Fall 10-31-1996

A controlled release technique using microporous membranes

Stephanie Farrell
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

A CONTROLLED RELEASE TECHNIQUE USING MICROPOROUS MEMBRANES

by
Stephanie Farrell

A novel controlled release device based on aqueous-organic partitioning is described. The device comprises a reservoir, bounded by a microporous or porous membrane in the form of a hollow fiber or flat film. The reservoir liquid phase and the pore liquid phase are immiscible. The agent partitions between the phases at the aqueous-organic interface of the reservoir and the pore mouth, and then diffuses through the membrane pore liquid into a surrounding aqueous solution. The partition coefficient significantly influences the rate of release of the agent by reducing the driving force for diffusion across the fluid-filled membrane pore. The performance of the system is evaluated using model agents benzoic acid, caffeine, nicotine and phenylalanine-glycine. Two aqueous-organic configurations were investigated: an agent in an organic reservoir solution with water-filled pores, and an agent in an aqueous reservoir with organic-filled pores. Specifically, the model systems included benzoic acid in three reservoir solvents (octanol, decanol, and mineral oil) partitioning into water-filled pores, an aqueous reservoir of nicotine partitioning into either mineral

oil- or octanol-filled pores, and caffeine or phenylalanine-glycine partitioning into octanol-filled pores. The peptide phenylalanine-glycine was used to investigate pH-based controlled release from this type of device. Studies using benzoic acid demonstrate the effectiveness of a thin, nonporous coating on the release rate. When a fast-dissolving dispersion of the agent is present in the reservoir, the period of zero order release is extended; when the dispersion dissolves slowly, the release rate is decreased and the period of zero order release is extended. Simultaneous release of two agents (benzoic acid and nicotine, nicotine and caffeine) from a single reservoir and from two separate reservoirs was achieved. Models are presented for many of these systems. Solutions have been developed to describe the observed release, and dimensional analysis was used to identify important parameters which govern the release rate of the agent from the device. Finally, a new technique is presented for achieving controlled release of liposomes from a membrane-type diffusion based controlled release system.

A CONTROLLED RELEASE TECHNIQUE USING MICROPOROUS MEMBRANES

by
Stephanie Farrell

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Chemical Engineering,
Chemistry and Environmental Science

October, 1996

APPROVAL PAGE
A CONTROLLED RELEASE TECHNIQUE USING MICROPOROUS MEMBRANES

Stephanie Farrell

Dr. Kamalesh Sirkar, Dissertation Advisor — Date
Professor of Chemical Engineering and Sponsored Chair, Membrane Separations and Biotechnology, NJIT

Dr. Basil Baltzis, Committee Member — Date
Professor of Chemical Engineering, NJIT

Dr. David Kafkewitz, Committee Member — Date
Professor of Microbiology, Rutgers University

Dr. David Kristol, Committee Member — Date
Professor of Chemistry, NJIT

Dr. Norman Loney, Committee Member — Date
Assistant Professor of Chemical Engineering, NJIT

BIBLIOGRAPHICAL SKETCH

Author: Stephanie Farrell

Degree: Doctor of Philosophy

Undergraduate and Graduate Education:

- Doctor of Philosophy in Chemical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1996
- Master of Science in Chemical Engineering,
  Stevens Institute of Technology, Hoboken, NJ, 1992
- Bachelor of Science in Chemical Engineering,
  University of Pennsylvania, Philadelphia, PA, 1986

Patent

K. K. Sirkar, S. Farrell, and R. Basu, "Novel Controlled Release Device and Method Based on Aqueous-organic Partitioning in Porous Membranes", United States Patent 08/205,996, applied for, February, 1994.

Presentations

S. Farrell and K. K. Sirkar, "Controlled Release Using Aqueous-Organic Partitioning and Dissolution as Rate Limiting Mechanisms," NAMS Annual Meeting, Ottawa, Canada, May, 1996.

S. Farrell and K. K. Sirkar, "Controlled Release Using Aqueous-organic Partitioning: Simultaneous Release of Two Agents and Agents Suspended in the Reservoir," A.I.Ch.E. Annual Meeting, Miami, FL, November, 1995.

S. Farrell and K. K. Sirkar, "A Novel Technique for Controlled Release," Controlled Release Society 22nd International Symposium on Controlled Release of Bioactive Materials, Seattle, WA, August, 1995.

S. Farrell and K. K. Sirkar, "A Novel Controlled Release Technique," A.I.Ch.E. Annual Meeting, San Francisco, CA, November, 1994.

Dedicated to my parents
who have supported my every endeavor.

ACKNOWLEDGEMENT

I am grateful to my advisor, Professor Kamalesh K. Sirkar for the continual challenge he provided. He has an unlimited capacity to generate novel ideas and an ability to lead his students to think creatively.

Pablo Delgado was a tremendous help to me, and despite his hectic schedule he always found time to work in the lab. Soumendu Bhattacharya from Rutgers University generously shared his time to help with the preparation of liposomes. I am indebted to Uttam Shanbhag, who is always willing to put down what he is doing to lend a hand to others. Clint Brockway from HSMRC has generously shared his time and resources to help make research go smoothly. Soemantri Widagdo taught me the most valuable lesson in problem solving: walk before you run.

I am lucky to have been surrounded by a very charismatic and cooperative group of colleagues and friends. Helen Androutsopoulou and Ann Marie Flynn helped make the days at N.J.I.T. something to look forward to. Harry Papadopoulos and his wife Maria Goulimari more than once provided a couch to sleep on when I had missed the last train home. The members of the Membrane Separations and Biotechnology research group have created a very supportive and enjoyable working and learning environment. Judy Kapp has made all our lives easier .

Jim Sheil has been a loyal and devoted friend for half a lifetime. His good sense of humor and unique perspectives give an extra spin to life. My

faithful companion Muti sat by my side on many cold nights while I worked at the computer, and frequently she even helped me type.

My husband Peter Cole has given me unconditional understanding and encouragement. He has patiently and unselfishly accepted so little of my time – I can only hope that it will be a pleasant adjustment for him when he sees more of me.

TABLE OF CONTENTS

Chapter **Page**

1 INTRODUCTION 1

1 .1 Background 1

1 .2 Controlled Release Using Aqueous-Organic Partitioning 3

1 .3 Liposome Release Studies 12

2 MATHEMATICAL MODEL 16

2 .1 Introduction 16

2 .1.1 The Problem 16

2 .1.2 Overview of the Solution Procedure 17

2 .1.3 Simplifying Assumptions 18

2 .2 Agent Dissolved in the Reservoir: Problem Formulation 19

2 .2.1 Dimensional Equations 19

2 .2.2 Dimensionless Equations 21

2 .3 Agent Dissolved in the Reservoir: Solutions 24

2 .3.1 Flat Membrane (Cartesian Coordinates) 24

2 .3.2 Hollow Fiber (Cylindrical Coordinates) 26

2 .3.3 Coated Hollow Fiber 27

2 .4 Dispersed Reservoir Phase: Problem Formulation 31

2 .5 Solution (Dispersed Phase in Reservoir) 36

2 .5.1 Flat Membrane (Cartesian Coordinates) 36

TABLE OF CONTENTS
(Continued)

Chapter **Page**

2 .5.2 Hollow Fiber (Cylindrical Coordinates) 37

3 EXPERIMENTAL PROCEDURES 39

3 .1 Solvents and Solutes 39

3 .2 Membranes 39

3 .3 Liposome Preparation 42

3 .4 Analysis 43

3 .4.1 Benzoic Acid, Caffeine, Nicotine and Phe-Gly 43

3 .4.2 Liposome 47

3 .5 Determination of Distribution Coefficient 50

3 .6 Diffusion Coefficient Measurement 51

3 .6.1 Benzoic Acid, Caffeine, and Nicotine 51

3 .6.2 Liposome 52

3 .7 Measurement of Solids Density in Solvent 53

3 .8 Membrane Preparation 53

3 .9 Introduction of Agent into Reservoir 55

3 .10 In Vitro Release Studies 58

4 RESULTS AND DISCUSSION 61

4 .1 Experimental Results 61

4 .1.1 Distribution Coefficients 61

TABLE OF CONTENTS
(Continued)

Chapter **Page**

4 .1.2 Diffusion Coefficients 62

4 .1.3 Solid Density 68

4 .1.4 Solution in Reservoir: In Vitro Experiments 68

4 .1.5 Suspension in Reservoir: In Vitro Experiments 79

4 .1.6 Coated Membrane Experiments 85

4 .1.7 Simultaneous Release of Two Agents 89

4 .1.8 Pure Liquid Agent in the Reservoir 94

4 .1.9 Peptides: In Vitro Experiments 95

4 .1.10 Liposomes: In Vitro Experiments 97

4 .2 Dimensional Analysis 98

4 .2.1 Uncoated Membrane, Solution in Reservoir 98

4 .2.2 Uncoated Membrane, Suspension in Reservoir 107

4 .2.3 The Coated Membrane 110

5 CONCLUSIONS AND RECOMMENDATIONS 112

5 .1 Conclusions 112

5 .2 Recommendations 114

NOTATION 116

APPENDIX 1 118

APPENDIX 2 164

TABLE OF CONTENTS
(Continued)

Chapter	Page
APPENDIX 3	188
REFERENCES	190

LIST OF TABLES

Table		Page
1	Membrane Dimensions	40
2	Membrane Materials	41
3	Distribution Coefficients for Agents in Aqueous-Organic Systems	61
4	Distribution Coefficients for Phe-Gly Between Water and Octanol at Various pH Values	62
5	Calculated$^{(W.-C.)}$ and Experimentally Measured Binary Diffusion Coefficients of Agents in Organic Solvents	63
6	Diffusion Coefficients of Agents in Water	67
7	Densities of Pure Powder Form Crystalline Agents	68
8	Nomenclature Equivalents for the "Flat Membrane, Solution in Reservoir" Problem	facing 118
9	Nomenclature Equivalents for the "Hollow Fiber: Solution in Reservoir" Problem	facing 130
10	Nomenclature Equivalents for the "Coated Fiber: Solution in Reservoir" Problem	facing 144
11	Nomenclature Equivelents for the "Flat Membrane: Dispersed Phase in Reservoir" Problem	facing 164
12	Nomenclature Equivalents for the "Hollow Fiber: Dispersed Phase in Reservoir" Problem	facing 175

LIST OF FIGURES

Figure		Page
1	The physical situation in a membrane controlled release device using aqueous-organic partitioning	4
2	The effect of partitioning on the driving force for diffusion across the liquid-filled membrane pores	6
3	The reservoir and membrane regions of the controlled release device.	19
4	The coated hollow fiber	28
5	The controlled release device with dispersed solids in the reservoir	32
6	Calibration for benzoic acid on Hypersil ODS C-18 column	44
7	Calibration for caffeine on Spherisorb column.	44
8	Calibration for nicotine on Spherisorb column	45
9	Calibration for benzoic acid on Spherisorb column	45
10	Calibration for Phe-Gly on Nucleosil 100-5 C18 column	46
11	Calibration for toluene on Hypersil ODS C-18 column	47
12	Absorbance spectrum of SUVETs	48
13	NBD fluorescence spectrum	49
14	Calibration for SUVET using absorbance spectrophotometry.	49
15	Calibration for SUVET using fluorescence spectrophotometry.	50
16	The Two-Reservoir Diffusion Cell System	52
17	The single reservoir controlled release cell for flat membrane studies	56
18	The two reservoir controlled release cell used for flat membrane studies	57

LIST OF FIGURES
(Continued)

Figure		Page
19	An in vitro release experiment using hollow fibers	59
20	An in vitro release experiment using a flat membrane	60
21	Diffusion coefficient of nicotine in mineral oil	64
22	Diffusion coefficient of nicotine in octanol	64
23	Diffusion coefficient of caffeine in octanol	65
24	Diffusion of benzoic acid in mineral oil	65
25	Diffusion coefficient of SUVET in water	67
26	The release profile of benzoic acid from a nylon porous hollow fiber with water-filled pores	facing 69
27	The release profile of benzoic acid from a nylon hollow fiber with water-filled pores.	70
28	The release profile of benzoic acid from a flat membrane device using a Celgard® 2400 flat membrane with water-filled pores.	71
29	Controlled release of benzoic acid using a water-swollen Cuprophan 150 PM regenerated cellulose membrane.	72
30	The release profile of nicotine from nylon hollow fibers with mineral oil-filled pores.	74
31	The release profile from a flat membrane device using a Celgard® 2400 flat membrane with mineral oil-filled pores.	75
32	The release profile for nicotine from a flat membrane device using a Celgard® 2400 flat membrane with mineral oil-filled pores	76
33	Extended release of benzoic acid using a suspension.	80
34	Extended release of benzoic acid using a suspension.	81

LIST OF FIGURES
(Continued)

Figure		Page
35	The effect of rate limiting dissolution on the release profile.	82
36	Extended release of caffeine using a suspension.	84
37	Extended release of caffeine using a suspension.	84
38	The effect of a nonporous external membrane coating on the release profile.	86
39	The release profile of benzoic acid from a silicone coated hollow fiber.	87
40	The effect of a wax coating on the release profile of benzoic acid from nylon porous hollow fibers.	88
41	Simultaneous release of nicotine and caffeine from separate reservoirs.	90
42	Release profiles for benzoic acid and nicotine from separate reservoirs.	91
43	Simultaneous release of nicotine and benzoic acid from nylon porous hollow fibers.	facing 92
44	Simultaneous release of nicotine and caffeine from a single reservoir.	93
45	Controlled release of an agent initially a pure liquid contained in the reservoir.	94
46	pH dependence of release profile for the peptide Phe-Gly	96
47	DPPC release profile	98
48	The effect of membrane resistance on the release profiles, for $m_{1,2} = 1$	100
49	The effect of membrane resistance on the release profile, for $m_{1,2}=100$	100
50	The effect of partitioning on the release profile, for $\psi=0.1$	101

LIST OF FIGURES
(Continued)

Figure **Page**

51 The effect of τ_m/τ_w on the release profile. $\psi = 1$, $m_{1,2} = 100$. 102

52 The effect of membrane resistance on the release profile for the hollow fiber configuration, $m_{1,2}=100$ 103

53 The effect of partitioning on the release profile from a hollow fiber device, $\psi = 0.1$ 104

54 Concentration profiles within the hollow fiber device, $m_{1,2} = 100$ and $\psi = 1$ 105

55 Concentration profiles within the hollow fiber device, $m_{1,2} = 100$ and $\psi = 0.1$ 106

56 Concentration profiles in the hollow fiber device for $m_{1,2} = 100$ and $\psi = 1.0$ 108

57 The effect of dissolution rate control on the release profile from a flat membrane device; $\tau_r/\tau_m = 1$ 109

58 The effect of ζ on the release profile; $\tau_d/\tau_r = 100$ 110

59 The effect of the coating thickness on the release profile 111

CHAPTER 1

INTRODUCTION

1.1 Background

Controlled release has gained increasing attention recently in medical, pharmaceutical, cosmetic, consumer product, and agricultural industries. Controlled release systems are designed to deliver an agent at a specific rate for a definite period of time. Some controlled release systems are designed to release an agent in response to environmental conditions, while others are built to deliver an agent at a constant rate over a period of time (zero order release).

Traditional methods of drug delivery include ingestion and injection. Controlled release technology has modified injectables and ingestibles to provide a supply of the drug over long periods of time: instead of taking an ordinary tablet four times per day, one might take a controlled release tablet once per day. Transdermal patches and medical implants have also been introduced as alternatives to traditional methods of drug delivery. A transdermal patch provides delivery of an agent through the skin for a period on the order of days; a medical implant delivers an agent for months or years. Pesticides, fertilizers, and other agricultural agents are traditionally applied by spraying or broadcasting. Controlled release tape or strip dispensers, and

erodible matrix systems provide attractive alternatives to these traditional methods.

With traditional methods of application or delivery of an agent, there is a transient increase and decrease in agent concentration levels: after administration of the agent there is a rapid rise in concentration of the agent, which sometimes reaches a toxic level before decreasing eventually to an ineffective level. Because a controlled release system is able to extend the duration of the agent's activity while maintaining an effective concentration of the agent, these periods of toxicity and ineffectiveness are avoided. Since less total agent is used, the system is more economical. There is reduced exposure to toxic compounds. For delivery of pharmaceutical agents and drugs, controlled release ensures improved patient compliance and sometimes localized delivery of the agent to the desired site.

Three basic types of controlled release mechanisms are reviewed by Langer [1]. These mechanisms are diffusion, chemical reaction, and solvent activation. Most controlled release systems used today employ one or more of these basic mechanisms for rate control, and there is a wide range of formulation of systems which use these mechanisms. This work involves diffusion-based controlled release, and a brief description of previously developed diffusion-based systems will provide the background necessary for a full description of the novel device presented and studied in this work.

There are two basic types of diffusion-based controlled release systems: the membrane system and the matrix system. The membrane system

consists of a reservoir initially containing the agent, surrounded by a rate controlling polymer (i.e. membrane) or film. Over a period of time the agent diffuses out through the membrane into the surroundings. Glade Plug Ins are a familiar example of a membrane type device. When the device is plugged in, the fragrance-containing reservoir warms; this increases the permeability of the fragrance through the membrane and controlled release of the fragrance is provided for a period of about 45 days. In a diffusion-based matrix system, an agent is initially dissolved or dispersed uniformly in a polymer matrix, and over a period of time the drug diffuses out of the matrix and into the surroundings, leaving the matrix intact. Flea collars commonly used to keep pets pest free for several months are a familiar example of a diffusion-based matrix type controlled release device.

1.2 Controlled Release Using Aqueous-Organic Partitioning

The release rate of an agent from a diffusion-limited reservoir-type polymeric controlled release device is governed mainly by the agent's rate of diffusion across the polymeric surface which surrounds the reservoir. The diffusion rate is typically influenced by the agent concentration in the reservoir, the agent solubility and diffusivity in the membrane, and the membrane thickness.

The novel device described here employs a rate-controlling mechanism based on the *partitioning of the agent between an aqueous and an organic*

phase. The device comprises an agent-containing reservoir bounded by a microporous /porous membrane, either a hollow fiber or a flat film. The pores of the microporous /porous membrane are filled with a liquid immiscible and sparingly soluble in the reservoir phase fluid. The agent partitions between an aqueous phase and an organic phase at the interface of the reservoir and the membrane pore mouth. The microporous /porous membrane may be uncoated, or coated with a layer of thin, nonporous material. The physical

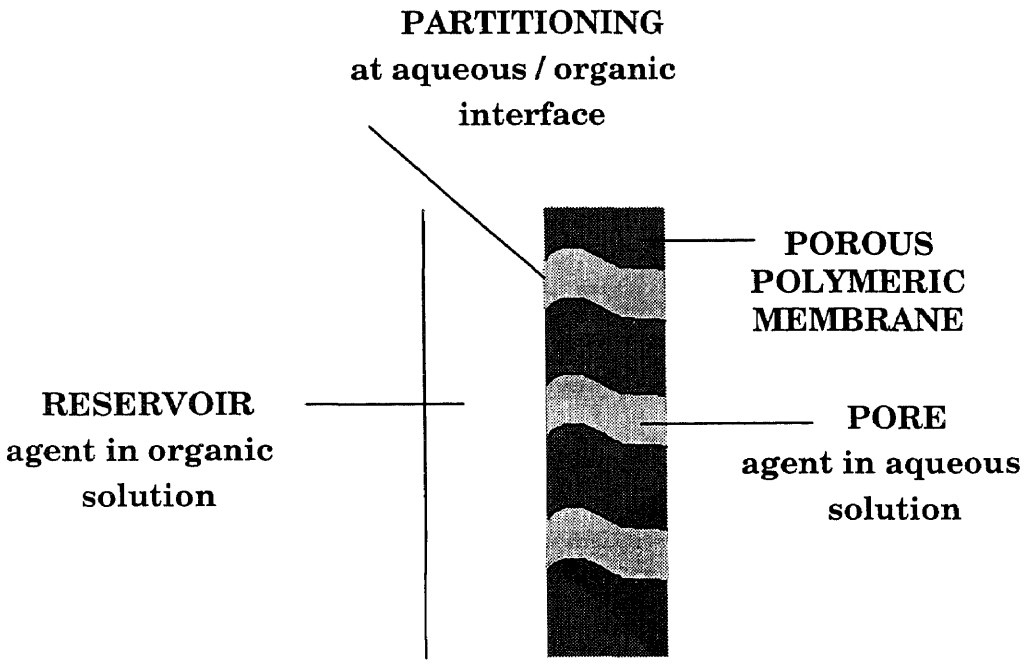

Figure 1. The physical situation in a membrane controlled release device using aqueous-organic partitioning

situation is shown in Figure 1 for the case of an agent in an organic reservoir solution, partitioning into water-filled pores. The device is physically similar to the membrane-bound reservoir device patented by Zaffaroni [2], but it

exploits aqueous-organic partitioning as a mechanism which drastically reduces the driving force for diffusion. As a result, longer term release can be achieved using a saturated and concentrated solution of the agent in the reservoir phase; release can be extended and additional rate control provided by using an agent in excess of its solubility concentration in the reservoir. The reservoir phase may even contain the pure liquid agent as long as it is immiscible with the pore liquid. Figure 2 shows the effect of partitioning at the aqueous-organic interface between the reservoir and the pore on the driving force for diffusion across the membrane.

The system described by Zaffaroni [2] lists a variety of reservoir and pore phases without suggesting specific combinations involving aqueous-organic partitioning. Further, the rate of diffusion through the membrane pores is suggested to be governed mainly by the diffusivity of the agent in the pore liquid, and the membrane tortuosity and porosity. In order to keep the driving force for diffusion across the membrane to a minimum, Zaffaroni's device requires an extremely low concentration of agent in the reservoir. This is in direct contrast to the device described here, in which the agent's reservoir concentration is ideally kept high, and the driving force for diffusion across the membrane is reduced by partitioning.

Physically, all systems in which the agent exists in two or more phases (i.e., a reservoir and a membrane) have partitioning of the agent between the two phases. The existence of partitioning in membrane-based controlled

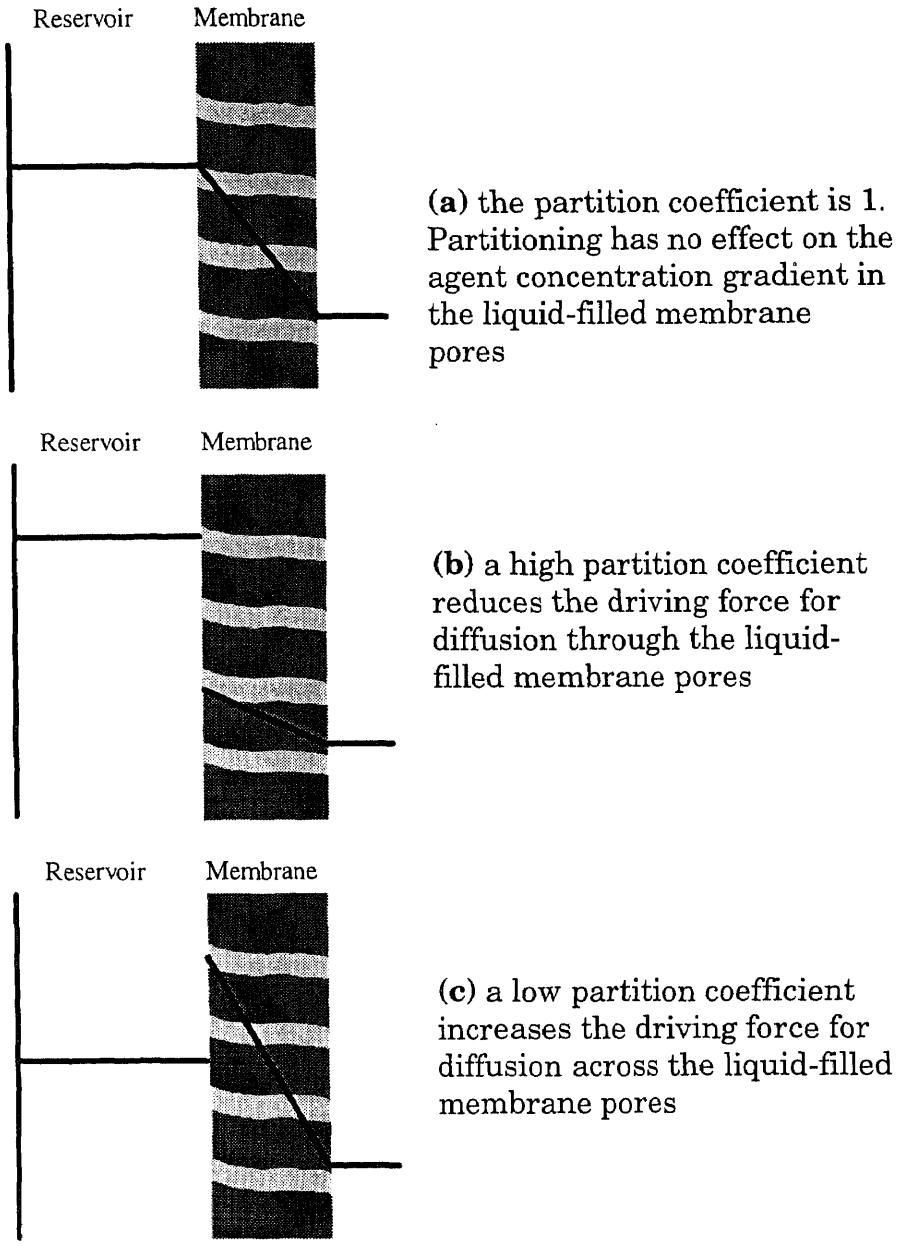

Figure 2. The effect of partitioning on the driving force for diffusion across the liquid-filled membrane pores

release systems has been pointed out by several authors [2], [3], [4], [5], [6], but aqueous-organic partitioning of the agent between the donor reservoir

solution and the pore fluid has not been experimentally or conceptually exploited as a means for controlling the release rate across the membrane by reducing the driving force for diffusion. In addition, the microporous /porous membrane with liquid-filled pores is a fundamental and distinguishing feature of the device used here, which affords flexibility in rate control that a nonporous membrane does not.

Many higher molecular weight nonpolar agents have significant solubility in common polymeric membrane materials a factor which is undesirable when a slow release rate is desired. The same agent, however, is likely to have a low solubility in water. By immobilizing water in the pores of a membrane, a small concentration difference across the membrane exists, and hence a slow release rate of such an agent is effected. If the agent were present in aqueous solution in the reservoir, the total amount of the agent present in solution would be limited by its low solubility in water. The potential to have a much larger total amount of agent initially present in the reservoir is provided by introducing the agent in an organic solution. The agent partitions between the organic (reservoir) phase and the aqueous (pore) phase at the interface between the reservoir and the pore mouth, with a high concentration on the organic side and a low concentration on the aqueous side. This system has the advantage of having a large amount of agent present initially in the reservoir, while maintaining a relatively slow rate of diffusion through the membrane. This is aqueous-organic partition-based controlled release.

Conversely, the controlled release of an agent having a high solubility in water may be achieved by containing an aqueous solution of the agent in the reservoir, with partitioning at the interface of the reservoir and the organic-filled pores.

So far the agent was considered to be in aqueous or organic solution in the reservoir of the device. However, higher agent loading can be achieved if the agent is present initially in suspension. Provided dissolution of the agent is fast compared with its rate of diffusion across the membrane, a zero order release rate, still controlled by aqueous-organic partitioning, will be achieved for an extended time. If, however, the interfacial mass transfer between the dispersed solids and the surrounding reservoir solution is slow relative to diffusion across the membrane, both dissolution and partitioning will control the release kinetics. In this case the device will be a hybrid membrane/matrix diffusion limited controlled release system.

This type of controlled release device, usable as a patch or an implant, has a number of advantages over conventional polymeric membrane devices. Device formulation is easy since the agent is contained in a preformed reservoir in aqueous or organic solution, eliminating the need for incorporating the agent into a polymer matrix and shaping it. Agent stability during the loading process is not a problem (this is a concern, for example, with heat sensitive agents during spinning conditions used in the melt spinning technique described by Dunn et al. [7]). Solvent handling problems such as arise in the wet phase inversion process for fibers [8] are avoided

since minimal amounts of solvent are used. By careful selection of the solvent/solute (agent)/membrane system, enormous flexibility in the rate of release of agent can be achieved: microporous membranes designed to be highly efficient mass transfer devices can be used instead as rate controlling devices. A zero order release rate can be achieved for an extended period of time; multiple agents may be released easily at different rates.

The objectives of this thesis are to demonstrate the feasibility and potential for microporous /porous membranes as controlled release devices in aqueous-organic partition-based systems, and provide a basic analysis as well as information regarding the agent release rates. The work is extended to include the hybrid matrix/membrane diffusion limited system described above. Various solvent/agent systems were investigated, different membranes were studied, and the significance of various factors in achieving zero order release of the agent are evaluated.

The controlled release behavior for systems having partitioning of the agent (1) between an organic reservoir solution and water-filled pores, and (2) between an aqueous reservoir solution and organic-filled pores was investigated using different agent/solvent systems.

The first systems studied used an agent in *solution* in an aqueous or organic reservoir. The organic reservoir/water-filled pores systems employed benzoic acid as an agent, and decanol or octanol as reservoir solvents. The aqueous reservoir/organic-filled pores systems studied used aqueous nicotine

in the reservoir, partitioning into pores filled with mineral oil. In these studies, benzoic acid was chosen as a model agent because of its high solubility in organic solvents such as octanol and decanol, and its relatively low solubility in water. Partitioning of benzoic acid between an aqueous phase and an organic phase such as octanol or decanol greatly favors the latter. Nicotine was chosen as a model agent for its high solubility in water: it was therefore an ideal candidate for high loading in an aqueous reservoir.

The release of an agent originally present in suspension is investigated for two rate-limiting types of release. Benzoic acid dissolves rather quickly into decanol and into octanol, and thus its release is likely to be partition-controlled. Benzoic acid dissolves very slowly into mineral oil; therefore its release is slowed by the dissolution step.

Caffeine has a moderately low solubility in both water and organic solvents like octanol. Despite its low solubility in water, dispersed caffeine dissolves fairly quickly into water. It was used to study the effect of fast agent dissolution into an aqueous reservoir.

Toluene has a low solubility in water, and is immiscible with water. Toluene was used as a model agent to investigate controlled release of a pure liquid agent in the reservoir, with rate control provided by limited solubility of toluene in the water-filled pores of the membrane.

This type of device lends itself quite naturally to effecting controlled release of two agents at different rates. This can be done using two agents in the same reservoir, being released simultaneously through the same

membrane. Alternatively, the two agents can exist in two separate and contiguous reservoirs bounded by separate membranes. The latter situation provides additional flexibility in controlling the release rate of each agent.

For simultaneous release of two agents from a single reservoir, the system of caffeine and nicotine in an aqueous reservoir diffusing through octanol-filled membrane pores was studied. For independent simultaneous release from two separate reservoirs, a system with one aqueous reservoir and one organic reservoir was studied: in the first reservoir was aqueous nicotine, which partitioned into octanol-filled pores; in the second reservoir was benzoic acid in octanol, which partitioned into water-filled pores.

The aqueous-organic partition coefficient of peptides is highly pH dependent. pH based controlled release of the dipeptide Phe-Gly was investigated using this type of aqueous-organic partition based system. Phe-Gly was present in an aqueous reservoir, and partitioned into octanol-filled pores. The configuration of the device is similar to the supported liquid membrane configuration used by Wong et al. [9] for peptide separation. This system was studied for a range of pH, to determine the effect of pH on the release profile.

Several types of microporous /porous hollow fibers and flat membranes (both hydrophilic and hydrophobic) were used to illustrate experimentally the influence of geometric parameters on the rate of release of solute through water-filled pores. Release rates from systems using a flat microporous film were also studied. Finally the release rates from silicone-coated hydrophobic

microporous hollow fibers, and the same fibers with an additional wax coating were studied. The performances of coated and noncoated fibers were compared to evaluate the effectiveness of the silicone and wax coatings in reducing the rate of agent release.

A model was developed for the following types of systems studied: dissolved agent in the reservoir (hollow fiber, coated hollow fiber, and flat film), and dispersed agent in the reservoir (both hollow fiber and flat film). Analytical solutions were obtained, and concentration profiles as a function of distance and time in the reservoir and the membrane regions of the device, as well as transient concentrations of the surrounding well stirred water bath are provided. Since the model considers concentration gradients within the device as well as the transient concentration of the surrounding bath, it provides a more realistic simulation of the physical situation than some of the simplified models used commonly to describe other diffusion-limited systems where the surrounding bath is assumed to have zero concentration always.

1.3 Liposome Release Studies

This thesis is primarily and overwhelmingly focused on aqueous-organic partitioning based controlled release of model agents present in a solution in a reservoir with or without an additional amount of agent being in suspension. A totally different concept of controlled release of agents has also been explored very briefly in this thesis. This involves liposomes in aqueous

solutions and their controlled release through microporous /porous membranes.

Liposomes are hollow structures having dimensions between 0.005 μm and 100 μm, made of a phospholipid shell, the structure of which is similar to biological membranes. This structure in an aqueous medium is that of a lipid bilayer in which the polar hydrophilic phosphate groups form the surfaces of a sandwich and the nonpolar hydrophobic tail is on the interior of the sandwich, protected from water. The structure is stabilized by wrapping itself into a sphere-like form which is relatively stable. The sphere-like shell encapsulates a liquid interior which can contain such substances as peptides and proteins, hormones, enzymes, antibiotic and antifungal agents, anticancer agents, DNA and whole virus [10, 11, 12]. Because of the structural similarity between the lipid bilayer and cell membranes, liposomes can penetrate cells effectively, and can act as drug carriers which effectively deliver drugs to cells that a free drug would not penetrate. Liposomes also have cosmetic, environmental, diagnostic, agrochemical, and other nonmedical applications [13]. Small liposomes that have a single lipid bilayer are called small unilamellar vesicles (SUVs). Large many-layered liposomes are sometimes produced; they are known as MLVs, multilamellar vesicles.

A free drug injected into the blood stream typically achieves a therapeutic concentration for a short duration, due to metabolism and excretion. Drugs encapsulated by liposomes achieve a therapeutic level for a

longer duration, as the drug must first be released from the liposome before metabolism or excretion. Conventional liposomes (CL) introduced by intravenous injections encounter rapid uptake by phagocytic cells of the immune system, predominantly in the liver and spleen [12]. Sterically stabilized liposomes exhibit specific reactivity and stay much longer in the blood. A drug encapsulated in a Stealth liposome (one capable of avoiding a macrophage in the circulatory system) will remain in a therapeutic concentration for a longer duration [12]. The drug or active agent inside the liposome may be released from the liposome into the cell by one or more of three mechanisms [11]: a) a liposome adsorbing onto the cell membrane may release its cargo into the extracellular fluid -- this could allow the free drug to pass through the cell membrane into the cell, b) if the liposome fuses with the cell membrane and releases the drug directly into the cytoplasm and c) if the liposome is ingested by endocytosis and is subjected to degradation by organelles called lysosomes inside the cell, which releases the liposome cargo into the cytoplasm.

If it is desired to maintain the desired therapeutic concentration of the drug in the body for a still longer period of time, there must be some way to control the delivery of the drug from the liposome to the cell. One way to do this would be to control the release of liposomes to the body or circulation system, which would, in turn, control the release of the drug from the liposomes to the cell. A method for controlling the release rate of the liposomes from the source (the reservoir of a controlled release device) is

presented in this thesis. The technique uses a device similar to the one described above in Section 1.2, except that the reservoir and pore fluids in the device comprise one continuous phase rather than an aqueous and an organic phase. The reason for using one continuous phase is that the liposomes assemble themselves to orient their bilayer according to the properties of the surrounding solution. Since the orientation of the bilayer of a liposome in aqueous (or polar) solution would be the reverse of its orientation in organic (non-polar) solution, using aqueous-organic partitioning would jeopardize the integrity of the lipid bilayer.

In this technique for controlled release of liposomes from an aqueous medium, the reservoir contains the liposomes in an aqueous solution. The reservoir is bound by a porous membrane, the pores of which are large enough to accommodate the passage of the liposomes. The pores of the membrane are filled with water, and the liposomes are delivered into aqueous surroundings. No specific driving force is employed except that of Brownian motion.

The objectives of this part of the thesis are to establish the potential for and feasibility of controlled release of liposomes using a porous membrane system, and to determine the basic release characteristics of the liposomes from the system.

CHAPTER 2

MATHEMATICAL MODEL

2.1 Introduction

2.1.1 The Problem

The mathematical model which predicts the release profile of an agent from the hollow fiber or flat membrane controlled release device described in detail in Chapter 1 is presented here. The model describes the release of small molecules such as nicotine, caffeine, and benzoic acid. Both hollow fiber and flat membrane device geometries are considered. Initially the solute to be released is present in solution or suspension in a reservoir. The reservoir is bounded by a microporous membrane, with or without a thin nonporous coating, the pores of which are filled with a pore liquid immiscible with the reservoir phase. The coating, if present, would be on the side not adjacent to the reservoir, i.e. on the outer surface of the membrane. The device is submerged in a well stirred water bath. The solute diffuses through the reservoir toward the membrane. At the interface between the reservoir and the pore, the solute partitions between the reservoir and the pore liquid phases. It continues to diffuse through the pore toward the external bath. At the interface between the pore and the external bath, the solute partitions again between phases (this may be one continuous phase if the pores are filled with water). The well stirred bath concentration changes with time. In

Sections 2.2 and 2.3 a solution of the agent in the reservoir is considered. An additional dispersed phase of the agent in the reservoir is considered in Sections 2.4 and 2.5.

Problems similar to this one have appeared in the literature, mainly with applications to heat transfer. Heat transfer problems typically have boundary conditions different from those posed herein. In heat transfer there is continuity of fluxes with no partitioning at the boundaries: the temperatures of two different phases are equal at the interface unless there is a contact resistance. In addition, problems addressing conduction in composite cylinders have been solved subject to the following surface conditions: insulation [14], constant temperature [15], exposure to an infinite medium initially at zero temperature [16].

Barrer [17] presents a formal theory for diffusion through membranes, for microporous membranes as well as laminates, but this development considers the case where the concentrations on both sides of the membrane are constant.

2 .1.2 Overview of the Solution Procedure

Models for the following physical situations have been developed: reservoir containing agent in solution (noncoated flat membrane, noncoated microporous hollow fiber, and coated hollow fiber), and reservoir containing dispersed as well as dissolved agent (noncoated flat membrane and noncoated microporous hollow fiber). Governing equations were solved analytically

together with their boundary and initial conditions in the Laplace domain. These solutions were inverted numerically to obtain concentration profiles of the agent within the device as functions of distance and time.

2.1.3 Simplifying Assumptions

In developing the models several simplifying assumptions were made. The following assumptions apply to each model presented (assumptions associated with particular models will be presented in the corresponding section):

- The diffusion of the agent molecules is Fickian.
- There are no interfacial boundary layers.
- Diffusivities of the agent in the reservoir and the pore liquid phase are independent of concentration.
- The aqueous-organic partition coefficient for the agent is independent of concentration.
- The surrounding water bath is perfectly stirred.
- The aqueous and organic phases are completely immiscible.
- The organic solvent and water are completely insoluble in each other.
- Reservoir and bath solutions are ideal.
- Temperature is uniform.

2.2 Agent Dissolved in the Reservoir: Problem Formulation

2.2.1 Dimensional Equations

Consider first the situation in which the agent initially exists in solution in the reservoir at a concentration below its saturation concentration in the carrier (solvent). Figure 3 represents the device in either flat or cylindrical geometry, with the distance variable designated x. Region 1, ($0 \leq x \leq a$),

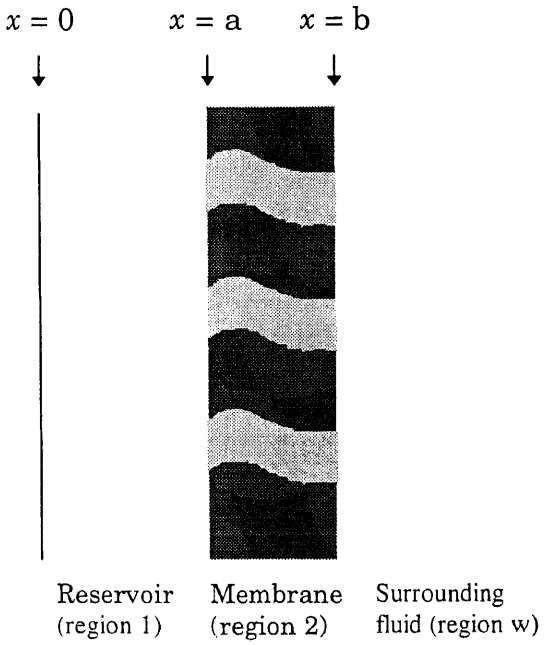

Figure 3. The reservoir and membrane regions of the controlled release device. The agent partitions between an aqueous and an organic phase at x = a, and is released to well-stirred surroundings at x = b.

represents the reservoir where the agent is in solution in an aqueous or organic carrier, or is in pure liquid form. Region 2 represents the membrane,

and extends from $x = a$ to $x = b$. The surrounding water bath, $x \geq b$, is designated region w. The governing equation for the concentration of agent in the reservoir (region 1) is

$$D_1 \nabla^2 C_1 = \frac{\partial C_1}{\partial t} \tag{1}$$

The governing equation for the agent in the membrane (region 2) is

$$D_2 \nabla^2 C_2 = \frac{\partial C_2}{\partial t} \tag{2}$$

Here D_2 is the effective diffusivity of the agent in the membrane, and is expressed as its free diffusivity in the pore liquid medium times the ratio of membrane porosity and tortuosity:

$$D_2 = \frac{D_{free}\varepsilon}{\tau} \tag{3}$$

Diffusivity is assumed to be independent of concentration.

The boundary conditions are:

$$C_1(0,t) < \infty \tag{4}$$

$$C_1(a,t) = m_{1,2} C_2(a,t) \tag{5}$$

$$D_1 \nabla C_1(a,t) = D_2 \nabla C_2(a,t) \tag{6}$$

$$V_w \frac{\partial C_2}{\partial t}(b,t) = -D_2 \alpha_2 m_{2,w} \nabla C_2(b,t) \tag{7}$$

The first boundary condition reflects the condition that the concentration at $x = 0$ (the center of the hollow fiber or the bottom of the reservoir) must be finite. The second boundary condition represents equilibrium partitioning at the interface between the reservoir and the pore liquid phases; $m_{1,2}$ is the

equilibrium partition coefficient, assumed to be independent of concentration ($m_{i,j}$ is defined as the ratio of equilibrium concentrations of agent in region i to region j):

$$m_{i,j} = \frac{C_i}{C_j} \tag{8}$$

The third boundary condition represents continuity of agent flux across the interface between the reservoir and the pore. The last boundary condition is a simple unsteady state material balance on the agent at the outer wall of the membrane; $m_{2,w}$ is the equilibrium partition coefficient between the pore phase and the water bath (equal to 1 if both are water and the pore does not impose any steric restrictions to solute partitioning etc.) and α_2 is the membrane area at the outer wall.

