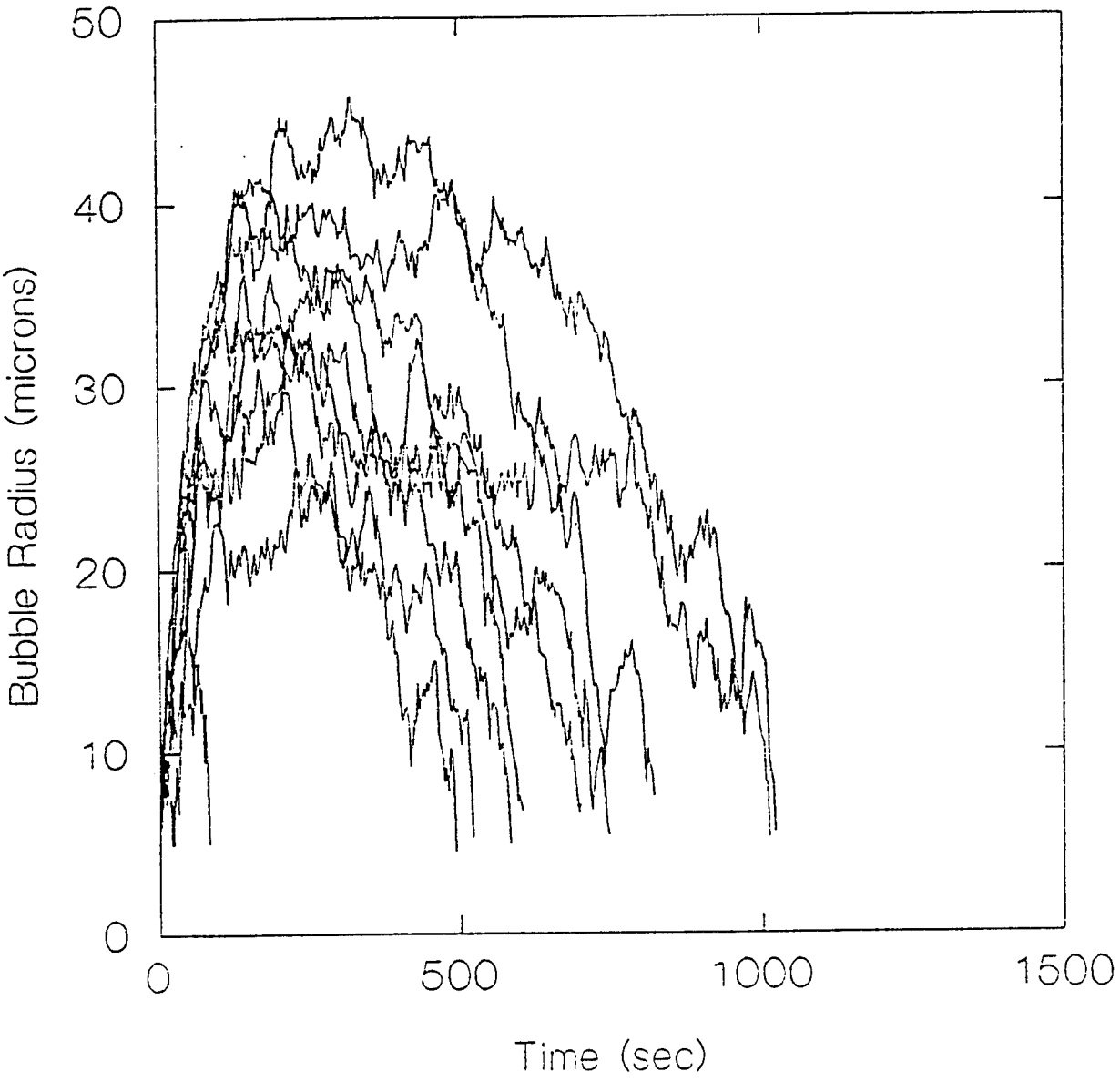

FIGURE 12 - Graph of bubble radius versus time for low precision simulation (2×10^8 particles/cm^3). Tissue volume is 1.25×10^{-4} cm^3.

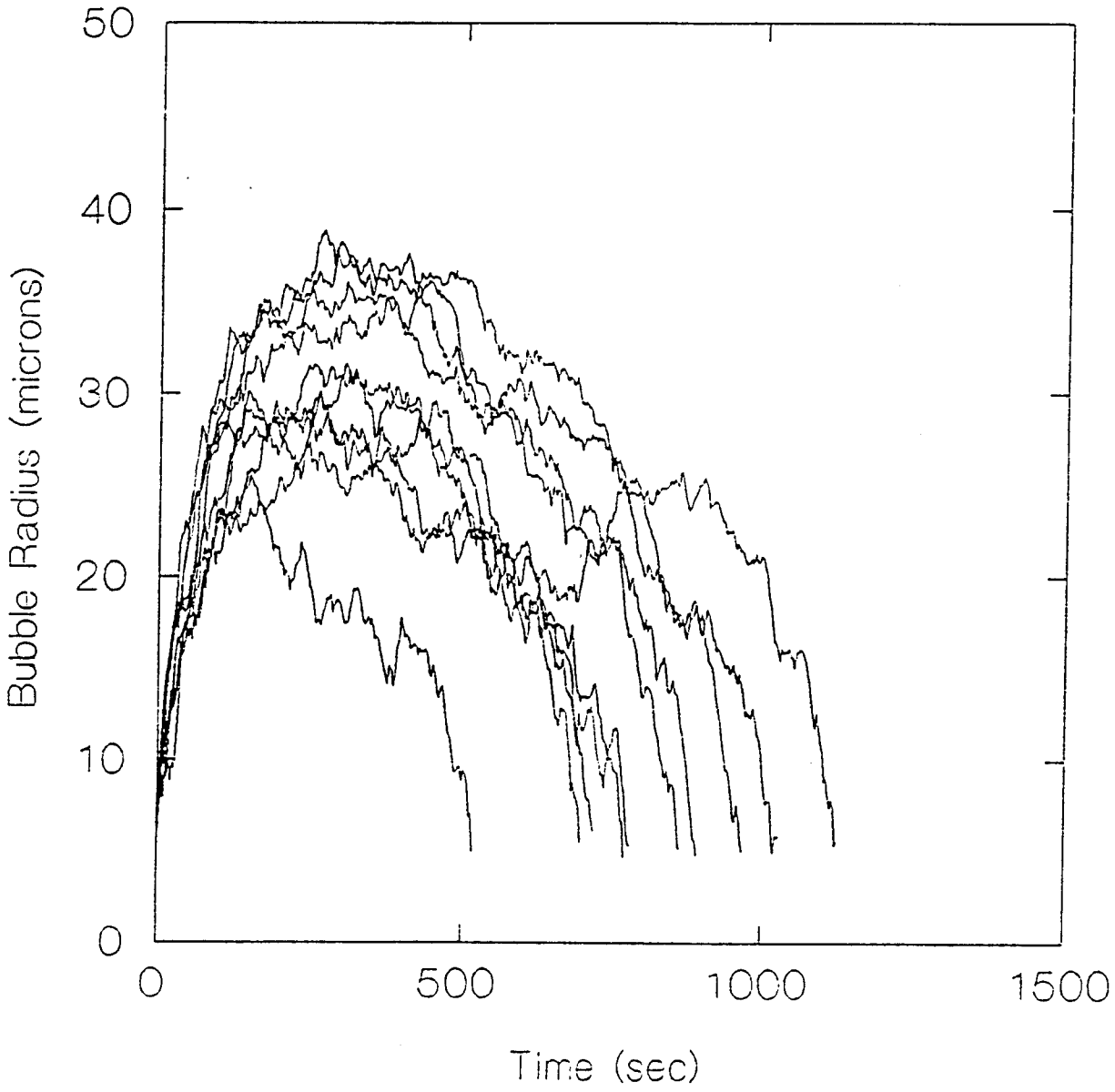

FIGURE 13 - Graph of bubble radius versus time for higher precision simulation (8×10^8 particles/cm^3). Tissue volume is 1.25×10^{-4} cm^3.

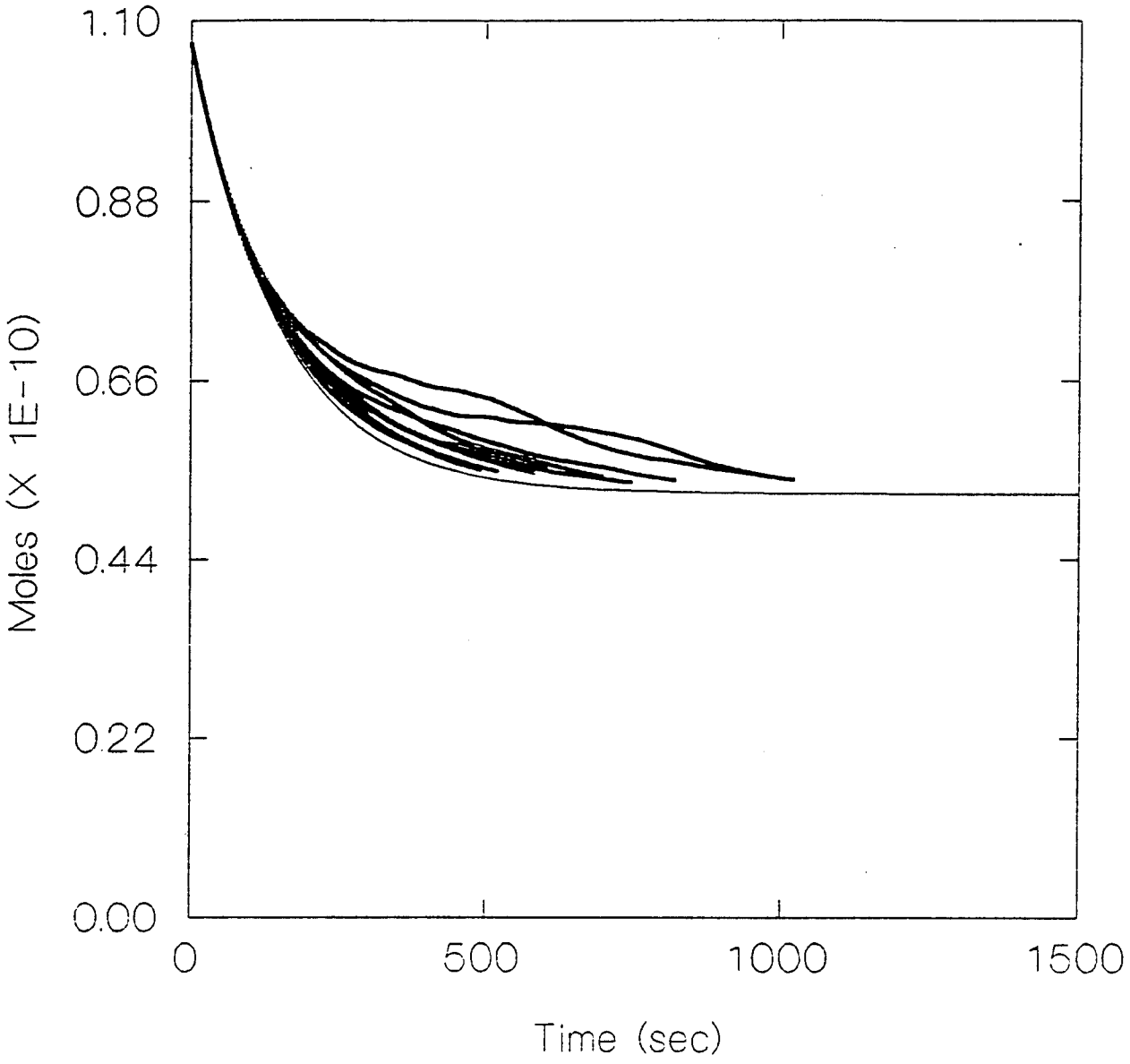

FIGURE 14 - Graph of gas wash-out versus time for low precision simulation (2×10^8 particles/cm^3). Thin line represents the expected wash-out for the tissue without a bubble. Tissue volume is 1.25×10^{-4} cm^3.

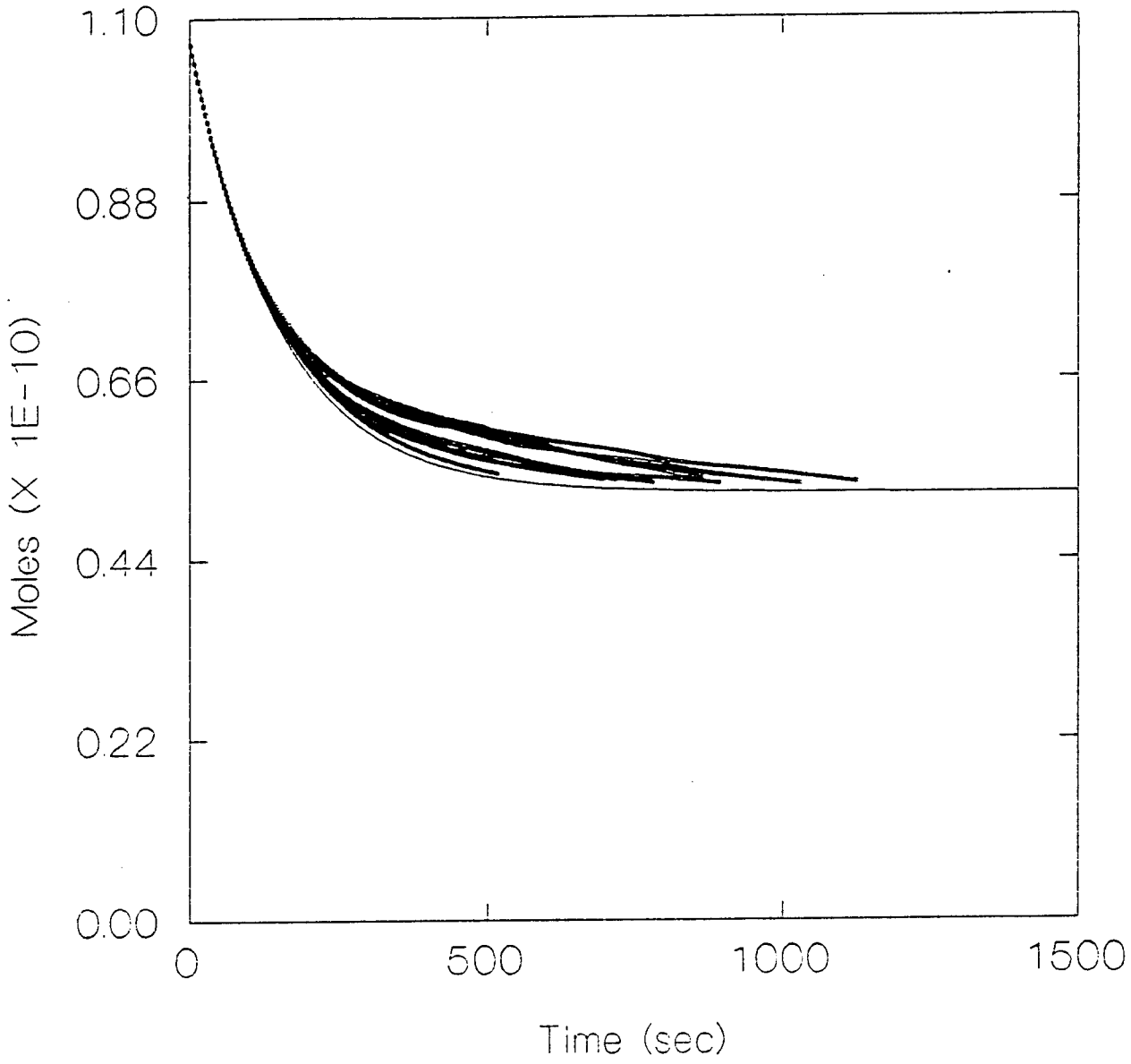

FIGURE 15 - Graph of gas wash-out versus time for higher precision simulation (8 x 10^8 particles/cm^3). Thin line represents the expected wash-out for the tissue without a bubble. Tissue volume is 1.25x 10^{-4} cm^3.