The initial conditions indicate that all of the agent is initially present in the reservoir, and none is present in the pore:

$$C_1(x \le a, 0) = C_1^0 \tag{9}$$

$$C_2(a \le x \le b, 0) = 0 \tag{10}$$

2 .2.2 Dimensionless Equations

Nondimensionalizing the equations facilitates identification of important design parameters. First, the following dimensionless variables are introduced:

$$\xi_1 = \frac{x}{a} \qquad \xi_2 = \frac{x}{(b-a)} \qquad U_n = \frac{C_n}{C_1^0} \qquad (n = 1, 2, s) \tag{11}$$

The dimensionless group ξ_n was introduced separately for each region to emphasize the importance of the different characteristic lengths in each region. Writing equations (1) and (2) in semidimensionless form (leaving t dimensional), the characteristic times for diffusion through each region become apparent:

$$\nabla^2 U_1 = \frac{a^2}{D_1}\frac{\partial U_1}{\partial t} \tag{12}$$

$$\nabla^2 U_2 = \frac{(b-a)^2}{D_2}\frac{\partial U_2}{\partial t} \tag{13}$$

The boundary conditions, in semidimensionless form, are

$$U_1(0,t) < \infty \tag{14}$$

$$U_1(\xi_1 = 1, t) = m_{1,2} U_2(\xi_2 = 0, t) \tag{15}$$

$$\frac{D_1}{a}\nabla U_1(\xi_1 = 1, t) = \frac{D_2}{b-a}\nabla U_2(\xi_2 = 0, t) \tag{16}$$

$$V_w \frac{\partial U_2}{\partial t}(\xi_2 = 1, t) = -\frac{D_2 \alpha_2 m_{2,w}}{b-a}\nabla U_2(\xi_2 = 1, t) \tag{17}$$

In order to make the equations fully dimensionless, the following dimensionless times are defined:

$$\theta_1 = \frac{D_1 t}{a^2} \qquad \theta_2 = \frac{D_2 t}{(b-a)^2} \tag{18}$$

Note also that the appropriate diffusivity for each region is used in the expression for θ_n. ξ_n and θ_n are, of course, related to x and t respectively. In order to solve the coupled differential equations, the independent variables

for each region must be the same. Let these general independent variables be called ξ and θ. ξ and θ are chosen to be normalized using b and D_1; this choice of definition has no bearing on the ultimate solution of the problem.

$$\xi = \frac{x}{b} \qquad \theta = \frac{D_1 t}{b^2} \tag{19}$$

The intermediate step of defining ξ_n and θ_n was performed to demonstrate how meaningful time constants are extracted from the governing equations. In dimensionless form using ξ and θ, the governing equations for regions 1 and 2 (equations (1) and (2)) are

$$\nabla^2 U_1 = \frac{\partial U_1}{\partial \theta} \tag{20}$$

$$\nabla^2 U_2 = \frac{D_1}{D_2}\frac{\partial U_2}{\partial \theta} \tag{21}$$

Nondimensionalizing the boundary conditions in equations (4) through (7) yields:

$$U_1(0,t) < \infty \tag{22}$$

$$U_1\left(\frac{a}{b},\theta\right) = m_{1,2} U_2\left(\frac{a}{b},\theta\right) \tag{23}$$

$$D_1 \nabla U_1\left(\frac{a}{b},\theta\right) = D_2 \nabla U_2\left(\frac{a}{b},\theta\right) \tag{24}$$

$$\frac{V_w D_1}{b m_{2,w} D_2 \alpha_2}\frac{\partial U_2}{\partial \theta}(1,\theta) = -\nabla U_2(1,\theta) \tag{25}$$

The dimensionless initial conditions are

$$U_1\left(\xi \le \frac{a}{b}, 0\right) = 1 \tag{26}$$

$$U_2\left(\frac{a}{b} \le \xi \le 1, 0\right) = 0 \tag{27}$$

Next the dimensionless governing equations for regions 1 and 2, and the boundary conditions are transformed with respect to θ into the Laplace domain. The governing equations (20) and (21) become

$$\nabla^2 \overline{U}_1 = -1 + s\overline{U}_1 \tag{28}$$

$$\nabla^2 \overline{U}_2 = \frac{D_1}{D_2} s\overline{U}_2 \tag{29}$$

Upon transformation of the boundary conditions, the following are obtained:

$$\overline{U}_1(0; s) < \infty \tag{30}$$

$$\overline{U}_1\left(\frac{a}{b}; s\right) = m_{1,2}\overline{U}_2\left(\frac{a}{b}; s\right) \tag{31}$$

$$D_1 \nabla \overline{U}_1\left(\frac{a}{b}; s\right) = D_2 \nabla \overline{U}_2\left(\frac{a}{b}; s\right) \tag{32}$$

$$\frac{V_w D_1}{b m_{2,w} D_2 \alpha_2} s\overline{U}_2(1, \theta) = -\nabla \overline{U}_2(1, \theta) \tag{33}$$

2.3 Agent Dissolved in the Reservoir: Solutions

2.3.1 Flat Membrane (Cartesian Coordinates)

The solutions to the governing differential equations for regions 1 and 2 in the Laplace domain (equations (28) and (29)), are

$$\overline{U}_1 = \frac{1}{s} + A_1 \exp\left[\xi\sqrt{s}\right] + A_2 \exp\left[-\xi\sqrt{s}\right] \tag{34}$$

$$\overline{U}_2 = A_3 \exp\left[\xi\sqrt{\frac{D_1}{D_2}s}\right] + A_4 \exp\left[-\xi\sqrt{\frac{D_1}{D_2}s}\right] \tag{35}$$

Where each A_i is a constant of integration. Next the boundary conditions (equations (22) through (25)) are written in terms of the solutions for regions 1 and 2 in the Laplace domain

$$A_1\sqrt{s} - A_2\sqrt{s} = 0 \tag{36}$$

$$\frac{1}{s} + A_1 \exp\left[\frac{a}{b}\sqrt{s}\right] + A_2 \exp\left[-\frac{a}{b}\sqrt{s}\right] = m_{1,2}\left(A_3 \exp\left[\frac{a}{b}\sqrt{\frac{D_1}{D_2}s}\right] + A_4 \exp\left[-\frac{a}{b}\sqrt{\frac{D_1}{D_2}s}\right]\right) \tag{37}$$

$$D_1\left(A_1\sqrt{s}\exp\left[\frac{a}{b}\sqrt{s}\right] - A_2\sqrt{s}\exp\left[-\frac{a}{b}\sqrt{s}\right]\right) = D_2\left(A_3\sqrt{\frac{D_1}{D_2}s}\exp\left[\frac{a}{b}\sqrt{\frac{D_1}{D_2}s}\right] - A_4\sqrt{\frac{D_1}{D_2}s}\exp\left[-\frac{a}{b}\sqrt{\frac{D_1}{D_2}s}\right]\right) \tag{38}$$

$$-\alpha_2 D_2\left(A_3\sqrt{\frac{D_1}{D_2}s}\exp\left[\sqrt{\frac{D_1}{D_2}s}\right] - A_4\sqrt{\frac{D_1}{D_2}s}\exp\left[-\sqrt{\frac{D_1}{D_2}s}\right]\right) = \frac{V_w D_1 s}{b m_{2,w}}\left(A_3\exp\left[\sqrt{\frac{D_1}{D_2}s}\right] + A_4\exp\left[-\sqrt{\frac{D_1}{D_2}s}\right]\right) \tag{39}$$

Equations (36) through (39) are solved simultaneously using Mathematica® (Wolfram Research, 1991) to obtain expressions for A_1, A_2, A_3, and A_4 as functions of s (see Appendix 1). These expressions are substituted into the analytical solutions for regions 1 and 2 given in equations (34) and (35), which are then inverted numerically by the IMSL (Visual Numerics, Houston, TX) subroutine DINLAP which uses the method of de Hoog [18]. The program was run on a VAX VMS Version 5.5-2 operating system using a DEC Fortran compiler Version 6.0-1 for OpenVMS VAX Systems. The FORTRAN code is listed in Appendix 1.

The results are concentration profiles of the agent in regions 1 and 2 as functions of dimensionless distance and time. The concentration of the agent in the well stirred water bath is related to the concentration of the agent at the outer wall of the membrane ($\xi = 1$) by

$$U_w = \frac{U_2(1,\theta)}{m_{2,w}} \tag{40}$$

2 .3.2 Hollow Fiber (Cylindrical Coordinates)

The solutions to the governing differential equations for regions 1 and 2 in the Laplace domain expressed by equations (28) and (29) in cylindrical coordinates are:

$$\overline{U}_1 = \frac{1}{s} + A_5 I_0\left(\xi\sqrt{s}\right) + A_6 K_0\left(\xi\sqrt{s}\right) \tag{41}$$

$$\overline{U}_2 = A_7 I_0\left(\xi\sqrt{\frac{D_1}{D_2}s}\right) + A_8 K_0\left(\xi\sqrt{\frac{D_1}{D_2}s}\right) \tag{42}$$

Where $I_0(x)$ and $K_0(x)$ are modified Bessel functions of the 0^{th} kind. Equation (30) can be applied immediately to equation (41) to obtain $A_6 = 0$. The remaining boundary conditions, written in terms of the solutions given by equations (41) and (42) are

$$\frac{1}{s} + A_5 I_0\left(\frac{a}{b}\sqrt{s}\right) = m_{1,2}\left(A_7 I_0\left(\frac{a}{b}\sqrt{\frac{D_1}{D_2}s}\right) + A_8 K_0\left(\frac{a}{b}\sqrt{\frac{D_1}{D_2}s}\right)\right) \tag{43}$$

$$A_5 I_1\left(\frac{a}{b}\sqrt{s}\right) = \sqrt{\frac{D_2}{D_1}}\left(A_7 I_1\left(\frac{a}{b}\sqrt{\frac{D_1}{D_2}s}\right) - A_8 K_1\left(\frac{a}{b}\sqrt{\frac{D_1}{D_2}s}\right)\right) \tag{44}$$

$$-\frac{\alpha_2 m_{2,w} b}{V_w}\left(A_7 I_1\left(\sqrt{\frac{D_1}{D_2}}s\right)-A_8 K_1\left(\sqrt{\frac{D_1}{D_2}}s\right)\right)= \sqrt{\frac{D_1}{D_2}}s\left(A_7 I_0\left(\sqrt{\frac{D_1}{D_2}}s\right)+A_8 K_0\left(\sqrt{\frac{D_1}{D_2}}s\right)\right) \tag{45}$$

These expressions are solved simultaneously by Mathematica® to obtain expressions for A_5, A_7, and A_8. Upon substitution of the expressions for A_5, A_7, and A_8 into equations (41) and (42) numerical inversion yields concentration profiles of the agent in regions 1 and 2 as functions of dimensionless distance and time. The FORTRAN code is listed in Appendix 1. The concentration of the agent in the well stirred water bath is related to the concentration of the agent at the outer wall of the membrane ($\xi = 1$) by equation (40).

2 .3.3 Coated Hollow Fiber

Modeling of controlled release using a microporous hollow fiber having a thin nonporous coating on the outside surface is an extension of the problem described in Section 2.3.2. The coated hollow fiber is shown in Figure 4.

In order to obtain time constants for each region, the governing equations are first written in semidimensionless form, as was described in Section 2.2.2 for the uncoated membrane:

$$\nabla^2 U_1 = \frac{a^2}{D_1}\frac{\partial U_1}{\partial t} \tag{12}$$

$$\nabla^2 U_2 = \frac{(b-a)^2}{D_2}\frac{\partial U_2}{\partial t} \tag{13}$$

$$\nabla^2 U_3 = \frac{(c-b)^2}{D_3}\frac{\partial U_3}{\partial t} \tag{46}$$

Where the coordinate for region 3 (the coating region) has been defined as ξ_3.

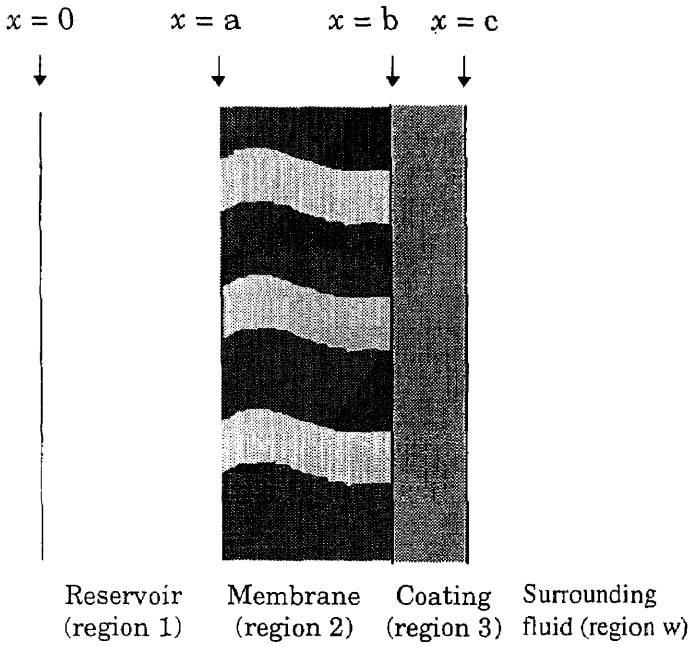

Figure 4. The coated hollow fiber

Next the equations are written in dimensionless form using dimensionless ξ and θ as independent variables. The governing equations for the reservoir (region 1) and the membrane (region 2) are the same as those for the uncoated hollow fiber; a third governing equation is introduced to describe diffusion through the coating (region 3) which extends from $x = b$ to $x = c$.

$$\nabla^2 U_1 = \frac{\partial U_1}{\partial \theta} \tag{20}$$

$$\nabla^2 U_2 = \frac{D_1}{D_2}\frac{\partial U_2}{\partial \theta} \tag{21}$$

$$\nabla^2 U_3 = \frac{D_1}{D_3}\frac{\partial U_3}{\partial \theta} \tag{47}$$

In equation (47) D_3 is the diffusivity of the solute in the coating material. The boundary conditions for the center as well as the reservoir-pore interface *at* $x = a$ $(\xi = \frac{a}{b})$ are the same as those for the uncoated hollow fiber. The boundary conditions at $x = b$, which is now the interface between the pore and the coating, are analogous to those at $x = a$ $(\xi = \frac{a}{b})$ where partitioning between phases occurs while continuity of fluxes exists.

$$U_1(0,t) < \infty \tag{22}$$

$$U_1\left(\frac{a}{b},\theta\right) = m_{1,2}U_2\left(\frac{a}{b},\theta\right) \tag{23}$$

$$D_1\nabla U_1\left(\frac{a}{b},\theta\right) = D_2\nabla U_2\left(\frac{a}{b},\theta\right) \tag{24}$$

$$U_2(1,\theta) = m_{2,3}U_3(1,\theta) \tag{48}$$

$$D_2\nabla U_2(1,\theta) = D_3\nabla U_3(1,\theta) \tag{49}$$

And finally, the boundary condition at the outer surface, $x = c$ $(\xi = \frac{c}{b})$ is analogous to the boundary condition expressed by equation (25)

$$\frac{V_w D_1}{b m_{3,w} D_3 \alpha_3}\frac{\partial U_3}{\partial \theta}\left(\frac{c}{b},\theta\right) = -\nabla U_2\left(\frac{c}{b},\theta\right) \tag{50}$$

The dimensionless initial conditions for regions 1 and 2 are identical to those for the uncoated fiber membrane; the initial condition for the coating reflects that there is initially no agent present in region 3.

$$U_1\left(\xi \le \frac{a}{b}, 0\right) = 1 \tag{26}$$

$$U_2\left(\frac{a}{b} \le \xi \le 1, 0\right) = 0 \tag{27}$$

$$U_3\left(1 \le \xi \le \frac{c}{b}, 0\right) = 0 \tag{51}$$

When the governing equations expressed by equations (20), (21) and (47) are written in the Laplace domain, the following expressions are obtained:

$$\nabla^2 \overline{U}_1 = -1 + s\overline{U}_1 \tag{28}$$

$$\nabla^2 \overline{U}_2 = \frac{D_1}{D_2} s\overline{U}_2 \tag{29}$$

$$\nabla^2 \overline{U}_3 = \frac{D_1}{D_3} s\overline{U}_3 \tag{52}$$

Following the same procedure described in Section 2.3.2, the following Laplace domain solutions are obtained for the reservoir, porous membrane and coating (regions 1, 2, and 3, respectively):

$$\overline{U}_1 = \frac{1}{s} + A_9 I_0\left(\xi\sqrt{s}\right) + A_{10} K_0\left(\xi\sqrt{s}\right) \tag{53}$$

$$\overline{U}_2 = A_{11} I_0\left(\xi\sqrt{\frac{D_1}{D_2}s}\right) + A_{12} K_0\left(\xi\sqrt{\frac{D_1}{D_2}s}\right) \tag{54}$$

$$\overline{U}_3 = A_{13} I_0\left(\xi\sqrt{\frac{D_1}{D_3}s}\right) + A_{14} K_0\left(\xi\sqrt{\frac{D_1}{D_3}s}\right) \tag{55}$$

The boundary condition at $x = 0$ can immediately be applied to the solution for region 1 to obtain $A_{10} = 0$. The boundary conditions are now written in terms of the solutions provided by the preceding equations to yield six algebraic equations that can be solved simultaneously using Mathematica® to obtain expressions for A_9 through A_{14} in terms of s. These expressions are shown in Appendix 1. The solutions are then inverted numerically using the IMSL subroutine DINLAP. The FORTRAN code is listed in Appendix 1. Finally, the surrounding bath concentration can be determined as a function of time using the relationship

$$U_w = \frac{U_3(1,\theta)}{m_{3,w}} \tag{56}$$

2.4 Dispersed Reservoir Phase: Problem Formulation

The problem treated here is an extension of the uncoated membrane device considered earlier, but now the reservoir is filled with dispersed as well as dissolved agent. A schema of the device with undissolved agent in the reservoir is shown in Figure 5. As described previously, the agent partitions between the reservoir and the pore liquid phases at their interface, and diffuses out through the membrane pore. However, in this case, as agent is depleted by diffusion through the membrane, it is replaced by agent dissolving into the reservoir solution. If dissolution is fast relative to

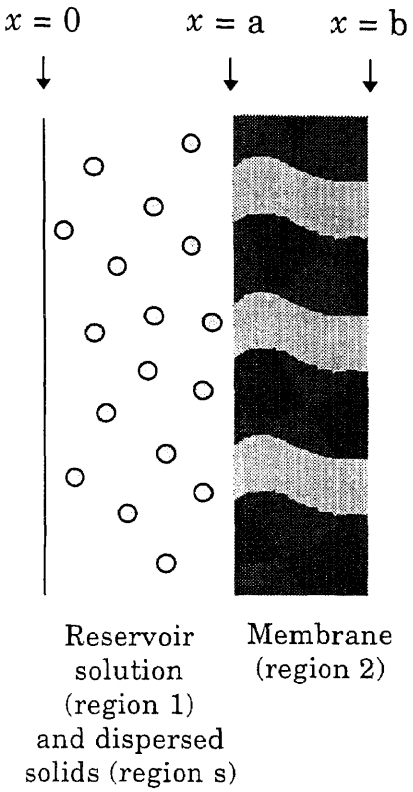

Figure 5. The controlled release device with dispersed solids in the reservoir

diffusion, the rate of release will be controlled primarily by aqueous-organic partitioning; if however, dissolution is slow, additional rate control will be provided. It should be noted that the solution in the reservoir would usually, but not always, be saturated initially. (If the agent dissolves very slowly into the reservoir solvent, it is possible that solids are added to the solvent and the experiment is begun before the solids reach equilibrium with the reservoir solvent.)

The physical situation that is assumed for the reservoir is similar to that described by Varelas et al.[19]. The dispersed agent occupies a constant

volume fraction ϕ; neglecting any change in volume fraction over time should be of little consequence since the volume fraction is typically well under 10%. The solid agent is assumed to occupy small spheres (it will become apparent that their geometry is unimportant), which are assumed to be dispersed evenly throughout the reservoir (the settling that may occur is neglected). The volume of each sphere is so small that diffusional resistance within the sphere may be neglected. The concentration of solid within each sphere changes with time as the solid dissolves into the reservoir phase. The concentration of the agent within the reservoir solution is a function of both distance and time.

The region of suspended solids is designated region s; regions 1 and 2 refer to the reservoir solution and the pore liquid phase, respectively. Equilibrium between the dispersed region and the reservoir exists at the interface between the two regions, and the concentrations are related by the pseudo-partition coefficient defined as $m_{1,s} = \left.\frac{C_1}{C_s}\right|_{eq}$, which is equivalent to a normalized solubility. Assuming that mass transfer across the interface between the dispersed and dissolved phases is first order, the governing equation for the dispersed region is (in semidimensionless and dimensionless form, respectively)

$$\frac{\phi}{m_{1,s}}\frac{\partial U_s^*}{\partial t} = -\frac{k}{V_1}\left(U_s^* - U_1^*\right) \qquad \textbf{(57)}$$

$$\frac{V_1 D_1 \phi}{b^2}\frac{\partial U_s}{\partial \theta} = -k\left(m_{1,s}U_s - U_1\right) \tag{58}$$

where k represents the product of the overall interfacial mass transfer coefficient at the interface between the dispersed and dissolved reservoir phases and the interfacial mass transfer area. In equation 57 the dimensionless concentrations used were written in terms of the saturation concentration in region 1, in order to facilitate identification of important time constants. The concentrations U_s^* and U_1^* are related to U_s and U_1 by

$$U_s^* = \frac{m_{1,s}U_s}{U_1^{sat}} \qquad U_1^* = \frac{U_1}{U_1^{sat}} \tag{59}$$

where,

$$U_1^{sat} = \frac{C_1^{sat}}{C_1^0} \tag{60}$$

The governing equation for the dissolved agent in the reservoir phase (in semidimensionless and dimensionless form, respectively) is

$$(1-\phi)\frac{\partial U_1}{\partial t} = (1-\phi)\frac{D_1}{a^2}\nabla^2 U_1 - \phi\frac{\partial U_s}{\partial t} \tag{61}$$

$$\frac{\partial U_1}{\partial \theta} = \nabla^2 U_1 - \frac{\phi}{1-\phi}\frac{\partial U_s}{\partial \theta} \tag{62}$$

And the governing equation for the pore liquid phase (in semidimensionless and dimensionless form respectively) is

$$\frac{D_2}{(b-a)^2}\nabla^2 U_2 = \frac{\partial U_2}{\partial t} \tag{13}$$

$$\nabla^2 U_2 = \frac{D_1}{D_2}\frac{\partial U_2}{\partial \theta} \tag{21}$$

Next the initial conditions are written.

$$U_s(0) = U_s^0 = \frac{C_s^0}{C_1^0} \tag{63}$$

$$U_1\left(\xi \le \frac{a}{b}, 0\right) = 1 \tag{26}$$

$$U_2\left(\frac{a}{b} \le \xi \le 1, 0\right) = 0 \tag{27}$$

Defining $\beta = \frac{kb^2}{V_1 D_1 \phi}$, the Laplace transform of equation (58) provides an expression for $\overline{U}_s$ in terms of $\overline{U}_1$, s, and other known quantities:

$$\overline{U}_s = \frac{\beta \overline{U}_1 + \frac{C_s^0}{C_1^0}}{s + \beta m_{1,s}} \tag{64}$$

Now equation (62) for the reservoir solution (region 1) can be written in the Laplace domain. Upon substitution of equation (64) for $\overline{U}_s$ into the Laplace domain governing equation for the reservoir solution, the following expression is obtained for region 1:

$$\nabla^2 \overline{U}_1 = \left[-1 - \frac{\phi}{1-\phi} \frac{C_s^0}{C_1^0} + \frac{s \frac{\phi}{1-\phi} \frac{C_s^0}{C_1^0}}{s + \beta m_{1,s}} \right] + \left[s \left(1 + \frac{\beta \frac{\phi}{1-\phi}}{s + \beta m_{1,s}} \right) \right] \overline{U}_1 \tag{65}$$

Setting the first term in square brackets equal to γ and the second term in square brackets equal to δ, this equation is written very simply as

$$\nabla^2 \overline{U}_1 = \gamma + \delta \overline{U}_1 \tag{66}$$

The boundary conditions are the same as those for the reservoir solution problem described in Section 2.2.2:

$$U_1(0,t) < \infty \tag{22}$$

$$U_1\left(\frac{a}{b},\theta\right) = m_{1,2}U_2\left(\frac{a}{b},\theta\right) \tag{23}$$

$$D_1\nabla U_1\left(\frac{a}{b},\theta\right) = D_2\nabla U_2\left(\frac{a}{b},\theta\right) \tag{24}$$

$$\frac{V_w D_1}{m_{2,w}D_2\alpha_2}\frac{\partial U_2}{\partial\theta}(1,\theta) = -\nabla U_2(1,\theta) \tag{25}$$

And, transformed into the Laplace domain, these boundary conditions become:

$$\overline{U}_1(0;s) < \infty \tag{30}$$

$$\overline{U}_1\left(\frac{a}{b};s\right) = m_{1,2}\overline{U}_2\left(\frac{a}{b};s\right) \tag{31}$$

$$D_1\nabla\overline{U}_1\left(\frac{a}{b};s\right) = D_2\nabla\overline{U}_2\left(\frac{a}{b};s\right) \tag{32}$$

$$\frac{V_w D_1}{m_{2,w}D_2\alpha_2}s\overline{U}_2(1,\theta) = -\nabla\overline{U}_2(1,\theta) \tag{33}$$

2.5 Solution (Dispersed Phase in Reservoir)

2.5.1 Flat Membrane (Cartesian Coordinates)

The solutions to the Laplace domain differential equations in Cartesian coordinates for regions 1 and 2, expressed by equations (66) and (21) are

$$\overline{U}_1 = A_{15}\exp\left[\xi\sqrt{\delta}\right] + A_{16}\exp\left[-\xi\sqrt{\delta}\right] - \frac{\gamma}{\delta} \tag{67}$$

$$\overline{U}_2 = A_{17}\exp\left[\xi\sqrt{\frac{D_1}{D_2}s}\right] + A_{18}\exp\left[-\xi\sqrt{\frac{D_1}{D_2}s}\right] \tag{68}$$

The boundary conditions are now written in terms of the solutions provided by equations (17) and (64) to yield algebraic equations which can be solved simultanously using Mathematica® to obtain expressions for A_{15}, A_{16}, A_{17}, and A_{18}. These expressions are listed in Appendix 2. The FORTRAN code is listed in Appendix 2. These expressions are then substituted into the solutions for regions 1 and 2 in equations (17) and (64) which are inverted numerically as described previously. The bath concentration can be determined as a function of time using equation (40).

2 .5.2 Hollow Fiber (Cylindrical Coordinates)

The solutions to the Laplace domain governing equations in cylindrical coordinates for regions 1 and 2 expressed by equations (62) and (21) are

$$\overline{U}_1 = \frac{1}{s} + A_{19}I_0\left(\xi\sqrt{s}\right) + A_{20}K_0\left(\xi\sqrt{s}\right) - \frac{\gamma}{\delta} \tag{69}$$

$$\overline{U}_2 = A_{21}I_0\left(\xi\sqrt{\frac{D_1}{D_2}s}\right) + A_{22}K_0\left(\xi\sqrt{\frac{D_1}{D_2}s}\right) \tag{70}$$

The boundary condition at $\xi = 0$ can immediately be used to determine that A_{20} must be zero. The remaining three boundary conditions are now written in terms of the solutions to obtain three algebraic equations are solved simultaneously for A_{19}, A_{20}, A_{21}, and A_{22} using Mathematica®. These expressions are shown in Appendix 2. The solutions are then inverted as

described previously using DINLAP, using the FORTRAN code listed in Appendix 2. Equation (40) can be used to determine the surrounding bath concentration as a function of time.

CHAPTER 3

EXPERIMENTAL PROCEDURES

3.1 Solvents and Solutes

Benzoic acid (Fisher Scientific, Fair Lawn, NJ), nicotine (Aldrich, Milwaukee, WI), caffeine (Sigma , St. Louis, MO), and toluene (Fisher Scientific, Fair Lawn, NJ) were used as model agents; decanol, octanol (Sigma, St. Louis, MO), and mineral oil (E. R. Squibb, Princeton, NJ) were used as organic solvents; all chemicals were used as received.

Dipalmitoylphosphatidylcholine (DPPC) and the 7-nitrobenz-2-oxa-1,3-diazol-4-yl (NBD) headgroup-labeled dipalmitoylphosphatidylethanolamine (NBD-PE) were the lipids used for liposome preparation; these were obtained from Avanti Polar Lipids, Alabaster, AL.

The dipeptide Phenylalanine-Glycine (Phe-Gly) was used as received from Sigma (St. Louis, MO).

3.2 Membranes

The membranes used in controlled release studies were hydrophilic nylon fibers (ENKA America, Asheville, NC), silicone coated 240 μm I. D. hydrophobic polypropylene fibers (AMT Inc., Minnetonka, MN), Celgard® X-20 240 μm I. D. microporous hollow fibers, Celgard® 2400 hydrophobic

Table 1. Membrane Dimensions

Membrane	OD μm[a]	ID μm[a]	Membrane thickness μm	Coating thickness μm
Nylon 6 (hollow fiber)	1000	600	200	--
Nylon 6 with wax coating (hollow fiber)	1000 (without coating)	600	200 (without coating)	not measured
Silicone-coated (hollow fiber)	300	240	30	1-2[b]
Silicone-coated (hollow fiber) with additional wax coating	300	240	30	1-2[b]
Cuprophan (flat)	--	--	22.1	--
Celgard® 2400 (flat)	--	--	25	--
Polycarbonate (flat)	--	--	12	--

[a] Supplied by manufacturer

[b] Determined experimentally by Papadopoulos [20]

polypropylene flat films (Hoechst Celanese SPD, Charlotte, NC), Cuprophan150 PM (ENKA) and polycarbonate track etched screen membranes (PCTE) (Poretics, Livermore, CA). Fibers were also modified in our laboratory by adding a wax coating to the nylon and the silicone-coated hollow fibers. The membrane dimensions are given in Table 1, and membrane materials and properties are summarized in Table 2.

The tortuosity of the PCTE membranes was calculated directly from

Table 2. Membrane Materials

Membrane	Material	Pore Size μm	Porosity[a]	Tortousity	Hydrophobic / Hydrophilic
Nylon 6	Polyamide 6	0.2	0.75	1.0[c]	Hydrophilic
Nylon 6 with wax coating	Polyamide 6 / wax	0.2	0.75	1.0[c]	Hydrophilic/ Hydrophobic
Silicone-coated	Polypropylene / silicone	0.03	0.4	2.5[b]	Hydrophobic
Silicone coated with additional wax coating	Polypropylene / silicone / wax	0.03	0.4	2.5[b]	Hydrophobic
Celgard® 2400	Polypropylene	0.03	0.38	5.0[b]	Hydrophobic
Cuprophan 150 PM	Regenerated cellulose	--	.65[d]	2.77[e]	Hydrophilic
PCTE	Polycarbonate (track etched)	18	0.1	1.44[f]	Hydrophilic

[a] Supplied by manufacturer

[b] Prasad and Sirkar [21]

[c] Assumed to be 1.0 due to large pore size

[d] Prasad and Sirkar [22]

[e] Colton et al. [23]

[f] Calculated from manufacturer information

manufacturer information. Knowing that pores in the track etched membranes are straight and at an angle of 34° of the surface, the tortuosity can be calculated from the following relationship:

$$\tau = \left(\frac{l_{eff}}{l}\right)^2 = \frac{1}{(\sin\phi)^2} \tag{71}$$

where τ is the tortuosity, l_{eff} is the effective pore length, and ϕ is the angle of the pore relative to the surface.

3.3 Liposome Preparation

The procedure for liposome preparation is described by Moss and Bhattacharya [24], and was implemented at the chemistry laboratories of Rutgers University, New Brunswick, NJ. 1.4 ml of DPPC in chloroform was mixed with 1.71 ml of the headgroup-labeled NBD-PE stock solution (1 mg/ml) in chloroform in a flask. The molar ratio of DPPC to NBD-PE was 1:7. Chloroform was evaporated under nitrogen to make a film of lipid, which was then dried for 2 hours under a high vacuum pump to remove traces of chloroform. The film was resuspended in 8 ml of 10 mM HEPES buffer (pH 7.4, 10 mM NaCl) and vortexed repeatedly at 55 °C until a suspension was formed. The suspension was then passed through an extruder (Lipex Biomembranes, Vancouver, Canada) at 55 °C using nitrogen pressure (400 psi), ten times through a stack of 50 nm polycarbonate membranes (Nuclepore, Cambridge, MA) to form small unilamellar vesicles via extrusion technique (SUVETs). The SUVETs were found to have a diameter of 35 nm [25]. Finally the SUVET solution was diluted with an additional 2 ml of HEPES buffer so that the final concentrations of DPPC and NBD-PE were 1.4 mM and 0.2 mM respectively. This was used as a stock solution to make dilutions of the SUVET solution in HEPES buffer at concentrations ranging

from 187 μM DPPC and 26.7 μM NBD-PE to 560 μM DPPC and 80 μM NBD-PE.

3.4 Analysis

3 .4.1 Benzoic Acid, Caffeine, Nicotine and Phe-Gly

Aqueous phase benzoic acid concentration was analyzed by a Hewlett Packard High Performance Liquid Chromatograph (HPLC) model 1090, using a Hypersil ODS C-18 100mm x 3 mm reverse phase column (Chrompack, Raritan, NJ). A 40% acetonitrile -60% water (v/v) carrier at a flow rate of 0.4 ml/min and UV detector at 254 nm were used. The samples were injected automatically into a 5 μl or 10 μl sample loop. A calibration plot for benzoic acid on Hypersil ODS is shown in Figure 6.

Aqueous phase nicotine and caffeine concentrations were analyzed in the same HPLC with a Spherisorb 5 mm ODS 150mm x 4.6 mm column (Phenomenex, Torrence, CA) at a detector wavelength of 260 nm. A mobile phase of 20% 0.0055 M sodium heptane sulfonic acid in water and 80% 0.0055 M sodium heptane sulfonic acid in methanol was used at a flow rate of 0.7 ml/min. Benzoic acid and nicotine were separated using the same method, but with a wavelength of 230 nm for benzoic acid and 260 nm for nicotine. Calibration plots for caffeine, nicotine, and benzoic acid on Spherisorb are shown in Figure 7, Figure 8, and Figure 9 respectively.

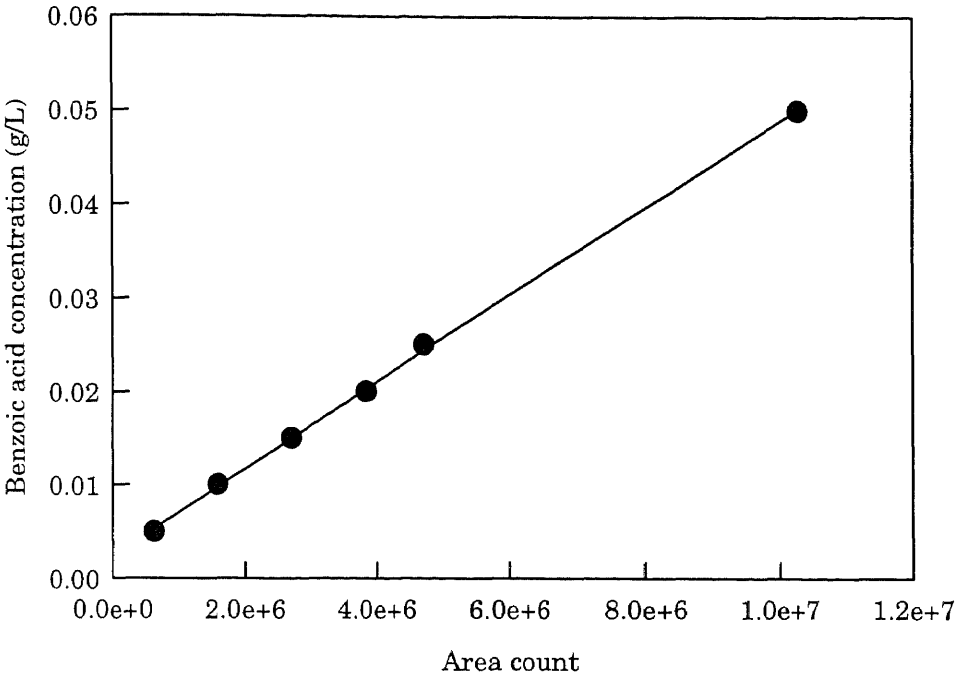

Figure 6. Calibration for benzoic acid on Hypersil ODS C-18 column

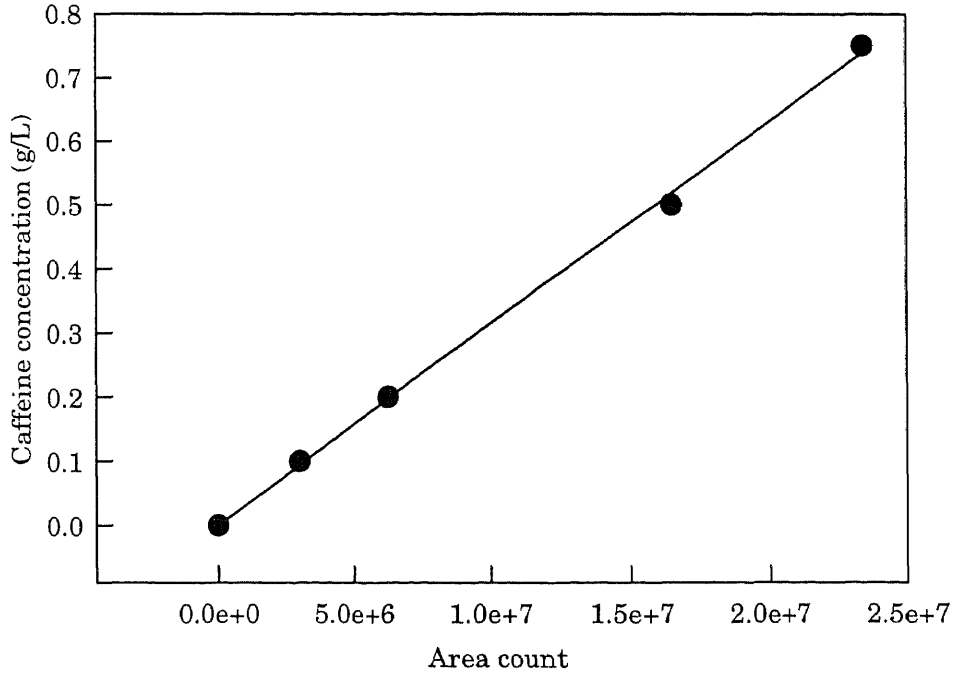

Figure 7. Calibration for caffeine on Spherisorb column.

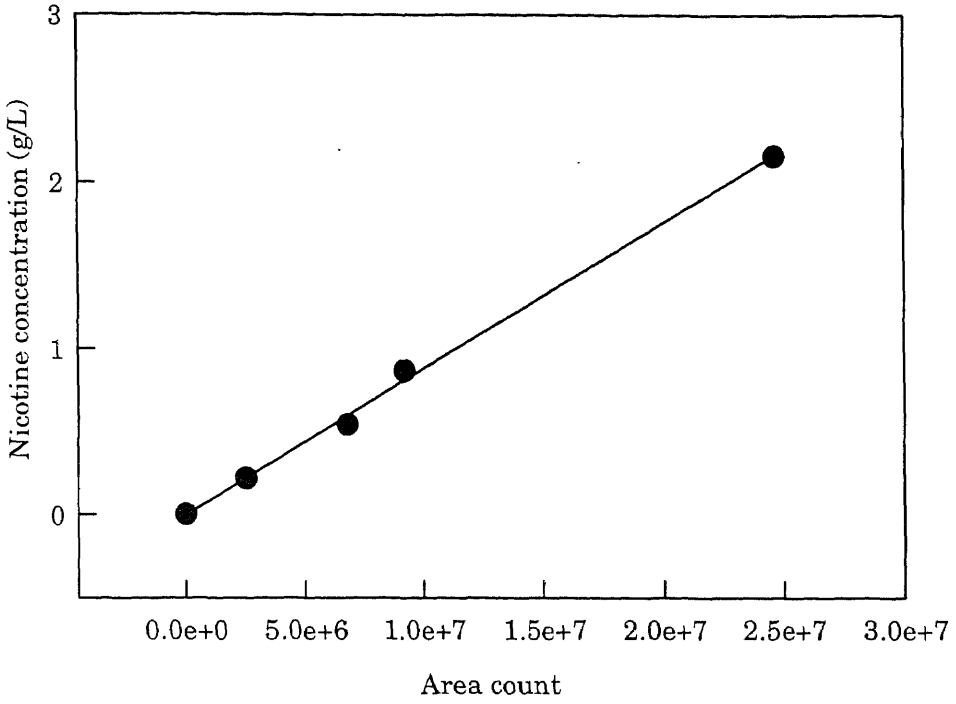

Figure 8. Calibration for nicotine on Spherisorb column

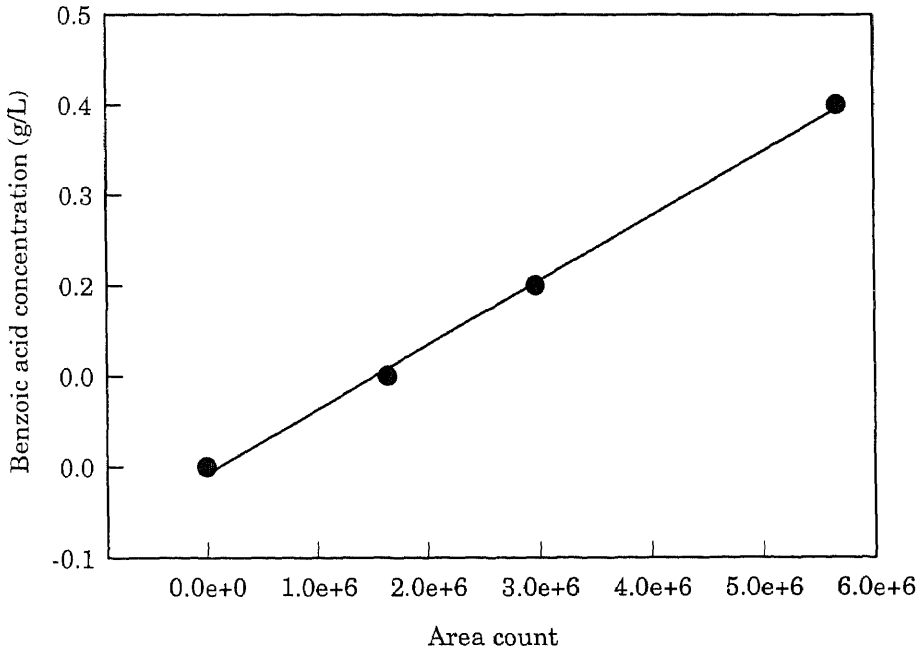

Figure 9. Calibration for benzoic acid on Spherisorb column

Phe-Gly concentration was analyzed by HPLC, using a Nucleosil 100-5 C18 column (Phenomenex) at a detector wavelength of 220 nm. An isocratic mobile phase of 75% potassium phosphate buffer (0.05M, pH adjusted to 2 with phosphoric acid) and 25% methanol was used at a flow rate of 0.5 ml/min. A calibration for Phe-Gly on the Nucleosil 100-5 C18 column is shown in Figure 10.

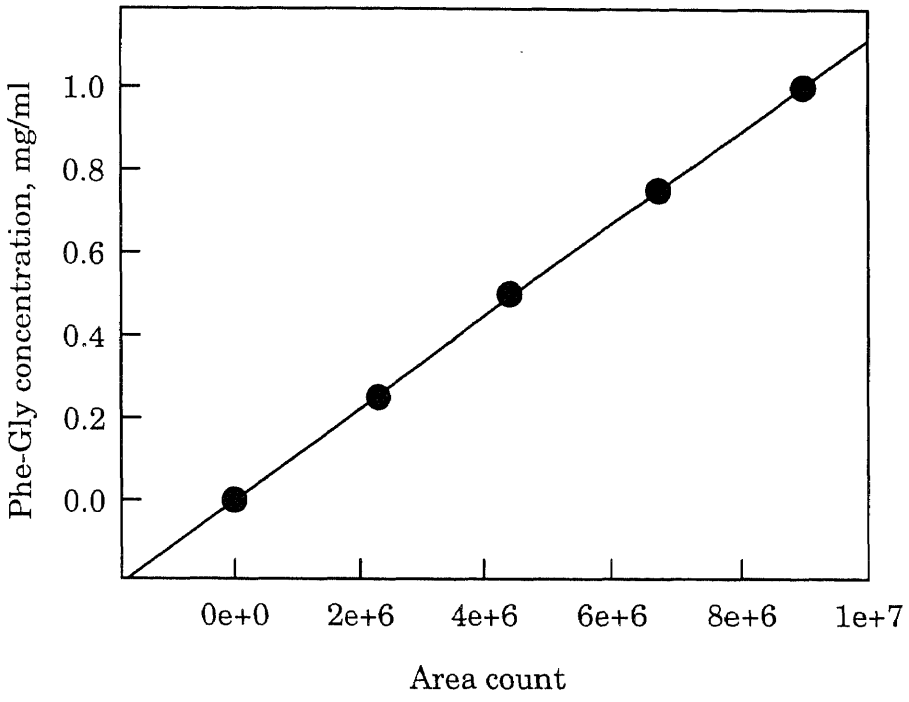

Figure 10. Calibration for Phe-Gly on Nucleosil 100-5 C18 column

Using to the method described by Shanbhag [26], toluene concentration was analyzed by HPLC using a Hypersil ODS 100 mm x 3 mm reverse phase column (Chrompack, Raritan, NJ) at a detector wavelength of 210 nm. A

mobile phase of 60% acetonitrile and 40% water was used at a flow rate of 0.5 ml/min. A calibration plot for toluene is shown in Figure 11.

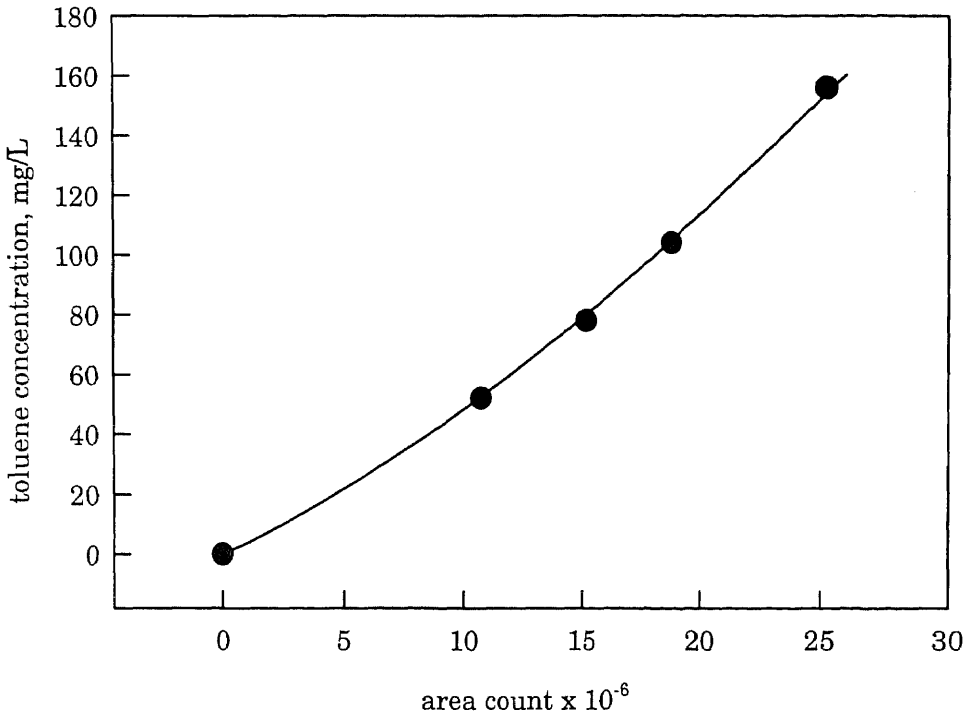

Figure 11. Calibration for toluene on Hypersil ODS C-18 column

3 .4.2 Liposome

Liposome concentration was analyzed by absorbance spectrophotometry for higher concentrations (in the range of about 0.1 mM DPPC and 14.3 μM NBD-PE to 0.56 mM DPPC and 80 mM NBD-PE) and by fluorescence spectrophotometry for lower concentrations of liposome (up to about 3.5 μM DPPC and 0.5 μM NBD-PE). Absorbance studies were performed at a wavelength of 465 nm on a Hitachi double beam spectrophotometer Model U-

2000 (Danbury, CT), and fluorescence studies were performed at an excitation wavelength of 469 nm and an emission wavelength of 540 nm on a Hitachi fluorescence spectrophotometer (Model F-3010). This spectrophotometer was available in the laboratories of HSMRC (NJIT, Newark, NJ). An absorbance spectrum for the SUVET solution of 7:1 DPPC:NBD-PE is shown

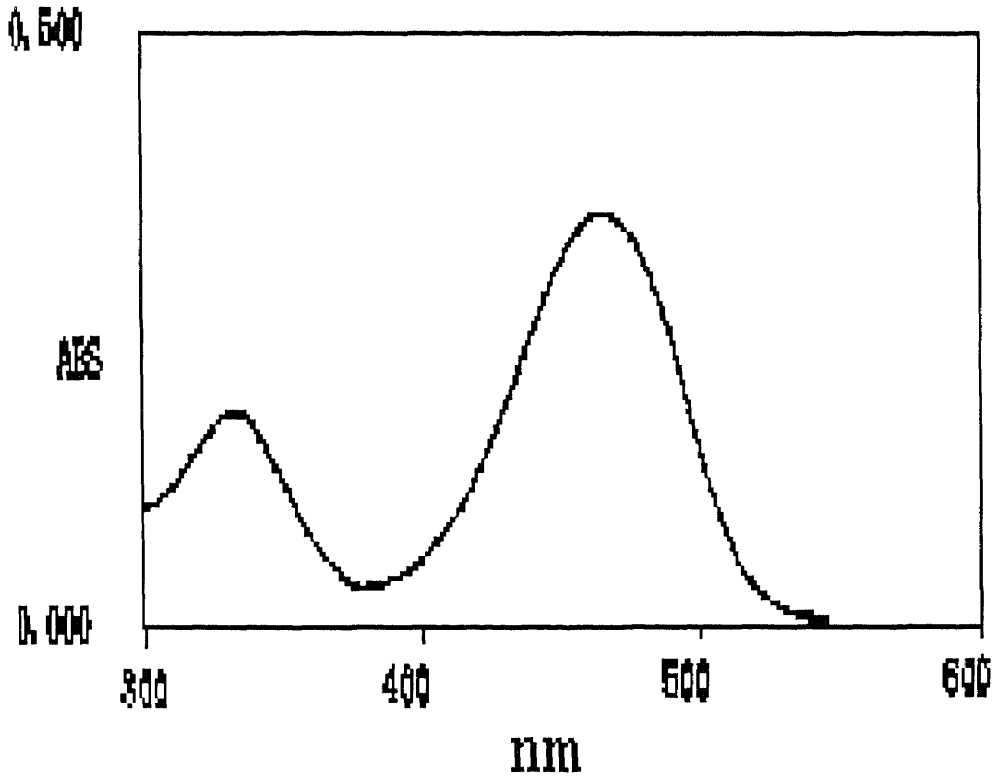

Figure 12. Absorbance spectrum of SUVETs

in Figure 12 (for 0.187 mM DPPC), and its fluorescence spectrum is shown in Figure 13 (for 0.56 μM DPPC). Calibration plots for absorbance and fluorescence analysis of the SUVETs are shown in Figure 14 and Figure 15.

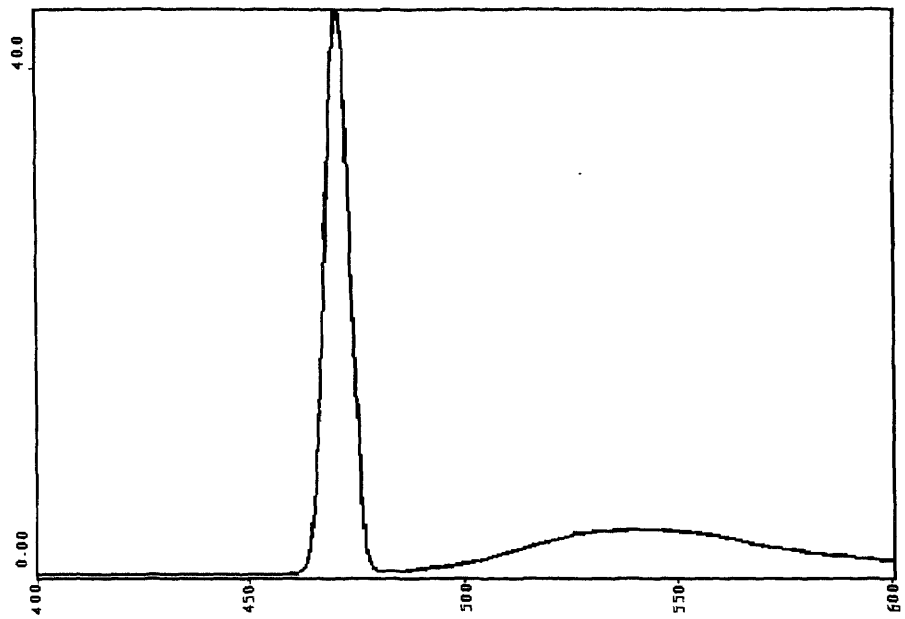

Figure 13. NBD fluorescence spectrum

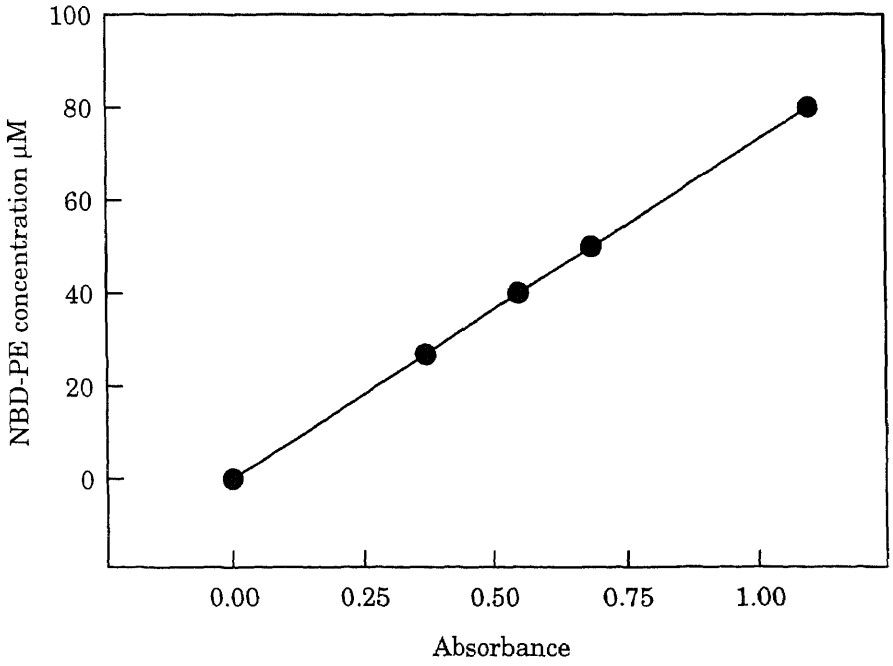

Figure 14. Calibration for SUVET using absorbance spectrophotometry.

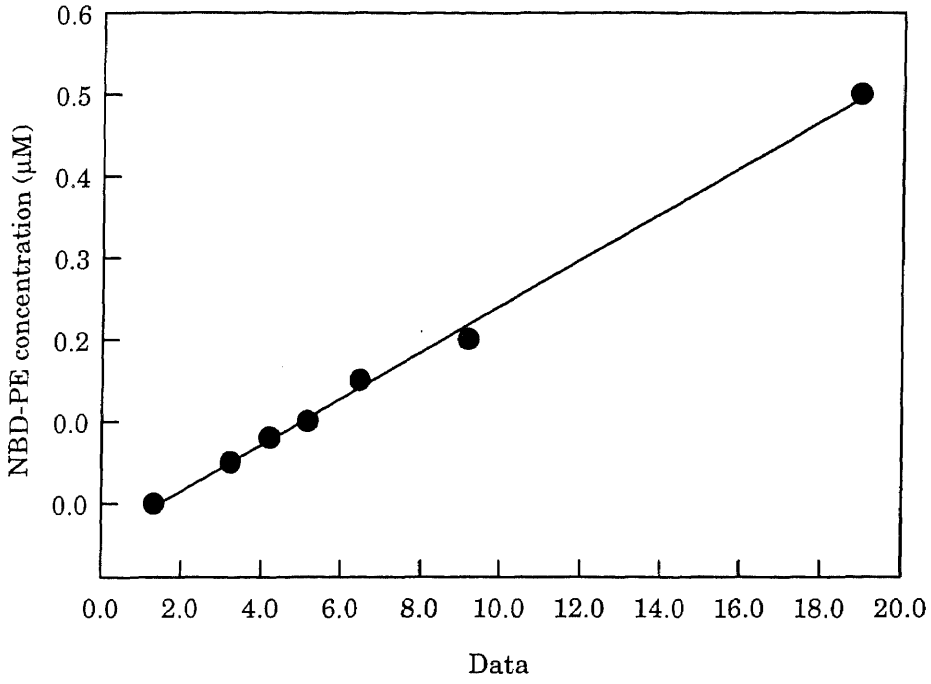

Figure 15. Calibration for SUVET using fluorescence spectrophotometry.

3.5 Determination of Distribution Coefficient

Distribution coefficients were determined experimentally as follows. An approximate volume of a benzoic acid solution of a known concentration in an organic solvent (octanol, decanol, or mineral oil) was weighed and contacted with a known volume of water and stirred for at least 24 hours. The organic phase was removed, and the aqueous phase was centrifuged as necessary to aid the removal of any remaining solvent. Aqueous samples were analyzed by HPLC as described in the preceding section, and organic concentrations were determined by material balance. For nicotine and caffeine, the

procedure described above for benzoic acid was followed, except that the solute was originally present in the aqueous phase at a known concentration.

3.6 Diffusion Coefficient Measurement

3.6.1 Benzoic Acid, Caffeine, and Nicotine

Diffusion coefficients were measured experimentally in the laboratory using a Crown Glass Two-reservoir Diffusion Cell System (Crown Glass, Somerville, NJ). The source solution containing the desired solute in aqueous solution was placed in one 3.0 ml reservoir; a receiving solution of pure water was placed in the other reservoir. A circular Celgard® 2400 film of 0.9 cm diameter (exposed) divided the reservoirs. 10 μL Samples were taken from the receiving side periodically and analyzed by HPLC by the methods described above, except that manual injection was used. The Two-reservoir Diffusion Cell System is shown in Figure 16. The slope of a plot of $\frac{1}{\beta}\log_e\left[\frac{C_s^i - C_r^i}{C_s(t) - C_r(t)}\right]$ vs. time yields the diffusion coefficient [27]. Here, β is the cell calibration constant, defined as $\frac{Am\varepsilon}{l\tau}\left(\frac{1}{V_{source}} + \frac{1}{V_{receiver}}\right)$, C_s and C_r are the respective source and receiver concentrations, m is the partition coefficient of the solute between the solvent in the pore and water, and ε, τ and l represent membrane porosity, tortuosity, and thickness respectively.

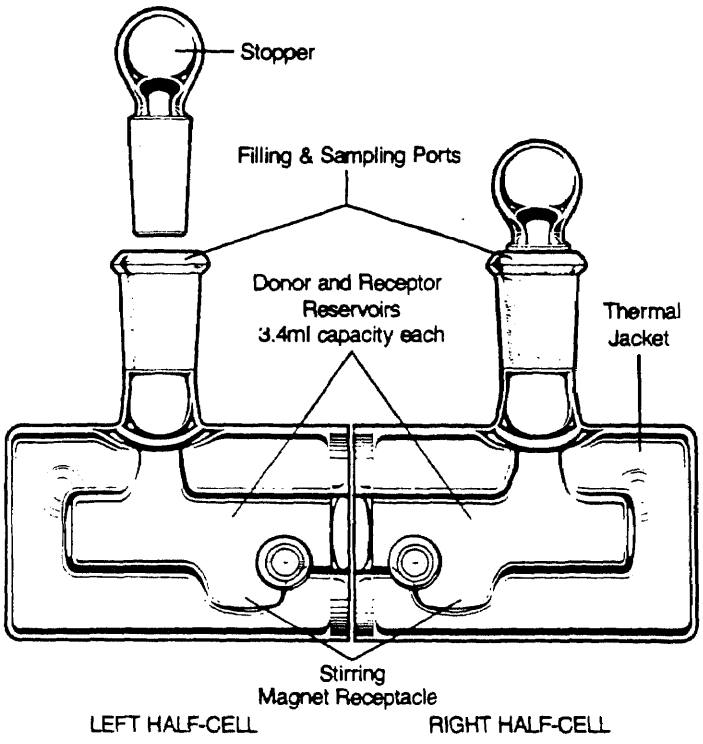

Figure 16. The Two-Reservoir Diffusion Cell System

3 .6.2 Liposome

Liposome diffusion coefficients were determined experimentally by a similar procedure: 3 ml of source liposome solution (530 μM DPPC and 76 μM NBD-PE in HEPES buffer) and 3 ml of HEPES buffer as receiver were used. The liposome diffused through an 18 μm diameter pore size polycarbonate membrane (Poretics, Livermore, CA), the pores of which were filled with water, to the receiver side. Samples were taken from both the source and receiver sides, analyzed by absorbance spectrophotometry as described above, and immediately returned to the diffusion cell.

3.7 Measurement of Solids Density in Solvent

Modeling of experiments using a dispersion of a solid agent in the reservoir requires knowledge of the density of the solid dispersed phase in the surrounding solvent phase. This density is the ratio of the mass of the dispersed solid to the volume occupied by that solid.

A known mass of the solid agent (benzoic acid or caffeine) was placed in a measured volume of a saturated solution of the agent in the solvent in a 10 ml graduated cylinder. The total volume after addition of the solid agent was recorded, and the displaced volume represents the volume occupied by the solid.

The density was then simply calculated from the ratio of the mass of the solid to the displaced volume.

3.8 Membrane Preparation

Fibers were cut into lengths of between 15 and 30 cm. One end of a fiber was inserted into an appropriate size B-D® Precision Glide^{T} needle (Becton Dickinson, Rutherford, NJ) and the union of the fiber and needle was then affixed to a glass microscope slide using epoxy, and allowed to cure for at least 24 hours. Celgard® 2400 films were cut to size (circles of about 3 cm diameter, needed for the controlled release cell used) before wetting.

Hydrophobic flat membranes and hollow fibers were wetted according to the procedures described in Bhave and Sirkar [28] and Bhave and Sirkar [29] respectively. Hollow fiber membranes were wetted with water after

being connected to needles and fixed on a glass slide. Hydrophobic membranes were initially wetted with 80% (v/v) ethanol in sterile deionized water in a Pyrex vessel. The ethanol was periodically exchanged with sterile deionized water over a period of at least 36 hours, until the solution was virtually ethanol free. Hydrophilic fibers were wetted with water without exchange by immersing them in a sterile deionized water bath for about 3 minutes while flowing water through the lumen. Prior to use, Cuprophan membranes were soaked in isopropyl alcohol for 40 hours to remove isopropyl myristate which is present in the pores and the bore of the membrane as supplied; after this step the membranes were exposed to water.

The organic solvent was introduced into the pores of a hollow fiber by injection using a syringe. Approximately 0.5 ml to 1 ml of solvent was passed through the lumen (being collected at the exit and discarded) to ensure that the organic had completely filled the pores. Water was injected through the lumen of the fiber to displace the organic solvent remaining in the lumen, and the fiber was blotted with a Kimwipe to reduce the amount of organic solvent present on the surface. Flat membranes, cut to size, were wetted with a few drops of organic solvent and blotted with a Kimwipe. Wax coatings were applied to Nylon and silicone-coated hollow fibers in the laboratory. The wetted fiber was immersed in melted wax. Both ends were kept above the surface to prevent wax from entering the lumen. No attempt was made to regulate the coating thickness.

Although water does not naturally wet the hydrophobic membranes, after the exchange process described above, water would remain intact in the pores throughout the experiment since the experiment was performed in a water bath. Since an organic solvent naturally wets a hydrophobic or hydrophilic membrane, the solvent would remain intact in the pores throughout the experiment.

3.9 Introduction of Agent into Reservoir

An aqueous or organic solution of agent was prepared and introduced into the bore of a hollow fiber by injection using a syringe. When introducing an organic solution to the hollow fiber membrane bore, the length of the hollow fiber was kept under a water bath. The excess organic solution pushed through the lumen formed droplets on exit and rose to the surface of the water bath, thus preventing the organic solvent from contacting the outside of the hydrophobic membrane. After the agent was introduced into the fiber lumen, a segment of length between 5 and 8 cm was cut from the end, and the ends were sealed with melted wax. Several segments were used so that the total fiber length was between 10 and 30 cm (the approximate total length was chosen so that the amount of agent present would permit aqueous samples to be analyzed directly).

Surface tension properties affecting pore filling are discussed by Kim and Hariott [30]. Since pore sizes are small and the reservoir and pore phases are chosen to form an immiscible aqueous-organic system, the

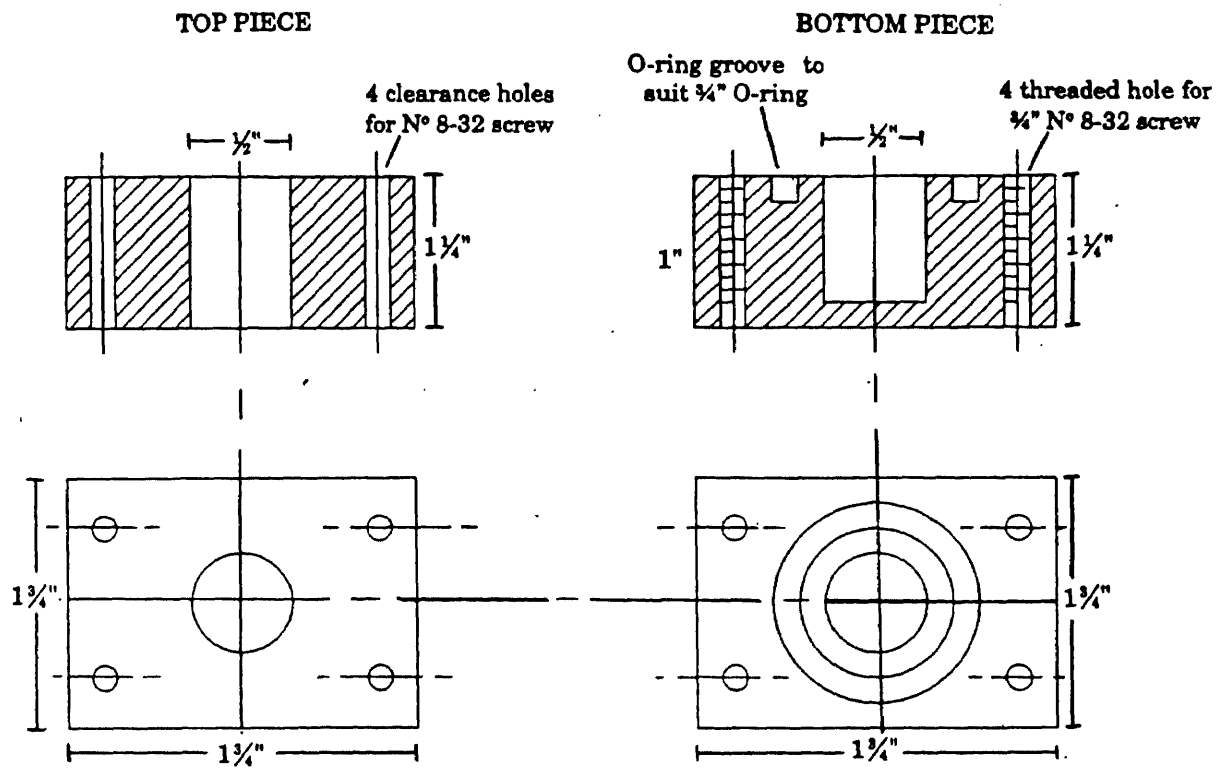

Figure 17. The single reservoir controlled release cell for flat membrane studies

interfacial tensions of which are large, the breakthrough pressures are large. The lumen was filled slowly, so the pressure drop was low. These factors ensured that the reservoir (lumen) phase did not break through the pore phase during the filling of the reservoir.

Two flat membrane cells were used in the studies: one with a reservoir volume of about 0.22 ml; and the other with two reservoirs, each having a volume of about 0.9 ml. The single reservoir controlled release cell is shown in Figure 17, and the two reservoir controlled release cell is shown in Figure 18.

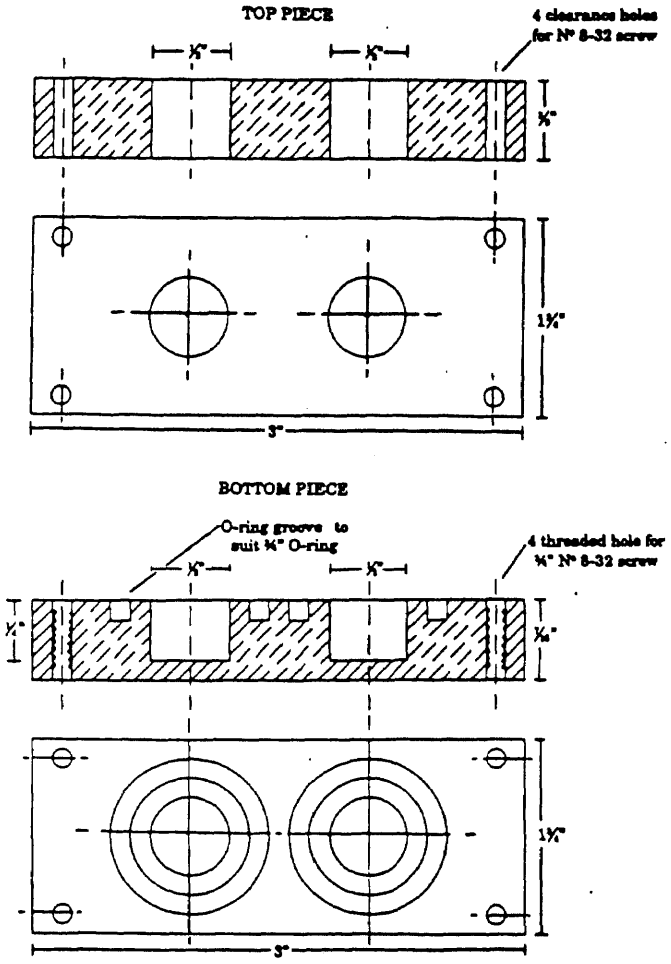

Figure 18. The two reservoir controlled release cell used for flat membrane studies

For the flat membrane system, a 250 μL syringe was used to load the reservoir with agent in solution; the wetted membrane was placed over the reservoir, and the cell was then closed.

A similar procedure was followed for the agent in suspension. The suspension of agent was well mixed and injected directly into the bore of the hollow fiber, as described above. In flat membrane studies, first a measured

amount of pure agent in solid form was placed in the reservoir, and then either pure solvent or a solution of the agent was introduced.

For the Phe-Gly studies, a measured amount of Phe-Gly was introduced into the reservoir before introducing an aqueous buffer to form a suspension. Studies were performed at various pH values: pK_a, pI and pK_b. (3.13, 5.38 and 7.62 respectively). The buffer was prepared by adjusting the pH of 0.067 M potassium phosphate with either phosphoric acid or sodium hydroxide as necessary.

310 In Vitro Release Studies

After preparation of the hollow fiber or flat membrane system, the device was submerged in water for in vitro release study. Hollow fibers were immersed in 100 ml sterile deionized water contained in a covered cylindrical glass vessel (Corning Glass, Corning, NY); the total fiber length was chosen to provide sufficient agent to achieve measurable and reproducible aqueous phase concentrations in the vessel during the experiment. The flat membrane device was immersed in a volume of water chosen for analytical convenience, between 150 - 750 cm^3. For Phe-Gly studies, the surrounding aqueous phase was a buffered potassium phosphate solution adjusted to the appropriate pH (the same pH as the reservoir solution), as described above. 200 µl samples were withdrawn periodically from the aqueous phase in the glass vessel until agent concentration remained constant for three successive

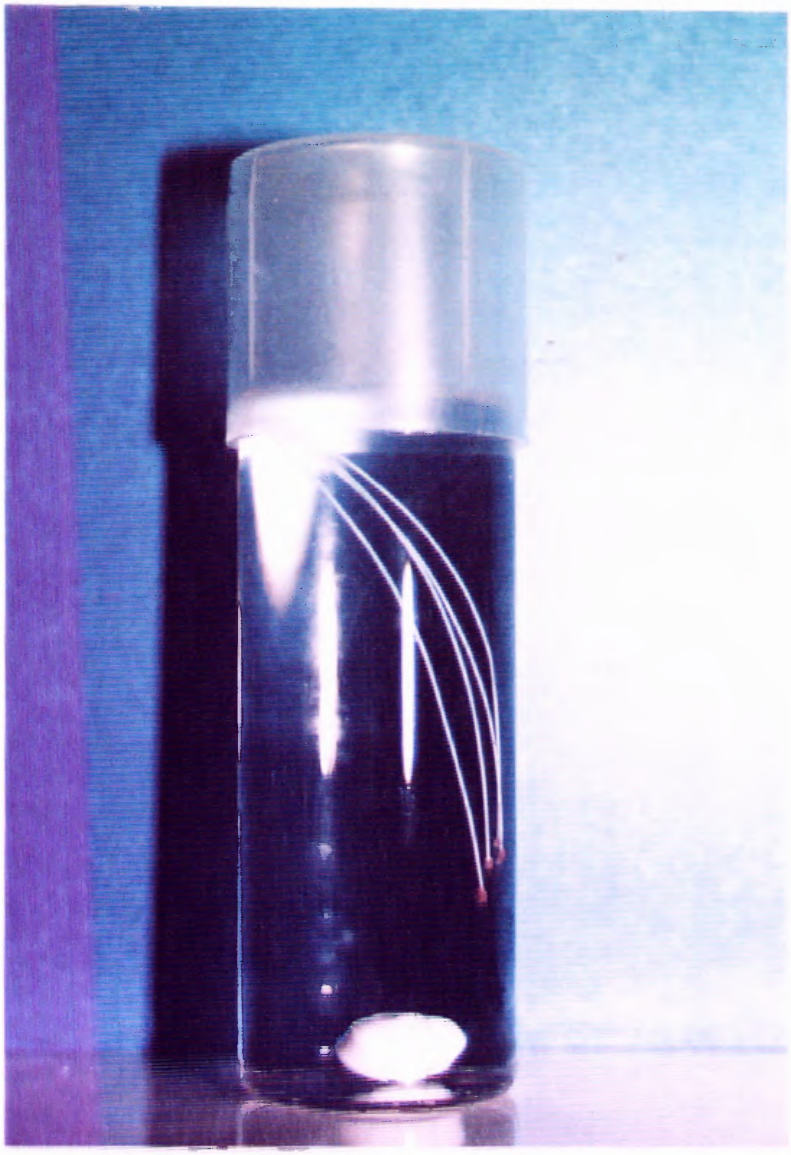

Figure 19. An in vitro release experiment using hollow fibers

measurements. The sample volume was negligible compared with the total aqueous volume, and agent concentration in the surrounding water was extremely low relative to its saturation concentration, therefore medium changes were not necessary. The mass of agent released was calculated from the aqueous concentration and bath volume, and plotted as a function of time to establish release profiles. A sample calculation is shown in Appendix 3.

Figure 20. An in vitro release experiment using a flat membrane

An in vitro experiment using hollow fibers is shown in Figure 19, and an in vitro experiment using a flat membrane system is shown in Figure 20. Most experiments were performed in duplicate, and it was established that they were reproducible.

Liposome release studies were performed using a PCTE membrane, and a continuous aqueous phase in the reservoir, pores, and surrounding bath. A 0.9 ml reservoir was filled with 560 μM DPPC and 80 μM NBD-PE in HEPES buffer-NaCl. The reservoir was bounded by a PCTE membrane wetted with HEPES buffer-NaCl. The surrounding water bath was 150 ml HEPES buffer-NaCl. 2 ml samples were withdrawn for analysis, analyzed immediately by fluorescence spectrophotometry, and replaced.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Experimental Results

4 .1.1 Distribution Coefficients

Distribution coefficients were measured for various agents in aqueous - organic systems used in these controlled release studies. Measured values of distribution coefficients are presented in Table 3. In accordance with the nomenclature set forth in Chapter 2, the aqueous-organic partition coefficients $m_{org,aq}$ tabulated are defined as the ratio of concentration of the

Table 3. Distribution Coefficients for Agents in Aqueous-Organic Systems

Solute	Solvent	Distribution coefficient $m_{org,aq} = \frac{C_{org}}{C_{aq}}$
Benzoic acid	Decanol	68
Benzoic acid	Octanol	88
Benzoic acid	Mineral oil	1
Nicotine	Octanol	1.6
Nicotine	Mineral oil	0.83
Caffeine	Octanol	0.77
Toluene	Toluene	1851

Table 4. Distribution Coefficients for Phe-Gly Between Water and Octanol at Various pH Values

pH	Distribution coefficient $m_{org,aq} = \frac{C_{org}}{C_{aq}}$
pK_a (3.13)	0.11
Isoelectric point (5.4)	0.093
pK_b (7.62)	0.096

agent in the organic phase to that in the aqueous phase at equilibrium. Distribution coefficients of amino acids and peptides between water and an organic solvent are known to be dependent on the pH of the aqueous phase. The distribution coefficients for octanol - water systems were measured at three values of pH: pK_a, pK_b, and the isoelectric point. The values were measured using equal mass of aqueous and organic phase, and an initial Phe-Gly concentration of about 1 mg/ml in the aqueous phase. These values of distribution coefficient are presented in Table 4.

4 .1.2 Diffusion Coefficients

Binary diffusion coefficients for agents in organic solvents (octanol, decanol, and mineral oil) were measured experimentally according to the procedure described in Section 3.6. The following binary diffusion coefficients were

derived from the slope of a plot of $\frac{1}{\beta}\log_e\left[\frac{C_s^i - C_r^i}{C_s(t) - C_r(t)}\right]$ vs. time: nicotine in mineral oil (Figure 21), nicotine in octanol (Figure 22), caffeine in octanol (Figure 23), and benzoic acid in mineral oil (Figure 24), . Diffusion coefficients for benzoic acid in octanol and decanol were calculated by the

Table 5. Calculated[W.-C.] and Experimentally Measured Binary Diffusion Coefficients of Agents in Organic Solvents

Agent	Solvent	Binary Diffusivity (cm^2/s)
nicotine	mineral oil	4.2×10^{-7}
nicotine	octanol	4.7×10^{-6}
caffeine	octanol	7.5×10^{-6}
benzoic acid	decanol	5.45×10^{-6} (W.-C)
benzoic acid	mineral oil	2.1×10^{-7}
benzoic acid	octanol	8.6×10^{-6} (W.-C)

Wilke-Chang (W.-C.) correlation in conjunction with the additive method of LeBas (in Reid et al., [31]). Values of calculated and experimentally measured binary diffusion coefficients are summarized in Table 5.

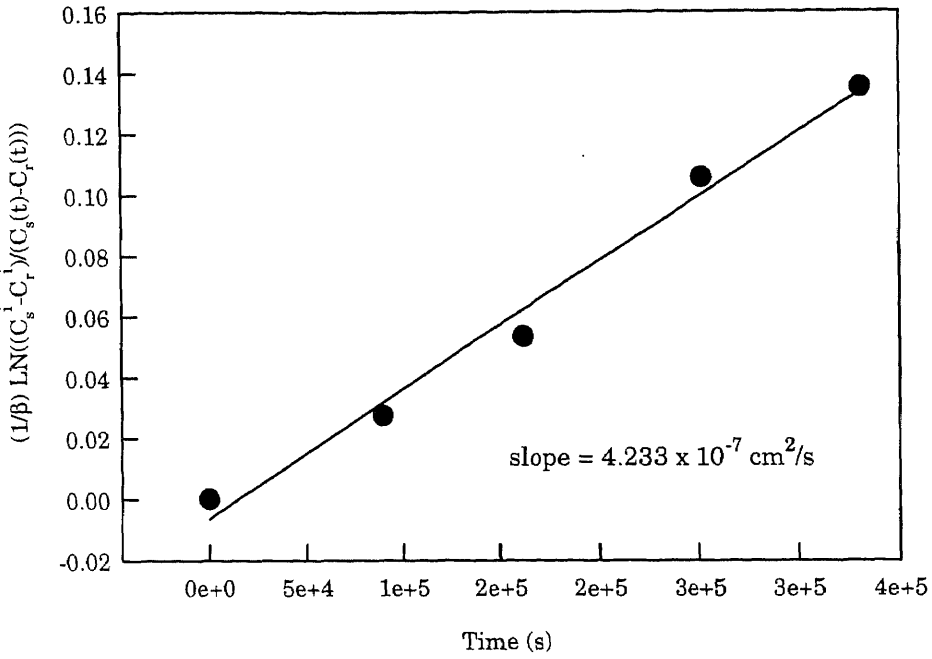

Figure 21. Diffusion coefficient of nicotine in mineral oil

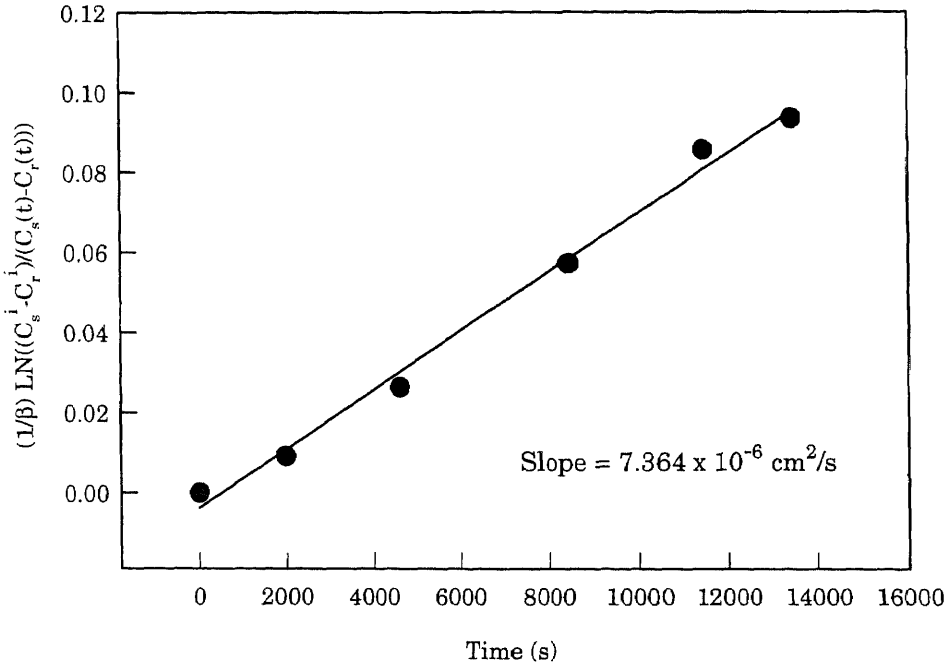

Figure 22. Diffusion coefficient of nicotine in octanol

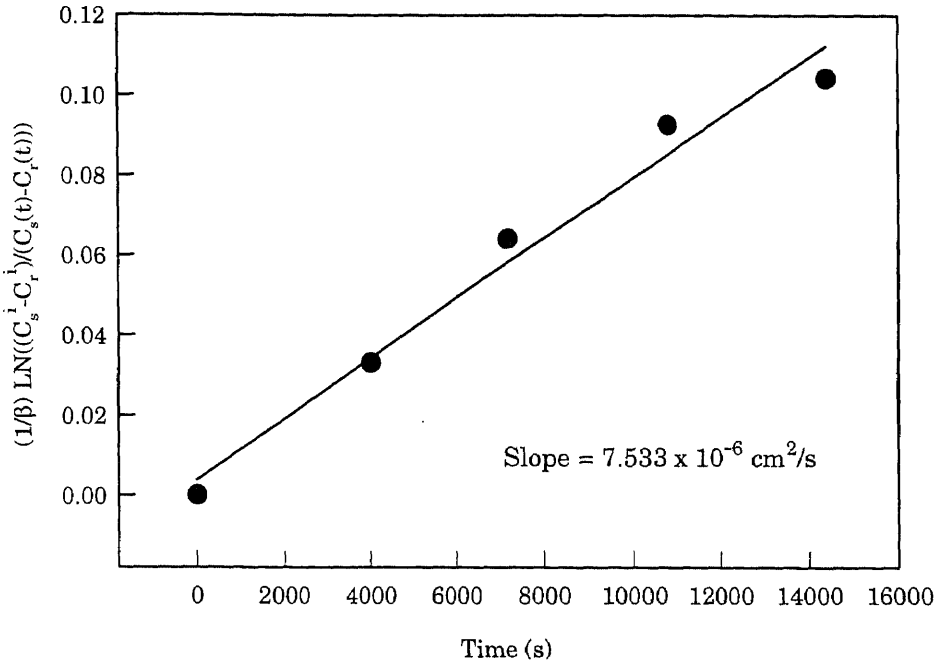

Figure 23. Diffusion coefficient of caffeine in octanol

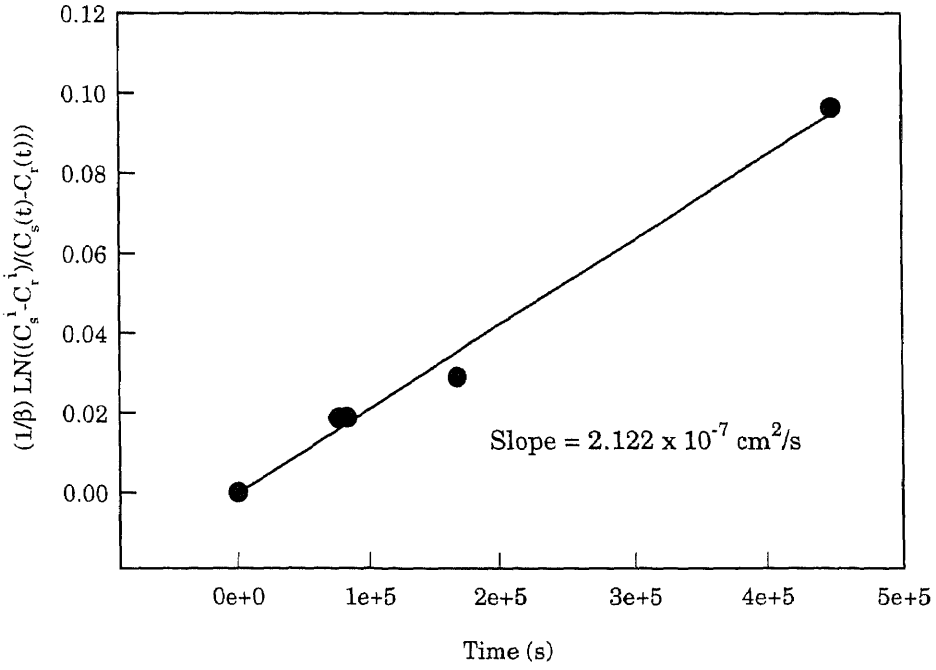

Figure 24. Diffusion of benzoic acid in mineral oil

The free diffusivity of 35 nm diameter DPPC in water was calculated by the Stokes-Einstien equation, and found to be 1.2×10^{-7} cm^2/s. The diffusivity of liposome through a 10 μm pore size polycarbonate membrane was measured experimentally by absorbance spectrophotometry, as mentioned in Chapter 3, and found to be 2.4×10^{-7} cm^2/s. Since the SUVET is a fairly spherical molecule, one would expect the measured diffusivity to be closer to the theoretical value derived from the Stokes-Einstein relation. A possible explanation for this discrepancy lies in the fact that the spectrophotometric absorbance of NBD is not environmentally sensitive, that is, free NBD absorbs just as well as NBD in the lipid bilayer. If small amounts of NBD "leak" out from the lipid bilayer into the surrounding free solution, they will diffuse faster through the membrane than the liposomes and will effect an apparent receiver side concentration of SUVET which is higher than the actual. The first four data points on the diffusion plot shown in Figure 25 lie nicely on a straight line and yield an experimental diffusion coefficient of 2.4×10^{-7} cm^2/s. The data points taken after the first 72 hours deviate from this line; the increase in the slope of the of the curve indicates that some NBD began to leak out of the lipid bilayer after about 72 hours. Fluorescence of NBD is environmentally sensitive (that is, NBD fluoresces only when embedded in the lipid bilayer) and was used to confirm that NBD is slowly lost from the lipid bilayer: fluorescence measurements of a sample were relatively constant for about 72 hours, and then began slowly to decrease.