15. These figures illustrate the variability inherent in a method based on Monte Carlo methods, and the need for summary measures for comparing results.

The summary measures, mean $\pm$ S.D. and coefficients of variation (S.D./mean) of the maximum radius achieved (r_{max}), time to maximum radius (t_{max}), time to bubble dissolution (t_{dis}), mean transit time, and relative dispersion of the transit times, for 10 simulation runs at each of the three particle density precisions (2 x $10^8/cm^3$, 4 x $10^8/cm^3$, and 8 x $10^8/cm^3$) are presented in Table 3. In our test system, 10 - 20 % of the bubbles with a starting radius of 6.5 microns shrink below the step size before they are able to reach a size close to the equilibrium radius. As a consequence, the results are biased towards fast-growing bubbles, although the size of the bias is probably small.

These results illustrate that increasing particle density in a bubble growth cycle does not affect the precision of outcome measures equally; since each particle's transit through the module is independent, we might expect that the precision of the outcome measures improves proportionally to the square root of the relative increase in particle density. This would imply that the coefficient of variation (cv) for the highest particle density run in Table 3 would be half the cv for the lowest density run. While r_{max}, t_{max} and t_{dis} approximate this expected improvement, the mean transit time and relative dispersion of the transit times do not. Since precision for all measures will improve proportionally with the square root of the number of complete runs, multiple runs at low particle density may be a more efficient use of computer resources than a few high precision runs for estimating the mean transit time and the relative dispersion. More exhaustive testing will be required to determine this definitively.

TABLE 3: Precision of Outcome Measures as a Function of Particle Density

	Particle Density		
	2 x 10^8/cm^3 (n=10)	**4 x 10^8/cm^3 (n=10)**	**8 x 10^8/cm^3 (n=10)**
r_{max}[1] (μ)	34.5±7.6, 0.22	33.3±4.6, 0.14	32.5±4.7, 0.14
t_{max}[1](s)	228.3±127.2, 0.56	271.9±122.2, 0.45	280.6±83.8, 0.30
t_{dis}[1](s)	657.1±273.4, 041	793.4±274.6, 0.35	838.1±177.4,0.21
Mean Transit Time[1,2,3]	204.3±45.3, 0.22	196.9±25.4, 0.13	200.4±28.9, 0.14
Relative Dispersion[1,2,3]	1.21±0.18, 0.15	1.23±0.17, 0.14	1.27±0.17, 0.13

1. Mean ± s.d., coefficient of variation (s.d./mean)).
2. Estimated from 1 (1,2, and 1 runs respectively) and 2 exponential fits of simulated wash-out data.
3. Mean transit time for liquid without a bubble is 150 seconds with a relative dispersion of 1.4. Estimated from 1-exponential fit of simulated wash-out data.
4. Estimated from 1-exponential fit of simulated wash-out data.
5. Mean transit time for liquid without a bubble is equal to the wash-out time. The relative dispersion is 1. Dissolution for the Monte Carlo model occurs when r_{bub} is 4.3 microns.

The program execution speed is directly proportional to the number of particles simulated. Since larger volumes require more particles to obtain the same number of bubble-liquid transitions, larger volumes require longer execution speeds. The relationship among the number of steps taken in a bubble growth cycle, the grid size and volume and execution speed is more complex since all these factors influence the proportion of particles that will be able to take a bigstep in the liquid region.

DISCUSSION

Our ultimate goal is the development of a physiologically based method of constructing decompression schedules. Since the effect of extravascular bubbles on inert gas exchange must be included in such an endeavor, we set out to develop a method for constructing models of bubble evolution during decompression that would allow for previously excluded details of the physiological environment such as complex microvascular architectures.

As a first step in this direction, this report introduces a model of bubble evolution during decompression in a homogenous liquid based on a Monte Carlo simulation of diffusion. The major advantage of this method is that in theory, details of the biological environment can be incorporated that are not feasible using traditional modeling methods. The major disadvantage is that Monte Carlo simulations require large amounts of computer execution time. Other disadvantages include the inherent variability in the model predictions, which requires multiple runs and the use of summary measures for comparisons, and the dependence of execution time on the volume of the liquid being simulated. Limitations of the current simplified model are the assumptions of liquid homogeneity, uniform gas wash-out,

uniform bubble size and distribution, radial symmetry, and the use of a spherical region which is not space filling.

Two features of this approach are novel to Monte Carlo models of diffusion and contribute to its relative efficiency. First, use of the probability distribution of the distance a particle travels in the liquid phase to replace many single diffusion steps dramatically increases the program's execution speed. It also invites the development of other distributional approaches. Second, treating the bubble as a high solubility region allows us to model gas particle flux without developing a detailed kinetic model of particle behavior inside the bubble.

The results presented in this report confirm the fact that a Monte Carlo approach can produce the equilibrium results expected from independent calculations and compares favorably with the predicted time course of bubble evolution predicted by a PDE model.

Future model development could progress in several ways. Effort at improving the efficiency of the algorithm could result in execution times that are competitive with traditional methods. One approach is to attempt to generalize the probability distribution function so as to calculate in advance a distribution that includes both diffusion and transition into and out of the bubble. Sampling from the distribution of particle residence times in the bubble could also improve efficiency. The final step would be to develop a single distribution which would combine all of these elements. For the special case of the spherically symmetric bubble-liquid module, the problem can be reduced to a one dimensional random walk. The theoretical background for these approaches is outlined in Appendix D.

Moving the program to a computer capable of parallel or vector processing might also

improve execution speed since each particle's path is independent of every other particle and could be calculated separately (33).

In the current implementation we treat metabolic gases as constants for simplicity. Alternatively, each type of gas could be treated stochastically based on its solubility, tissue time and diffusion coefficient and tracked separately through a bubble growth cycle. An additional "metabolic conversion" probability would need to be applied to account for the consumption of oxygen and production of carbon dioxide. Multiple inert gases could also be handled individually. Multiple gas simulations are restricted by execution speed since each new gas requires the addition of an equal number of particles to maintain precision. So a model with nitrogen, helium and oxygen would require three times as many particles as the simulations in this report.

The influence of bubble interactions may be another area of study. This would require a further partitioning of the liquid region in order to more precisely record the gradient in the liquid. Since each liquid region requires a minimum number of particles to ensure a large enough number of particle transitions between regions in each bubble growth cycle, this approach would also require considerably more execution time. A similar approach could be taken to include details of the tissue micro-architecture. Incorporation of a symmetrical, space-filling liquid region would allow generalization from one liquid module to an entire liquid slab without concern about errors introduced by the use of non space-filling liquid regions currently employed (34).

Ultimate model testing depends on experiments conducted in *in vitro* and *in vivo* systems. If experimental methods with sufficient resolving power are developed, direct

testing could be conducted by comparing predicted r_{max}, t_{max} and t_{dis} with experimental results. An indirect approach could be taken with present technology by comparing predicted transit time measures with those observed in animal experiments conducted with radiolabeled gases (24,25).

REFERENCES

1. Weathersby, P.K., Homer, L.D., Parker, E.C., Thalmann, E.D., "Predicting the time of occurrence of decompression sickness," Journal of Applied Physiology, Vol. 72, pp. 1541-1548, 1992.

2. Parker, E.C., Survanshi, S.S., Weathersby, P.K., Thalmann, E.D., *Statistically based decompression tables VIII: Linear-exponential kinetics*, NMRI No. 92-73, Naval Medical Research Institute, Bethesda, MD, 1992.

3. Weathersby, P.K., Survanshi, S.S., Nishi, R.Y., Thalmann, E.D., *Statistically Based Decompression Tables VII: Selection and Treatment of Primary Air and N_2O_2 Data.* NSMRL No. 1182 and NMRI No. 92-85, Naval Submarine Medical Research Laboratory, Groton, CT; Naval Medical Research Institute, Betheda, MD, 1992.

4. Thalmann, E.D., *Phase II testing of decompression algorithms for use in the U.S. Navy underwater decompression computer*, NEDU No. 1-84, Navy Experimental Diving Unit, Panama City, FL, 1984.

5. Thalmann, E.D., *Development of a decompression algorithm for constant 0.7 ATA oxygen partial pressure in helium diving*, NEDU No. 1-85, Navy Experimental Diving Unit, Panama City, FL, 1985a.

6. Thalmann, E.D., *Air-N_20_2 decompression computer algorithm development*, NEDU No. 8-85, Navy Experimental Diving Unit, Panama City, FL, 1985b.

7. D'Aoust, B.G., Smith, K.H., Swanson, H.T., "Decompression-induced decrease in nitrogen elimination rate in awake dogs," *Journal of Applied Physiology*, Vol. 41, pp. 348-355, 1976.

8. Hills, B.A., "Effect of decompression per se on nitrogen elimination," *Journal of Applied Physiology: Respiration, Environmental, and Exercise Physiology*, Vol. 45, pp. 916-921, 1978.

9. Kindwall, E.P., Baz, A., Lightfoot, E.N., Lanphier, E.H., Seirig, A. , "Nitrogen elimination in man during decompression," *Undersea Biomedical Research*, Vol 2, pp 285-297, 1975.

10. Homer, L.D., Weathersby, P.K.. "How well mixed is inert gas in tissues?" *Journal of Applied Physiology*, Vol. 60, pp. 2079-2088, 1986.

11. Homer LD, Weathersby PK, Survanshi S. How countercurrent blood flow and uneven perfusion affect the motion of inert gas. J Appl Physiol 69:162-170, 1990.

12. Himm, J., Homer, L.D., Novotny, J.A., "The effect of lipid on xenon kinetics," *Journal of Applied Physiology* (in press).

13. Epstein, P.S., Plesset , "On the stability of gas bubbles in liquid-gas solutions," *Journal of Chemistry and Physics*, Vol. 18, pp. 1505-1509, 1950.

14. Law, A.M., Kelton, W.D., *Simulation Modeling and Analysis*, McGraw-Hill, 1982.

15. Weathersby, P.K., Homer, L.D., Flynn, E.T., "Homogenous nucleation of gas bubbles in vivo," *Journal of Applied Physiology*, Vol. 53, pp. 940-946, 1982.

16. Tikuisis, P,. "Modeling the observations of in vivo bubble formation with hydrophobic crevices," *Undersea Biomedical Research*, Vol. 13, pp. 165-180, 1986.

17. Ward, C.A., Tikuisis, P., Venter, R.D., "Stability of bubbles in a closed volume of liquid-gas solution," *Journal of Applied Physics*, Vol. 53, pp. 6076-6084, 1982.

18. Tikuisis, P., Ward, C.A., Venter, R.D., "Bubble evolution in a stirred volume of liquid closed to mass transport," *Journal of Applied Physics*, Vol. 54, pp. 1-9, 1983.

19. Wilks, S.S., *Mathematical Statistics*, John Wiley and Sons, New York, 1962.

20. Crank, J., *Mathematics of Diffusion*, Oxford University Press, 1975, p 29.

21. Cussler, E., *Diffusion mass transfer in liquid systems*. Section 5.1, Cambridge University Press, New York, 1984.

22. West, .JB., *Respiratory physiology-The essentials*, Williams and Wilkins, 1979

23. Weathersby, P.K., Barnard, E.E.P., Homer, L.D., Mendenhall, K.G., "A stochastic description of inert gas exchange.," *Journal of Applied Physiology*, Vol. 47, pp. 1263-69, 1979.

24. Weathersby, P.K., Mendenhall, K.G., Barnard, E.E.P., Homer, LD., Survanshi, S., Vieras, F., "Distribution of xenon exchange rates in dogs," *Journal of Applied Physiology*, Vol. 50, pp. 1325-1336, 1981.

25. Novotny, J.A., Mayers, D.L., Parsons, Y.J., Survanshi, S.S., Weathersby, P.K., Homer, L.D., "Xenon kinetics in muscle are not explained by a model of parallel perfusion-limited compartments," *Journal of Applied Physiology*, Vol. 68:, pp. 76-890, 1990.

26. *SYSTAT: The system for statistics*, Evanston, IL: SYSTAT, Inc, 1990.

27. Van Liew, H.D., "Simulation of the dynamics of decompression sickness bubbles and the generation of new bubbles," *Undersea Biomedical Research*, Vol. 18, pp. 333-345, 1991.

28. Weathersby, P.K., Homer, L.D., "Solubility of inert gases in biological fluids and tissues: a review," *Undersea Biomedical Research*, Vol. 7, pp. 277-296, 1980.

29. Varysel, P., "Effect of percentage water content in tissues and solids on the diffusion coefficients of O_2, CO_2, and N_2," *Pfluegers Archives*, Vol. 361, pp. 201-204, 1976.

30. Allen, T.H., Krzywicki, H.J., Roberts, J.E., "Density, fat, water and solids in freshly isolated tissues," *Journal of Applied Physiology*, Vol. 4, pp. 1005-1008, 1959.

31. Francis, T.J.R., Pezeshkpour, G.H., Dutka, J., Hallenback, J.M., Flynn, E.T., "Is there a role for autochthonous bubbles in the pathogenesis of spinal cord decompression sickness?" *Journal of Neuropathology and Experimental Neurology*, Vol. 47, pp. 475-487, 1988.

32. Francis, T.J.R., Griffin, J.L., Homer, L.D., Pezeshkpour, G.H., Dutka, A.J., Flynn, E.T., "Bubble-induced dysfunction in acute spinal cord decompression sickness," *Journal of Applied Physiology,* Vol. 68, pp. 1368-1375, 1990.

33. *International youth workshop on Monte Carlo methods and parallel algorithms*, Sendov, B.H., Dimov, I., eds., World Scientific, 1989.

34. Theodorou, D.N., Suter, U.W., "Geometrical considerations in model systems with periodic boundaries," *Journal of Chemistry and Physics*, Vol. 82, pp.955-966, 1985.

35. Pollack, G.L., Kennan, R.P., Himm, J.F., Carr, P.W., "Solubility of xenon in 45 organic solvents including cycloalkanes, acids and alkanals: Experiment and theory," *Journal of Chemistry and Physics*, Vol. 90, pp. 6569-6579, 1989.

APPENDIX A: Determination of Equilibrium Radii

The polynomial (17,18) which describes the 2 equilibrium radii in a closed system with one inert gas is:

$$a_4 \cdot r^4 + a_3 \cdot r^3 + a_1 \cdot r + a_0 = 0 \qquad \text{(A1)}$$

where:

$$a_4 = 4\Pi \cdot P(1 - P_{vap}/P)/3 \cdot c_s \cdot k \cdot T$$
$$a_3 = 8\Pi \cdot \gamma/3 \cdot c_s \cdot k \cdot T$$
$$a_1 = V(1 - P_{vap}/P) - N/c_s$$
$$a_0 = 2\gamma \cdot V/P$$

and

r = bubble equilibrium radii, (cm)

P = ambient inert gas partial pressure after pressure step, (mmHg)

P_{vap} = vapor pressure of water, (constant), (mmHg)

c_s = saturation concentration of inert gas in the solvent, ($moles/cm^3$)

For weak solutions $c_s = c_l \cdot P/K_H$, where:

c_l = molar concentration of the solvent, ($moles/cm^3$)

K_H = Henry's coefficient = $P_H/([(k \cdot T/(V_L \cdot \alpha \cdot P_H)) + 1]^{-1})$, (mmHg) (35)

where:

V_L = molar volume of solvent ($1/c_l$), $cm^3/mole$

α = Ostwald solubility coefficient

P_H = 760 mmHg

k = Universal gas constant in (cm^3 mmHg)/(mole K)

T = temperature in Kelvin

γ = surface tension, (mmHg-cm)

V = volume of depleted region (inverse of bubble density), (cm^3)

N = number of moles of inert gas in the volume V at c_s, (moles)

APPENDIX B: Derivation of Initial Particle Placement Coordinates

We need to create the desired distribution from a uniform random variable between 0 and 1 generated by the computer. Because our volume is a sphere, we begin in spherical coordinates ρ, θ and ϕ where ρ is the radial coordinate, θ the azimuthal angle between 0 and π, and ϕ orthogonal to θ, between 0 and 2π. These variables are mutually independent, therefore we must generate a distribution for each of them:

Let **U** be a uniform random variable on [0,1], with the probability density function (PDF) f(u) and the cumulative distribution function (CDF) F(u), where u is a particular value of **U**. Let ρ, θ and ϕ be random variables with PDF's g_ρ, g_θ and g_ϕ and CDF's G_ρ, G_θ and G_ϕ such that ρ, θ and ϕ are distributed uniformly by volume. For illustrative purposes we will demonstrate the procedure in detail for ρ, although similar expressions can be written for the other variables, θ and ϕ. By definition,

$$Prob\{\rho_0 \le \rho \le \rho_1\} = G_\rho(\rho_1) - G_\rho(\rho_0) = \frac{volume\ of\ a\ sphere\ with\ \rho_0 \le \rho \le \rho_1}{total\ volume\ of\ the\ sphere} \quad \text{(B1)}$$

where ρ_0 and ρ_1 are arbitrarily selected values.