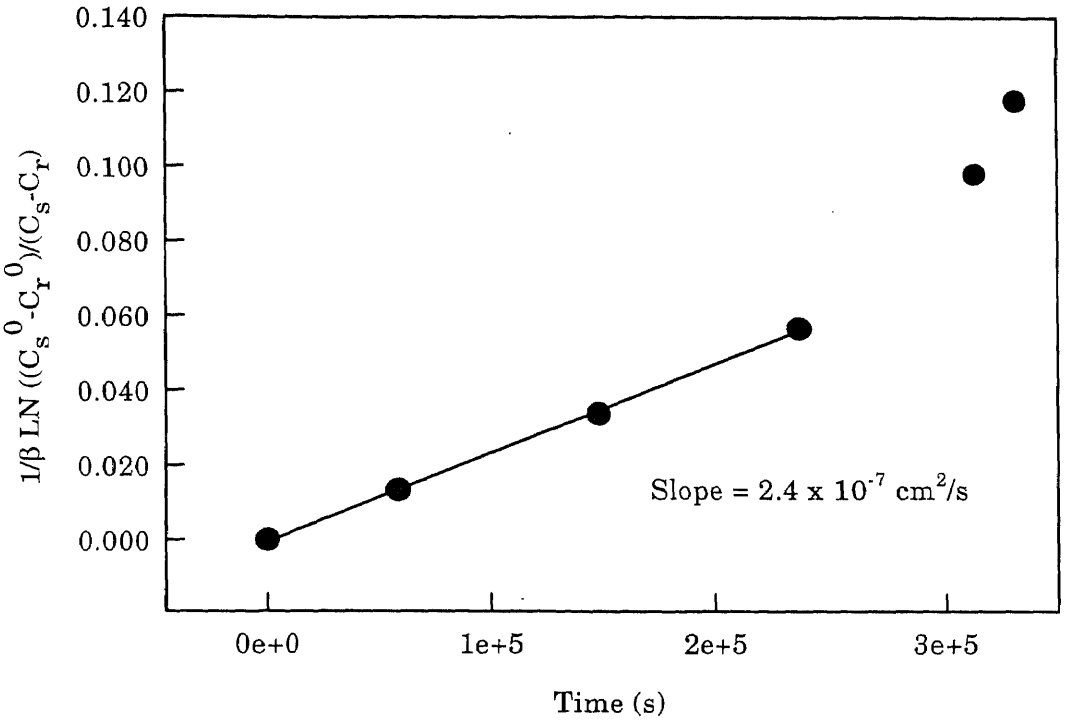

Figure 25. Diffusion coefficient of SUVET in water

Binary diffusion coefficients for nicotine in water and caffeine in water were calculated using the Wilke-Chang correlation in conjunction with the additive method of LeBas (in Reid et al., [31]). The diffusivity of benzoic acid in water was taken from Reid et al. [31]. Diffusivities of agents in water are summarized in Table 6.

Table 6. Diffusion Coefficients of Agents in Water

Agent	Diffusivity (cm^2/s)	Method
Benzoic acid	1.21×10^{-5}	Experiment [31]
Caffeine	6.82×10^{-6}	Wilke-Chang [31]
Nicotine	6.50×10^{-6}	Wilke-Chang [31]

4.1.3 Solid Density

Solid density, or the volume occupied by a given mass of solid agent suspended in a surrounding liquid, was measured for caffeine in water and benzoic acid in decanol. These values are tabulated in Table 7. It should be noted here that the solid was suspended in a *saturated* solution of agent in solvent for these measurements so that none of the measured solids would dissolve in the surrounding solution. The volume displaced by a given mass of solids is assumed to be independent of the liquid in which the solids are dispersed. Values for benzoic acid in octanol and in mineral oil are assumed to be equal to that for benzoic acid in decanol.

Table 7. Densities of Pure Powder Form Crystalline Agents

Agent	**Density** $\left(\frac{\text{mg}}{\text{ml displaced}}\right)$
Caffeine	1298
Benzoic acid	2093

4.1.4 Solution in Reservoir: In Vitro Experiments

Using an agent initially in solution in the reservoir, the loading of the agent in the device is limited by the agent's solubility in the reservoir solvent. If the agent has a high solubility in the reservoir solvent, it is possible to

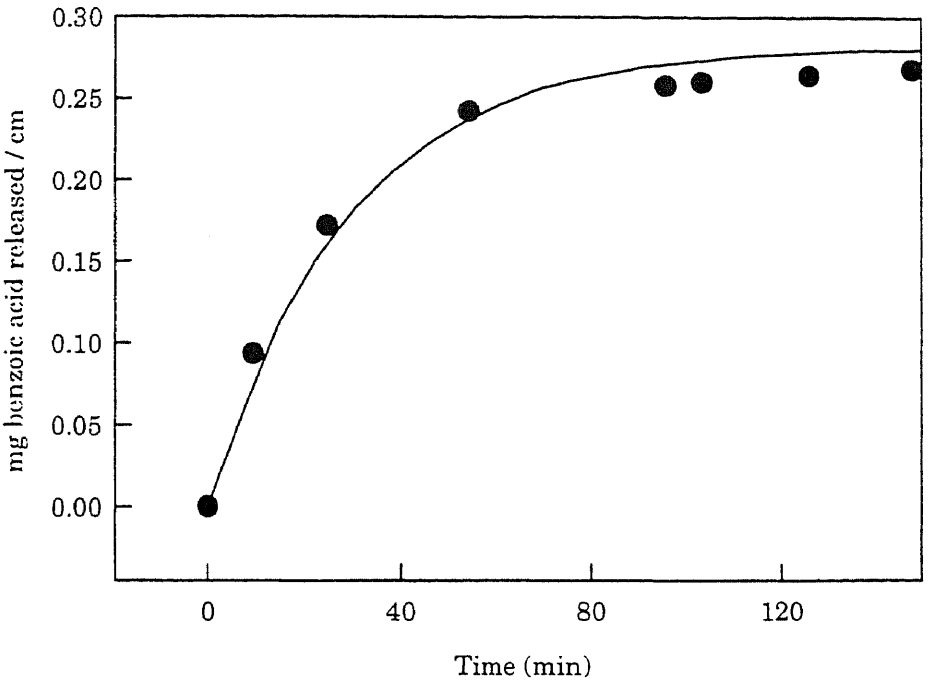

Figure 26. The release profile of benzoic acid from a nylon porous hollow fiber with water-filled pores. Benzoic acid was initially contained in a decanol solution in the reservoir (100 mg/ml). Prediction is shown as a solid curve.

achieve high loading in the device. If the partition coefficient of the agent between the reservoir and the pore is high, the driving force for diffusion across the pore is reduced. This has the effect of maintaining a high reservoir concentration (relative to the pore and bath concentrations) for an extended time. A system which effects a high reservoir concentration (and a low bath concentration) for an extended time will exhibit a relatively constant release rate for an extended time.

In this section results are presented for experiments using an agent in solution. Both organic reservoir/aqueous pore and aqueous reservoir/organic pore configurations are considered, and high and low partition coefficients are also investigated.

The solubility of benzoic acid in decanol is 45 times its solubility in water. Its partition coefficient between decanol and water is 68. Benzoic acid, therefore, is a good candidate to be used for studies of release from a device with an organic reservoir solution and water filled pores.

Figure 26 shows the release profile obtained using a Nylon porous hollow fiber device with water in the pores of the membrane and a reservoir containing benzoic acid in decanol solution. The rate of release is approximately constant for about 30 minutes. The model simulations describe the data quite well.

Figure 27 shows the release profile of benzoic acid from a Nylon porous hollow fiber device with water in the membrane pores. In this study, benzoic acid was initially contained in an octanol reservoir solution. The release rate is approximately constant for about 45 minutes. Comparing the release profiles shown in Figure 26 and Figure 27, the longer period of pseudo zero order release in the latter is due to the larger partition coefficient of benzoic acid between octanol and water.

Benzoic acid in octanol was also used as a reservoir solution in a flat membrane system. Using a Celgard® 2400 flat film, the release profile shown

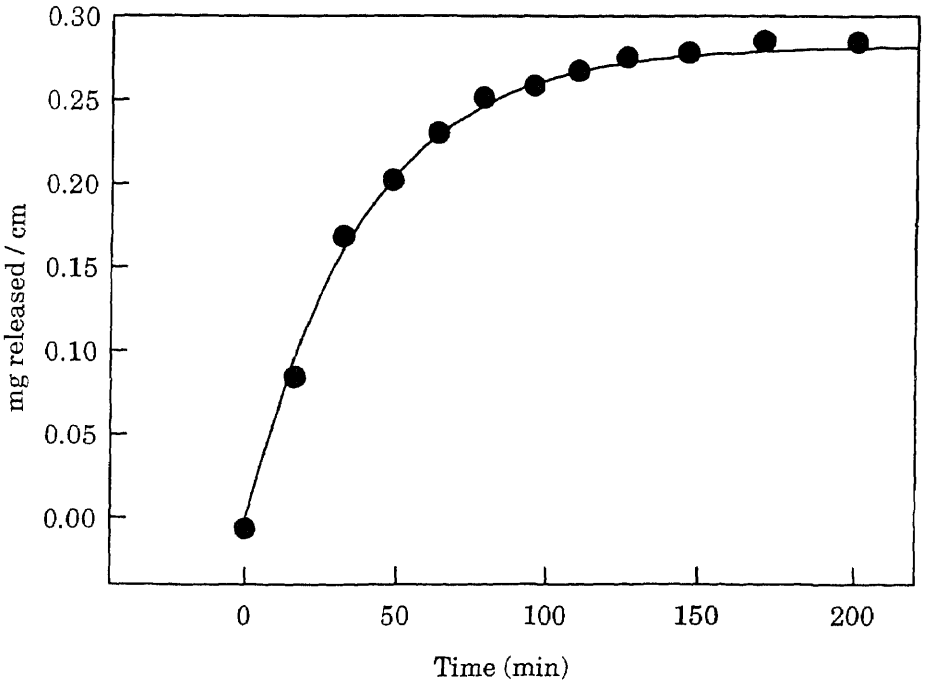

Figure 27. The release profile of benzoic acid from a nylon hollow fiber with water-filled pores. Benzoic acid was initially contained in an octanol solution in the reservoir (100 mg/ml). Prediction is shown as a solid curve.

in Figure 28 was obtained. The release rate was approximately constant for a period of 24 hours. When compared with the release profile of benzoic acid from the nylon hollow fiber shown in Figure 27, the duration of zero order

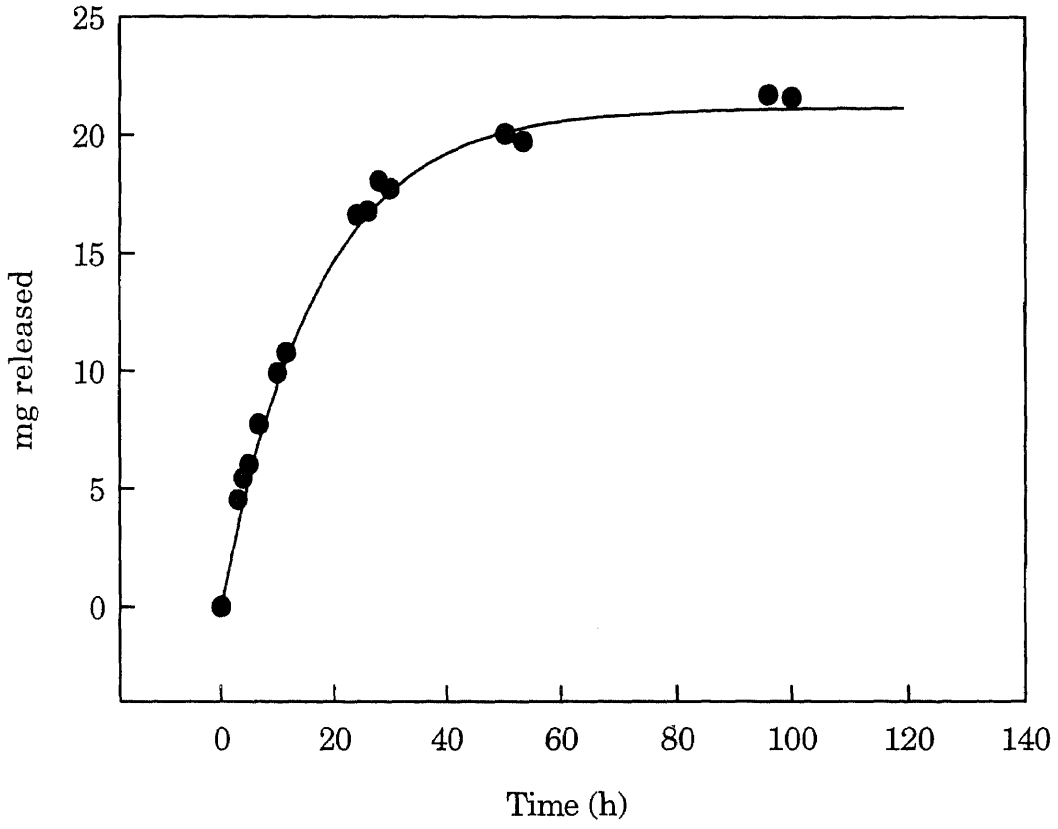

Figure 28. The release profile of benzoic acid from a flat membrane device using a Celgard® 2400 flat membrane with water-filled pores. Benzoic acid was initially contained in an octanol reservoir solution (100 mg/ml). Predicted results are shown by the solid curve.

release is longer for the flat membrane due, essentially, to a larger reservoir volume. The effects of reservoir and membrane resistance are examined in Section 4.2.

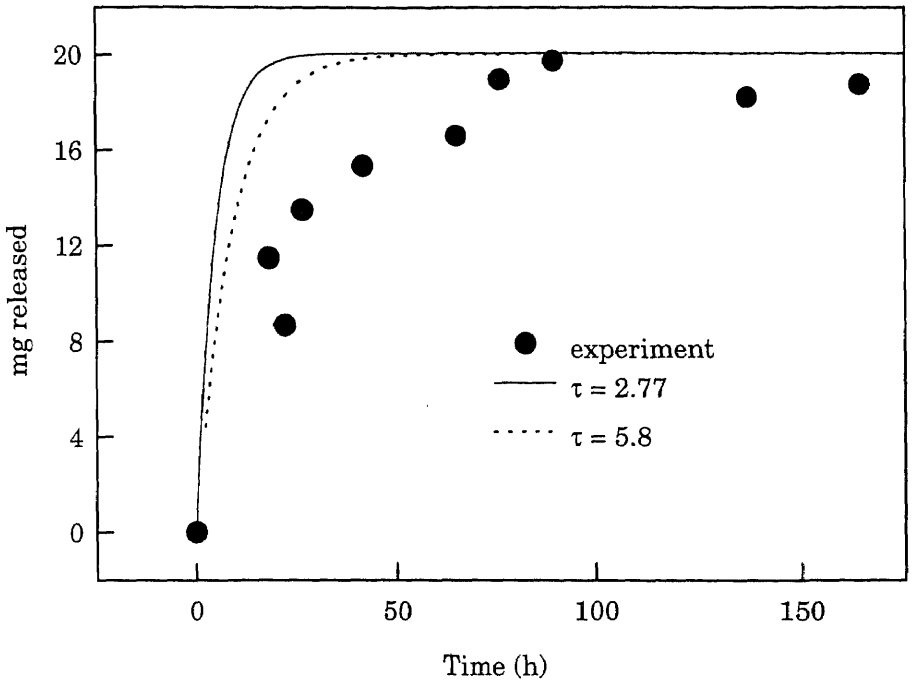

Figure 29. Controlled release of benzoic acid using a water-swollen Cuprophan 150 PM regenerated cellulose membrane. Reservoir initially contained 100 mg/ml benzoic acid in octanol. Curves show predictions for two values of tortuosity.

Figure 29 shows the release profile for benzoic acid from a flat membrane device using a Cuprophan 150 PM regenerated cellulose membrane. The solid curve ($\tau = 2.77$) shows the release profile obtained using the model.

Following is a possible explanation for the disagreement between the experimental and predicted results: When in contact with water, the Cuprophan membrane becomes a hydrogel. Colton et al. [23] have shown that the membrane permeability for a similar Cuprophan membrane decreases with increasing molecular weight and size. The membrane

tortuosity is extracted from experimental data for permeability as a function of solute radius, extrapolated to a molecular radius of zero; this value of tortuosity is 2.77. Hindrance due to increasing molecular radius results in a decrease in the membrane permeability, which corresponds to an increase in the "effective tortuosity". Prasad and Sirkar [21] find an "effective tortuosity" for phenol in the Cuprophan 150 PM membrane to be 5.8. By the arguments of Colton et al. [23] one would expect the Cuprophan membrane "effective tortuosity" values for phenol and benzoic acid to be close, since the molecules are similar in structure and molecular weight. The dotted line in Figure 29 shows the predicted profile using a value of 5.8 for the effective tortuosity, and this curve shows somewhat better agreement with the experimental data. An additional source for deviation would be differential swelling of the membrane on the organic side, leading to reduced swelling of the membrane and correspondingly, higher tortuosity.

Nicotine was used as a model agent to investigate the release of an agent with high water solubility from an aqueous reservoir through membrane pores filled with an organic solvent. Mineral oil was used as the organic liquid filling the membrane pores.

Results of a study of release of nicotine from nylon porous hollow fibers with mineral oil-filled pores are shown in Figure 30. In this system, the aqueous-organic partition coefficient for nicotine between water and mineral oil 1.2. This low value of the partition coefficient fails to provide the drastic

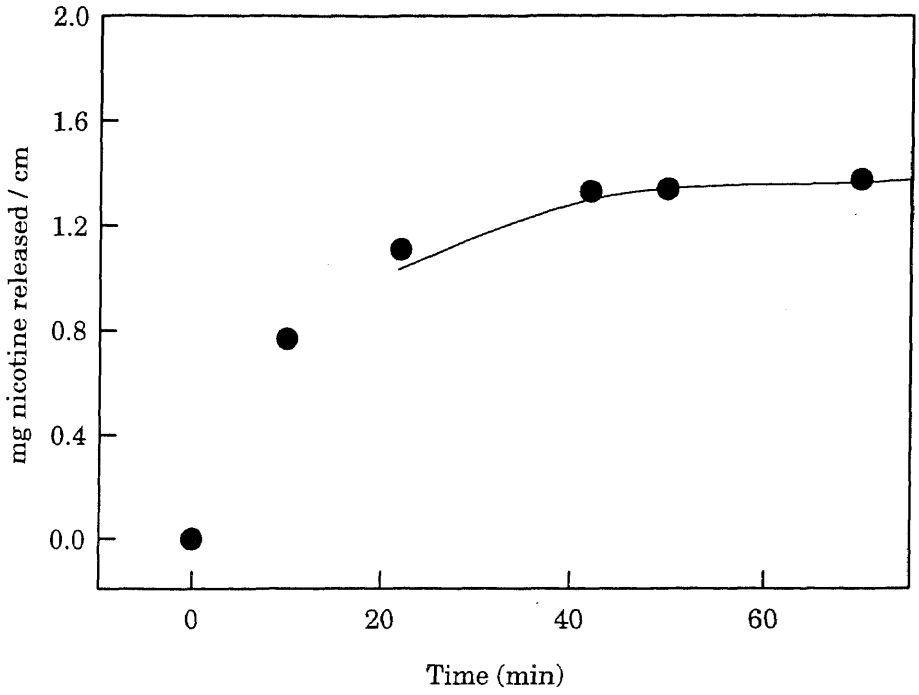

Figure 30. The release profile of nicotine from nylon hollow fibers with mineral oil-filled pores. Nicotine was originally contained in an aqueous reservoir (505 mg/ml). Predicted results are shown by the solid curve.

reduction in driving force across the membrane that was seen for the benzoic acid-octanol-water and the benzoic acid-decanol-water systems. However, the diffusivity of nicotine through mineral oil is small (4.2×10^{-7} cm^2/s), and this increases the resistance provided by the membrane, thereby slowing the release rate.

Nicotine release from a flat membrane device was also investigated. Using a Celgard® flat membrane with mineral oil-filled pores, nicotine release from two different cells was studied. The cells had different volumes and different mass transfer surface area to volume ratios. Figure 31 shows the release profile obtained using the 0.22 ml cell, and a constant release rate

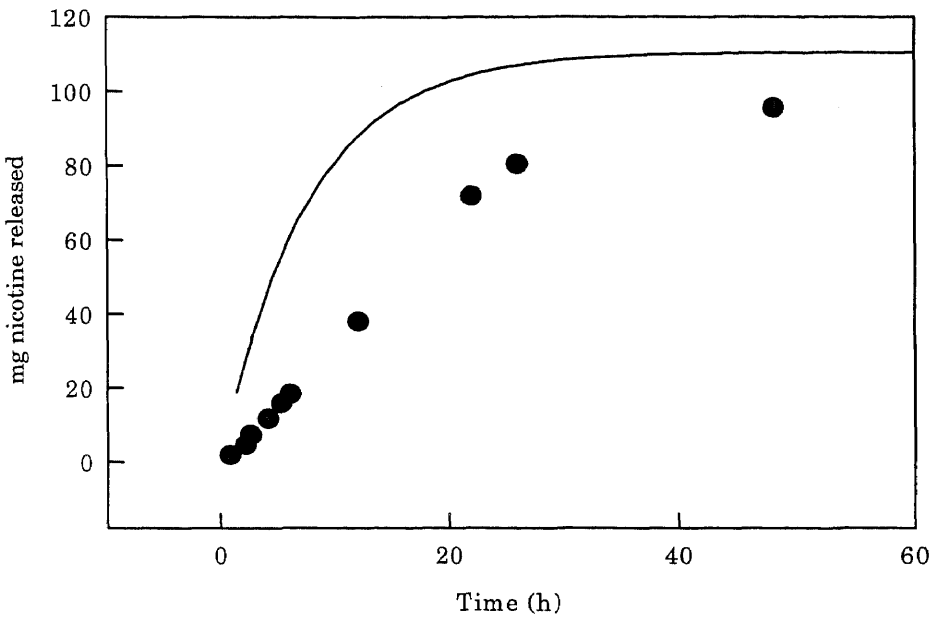

Figure 31. The release profile from a flat membrane device using a Celgard® 2400 flat membrane with mineral oil-filled pores. Nicotine was contained initially in a 0.22 ml aqueous reservoir (505 mg/ml). The predicted results are shown with the solid curve.

was obtained for a period of about 24 hours. Notice that the experimental data are very clearly zero order for times up to about 27 hours, but the predicted results do not reflect the constant release rate achieved experimentally. The discrepancy between predicted and observed release rates for nicotine is discussed at the end of this section.

The next nicotine release study was done using the same solvent / solute / microporous membrane system, but with the 0.95 ml reservoir cell; this release profile is shown in Figure 32. The data indicate a zero order release for at least 50 hours, but again the model does not so clearly predict this trend.

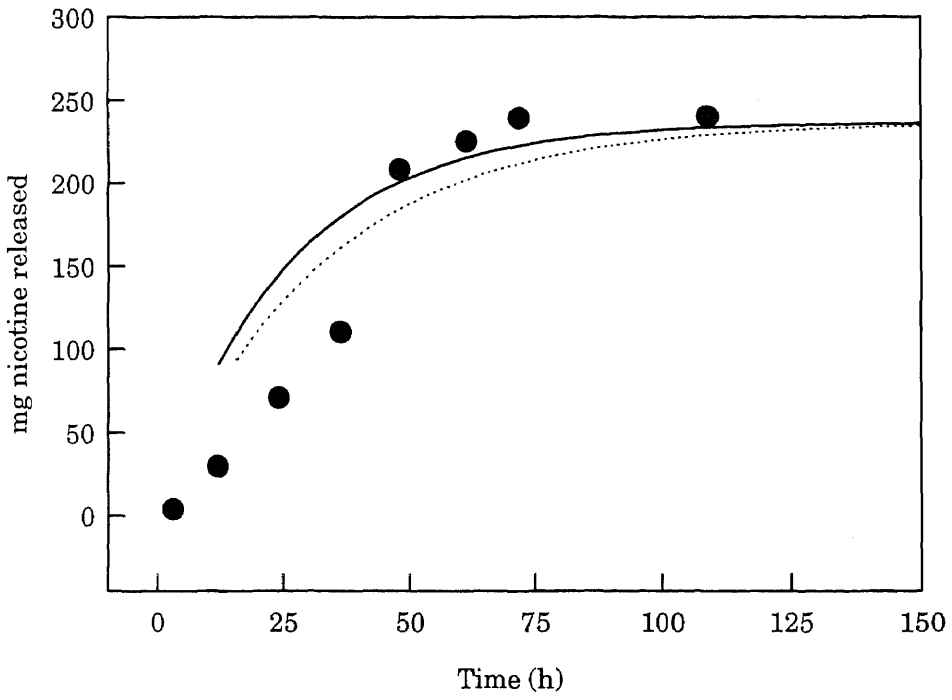

Figure 32. The release profile for nicotine from a flat membrane device using a Celgard® 2400 flat membrane with mineral oil-filled pores. Nicotine was contained initially in a 0.95 ml aqueous reservoir (505 mg/ml). The predicted results are shown with the solid curve. Dotted line shows the predicted curve using a 20% reduction in D_1 and D_2.

At this point it is appropriate to make a few points about the experimental data and the predicted results. The data for many experiments indicate a constant release rate for an extended time. The possibility exists that the data could fit a first order concentration dependence, which would be psuedo-zero order at short times. In order to determine whether the release rate is zero order, some statistical analysis must be performed. The procedure was to find the best zero order and first order fit for the data in a given time range, and then to perform a paired Student's t-test between

dependent values predicted by the best fit and those obtained experimentally. The t-test gives a measure of the probability (P) that one is incorrect in stating that the two means are different. Consider Figure 28 for the benzoic acid system and Figure 31 for the nicotine system. For Figure 28, the t-test performed on data for times between 0 and 30 hours gives a P value of 0.998 for the zero order case and 0.854 for the first order fit. For Figure 31 the t-test result for times up to 26 hours yielded P equal to 0.9993 for the zero order fit and 0.6595 for the first order fit. The results for both of these systems are more closely zero order than first order.

The second point worth mentioning here concerns the data generated by the computer simulation. In Figure 30, there are no predicted values for times less than about 21 minutes. This is due to a numerical overflow in the calculation of A_7 in Equation (42). Using double precision, an overflow occurs when the value of any calculated real variable exceeds the range of about 2.2×10^{-308} to 1.8×10^{308} (or the real or imaginary part of a complex number exceeds that range). This overflow occurs somewhere in the calculation of the coefficient of A_7. For other solutions developed in Chapter 2, overflow is encountered at small times typically in the calculation of one of the coefficients of a Bessel or exponential function. In some cases, predicted data can be generated for times small enough that a smooth curve can be drawn from the initial condition; while in other cases there is an obvious gap. The problem is usually worse when the partition coefficient is small.

And finally, the simulated data for nicotine generally shows a faster rate of release than is observed experimentally. Possible explanations may lie in the fact that the model was developed for dilute solutions, with diffusivity and partition coefficient independent of concentration. However, the experiments using nicotine as a model agent used up to 50% (505 mg/ml) nicotine in the reservoir initially; at such high concentrations the diffusion coefficient could be significantly reduced by interactions between molecules. Reid et al. [31] discuss several models for variation of the diffusion coefficient with concentration in concentrated binary solutions. A reduction in the diffusion coefficient at higher concentrations would cause a reduction in release rate from the device at earlier times (when the concentration of nicotine is high, and the release rate is apparently constant). This could result in better agreement between predicted and observed results.

If the diffusivity is dependent on composition, both D_1 and D_2 (for the reservoir and the membrane) respectively, will vary with position in the device and with time. Though the model does not consider the concentration dependence of diffusivity, the effect of a reduction in the concentration-independent diffusivities throughout the device (both reservoir and membrane) was examined. The dotted line in Figure 32 shows the predicted profile when both the reservoir and membrane diffusivities are reduced by 20%. Comparison of the two predicted release profiles shows simply the effect of diffusivity on the release rate, and one can reason that composition

dependence of the diffusivity could affect the release profile as mentioned above.

4 .1.5 Suspension in Reservoir: In Vitro Experiments

By having a solution of the agent in the reservoir of the membrane-based controlled release device, the loading of the agent in the reservoir is limited by its solubility in the reservoir solvent. If, however, the saturation concentration of the agent in the reservoir solvent is exceeded, a suspension of the agent will result. In this section results are presented for experiments using a reservoir initially containing a suspension of the agent. These experiments were performed using both hollow fibers and flat films, and for both an organic reservoir and an aqueous reservoir. The agents studied were benzoic acid (organic reservoir and water-filled pores) and caffeine (aqueous reservoir and organic-filled pores).

Figure 33 shows a comparison between release profiles obtained by using a suspension of benzoic acid in decanol, and a solution of benzoic acid in decanol. While the release profile obtained by using the solution is approximately zero order for a period of 30 minutes, this constant release is extended to about 180 minutes using a suspension. In addition, more benzoic acidis ultimately released from the suspension system, due to its higher loading initially. The model predicts that dissolution of solid benzoic acid into the reservoir will be fast relative to its diffusion out of the reservoir; in other words, as benzoic acid diffuses out of the reservoir it is replaced

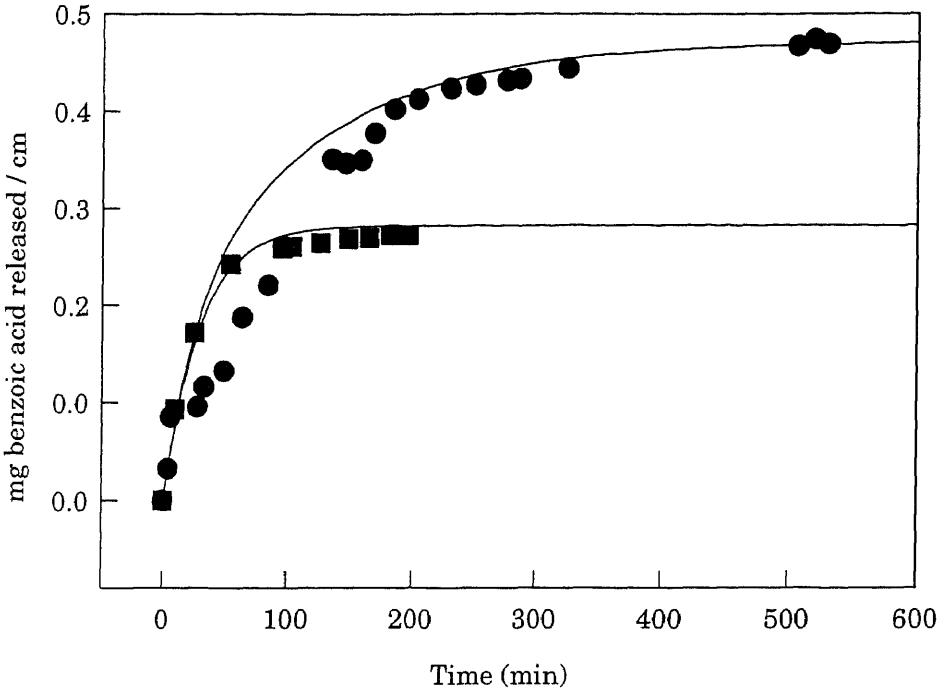

Figure 33. Extended release of benzoic acid using a suspension. System used: nylon porous hollow fibers with water-filled pores; benzoic acid in decanol suspension in the reservoir (170 mg/ml). Predicted results are shown with the solid curves. $k = 2 \times 10^{-7}$.

immediately by solids dissolving into the solution. (The value of the parameter k is found by running the simulation for different values of k (corresponding to different

rate limiting regimes) and taking the k which provides a qualitatively good fit.) The data show that the release rate from the device with the suspension in the reservoir is slightly slower than the release rate from the device with no suspension. This indicates that the dissolution is slightly rate limiting, and that a larger volume fraction of solids was actually present than was assumed in the model. This concept and the effects of dissolution rate and volume fraction of solids will be explored more thoroughly in Section 4.2.

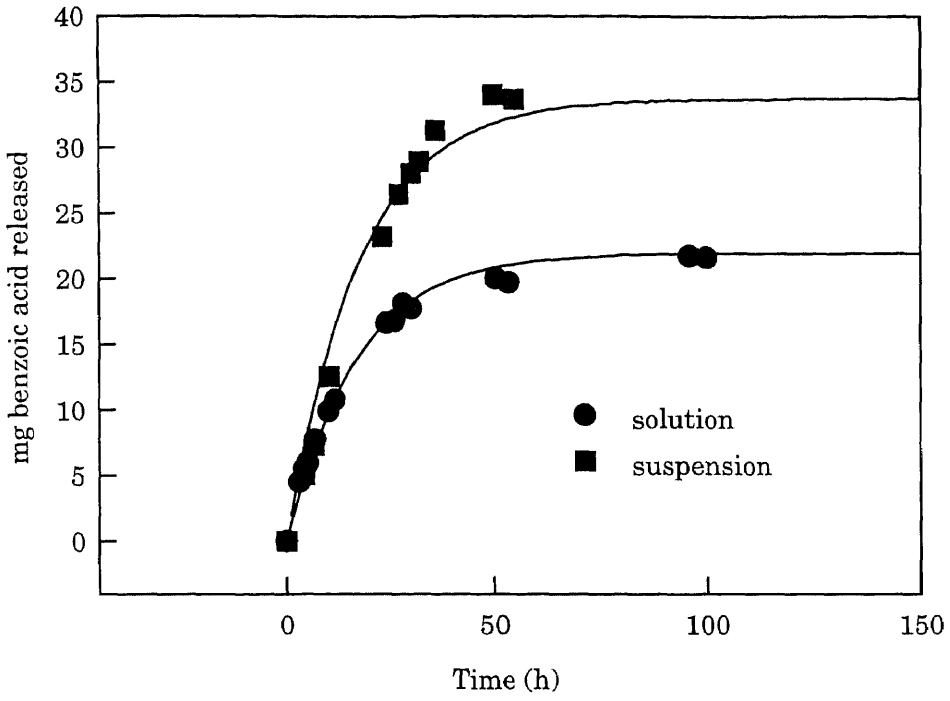

Figure 34. Extended release of benzoic acid using a suspension. System used: benzoic acid in octanol suspension (40.7 mg/ml), Celgard® flat membrane with water-filled pores. Predicted results are shown with the solid curves. $k = 1 \times 10^{-4}$

A similar system was studied using the flat membrane device. In this experiment, a suspension of benzoic acid in octanol was used for release through water-filled pores of a Celgard® 2400 membrane. Figure 34 shows the results of this experiment, as well as the results of an experiment which used a solution of benzoic acid in the reservoir. When the suspension of benzoic acid is used in the reservoir, the period of constant release is extended to about 40 hours. In this case the model describes the data well. The rate of release (for the constant release portion of the curve) is the same for the solution and suspension experiments, and from the data it can be concluded that the dissolution of the solids into the reservoir solution is rapid

relative to the depletion of benzoic acid from the reservoir solution due to diffusion.

Benzoic acid has a very limited solubility in mineral oil. In order to

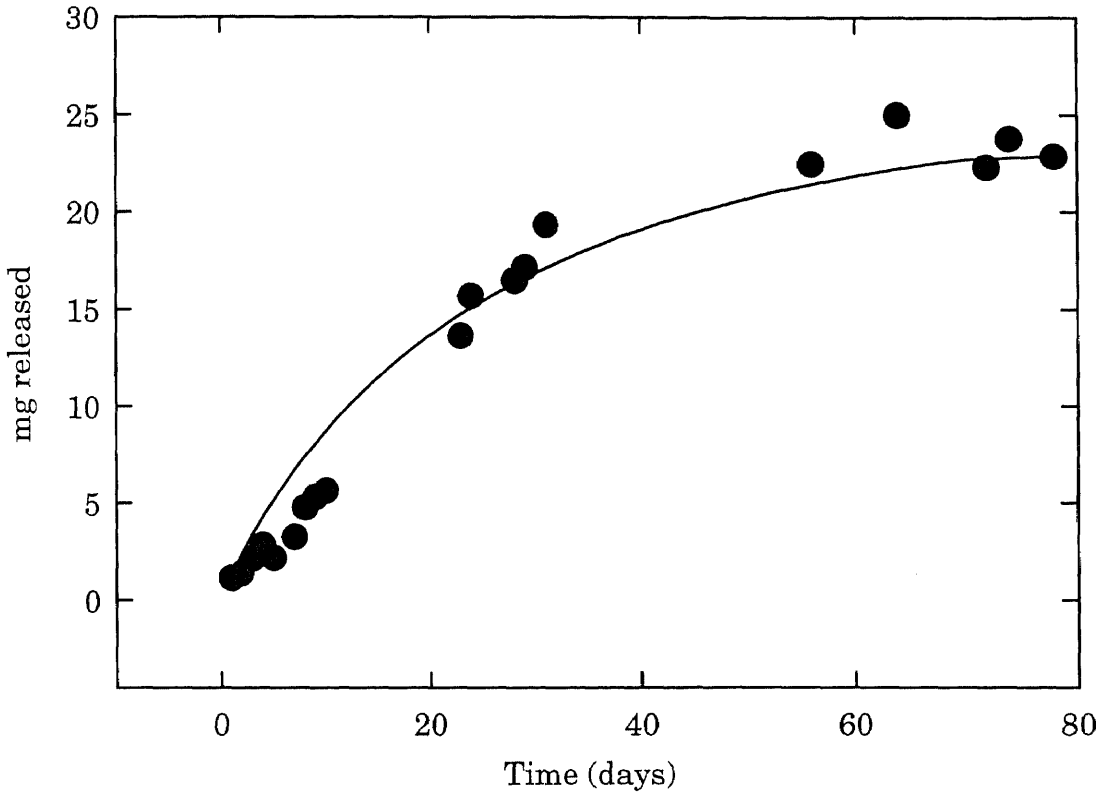

Figure 35. The effect of rate limiting dissolution on the release profile. Benzoic acid initially contained in a mineral oil suspension (106 mg/ml), Celgard® flat membrane with water-filled pores. Predicted results are shown by the solid curve. $k = 6 \times 10^{-6}$.

achieve a high loading in a system with a reservoir containing benzoic acid in mineral oil, a suspension of benzoic acid must be used. Figure 35 shows the release profile from such a system, with a suspension of benzoic acid in mineral oil in the reservoir of a flat membrane system, using a Celgard® 2400

flat membrane with water-filled pores. The partition coefficient of benzoic acid between mineral oil and water is 1.03, and no benefit is derived from partitioning. In this system the rate of release of benzoic acid from the device is severely dissolution limited. The release rate is constant for a period of about 45 days. The model describes the data well, but does not predict the constant release rate that is observed experimentally.

The next system that was investigated was a caffeine suspension in an aqueous reservoir, diffusing through octanol-filled pores of a Celgard® 2400 flat membrane. This system was investigated using both the 0.22 ml reservoir cell and the 0.95 ml reservoir cell.

Figure 36 shows the release profile obtained by using an aqueous caffeine suspension in the reservoir of the smaller (0.22 ml) cell. The release profile is constant for a period of 24 hours. While the model describes the data fairly well, it does not predict the constant release rate that the data reveal for the first 24 hours.

Figure 37 shows the results of a similar experiment using the larger (0.95 ml) cell. The data show a constant release rate for a period of 100 hours. Again, the model describes the data fairly well, but does not predict the same constant release rate that is observed experimentally.

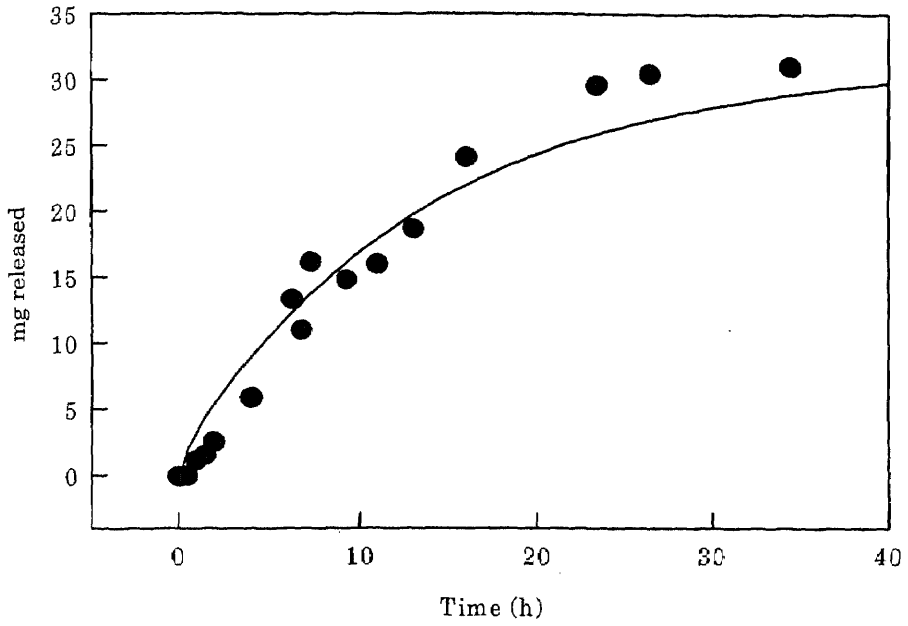

Figure 36. Extended release of caffeine using a suspension. System was caffeine in an aqueous 0.22 ml reservoir (146 mg/ml), Celgard® 2400 membrane with octanol-filled pores. Predicted results are shown by the solid curve. $k = 5 \times 10^{-5}$.

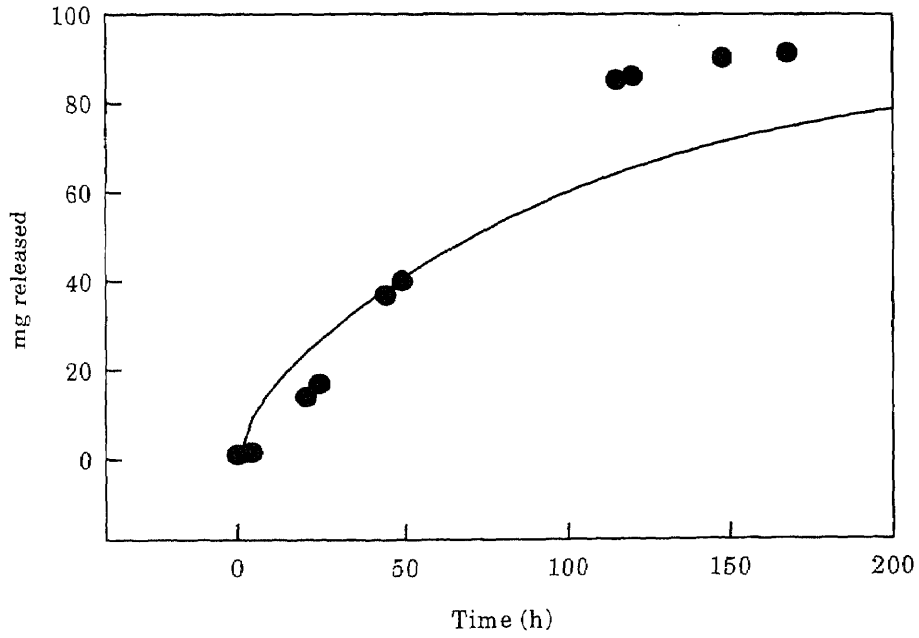

Figure 37. Extended release of caffeine using a suspension. System studied was caffeine in an aqueous 0.95 ml reservoir, and a Celgard® 2400 flat membrane with octanol-filled pores. Predicted results are shown by the solid curve. $k = 1.2 \times 10^{-5}$.

4 .1.6 Coated Membrane Experiments

A microporous membrane with a thin, nonporous coating on the outside presents two advantages over the uncoated membrane. It affords additional resistance to diffusion, which, in turn, means additional rate control. The coating also represents a barrier to loss of the pore fluid by evaporation. This is particularly important when a hydrophobic membrane with water-filled pores is exposed to air, a system which is prone to loss of pore fluid by evaporation.

In these experiments, microporous and porous hollow fibers with a coating on the external surface of the membrane were used to investigate the release of benzoic acid from a reservoir containing either a solution or a suspension. The specific membranes used were (1) a microporous polypropylene hollow fiber with a thin, nonporous silicone coating, (2) a microporous polypropylene hollow fiber with a thin, nonporous silicone coating plus an additional wax coating, and (3) a porous nylon hollow fiber with a wax coating.

Figure 38 shows data from two experiments using coated fibers. In one experiment benzoic acid, initially in a solution of octanol in the reservoir, was released from a polypropylene microporous hollow fiber with a thin, nonporous silicone coating. The second experiment was performed with the same type of silicone-coated membrane, with an additional wax coating that was applied in the laboratory. Both of these sets of data are compared with

the predicted results for a similar system with microporous hollow fiber with no coating at all. The simulation was not run for the coated membrane due to lack of information on the experimental parameters involving the coating layer. The data indicate that the silicone coating provides some additional

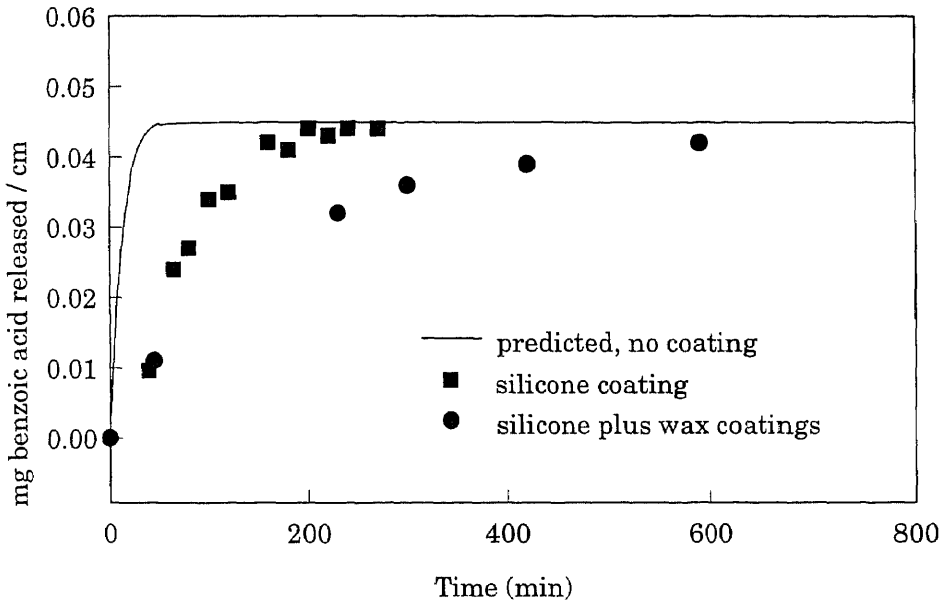

Figure 38. The effect of a nonporous external membrane coating on the release profile. System used was benzoic acid in an octanol solution (100 mg/ml) and a silicone coated microporous membrane with water-filled pores. The release profile using an additional wax coating is also shown. The predicted curves are shown by the solid lines.

resistance to diffusion, when compared to the predicted release profile for the same fiber without any coating. The release rate is constant for about 180 minutes. The wax coating provides an additional resistance, as indicated by

the lower release rate from the device. No model is provided for the system with two coatings.

Another experiment was performed using the silicone coated hollow fiber with a suspension of benzoic acid in decanol in the reservoir. This

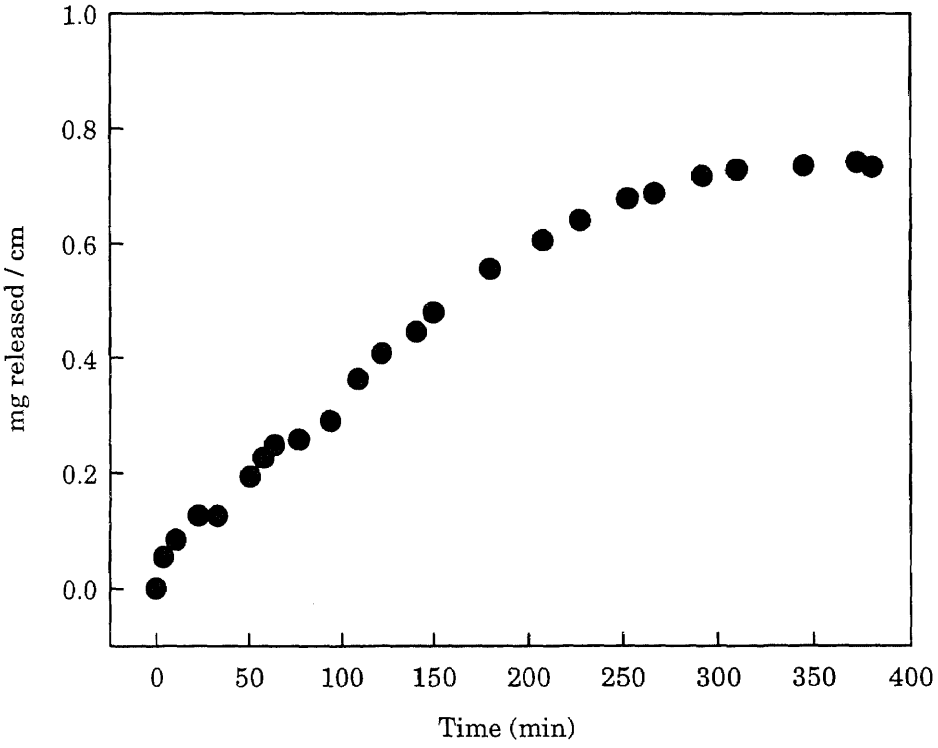

Figure 39. The release profile of benzoic acid from a silicone coated hollow fiber. System used was benzoic acid in octanol suspension (161 mg/ml), silicone coated hollow fiber with water-filled pores.

system provides extra loading from the suspension (benzoic acid dissolves quickly into decanol, relative to its rate of diffusion, so dissolution is not rate controlling) and additional rate control from the coating. Figure 39 shows the data obtained from such an experiment; the release rate appears

constant for the first 250 minutes. No model was developed for a system with a coated microporous membrane and a suspension in the reservoir.

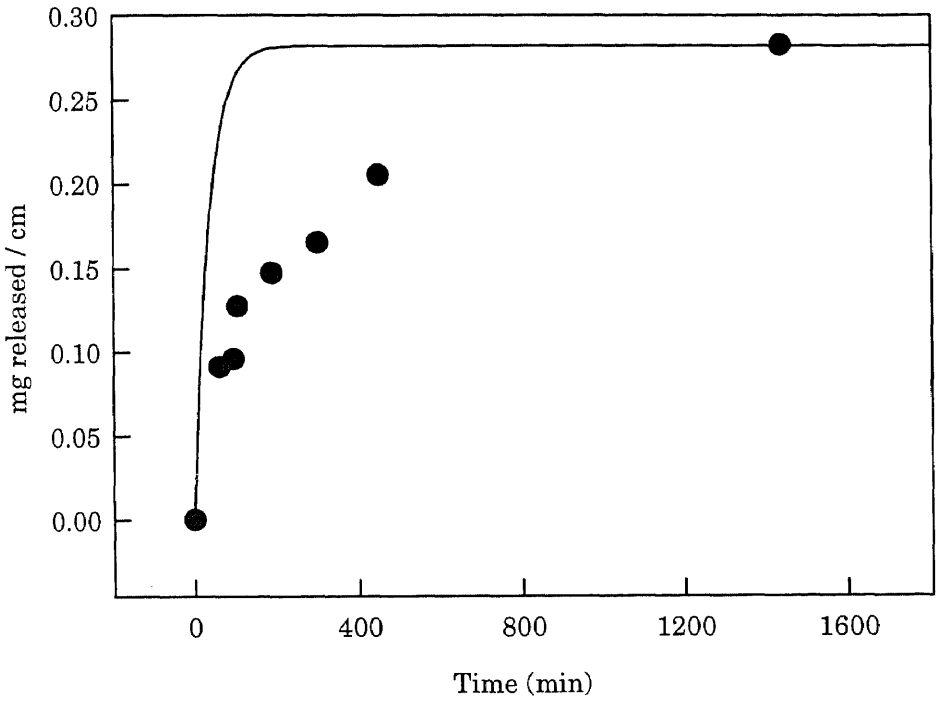

Figure 40. The effect of a wax coating on the release profile of benzoic acid from nylon porous hollow fibers. Benzoic acid was initially contained in a reservoir of 100 mg/ml octanol solution; pores of hollow fiber were filled with water. Predicted results for a similar system without an additional wax coating are shown by the solid curve.

Figure 40 shows the effect of a wax coating on the release profile of benzoic acid from nylon porous hollow fibers with water-filled pores. The agent was initially contained in an organic reservoir solution (100 mg/ml benzoic acid in octanol). The data can be compared to the predicted results for a similar system without a wax coating on the external surface of the

hollow fibers. The comparison of the data to this curve indicates that the wax coating provides an additional barrier to release of the agent; the release rate is significantly reduced using the system with the wax coating.

4 .1.7 Simultaneous Release of Two Agents

The configuration of this type of membrane based controlled release device lends itself naturally to the simultaneous release of two agents. If two agents are to be released from a single reservoir through the same membrane, the reservoir solvent and pore liquid must be chosen to effect the desired release rate for each agent. If, however, two reservoirs are provided within the same device, and each reservoir is bound by a separate membrane, then each solvent/agent/pore liquid system may be chosen independently. This type of system with two separate reservoirs provides more flexibility in system design than the type of system with one reservoir. In addition, concern exists that interaction between two agents within a single reservoir may alter the release rate of one or both of the agents. In this section results from experiments using both types of configuration are presented. Simultaneous release of benzoic acid and nicotine as well as simultaneous release of nicotine and caffeine are studied. Both flat membranes and hollow fiber devices are studied. The specific systems (each of which will be described in detail) involve various combinations of reservoir and pore configurations (e.g. aqueous reservoir/organic-filled pore, organic reservoir/water-filled pore, and reservoir suspension and reservoir solution). Experimental data are

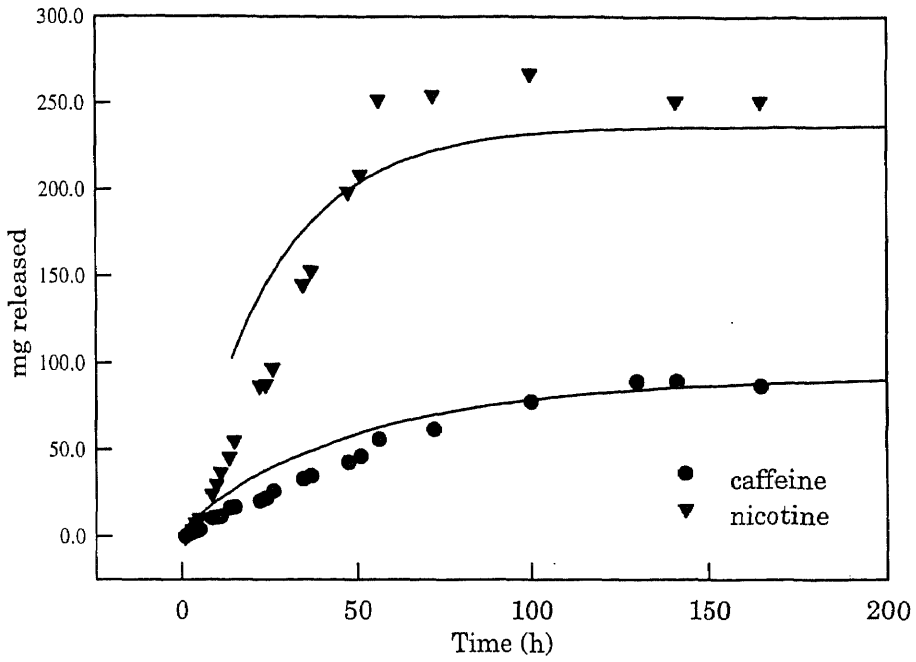

Figure 41. Simultaneous release of nicotine and caffeine from separate reservoirs. Systems used: Nicotine in aqueous solution(265 mg/ml) with a Celgard® 2400 flat membrane and mineral oil-filled pores; aqueous caffeine suspension (94.4 mg/ml) with a Celgard® 2400 flat membrane with octanol-filled pores. Predicted results are shown with solid curves. For caffeine simulation $k = 5 \times 10^{-5}$.

compared with predicted release profiles based on the model for single component release. This is justified for systems in which there is little interaction between the two agents, either in the reservoir or in the surrounding water bath.

Figure 41 shows the results of an experiment performed with caffeine and nicotine released from separate reservoirs. Nicotine was present in an aqueous solution in one reservoir, while caffeine was present initially in an aqueous suspension in the adjacent reservoir. Both reservoirs were bounded by a Celgard® film, but the pore fluids were different. The nicotine system

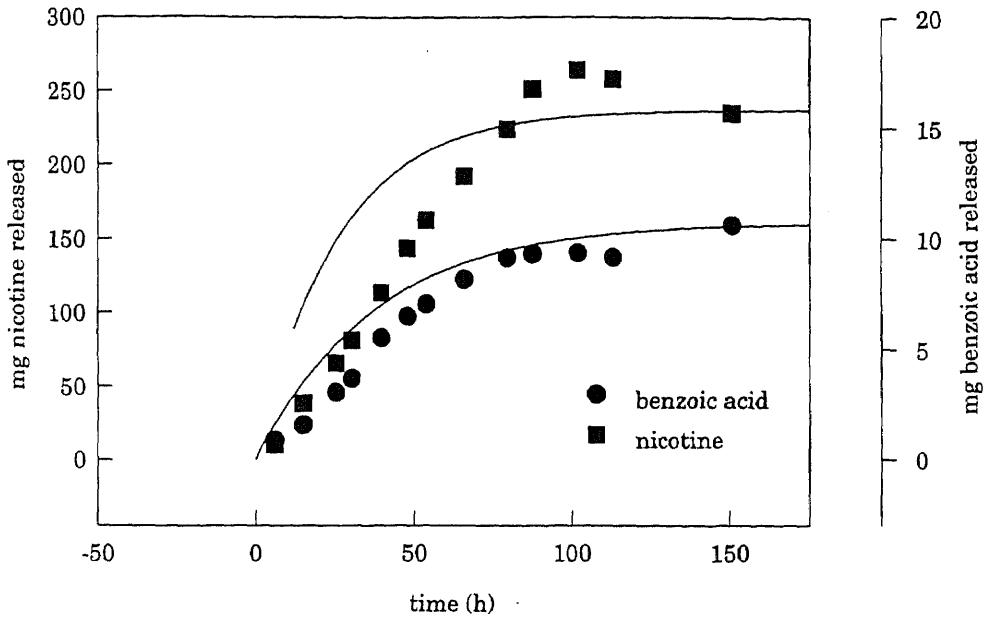

Figure 42. Release profiles for benzoic acid and nicotine from separate reservoirs. System used: Benzoic acid in octanol solution (100 mg/ml) and Celgard® 2400 film with water-filled pores; Nicotine in aqueous solution (265 mg/ml) with a Celgard® 2400 film with mineral oil-filled pores. Predictions are shown by the solid curves.

used mineral oil-filled pores, while the caffeine system used octanol-filled pores. The lower initial caffeine concentration and rate limiting dissolution contribute to an overall slower rate of release of caffeine than nicotine. The nicotine release rate appears constant for 50 hours, and the caffeine release appears constant for 140 hours. Again, the model describes the data well, but does not predict the constant release rate that both the nicotine and the caffeine data reveal.

Figure 42 shows release profiles for nicotine and benzoic acid from separate reservoirs. In this study the benzoic acid was initially contained in an organic reservoir and its membrane had water-filled pores, while nicotine

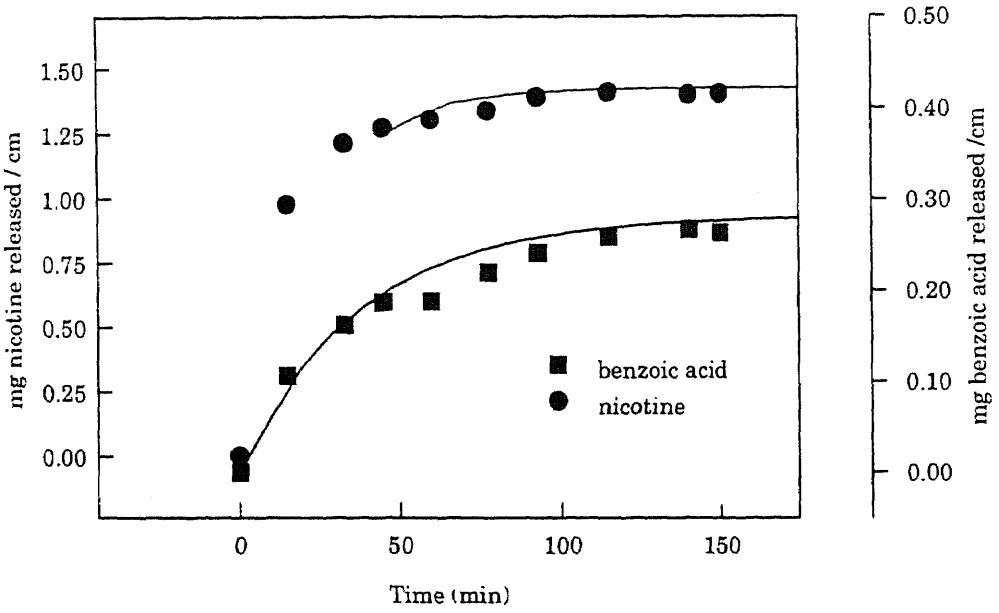

Figure 43. Simultaneous release of nicotine and benzoic acid from nylon porous hollow fibers. System used: Nicotine in aqueous solution (505 mg/ml) and a porous nylon hollow fiber with mineral oil-filled pores; Benzoic acid in octanol solution (100 mg/ml) and porous nylon hollow fiber with water-filled pores. Predictions are shown by the solid curves.

was present in an aqueous reservoir bounded by a membrane with mineral oil in its pores. Specifically, the benzoic acid was present in an octanol solution, and both membranes were Celgard 2400® films. The lower initial benzoic acid concentration in conjunction with the high partition coefficient of benzoic acid between octanol and water results in a slower release rate for benzoic acid than for nicotine. The release rate for benzoic acid appeared constant for approximately 60 hours, while a constant nicotine release was maintained for about 100 hours. While the data for benzoic acid lie fairly close to the predicted results, the data for nicotine lie above the predicted results at intermediate times.

Figure 43 shows simultaneous release profiles of nicotine and benzoic acid from porous nylon hollow fibers. Nicotine and benzoic acid were initially contained in separate reservoirs in the lumen of separate nylon hollow fiber segments. The nicotine was initially in solution in an aqueous reservoir and partitioned into mineral oil-filled pores of the porous nylon hollow fiber. The benzoic acid, initially contained in an octanol solution in the reservoir of the nylon hollow fiber, partitioned into water-filled pores. In this study, the release of benzoic acid was relatively constant for about 40 minutes, while the nicotine was released comparatively quickly and showed a constant release rate for less than 20 minutes.

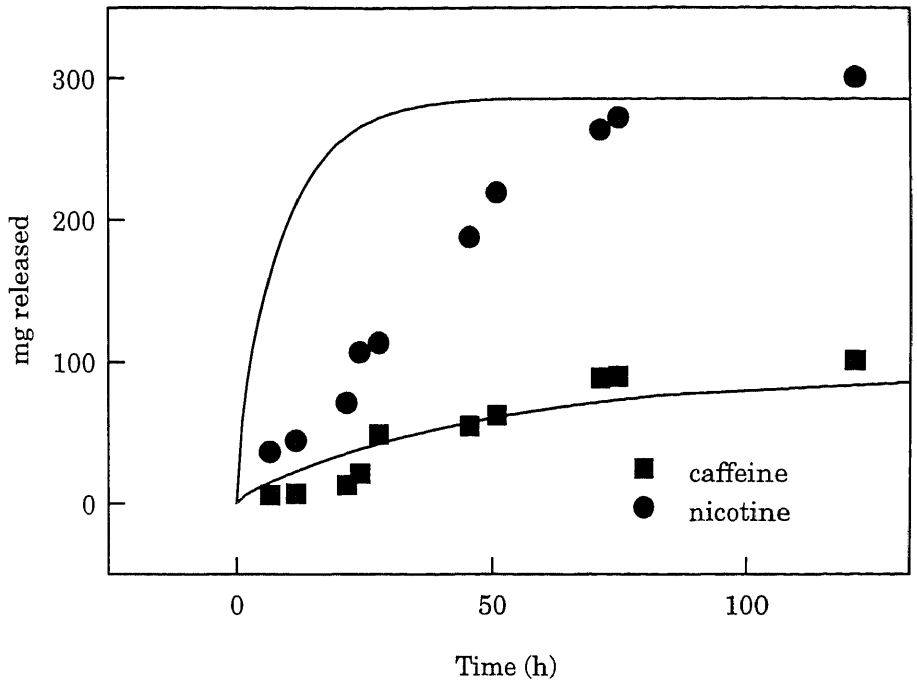

Figure 44. Simultaneous release of nicotine and caffeine from a single reservoir. Systems used: Aqueous reservoir of nicotine (319 mg/ml) and caffeine (99.4 mg/ml), Celgard® 2400 membrane with octanol-filled pores. Predicted results are shown by solid curves.

Figure 44 shows concentration profiles for nicotine and caffeine, obtained from an experiment in which the two agents were contained initially in the same aqueous reservoir. The nicotine was initially in solution, while the caffeine was in suspension. Both nicotine and caffeine release rates were constant for about 70 hours. A comparison of the simulated data and the experimental data shows again a discrepancy between the predicted and actual nicotine release rates.

4 .1.8 Pure Liquid Agent in the Reservoir

Figure 45 shows the release profile of toluene from a flat membrane device. In this experiment, toluene was contained in the reservoir as a pure solvent, and was released by diffusion through water-filled pores of the Celgard® 2400 flat membrane. The driving force for diffusion across the membrane was limited by the *solubility* of toluene in water, rather than by aqueous-organic partitioning. The experimental data indicate a constant release rate for a period of 150 hours. No model is presented for a system with a pure liquid in the reservoir.

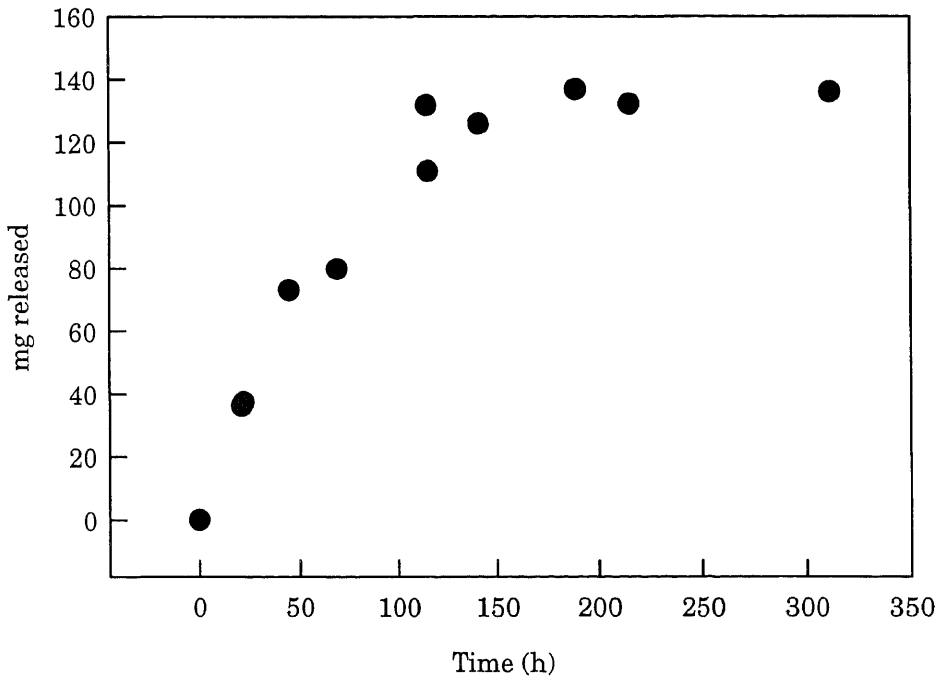

Figure 45. Controlled release of an agent initially a pure liquid contained in the reservoir. Reservoir: 0.95 ml toluene, Membrane: Celgard® 2400 film with water-filled pores.

4 .1.9 Peptides: In Vitro Experiments

Results of controlled release studies for Phe-Gly in an aqueous-organic partition-based controlled release device are presented here.

Partitioning of a peptide between an aqueous phase and an organic phase can exhibit a strong dependence on the pH of the aqueous solution. At low pH, the carboxyl group of the peptide in aqueous solution is protonated, and partitioning favors the organic phase. At intermediate and high pH, the carboxyl group is ionized, and the partitioning favors the aqueous phase.

This set of experiments was designed to investigate the potential for exploiting pH and aqueous-organic partitioning in the controlled release of peptides. The peptide Phe-Gly was chosen because it has an intermediate hydrophobicity of about 1250 cal/mol [9], and was expected to exhibit a measurable release from the device under a range of pH conditions. The reservoir contained a buffered aqueous dispersion of the peptide; this was bound by a flat Celgard® polypropylene membrane with octanol-filled pores. The surrounding aqueous phase was a buffer solution at the same pH as the reservoir. (Note that the surrounding bath does not necessarily have to be at the same pH as the reservoir, and this would of course affect the release profile). Experiments were performed at three values of pH: pK_a, pK_b, and pI.

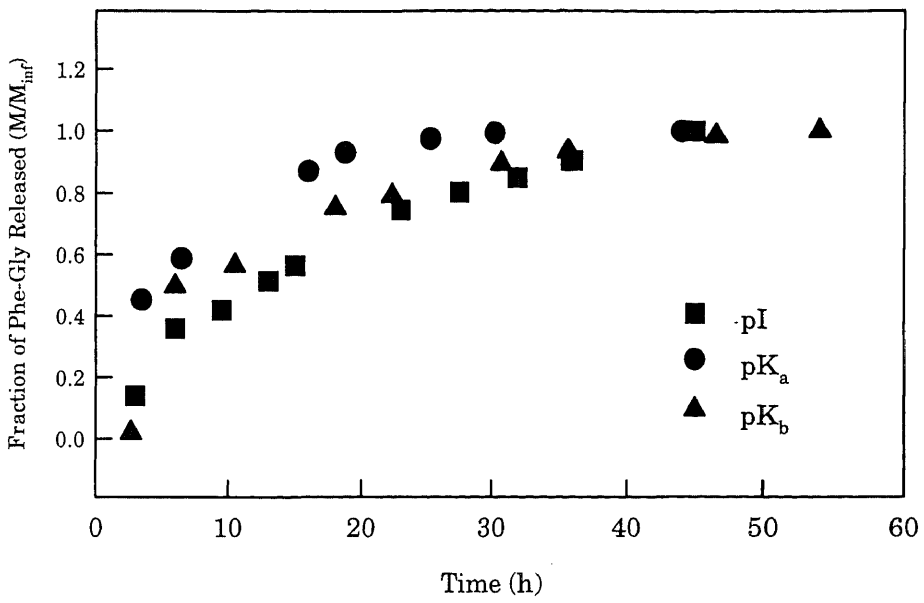

Figure 46. pH dependence of release profile for the peptide Phe-Gly

The release profile of Phe-Gly as a function of pH is shown in Figure 46. Here the fraction of peptide released (M/M_{inf} where M_{inf} is the mass released at infinite time) is plotted as a function of time. At a low pH (pK_a) partitioning of the peptide favors the octanol in the membrane pores, and the release is relatively fast. Most of the agent has been released within 20 hours. At higher pH (pI), partitioning favors the aqueous reservoir, and the release is slow. The agent continues to be released for about 45 hours. There is little change when the pH is raised to the pK_b of the peptide. The dramatic difference in release rate between experiments performed at pK_a and pI is due to the strong dependence of the partition coefficient on the ionized or unionized state of the carboxyl group. The partition coefficient does not exhibit a strong dependence on the ionization of the amino group, and

therefore there is little difference between release profiles for experiments at pI and pK_b.

4 .1.10 Liposomes: In Vitro Experiments

It was mentioned in Section 4.1.2 that the fluorescence of NBD is environmentally sensitive. In a controlled release study it is desired to measure the amount of SUVET that is released from the reservoir within a given time period. Since it is not possible to prevent the loss of small amounts of NBD from the lipid bilayer, fluorescence studies were performed in order to ensure that only NBD embedded in the lipid bilayer of the SUVET is detected, and that free NBD cannot give a false indication of the presence of SUVET.

The controlled release experiment was performed using a cell with a 0.9 ml volume, bounded by a polycarbonate 18 μm diameter pore size membrane. Polycarbonate was chosen as the membrane material because it is known that adsorption of the liposome onto its surface is negligible, and polycarbonate membranes are used for extrusion of the liposomes. A 150 ml surrounding aqueous buffer phase was used. Figure 46 shows the release profile of the liposome from the device. The release profile is zero order for the first 30 hours, during which almost all of the agent is released.

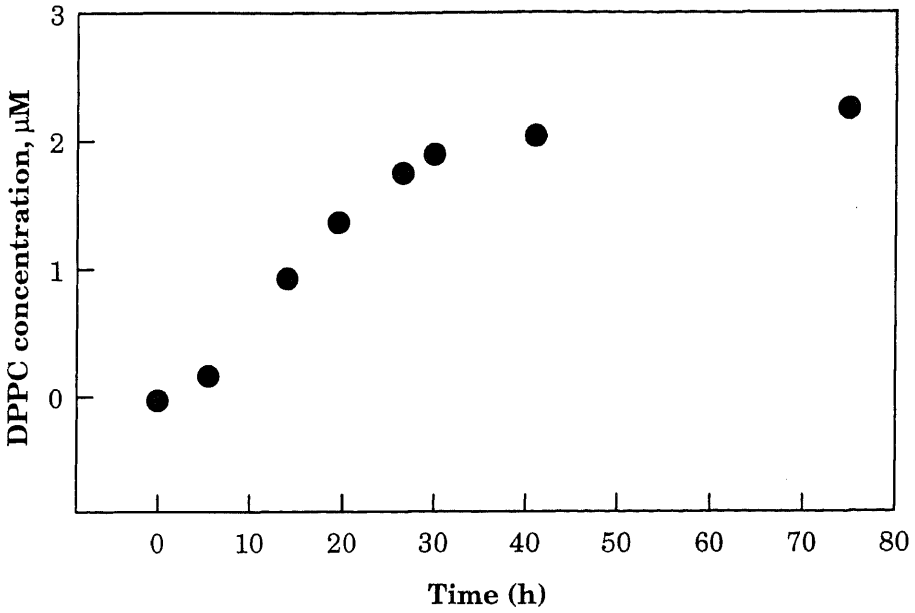

Figure 47. DPPC release profile

4.2 Dimensional Analysis

4 .2.1 Uncoated Membrane, Solution in Reservoir

In developing the model for this problem, writing the governing equations in semidimensionless form facilitated the identification of time constants τ_r and τ_m for the reservoir and pore liquid regions respectively. These time constants are derived from equations (12) and (13) for the reservoir and pore liquid regions respectively:

$$\tau_r = \frac{a^2}{D_1} \qquad \tau_m = \frac{(b-a)^2}{D_2} \tag{72}$$

The ratio of these time constants gives an indication of which resistance (reservoir or membrane) controls the release rate from the device. Since a large time constant corresponds to a large resistance in a particular region, a

large value of $\psi = \tau_r/\tau_m$ (>>1) indicates that the reservoir resistance controls, while a small value of ψ (<<1) indicates that membrane resistance controls. From the first boundary condition at ξ=a/b (equation (23)), the partition coefficient $m_{1,2}$ is isolated as an important parameter in the model. In this section the effect of these dimensionless groups will be studied via numerical simulations of the governing equations.

Figure 48 shows the effect of pore liquid resistance on the release profile, when there is no effect from partitioning. Here the ordinate is M/M_{inf}, the ratio of agent released at time θ to that released at infinite time (equilibrium). When ψ is small, the pore liquid limits the rate of release from the device, and a lower flux (slope of the release profile in Figure 48) results. When ψ is large, the membrane does not provide significant mass transfer resistance, and the solute is released quickly. Figure 49 shows the effect of membrane resistance on the release profile when there is significant partitioning between the reservoir and pore liquid phases ($m_{1,2}$ = 100). The effect of ψ shows the same trend as it did without the effect of partitioning, but with lower fluxes due to the reduction of driving force for diffusion across the membrane due to partitioning. For membrane control (ψ=0.1) with significant partitioning ($m_{1,2}$ = 100), the flux is maintained at a low and constant value for an extended time.

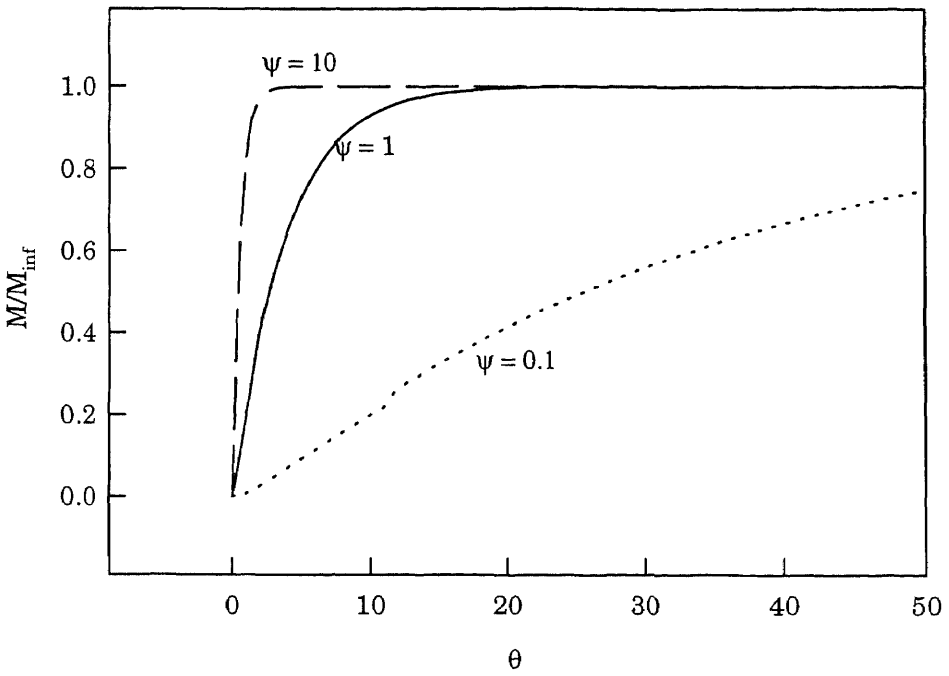

Figure 48. The effect of membrane resistance on the release profiles, for $m_{1,2} = 1$

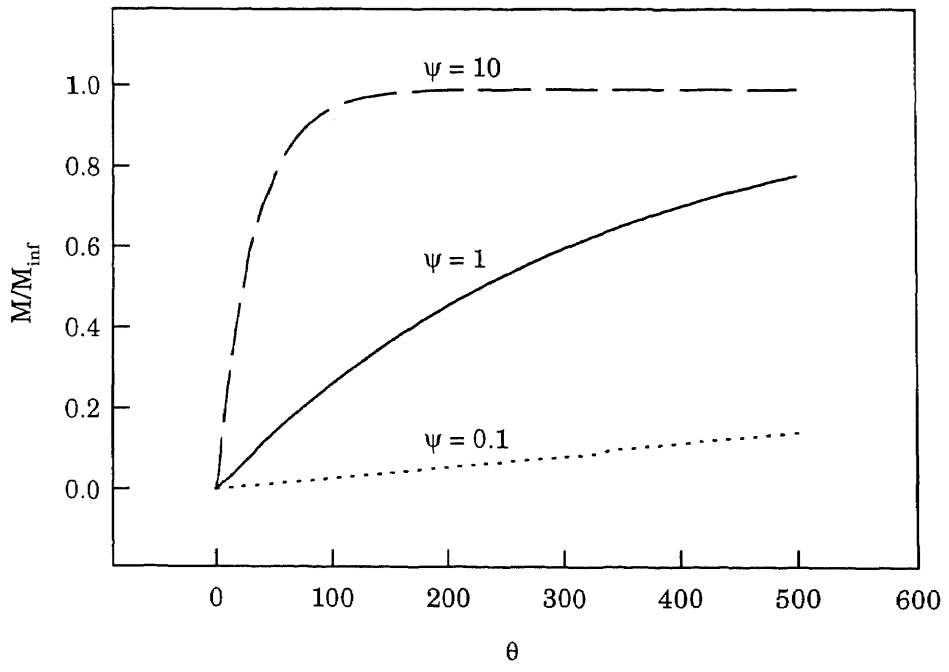

Figure 49. The effect of membrane resistance on the release profile, for $m_{1,2}$=100

A large value of $m_{1,2}$ indicates that the local concentration of agent in the reservoir is larger than its local concentration in the pore, at the interface between the two phases. This reduces the driving force for diffusion through the pore liquid phase, resulting in a lower flux. This effect is shown in Figure 50 for membrane-controlled release. When $m_{1,2}$ =1 there is no effect from partitioning; with increasing $m_{1,2}$ the flux is reduced and the time for which the agent appears to be released at a constant rate is extended.

Another important time constant which can be identified from the model (Equation(17)) is the ratio $\frac{V_w(b-\alpha)}{D_2\alpha_2}$. This is a measure of the

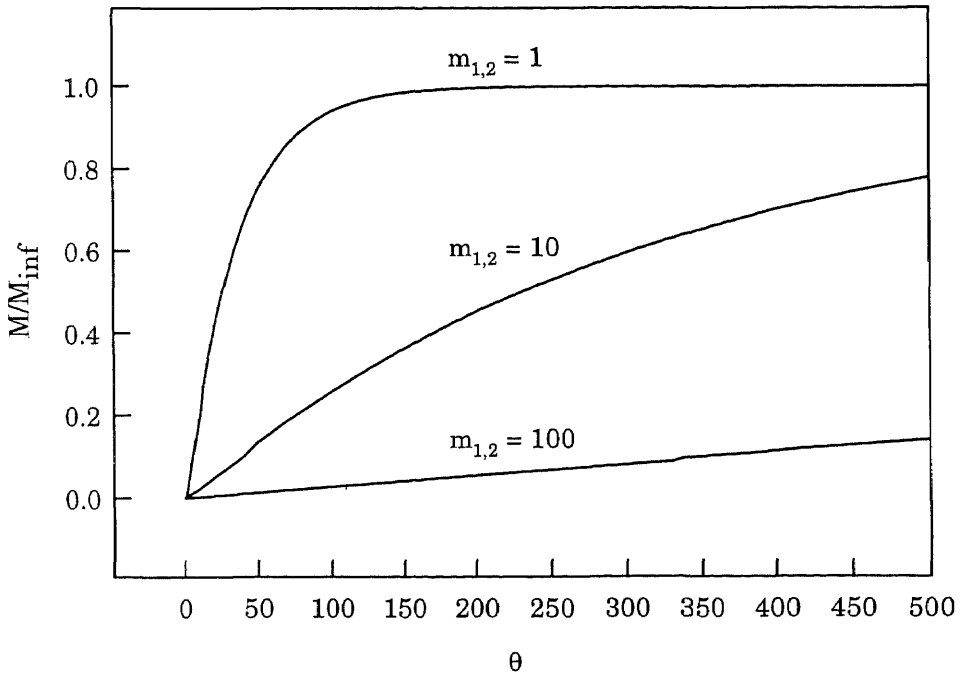

Figure 50. The effect of partitioning on the release profile, for ψ=0.1

characteristic time for diffusion out of the device at the external surface of the membrane, and will be designated τ_w.