We need to find the monotonic transformations $h_\rho(u)$, $h_\theta(u)$ and $h_\phi(u)$, such that $\rho = h_\rho(u)$, $\theta = h_\theta(u)$ and $\phi = h_\phi(u)$. So for example, we must find $h_\rho(u)$ such that,

$$Prob\{\rho_0 \le h_\rho(u) \le \rho_1\} = \frac{volume\ of\ a\ sphere\ with\ \rho_0 \le \rho \le \rho_1}{total\ volume\ of\ the\ sphere} = \frac{\frac{4\pi}{3}(\rho_1^3 - \rho_0^3)}{\frac{4\pi}{3}R^3} \quad \text{(B2)}$$

where R is the radius of the sphere.

To generate the desired distribution from the uniform distribution, we require, $\text{Prob}\{\rho \le h_\rho(u)\} = \text{Prob}\{U \le u\}$ which by definition of a CDF implies that $G_\rho(h_\rho(u)) = F(u)$.

$$G_\rho(h_\rho(u)) = F(u) \tag{B3a}$$

Differentiating both sides of (B3a) and applying the chain rule gives,

$$g_\rho(\rho) \cdot h'_\rho(u) = f(u) \tag{B3b}$$

By substituting $h_\rho^{-1}(\rho)$ for u and recognizing that $f(u)=1$ on [0,1], we obtain,

$$g_\rho(\rho) = \frac{1}{h'_\rho(u)} = \frac{1}{h'_\rho(h_\rho^{-1}(\rho))} \tag{B4}$$

Integrating gives,

$$G_\rho(\rho) = \int \frac{1}{h'_\rho(h_\rho^{-1}(\rho))} d\rho \tag{B5}$$

Combining equations B1, B2, and B5 for ρ we obtain,

$$\int_{\rho_0}^{\rho_1} \frac{1}{h'(h^{-1}(\rho)} d\rho = \frac{\frac{4\pi}{3}(\rho_1^3 - \rho_0^3)}{\frac{4\pi}{3} R^3} \tag{B6}$$

Taking the derivative with respect to ρ_1 yields,

$$\frac{1}{h'_\rho(h_\rho^{-1}(\rho))} = \frac{3\rho_1^2}{R^3} \tag{B7}$$

which is equivalent to,

$$\frac{1}{h_\rho'(u)} = \frac{3h_\rho^2(u)}{R^3} \tag{B8}$$

since $h_\rho(u)=\rho$ by definition. We can rearrange to obtain the differential equation,

$$\frac{dh_\rho}{du} = \frac{R^3}{3h_\rho^2(u)} \tag{B9}$$

which can be integrated to obtain the expression,

$$h_\rho = Ru^{\frac{1}{3}} \tag{B10}$$

Recognizing that for θ,

$$\frac{\textit{volume of a sphere with } \theta_0 \le \theta \le \theta_1}{\textit{total volume of the sphere}} = \frac{\frac{2\pi}{3}R^3(-\cos(\theta_1)+\cos(\theta_0))}{\frac{4\pi}{3}R^3} \tag{B11}$$

and for ϕ,

$$\frac{\textit{volume of a sphere with } \phi_0 \le \phi \le \phi_1}{\textit{total volume of the sphere}} = \frac{\frac{2}{3}R^3(\phi_1-\phi_0)}{\frac{4\pi}{3}R^3} \tag{B12}$$

similar approaches can be taken to obtain results for these other variables. Solving for h in each case we obtain,

$$h_\theta(u) = \theta = \arccos(1-2u) \tag{B13}$$

and

$$h_\phi(u) = \phi = 2\pi u \tag{B14}$$

APPENDIX C: Approaches to Increasing Execution Speed

1. Distribution of Particle Residence Times in the Bubble

The probability of the particle being available to exit the bubble after n steps can be calculated. During a single step the particle has one chance of being available to exit, so the probability of being available after n steps can be calculated as,

$$P_n = p_{exit}(1+(1-p_{exit})+(1-p_{exit})^2+\ldots+p_{exit}(1-p_{exit})^{n-1}) \tag{C1}$$

which can be expressed as,

$$P_n = p_{exit}\cdot\frac{1-(1-p_{exit})^n}{1-(1-p_{exit})} \tag{C2}$$

From this we can solve for $(1-p_{exit})^n$,

$$(1-p_{exit})^n = 1-\frac{P_n}{p_{exit}}(1-(1-p_{exit})) \tag{C3}$$

then applying logarithms we obtain,

$$n = \frac{\ln(1-\frac{P_n}{p_{exit}}(1-(1-p_{exit})))}{\ln(1-p_{exit})} \tag{C4}$$

Since $p_{exit}=1-(1-p_{exit})$,

$$n=\frac{\ln(1-P_n)}{\ln(1-p_{exit})} \tag{C5}$$

so that if we randomly generate P_n between 0 and 1, the number of steps before the particle is available can be calculated.

Alternatively, for $p_{exit} \leq 0.075$ the geometric distribution can be approximated by the exponential distribution (21) so that,

$$n = -\frac{1}{p_{exit}} \cdot \ln(u) \qquad \text{(C6)}$$

where u is a uniform random variable.

If the number of steps generated from the distribution is greater than the number of steps in a bubble growth cycle, n_{cycle}, the particle remains unavailable to exit during the bubble growth cycle. If it is less than n_{cycle} then for $(n_{cycle} - n)$ the particle takes individual steps.

This approach can be applied if p_{exit} is defined as described in the main body of the text or if the additional p_{out} factor were included.

2. Distribution of particles outside the bubble

If x is the random variable that describes the distance outside the bubble boundary, we can obtain an expression for the distribution of the particles outside the bubble by integrating $P(t \mid g)$ with respect to z and normalizing the result by the expression for p_{exit},

$$H(x) = \frac{\int_o^x \frac{3\alpha[2a(R+z)-(a^2+2Rz+z^2)][R+z]}{4aR^3} dz}{\alpha \frac{a(12R^2-a^2)}{16R^3}} \qquad \text{(C7)}$$

which reduces to,

$$H(x) = \frac{-x(3x^3-8ax^2+12Rx^2+12xR^2-24axR+6a^2x+12Ra^2-24aR^2)}{a^2(12R^2-a^2)} \qquad \text{(C8)}$$

3. One dimensional random walk for a spherically symmetric bubble-liquid module

The algorithm described in the main body of this report is general enough to accommodate a heterogeneous, asymmetric diffusion environment. However, a spherically symmetric bubble-liquid module can be studied as a first approximation to a more complicated geometry. Because of the symmetry in this environment, the three-dimensional random walk can be reduced to a one-dimensional random walk in order to increase the efficiency of the algorithm. A subroutine FASTSTEP has been developed that implements this one-dimensional algorithm.

Subroutine FASTSTEP uses the law of cosines to generate a new position r' for a particle initially at a distance r from the origin, and stepping a distance (a) at an angle ϕ to the radius vector, as shown in Figure 14. The equation giving r' is then,

$$r'^2 = a^2+r^2-2\cdot a\cdot r\cdot\cos(\phi) \tag{C9}$$

where the cosine term is randomly generated from a uniform random distribution by,

$$\cos(\phi) = 1 - 2\cdot RAN(I1) \tag{C10}$$

and where I1 is a seed for the random number generating function RAN(x).

Using the geometry of Figure 15, subroutine FASTREFL takes particles that have stepped out of the system and return them back into the system using reflection from the outer boundary. In Figure 15, a particle initially at point P steps to point P', which is at a distance greater than R, the radius of the system. In order to reflect the particle from the outer boundary to point P'', FASTREFL:

(1) Uses the quantities r, R, and cos(ϕ) (which is returned from subroutine FASTSTEP) and the law of cosines to find distance d_1,

$$d_1^2 = r^2+R^2-2\cdot r\cdot R\cdot\cos(\phi) \tag{C11}$$

(2) Uses the law of sines to determine angle Θ,

$$\sin(\theta) = \frac{r}{R}\cdot\sin(\phi) \tag{C12}$$

(3) Applies the law of cosines a final time to determine r'',

$$r''^2 = d_2^2+R^2-2\cdot d_2\cdot R\cdot\cos(\theta) \tag{C13}$$

where $d_2 = a - d_1$.

APPENDIX D: Fortran Source Code

The program was written using a Fortran compiler for Avalon accelerator boards installed in a Vax 3800. The Fortran source code for the program is listed in Section D1. In order to execute the program, in addition to the executable code, an input file such as the example in Section D2, and a file "DISTPROB.DAT" (which contains the values of the diffusion probability distribution function obtained from numerical integration) are required. This file is produced by the program listed in Section D3. An additional binary file, "BUBMOD.DMP", must also exist prior to execution. This file contains information on the gradient that can be used to restart the program in the middle of a run. The statement, iprofile=1 in the input file tells the program to use the information in the file. The output file created by the program contains information on the time course of the bubble radius. A program listed in Section D4 can create alternative initial boundary value conditions for the liquid surrounding the bubble. Section D5 contains subroutines that can be used to implement the one-dimensional random walk version of the program.

Section D1. FORTRAN source code for Monte Carlo bubble simulation.

```
C-  This program simulates bubble growth beginning with a bubble of
C-  critical radius as defined by Tikuisis.  The bubble is represented as
C-  a region of high solubility surrounded by a "depleted region" from
C-  which the inert gas is extracted.  These two regions are placed in
C-  a third region which represents the surrounding tissue.

      implicit real*8  (a-h,o-z)
       common /com1/xmin,xmax,ymin,ymax,zmin,zmax,rbub,rdep
      common/com2/pi,i1
       common/distarr/distance(10000)
       common/sincos/fact,scale,sinmat(0:9999),cosmat(0:9999)
      common/com3/IA(3,24),aa(3,24)
       common /com5/twopi,onethird
      real*8 N2,N2dep,k,kH,molezone(500)
      Dimension xx(3),rzonebeg(0:500),rzoneend(0:500),widezone(500)
      dimension strtzone(500),endzone(500),fraczone(500),conczone(500)
      dimension cummsum(0:500),vzone(500),rzsq(0:500)
      data IA/1,1,1,1,1,-1,1,-1,1,1,1,-1,-1,-1,1,1,1,-1,1,1,-1,-1,-1,
     +  1,-1,-1,-1,
     +  1,1,1,-1,1,1,1,1,-1,1,-1,-1,1,1,1,1,-1,-1,1,-1,1,1,-1,-1,-1,-1,-1,
     +  1,1,1,1,1,1,-1,-1,1,1,-1,1,-1,1,-1,1,1,1,-1,-1,-1,-1,1,-1,-1,-1,-1/
C- Define constants
       zed=0.d0
      Pi=4.d0*datan(1.d0)
       twopi=2.d0*pi
       scale=10000.d0
       fact=scale/twopi
       call gensine
c
c    change 12/16/92 to include oxygen and co2 as additional gas
c    components of the bubble, using values from van liew, pg. 336
c
c      ph2o     = 6.2 kPa = 46.52 mmHg
c      po2      = 5.3 kPa = 39.76 mmHg
c      pco2     = 5.9 kPa = 44.26 mmHg
c      pco2lung = 5.3 kPa = 39.76 mmHg
c       pvap= 46.52d0 + 39.76d0 + 44.26d0
c   read in from input file below   1/18/93
c       ph2o=46.52d0
c       po2=39.76d0
c       pco2=44.26d0
c       pco2lung=39.76d0
```

```
c       pvap=ph2o+po2+pco2
C- vapor pressure of water in bubble @ 37 C in mmHg
        C1125=0.055346
        C11=0.055140
C- pure solvent conc of water @ 37 C in moles/cc
        k=62358.0
C- Boltzman's gas constant in mmHG-cc/mole-K
C- based on PV=kNT and STP conditions
        R=62358.0
C- Universal gas constant in mmHg-cc/mole-K
C- based on published value of 0.08205 L-atm/mole-K
        T=37.0+273.1
C- temperature in Kelvin
        onethird=1.d0/3.d0
C       Pvap=Pvap25
C       c11=c1125

C- Input initial conditions and control variables.
C- ngrow=number of bubble growth cycles,
C- ntrip=number of independent trips thru tissue, nstep=number of steps per
C- trip, Surten=surface tension of bubble (dynes/cm), PCbub=partition
C- coefficient of bubble (1/ostwald coefficient), D=diffusion coefficient
C- (cm sq/sec), Voldep0=initial volume of the depleted region (cu cc),
C- Voltiss = volume of cubical tissue region
C- P0l=initial tissue pressure (mmHg), Pstep=decompression step (mmHg)
        Rewind(1)
        Read(1,900)ngrow,ntrip,nstep,surten,Pcbub,D,gridsize,Voldep0,
     +             Voltiss,P0l,Pstep,bubfact
900     Format(I12/I12/I12/g15.7/g15.7/g15.7/g15.7/g15.7/
     +         g15.7/g15.7/g15.7/g15.7)
        read(1,1900)iprofile,nmicro,washoutt,pctn2
1900    format(i12/i12/g15.7/g15.7)
        read(1,1901)switch,volref,fracsd,ph2o,po2,pco2,pco2lung
1901    format(g15.7)
        close(1)
        pvap=ph2o+po2+pco2

C- Input the array of diffusion distances as a function of probability
C- given in terms of the standard deviation
        Open(unit=1,file='distprob.dat',type='old')
        Read(1,899)(distance(i),i=1,10000)
899     Format(f17.2)
        Close(1)
```

```
      vol=voltiss+voldep0
C- Calculate dimensions of cubical region
      call resize(vol,cubeside)
C- adjust units to mmHG-cm knowing 760 mmHg=1.013X10^6 dyn/cm^2
      Surten=Surten/1332.8
C- Calculate Henry's coefficient from PCbub
      KH=760./((((R*T)/((1./c1l)*(1./PCbub)*760.))+1)**-1.)
C- Calculate stepsize
c     modify stepping array
      call regrid(ia,aa,gridsize,stepsize)
C- Calculate bubble growth cycle time and tissue washout time
      Dtime=stepsize*stepsize/(6.*D)
      cyclet=nstep*Dtime
       poofprob=1.d0-dexp(-cyclet/washoutt)
       if(washoutt.gt.86400.d0)poofprob=0.d0
C- Calculate the maximum distance a particle can travel by diffusion
       onesd=dsqrt(2.d0*d*cyclet)
       diffdist=5.d0*onesd
c
c  figure out the width of the diffusion shells. switch = 0 is for
c  a dimensionless run, with reference volume volref. switch <> 0 is
c  an absolute run, with parameters unscaled.
c
       if(switch.eq.0.d0)then
          width=onesd*fracsd*(vol/volref)**onethird
          if(width.gt.onesd)width=onesd
       else
          width=onesd*fracsd
       end if
C- Write out the starting conditions
      Open(unit=2,file='for002.dat',type='new')
      Write(2,901) Ngrow,Ntrip,Nstep,Surten*1332.8,PCbub,D,
     +  gridsize,stepsize,P0l,Pstep,Voldep0,voltiss
      Close(2)
901    Format(3I9/5G15.4/4G15.4)

C- Begin a bubble growth cycle
      time=0.
      Pl=P0l
       p1n2=(p0l-ph2o-pco2lung)*pctn2
       pfn2=(p0l-pstep-ph2o-pco2lung)*pctn2
      Voldep=Voldep0
      i1=182361+2*secnds(0.0)
       lstrt=1
```

```
      if(iprofile.ne.0)then
c     for a nonuniform initial concentration profile, put code in here
        open(unit=7,file='bubmod.dmp',type='old',form='unformatted')
        rewind(7)
        read(7)lstrt,i1,nzonebeg,time,volbub,calcpc,rbub,rdep,rbub0
        read(7)re,rc,currmole
        read(7)(fraczone(i),molezone(i),conczone(i),i=1,nzonebeg)
        read(7)(rzonebeg(i),rzsq(i),vzone(i),i=1,nzonebeg)
        close(7)
        lstrt=lstrt+1
        cs=c1l*p1n2/kh
       N2=Cs*Vol
       N2dep=Cs*Voldep
       orgmoles=N2
       If (Voldep0.gt.vol) orgmoles=n2dep
       Pl=Pl-Pstep
        csequil=c1l*pfn2/kh
        rzonebeg(0)=0.d0
        rzoneend(0)=0.d0
        bubmoles=molezone(1)
        open(unit=3,file='for003.dat',type='new')
        write(3,3000)0,time,rbub,volbub,calcpc,
     +              (fraczone(i),rzonebeg(i),conczone(i),i=1,nzonebeg)
        close(3)
        depmoles=currmole-bubmoles
        bubfrac=bubmoles/currmole
        depfrac=depmoles/currmole
      end if
     Do 10 L=lstrt,ngrow

C- Calculate number of moles of inert gas (N2) at start of the cycle
C- in the entire tissue and in the depleted region alone (N2dep).
      If((L.eq.1).and.(iprofile.eq.0)) then
5432    continue
        cs=c1l*p1n2/kh
       N2=Cs*Vol
       N2dep=Cs*Voldep
       orgmoles=N2
       If (Voldep0.gt.vol) orgmoles=n2dep
        currmole=orgmoles
       Pl=Pl-Pstep
        csequil=c1l*pfn2/kh
C- Calculate bubble dimensions
C- First, calculate the critical and equilibrium radii
```

```
        a0=2.*surten*Voldep/Pl
        Cs=C1l*Pl/KH
        a1=Voldep*(1.-Pvap/Pl)-N2dep/Cs
        a3=8.*pi*surten/(3.*Cs*k*T)
        a4=4.*pi*Pl*(1.-Pvap/Pl)/(3.*Cs*k*T)
        c0=a0/a4
        c1=a1/a4
        c3=a3/a4
        Call roots(c0,c1,c3,rc,re)
        if (rc.lt.0.) goto 999
C- Assign radius to test bubble
         rbub=re*bubfact+rc*(1.d0-bubfact)
         rbub=dmax1(rbub,(1.5d0*stepsize))
        rbub0=rbub
        VolBub=4.*pi*rbub*rbub*rbub/3.
        rdep=(3.*(Voldep+Volbub)/(4*pi))**(1./3.)
c    check here to make sure the bubble fits inside the cube. if not, spread
c    the excess volume onto the depleted region, and make it larger
         if((rdep.gt.xmax).and.(voltiss.gt.zed))then
            voldep0=vol
            voldep=voldep0
            voltiss=zed
            goto 5432
         end if
         rzonebeg(0)=0.d0
         rzoneend(0)=0.d0
         call genzone(rbub,rdep,width,nmicro,rzonebeg,rzsq,vzone,nzonebeg)
        Bubmoles=Volbub*(Pl-Pvap+2.*surten/rbub)/(R*T)
        Depmoles=N2Dep-Bubmoles
        Tismoles=N2-N2dep
        Bubfrac=Bubmoles/orgmoles
        Depfrac=Depmoles/orgmoles
        Tisfrac=Tismoles/orgmoles
         if(iprofile.eq.0)then
            fraczone(1)=bubfrac
            molezone(1)=bubmoles
            conczone(1)=bubmoles/volbub
            do 1100 izone=2,nzonebeg
               fraczone(izone)=depfrac*vzone(izone)/voldep
               molezone(izone)=fraczone(izone)*orgmoles
               conczone(izone)=depmoles/voldep
1100        continue
         end if
      End if
```

```
        call genzone(rbub,rdep,width,nmicro,rzoneend,rzsq,vzone,nzoneend)

C- Adjust the dimensions of the module
      Volcube=vol+volbub
       call resize(volcube,cubeside)

C- Begin the random walk thru the module
C- Calculate the probability of staying in the bubble
      shellsize=rbub-stepsize
       stepsiz2=stepsize*stepsize
       shelsiz3=shellsize*shellsize*shellsize
      rbub2=rbub*rbub
      rbub3=rbub2*rbub
       rdep2=rdep*rdep
      Pf=1.d0/Pcbub*(rbub3-shelsiz3)/rbub3
      If (rbub.lt.stepsize) pf=1.d0/PCbub
       x1dpf=-1.d0/pf
      Startbub=0.
      Startdep=0.
      Starttis=0.
      EndInbub=0.
      EndIndep=0.
      EndIntis=0.
       do 1005 izone=1,nzoneend
          strtzone(izone)=0.d0
          endzone(izone)=0.d0
1005   continue
       cummsum(0)=0.d0
       do 1004 izone=1,nzonebeg
          cummsum(izone)=cummsum(izone-1)+fraczone(izone)
1004   continue
       tisfrac=1.d0-cummsum(nzonebeg)
       if(tisfrac.lt.0.d0)tisfrac=0.d0
       tismoles=tisfrac*currmole
C- Calculate the number of particles to be place
      Do 20 II=1,Ntrip
C- Place a particle randomly in the module with probability weighted
C- by mole fraction.
      prob=ran(i1)
c
c   new code to place particles in the various zones
       if(prob.ge.cummsum(nzonebeg))then
          call putinbox(xx(1),xx(2),xx(3),zed,zed,zed,cubeside,partdist)
       else
```

```
          do 1001 izone=1,nzonebeg
            if(prob.lt.cummsum(izone))then
              call zone(xx(1),xx(2),xx(3),zed,zed,zed,rzonebeg(izone-1),
     +                  rzonebeg(izone),partdist)
              goto 1002
            end if
1001      continue
1002      continue
        end if

C- If the particle is placed so far from the bubble that it cannot
C- reach it in nstep steps and the module only consists of the
C- bubble and a depleted region, the particle will stay in the depleted
C- region.
        x=xx(1)
        y=xx(2)
        z=xx(3)
        If (partdist.gt.(diffdist+rbub)) then
          Call Bigstep(x,y,z,D,cyclet)
C- Check to see if the particle is outside the spherical depleted region.
C- If it is reflect it back.
         if(voltiss.le.zed)then
         else
           call inbox(x,y,z)
         end if
          Istep=nstep

C- The particle can reach the bubble by diffusion
        Else If (partdist.gt.rbub) then
          Istep=int((partdist-rbub)/stepsize)
          Steptime=Istep*Dtime
          Call Bigstep(x,y,z,D,steptime)
           if(voltiss.le.zed)then
           else
             call inbox(x,y,z)
           end if
C- The particle is in the bubble
         Else
           Istep=0
       End if

C- Carry out random walk for NSTEP steps of "stepsize"
        Do 30 JJ=Istep+1,nstep
```

```
        Inbub=0
C- Check to see if particle is in the bubble by comparing the
C- particle distance from origin with bubbble radius
          if(distfun2(x,y,z,zed,zed,zed).le.rbub2)
     +      inbub=1

C- If the particle is in the bubble place in the transition shell
C- based on the probability factor.
        If (Inbub.eq.1) then
          if (ran(i1).lt.pf)then
              if(rbub.le.stepsize)then
                 call putinorb(x,y,z,zed,zed,zed,rbub)
              else
                 call zone(x,y,z,zed,zed,zed,rbub-stepsize,rbub,dummyr)
              end if
              call step(x,y,z,gridsize)
          end if
        else
C- Take a step
          call step(x,y,z,gridsize)
c
c    assume now that if particle has just left the bubble, it cannot
c    reach the boundaries of the system during a single time cycle.
c    we can thus eliminate the rest of this do loop.
c
        End if
30      Continue

        If (distfun2(x,y,z,zed,zed,zed).gt.rdep2)
     +      Call Reflect(x,y,z,xx(1),xx(2),xx(3),zed,zed,zed,rdep)

C- Reassign coordinates
      xx(1)=x
      xx(2)=y
      xx(3)=z


C- Keep track of the region in which the particle ends
       dist=distfun2(x,y,z,zed,zed,zed)
       do 1010 izone=1,nzoneend
         if(dist.lt.rzsq(izone))then
              endzone(izone)=endzone(izone)+1.d0
           goto 1011
         end if
```

```
1010   continue
1011   continue
       If (dist.le.rbub2) then
         EndInbub=EndInbub+1.
       else If(dist.le.rdep2) then
         EndIndep=EndIndep+1.
       else
         EndIntis=EndIntis+1.
       end if

20     Continue
       if((voltiss.le.zed).and.(endintis.gt.zed))then
        open(unit=2,file='for002.dat',type='old',access='append')
        write(2,8888)endintis,l
8888    format('  accounting error...',f10.2,' particles in tissue'/
     +          ' on the ',i8,'th cycle')
         stop
       end if

C- Calculate the gas content in each region
       If (L.eq.1) then
         open(unit=2,file='for002.dat',type='old',access='append')
         write(2,904)time,rbub,volbub,voldep,
     +  (vol-voldep)
          write(2,904)cyclet,poofprob,washoutt
          Write (2,905) rc,re,rbub0
         close(2)
       end if
c
c   this is where washout takes place. in older versions (commented out)
c   washout was done on a zone by zone basis. this version (2/23/93) does
c   washout over the entire depletion region at the same time. this is the
c   same mathematically as doing them separately.
c
       dfnt=dfloat(ntrip)
       dfnt0=dfnt
       actual=endindep
       if(poofprob.gt.0.d0)then
          const=dfnt*csequil/currmole
          expected=const*voldep
          actual=endindep-(endindep-expected)*poofprob
          dfnt=actual+endinbub
1025   continue
1015     continue
```

```
c
          if(voltiss.gt.zed)dfnt=dfnt+endintis
          currmole=currmole*dfnt/dfnt0
        end if
c
c  end of washout code
c
c    bookkeeping. note if statement to get proper amount in bubble
c
        do 1020 izone=1,nzoneend
          fraczone(izone)=endzone(izone)*actual/endindep/dfnt
          if(izone.eq.1)fraczone(izone)=endzone(izone)/dfnt
          molezone(izone)=fraczone(izone)*currmole
          conczone(izone)=molezone(izone)/vzone(izone)
1020    continue
        calcpc=fraczone(1)*vzone(2)/(fraczone(2)*vzone(1))
       Bubfrac=EndInBub/dfnt
        depfrac=actual/dfnt
       Tisfrac=Endintis/dfnt
        bubmoles=currmole*bubfrac
        depmoles=currmole*depfrac
        tismoles=currmole*tisfrac

C- If Bubfrac is very small it will soon vanish so trap it
       If (bubfrac.lt.0.0000000001) goto 999

C- Calculate the new volume of each region
C- the volume of the depleted region plus the volume of
C- the supersaturated tissue region is a constant, so bubble
C- growth only adds to max and min dimensions, but not volumes
       A3=(4.*pi*(Pl-Pvap))/3.
       A2=(8.*pi*surten)/3.
        a0=-molezone(1)*R*T
       C0=A0/A3
       C2=A2/A3
       Call Roots2(c0,c2,rbub)
        if(rbub.lt.stepsize)goto 999
       VolBub=(4*pi*(rbub)**3.)/3.
       tisVNew=Tismoles/(C1l*P0l/KH)
       Voldep=Vol-tisvnew
       If (Voldep0.ge.vol) Voldep=voldep0-tisvnew
       rdep=(3.*(Voldep+Volbub)/(4*pi))**(1./3.)
        call expand(rbub,rdep,rzoneend,rzonebeg,nzoneend)