Figure 51 shows the effect of the ratio of τ_m/τ_w on the release profile. Since the volume of the well stirred water bath is much larger than the volume of the device (at least 2 orders of magnitude in the experiments), the

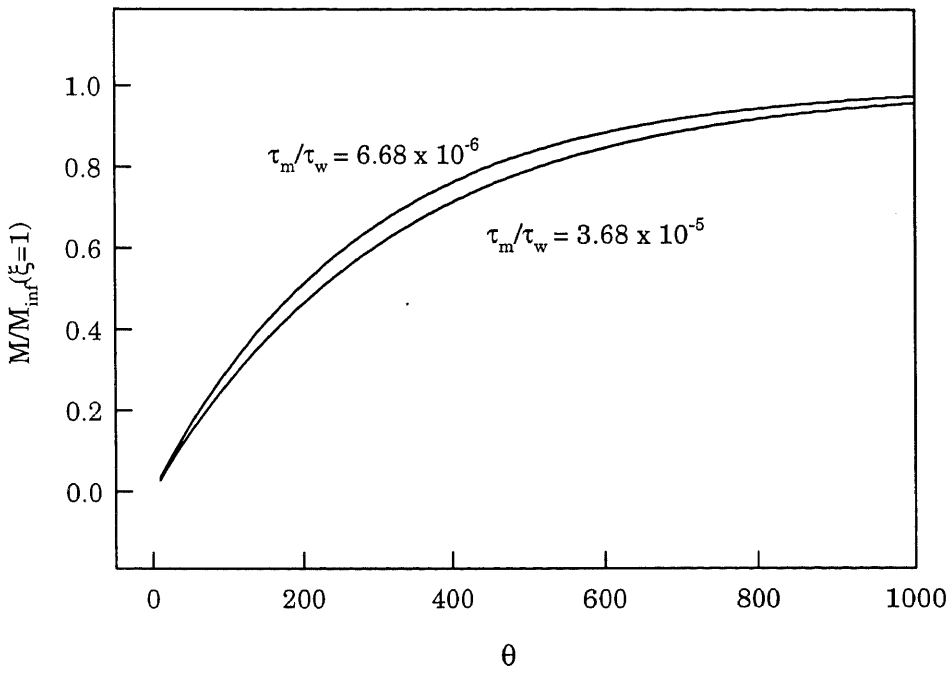

Figure 51. The effect of τ_m/τ_w on the release profile. $\psi = 1$, $m_{1,2} = 100$.

ratio of τ_m/τ_w is, for practical purposes, always very small. In this simulation the volume of the water bath was 5850 times the volume of the reservoir for $\tau_m/\tau_w = 3.68\text{x } 10^{-6}$, and 585 times the volume of the reservoir for $\tau_m/\tau_w = 3.68 \text{ x } 10^{-5}$. Despite this enormous difference, a one order of magnitude change in

the ratio of τ_m/τ_w has a minor but noticeable impact on the release profile from the device.

For the flat membrane, it has been demonstrated that a large partition coefficient and a small value of ψ (ratio of reservoir to membrane time constants) effect a constant release rate which appears constant for a relatively long period of time. Now these same parameters will be investigated for the hollow fiber configuration.

Figure 52 shows the effect of membrane resistance on the release profile using a hollow fiber system. Here, the partition coefficient $m_{1,2}$ is taken to be 100. Release profiles are shown for $\psi = 10$, $\psi = 1$, and $\psi = 0.1$,

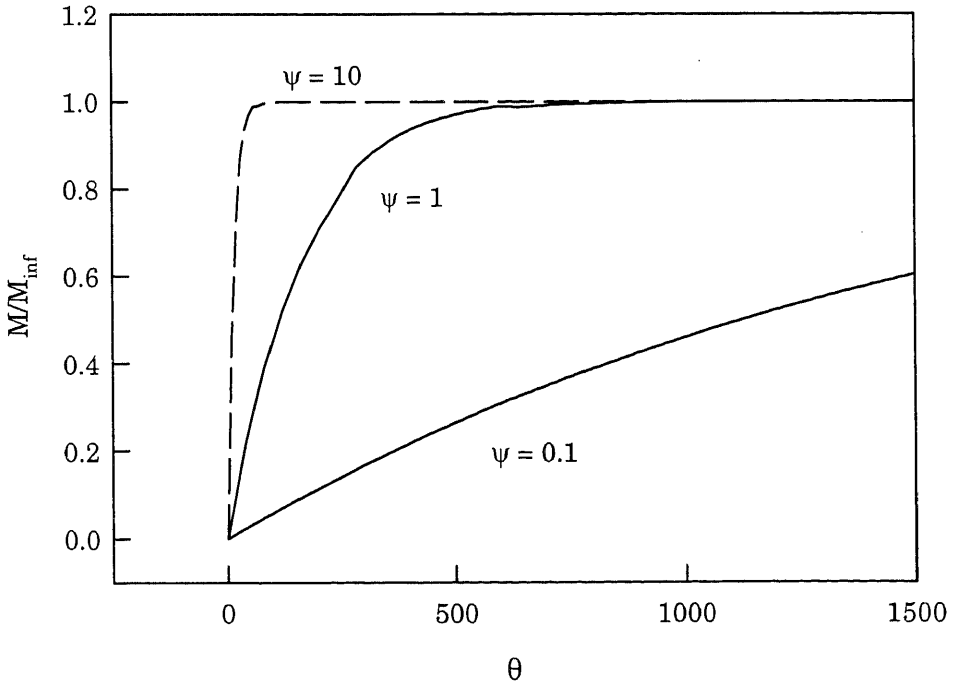

Figure 52. The effect of membrane resistance on the release profile for the hollow fiber configuration, $m_{1,2}$=100

representing increasing membrane resistance relative to reservoir resistance. The trend is the same as that established for the flat membrane configuration: A smaller value of ψ corresponding to a larger membrane resistance effects a slower release rate from the device, and one which appears relatively constant for a longer period of time.

Figure 53 shows the effect of partitioning at the interface of the reservoir and the pore on the release profile from a hollow fiber device. The curves were generated for $\psi = 0.1$. As expected, the trend is the same for the hollow fiber as it was for the flat membrane: A high value of partition coefficient $m_{1,2}$ effects a slower release rate as it reduces the driving force for

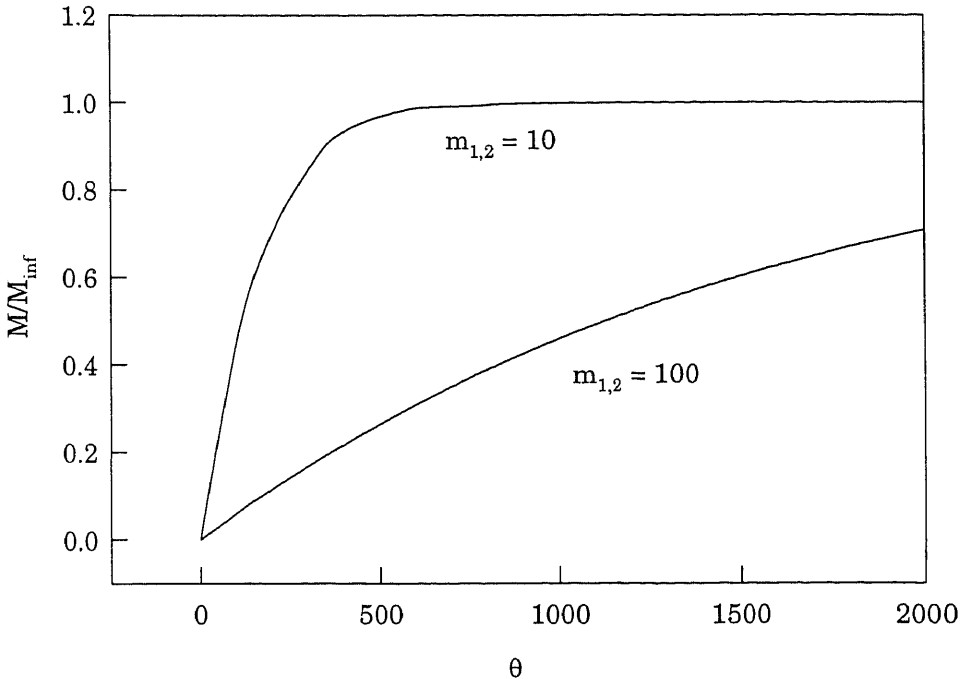

Figure 53. The effect of partitioning on the release profile from a hollow fiber device, $\psi = 0.1$

diffusion across the membrane. The release rate appears relatively constant for longer times with increasing partition coefficient.

It is interesting to look at the concentration profiles within the system. Figure 54 shows dimensionless concentration profiles as a function of radial position in the reservoir and the pore regions. The profiles were generated using $\psi = 1.0$, $m_{1,2} = 100$, and θ as a parameter. For the reservoir region, the concentration decreases with time, and appears to have an extremely weak, if any, dependence on radial position. Within the pore, however, the concentration profile appears to be linear with respect to radial position, and the slope of this profile decreases with time. The flux of the agent from the device is proportional to the slope of the concentration profile in the pore at ξ

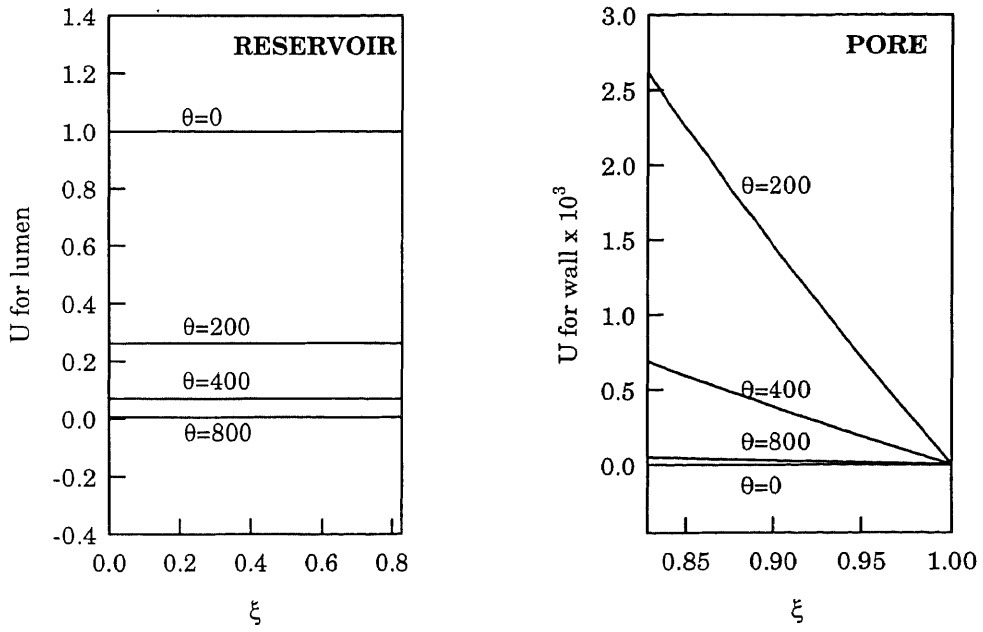

Figure 54. Concentration profiles within the hollow fiber device, $m_{1,2} = 100$ and $\psi = 1$

= 1, and also appears to decrease with time. However, this flux decreases fairly slowly with time – the flux decreases by a factor of three over 200 time constants (from $\theta = 200$ to $\theta = 400$).

Figure 55 shows the concentration profiles inside the hollow fiber device, for $\psi = 0.1$, $m_{1,2} = 100$, and θ as a parameter. In this case the time constant for diffusion across the pore is 10 times that for diffusion through the reservoir. While the reservoir concentration profile appears still to be a function only of time, the concentration profile of agent in the membrane is a function of both radial position and time. The slope of the profile at the outer surface of the device ($\xi = 1$), which is proportional to the flux, changes even

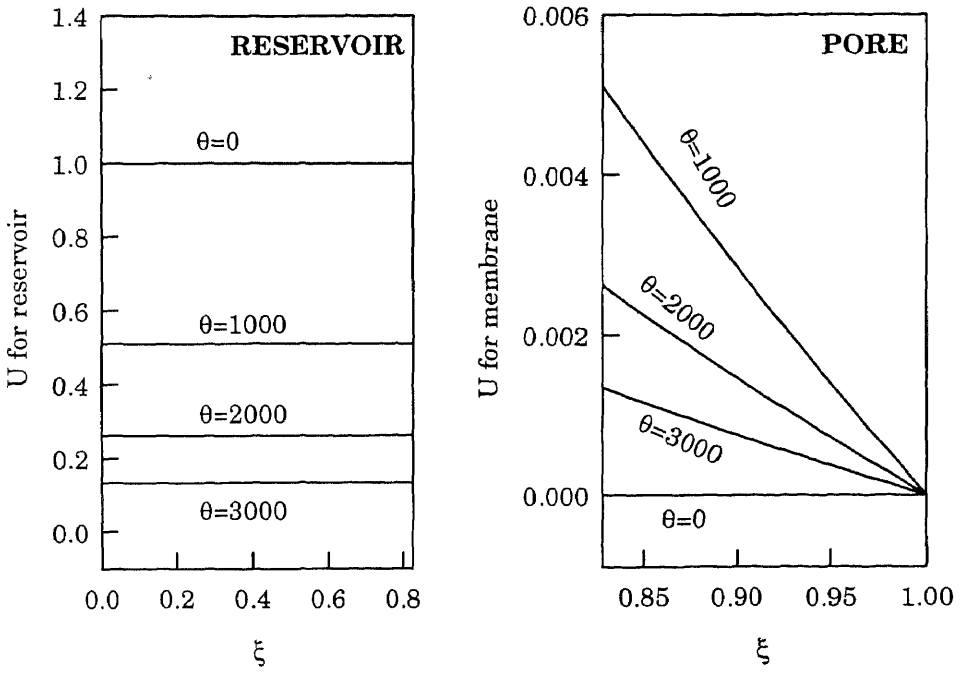

Figure 55. Concentration profiles within the hollow fiber device, $m_{1,2} = 100$ and $\psi = 0.1$

more slowly with time. The flux at the outer surface decreases only by a factor of 2 over a period of one thousand time constants. This accounts for the relatively constant release rate from the device that is apparent from the release profile shown in Figure 52 for $\psi = 0.1$ and $m_{1,2} = 100$.

In both Figure 54 and Figure 55 the concentration profiles in the reservoir appear to change with time only (i.e., concentration is independent of radial position), while pore concentration varies linearly (i.e. the slope of concentration vs. distance is constant) with radial position and is also a function of time. In cases such as these, a simpler model may have been applied. However, the model is a general one which considers the possibility of reservoir and pore concentrations and fluxes varying with both time and position. Figure 56 shows concentration profiles in the device for a case in which the reservoir concentration is a function of position.

4 .2.2 Uncoated Membrane, Suspension in Reservoir

When the solubility concentration of the agent in the reservoir solution is exceeded, some of the agent will be present in a solid phase in the reservoir. As the agent in the reservoir solution diffuses out, it is replenished by solid phase agent dissolving into the reservoir solution. If this dissolution is fast relative to diffusion of the agent out of the reservoir, the reservoir concentration will be maintained at its saturation level until all of the solids have dissolved. If, however, the dissolution of the agent into the reservoir

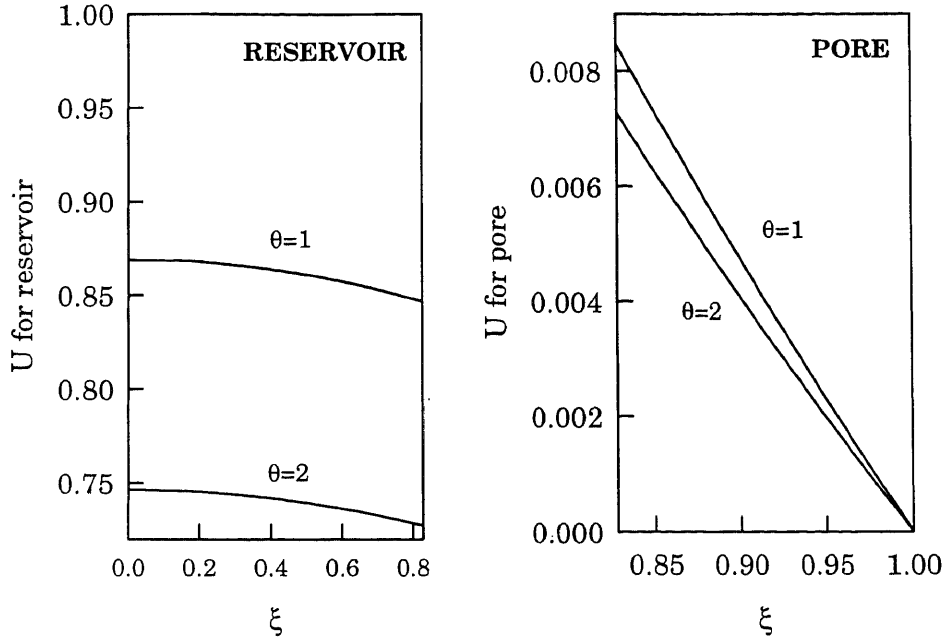

Figure 56. Concentration profiles in the hollow fiber device for $m_{1,2}$ = 100 and ψ = 1.0

solution is slow compared to diffusion, the rate at which the solid dissolves may be the controlling factor in its release from the device. Intuition tells us that the more solids originally present in the reservoir, the longer this slow rate of release will be maintained. Next the effect of these parameters will be examined quantitatively.

The time constant for dissolution, τ_d =k/V, is extracted from Equation 57 (the coefficient of the driving force for mass transfer between the dispersed solid and liquid reservoir phases). When the ratio of τ_r to τ_d, is large, diffusion through the reservoir is rate controlling; when this ratio is small, dissolution is rate controlling. In Figure 57 the effect of this ratio on the release profile is examined. As τ_r/τ_d decreases, dissolution becomes more rate

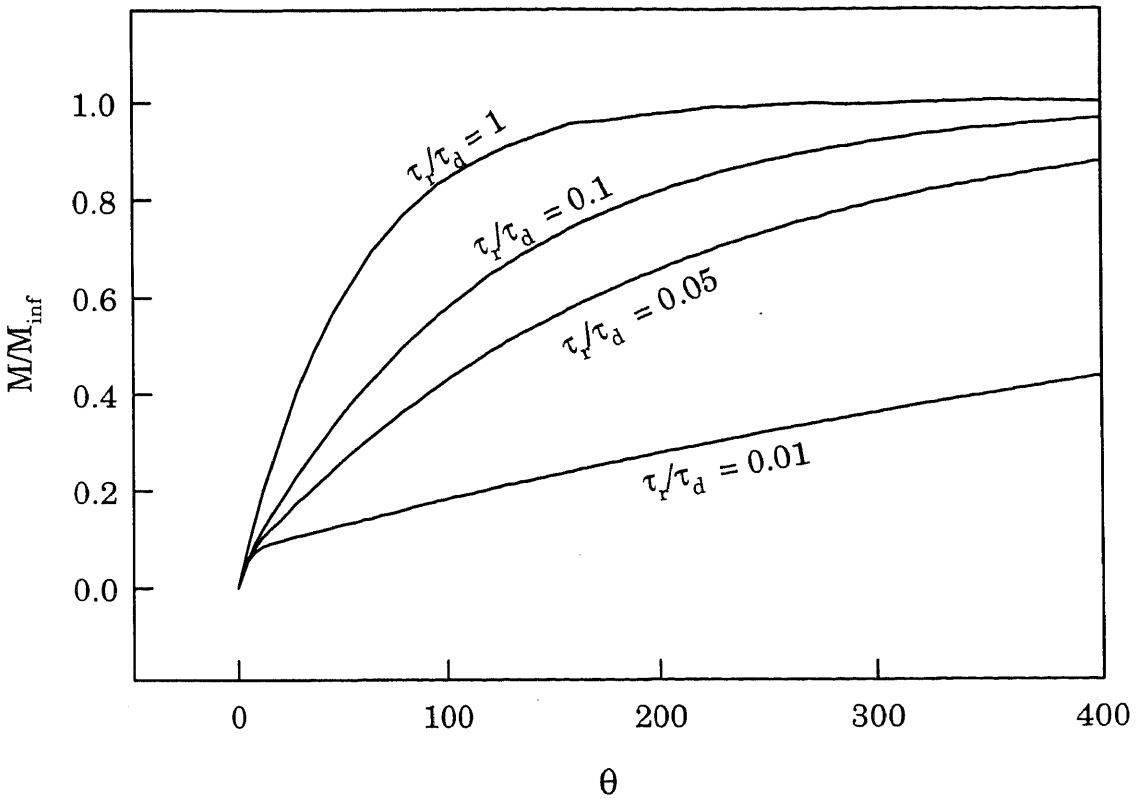

Figure 57. The effect of dissolution rate control on the release profile from a flat membrane device; $\tau_r/\tau_m = 1$

controlling. And the flux is lower and apparently constant for a longer period of time.

Next the effect of the group $\zeta = (1 - \phi)/\phi m_{1,s}$ on the release profile is examined. This group represents the ratio of the time constant for change in dispersed solid concentration and the time constant for the change in reservoir solution concentration, and can be extracted from equations (57) and (61). This group is the ratio of the amount of agent present in the reservoir solution to the amount present in the solid -- the smaller its value, the greater the fraction of agent present in the solid region: As shown in

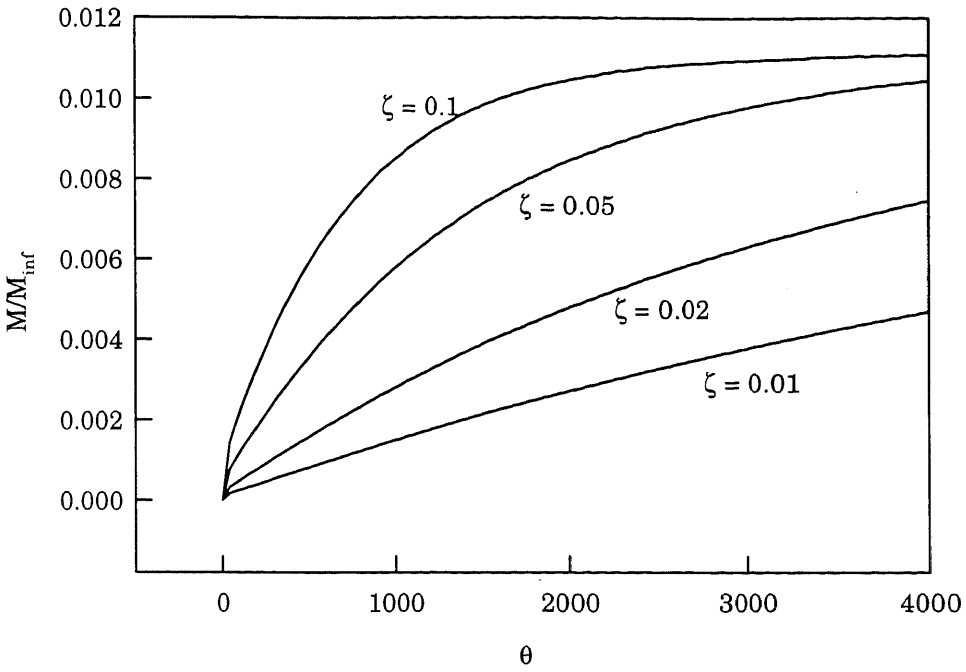

Figure 58. The effect of ζ on the release profile; $\tau_d/\tau_r = 100$

Figure 58, as ζ decreases, more rate control is derived from dissolution, and the flux of agent from the device is lower and apparently constant for a longer time.

4 .2.3 The Coated Membrane

The model for the coated hollow fiber controlled release device was not used to describe the experimental results because of a lack of information on the values of certain physical parameters relating to the coating. It is useful to examine the effect of the coating thickness on the release profile for at least a hypothetical situation. The simulation was run for two cases: one in which the coating thickness was zero – in this situation, the simulated results are

exactly the same as they would be for the uncoated fiber. The second case was for a system with a thicker coating than the one used in the experiment, but this was necessary to observe the effect of the coating thickness with all other system parameters held constant. The release profiles are shown in Figure 59. The dimensions of the fiber in this second case were a=0.012 cm, b = 0.015 cm, and c = 0.017 cm. This coating thickness is ten times the

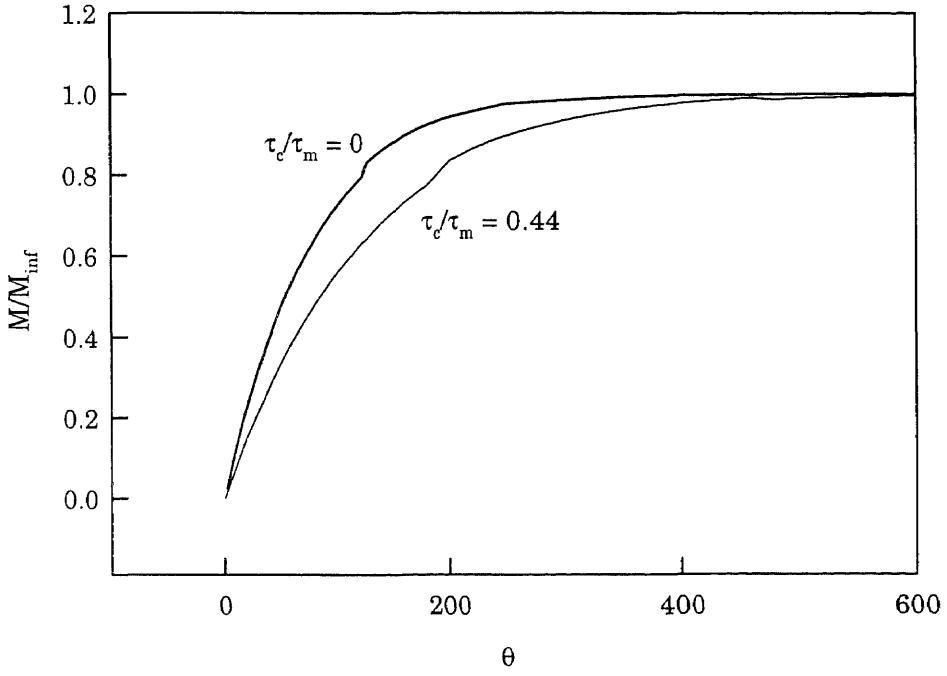

Figure 59. The effect of the coating thickness on the release profile

coating thickness of the silicone-coated hollow fibers.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The feasibility and potential for microporous /porous membranes as controlled release devices in aqueous-organic partition-based systems was established, and a basic analysis as well as information regarding the agent release rates were presented. The work was extended to include a hybrid dispersed-reservoir phase/membrane diffusion limited system. Various solvent/agent systems were investigated, different membranes were studied, and the significance of various factors in achieving zero order release of the agent were evaluated. Following are the conclusions drawn from the results of these studies.

- Aqueous-organic partitioning is an effective mechanism for controlling the release rate of an agent from a membrane-bound reservoir system. A high partition coefficient reduces the concentration driving force for diffusion across the membrane thereby reducing the release rate of the agent. A constant release rate can be achieved for an extended period if the partition coefficient is high.
- When an agent is present in excess of its solubility concentration in the reservoir, the duration of constant release rate is extended further, and more agent is released.

- Slow dissolution of an agent in excess of its solubility concentration in the reservoir can be an additional rate controlling mecahnism.
- A thin nonporous coating on the external surface of the membrane provides an additional resistance to diffusion of the agent, and can slow its release rate significantly.
- Simultaneous release of two agents from an aqueous-organic partition-based system can be achieved. The release rates of the two agents may be independent, or interaction between the two agents (either in the reservoir or in the water bath after release) may impact the release rates.
- Controlled release of a pure liquid agent, an organic with low water solubility, can be achieved using this type of device. This was demonstrated using toluene as a model agent, and demonstrates the potential to use a very high concentration of agent in the reservoir, while achieving a controlled release rate by extreme aqueous-organic partitioning which favors the organic reservoir phase.
- Controlled release of a peptide can be implemented using this type of aqueous-organic partition based system, exploiting the pH dependence of the partition coefficient of the peptide between water and an organic phase to achieve different release rates.
- Controlled release of liposomes from an aqueous reservoir can be achieved using a water-filled microporous membrane-bound reservoir device.

- The models developed for the aqueous-organic partition based systems describe the data well unless the physical situation deviates significantly from the ideal system of the model, e.g. when one or more regions contain a very concentrated solution of the agent.
- Parametric studies of the model developed for the system with a solution in the reservoir reveal that either a high aqueous-organic partition coefficient or severe membrane resistance can effect a constant release rate for an extended time.
- Parametric studies of the model developed for the system with a suspension in the reservoir indicate that dissolution rate control and high loading of undissolved agent in the reservoir can effect a constant release rate for an extended time.
- The model for the coated hollow fiber shows that a thin coating on the external surface of the membrane reduces the release rate of the agent.

5.2 Recommendations

Future work should include a thorough systematic experimental investigation of the system, considering the important dimensionless groups that were identified from the model. Reservoir resistance and membrane resistance should be varied independently. A more thorough experimental investigation of the effects of the aqueous-organic distribution coefficient would be useful.

For the system of a suspension initially in the reservoir, the effect of dissolution kinetics should be investigated more thoroughly. Values of the mass transfer coefficient k should be measured experimentally. The effect of the volume fraction of solids would also be interesting to investigate experimentally.

In the simultaneous release studies, there is a possibility of interaction between two agents having an effect on the release rate of one or both. This was seen in systems with a weak acid and a weak base. An investigation of these interactions should be conducted, beginning with a system that is known to be non-interactive.

Liposomes have been used effectively in controlled delivery and targeting. Using the method of controlled delivery of liposomes presented here, a future investigation should consider controlled release of an agent from a liposome in conjunction with controlled liposome delivery.

Consideration should be given to improving the mathematical model which describes the system. Several assumptions were made in developing the model, and these are obvious places to begin the improvements. The most fundamental modification to be considered would be the concentration dependence of the distribution coefficient. In the model for suspension initially in the reservoir, consideration should be given to the decrease in volume fraction of solids as a function of time.

NOTATION

a	distance from $\xi = 0$ to inner surface of membrane, cm.
A_i	constant of integration; i = 1 to 22.
A	membrane area.
b	distance from x = 0 to outer surface of membrane, cm.
c	distance from x = 0 to outer surface of coating, cm.
C_n	concentration of agent in region n, mol/cm^3; C_n^0 initial concentration of agent in region n; n = 1, 2, 3, s.
D_{free}	free diffusivity of the agent in the liquid medium of the region being considered, cm^2/s.
D_n	effective diffusivity of the agent in region n, cm^2/s.
k	first order overall mass transfer coefficient between solid and reservoir phases, cm^3/s.
$m_{i,j}$	partition coefficient of agent between regions i and j (equation (8)). m is the partition coefficient of agent between a source or receiver region and the membrane.
s	Laplace variable.
t	time, s.
U_n	dimensionless concentration (equation (11)); n = 1, 2, 3, s.
V_n	volume of region n, cm^3, V_{source} is the volume of the source region; $V_{receiver}$ is the volume of the receiver region.
x	cartesian or cylindrical coordinate.

Greek letters

α_n outside surface area of region n, cm^2.

β dimensionless group used in model for dispersed phase in reservoir.

γ dimensionless group used in model for dispersed phase in reservoir.

δ dimensionless group used in model for dispersed phase in reservoir.

ε membrane porosity.

ϕ volume fraction occupied by tsolids dispersed in the reservoir.

θ dimensionless time.

τ membrane tortuosity.

τ_n time constant for mass transfer in a given region n, used in dimensional analysis; n = r (reservoir), m (membrane), c (coating), d (dissolution).

ξ dimensionless cartesian or cylindrical coordinate (equation (19)); ξ_1 and ξ_2 for regions 1 and 2 (equation (11)), and ξ_3 for the coating region.

ψ ratio of time constant for diffusion through reservoir to time constant for diffusion through membrane, equal to τ_r/τ_m.

ζ dimensionless group used in model for dispersed phase in reservoir, equal to $(1-\phi)/\phi m_{1,s}$.

Subscripts

1 region 1, reservoir.

2 region 2, membrane pore liquid.

3 region 3, coating.

s solid region

w surrounding water region (bath)

Table 8. Nomenclature Equivalents for the "Flat Membrane, Solution in Reservoir" Problem.

Text	Mathematica®	Fortran
$\frac{a}{b}\sqrt{s}$	arg1	C16_arg1
$\frac{a}{b}\sqrt{\frac{D_1}{D_2}s}$	arg2	C16_arg2
$\sqrt{\frac{D_1}{D_2}s}$	arg3	C16_arg3
C_1^0	--	R8_G_INIT_CONC
α_2	alpha	R8_area
a	--	R8_G_a
A_1	a1	--
A_2	a2	--
A_3	a3	C16_a3
A_4	a4	C16_a4
b	--	R8_G_b
D_1	d1	R8_G_d1
D_2	d2	R8_G_d2
$m_{1,2}$	xm	R8_G_xm
$m_{2,w}$	xm2	R8_G_xm2
s	s	C16_S
V_w	vw	R8_G_vw

APPENDIX 1

Flat Membrane, Solution in Reservoir

Presented below is the Mathematica® program used to determine expressions for the coefficients A_1, A_2, A_3, and A_4 for the "flat membrane: solution in reservoir" problem. It is followed by the Fortran code used to invert the solution numerically from the Laplace domain into the time domain. The reader is referred to Table 8 for nomenclature relationships between variables in the text of Chapter 2 and Mathematica® programs and Fortran code.

```
bcflat = Solve[{a1 Sqrt[s] == a2 Sqrt[s],

      d1 Sqrt[s] (a1 exp[arg1] - a2 exp[-arg1]) ==
      d2 arg3 (a3 exp[arg2] - a4 exp[-arg2]),

      1.0/s + a1 exp[arg1] + a2 exp[-arg1]==
      xm (a3 exp[arg2] + a4 exp[-arg2]),

      -alpha xm2 d2 arg3 b (a3 exp[arg3]  -
      a4 exp[-arg3]) ==
      vw d1 s (a3 exp[arg3] + a4 exp[-arg3])},

      {a1, a2, a3, a4}];

A1 =a1 /. bcflat;
A2 = a2 /. bcflat;
A3 = a3 /. bcflat;
A4 = a4 /. bcflat;

FortranForm[A1]
List(-1./(s*exp(-arg1) + s*exp(arg1)) -
     -   1.*s*xm*(-1.*d1*Sqrt(s)*exp(-arg1) +
     -       d1*Sqrt(s)*exp(arg1))*exp(arg2)*
     -     (alpha*xm2*arg3*b*d2*exp(-arg3) -
     -       1.*d1*s*vw*exp(-arg3))/
     -    ((s*exp(-arg1) + s*exp(arg1))*
     -      (-1.*(s*xm*
     -            (-1.*d1*Sqrt(s)*exp(-arg1) +
     -              d1*Sqrt(s)*exp(arg1))*
     -            exp(arg2) -
     -           1.*arg3*d2*
     -            (s*exp(-arg1) + s*exp(arg1))*
     -            exp(arg2))*
     -         (alpha*xm2*arg3*b*d2*exp(-arg3) -
     -           1.*d1*s*vw*exp(-arg3)) +
     -        (s*xm*
     -            (-1.*d1*Sqrt(s)*exp(-arg1) +
     -              d1*Sqrt(s)*exp(arg1))*
     -            exp(-arg2) +
     -           arg3*d2*
     -            (s*exp(-arg1) + s*exp(arg1))*
     -            exp(-arg2))*
     -         (-1.*alpha*xm2*arg3*b*d2*exp(arg3) -
     -           1.*d1*s*vw*exp(arg3)))) +
     -   1.*s*xm*(-1.*d1*Sqrt(s)*exp(-arg1) +
     -       d1*Sqrt(s)*exp(arg1))*exp(-arg2)*
     -     (-1.*alpha*xm2*arg3*b*d2*exp(arg3) -
     -       1.*d1*s*vw*exp(arg3))/
     -    ((s*exp(-arg1) + s*exp(arg1))*
     -      (-1.*(s*xm*
```

```
      -            (-1.*d1*Sqrt(s)*exp(-arg1) +
      -              d1*Sqrt(s)*exp(arg1))*
      -             exp(arg2) -
      -           1.*arg3*d2*
      -            (s*exp(-arg1) + s*exp(arg1))*
      -            exp(arg2))*
      -         (alpha*xm2*arg3*b*d2*exp(-arg3) -
      -           1.*d1*s*vw*exp(-arg3)) +
      -        (s*xm*
      -            (-1.*d1*Sqrt(s)*exp(-arg1) +
      -              d1*Sqrt(s)*exp(arg1))*
      -            exp(-arg2) +
      -           arg3*d2*
      -            (s*exp(-arg1) + s*exp(arg1))*
      -            exp(-arg2))*
      -         (-1.*alpha*xm2*arg3*b*d2*exp(arg3) -
      -           1.*d1*s*vw*exp(arg3)))))

FortranForm[A2]
List(-1./(s*exp(-arg1) + s*exp(arg1)) -
      -   1.*s*xm*(-1.*d1*Sqrt(s)*exp(-arg1) +
      -       d1*Sqrt(s)*exp(arg1))*exp(arg2)*
      -     (alpha*xm2*arg3*b*d2*exp(-arg3) -
      -       1.*d1*s*vw*exp(-arg3))/
      -    ((s*exp(-arg1) + s*exp(arg1))*
      -      (-1.*(s*xm*
      -            (-1.*d1*Sqrt(s)*exp(-arg1) +
      -              d1*Sqrt(s)*exp(arg1))*
      -            exp(arg2) -
      -           1.*arg3*d2*
      -            (s*exp(-arg1) + s*exp(arg1))*
      -            exp(arg2))*
      -         (alpha*xm2*arg3*b*d2*exp(-arg3) -
      -           1.*d1*s*vw*exp(-arg3)) +
      -        (s*xm*
      -            (-1.*d1*Sqrt(s)*exp(-arg1) +
      -              d1*Sqrt(s)*exp(arg1))*
      -            exp(-arg2) +
      -           arg3*d2*
      -            (s*exp(-arg1) + s*exp(arg1))*
      -            exp(-arg2))*
      -         (-1.*alpha*xm2*arg3*b*d2*exp(arg3) -
      -           1.*d1*s*vw*exp(arg3))) +
      -   1.*s*xm*(-1.*d1*Sqrt(s)*exp(-arg1) +
      -       d1*Sqrt(s)*exp(arg1))*exp(-arg2)*
      -     (-1.*alpha*xm2*arg3*b*d2*exp(arg3) -
      -       1.*d1*s*vw*exp(arg3))/
      -    ((s*exp(-arg1) + s*exp(arg1))*
      -      (-1.*(s*xm*
      -            (-1.*d1*Sqrt(s)*exp(-arg1) +
```

```
     -              d1*Sqrt(s)*exp(arg1))*
     -            exp(arg2) -
     -           1.*arg3*d2*
     -            (s*exp(-arg1) + s*exp(arg1))*
     -            exp(arg2))*
     -          (alpha*xm2*arg3*b*d2*exp(-arg3) -
     -            1.*d1*s*vw*exp(-arg3)) +
     -         (s*xm*
     -             (-1.*d1*Sqrt(s)*exp(-arg1) +
     -               d1*Sqrt(s)*exp(arg1))*
     -            exp(-arg2) +
     -           arg3*d2*
     -            (s*exp(-arg1) + s*exp(arg1))*
     -            exp(-arg2))*
     -          (-1.*alpha*xm2*arg3*b*d2*exp(arg3) -
     -            1.*d1*s*vw*exp(arg3)))))

FortranForm[A3]
List(-1.*(-1.*d1*Sqrt(s)*exp(-arg1) +
     -       d1*Sqrt(s)*exp(arg1))*
     -     (alpha*xm2*arg3*b*d2*exp(-arg3) -
     -       1.*d1*s*vw*exp(-arg3))/
     -    (-1.*(s*xm*
     -          (-1.*d1*Sqrt(s)*exp(-arg1) +
     -            d1*Sqrt(s)*exp(arg1))*exp(arg2)\
     -           - 1.*arg3*d2*
     -          (s*exp(-arg1) + s*exp(arg1))*
     -          exp(arg2))*
     -        (alpha*xm2*arg3*b*d2*exp(-arg3) -
     -          1.*d1*s*vw*exp(-arg3)) +
     -       (s*xm*(-1.*d1*Sqrt(s)*exp(-arg1) +
     -             d1*Sqrt(s)*exp(arg1))*exp(-arg2)
     -           + arg3*d2*
     -          (s*exp(-arg1) + s*exp(arg1))*
     -          exp(-arg2))*
     -        (-1.*alpha*xm2*arg3*b*d2*exp(arg3) -
     -          1.*d1*s*vw*exp(arg3))))

FortranForm[A4]
List(1.*(-1.*d1*Sqrt(s)*exp(-arg1) +
     -       d1*Sqrt(s)*exp(arg1))*
     -     (-1.*alpha*xm2*arg3*b*d2*exp(arg3) -
     -       1.*d1*s*vw*exp(arg3))/
     -    (-1.*(s*xm*
     -          (-1.*d1*Sqrt(s)*exp(-arg1) +
     -            d1*Sqrt(s)*exp(arg1))*exp(arg2)\
     -           - 1.*arg3*d2*
     -          (s*exp(-arg1) + s*exp(arg1))*
     -          exp(arg2))*
     -        (alpha*xm2*arg3*b*d2*exp(-arg3) -
```

```
-         1.*d1*s*vw*exp(-arg3)) +
-      (s*xm*(-1.*d1*Sqrt(s)*exp(-arg1) +
-            d1*Sqrt(s)*exp(arg1))*exp(-arg2)
-           + arg3*d2*
-          (s*exp(-arg1) + s*exp(arg1))*
-          exp(-arg2))*
-       (-1.*alpha*xm2*arg3*b*d2*exp(arg3) -
-       1.*d1*s*vw*exp(arg3))))
```

```
C ***********************************************************************
C * Program model_flat provides release profiles for the flat membrane-
C * solution in reservoir problem.
C *
C * Stephanie Farrell
C * October 28, 1995
C *
C * (C) Copyright 1995, 1996 Stephanie Farrell
C *
C * FILE: FLAT.FOR --> FLAT.EXE
C ***********************************************************************
      program MODEL_FLAT

C=======================================================================
C Function declarations
C=======================================================================
      real*8 REALTIME
      real*8 REALCONC
      external C16_FWALL
      external SUB_WALL
      external CALCULATION
      external DINLAP
      external REALTIME
      external REALCONC

C=======================================================================
C Variable declarations
C=======================================================================
      integer I_INDEX
      integer I_KMAX
      integer NOUT

      real*8 R8_ALPHA
      real*8 R8_RELERR

      real*8 R8_T(1)
      real*8 R8_FINV(1)

      complex*8 C8_arg1
      complex*8 C8_arg2
      complex*8 C8_arg3
      complex*8 C8_a1

C ***********************************************************************
C * Data input parameters
C ***********************************************************************
      character TITLE*80
      character VAR_NAME*30

      integer I_NUM_TIMES

      real*8 R8_TIME_INTERVAL
      real*8 R8_INIT_CONC
      real*8 R8_G_a
      real*8 R8_G_b
      real*8 R8_G_d1
      real*8 R8_G_d2
      real*8 R8_G_xm
      real*8 R8_G_vw
      real*8 R8_G_xm2
      real*8 R8_G_r
```

```
      character EOF*80

      common /IN_PARMS/
     &  R8_G_a,
     &  R8_G_b,
     &  R8_G_d1,
     &  R8_G_d2,
     &  R8_G_xm,
     &  R8_G_vw,
     &  R8_G_xm2,
     &  R8_G_r

C ***************************************************************************
C * Read in data
C ***************************************************************************
      read *, TITLE
      read *, VAR_NAME, I_NUM_TIMES
      read *, VAR_NAME, R8_TIME_INTERVAL
      read *, VAR_NAME, R8_INIT_CONC

      read *, VAR_NAME, R8_G_a
      read *, VAR_NAME, R8_G_b
      read *, VAR_NAME, R8_G_d1
      read *, VAR_NAME, R8_G_d2
      read *, VAR_NAME, R8_G_xm
      read *, VAR_NAME, R8_G_vw
      read *, VAR_NAME, R8_G_xm2
      read *, VAR_NAME, R8_G_r
      read *, EOF

      if (EOF .NE. 'EOF') then
         print *,'Input file format is incorrect. Aborting.'
         goto 1000
      endif

      print *, TITLE
      print *, 'I_NUM_TIMES      ,', I_NUM_TIMES
      print *, 'R8_TIME_INTERVAL ,', R8_TIME_INTERVAL
      print *, 'R8_INIT_CONC     ,', R8_INIT_CONC
      print *, 'R8_G_a           ,', R8_G_a
      print *, 'R8_G_b           ,', R8_G_b
      print *, 'R8_G_d1          ,', R8_G_d1
      print *, 'R8_G_d2          ,', R8_G_d2
      print *, 'R8_G_xm          ,', R8_G_xm
      print *, 'R8_G_vw          ,', R8_G_vw
      print *, 'R8_G_xm2         ,', R8_G_xm2
      print *, 'R8_G_r           ,', R8_G_r

C=========================================================================
C   User-supplied C8_arguments for the IMSL subroutine DINLAP
C=========================================================================
      R8_ALPHA = 0
      I_KMAX = 10000
      R8_RELERR = 5.0E-5

C=========================================================================
C   Evaluate the solution for the wall section.
C=========================================================================
      print *,'THETA,U,REAL TIME,REAL CONC.'
      do I_INDEX = I_NUM_TIMES, 1, -1
```

```
            R8_T(1) =  R8_TIME_INTERVAL * DFLOAT(I_INDEX)
            call DINLAP (C16_FWALL,
     &                   1,
     &                   R8_T,
     &                   R8_ALPHA,
     &                   R8_RELERR,
     &                   I_KMAX,
     &                   R8_FINV)

            print '(E9.4, A, E9.4, A, F15.2, A, E15.4)',
     &             R8_T(1), ',',
     &             R8_FINV(1), ',',
     &             REALTIME(R8_T(1), R8_G_b, R8_G_d1), ',',
     &             REALCONC(R8_FINV(1), R8_INIT_CONC)
      end do

 1000 continue
      end

C*********************************************************************
C User-supplied function to which the inverse Laplace transform
C will be computed
C*********************************************************************
      complex*16 FUNCTION C16_FWALL(C16_S)

      intrinsic CSQRT
      intrinsic CEXP

      complex*16 C16_argWall
      complex*16 C16_Temp

      complex*16 C16_S
      complex*16 C16_a3
      complex*16 C16_a4

      real*8 R8_d1
      real*8 R8_d2

      call SUB_WALL(C16_S, R8_d1, R8_d2, C16_a3, C16_a4)

      C16_Temp = C16_S * R8_d1 / R8_d2

      C16_argWall = CDSQRT(C16_Temp)

      C16_FWALL =  C16_a3 * CDEXP(C16_argWall) + C16_a4 *
     &    CDEXP(-C16_argWall)

      return
      end

C **********************************************************************
C * Subroutine SUB_WALL evaluates a3 and a4 which are used in
C * FWALL.
C **********************************************************************
      subroutine SUB_WALL(C16_S,
     &                    R8_d1,
     &                    R8_d2,
     &                    C16_a3,
     &                    C16_a4)

      complex*16 C16_S
```

```
      complex*16 C16_arg1
      complex*16 C16_arg2
      complex*16 C16_arg3
      complex*16 C16_a4
      complex*16 C16_a3

      real*8 R8_a
      real*8 R8_b
      real*8 R8_d1
      real*8 R8_d2
      real*8 R8_xm
      real*8 R8_vw
      real*8 R8_xm2
      real*8 R8_r
      real*8 R8_area
      real*8 R8_vw_R8_xm2

C=====================================================================
C Call calculation to calculate everything necessary for
C evaluation of C16_a3 and C16_a4
C=====================================================================
      call CALCULATION (C16_S,
     &                  C16_arg1,
     &                  C16_arg2,
     &                  C16_arg3,
     &                  R8_d1,
     &                  R8_d2,
     &                  R8_vw,
     &                  R8_vw_R8_xm2,
     &                  R8_a,
     &                  R8_b,
     &                  R8_xm,
     &                  R8_area)

C=====================================================================
C Compute C16_a3
C=====================================================================
      C16_a3 = (-1. * (-1. * R8_d1 * CDSQRT(C16_S) *
     &    CDEXP(-C16_arg1) +
     &      R8_d1 * CDSQRT(C16_S) * CDEXP(C16_arg1)) *
     &    (R8_area * C16_arg3 * R8_b * R8_d2 * CDEXP(-C16_arg3) -
     &      1. * R8_d1 * C16_S * R8_vw_R8_xm2 * CDEXP(-C16_arg3)) /
     &   (-1. * (C16_S * R8_xm * (-1. * R8_d1 * CDSQRT(C16_S) *
     &   CDEXP(-C16_arg1) +
     &           R8_d1 * CDSQRT(C16_S) * CDEXP(C16_arg1)) *
     &   CDEXP(C16_arg2) -
     &        1. * C16_arg3 * R8_d2 * (C16_S * CDEXP(-C16_arg1) +
     &   C16_S * CDEXP(C16_arg1)) *
     &    CDEXP(C16_arg2)) * (R8_area * C16_arg3 * R8_b * R8_d2 *
     &   CDEXP(-C16_arg3) -
     &        1. * R8_d1 * C16_S * R8_vw_R8_xm2 * CDEXP(-C16_arg3))
     &   + (C16_S * R8_xm * (-1. * R8_d1 * CDSQRT(C16_S) *
     &   CDEXP(-C16_arg1) +
     &           R8_d1 * CDSQRT(C16_S) * CDEXP(C16_arg1)) *
     &   CDEXP(-C16_arg2) +
     &        C16_arg3 * R8_d2 * (C16_S * CDEXP(-C16_arg1) + C16_S *
     &   CDEXP(C16_arg1)) *
     &      CDEXP(-C16_arg2)) * (-1. * R8_area * C16_arg3 * R8_b * R8_d2
     &   * CDEXP(C16_arg3) -
     &        1. * R8_d1 * C16_S * R8_vw_R8_xm2 * CDEXP(C16_arg3))))
```

```
C=======================================================================
C Compute C16_a4
C=======================================================================
      C16_a4 = (1. * (-1. * R8_d1 * CDSQRT(C16_S) * CDEXP(-C16_arg1) +
     &      R8_d1 * CDSQRT(C16_S) * CDEXP(C16_arg1)) *
     &    (-1. * R8_area * C16_arg3 * R8_b * R8_d2 * CDEXP(C16_arg3) -
     &      1. * R8_d1 * C16_S * R8_vw_R8_xm2 * CDEXP(C16_arg3)) /
     &   (-1. * (C16_S * R8_xm * (-1. * R8_d1 * CDSQRT(C16_S) *
     &       CDEXP(-C16_arg1) +
     &           R8_d1 * CDSQRT(C16_S) * CDEXP(C16_arg1)) *
     &      CDEXP(C16_arg2) -
     &        1. * C16_arg3 * R8_d2 * (C16_S * CDEXP(-C16_arg1) + C16_S
     &    * CDEXP(C16_arg1)) *
     &         CDEXP(C16_arg2)) * (R8_area * C16_arg3 * R8_b * R8_d2 *
     &    CDEXP(-C16_arg3) -
     &        1. * R8_d1 * C16_S * R8_vw_R8_xm2 * CDEXP(-C16_arg3)) +
     &     (C16_S * R8_xm * (-1. * R8_d1 * CDSQRT(C16_S) *
     &    CDEXP(-C16_arg1) +
     &           R8_d1 * CDSQRT(C16_S) * CDEXP(C16_arg1)) *
     &    CDEXP(-C16_arg2) +
     &        C16_arg3 * R8_d2 * (C16_S * CDEXP(-C16_arg1) + C16_S *
     &    CDEXP(C16_arg1)) *
     &      CDEXP(-C16_arg2)) * (-1. * R8_area * C16_arg3 * R8_b * R8_d2
     &    * CDEXP(C16_arg3) -
     &        1. * R8_d1 * C16_S * R8_vw_R8_xm2 * CDEXP(C16_arg3))))

      end

C**********************************************************************
C Subroutine CALCULATION computes everything necessary for evaluation
C of C16_a1, a3, and a4
C**********************************************************************
      subroutine CALCULATION(C16_S,
     &                       C16_arg1,
     &                       C16_arg2,
     &                       C16_arg3,
     &                       R8_d1,
     &                       R8_d2,
     &                       R8_vw,
     &                       R8_vw_R8_xm2,
     &                       R8_a,
     &                       R8_b,
     &                       R8_xm,
     &                       R8_area)

C=======================================================================
C Constants
C=======================================================================
      real*8 R8_PI
      parameter (R8_PI = 3.14159265359)

C=======================================================================
C Commons
C=======================================================================
      real*8 R8_G_a
      real*8 R8_G_b
      real*8 R8_G_d1
      real*8 R8_G_d2
      real*8 R8_G_xm
      real*8 R8_G_vw
      real*8 R8_G_xm2
```

```
      real*8 R8_G_r

      common /IN_PARMS/
     &  R8_G_a,
     &  R8_G_b,
     &  R8_G_d1,
     &  R8_G_d2,
     &  R8_G_xm,
     &  R8_G_vw,
     &  R8_G_xm2,
     &  R8_G_r

C=======================================================================
C Variables
C=======================================================================
      complex*16 C16_arg1
      complex*16 C16_arg2
      complex*16 C16_arg3
      complex*16 C16_a1
      complex*16 C16_S
      complex*16 C16_Temp

      real*8 R8_a
      real*8 R8_b
      real*8 R8_d1
      real*8 R8_d2
      real*8 R8_xm
      real*8 R8_vw
      real*8 R8_xm2
      real*8 R8_r
      real*8 R8_area
      real*8 R8_vw_R8_xm2

C=======================================================================
C Computations
C=======================================================================
      R8_a           = R8_G_a          ! Inside distance
      R8_b           = R8_G_b          ! Outside distance
      R8_d1          = R8_G_d1         ! Diffusivity in reservoir
      R8_d2          = R8_G_d2         ! Effective membrane diffusivity
      R8_xm          = R8_G_xm         ! Partition coefficient at A
      R8_vw          = R8_G_vw         ! External aqueous phase volume
      R8_xm2         = R8_G_xm2
      R8_r           = R8_G_r          ! Radius of membrane circle
      R8_area        = R8_r**2 * R8_PI ! Outside area
      R8_vw_R8_xm2   = R8_vw / R8_xm2  ! Ratio of R8_vw to R8_xm2

C=======================================================================
C C16_arguments of the Bessel functions that appear in the expressions
C for C16_a1, a3 and a4.
C=======================================================================
      C16_arg1 = R8_a * CDSQRT(C16_S) / R8_b

      C16_Temp = C16_S * R8_d1 / R8_d2
      C16_arg2 = R8_a * CDSQRT(C16_Temp) / R8_b

      C16_Temp = C16_S * R8_d1 / R8_d2
      C16_arg3 = CDSQRT(C16_Temp)

      end
```

```
C ***********************************************************************
C * Subroutine REALTIME converts to real time.
C ***********************************************************************
      real*8 function REALTIME(R8_THETA, R8_b, R8_d1)

      real*8 R8_THETA
      real*8 R8_b
      real*8 R8_d1

      REALTIME = R8_THETA * (R8_b ** 2) / R8_d1

      return
      end

C ***********************************************************************
C * Subroutine REALCONC converts to real concentration.
C ***********************************************************************
      real*8 function REALCONC(R8_U, R8_INIT_CONC)

      real*8 R8_U
      real*8 R8_INIT_CONC

      REALCONC = R8_U * R8_INIT_CONC

      return
      end
C***********************************************************************
C* EOF
C***********************************************************************
```