        nzonebeg=nzoneend
```

```
      Volcube=vol+volbub
       call resize(volcube,cubeside)

C- Output information for 1 bubble growth cycle
      time=time+cyclet
      open(unit=2,file='for002.dat',type='old',access='append')
      Write(2,904) time,rbub,calcpc,currmole
      close(2)
904    Format(5G15.7)
3000   format('  cycle # ',i5,4g12.4/50(3g15.6/))
       open(unit=7,file='bubmod.dmp',type='old',form='unformatted')
       rewind(7)
       write(7)L,i1,nzonebeg,time,volbub,calcpc,rbub,rdep,rbub0
       write(7)re,rc,currmole
       write(7)(fraczone(i),molezone(i),conczone(i),i=1,nzonebeg)
       write(7)(rzonebeg(i),rzsq(i),vzone(i),i=1,nzonebeg)
       close(unit=7)
10     Continue
999    Continue
905    Format(3G15.4)


      Stop
      End

c

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   generate sin and cos matrices
c
      subroutine gensine
      implicit real*8 (a-h,o-z)
      common/sincos/fact,scale,sinmat(0:9999),cosmat(0:9999)
      do 10 i=0,9999
        cosmat(i)=dcos(dfloat(i)/fact)
        sinmat(i)=dsin(dfloat(i)/fact)
10    continue
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine expand(rbub,rdep,rz1,rz2,nz)
      implicit real*8 (a-h,o-z)
```

```
      dimension rz1(0:500),rz2(0:500)
      common /com2/pi,i1
      common /com5/twopi,onethird
      data nzmax,eps/50,1.d-05/
      coeff=4.d0*pi*onethird
      rz2(1)=rbub
      do 10 i=2,nz
        rz2(i)=(rz1(i)**3-rz1(i-1)**3+rz2(i-1)**3)**onethird
10    continue
      do 20 i=nz+1,nzmax
        rz2(i)=0.d0
20    continue
      if(dabs(rz2(nz)-rdep).gt.eps)then
        write(6,100)nz,rz2(nz),rdep
100     format('  expansion error:  rz2(',i2,') = ',f10.3,
     +          ' rdep = ',f10.3)
        stop
      end if
      return
      end
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine genzone(rbub,rdep,width,nm,rzone,rzsq,vzone,nzone)
      implicit real*8 (a-h,o-z)
      dimension rzone(0:500),rzsq(0:500),vzone(500)
      common/com2/pi,i1
      common /com5/twopi,onethird
      nzone=1
      rzone(0)=0.d0
      rzone(1)=rbub
      coeff=4.d0*pi*onethird
      vzone(1)=coeff*rzone(1)**3
      vtot=vzone(1)
      do 10 i=2,500
        nzone=nzone+1
        rzone(i)=rzone(i-1)+width
        if(rzone(i).gt.rdep)rzone(i)=rdep
        v=coeff*rzone(i)**3
        vzone(i)=v-vtot
        if(rzone(i).ge.rdep)goto 20
        vtot=v
10    continue
```

```
20      continue
        do 40 i=0,nzone
          rzsq(i)=rzone(i)*rzone(i)
40      continue
        return
        end
c
C*********************************************************************
        Subroutine Roots(c0,c1,c3,rc,re)
        implicit real*8 (a-h,o-z)
C- This subroutine calculates the roots of the Tikuisis polynomial.
C- First, find the minimum because 1 root is below it and the other
C- root is above it.
100     format(' # ',i5)
        If (c1.ge.0.) then
          rc=-999.
        else
          x1=-c0/c1
10         x=x1
          g=c1+x*x*(3.*c3+4.*x)
          dgdx=x*(6.*c3+12.*x)
          x1=x-g/dgdx
          if ((abs(g).gt.0.0000001).or.(abs(x1-x).gt.0.0000001))
     +     goto 10
C- Now apply Newton's method to find the roots
          f=rcfunc(x1,c0,c1,c3)
          if (f.gt.0.) then
            rc=-998
          else
            do 1 j=1,2
              if (j.eq.1) then
                mult=-1
              else
                mult=1
              end if
              x2=x1*1.+0.1*mult
20             x=x2
              f0=rcfunc(x,c0,c1,c3)
              f1=c1+x*x*(3.*c3+4.*x)
              dx1=-f0/f1
              x2=dx1+x
              if (abs(1.-(x2/x)).gt.0.0001) goto 20
              if (mult.lt.0) rc=x2
              if (mult.gt.0) re=x2
```

```
1          continue
          end if
        end if
        return
        end


        Real*8 Function rcfunc(x,c0,c1,c3)
        implicit real*8 (a-h,o-z)
        z=c0+x*(c1+x*x*(c3+x))
        rcfunc=z
        return
        end

C*******************************************************************
        Subroutine Roots2(c0,c2,rbub)
        implicit real*8 (a-h,o-z)
C- This subroutine calculates the roots of the polynomial describing
C- bubble radius.
C- Apply Newton's method to find the roots
              x2=rbub
20             x=x2
              f0=rbubfunc(x,c0,c2)
              f1=x*(2.*c2+3.*x)
              dx1=-f0/f1
              x2=dx1+x
              if (abs(1.-(x2/x)).gt.0.0001) goto 20
              rbub=x2
        return
        end

        Real*8 Function rbubfunc(x,c0,c2)
        implicit real*8 (a-h,o-z)
        z=c0+x*x*(c2+x)
        rbubfunc=z
        return
        end
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       subroutine to reflect a particle from the inside surface of a sphere
c
c       (x,y,z)          current position of the particle
c       (xo,yo,zo)        previous position of particle
```

```
c       (xc,yc,zc)      center of the bubble/depleted region
c       r               radius of the depleted region
c
        subroutine reflect(x,y,z,xo,yo,zo,xc,yc,zc,r)
        implicit real*8 (a-h,o-z)
        iperm=0
        xosave=xo
        yosave=yo
        zosave=zo
10      continue
        delx=x-xo
        dely=y-yo
        delz=z-zo
        if((delx.eq.0.d0).and.(dely.eq.0.d0).and.(delz.eq.0.d0))return
        if(delz.eq.0.d0)then
          call permute(x,y,z,xo,yo,zo,xc,yc,zc)
          iperm=iperm+1
          goto 10
        end if
        cx=delx/delz
        cy=dely/delz
        dx=xo-xc-cx*zo
        dy=yo-yc-cy*zo
        aq=cx*cx+cy*cy+1.d0
        bq=2.d0*(cx*dx+cy*dy-zc)
        cq=dx*dx+dy*dy+zc*zc-r*r
        root=dsqrt(bq*bq-4.d0*aq*cq)
        zplus=(-bq+root)/(2.d0*aq)
        zminus=(-bq-root)/(2.d0*aq)
        xplus=cx*(zplus-zo)+xo
        yplus=cy*(zplus-zo)+yo
        xminus=cx*(zminus-zo)+xo
        yminus=cy*(zminus-zo)+yo
        dplus=distfun2(x,y,z,xplus,yplus,zplus)
        dminus=distfun2(x,y,z,xminus,yminus,zminus)
        if(dplus.lt.dminus)then
          xp=xplus
          yp=yplus
          zp=zplus
        else
          xp=xminus
          yp=yminus
          zp=zminus
        end if
```

```
      cpx=xp-xc
      cpy=yp-yc
      cpz=zp-zc
      ptx=x-xp
      pty=y-yp
      ptz=z-zp
c     get dot product between these two vectors
      dot=cpx*ptx+cpy*pty+cpz*ptz
      dt2or2=2.d0*dot/(r*r)
      x=x-dt2or2*cpx
      y=y-dt2or2*cpy
      z=z-dt2or2*cpz
      if(distfunc(x,y,z,xc,yc,zc).gt.r)then
        xo=xp
        yo=yp
        zo=zp
        goto 10
      end if
      if(iperm.ne.0)then
        do 20 i=jmod(iperm+1,3),jmod(iperm+3,3)
          call permute(x,y,z,xo,yo,zo,xc,yc,zc)
20      continue
      end if
      xo=xosave
      yo=yosave
      zo=zosave
      return
      end
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine permute(a1,b1,c1,a2,b2,c2,a3,b3,c3)
      implicit real*8 (a-h,o-z)
      save1=a1
      save2=a2
      save3=a3
      a1=b1
      b1=c1
      c1=save1
      a2=b2
      b2=c2
      c2=save2
      a3=b3
```

```
      b3=c3
      c3=save3
      return
      end
c
c
c*************************************************************************
c
c     subroutine to take a random step to a solid angle on a sphere
c     of radius a
c
      subroutine stepout(x,y,z,x0,y0,z0,bigr2,a)
      implicit real*8 (a-h,o-z)
      common /com2/pi,i1
      common/sincos/fact,scale,sinmat(0:9999),cosmat(0:9999)
c
c     get the distance from the center of the sphere
c
      rx=x-x0
      ry=y-y0
      rz=z-z0
      d2=rx*rx+ry*ry+rz*rz
      d=dsqrt(d2)
      costhetb=(bigr2-a*a-d2)/(2.d0*a*d)
      ranmax=(1.d0-costhetb)/2.d0
      costheta=(1.d0-2.d0*ranmax*ran(i1))
      sintheta=dsqrt(1.d0-costheta*costheta)
      delr=a*costheta
      delperp=a*sintheta
      urx=rx/d
      ury=ry/d
      urz=rz/d
      rperp=dsqrt(urx*urx+ury*ury)
      if(rperp.gt.0.d0)then
        u1x=-ury/rperp
        u1y=urx/rperp
      else
        u1x=1.d0
        u1y=0.d0
      end if
      u2x=u1y*urz
      u2y=-u1x*urz
      u2z=u1x*ury-urx*u1y
      index=scale*ran(i1)
```

```
      anorm=delr/d
      x=x+delperp*(cosmat(index)*u1x+sinmat(index)*u2x)+anorm*rx
      y=y+delperp*(cosmat(index)*u1y+sinmat(index)*u2y)+anorm*ry
      z=z+delperp*(sinmat(index)*u2z)+anorm*rz
      return
      end
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     subroutine to move distance a from a point in the boundary region
c     away from the center of a sphere.
c
      subroutine outstep(x,y,z,x0,y0,z0,a)
      implicit real*8 (a-h,o-z)
      dx=x-x0
      dy=y-y0
      dz=z-z0
      anorm=a/dsqrt(dx*dx+dy*dy+dz*dz)
      x=x+anorm*dx
      y=y+anorm*dy
      z=z+anorm*dz
      return
      end
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     subroutine to take a single large diffusion step in three
c     dimensions. uses subroutine stepit, the random direction
c     stepping routine, and also calculates a normally distributed
c     distance that the particle steps.
c
      subroutine bigstep(x,y,z,diffcoef,t)
      implicit real*8 (a-h,o-z)
      common /com2/pi,i1
      common/distarr/distance(10000)
c
c     the distance stepped as a function of the probability, given in terms
c     of the standard deviation, is located infile distprob.dat
c
      index=aint(1.d0+10000.d0*ran(i1))
      sd=dsqrt(2.d0*diffcoef*t)
      d=sd*distance(index)
```

```
      call stepit(x,y,z,d)
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     subroutine to take a random step of length r0 in 3 dimensions
c
      subroutine stepit(x,y,z,r0)
      implicit real*8 (a-h,o-z)
      common /com2/pi,i1
      common/sincos/fact,scale,sinmat(0:9999),cosmat(0:9999)
      common /com5/twopi,onethird
      xr=1.d0-2.d0*ran(i1)
      sintheta=dsqrt(1.d0-xr*xr)
      index=scale*ran(i1)
      x=x+r0*sintheta*cosmat(index)
      y=y+r0*sintheta*sinmat(index)
      z=z+r0*xr
      return
      end
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     subroutine to position a particle randomly within a spherical
c     shell with outer radius router,and inner radius rinner
c
c
      subroutine zone(x,y,z,x0,y0,z0,rinner,router,rho)
      implicit real*8 (a-h,o-z)
      common /com2/pi,i1
      common/sincos/fact,scale,sinmat(0:9999),cosmat(0:9999)
      common /com5/twopi,onethird
c
c     use spherical coordinates rho,theta,phi
c
      x=x0
      y=y0
      z=z0
      if(rinner.le.0.d0)then
        rho=0.d0
        return
      end if
```

```
      rin3=rinner*rinner*rinner
      rout3=router*router*router
      range=rout3-rin3
      rho=(rin3+range*ran(i1))**onethird
      call stepit(x,y,z,rho)
      return
      end
c
C*************************************************************************
c
c     subroutine to position a particle randomly within a spherical
c     shell with outerradius bigr,and inner radius bigr-a
c
c
      subroutine shell(x,y,z,x0,y0,z0,r0,a)
      implicit real*8 (a-h,o-z)
      common /com2/pi,i1
      common/sincos/fact,scale,sinmat(0:9999),cosmat(0:9999)
      common /com5/twopi,onethird
c
c     use spherical coordinates rho,theta,phi
c
      if(r0.lt.a)then
        call putinorb(x,y,z,x0,y0,z0,r0)
        return
      end if
      bma=r0-a
      bma3=bma*bma*bma
      b3=r0*r0*r0
      range=b3-bma3
      rho=(bma3+range*ran(i1))**onethird
      phi=twopi*ran(i1)
      theta=dacos(1.d0-2.d0*ran(i1))
      x=x0+rho*dsin(theta)*dcos(phi)
      y=y0+rho*dsin(theta)*dsin(phi)
      z=z0+rho*dcos(theta)
      return
      end
c
c*************************************************************************
C
      subroutine step(x,y,z,grid)
      implicit real*8 (a-h,o-z)
      common/com2/pi,i1
```

```
      common/com3/IA(3,24),aa(3,24)

      JA=1+int(24.*ran(i1))
       x=x+aa(1,ja)
       y=y+aa(2,ja)
       z=z+aa(3,ja)
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     subroutine to test for presence of a particle in a sphere of radius r
c     centered at (x0,y0,z0), given position (x,y,z)
c
      subroutine insphere(x,y,z,x0,y0,z0,r,in)
      implicit real*8 (a-h,o-z)
      logical in
      r2=r*r
      if(distfun2(x,y,z,x0,y0,z0).le.r2)then
        in=.true.
      else
        in=.false.
      end if
      return
      end

c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     subroutine to place particle randomly within a sphere of radius r
c     centered at (x0,y0,z0)
c
      subroutine putinorb(x,y,z;x0,y0,z0,r0)
      implicit real*8 (a-h,o-z)
      common /com2/pi,i1
      common/sincos/fact,scale,sinmat(0:9999),cosmat(0:9999)
      common /com5/twopi,onethird
c
c     use spherical coordinates rho,theta,phi
c
      rho=r0*(ran(i1)**onethird)
      x=x0
      y=y0
      z=z0
```

```
      call stepit(x,y,z,rho)
      return
      end
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     subroutine to see if particle has left the box and if so, to
c     reflect it from the boundary.
c
      subroutine inbox(x,y,z)
      implicit real*8 (a-h,o-z)
      common /com1/xmin,xmax,ymin,ymax,zmin,zmax,rbub,rdep
c
c     reflection conditions
c
      if(x.lt.xmin)x=xmin+(xmin-x)
      if(x.gt.xmax)x=xmax+(xmax-x)
      if(y.lt.ymin)y=ymin+(ymin-y)
      if(y.gt.ymax)y=ymax+(ymax-y)
      if(z.lt.zmin)z=zmin+(zmin-z)
      if(z.gt.zmax)z=zmax+(zmax-z)
c
c     translation conditions
c
c     if(xx(1).lt.xmin)xx(1)=xmax-(xmin-xx(1))
c     if(xx(1).gt.xmax)xx(1)=xmin-(xmax-xx(1))
c     if(xx(2).lt.ymin)xx(2)=ymax-(ymin-xx(2))
c     if(xx(2).gt.ymax)xx(2)=ymin-(ymin-xx(2))
c     if(xx(3).lt.zmin)xx(3)=zmax-(zmin-xx(3))
c     if(xx(3).gt.zmax)xx(3)=zmin-(zmax-xx(3))
c
      return
      end
c
c
C**********************************************************************
c     function to find the distance between two points
c
      function distfunc(x,y,z,x0,y0,z0)
      implicit real*8 (a-h,o-z)
      dx=x-x0
      dy=y-y0
      dz=z-z0
```

```
      distfunc=dsqrt(dx*dx+dy*dy+dz*dz)
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   function to calculate the square of the distance between points
c
      function distfun2(x,y,z,x0,y0,z0)
      implicit real*8 (a-h,o-z)
      dx=x-x0
      dy=y-y0
      dz=z-z0
      distfun2=dx*dx+dy*dy+dz*dz
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine regrid(ia,aa,gridsize,stepsize)
      implicit real*8 (a-h,o-z)
      dimension ia(3,24),aa(3,24)
      do 1901 iii=1,3
        do 1901 iiii=1,24
          aa(iii,iiii)=dfloat(ia(iii,iiii))*gridsize
1901  continue
     stepsize=gridsize*dsqrt(3.d0)
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine resize(vol,s)
      implicit real*8 (a-h,o-z)
      common /com1/xmin,xmax,ymin,ymax,zmin,zmax,rbub,rdep
      common /com5/twopi,onethird

      s=(vol**onethird)
     xmax=s/2.
     xmin=-xmax
     ymax=xmax
     ymin=xmin
     zmax=xmax
     zmin=xmin
```

```
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine putinbox(x,y,z,x0,y0,z0,s,dist)
      implicit real*8 (a-h,o-z)
      common /com1/xmin,xmax,ymin,ymax,zmin,zmax,rbub,rdep
      common /com2/pi,i1
10    continue
      x=xmin+s*ran(i1)
      y=ymin+s*ran(i1)
      z=zmin+s*ran(i1)
      dist=distfunc(x,y,z,x0,y0,z0)
      if(dist.lt.rdep)goto 10
      return
      end
c
```

Section D2. Sample input file for source code of Section D1.

```
   8000,              number of cycles
 400000,              number of trips per cycle
     50,              number of steps per trip
  50.00,              surface tension
  70.00,              partition coefficient
.0000050,             diffusion coeff
.00025,                     grid size
.125E-03,            depletion region volume
0.0,                 extra-depletion volume
 1520.00,            initial inert gas partial pressure
  760.00,            pressure step
.1706,                      initial bubble radius (as fraction of equil. radius)
 0,              iprofile      # of initial concentration profile
 0,              nmicro         # of sub-zones to be used
   150.,          washoutt        tissue time constant in seconds
 0.7900,          pctn2          percent nitrogen during washout
  1.d0,           switch   =0 => dimensionless run
 .100e-02,        volref          reference vol for dimless run
 1.d0,            fracsd         fraction of onesd for shell widths
 46.52d0,         ph2o           partial pressure water vapor
 39.76d0,          po2           partial pressure oxygen in bubble
 44.26d0,         pco2           partial pressure CO2 in bubble
 39.76d0,     pco2lung           part. press. CO2 in lungs
 1.d0,              bndfact           boundary factor for PDE3
   1/19/93   test run for .ch9
```

Section D3. FORTRAN source code for generation of cummulative distribution function for diffusion in three dimensions.

```
c       program to generate the file 'distprob.dat', which contains the
c       number of standard deviations stepped by a diffusing particle as
c       a function of probability. it uses integration routine from the
c       numerical recipes directory on piggy.
        implicit real*8 (a-h,o-z)
        npts=10000
        nstep=1000000
        dfnp=dfloat(npts)
        sum=0.d0
        sumnext=0.d0
        xlow=0.d0
        delx=0.0001
        open(unit=1,file='distprob.dat',type='new')
        do 10 i=1,npts-1
          prob=dfloat(i)/dfnp
          xhigh=xlow
          do 20 j=1,nstep
            xhigh=xhigh+delx
            call qromb(xlow,xhigh,sumnext)
            if((sum+sumnext).lt.prob)goto 20
            sum=sum+sumnext
            write(1,100)(xlow+xhigh)/2.d0,i
100         format(f15.6,i10)
            xlow=xhigh
            goto 10
20        continue
10      continue
        close(1)
        stop
        end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        function func(t)
        implicit real*8 (a-h,o-z)
        data init/0/
        if(init.eq.0)then
          pi=4.d0*datan(1.d0)
          const=dsqrt(2.d0/pi)
          init=1
```

```
      end if
      func=const*t*t*exp(-t*t/2.d0)
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      SUBROUTINE QROMB(A,B,SS)
        implicit real*8 (a-h,o-z)
      PARAMETER(EPS=1.E-6,JMAX=20,JMAXP=JMAX+1,K=5,KM=4)
      DIMENSION S(JMAXP),H(JMAXP)
      H(1)=1.
      DO 11 J=1,JMAX
        CALL TRAPZD(A,B,S(J),J)
        IF (J.GE.K) THEN
          L=J-KM
          CALL POLINT(H(L),S(L),K,0.,SS,DSS)
          IF (ABS(DSS).LT.EPS*ABS(SS)) RETURN
        ENDIF
        S(J+1)=S(J)
        H(J+1)=0.25*H(J)
11    CONTINUE
      PAUSE 'Too many steps.'
      END
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      SUBROUTINE TRAPZD(A,B,S,N)
        implicit real*8 (a-h,o-z)
      IF (N.EQ.1) THEN
        S=0.5*(B-A)*(FUNC(A)+FUNC(B))
        IT=1
      ELSE
        TNM=IT
        DEL=(B-A)/TNM
        X=A+0.5*DEL
        SUM=0.
        DO 11 J=1,IT
          SUM=SUM+FUNC(X)
          X=X+DEL
11      CONTINUE
        S=0.5*(S+(B-A)*SUM/TNM)
        IT=2*IT
      ENDIF
```

```
      RETURN
      END
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      SUBROUTINE POLINT(XA,YA,N,X,Y,DY)
        implicit real*8 (a-h,o-z)
      PARAMETER (NMAX=10)
      DIMENSION XA(N),YA(N),C(NMAX),D(NMAX)
      NS=1
      DIF=ABS(X-XA(1))
      DO 11 I=1,N
        DIFT=ABS(X-XA(I))
        IF (DIFT.LT.DIF) THEN
          NS=I
          DIF=DIFT
        ENDIF
        C(I)=YA(I)
        D(I)=YA(I)
11    CONTINUE
      Y=YA(NS)
      NS=NS-1
      DO 13 M=1,N-1
        DO 12 I=1,N-M
          HO=XA(I)-X
          HP=XA(I+M)-X
          W=C(I+1)-D(I)
          DEN=HO-HP
          IF(DEN.EQ.0.)PAUSE
          DEN=W/DEN
          D(I)=HP*DEN
          C(I)=HO*DEN
12      CONTINUE
        IF (2*NS.LT.N-M)THEN
          DY=C(NS+1)
        ELSE
          DY=D(NS)
          NS=NS-1
        ENDIF
        Y=Y+DY
13    CONTINUE
      RETURN
      END
```

Section D4. FORTRAN source code for the generation of non-uniform initial gas concentration distributions.

```
c   this program is a modification of the beginning of the bubble simulation
c   program. it is designed to generate a particular non-uniform concentration
c   gradient for use by the main program. the main program assumes an initial
c   uniform concentration profile wherein the gas used to create the bubble
c   is drawn from throughout the tissue volume. this program generates a
c   profile from the opposite extreme; the gas for the bubble is taken from
c   the innermost shells, leading to a step function profile. shells near the
c   bubble have initial concentration cs = concentration of gas in the bubble.
c   shells far from the bubble have concentration cs0, in equilibrium with
c   the initial pressure of inert gas. there is one transition shell with a
c   concentration between these two values. the concentration profile is then
c   written out to file bubmod.dmp, where it can be used by the main program
c   by setting iprofile = 1
c

        implicit real*8  (a-h,o-z)
         common /com1/xmin,xmax,ymin,ymax,zmin,zmax,rbub,rdep
        common/com2/pi,i1
        common/com3/IA(3,24),aa(3,24)
         common /com5/twopi,onethird
        real*8 N2,N2dep,k,kH,molezone(500)
        Dimension xx(3),rzonebeg(0:500),rzoneend(0:500),widezone(500)
        dimension strtzone(500),endzone(500),fraczone(500),conczone(500)
        dimension cummsum(0:500),vzone(500),rzsq(0:500)
        data IA/1,1,1,1,1,-1,1,-1,1,1,-1,-1,-1,1,1,-1,1,-1,-1,-1,
     +  1,-1,-1,-1,
     +  1,1,1,-1,1,1,1,-1,1,-1,-1,1,1,1,-1,-1,1,-1,1,-1,-1,-1,-1,-1,
     +  1,1,1,1,1,-1,-1,1,1,-1,1,-1,1,-1,1,1,1,-1,-1,-1,-1,1,-1,-1,-1/
C- Define constants
        zed=0.d0
        Pi=4.d0*datan(1.d0)
         twopi=2.d0*pi
         fourpio3=4.d0*pi/3.d0
         onethird=1.d0/3.d0
c        Pvap25=23.756
c        Pvap=47.067
c    change 12/16/92 to include oxygen and co2 as additional gas
c    components of the bubble, using values from van liew, pg. 336
c
c      ph2o     = 6.2 kPa = 46.52 mmHg
c      po2      = 5.3 kPa = 39.76 mmHg
c      pco2     = 5.9 kPa = 44.26 mmHg
```

```
c      pco2lung = 5.3 kPa = 39.76 mmHg
c       pvap= 46.52d0 + 39.76d0 + 44.26d0
        ph2o=46.52d0
        po2=39.76d0
        pco2=44.26d0
        pco2lung=39.76d0
        pvap=ph2o+po2+pco2
C- vapor pressure of water in bubble @ 37 C in mmHg
        C1125=0.055346
        C11=0.055140
C- pure solvent conc of water @ 37 C in moles/cc
        k=62358.0
C- Boltzman's gas constant in mmHG-cc/mole-K
C- based on PV=kNT and STP conditions
        R=62358.0
C- Universal gas constant in mmHg-cc/mole-K
C- based on published value of 0.08205 L-atm/mole-K
        T=37.0+273.1
C- temperature in Kelvin
C        Pvap=Pvap25
C        c11=c1125

C- Input initial conditions and control variables.
C- ngrow=number of bubble growth cycles,
C- ntrip=number of independent trips thru tissue, nstep=number of steps per
C- trip, Surten=surface tension of bubble (dynes/cm), PCbub=partition
C- coefficient of bubble (1/ostwald coefficient), D=diffusion coefficient
C- (cm sq/sec), Voldep0=initial volume of the depleted region (cu cc),
C- Voltiss = volume of cubical tissue region
C- P0l=initial tissue pressure (mmHg), Pstep=decompression step (mmHg)
        Rewind(1)
        Read(1,900)ngrow,ntrip,nstep,surten,Pcbub,D,gridsize,Voldep0,
     +  Voltiss,P0l,Pstep,bubfact
900     Format(I9/I9/I9/F9.2/F9.2/F9.7/F9.7/F9.7/F9.7/F9.2/F9.2/F9.7)
        read(1,1900)iprofile,nmicro,washoutt,pctn2
1900    format(i12/i12/g15.7/g15.7)
        close(1)

        vol=voltiss+voldep0
C- Calculate dimensions of cubical region
C- adjust units to mmHG-cm knowing 760 mmHg=1.013X10^6 dyn/cm^2
        Surten=Surten/1332.8
C- Calculate Henry's coefficient from PCbub
        KH=760./((((R*T)/((1./c11)*(1./PCbub)*760.))+1)**-1.)
```

```
C- Calculate stepsize
c       modify stepping array
        call regrid(ia,aa,gridsize,stepsize)
C- Calculate bubble growth cycle time and tissue washout time
        Dtime=stepsize*stepsize/(6.*D)
        cyclet=nstep*Dtime
C- Calculate the maximum distance a particle can travel by diffusion
        onesd=dsqrt(2.d0*d*cyclet)
C- Begin a bubble growth cycle
        time=0.
        Pl=P0l
        p1n2=(p0l-ph2o-pco2lung)*pctn2
        pfn2=(p0l-pstep-ph2o-pco2lung)*pctn2
        Voldep=Voldep0
        lstrt=1

C- Calculate number of moles of inert gas (N2) at start of the cycle
C- in the entire tissue and in the depleted region alone (N2dep).
5432    continue
c           Cs=C1l*Pl/KH
            cs=c1l*p1n2/kh
            cs0=cs
            N2=Cs*Vol
            N2dep=Cs*Voldep
            orgmoles=N2
            If (Voldep0.gt.vol) orgmoles=n2dep
            currmole=orgmoles
            Pl=Pl-Pstep
            csequil=c1l*pfn2/kh
C- Calculate bubble dimensions
C- First, calculate the critical and equilibrium radii
            a0=2.*surten*Voldep/Pl
c           Cs=C1l*Pl/KH
            cs=c1l*(pl-pvap)/kh
            a1=Voldep*(1.-Pvap/Pl)-N2dep/Cs
            a3=8.*pi*surten/(3.*Cs*k*T)
            a4=4.*pi*Pl*(1.-Pvap/Pl)/(3.*Cs*k*T)
            c0=a0/a4
            c1=a1/a4
            c3=a3/a4
            Call roots(c0,c1,c3,rc,re)
            if (rc.lt.0.) goto 999
C- Assign radius to test bubble
            rbub=re*bubfact
```

```
          rbub=dmax1(rbub,(1.5d0*stepsize))
          rbub0=rbub
          VolBub=4.*pi*rbub*rbub*rbub/3.
c          rdep=(3.*(Voldep+Volbub)/(4*pi))**(1./3.)
          rdep=((voldep+volbub)/fourpio3)**onethird
c    check here to make sure the bubble fits inside the cube. if not, spread
c    the excess volume onto the depleted region, and make it larger
          if((rdep.gt.xmax).and.(voltiss.gt.zed))then
             voldep0=vol
             voldep=voldep0
             voltiss=zed
             goto 5432
          end if
          rzonebeg(0)=0.d0
          rzoneend(0)=0.d0
          call genzone(rbub,rdep,onesd,nmicro,rzonebeg,rzsq,vzone,nzonebeg)
          Bubmoles=Volbub*(Pl-Pvap+2.*surten/rbub)/(R*T)
          Depmoles=N2Dep-Bubmoles
          Tismoles=N2-N2dep
          Bubfrac=Bubmoles/orgmoles
          Depfrac=Depmoles/orgmoles
          Tisfrac=Tismoles/orgmoles
          fraczone(1)=bubfrac
          molezone(1)=bubmoles
          conczone(1)=bubmoles/volbub
          delcs=cs0-cs
          vinner=bubmoles/delcs
          vrunold=0.d0
          do 1100 izone=2,nzonebeg
             vrun=vrunold+vzone(izone)
             if(vrun.le.vinner)then
                conczone(izone)=cs
             else if((vrunold.le.vinner).and.(vinner.lt.vrun))then
                conczone(izone)=((vinner-vrunold)*cs+(vrun-vinner)*cs0)/
     +                          vzone(izone)
             else
                conczone(izone)=cs0
             end if
             vrunold=vrun
             molezone(izone)=conczone(izone)*vzone(izone)
             fraczone(izone)=molezone(izone)/orgmoles
1100         continue
3000    format('  cycle # ',i5,4g12.4/50(3g15.6/))
        l=0
```

```
      rbub0=rbub
      time=0.d0
      i1=237645+2*secnds(0.0)
      calcpc=fraczone(1)*vzone(2)/(fraczone(2)*vzone(1))
      open(unit=7,file='bubmod.dmp',type='new',form='unformatted')
      rewind(7)
      write(7)L,i1,nzonebeg,time,volbub,calcpc,rbub,rdep,rbub0
      write(7)re,rc,currmole
      write(7)(fraczone(i),molezone(i),conczone(i),i=1,nzonebeg)
      write(7)(rzonebeg(i),rzsq(i),vzone(i),i=1,nzonebeg)
      close(unit=7)
999   Continue

      Stop
      End

c

c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine genzone(rbub,rdep,width,nm,rzone,rzsq,vzone,nzone)
      implicit real*8 (a-h,o-z)
      dimension rzone(0:500),rzsq(0:500),vzone(500)
      common/com2/pi,i1
      common /com5/twopi,onethird
      nzone=1
      rzone(0)=0.d0
      rzone(1)=rbub
      coeff=4.d0*pi*onethird
      vzone(1)=coeff*rzone(1)**3
      vtot=vzone(1)
      do 10 i=2,50
        nzone=nzone+1
        rzone(i)=rzone(i-1)+width
        if(rzone(i).gt.rdep)rzone(i)=rdep
        v=coeff*rzone(i)**3
        vzone(i)=v-vtot
        if(rzone(i).ge.rdep)goto 20
        vtot=v
10    continue
20    continue
      do 40 i=0,nzone
        rzsq(i)=rzone(i)*rzone(i)
```

```
40      continue
        return
        end
c
C*********************************************************************
        Subroutine Roots(c0,c1,c3,rc,re)
        implicit real*8 (a-h,o-z)
C- This subroutine calculates the roots of the Tikuisis polynomial.
C- First, find the minimum because 1 root is below it and the other
C- root is above it.
100     format(' # ',i5)
        If (c1.ge.0.) then
          rc=-999.
        else
          x1=-c0/c1
10         x=x1
          g=c1+x*x*(3.*c3+4.*x)
          dgdx=x*(6.*c3+12.*x)
          x1=x-g/dgdx
          if ((abs(g).gt.0.0000001).or.(abs(x1-x).gt.0.0000001))
     +     goto 10
C- Now apply Newton's method to find the roots
          f=rcfunc(x1,c0,c1,c3)
          if (f.gt.0.) then
            rc=-998
          else
            do 1 j=1,2
              if (j.eq.1) then
                mult=-1
              else
                mult=1
              end if
              x2=x1*1.+0.1*mult
20             x=x2
              f0=rcfunc(x,c0,c1,c3)
              f1=c1+x*x*(3.*c3+4.*x)
              dx1=-f0/f1
              x2=dx1+x
              if (abs(1.-(x2/x)).gt.0.0001) goto 20
              if (mult.lt.0) rc=x2
              if (mult.gt.0) re=x2
1           continue
          end if
        end if
```

```
      return
      end


      Real*8 Function rcfunc(x,c0,c1,c3)
      implicit real*8 (a-h,o-z)
      z=c0+x*(c1+x*x*(c3+x))
      rcfunc=z
      return
      end

c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine regrid(ia,aa,gridsize,stepsize)
      implicit real*8 (a-h,o-z)
      dimension ia(3,24),aa(3,24)
      do 1901 iii=1,3
         do 1901 iiii=1,24
            aa(iii,iiii)=dfloat(ia(iii,iiii))*gridsize
1901  continue
     stepsize=gridsize*dsqrt(3.d0)
      return
      end
c
```