Hollow Fiber: Solution in Reservoir

Presented below is the Mathematica® program used to determine expressions for the coefficients A_5, A_7, and A_8 for the "hollow fiber: solution in reservoir" problem. It is followed by the Fortran code used to invert the solution numerically from the Laplace domain into the time domain. The reader is referred to Table 9 for nomenclature relationships between variables in the text of Chapter 2 and Mathematica® programs and Fortran code.

Table 9. Nomenclature Equivalents for the "Hollow Fiber: Solution in Reservoir" Problem.

Text	Mathematica®	FORTRAN
$\frac{a}{b}\sqrt{s}$	arg1	C16_arg1
$\frac{a}{b}\sqrt{\frac{D_1}{D_2}s}$	arg2	C16_arg2
$\sqrt{\frac{D_1}{D_2}s}$	arg3	C16_arg3
$\sqrt{\frac{D_1}{D_2}}$	X	R8_X
$\frac{V_w\sqrt{s}}{\alpha_2 b m_{2,w}}$	Z	C16_Z
a		R8_G_a
A_5	A1	C16_a1
A_7	A3	C16_a3
A_8	A4	C16_a4
b		R8_G_b
D_1		R8_G_d1
D_2		R8_G_d2
I_0	I0	C16_I0
I_1	I1	C16_I1
K_0	K0	C16_K0
K_1	K1	C16_K1
$m_{1,2}$		R8_G_xm
$m_{2,w}$		R8_G_xm2
s	s	R8_G_s
V_w		R8_G_vw

```
Bcmhf = Solve[{1.0/s + A1 I0arg1 ==

     xm (A3 I0arg2 + A4 K0arg2),
     X A1 I1arg1 ==
     A3 I1arg2 - A4 K1arg2,
     X Z (A3 I0arg3 + A4 K0arg3) ==
     -(A3 I1arg3 - A4 K1arg3)},{A1, A3, A4}];

a1 = A1 /. Bcmhf;
a3 = A3 /. Bcmhf;
a4 = A4 /. Bcmhf;

FortranForm[a1]
List(-1./(I0arg1*s) +
     -    1.*I1arg1*K0arg2*X*xm*(I1arg3 +
     -    I0arg3*X*Z)/
     -    (I0arg1*((I0arg1*K1arg2*s +
     -    I1arg1*K0arg2*s*X*xm)*
     -    (I1arg3 + I0arg3*X*Z) -
     -    1.*(-1.*I0arg1*I1arg2*s +
     -    I0arg2*I1arg1*s*X*xm)*
     -    (-1.*K1arg3 + K0arg3*X*Z))) -
     -    1.*I0arg2*I1arg1*X*xm*
     -    (-1.*K1arg3 + K0arg3*X*Z)/
     -    (I0arg1*((I0arg1*K1arg2*s +
     -    I1arg1*K0arg2*s*X*xm)*
     -    (I1arg3 + I0arg3*X*Z) -
     -    1.*(-1.*I0arg1*I1arg2*s +
     -    I0arg2*I1arg1*s*X*xm)*
     -    (-1.*K1arg3 + K0arg3*X*Z))))

FortranForm[a3]
List(-1.*I1arg1*X*(-1.*K1arg3 + K0arg3*X*Z)/
     -    ((I0arg1*K1arg2*s + I1arg1*K0arg2*s*X*xm)*
     -    (I1arg3 + I0arg3*X*Z) -
     -    1.*(-1.*I0arg1*I1arg2*s +
     -    I0arg2*I1arg1*s*X*xm)*
     -    (-1.*K1arg3 + K0arg3*X*Z)))

FortranForm[a4]
List(1.*I1arg1*X*(I1arg3 + I0arg3*X*Z)/
     -    ((I0arg1*K1arg2*s + I1arg1*K0arg2*s*X*xm)*
     -    (I1arg3 + I0arg3*X*Z) -
     -    1.*(-1.*I0arg1*I1arg2*s +
     -    I0arg2*I1arg1*s*X*xm)*
     -    (-1.*K1arg3 + K0arg3*X*Z)))
```

```
C ***********************************************************************
C * Program model_mhf provides release profiles for the holllow fiber-
C * solution in reservoir problem.
C *
C * Stephanie Farrell
C * August 1995
C *
C * © Copyright 1995, 1996 Stephanie Farrell
C *
C * FILE: MHF.FOR → MHF.EXE
C ***********************************************************************
      program MODEL_MHF

C ***********************************************************************
C * Function definitions
C ***********************************************************************
      real*8 R8_REALTIME
      real*8 R8_REALCONC
      complex*16 C16_F_LUMEN
      complex*16 C16_F_WALL

      external R8_REALTIME
      external R8_REALCONC
      external C16_F_LUMEN
      external C16_F_WALL
      external LUMEN
      external WALL
      external CALCULATION
      external DINLAP
      external BESK0
      external BESI0
      external BESI1
      external BESK1

C ***********************************************************************
C * Parameters relating to independent variables (radius and time)
C ***********************************************************************
      real*8 R8_MIN_RADIUS
      parameter (R8_MIN_RADIUS = 0.0)

      real*8 R8_MAX_RADIUS
      parameter (R8_MAX_RADIUS = 1.0)

C ***********************************************************************
C * Main program.
C ***********************************************************************
      integer I_INDEX
      integer I_KMAX
      integer I_NOUT
      integer I_LUMEN
      integer I_WALL

      real*8 R8_ALPHA
      real*8 R8_EXP
      real*8 R8_FLOAT
      real*8 R8_RELERR

      real*8 R8_T(1)
      real*8 R8_FINV(1)

      real*8 R8_BOUNDARY
```

```
      real*8 R8_LUMEN_RADIUS_INCREMENT
      real*8 R8_WALL_RADIUS_INCREMENT

      real*8 R8_RADIUS
      common R8_RADIUS

C *********************************************************************
C * Data input parameters
C *********************************************************************
      character C_TITLE*80
      character C_VAR_NAME*30
      character C_USE_DISTANCE*1

      integer I_NUM_TIMES
      integer I_NUM_LUMEN_GRIDS
      integer I_NUM_WALL_GRIDS

      real*8 R8_TIME_INTERVAL
      real*8 R8_INIT_CONC
      real*8 R8_G_a
      real*8 R8_G_b
      real*8 R8_G_d1
      real*8 R8_G_d2
      real*8 R8_G_xm
      real*8 R8_G_vw
      real*8 R8_G_xm2

      character C_EOF*80

      common /IN_PARMS/
     &  R8_G_a,
     &  R8_G_b,
     &  R8_G_d1,
     &  R8_G_d2,
     &  R8_G_xm,
     &  R8_G_vw,
     &  R8_G_xm2

C *********************************************************************
C * Read in data
C *********************************************************************
      read *, C_TITLE
      read *, C_VAR_NAME, C_USE_DISTANCE

      read *, C_VAR_NAME, I_NUM_TIMES
      read *, C_VAR_NAME, I_NUM_LUMEN_GRIDS
      read *, C_VAR_NAME, I_NUM_WALL_GRIDS
      read *, C_VAR_NAME, R8_TIME_INTERVAL
      read *, C_VAR_NAME, R8_INIT_CONC

      read *, C_VAR_NAME, R8_G_a
      read *, C_VAR_NAME, R8_G_b
      read *, C_VAR_NAME, R8_G_d1
      read *, C_VAR_NAME, R8_G_d2
      read *, C_VAR_NAME, R8_G_xm
      read *, C_VAR_NAME, R8_G_vw
      read *, C_VAR_NAME, R8_G_xm2
      read *, C_EOF

      if (C_EOF .NE. 'EOF') then
         print *,'Input file format is incorrect. Aborting.'
```

```
         Goto 1000
      endif

      print *, C_TITLE
      print *, 'C_USE_DISTANCE    ,', C_USE_DISTANCE
      print *, 'I_NUM_TIMES       ,', I_NUM_TIMES
      print *, 'I_NUM_LUMEN_GRIDS,', I_NUM_LUMEN_GRIDS
      print *, 'I_NUM_WALL_GRIDS ,', I_NUM_WALL_GRIDS
      print *, 'R8_TIME_INTERVAL ,', R8_TIME_INTERVAL
      print *, 'R8_INIT_CONC      ,', R8_INIT_CONC
      print *, 'R8_G_a            ,', R8_G_a
      print *, 'R8_G_b            ,', R8_G_b
      print *, 'R8_G_d1           ,', R8_G_d1
      print *, 'R8_G_d2           ,', R8_G_d2
      print *, 'R8_G_xm           ,', R8_G_xm
      print *, 'R8_G_vw           ,', R8_G_vw
      print *, 'R8_G_xm2          ,', R8_G_xm2

C ***********************************************************************
C *   User-supplied arguments for the IMSL routine DINLAP
C ***********************************************************************
      R8_ALPHA = 0
      R8_RELERR = 5.0E-5
      I_KMAX = 10000

C ***********************************************************************
C *   Evaluate the solution for the lumen section.
C ***********************************************************************
      if (C_USE_DISTANCE .EQ. 'Y') then
         do I_LUMEN = 1, I_NUM_LUMEN_GRIDS

            R8_BOUNDARY = R8_G_a / R8_G_b
            R8_LUMEN_RADIUS_INCREMENT = (R8_BOUNDARY - R8_MIN_RADIUS)  /
     &         FLOAT(I_NUM_LUMEN_GRIDS - 1)
            R8_RADIUS = R8_MIN_RADIUS + FLOAT(I_LUMEN - 1) *
     &         R8_LUMEN_RADIUS_INCREMENT

            print *, 'Inside the lumen: RADIUS = ', R8_RADIUS
            print *,'THETA,U,REAL TIME,REAL CONC.'

            Do I_INDEX = I_NUM_TIMES, 1, -1
               R8_T(1) =  R8_TIME_INTERVAL * Float(I_INDEX)
               call DINLAP (C16_F_LUMEN,
     &                      1,
     &                      R8_T,
     &                      R8_ALPHA,
     &                      R8_RELERR,
     &                      I_KMAX,
     &                      R8_FINV)

               print '(E15.4, A, E15.4, A, F15.2, A, E15.4)',
     &               R8_T(1), ',',
     &               R8_FINV(1), ',',
     &               R8_REALTIME(R8_T(1), R8_G_b, R8_G_d1), ',',
     &               R8_REALCONC(R8_FINV(1), R8_INIT_CONC)
            end do

         end do
       end if

C ***********************************************************************
```

```
C *   Evaluate the solution for the wall section.
C ***********************************************************************

      do I_WALL = 1, I_NUM_WALL_GRIDS
         if (C_USE_DISTANCE .EQ. 'Y') then
            R8_BOUNDARY = R8_G_a / R8_G_b
            R8_WALL_RADIUS_INCREMENT = (R8_MAX_RADIUS - R8_BOUNDARY)  /
     &          FLOAT(I_NUM_WALL_GRIDS - 1)
            R8_RADIUS = R8_BOUNDARY + FLOAT(I_WALL - 1) *
     &          R8_WALL_RADIUS_INCREMENT
         else
            R8_RADIUS = 1.0
         end if

         print *, 'Inside the wall: RADIUS = ', R8_RADIUS
         print *,'THETA,U,REAL TIME,REAL CONC.'

         Do I_INDEX = I_NUM_TIMES, 1, -1
            R8_T(1) =  R8_TIME_INTERVAL * Float(I_INDEX)
            call DINLAP (C16_F_WALL,
     &                   1,
     &                   R8_T,
     &                   R8_ALPHA,
     &                   R8_RELERR,
     &                   I_KMAX,
     &                   R8_FINV)

             print '(E15.4, A, E15.4, A, F15.2, A, E15.4)',
     &              R8_T(1), ',',
     &              R8_FINV(1), ',',
     &              R8_REALTIME(R8_T(1), R8_G_b, R8_G_d1), ',',
     &              R8_REALCONC(R8_FINV(1), R8_INIT_CONC)
         end do

      end do

 1000 continue
      end


C ***********************************************************************
C * User-supplied function to which the inverse Laplace transform
C * will be computed
C ***********************************************************************
      complex*16 function C16_F_WALL(C16_s)
      common R8_RADIUS

      real*8 R8_RADIUS

      real*8 R8_d1
      real*8 R8_d2

      complex*16 C16_I0argWall
      complex*16 C16_K0argWall
      complex*16 C16_argWall

      complex*16 C16_s
      complex*16 C16_a1
      complex*16 C16_a3
      complex*16 C16_a4
```

```
      intrinsic CDSQRT

      call WALL(C16_s, R8_d1, R8_d2, C16_a3, C16_a4)

      C16_argWall = R8_RADIUS * CDSQRT(C16_s * R8_d1 / R8_d2)

      call BESI0(C16_argWall, C16_I0argWall)
      call BESK0(C16_argWall, C16_K0argWall)

      C16_F_WALL = (C16_a3 * C16_I0argWall + C16_a4 * C16_K0argWall)

      return
      end

C ***********************************************************************
C * User-supplied function to which the inverse Laplace transform
C * will be computed
C ***********************************************************************
      complex*16 function C16_F_LUMEN(C16_s)
      common R8_RADIUS
      intrinsic CDSQRT

      real*8 R8_RADIUS

      complex*16 C16_s
      complex*16 C16_a1
      complex*16 C16_argLumen
      complex*16 C16_I0argLumen

      call LUMEN(C16_s, C16_a1)

      C16_argLumen = R8_RADIUS * CDSQRT(C16_s)
      call BESI0(C16_argLumen, C16_I0argLumen)

      C16_F_LUMEN = C16_a1 * C16_I0argLumen
      return
      end

C ***********************************************************************
C * BESI0 computes Bessel(I,0)
C ***********************************************************************
      subroutine BESI0 (C16_X, C16_I0)

      complex*16 C16_X
      complex*16 C16_I0

      C16_I0 = 1.0 + C16_X**2 / 4.0 + C16_X**4 / 64.0 +
     &   C16_X**6 / 2304.0 + C16_X**8 / 147456.0 +
     &   C16_X**10 / 14745600.0 + C16_X**12 / 2123366400.0 +
     &   C16_X**14 / 416179814400.0 +
     &   C16_X**16 / 106542032486400.0 +
     &   C16_X**18 / 34519618525593600.0 +
     &   C16_X**20 / 13807847410237440000.0

      return
      end

C ***********************************************************************
C * BESI1 computes Bessel(I,1)
C ***********************************************************************
      subroutine BESI1 (C16_X, C16_I1)
```

```
      complex*16 C16_X
      complex*16 C16_I1

      C16_I1 = C16_X/2.0 + C16_X**3/16.0 +
     &     C16_X**5/384.0 + C16_X**7/18432.0 +
     &     C16_X**9/1474560.0 + C16_X**11/176947200.0 +
     &     C16_X**13/29727129600.0 +
     &     C16_X**15/6658877030400.0 +
     &     C16_X**17/1917756584755200.0 +
     &     C16_X**19/690392370511872000.0

      return
      end

C ***********************************************************************
C * BESK0 computes Bessel(K,0)
C ***********************************************************************
      subroutine BESK0 (C16_X, C16_K0)

      complex*16 C16_X
      complex*16 C16_XL
      complex*16 C16_K0

      intrinsic CDLOG
      intrinsic CDSQRT

      C16_XL = LOG(1.0/2.0) + CDLOG(C16_X)

      C16_K0 = -0.577215 - Log(1.0/2.0) +
     &    C16_X**20*((7381.0/1260.0 - 2*0.577215)/27615694820474880000.0
     &    + (-C16_XL)/13807847410237440000.0) +
     &    C16_X**18*((7129.0/1260.0 - 2*0.577215)/69039237051187200.0 +
     &    (-C16_XL)/3451961852559360.0) +
     &    C16_X**16*((761.0/140.0 - 2*0.577215)/213084064972800.0 +
     &    (-C16_XL)/106542032486400.0) +
     &    C16_X**14*((363.0/70.0 - 2*0.577215)/832359628800.0 +
     &    (-C16_XL)/416179814400.0) +
     &    C16_X**12*((49.0/10.0 - 2*0.577215)/4246732800.0 +
     &    (-C16_XL)/2123366400.0) +
     &    C16_X**10*((137.0/30.0 - 2*0.577215)/29491200.0 +
     &    (-C16_XL)/14745600.0) +
     &    C16_X**8*((25.0/6.0 - 2*0.577215)/294912.0 +
     &    (-C16_XL)/147456.0) +
     &    C16_X**6*((11.0/3.0 - 2*0.577215)/4608.0 + (-C16_XL)/2304.0) +
     &    C16_X**4*((3.0 - 2*0.577215)/128.0 + (-C16_XL)/64.0) +
     &    C16_X**2*((2.0 - 2*0.577215)/8.0 + (-C16_XL)/4.0) -
     &    CDLOG(C16_X)

      return
      end

C ***********************************************************************
C * BESK1 computes Bessel(K,1)
C ***********************************************************************
      subroutine BESK1(C16_X, C16_K1)

      complex*16 C16_X
      complex*16 C16_XL
      complex*16 C16_K1
```

```
      intrinsic CDLOG
      intrinsic CDSQRT

      C16_XL = LOG(1.0/2.0) + CDLOG(C16_X)

      C16_K1 = 1.0/C16_X + C16_X**17*((-6989.0/1260.0 + 2*0.577215)/
     &  3835513169510400.0 + C16_XL/1917756584755200.0) +
     &  C16_X**15*((-1487.0/280.0 + 2*0.577215)/13317754060800.0 +
     &  C16_XL/6658877030400.0) +
     &  C16_X**13*((-353.0/70.0 + 2*0.577215)/59454259200.0 +
     &  C16_XL/29727129600.0) +
     &  C16_X**11*((-71.0/15.0 + 2*0.577215)/353894400.0 +
     &  C16_XL/176947200.0) +
     &  C16_X**9*((-131.0/30.0 + 2*0.577215)/2949120.0 +
     &  C16_XL/1474560.0) +
     &  C16_X**7*((-47.0/12.0 + 2*0.577215)/36864.0 + C16_XL/18432.0) +
     &  C16_X**5*((-10.0/3.0 + 2*0.577215)/768.0 + C16_XL/384.0) +
     &  C16_X**3*((-5.0/2.0 + 2*0.577215)/32.0 + C16_XL/16.0) +
     &  C16_X*((-1.0 + 2*0.577215)/4.0 +  C16_XL/2.0)

      return
      end

C ***********************************************************************
C * WALL evaluates a3 and a4 which are used in C16_F_WALL
C ***********************************************************************
      subroutine WALL (C16_s, R8_d1, R8_d2, C16_a3, C16_a4)

      complex*16 C16_s
      complex*16 C16_I0arg1
      complex*16 C16_I0arg2
      complex*16 C16_I0arg3
      complex*16 C16_I1arg1
      complex*16 C16_I1arg2
      complex*16 C16_I1arg3
      complex*16 C16_K0arg2
      complex*16 C16_K0arg3
      complex*16 C16_K1arg2
      complex*16 C16_K1arg3
      complex*16 C16_arg1
      complex*16 C16_arg2
      complex*16 C16_arg3
      complex*16 C16_Z
      complex*16 C16_a3
      complex*16 C16_a4

      real*8 R8_D1
      real*8 R8_D2
      real*8 R8_VW
      real*8 R8_V1
      real*8 R8_A
      real*8 R8_B
      real*8 R8_X
      real*8 R8_XM
      real*8 R8_XM2
      real*8 R8_AREA

C ***********************************************************************
C * Call CALCULATION to calculate everything necessary for
C * evaluation of a3 and a4.
C ***********************************************************************
```

```
      call CALCULATION (C16_s,
     &                  C16_I0arg1,
     &                  C16_I0arg2,
     &                  C16_I0arg3,
     &                  C16_I1arg1,
     &                  C16_I1arg2,
     &                  C16_I1arg3,
     &                  C16_K0arg2,
     &                  C16_K0arg3,
     &                  C16_K1arg2,
     &                  C16_K1arg3,
     &                  C16_arg1,
     &                  C16_arg2,
     &                  C16_arg3,
     &                  R8_d1,
     &                  R8_d2,
     &                  R8_vw,
     &                  R8_v1,
     &                  R8_a,
     &                  R8_b,
     &                  R8_X,
     &                  C16_Z,
     &                  R8_xm,
     &                  R8_xm2,
     &                  R8_area)

C ***************************************************************************
C * Compute a3
C ***************************************************************************
          C16_a3 = (-1.0 * C16_I1arg1 * R8_X * (-1.0 * C16_K1arg3 +
     &    C16_K0arg3 * R8_X * C16_Z)/
     &    ((C16_I0arg1 * C16_K1arg2 * C16_s + C16_I1arg1 * C16_K0arg2 *
     &    C16_s * R8_X * R8_Xm)*
     &    (C16_I1arg3 + C16_I0arg3 * R8_X * C16_Z) -
     &    1.0 * (-1.0 * C16_I0arg1 * C16_I1arg2 * C16_s +
     &    C16_I0arg2 * C16_I1arg1 * C16_s * R8_X * R8_Xm) *
     &    (-1.0 * C16_K1arg3 + C16_K0arg3 * R8_X * C16_Z)))

C ***************************************************************************
C * Compute a4
C ***************************************************************************
          C16_a4 = (1.0 * C16_I1arg1 * R8_X * (C16_I1arg3 + C16_I0arg3 *
     &    R8_X * C16_Z)/
     &    ((C16_I0arg1 * C16_K1arg2 * C16_s + C16_I1arg1 * C16_K0arg2 *
     &    C16_s * R8_X * R8_Xm) *
     &    (C16_I1arg3 + C16_I0arg3 * R8_X * C16_Z) -
     &    1.0 * (-1.0 * C16_I0arg1 * C16_I1arg2 * C16_s +
     &    C16_I0arg2 * C16_I1arg1 * C16_s * R8_X * R8_Xm) *
     &    (-1.0 * C16_K1arg3 + C16_K0arg3 * R8_X * C16_Z)))

      end

C ***************************************************************************
C * LUMEN evaluates a1 which is used in C16_F_LUMEN
C ***************************************************************************
      subroutine LUMEN (C16_s, C16_a1)

      complex*16 C16_s
      complex*16 C16_I0arg1
      complex*16 C16_I0arg2
```

```
      complex*16 C16_I0arg3
      complex*16 C16_I1arg1
      complex*16 C16_I1arg2
      complex*16 C16_I1arg3
      complex*16 C16_K0arg2
      complex*16 C16_K0arg3
      complex*16 C16_K1arg2
      complex*16 C16_K1arg3
      complex*16 C16_arg1
      complex*16 C16_arg2
      complex*16 C16_arg3
      complex*16 C16_Z
      complex*16 C16_a1

      real*8 R8_D1
      real*8 R8_D2
      real*8 R8_VW
      real*8 R8_V1
      real*8 R8_A
      real*8 R8_B
      real*8 R8_X
      real*8 R8_XM
      real*8 R8_XM2
      real*8 R8_AREA

C ***********************************************************************
C * Call CALCULATION to calculate everything necessary for
C * evaluation of a3 and a4
C ***********************************************************************
      call CALCULATION (C16_s,
     &                  C16_I0arg1,
     &                  C16_I0arg2,
     &                  C16_I0arg3,
     &                  C16_I1arg1,
     &                  C16_I1arg2,
     &                  C16_I1arg3,
     &                  C16_K0arg2,
     &                  C16_K0arg3,
     &                  C16_K1arg2,
     &                  C16_K1arg3,
     &                  C16_arg1,
     &                  C16_arg2,
     &                  C16_arg3,
     &                  R8_d1,
     &                  R8_d2,
     &                  R8_vw,
     &                  R8_v1,
     &                  R8_a,
     &                  R8_b,
     &                  R8_X,
     &                  C16_Z,
     &                  R8_xm,
     &                  R8_xm2,
     &                  R8_area)


C ***********************************************************************
C * Compute a1
C ***********************************************************************
         C16_a1 = (-1.0 / (C16_I0arg1 * C16_s) +
     &   1.0 * C16_I1arg1 * C16_K0arg2 * R8_X * R8_Xm * (C16_I1arg3 +
```

```
     &     C16_I0arg3 * R8_X * C16_Z) /
     &    (C16_I0arg1 *((C16_I0arg1 * C16_K1arg2 * C16_s +
     &    C16_I1arg1  *  C16_K0arg2  *  C16_s * R8_X * R8_Xm) *
     &    (C16_I1arg3 + C16_I0arg3 * R8_X * C16_Z) -
     &    1.0 * (-1.0 * C16_I0arg1 * C16_I1arg2 * C16_s +
     &    C16_I0arg2 * C16_I1arg1 * C16_s * R8_X * R8_Xm) *
     &    (-1.0 * C16_K1arg3 + C16_K0arg3 * R8_X * C16_Z))) -
     &    1.0 * C16_I0arg2 * C16_I1arg1 * R8_X * R8_Xm *
     &    (-1.0 * C16_K1arg3 + C16_K0arg3 * R8_X * C16_Z) /
     &    (C16_I0arg1 * ((C16_I0arg1 * C16_K1arg2 * C16_s +
     &    C16_I1arg1 * C16_K0arg2 * C16_s * R8_X * R8_Xm) *
     &    (C16_I1arg3 + C16_I0arg3 * R8_X * C16_Z) -
     &    1.0 * (-1.0 * C16_I0arg1 * C16_I1arg2 * C16_s +
     &    C16_I0arg2 * C16_I1arg1 * C16_s * R8_X * R8_Xm) *
     &    (-1.0 * C16_K1arg3 + C16_K0arg3 * R8_X * C16_Z))))

      end

C ***********************************************************************
C * CALCULATION computes everything necessary for the
C * evaluation of a1, a3, and a4.
C ***********************************************************************
      subroutine CALCULATION (C16_s,
     &                        C16_I0arg1,
     &                        C16_I0arg2,
     &                        C16_I0arg3,
     &                        C16_I1arg1,
     &                        C16_I1arg2,
     &                        C16_I1arg3,
     &                        C16_K0arg2,
     &                        C16_K0arg3,
     &                        C16_K1arg2,
     &                        C16_K1arg3,
     &                        C16_arg1,
     &                        C16_arg2,
     &                        C16_arg3,
     &                        R8_d1,
     &                        R8_d2,
     &                        R8_vw,
     &                        R8_v1,
     &                        R8_a,
     &                        R8_b,
     &                        R8_X,
     &                        C16_Z,
     &                        R8_xm,
     &                        R8_xm2,
     &                        R8_area)

C ***********************************************************************
C * Constants
C ***********************************************************************
      real*8 R8_PI
      parameter (R8_PI = 3.14159265359)

C***********************************************************************
C Commons
C***********************************************************************
      real*8 R8_G_a
      real*8 R8_G_b
      real*8 R8_G_d1
      real*8 R8_G_d2
```

```
      real*8 R8_G_xm
      real*8 R8_G_vw
      real*8 R8_G_xm2

      common /IN_PARMS/
     &  R8_G_a,
     &  R8_G_b,
     &  R8_G_d1,
     &  R8_G_d2,
     &  R8_G_xm,
     &  R8_G_vw,
     &  R8_G_xm2

C ***************************************************************************
C * Variables
C ***************************************************************************
      complex*16 C16_s
      complex*16 C16_arg1
      complex*16 C16_arg2
      complex*16 C16_arg3
      complex*16 C16_I0arg1
      complex*16 C16_I0arg2
      complex*16 C16_I0arg3
      complex*16 C16_I1arg1
      complex*16 C16_I1arg2
      complex*16 C16_I1arg3
      complex*16 C16_K0arg2
      complex*16 C16_K0arg3
      complex*16 C16_K1arg2
      complex*16 C16_K1arg3
      complex*16 C16_Z
      complex*16 C16_a1
      complex*16 C16_a3
      complex*16 C16_a4

      real*8 R8_a
      real*8 R8_b
      real*8 R8_d1
      real*8 R8_d2
      real*8 R8_xm
      real*8 R8_vw
      real*8 R8_area
      real*8 R8_xm2
      real*8 R8_x
      real*8 R8_v1

C ***************************************************************************
C * Computations
C ***************************************************************************
      R8_a    = R8_G_a           ! Inside radius
      R8_b    = R8_G_b           ! Outside radius
      R8_d1   = R8_G_d1          ! Diffusivity in reservoir
      R8_d2   = R8_G_d2          ! Effective membrane diffusivity
      R8_xm   = R8_G_xm          ! Partition coefficient at A
      R8_vw   = R8_G_vw          ! External aqueous phase volume
      R8_xm2  = R8_G_xm2
      R8_area = 2.0 * R8_pi * R8_b  ! Outside area

      R8_X = sqrt(R8_d1 / R8_d2)
      C16_Z = R8_vw * CDSQRT(C16_s) / (R8_area * R8_b * R8_xm2)
```

```
C ***********************************************************************
C *   Arguments of the Bessel functions that appear in the expressions
C *   for a1, a3 and a4
C ***********************************************************************
      C16_arg1 = R8_a * CDSQRT(C16_s) / R8_b
      C16_arg2 = R8_a * CDSQRT(C16_s * R8_d1 / R8_d2) / R8_b
      C16_arg3 = CDSQRT(C16_s * R8_d1 / R8_d2)

C ***********************************************************************
C *   Evaluate the Bessel functions.
C ***********************************************************************
      call BESI0 (C16_arg1, C16_I0arg1)
      call BESI0 (C16_arg2, C16_I0arg2)
      call BESI0 (C16_arg3, C16_I0arg3)

      call BESI1 (C16_arg1, C16_I1arg1)
      call BESI1 (C16_arg2, C16_I1arg2)
      call BESI1 (C16_arg3, C16_I1arg3)

      call BESK0 (C16_arg2, C16_K0arg2)
      call BESK0 (C16_arg3, C16_K0arg3)

      call BESK1 (C16_arg2, C16_K1arg2)
      call BESK1 (C16_arg3, C16_K1arg3)

      return
      end

C ***********************************************************************
C * R8_REALTIME converts to real time.
C ***********************************************************************
      real*8 function R8_REALTIME(R8_THETA, R8_b, R8_d1)

      real*8 R8_THETA
      real*8 R8_b
      real*8 R8_d1

      R8_REALTIME = R8_THETA * (R8_b ** 2) / R8_d1

      return
      end

C ***********************************************************************
C * REALCONC converts to real concentration.
C ***********************************************************************
      real*8 function R8_REALCONC(R8_U, R8_INIT_CONC)

      real*8 R8_U
      real*8 R8_INIT_CONC

      R8_REALCONC = R8_U * R8_INIT_CONC

      return
      end
C ***********************************************************************
C * EOF
C ***********************************************************************
```

Table 10. Nomenclature Equivalents for the "Coated Fiber: Solution in Reservoir" Problem.