Section D5. Fortran source code for Monte Carlo simulation which utilizes radially symmetry of problem to speed computation.

```
C-  This program simulates bubble growth beginning with a bubble of
C-  critical radius as defined by Tikuisis.  The bubble is represented as
C-  a region of high solubility surrounded by a "depleted region" from
C-  which the inert gas is extracted.  These two regions are placed in
C-  a third region which represents the surrounding tissue.
c   this program also uses 1 dimension, the radial one, to keep
c   track of the particle. should speed things up considerably.

      implicit real*8  (a-h,o-z)
      common/com2/pi,i1
       common/comroot/probroot,firstime
       common/distarr/distance(10000),sd(500)
       common /com5/twopi,onethird,vcoeff
      real*8 N2,N2dep,k,kH,molezone(500)
      Dimension xx(3),sumcumm(0:500)
      dimension endzone(500),fraczone(500),conczone(500)
      dimension cummsum(0:500),rzrngb(500),rzrnge(500)
       dimension rzb(0:500),rz2b(0:500),rz3b(0:500),vzb(500)
       dimension rze(0:500),rz2e(0:500),rz3e(0:500),vze(500)
C- Define constants
       zed=0.d0
      Pi=4.d0*datan(1.d0)
       onethird=1.d0/3.d0
       vcoeff=4.d0*pi*onethird
       twopi=2.d0*pi
c    change 12/16/92 to include oxygen and co2 as additional gas
c    components of the bubble, using values from van liew, pg. 336
c
c      ph2o     = 6.2 kPa = 46.52 mmHg
c      po2      = 5.3 kPa = 39.76 mmHg
c      pco2     = 5.9 kPa = 44.26 mmHg
c      pco2lung = 5.3 kPa = 39.76 mmHg
c       pvap= 46.52d0 + 39.76d0 + 44.26d0
c   read in from input file below   1/18/93
c       ph2o=46.52d0
c       po2=39.76d0
c       pco2=44.26d0
c       pco2lung=39.76d0
c       pvap=ph2o+po2+pco2
C- vapor pressure of water in bubble @ 37 C in mmHg
      C1125=0.055346
```

```
      C11=0.055140
C- pure solvent conc of water @ 37 C in moles/cc
      k=62358.0
C- Boltzman's gas constant in mmHG-cc/mole-K
C- based on PV=kNT and STP conditions
      gasR=62358.0
C- Universal gas constant in mmHg-cc/mole-K
C- based on published value of 0.08205 L-atm/mole-K
      T=37.0+273.1
C- temperature in Kelvin

C- Input initial conditions and control variables.
C- ngrow=number of bubble growth cycles,
C- ntrip=number of independent trips thru tissue, nstep=number of steps per
C- trip, Surten=surface tension of bubble (dynes/cm), PCbub=partition
C- coefficient of bubble (1/ostwald coefficient), D=diffusion coefficient
C- (cm sq/sec), Voldep0=initial volume of the depleted region (cu cc),
C- Voltiss = volume of cubical tissue region
C- P0l=initial tissue pressure (mmHg), Pstep=decompression step (mmHg)
      Rewind(1)
      Read(1,900)ngrow,ntrip,nstep,surten,Pcbub,D,gridsize,Voldep0,
     +           Voltiss,P0l,Pstep,bubfact
900    Format(I12/I12/I12/g15.7/g15.7/g15.7/g15.7/g15.7/
     +       g15.7/g15.7/g15.7/g15.7)
       read(1,1900)iprofile,nmicro,washoutt,pctn2
1900   format(i12/i12/g15.7/g15.7)
       read(1,1901)switch,volref,fracsd,ph2o,po2,pco2,pco2lung
1901   format(g15.7)
      close(1)
       pvap=ph2o+po2+pco2

C- Input the array of diffusion distances as a function of probability
C- given in terms of the standard deviation
      Open(unit=1,file='distprob.dat',type='old')
      Read(1,899)(distance(i),i=1,10000)
899    Format(f17.2)
      Close(1)

       vol=voltiss+voldep0
C- adjust units to mmHG-cm knowing 760 mmHg=1.013X10^6 dyn/cm^2
      Surten=Surten/1332.8
C- Calculate Henry's coefficient from PCbub
      KH=760./((((gasR*T)/((1./c11)*(1./PCbub)*760.))+1)**-1.)
C- Calculate stepsize
```

```
      stepsize=gridsize*dsqrt(3.d0)
C- Calculate bubble growth cycle time and tissue washout time
      Dtime=stepsize*stepsize/(6.*D)
      cyclet=nstep*Dtime
       do 333 kk=1,nstep
         xtime=kk*dtime
         sd(kk)=dsqrt(2.d0*d*xtime)
333    continue
       poofprob=1.d0-dexp(-cyclet/washoutt)
       if(washoutt.gt.86400.d0)poofprob=0.d0
C- Calculate the maximum distance a particle can travel by diffusion
       onesd=dsqrt(2.d0*d*cyclet)
       diffdist=5.d0*onesd
c
c   figure out the width of the diffusion shells. switch = 0 is for
c   a dimensionless run, with reference volume volref. switch <> 0 is
c   an absolute run, with parameters unscaled.
c
       if(switch.eq.0.d0)then
         width=onesd*fracsd*(vol/volref)**onethird
         if(width.gt.onesd)width=onesd
       else
         width=onesd*fracsd
       end if
       if(iprofile.eq.0)then
C- Write out the starting conditions
        Open(unit=2,file='for002.dat',type='new')
        Write(2,901) Ngrow,Ntrip,Nstep,Surten*1332.8,PCbub,D,
     +    gridsize,stepsize,P0l,Pstep,Voldep0,voltiss
        Close(2)
901      Format(3I9/5G15.4/4G15.4)
       end if

C- Begin a bubble growth cycle
      time=0.
      Pl=P0l
       p1n2=(p0l-ph2o-pco2lung)*pctn2
       pfn2=(p0l-pstep-ph2o-pco2lung)*pctn2
      Voldep=Voldep0
      i1=182361+2*secnds(0.0)
       lstrt=1
       if(iprofile.ne.0)then
c    for a nonuniform initial concentration profile, put code in here
         open(unit=7,file='bubmod.dmp',type='old',form='unformatted')
```

```
        rewind(7)
        read(7)lstrt,i1,nzonebeg,time,volbub,calcpc,rbub,rdep,rbub0
        read(7)re,rc,currmole
        read(7)(fraczone(i),molezone(i),conczone(i),i=1,nzonebeg)
        read(7)(rzb(i),rz2b(i),rz3b(i),vzb(i),rzrngb(i),i=1,nzonebeg)
        read(7)(rze(i),rz2e(i),rz3e(i),vze(i),rzrnge(i),i=1,nzoneend)
        close(7)
        lstrt=lstrt+1
        cs=c1l*p1n2/kh
       N2=Cs*Vol
       N2dep=Cs*Voldep
       orgmoles=N2
       If (Voldep0.gt.vol) orgmoles=n2dep
       Pl=Pl-Pstep
        csequil=c1l*pfn2/kh
        rzb(0)=0.d0
        rze(0)=0.d0
        bubmoles=molezone(1)
        open(unit=3,file='for003.dat',type='new')
        write(3,3000)0,time,rbub,volbub,calcpc,
     +              (fraczone(i),rzb(i),conczone(i),i=1,nzonebeg)
        close(3)
        depmoles=currmole-bubmoles
        bubfrac=bubmoles/currmole
        depfrac=depmoles/currmole
      end if
     Do 10 L=lstrt,ngrow

C- Calculate number of moles of inert gas (N2) at start of the cycle
C- in the entire tissue and in the depleted region alone (N2dep).
      If((L.eq.1).and.(iprofile.eq.0)) then
5432    continue
        cs=c1l*p1n2/kh
       N2=Cs*Vol
       N2dep=Cs*Voldep
       orgmoles=N2
       If (Voldep0.gt.vol) orgmoles=n2dep
        currmole=orgmoles
       Pl=Pl-Pstep
        csequil=c1l*pfn2/kh
C- Calculate bubble dimensions
C- First, calculate the critical and equilibrium radii
       a0=2.*surten*Voldep/Pl
       Cs=C1l*Pl/KH
```

```
          a1=Voldep*(1.-Pvap/Pl)-N2dep/Cs
          a3=8.*pi*surten/(3.*Cs*k*T)
          a4=4.*pi*Pl*(1.-Pvap/Pl)/(3.*Cs*k*T)
          c0=a0/a4
          c1=a1/a4
          c3=a3/a4
          Call roots(c0,c1,c3,rc,re)
          if (rc.lt.0.) goto 999
C- Assign radius to test bubble
           rbub=re*bubfact+rc*(1.d0-bubfact)
           rbub=dmax1(rbub,(1.5d0*stepsize))
          rbub0=rbub
          VolBub=4.*pi*rbub*rbub*rbub/3.
          rdep=(3.*(Voldep+Volbub)/(4*pi))**(1./3.)
c    check here to make sure the bubble fits inside the cube. if not, spread
c    the excess volume onto the depleted region, and make it larger
           rzb(0)=0.d0
           rze(0)=0.d0
           call fastgenz(rbub,rdep,width,nzonebeg,rzb,rz2b,rz3b,vzb,rzrngb)
          Bubmoles=Volbub*(Pl-Pvap+2.*surten/rbub)/(gasR*T)
          Depmoles=N2Dep-Bubmoles
          Tismoles=N2-N2dep
          Bubfrac=Bubmoles/orgmoles
          Depfrac=Depmoles/orgmoles
          Tisfrac=Tismoles/orgmoles
           if(iprofile.eq.0)then
             fraczone(1)=bubfrac
             molezone(1)=bubmoles
             conczone(1)=bubmoles/volbub
             do 1100 izone=2,nzonebeg
                fraczone(izone)=depfrac*vzb(izone)/voldep
                molezone(izone)=fraczone(izone)*orgmoles
                conczone(izone)=depmoles/voldep
1100         continue
           end if
       End if
          call fastgenz(rbub,rdep,width,nzoneend,rze,rz2e,rz3e,vze,rzrnge)

C- Begin the random walk thru the module
C- Calculate the probability of staying in the bubble
        rbpw=rbub+width
        stepsiz2=stepsize*stepsize
       rbub2=rbub*rbub
       rbub3=rbub2*rbub
```

```
      rdep2=rdep*rdep
      pf=stepsize*(12.d0*rbub2-stepsiz2)/(16.d0*(rbub3)*pcbub)
      ddprbub=diffdist+rbub
      firstime=zed
     EndInbub=0.
     EndIndep=0.
     EndIntis=0.
      do 1005 izone=1,nzoneend
        endzone(izone)=0.d0
1005  continue
      sumcumm(0)=zed
      cummsum(0)=0.d0
      do 1004 izone=1,nzonebeg
        cummsum(izone)=cummsum(izone-1)+fraczone(izone)
1004  continue
      if(cummsum(nzonebeg).gt.zed)then
        if(voltis.le.zed)then
          do 1904 izone=0,nzonebeg
            cummsum(izone)=cummsum(izone)/cummsum(nzonebeg)
            sumcumm(nzonebeg-izone)=1.d0-cummsum(izone)
1904      continue
        end if
      end if
C- Calculate the number of particles to be place
      Do 20 II=1,Ntrip
C- Place a particle randomly in the module with probability weighted
C- by mole fraction.
      prob=ran(i1)
c
c   new code to place particles in the various zones
      do 1001 izone=1,nzonebeg
        if(prob.lt.sumcumm(izone))then
          call place(r,rz3b(nzonebeg-izone),rzrngb(nzonebeg-izone+1))
          r2=r*r
          rp=r
          rp2=r2
          goto 1002
        end if
1001  continue
1002  continue

C- If the particle is placed so far from the bubble that it cannot
C- reach it in nstep steps and the module only consists of the
C- bubble and a depleted region, the particle will stay in the depleted
```

```
C- region.

        If (r.gt.ddprbub) then
          Istep=nstep
          call getsize(a,a2,istep)
          call faststep(a,a2,r,r2,rp,rp2,cosphi)
          goto 31

       Else If (r.gt.rbpw) then
C- The particle can reach the bubble by diffusion
         Istep=int((r-rbub)/stepsize)
          call getsize(a,a2,istep)
          call faststep(a,a2,r,r2,rp,rp2,cosphi)

        Else
C- The particle is in the bubble
          Istep=0
      End if

C- Carry out random walk for NSTEP steps of "stepsize"

          a=stepsize
          a2=stepsiz2
       Do 30 JJ=Istep+1,nstep
          Inbub=0
C- Check to see if particle is in the bubble by comparing the
C- particle distance from origin with bubbble radius

          if(rp.lt.rbub)inbub=1

C- If the particle is in the bubble, leave it
C- based on the probability factor.
          If (Inbub.eq.1) then
            if (ran(i1).lt.pf)then
                call fastout(r,a,rbub)
                rp=r
                rp2=rp*rp
            end if
          else
C- Take a step
            r=rp
            r2=rp2
            call faststep(a,a2,r,r2,rp,rp2,cosphi)
            if(rp.gt.rdep)then
```

```
                call fastrefl(a,r,r2,rp,rp2,rdep,rdep2,cosphi)
                r=rp
                r2=rp2
              end if
            End if
30      Continue
31      Continue
        if(rp.gt.rdep)then
          call fastrefl(a,r,r2,rp,rp2,rdep,rdep2,cosphi)
          r=rp
          r2=rp2
          if(r.gt.rdep)then
c   if the particle is still outside the depletion region after reflection,
c   assume that further reflections will leave it in the outermost zone.
            call place(r,rz3b(nzonebeg-1),rzrngb(nzonebeg))
            r2=r*r
          end if
        else
          r=rp
          r2=rp2
        end if

C- Keep track of the region in which the particle ends
        do 1010 izone=1,nzoneend
          if(r.lt.rze(izone))then
                endzone(izone)=endzone(izone)+1.d0
            goto 1011
          end if
1010    continue
1011    continue
        If (r.le.rbub) then
          EndInbub=EndInbub+1.
        else If(r.le.rdep) then
          EndIndep=EndIndep+1.
        else
          EndIntis=EndIntis+1.
        end if

20      Continue
        if((voltiss.le.zed).and.(endintis.gt.zed))then
         open(unit=2,file='for002.dat',type='old',access='append')
          write(2,8888)endintis,1
8888      format('  accounting error...',f10.2,' particles in tissue'/
     +           ' on the ',i8,'th cycle')
```

```
          stop
        end if

C- Calculate the gas content in each region
       If (L.eq.1) then
         open(unit=2,file='for002.dat',type='old',access='append')
         write(2,904)time,rbub,volbub,voldep,
     +  (vol-voldep)
          write(2,904)cyclet,poofprob,washoutt
          Write (2,905) rc,re,rbub0
         close(2)
       end if
        dfnt=dfloat(ntrip)
        dfnt0=dfnt
        actual=endindep
        if(poofprob.gt.0.d0)then
           const=dfnt*csequil/currmole
           expected=const*voldep
           actual=endindep-(endindep-expected)*poofprob
           dfnt=actual+endinbub
1025   continue
1015      continue
          if(voltiss.gt.zed)dfnt=dfnt+endintis
          currmole=currmole*dfnt/dfnt0
        end if
        do 1020 izone=1,nzoneend
          fraczone(izone)=endzone(izone)/dfnt*actual/endindep
          if(izone.eq.1)fraczone(izone)=endzone(izone)/dfnt
          molezone(izone)=fraczone(izone)*currmole
          conczone(izone)=molezone(izone)/vze(izone)
1020   continue
        calcpc=fraczone(1)*vze(2)/(fraczone(2)*vze(1))
       Bubfrac=EndInBub/dfnt
       depfrac=actual/dfnt
       Tisfrac=Endintis/dfnt
        bubmoles=currmole*bubfrac
        depmoles=currmole*depfrac
        tismoles=currmole*tisfrac

C- If Bubfrac is very small it will soon vanish so trap it
       If (bubfrac.lt.0.0000000001) goto 999

C- Calculate the new volume of each region
C- the volume of the depleted region plus the volume of
```

```
C- the supersaturated tissue region is a constant, so bubble
C- growth only adds to max and min dimensions, but not volumes
      A3=(4.*pi*(Pl-Pvap))/3.
      A2=(8.*pi*surten)/3.
       a0=-molezone(1)*gasR*T
      C0=A0/A3
      C2=A2/A3
      Call Roots2(c0,c2,rbub)
       if(rbub.lt.stepsize)goto 999
      VolBub=(4*pi*(rbub)**3.)/3.
      tisVNew=Tismoles/(C11*P01/KH)
      Voldep=Vol-tisvnew
      If (Voldep0.ge.vol) Voldep=voldep0-tisvnew
       rdep=((voldep+volbub)/vcoeff)**onethird
       call fastexpnd(rbub,rdep,nzoneend,rzb,rz2b,rz3b,rzrnge,rzrngb)
       nzonebeg=nzoneend

C- Output information for 1 bubble growth cycle
      time=time+cyclet
      open(unit=2,file='for002.dat',type='old',access='append')
      Write(2,904) time,rbub,calcpc,currmole
      close(2)
904    Format(5G15.7)
3000   format('  cycle # ',i5,4g12.4/50(3g15.6/))
       open(unit=7,file='bubmod.dmp',type='old',form='unformatted')
       rewind(7)
       write(7)L,i1,nzonebeg,time,volbub,calcpc,rbub,rdep,rbub0
       write(7)re,rc,currmole
       write(7)(fraczone(i),molezone(i),conczone(i),i=1,nzonebeg)
       write(7)(rzb(i),rz2b(i),rz3b(i),vzb(i),rzrngb(i),i=1,nzonebeg)
       write(7)(rze(i),rz2e(i),rz3e(i),vze(i),rzrnge(i),i=1,nzoneend)
       close(7)
10     Continue
999    Continue
905    Format(3G15.4)


      Stop
      End

c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
       subroutine fastout(r,a,rb)
```

```
      implicit real*8 (a-h,o-z)
      common /com2/pi,i1
      common/comroot/probroot,firstime
      probroot=ran(i1)
      s=rtsafe(0.d0,1.d0,1.d-10,rb,a)*a
      r=s+rb
      return
      end
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   subroutines to generate c.d.f. for particles stepping out of a bubble.
c   some parameters are supplied to the subroutine via the common block
c   comroot.
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      FUNCTION RTSAFE(X1,X2,XACC,b,a)
        implicit real*8 (a-h,o-z)
      PARAMETER (MAXIT=100)
      CALL FUNCD(X1,FL,DF,b,a)
      CALL FUNCD(X2,FH,DF,b,a)
      IF(FL*FH.GE.0.) PAUSE 'root must be bracketed'
      IF(FL.LT.0.)THEN
        XL=X1
        XH=X2
      ELSE
        XH=X1
        XL=X2
        SWAP=FL
        FL=FH
        FH=SWAP
      ENDIF
      RTSAFE=.5*(X1+X2)
      DXOLD=ABS(X2-X1)
      DX=DXOLD
      CALL FUNCD(RTSAFE,F,DF,b,a)
      DO 11 J=1,MAXIT
        IF(((RTSAFE-XH)*DF-F)*((RTSAFE-XL)*DF-F).GE.0.
     *      .OR. ABS(2.*F).GT.ABS(DXOLD*DF) ) THEN
          DXOLD=DX
          DX=0.5*(XH-XL)
          RTSAFE=XL+DX
          IF(XL.EQ.RTSAFE)RETURN
        ELSE
```

```
        DXOLD=DX
        DX=F/DF
        TEMP=RTSAFE
        RTSAFE=RTSAFE-DX
        IF(TEMP.EQ.RTSAFE)RETURN
      ENDIF
      IF(ABS(DX).LT.XACC) RETURN
      CALL FUNCD(RTSAFE,F,DF,b,a)
      IF(F.LT.0.) THEN
        XL=RTSAFE
        FL=F
      ELSE
        XH=RTSAFE
        FH=F
      ENDIF
11    CONTINUE
    PAUSE 'RTSAFE exceeding maximum iterations'
    RETURN
    END
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine funcd(x,f,d,b,a)
      implicit real*8 (a-h,o-z)
      common/comroot/probroot,firstime
      if(firstime.eq.0.d0)then
        roa=b/a
        roa2=roa*roa
        f1=(12.d0*roa-24.d0*roa2)
        f2=(6.d0-24.d0*roa+12.d0*roa2)
        f3=(12.d0*roa-8.d0)
        f4=3.d0
        d0=f1
        d1=2.d0*f2
        d2=3.d0*f3
        d3=4.d0*f4
        firstime=1.d0
      end if
      f0=(12.d0*roa2-1.d0)*probroot
      f=((((f4*x+f3)*x+f2)*x+f1)*x+f0)
      d=(((d3*x+d2)*x+d1)*x+d0)
      return
      end
c
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine fastexpnd(rb,rd,nz,rz,rz2,rz3,rzre,rzrb)
      implicit real*8 (a-h,o-z)
      dimension rz(0:500),rz2(0:500),rz3(0:500),rzre(500),rzrb(500)
      common/com5/twopi,onethird,coeff
      data nzmax,eps/500,1.d-5/
      rz(0)=0.d0
      rz2(0)=0.d0
      rz3(0)=0.d0
      rz(1)=rb
      rz2(1)=rb*rb
      rz3(1)=rb*rz2(1)
      rzrb(1)=rzre(1)
      do 10 i=2,nz
        rz3(i)=rz3(i-1)+rzre(i)
        rz(i)=rz3(i)**onethird
        rz2(i)=rz(i)*rz(i)
        rzrb(i)=rzre(i)
10    continue
      do 50 i=nz+1,nzmax
        rz(i)=0.d0
        rz2(i)=0.d0
        rz3(i)=0.d0
        rzrb(i)=rzre(i)
50    continue
      if(dabs(rz(nz)-rd).gt.eps)then
        write(3,100)nz,rz(nz),rd
100     format('   expansion error:   rz(',i3,') = ',g15.6,
     +          '   rdep = ',g15.6)
        stop
      end if
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  variables:        a      = stepsize
c                    r1     = initial radial position
c                    r12    = r1*r1
c                    r2     = radial distance > rd  (do not use)
c                    r22    = r2*r2        (do not use)
c                    r3     = final radial position
c                    r32    = r3*r3
```

```
c                  rd     = depletion radius
c                  rd2    = rd*rd
c                  cp     = cos(phi), from faststep
c                  st     = sin(theta)
c                  st2    = st*st
c                  ct     = cos(theta)
c
c   NOTE:  even after reflection from the outer surface, the particle
c          may still be outside the depletion region. have to check for
c          this in the main program.
c
      subroutine fastrefl(a,r1,r12,r3,r32,rd,rd2,cp)
      implicit real*8 (a-h,o-z)
      common/com2/pi,i1
      common/com5/twopi,onethird,coeff
      cp2=cp*cp
      sp2=1.d0-cp2
      if(sp2.lt.0.d0)sp2=0.d0
      d1=r1*cp+dsqrt(rd2-r12*sp2)
      st2=r12*sp2/rd2
      if(st2.ge.1.d0)then
        ct=0.d0
      else
        ct=dsqrt(1.d0-st2)
      end if
      d2=a-d1
      d22=d2*d2
      r32=rd2+d22-2.d0*rd*d2*ct
      r3=dsqrt(r32)
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine faststep(a,a2,r,r2,rp,rp2,cosphi)
      implicit real*8 (a-h,o-z)
      common/com2/pi,i1
      common/com5/twopi,onethird,coeff
      cosphi=1.d0-2.d0*ran(i1)
      rp2=a2+r2-2.d0*a*r*cosphi
      rp=dsqrt(rp2)
      return
      end
c
```

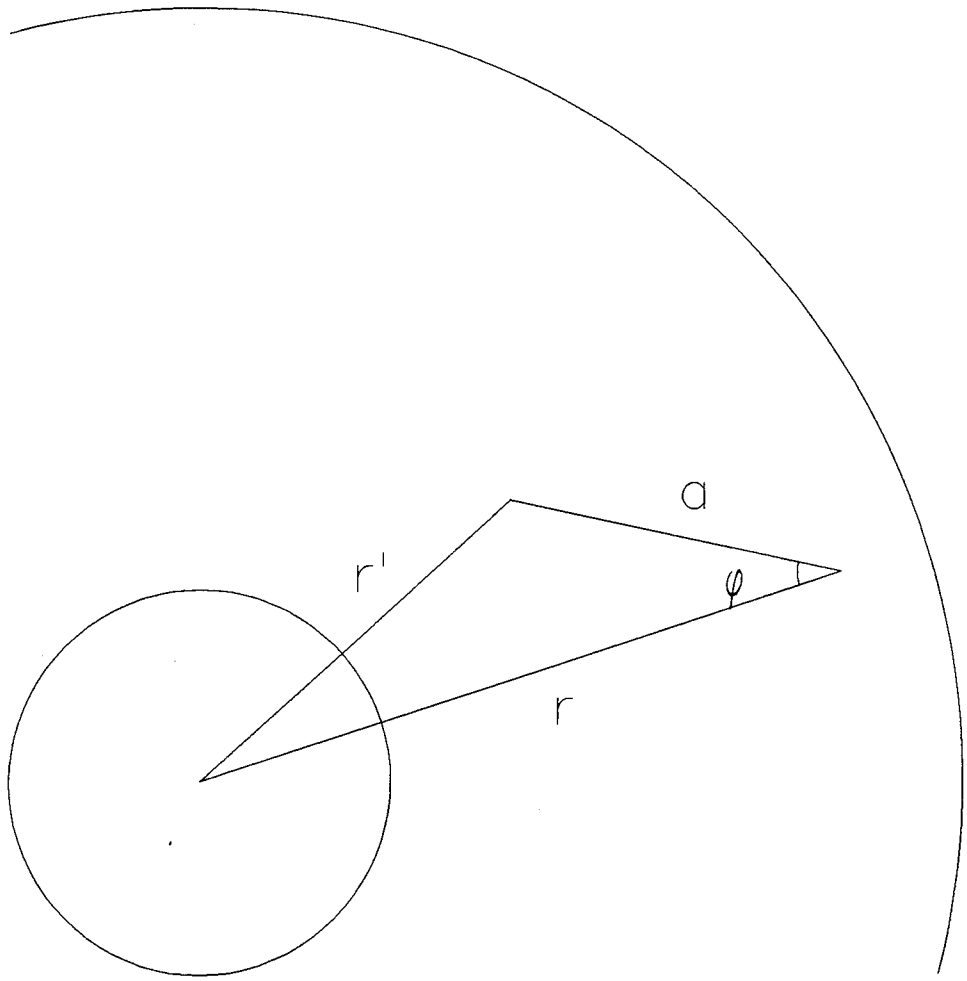

FIGURE 16 - Geometry used in subroutine FASTSTEP for a radially symmetric bubble-liquid module

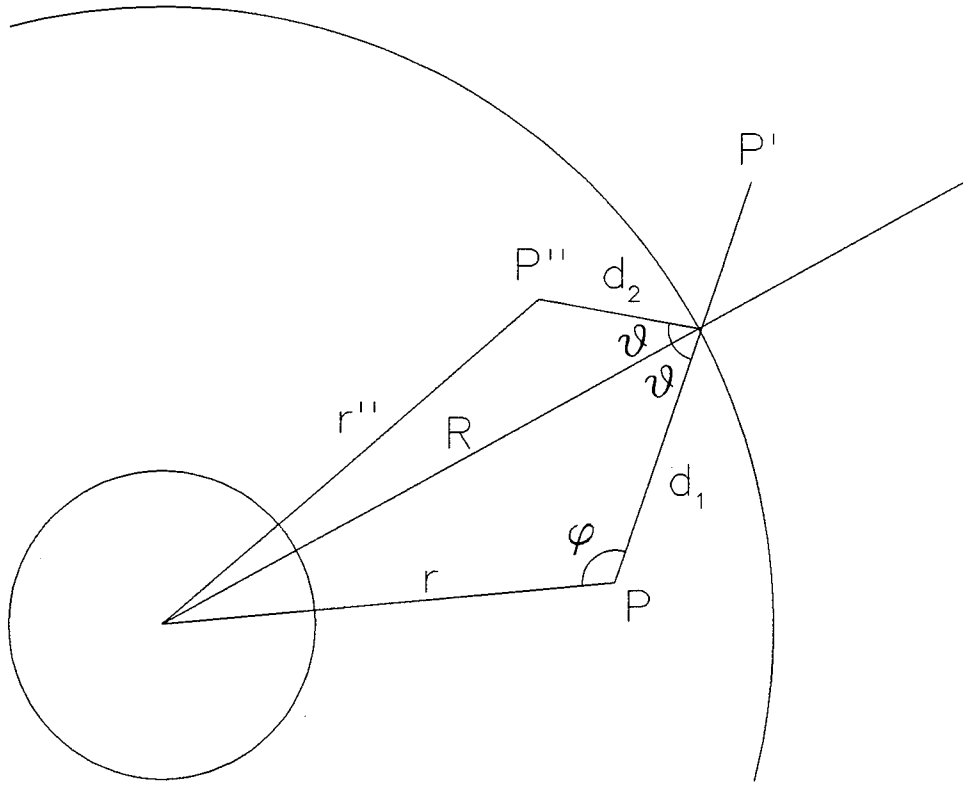

FIGURE 17 - Geometry used in subroutine FASTREFL for a radially symmetric bubble-liquid module.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine place(r,rz3,rzrng)
      implicit real*8 (a-h,o-z)
      common/com2/pi,i1
      common/com5/twopi,onethird,coeff
      r=(rz3+rzrng*ran(i1))**onethird
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine fastgenz(rb,rd,w,nz,rz,rz2,rz3,vz,rzrng)
      implicit real*8 (a-h,o-z)
      dimension rz(0:500),rz2(0:500),rz3(0:500),vz(500),rzrng(500)
      common/com5/twopi,onethird,coeff
      data nzmax/500/
      nz=1
      rz(0)=0.d0
      rz(1)=rb
      do 10 i=2,nzmax
        nz=nz+1
        rz(i)=rz(i-1)+w
        if(rz(i).ge.rd)then
          rz(i)=rd
          goto 20
        end if
10    continue
20    continue
      rz2(0)=0.d0
      rz3(0)=0.d0
      do 40 i=1,nz
        rz2(i)=rz(i)*rz(i)
        rz3(i)=rz2(i)*rz(i)
        rzrng(i)=rz3(i)-rz3(i-1)
        vz(i)=coeff*rzrng(i)
40    continue
      do 50 i=nz+1,nzmax
        rz2(i)=0.d0
        rz3(i)=0.d0
        rzrng(i)=0.d0
        vz(i)=0.d0
50    continue
      return
```

```
      end
c
C*********************************************************************
      Subroutine Roots(c0,c1,c3,rc,re)
      implicit real*8 (a-h,o-z)
C- This subroutine calculates the roots of the Tikuisis polynomial.
C- First, find the minimum because 1 root is below it and the other
C- root is above it.
100   format(' # ',i5)
      If (c1.ge.0.) then
        rc=-999.
      else
        x1=-c0/c1
10      x=x1
        g=c1+x*x*(3.*c3+4.*x)
        dgdx=x*(6.*c3+12.*x)
        x1=x-g/dgdx
        if ((abs(g).gt.0.0000001).or.(abs(x1-x).gt.0.0000001))
     +    goto 10
C- Now apply Newton's method to find the roots
        f=rcfunc(x1,c0,c1,c3)
        if (f.gt.0.) then
          rc=-998
        else
          do 1 j=1,2
            if (j.eq.1) then
              mult=-1
            else
              mult=1
            end if
            x2=x1*1.+0.1*mult
20           x=x2
            f0=rcfunc(x,c0,c1,c3)
            f1=c1+x*x*(3.*c3+4.*x)
            dx1=-f0/f1
            x2=dx1+x
            if (abs(1.-(x2/x)).gt.0.0001) goto 20
            if (mult.lt.0) rc=x2
            if (mult.gt.0) re=x2
1         continue
        end if
      end if
      return
      end
```

```
      Real*8 Function rcfunc(x,c0,c1,c3)
      implicit real*8 (a-h,o-z)
      z=c0+x*(c1+x*x*(c3+x))
      rcfunc=z
      return
      end

C*********************************************************************
      Subroutine Roots2(c0,c2,rbub)
      implicit real*8 (a-h,o-z)
C- This subroutine calculates the roots of the polynomial describing
C- bubble radius.
C- Apply Newton's method to find the roots
            x2=rbub
20           x=x2
            f0=rbubfunc(x,c0,c2)
            f1=x*(2.*c2+3.*x)
            dx1=-f0/f1
            x2=dx1+x
            if (abs(1.-(x2/x)).gt.0.0001) goto 20
            rbub=x2
      return
      end

      Real*8 Function rbubfunc(x,c0,c2)
      implicit real*8 (a-h,o-z)
      z=c0+x*x*(c2+x)
      rbubfunc=z
      return
      end
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       subroutine to take find the size of a single large diffusion
c       step for use in subroutine faststep
c
c       subroutine getsize(a,a2,diffcoef,t)
        subroutine getsize(a,a2,isd)
        implicit real*8 (a-h,o-z)
        common /com2/pi,i1
        common/distarr/distance(10000),sd(500)
c
c       the distance stepped as a function of the probability, given in terms
```

```
c       of the standard deviation, is located infile distprob.dat
c
        index=aint(1.d0+10000.d0*ran(i1))
        a=sd(isd)*distance(index)
        a2=a*a
        return
        end
c
```