Text	Mathematica®	Fortran
$\sqrt{D_1 / D_2}$	X	C16_X
$\sqrt{D_2 / D_3}$	Y	C16_Y
$\frac{m_{3,w} b \alpha_3}{V_w} \sqrt{\frac{D_3}{D_1 s}}$	Z	C16_Z
$a / b\sqrt{s}$	arg1	C16_arg1
$\sqrt{s D_1 / D_3}$	arg4	C16_arg4
$\frac{c}{b}\sqrt{\frac{D_1}{D_3}s}$	arg5	C16_arg5
$\frac{a}{b}\sqrt{\frac{D_1}{D_2}s}$	arg2	C16_arg2
$\sqrt{\frac{D_1}{D_2}s}$	arg3	C16_arg3
C_1^0		C16_G_INIT_CONC
a		C16_G_a
A_{11}	A3	C16_a3
A_{12}	A4	C16_a4
A_{13}	A5	C16_a5
A_{14}	A6	C16_a6
A_9	A1	C16_a1
b		C16_G_b
c		C16_G_c
D_1		C16_G_d1
D_2		C16_G_d2
D_3		C16_G_d3
I_0	I0	C16_I0
I_1	I1	C16_I1
K_0	K0	C16_K0
K_1	K1	C16_K1
$m_{1,2}$		C16_G_xm
$m_{2,3}$		C16_G_xm2
$m_{3,w}$		C16_G_xm3
s	s	C16_s
V_w		C16_G_vw

Coated Fiber: Solution in Reservoir

Presented below is the Mathematica® program used to determine expressions for the coefficients A_9, A_{11}, A_{12}, A_{13},, and A_{14} for the "coated hollow fiber: solution in reservoir" problem. It is followed by the Fortran code used to invert the solution numerically from the Laplace domain into the time domain. The reader is referred to Table 10 for nomenclature relationships between variables in the text of Chapter 2 and Mathematica® programs and Fortran code.

```
Const = Solve[{1./s + A1 I0arg1 ==

     xm (A3 I0arg2 + A4 K0arg2),

     X A1 I1arg1 ==
     A3 I1arg2 - A4 K1arg2,

     A3 I0arg3 + A4 K0arg3 ==
     xm2 (A5 I0arg4 + A6 K0arg4),

     Y (A3 I1arg3 - A4 K1arg3) ==
      (A5 I1arg4 - A6 K1arg4),

     -Z (A5 I1arg5 - A6 K1arg5) ==
      (A5 I0arg5 + A6 K0arg5)},
     {A1, A3, A4, A5, A6}];


a1 = A1 /. Const;
a3 = A3 /. Const;
a4 = A4 /. Const;
a5 = A5 /. Const;
a6 = A6 /. Const;

FortranForm[a1]
FortranForm[a3]
FortranForm[a4]
FortranForm[a5]
FortranForm[a6]

List(-1./(I0arg1*s) - 1.*K0arg2*xm*
     -    (1.*I0arg3*I1arg1*X*(I0arg3*K1arg4 +
     -    I1arg3*K0arg4*xm2*Y)*
     -    (-1.*I0arg5 - 1.*I1arg5*Z)/
     -    ((K0arg4*(-1.*I0arg1*I1arg2*s +
     -    I0arg2*I1arg1*s*X*xm)*xm2*
     -    (-1.*I1arg3*K0arg3*Y -
     -    1.*I0arg3*K1arg3*Y) -
     -    1.*(-1.*K0arg3*
     -    (-1.*I0arg1*I1arg2*s +
     -    I0arg2*I1arg1*s*X*xm) +
     -    I0arg3*(I0arg1*K1arg2*s +
     -    I1arg1*K0arg2*s*X*xm))*
     -    (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y))*
     -    (-1.*I0arg5 - 1.*I1arg5*Z) -
     -    1.*(I0arg4*(-1.*I0arg1*I1arg2*s +
     -    I0arg2*I1arg1*s*X*xm)*
     -    xm2*(-1.*I1arg3*K0arg3*Y -
     -    1.*I0arg3*K1arg3*Y) -
     -    1.*(-1.*K0arg3*
```

```
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
-     I0arg3*(I0arg1*K1arg2*s +
-     I1arg1*K0arg2*s*X*xm))*
-     (-1.*I0arg3*I1arg4 +
-     I0arg4*I1arg3*xm2*Y))*
-     (-1.*K0arg5 + K1arg5*Z)) -
-     1.*I0arg3*I1arg1*X*(-1.*I0arg3*I1arg4 +
-     I0arg4*I1arg3*xm2*Y)*
-     (-1.*K0arg5 + K1arg5*Z)/
-     ((K0arg4*(-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm)*xm2*
-     (-1.*I1arg3*K0arg3*Y -
-     1.*I0arg3*K1arg3*Y) -
-     1.*(-1.*K0arg3*
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
-     I0arg3*(I0arg1*K1arg2*s +
-     I1arg1*K0arg2*s*X*xm))*
-     (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y))*
-     (-1.*I0arg5 - 1.*I1arg5*Z) -
-     1.*(I0arg4*(-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm)*
-     xm2*(-1.*I1arg3*K0arg3*Y -
-     1.*I0arg3*K1arg3*Y) -
-     1.*(-1.*K0arg3*
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
-     I0arg3*(I0arg1*K1arg2*s +
-     I1arg1*K0arg2*s*X*xm))*
-     (-1.*I0arg3*I1arg4 +
-     I0arg4*I1arg3*xm2*Y))*
-     (-1.*K0arg5 + K1arg5*Z)))/I0arg1 -
-     1.*I0arg2*xm*(-1.*I1arg1*K0arg4*X*xm2*
-     (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y)*
-     (-1.*I0arg5 - 1.*I1arg5*Z)/
-     ((K0arg4*(-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm)*xm2*
-     (-1.*I1arg3*K0arg3*Y -
-     1.*I0arg3*K1arg3*Y) -
-     1.*(-1.*K0arg3*
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
-     I0arg3*(I0arg1*K1arg2*s +
-     I1arg1*K0arg2*s*X*xm))*
-     (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y))*
-     (-1.*I0arg5 - 1.*I1arg5*Z) -
-     1.*(I0arg4*(-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm)*
-     xm2*(-1.*I1arg3*K0arg3*Y -
```

```
-     1.*I0arg3*K1arg3*Y) -
-     1.*(-1.*K0arg3*
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
-     I0arg3*(I0arg1*K1arg2*s +
-     I1arg1*K0arg2*s*X*xm))*
-     (-1.*I0arg3*I1arg4 +
-     I0arg4*I1arg3*xm2*Y))*
-     (-1.*K0arg5 + K1arg5*Z)) +
-     1.*I0arg4*I1arg1*X*xm2*
-     (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y)*
-     (-1.*K0arg5 + K1arg5*Z)/
-     ((K0arg4*(-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm)*xm2*
-     (-1.*I1arg3*K0arg3*Y -
-     1.*I0arg3*K1arg3*Y) -
-     1.*(-1.*K0arg3*
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
-     I0arg3*(I0arg1*K1arg2*s +
-     I1arg1*K0arg2*s*X*xm))*
-     (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y))*
-     (-1.*I0arg5 - 1.*I1arg5*Z) -
-     1.*(I0arg4*(-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm)*
-     xm2*(-1.*I1arg3*K0arg3*Y -
-     1.*I0arg3*K1arg3*Y) -
-     1.*(-1.*K0arg3*
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
-     I0arg3*(I0arg1*K1arg2*s +
-     I1arg1*K0arg2*s*X*xm))*
-     (-1.*I0arg3*I1arg4 +
-     I0arg4*I1arg3*xm2*Y))*
-     (-1.*K0arg5 + K1arg5*Z)) -
-     1.*K0arg3*(1.*I0arg3*I1arg1*X*
-     (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y)*
-     (-1.*I0arg5 - 1.*I1arg5*Z)/
-     ((K0arg4*(-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm)*
-     xm2*(-1.*I1arg3*K0arg3*Y -
-     1.*I0arg3*K1arg3*Y) -
-     1.*(-1.*K0arg3*
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
-     I0arg3*(I0arg1*K1arg2*s +
-     I1arg1*K0arg2*s*X*xm))*
-     (I0arg3*K1arg4 +
-     I1arg3*K0arg4*xm2*Y))*
-     (-1.*I0arg5 - 1.*I1arg5*Z) -
```

```
      -    1.*(I0arg4*
      -    (-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm)*xm2*
      -    (-1.*I1arg3*K0arg3*Y -
      -    1.*I0arg3*K1arg3*Y) -
      -    1.*(-1.*K0arg3*
      -    (-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm) +
      -    I0arg3*(I0arg1*K1arg2*s +
      -    I1arg1*K0arg2*s*X*xm))*
      -    (-1.*I0arg3*I1arg4 +
      -    I0arg4*I1arg3*xm2*Y))*
      -    (-1.*K0arg5 + K1arg5*Z)) -
      -    1.*I0arg3*I1arg1*X*
      -    (-1.*I0arg3*I1arg4 +
      -    I0arg4*I1arg3*xm2*Y)*
      -    (-1.*K0arg5 + K1arg5*Z)/
      -    ((K0arg4*(-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm)*
      -    xm2*(-1.*I1arg3*K0arg3*Y -
      -    1.*I0arg3*K1arg3*Y) -
      -    1.*(-1.*K0arg3*
      -    (-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm) +
      -    I0arg3*(I0arg1*K1arg2*s +
      -    I1arg1*K0arg2*s*X*xm))*
      -    (I0arg3*K1arg4 +
      -    I1arg3*K0arg4*xm2*Y))*
      -    (-1.*I0arg5 - 1.*I1arg5*Z) -
      -    1.*(I0arg4*
      -    (-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm)*xm2*
      -    (-1.*I1arg3*K0arg3*Y -
      -    1.*I0arg3*K1arg3*Y) -
      -    1.*(-1.*K0arg3*
      -    (-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm) +
      -    I0arg3*(I0arg1*K1arg2*s +
      -    I1arg1*K0arg2*s*X*xm))*
      -    (-1.*I0arg3*I1arg4 +
      -    I0arg4*I1arg3*xm2*Y))*
      -    (-1.*K0arg5 +
      -    K1arg5*Z)))/I0arg3)/I0arg1)

List(1.*I1arg1*K0arg4*X*xm2*
      -    (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y)*
      -    (-1.*I0arg5 - 1.*I1arg5*Z)/
      -    ((K0arg4*(-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm)*xm2*
      -    (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y) -
```

```
-     1.*(-1.*K0arg3*
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
-     I0arg3*(I0arg1*K1arg2*s +
-     I1arg1*K0arg2*s*X*xm))*
-     (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y))*
-     (-1.*I0arg5 - 1.*I1arg5*Z) -
-     1.*(I0arg4*(-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm)*xm2*
-     (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y) -
-     1.*(-1.*K0arg3*
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
-     I0arg3*(I0arg1*K1arg2*s +
-     I1arg1*K0arg2*s*X*xm))*
-     (-1.*I0arg3*I1arg4 + I0arg4*I1arg3*xm2*Y))*
-     (-1.*K0arg5 + K1arg5*Z)) -
-     1.*I0arg4*I1arg1*X*xm2*(-1.*I1arg3*K0arg3*Y -
-     1.*I0arg3*K1arg3*Y)*
-     (-1.*K0arg5 + K1arg5*Z)/
-     ((K0arg4*(-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm)*xm2*
-     (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y) -
-     1.*(-1.*K0arg3*
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
-     I0arg3*(I0arg1*K1arg2*s +
-     I1arg1*K0arg2*s*X*xm))*
-     (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y))*
-     (-1.*I0arg5 - 1.*I1arg5*Z) -
-     1.*(I0arg4*(-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm)*xm2*
-     (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y) -
-     1.*(-1.*K0arg3*
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
-     I0arg3*(I0arg1*K1arg2*s +
-     I1arg1*K0arg2*s*X*xm))*
-     (-1.*I0arg3*I1arg4 + I0arg4*I1arg3*xm2*Y))*
-     (-1.*K0arg5 + K1arg5*Z)) +
-     K0arg3*(1.*I0arg3*I1arg1*X*(I0arg3*K1arg4 +
-     I1arg3*K0arg4*xm2*Y)*
-     (-1.*I0arg5 - 1.*I1arg5*Z)/
-     ((K0arg4*(-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm)*xm2*
-     (-1.*I1arg3*K0arg3*Y -
-     1.*I0arg3*K1arg3*Y) -
-     1.*(-1.*K0arg3*
-     (-1.*I0arg1*I1arg2*s +
-     I0arg2*I1arg1*s*X*xm) +
```

```
      -    I0arg3*(I0arg1*K1arg2*s +
      -    I1arg1*K0arg2*s*X*xm))*
      -    (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y))*
      -    (-1.*I0arg5 - 1.*I1arg5*Z) -
      -    1.*(I0arg4*(-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm)*
      -    xm2*(-1.*I1arg3*K0arg3*Y -
      -    1.*I0arg3*K1arg3*Y) -
      -    1.*(-1.*K0arg3*
      -    (-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm) +
      -    I0arg3*(I0arg1*K1arg2*s +
      -    I1arg1*K0arg2*s*X*xm))*
      -    (-1.*I0arg3*I1arg4 +
      -    I0arg4*I1arg3*xm2*Y))*
      -    (-1.*K0arg5 + K1arg5*Z)) -
      -    1.*I0arg3*I1arg1*X*(-1.*I0arg3*I1arg4 +
      -    I0arg4*I1arg3*xm2*Y)*
      -    (-1.*K0arg5 + K1arg5*Z)/
      -    ((K0arg4*(-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm)*xm2*
      -    (-1.*I1arg3*K0arg3*Y -
      -    1.*I0arg3*K1arg3*Y) -
      -    1.*(-1.*K0arg3*
      -    (-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm) +
      -    I0arg3*(I0arg1*K1arg2*s +
      -    I1arg1*K0arg2*s*X*xm))*
      -    (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y))*
      -    (-1.*I0arg5 - 1.*I1arg5*Z) -
      -    1.*(I0arg4*(-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm)*
      -    xm2*(-1.*I1arg3*K0arg3*Y -
      -    1.*I0arg3*K1arg3*Y) -
      -    1.*(-1.*K0arg3*
      -    (-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm) +
      -    I0arg3*(I0arg1*K1arg2*s +
      -    I1arg1*K0arg2*s*X*xm))*
      -    (-1.*I0arg3*I1arg4 +
      -    I0arg4*I1arg3*xm2*Y))*
      -    (-1.*K0arg5 + K1arg5*Z)))/I0arg3)

List(-1.*I0arg3*I1arg1*X*(I0arg3*K1arg4 +
      -    I1arg3*K0arg4*xm2*Y)*
      -    (-1.*I0arg5 - 1.*I1arg5*Z)/
      -    ((K0arg4*(-1.*I0arg1*I1arg2*s +
      -    I0arg2*I1arg1*s*X*xm)*xm2*
      -    (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y) -
      -    1.*(-1.*K0arg3*
```

```
      -     (-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm) +
      -     I0arg3*(I0arg1*K1arg2*s +
      -     I1arg1*K0arg2*s*X*xm))*
      -     (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y))*
      -     (-1.*I0arg5 - 1.*I1arg5*Z) -
      -     1.*(I0arg4*(-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm)*xm2*
      -     (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y) -
      -     1.*(-1.*K0arg3*
      -     (-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm) +
      -     I0arg3*(I0arg1*K1arg2*s +
      -     I1arg1*K0arg2*s*X*xm))*
      -     (-1.*I0arg3*I1arg4 + I0arg4*I1arg3*xm2*Y))*
      -     (-1.*K0arg5 + K1arg5*Z)) +
      -     1.*I0arg3*I1arg1*X*(-1.*I0arg3*I1arg4 +
      -     I0arg4*I1arg3*xm2*Y)*
      -     (-1.*K0arg5 + K1arg5*Z)/
      -     ((K0arg4*(-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm)*xm2*
      -     (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y) -
      -     1.*(-1.*K0arg3*
      -     (-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm) +
      -     I0arg3*(I0arg1*K1arg2*s +
      -     I1arg1*K0arg2*s*X*xm))*
      -     (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y))*
      -     (-1.*I0arg5 - 1.*I1arg5*Z) -
      -     1.*(I0arg4*(-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm)*xm2*
      -     (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y) -
      -     1.*(-1.*K0arg3*
      -     (-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm) +
      -     I0arg3*(I0arg1*K1arg2*s +
      -     I1arg1*K0arg2*s*X*xm))*
      -     (-1.*I0arg3*I1arg4 + I0arg4*I1arg3*xm2*Y))*
      -     (-1.*K0arg5 + K1arg5*Z)))

List(-1.*I0arg3*I1arg1*X*(-1.*I1arg3*K0arg3*Y -
      -     -1.*I0arg3*K1arg3*Y)*
      -     (-1.*K0arg5 + K1arg5*Z)/
      -     ((K0arg4*(-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm)*xm2*
      -     (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y) -
      -     1.*(-1.*K0arg3*
      -     (-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm) +
      -     I0arg3*(I0arg1*K1arg2*s +
```

```
      -     I1arg1*K0arg2*s*X*xm))*
      -     (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y))*
      -     (-1.*I0arg5 - 1.*I1arg5*Z) -
      -     1.*(I0arg4*(-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm)*xm2*
      -     (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y) -
      -     1.*(-1.*K0arg3*
      -     (-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm) +
      -     I0arg3*(I0arg1*K1arg2*s +
      -     I1arg1*K0arg2*s*X*xm))*
      -     (-1.*I0arg3*I1arg4 + I0arg4*I1arg3*xm2*Y))*
      -     (-1.*K0arg5 + K1arg5*Z)))

List(1.*I0arg3*I1arg1*X*(-1.*I1arg3*K0arg3*Y -
      -     1.*I0arg3*K1arg3*Y)*
      -     (-1.*I0arg5 - 1.*I1arg5*Z)/
      -     ((K0arg4*(-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm)*xm2*
      -     (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y) -
      -     1.*(-1.*K0arg3*
      -     (-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm) +
      -     I0arg3*(I0arg1*K1arg2*s +
      -     I1arg1*K0arg2*s*X*xm))*
      -     (I0arg3*K1arg4 + I1arg3*K0arg4*xm2*Y))*
      -     (-1.*I0arg5 - 1.*I1arg5*Z) -
      -     1.*(I0arg4*(-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm)*xm2*
      -     (-1.*I1arg3*K0arg3*Y - 1.*I0arg3*K1arg3*Y) -
      -     1.*(-1.*K0arg3*
      -     (-1.*I0arg1*I1arg2*s +
      -     I0arg2*I1arg1*s*X*xm) +
      -     I0arg3*(I0arg1*K1arg2*s +
      -     I1arg1*K0arg2*s*X*xm))*
      -     (-1.*I0arg3*I1arg4 + I0arg4*I1arg3*xm2*Y))*
      -     (-1.*K0arg5 + K1arg5*Z)))
```

```
C ***********************************************************************
C * Program model_coat provides release profiles for the coated fiber-
C * solution in reservoir problem.
C *
C * Stephanie Farrell
C * September 1995
C *
C * © Copyright 1995, 1996 Stephanie Farrell
C *
C * FILE: COAT.FOR → COAT.EXE
C ***********************************************************************
      program MODEL_COAT

C ***********************************************************************
C * Function declarations
C ***********************************************************************
      real FLOAT
      real*8 REALCONC
      real*8 REALTIME

      external C16_F_COAT
      external COAT
      external CALCULATION
      external DINLAP
      external BESI0
      external BESI1
      external BESK0
      external BESK1
      external REALCONC
      external REALTIME

C ***********************************************************************
C * Variable declarations
C ***********************************************************************
      integer I_INDEX
      integer I_KMAX

      real*8 R8_ALPHA
      real*8 R8_RELERR
      real*8 R8_T(1)
      real*8 R8_FINV(1)

C ***********************************************************************
C * Data input parameters
C ***********************************************************************
      character TITLE*80
      character VAR_NAME*30

      integer I_NUM_TIMES

      real*8 R8_TIME_INTERVAL
      real*8 R8_INIT_CONC

      complex*16 C16_G_a
      complex*16 C16_G_b
      complex*16 C16_G_c
      complex*16 C16_G_d1
      complex*16 C16_G_d2
      complex*16 C16_G_d3
      complex*16 C16_G_xm
      complex*16 C16_G_xm2
```

```
      complex*16 C16_G_xm3
      complex*16 C16_G_vw

      character EOF*80

      common /IN_PARMS/
     &  C16_G_a,
     &  C16_G_b,
     &  C16_G_c,
     &  C16_G_d1,
     &  C16_G_d2,
     &  C16_G_d3,
     &  C16_G_xm,
     &  C16_G_xm2,
     &  C16_G_xm3,
     &  C16_G_vw

C ************************************************************************
C * Read in data
C ************************************************************************
      read *, TITLE
      read *, VAR_NAME, I_NUM_TIMES
      read *, VAR_NAME, R8_TIME_INTERVAL
      read *, VAR_NAME, R8_INIT_CONC

      read *, VAR_NAME, C16_G_a
      read *, VAR_NAME, C16_G_b
      read *, VAR_NAME, C16_G_c
      read *, VAR_NAME, C16_G_d1
      read *, VAR_NAME, C16_G_d2
      read *, VAR_NAME, C16_G_d3
      read *, VAR_NAME, C16_G_xm
      read *, VAR_NAME, C16_G_xm2
      read *, VAR_NAME, C16_G_xm3
      read *, VAR_NAME, C16_G_vw
      read *, EOF

      if (EOF .NE. 'EOF') then
         print *,'Input file format is incorrect. Aborting.'
         Goto 1000
      endif

      print *, TITLE
      print *, 'I_NUM_TIMES      ,', I_NUM_TIMES
      print *, 'R8_TIME_INTERVAL ,', R8_TIME_INTERVAL
      print *, 'R8_INIT_CONC     ,', R8_INIT_CONC
      print *, 'C16_G_a          ,', C16_G_a
      print *, 'C16_G_b          ,', C16_G_b
      print *, 'C16_G_c          ,', C16_G_c
      print *, 'C16_G_d1         ,', C16_G_d1
      print *, 'C16_G_d2         ,', C16_G_d2
      print *, 'C16_G_d3         ,', C16_G_d3
      print *, 'C16_G_xm         ,', C16_G_xm
      print *, 'C16_G_xm2        ,', C16_G_xm2
      print *, 'C16_G_xm3        ,', C16_G_xm3
      print *, 'C16_G_vw         ,', C16_G_vw

C ************************************************************************
C * User-supplied arguments for the IMSL subroutine DINLAP
C ************************************************************************
      R8_ALPHA = 0
```

```
      I_KMAX = 10000
      R8_RELERR = 5.0E-5

C ***********************************************************************
C * Evaluate the solution for the coating section.
C ***********************************************************************
      print *,'THETA,U,REAL TIME,REAL CONC.'
      Do I_INDEX = I_NUM_TIMES, 1, -1
          R8_T(1) = R8_TIME_INTERVAL * FLOAT(I_INDEX)
          call DINLAP (C16_F_COAT,
     &                 1,
     &                 R8_T,
     &                 R8_ALPHA,
     &                 R8_RELERR,
     &                 I_KMAX,
     &                 R8_FINV)

          print '(E15.4, A, E15.4, A, F15.2, A, E15.4)',
     &           R8_T(1), ',',
     &           R8_FINV(1), ',',
     &           REALTIME(R8_T(1), C16_G_b, C16_G_d1), ',',
     &           REALCONC(R8_FINV(1), R8_INIT_CONC)
       end do

 1000 continue
      end

C ***********************************************************************
C * User-supplied function to which the inverse Laplace transform
C * will be computed
C ***********************************************************************
      complex*16 function C16_F_COAT(C16_S)

      intrinsic CDSQRT

      complex*16 C16_S
      complex*16 C16_I0arg5
      complex*16 C16_K0arg5
      complex*16 C16_a5
      complex*16 C16_a6
      complex*16 C16_xm3

      call COAT(C16_S,
     &          C16_I0arg5,
     &          C16_K0arg5,
     &          C16_xm3,
     &          C16_a5,
     &          C16_a6)

      C16_F_COAT = (C16_a5 * C16_I0arg5 + C16_a6 * C16_K0arg5)

      return
      end

C ***********************************************************************
C * Subroutine BESI0 computes Bessel(I,0)
C ***********************************************************************
      subroutine BESI0 (C16_X, C16_I0)

      complex*16 C16_X
      complex*16 C16_I0
```

```
      C16_I0 = 1.0 + C16_X ** 2 / 4.0 + C16_X ** 4 / 64.0 +
     &   C16_X ** 6 / 2304.0 + C16_X**8 / 147456.0 +
     &   C16_X ** 10 / 14745600.0 + C16_X**12 / 2123366400.0 +
     &   C16_X ** 14 / 416179814400.0 +
     &   C16_X ** 16 / 106542032486400.0 +
     &   C16_X ** 18 / 34519618525593600.0 +
     &   C16_X ** 20 / 13807847410237440000.0

      return
      end

C ***********************************************************************
C * Subroutine BESI1 computes Bessel(I,1)
C ***********************************************************************
      subroutine BESI1 (C16_X, C16_I1)

      complex*16 C16_X
      complex*16 C16_I1

      C16_I1 = C16_X/2.0 + C16_X**3/16.0 +
     &     C16_X**5/384.0 + C16_X**7/18432.0 +
     &     C16_X**9/1474560.0 + C16_X**11/176947200.0 +
     &     C16_X**13/29727129600.0 +
     &     C16_X**15/6658877030400.0 +
     &     C16_X**17/1917756584755200.0 +
     &     C16_X**19/690392370511872000.0

      return
      end

C ***********************************************************************
C * Subroutine BESK0 computes Bessel(K,0)
C ***********************************************************************
      subroutine BESK0 (C16_X, C16_K0)

      complex*16 C16_X
      complex*16 C16_XL
      complex*16 C16_K0

      intrinsic CDLOG
      intrinsic LOG
      intrinsic CDSQRT

      C16_XL = LOG(1.0/2.0) + CDLOG(C16_X)

      C16_K0 = -0.577215 - LOG(1.0/2.0) +
     &    C16_X ** 20 * ((7381.0 / 1260.0 - 2 * 0.577215) /
     &    27615694820474880000.0 +
     &    (-C16_XL) / 13807847410237440000.0) +
     &    C16_X ** 18 *
     &    ((7129.0 / 1260.0 - 2 * 0.577215)/69039237051187200.0 +
     &    (-C16_XL) / 34519618525593600.0) +
     &    C16_X ** 16*((761.0/140.0 - 2*0.577215) /
     &    213084064972800.0 +
     &    (-C16_XL) / 106542032486400.0) +
     &    C16_X ** 14 * ((363.0/70.0 - 2 * 0.577215) /
     &    832359628800.0 +
     &    (-C16_XL) / 416179814400.0) +
     &    C16_X ** 12*((49.0/10.0 - 2*0.577215)/4246732800.0 +
     &    (-C16_XL) / 2123366400.0) +
```

```
     &     C16_X ** 10 * ((137.0/30.0 - 2 * 0.577215)/29491200.0 +
     &     (-C16_XL) / 14745600.0) +
     &     C16_X ** 8 * ((25.0/6.0 - 2 * 0.577215) /
     &     294912.0 + (-C16_XL) /
     &     147456.0) +
     &     C16_X ** 6 * ((11.0/3.0 - 2 * 0.577215)/4608.0 +
     &     (-C16_XL) / 2304.0) +
     &     C16_X ** 4 * ((3.0 - 2 * 0.577215)/128.0 +
     &     (-C16_XL)/64.0) +
     &     C16_X ** 2 * ((2.0 - 2 * 0.577215)/8.0 +
     &     (-C16_XL)/4.0) - CDLOG(C16_X)

      return
      end

C ***********************************************************************
C * Subroutine BESK1 computes Bessel(K,1)
C ***********************************************************************
      subroutine BESK1 (C16_X, C16_K1)

      complex*16 C16_X
      complex*16 C16_XL
      complex*16 C16_K1

      intrinsic CDLOG
      intrinsic LOG
      intrinsic CDSQRT

      C16_XL = LOG(1.0/2.0) + CDLOG(C16_X)

      C16_K1 = 1.0/ C16_X + C16_X ** 17 *
     &  ((-6989.0/1260.0 + 2*0.577215)/
     &  3835513169510400.0 + C16_XL / 1917756584755200.0) +
     &  C16_X **15*((-1487.0/280.0 + 2*0.577215)/13317754060800.0 +
     &  C16_XL/6658877030400.0) +
     &  C16_X **13*((-353.0/70.0 + 2*0.577215)/59454259200.0 +
     &  C16_XL/29727129600.0) +
     &  C16_X **11*((-71.0/15.0 + 2*0.577215)/353894400.0 +
     &  C16_XL/176947200.0) +
     &  C16_X **9*((-131.0/30.0 + 2*0.577215)/2949120.0 +
     &  C16_XL/1474560.0) +
     &  C16_X **7*((-47.0/12.0 + 2*0.577215)/36864.0 +
     &  C16_XL/18432.0) +
     &  C16_X **5*((-10.0/3.0 + 2*0.577215)/768.0 + C16_XL/384.0) +
     &  C16_X **3*((-5.0/2.0 + 2*0.577215)/32.0 + C16_XL/16.0) +
     &  C16_X *((-1.0 + 2*0.577215)/4.0 + C16_XL/2.0)

      return
      end

C ***********************************************************************
C * Subroutine COAT evaluates C16_a5 and C16_a6 which
C * are used in C16_F_COAT.
C ***********************************************************************
      subroutine COAT (C16_S,
     &                 C16_I0arg5,
     &                 C16_K0arg5,
     &                 C16_xm3,
     &                 C16_a5,
     &                 C16_a6)
```

```
      complex*16 C16_S
      complex*16 C16_I0arg1
      complex*16 C16_I0arg2
      complex*16 C16_I0arg3
      complex*16 C16_I0arg4
      complex*16 C16_I0arg5
      complex*16 C16_I1arg1
      complex*16 C16_I1arg2
      complex*16 C16_I1arg3
      complex*16 C16_I1arg4
      complex*16 C16_I1arg5
      complex*16 C16_K0arg2
      complex*16 C16_K0arg3
      complex*16 C16_K0arg4
      complex*16 C16_K0arg5
      complex*16 C16_K1arg2
      complex*16 C16_K1arg3
      complex*16 C16_K1arg4
      complex*16 C16_K1arg5
      complex*16 C16_arg1
      complex*16 C16_arg2
      complex*16 C16_arg3
      complex*16 C16_arg4
      complex*16 C16_arg5
      complex*16 C16_a5sub1
      complex*16 C16_a5sub2
      complex*16 C16_a6sub1
      complex*16 C16_a6sub2
      complex*16 C16_a5
      complex*16 C16_a6
      complex*16 C16_a
      complex*16 C16_b
      complex*16 C16_c
      complex*16 C16_d1
      complex*16 C16_d2
      complex*16 C16_d3
      complex*16 C16_area
      complex*16 C16_vw
      complex*16 C16_X
      complex*16 C16_Y
      complex*16 C16_Z
      complex*16 C16_xm
      complex*16 C16_xm2
      complex*16 C16_xm3

C ***********************************************************************
C * Call calculation to calculate everything necessary for
C *  evaluation of C16_a5 and C16_a6.
C ***********************************************************************

       call CALCULATION(C16_S,
     &                  C16_I0arg1,
     &                  C16_I0arg2,
     &                  C16_I0arg3,
     &                  C16_I0arg4,
     &                  C16_I0arg5,
     &                  C16_I1arg1,
     &                  C16_I1arg2,
     &                  C16_I1arg3,
     &                  C16_I1arg4,
     &                  C16_I1arg5,
```

```
     &                 C16_K0arg2,
     &                 C16_K0arg3,
     &                 C16_K0arg4,
     &                 C16_K0arg5,
     &                 C16_K1arg2,
     &                 C16_K1arg3,
     &                 C16_K1arg4,
     &                 C16_K1arg5,
     &                 C16_arg1,
     &                 C16_arg2,
     &                 C16_arg3,
     &                 C16_arg4,
     &                 C16_arg5,
     &                 C16_d1,
     &                 C16_d2,
     &                 C16_d3,
     &                 C16_vw,
     &                 C16_a,
     &                 C16_b,
     &                 C16_c,
     &                 C16_X,
     &                 C16_Y,
     &                 C16_Z,
     &                 C16_xm,
     &                 C16_xm2,
     &                 C16_xm3,
     &                 C16_area)

C ***********************************************************************
C * Compute C16_a5
C ***********************************************************************

      C16_a5sub1 = (C16_K0arg4*(-1.*C16_I0arg1*C16_I1arg2*C16_s +
     &   C16_I0arg2*C16_I1arg1*C16_s*C16_X*C16_xm)*C16_xm2*
     &  (-1.*C16_I1arg3*C16_K0arg3*C16_Y -
     &   1.*C16_I0arg3*C16_K1arg3*C16_Y) -
     &        1.*(-1.*C16_K0arg3*
     &   (-1.*C16_I0arg1*C16_I1arg2*C16_s + C16_I0arg2*C16_I1arg1*
     &  C16_s*C16_X*C16_xm) +
     &    C16_I0arg3*(C16_I0arg1*C16_K1arg2*C16_s +
     &  C16_I1arg1*C16_K0arg2*C16_s*C16_X*C16_xm))*
     &    (C16_I0arg3*C16_K1arg4 + C16_I1arg3*C16_K0arg4*C16_xm2*C16_Y))

       C16_a5sub2 = (C16_I0arg4*(-1.*C16_I0arg1*C16_I1arg2*C16_s +
     &    C16_I0arg2*C16_I1arg1*C16_s*C16_X*C16_xm)*C16_xm2*
     &         (-1.*C16_I1arg3*C16_K0arg3*C16_Y -
     &  1.*C16_I0arg3*C16_K1arg3*C16_Y) -
     &        1.*(-1.*C16_K0arg3*
     &            (-1.*C16_I0arg1*C16_I1arg2*C16_s +
     &  C16_I0arg2*C16_I1arg1*C16_s*C16_X*C16_xm) +
     &           C16_I0arg3*(C16_I0arg1*C16_K1arg2*C16_s +
     &  C16_I1arg1*C16_K0arg2*C16_s*C16_X*C16_xm))*
     &   (-1.*C16_I0arg3*C16_I1arg4 +
     &C16_I0arg4*C16_I1arg3*C16_xm2*C16_Y))

      C16_a5 = (-1.*C16_I0arg3*C16_I1arg1*C16_X*
     &  (-1.*C16_I1arg3*C16_K0arg3*C16_Y -
     &     1.*C16_I0arg3*C16_K1arg3*C16_Y)*
     &    (-1.*C16_K0arg5 + C16_K1arg5*C16_Z)/
     &   (C16_a5sub1*
     &      (-1.*C16_I0arg5 - 1.*C16_I1arg5*C16_Z) -
```

```
     &     1.*C16_a5sub2*
     &      (-1.*C16_K0arg5 + C16_K1arg5*C16_Z)))

C ***********************************************************************
C * Compute C16_a6
C ***********************************************************************


       C16_a6sub1 = (C16_K0arg4*(-1.*C16_I0arg1*C16_I1arg2*C16_s +
     &   C16_I0arg2*C16_I1arg1*C16_s*C16_X*C16_xm)*C16_xm2*
     &         (-1.*C16_I1arg3*C16_K0arg3*C16_Y -
     &  1.*C16_I0arg3*C16_K1arg3*C16_Y) -
     &        1.*(-1.*C16_K0arg3*
     &   (-1.*C16_I0arg1*C16_I1arg2*C16_s + C16_I0arg2*C16_I1arg1*C16_s*
     &  C16_X*C16_xm) +
     &    C16_I0arg3*(C16_I0arg1*C16_K1arg2*C16_s +
     &   C16_I1arg1*C16_K0arg2*C16_s*
     &   C16_X*C16_xm))*
     &   (C16_I0arg3*C16_K1arg4 + C16_I1arg3*C16_K0arg4*C16_xm2*C16_Y))

       C16_a6sub2 = (C16_I0arg4*(-1.*C16_I0arg1*C16_I1arg2*C16_s +
     &   C16_I0arg2*C16_I1arg1*C16_s*C16_X*C16_xm)*C16_xm2*
     &         (-1.*C16_I1arg3*C16_K0arg3*C16_Y -
     &    1.*C16_I0arg3*C16_K1arg3*C16_Y) -
     &        1.*(-1.*C16_K0arg3*
     &     (-1.*C16_I0arg1*C16_I1arg2*C16_s +
     &   C16_I0arg2*C16_I1arg1*C16_s*C16_X*C16_xm) +
     &     C16_I0arg3*(C16_I0arg1*C16_K1arg2*C16_s +
     &    C16_I1arg1*C16_K0arg2*C16_s*C16_X*C16_xm))*
     &    (-1.*C16_I0arg3*C16_I1arg4 +
     & C16_I0arg4*C16_I1arg3*C16_xm2*C16_Y))

      C16_a6 = (1.*C16_I0arg3*C16_I1arg1*C16_X*
     &    (-1.*C16_I1arg3*C16_K0arg3*C16_Y -
     &  1.*C16_I0arg3*C16_K1arg3*C16_Y)*
     &    (-1.*C16_I0arg5 - 1.*C16_I1arg5*C16_Z)/
     &   (C16_a6sub1*
     &      (-1.*C16_I0arg5 - 1.*C16_I1arg5*C16_Z) -
     &     1.*C16_a6sub2*
     &      (-1.*C16_K0arg5 + C16_K1arg5*C16_Z)))

      return
      end

C ***********************************************************************
C * Subroutine CALCULATION computes everything necessary for evaluation
C * of C16_a5, and C16_a6
C ***********************************************************************
      subroutine CALCULATION(C16_S,
     &                       C16_I0arg1,
     &                       C16_I0arg2,
     &                       C16_I0arg3,
     &                       C16_I0arg4,
     &                       C16_I0arg5,
     &                       C16_I1arg1,
     &                       C16_I1arg2,
     &                       C16_I1arg3,
     &                       C16_I1arg4,
     &                       C16_I1arg5,
     &                       C16_K0arg2,
     &                       C16_K0arg3,
```

```
     &                        C16_K0arg4,
     &                        C16_K0arg5,
     &                        C16_K1arg2,
     &                        C16_K1arg3,
     &                        C16_K1arg4,
     &                        C16_K1arg5,
     &                        C16_arg1,
     &                        C16_arg2,
     &                        C16_arg3,
     &                        C16_arg4,
     &                        C16_arg5,
     &                        C16_d1,
     &                        C16_d2,
     &                        C16_d3,
     &                        C16_vw,
     &                        C16_a,
     &                        C16_b,
     &                        C16_c,
     &                        C16_X,
     &                        C16_Y,
     &                        C16_Z,
     &                        C16_xm,
     &                        C16_xm2,
     &                        C16_xm3,
     &                        C16_area)

C ***************************************************************************
C * Commons
C ***************************************************************************
      complex*16 C16_G_a
      complex*16 C16_G_b
      complex*16 C16_G_c
      complex*16 C16_G_d1
      complex*16 C16_G_d2
      complex*16 C16_G_d3
      complex*16 C16_G_xm
      complex*16 C16_G_xm2
      complex*16 C16_G_xm3
      complex*16 C16_G_vw

      common /IN_PARMS/
     &  C16_G_a,
     &  C16_G_b,
     &  C16_G_c,
     &  C16_G_d1,
     &  C16_G_d2,
     &  C16_G_d3,
     &  C16_G_xm,
     &  C16_G_xm2,
     &  C16_G_xm3,
     &  C16_G_vw

C ***************************************************************************
C * Constants
C ***************************************************************************
      real*8 R8_pi
      parameter (R8_pi = 3.14159265359)

C ***************************************************************************
C * Computations
C ***************************************************************************
```

```
intrinsic CDSQRT

complex*16 C16_S
complex*16 C16_I0arg1
complex*16 C16_I0arg2
complex*16 C16_I0arg3
complex*16 C16_I0arg4
complex*16 C16_I0arg5
complex*16 C16_I1arg1
complex*16 C16_I1arg2
complex*16 C16_I1arg3
complex*16 C16_I1arg4
complex*16 C16_I1arg5
complex*16 C16_K0arg2
complex*16 C16_K0arg3
complex*16 C16_K0arg4
complex*16 C16_K0arg5
complex*16 C16_K1arg2
complex*16 C16_K1arg3
complex*16 C16_K1arg4
complex*16 C16_K1arg5
complex*16 C16_arg1
complex*16 C16_arg2
complex*16 C16_arg3
complex*16 C16_arg4
complex*16 C16_arg5
complex*16 C16_X
complex*16 C16_Y
complex*16 C16_Z
complex*16 C16_d1
complex*16 C16_d2
complex*16 C16_d3
complex*16 C16_a
complex*16 C16_b
complex*16 C16_c
complex*16 C16_area
complex*16 C16_vw
complex*16 C16_xm
complex*16 C16_xm2
complex*16 C16_xm3

C16_a    = C16_G_a      ! Inside radius
C16_b    = C16_G_b      ! Outside radius excluding coating
C16_c    = C16_G_c      ! Outside radius including coating
C16_d1   = C16_G_d1     ! Diffusivity in reservoir
C16_d2   = C16_G_d2     ! Effective membrane diffusivity
C16_d3   = C16_G_d3     ! Diffusivity in coating
C16_xm   = C16_G_xm     ! Partition coefficient at C16_a
C16_xm2  = C16_G_xm2    ! Partition coefficient at C16_b
C16_xm3  = C16_G_xm3    ! Partition coefficient at C16_c
C16_vw   = C16_G_vw     ! External aqueous phase volume
C16_area = 2.0 * R8_pi * C16_c ! Outside area

C16_arg1 = C16_a * CDSQRT(C16_S) / C16_b
C16_arg2 = C16_a * CDSQRT(C16_S * C16_d1 / C16_d2) / C16_b
C16_arg3 = CDSQRT(C16_S * C16_d1 / C16_d2)
C16_arg4 = CDSQRT(C16_S * C16_d1 / C16_d3)
C16_arg5 = C16_c * CDSQRT(C16_S * C16_d1 / C16_d3) / C16_b

C16_X = CDSQRT(C16_d1 / C16_d2)
C16_Y = CDSQRT(C16_d2 / C16_d3)
```

```
      C16_Z = (C16_xm3 * C16_b * C16_area / C16_vw) *
     &        CDSQRT(C16_d3 / (C16_d1 * C16_S))

C ***********************************************************************
C * Call subroutines to evaluate the Bessel functions.
C ***********************************************************************
      call BESI0(C16_arg1, C16_I0arg1)
      call BESI0(C16_arg2, C16_I0arg2)
      call BESI0(C16_arg3, C16_I0arg3)
      call BESI0(C16_arg4, C16_I0arg4)
      call BESI0(C16_arg5, C16_I0arg5)

      call BESI1(C16_arg1, C16_I1arg1)
      call BESI1(C16_arg2, C16_I1arg2)
      call BESI1(C16_arg3, C16_I1arg3)
      call BESI1(C16_arg4, C16_I1arg4)
      call BESI1(C16_arg5, C16_I1arg5)

      call BESK0(C16_arg2, C16_K0arg2)
      call BESK0(C16_arg3, C16_K0arg3)
      call BESK0(C16_arg4, C16_K0arg4)
      call BESK0(C16_arg5, C16_K0arg5)

      call BESK1(C16_arg2, C16_K1arg2)
      call BESK1(C16_arg3, C16_K1arg3)
      call BESK1(C16_arg4, C16_K1arg4)
      call BESK1(C16_arg5, C16_K1arg5)

      return
      end

C ***********************************************************************
C * Subroutine REALTIME converts to real time.
C ***********************************************************************
      real*8 function REALTIME(R8_THETA, C16_b, C16_d1)

      real*8 R8_THETA
      complex*16 C16_b
      complex*16 C16_d1

      REALTIME = R8_THETA * (C16_b ** 2) / C16_d1

      return
      end

C ***********************************************************************
C * Subroutine REALCONC converts to real concentration.
C ***********************************************************************
      real*8 function REALCONC(C16_U, C16_INIT_CONC)

      complex*16 C16_U
      complex*16 C16_INIT_CONC

      REALCONC = C16_U * C16_INIT_CONC

      return
      end
C ***********************************************************************
C * EOF
C ***********************************************************************
```

Table 11. Nomenclature Equivelents for the "Flat Membrane: Dispersed Phase in Reservoir" Problem.

Text	Mathematica®	Fortran
$\frac{a}{b}\sqrt{\delta}$	arg1	C16_arg1
$\frac{a}{b}\sqrt{\frac{D_1}{D_2}}s$	arg2	C16_arg2
$\sqrt{\frac{D_1}{D_2}}s$	arg3	C16_arg3
$\frac{\alpha_2 b m_{2,w}}{V_w}\sqrt{\frac{D_2}{sD_1}}$	Z	C16_Z
$\sqrt{\frac{D_1\delta}{sD_2}}$	X	C16_X
$\frac{C_s^0}{C_1^0}$		R8_G_CS0_C10
a		R8_G_a
A_{15}	a1	
A_{16}	a2	
A_{17}	a3	C16_a3
A_{18}	a4	C16_a4
D_1	d1	R8_G_d1
D_2	d2	R8_G_d2
b		R8_G_b
δ	delta	C16_delta
ϕ		R8_G_phi
γ	gamma	C16_gamma
k		R8_G_k
$m_{1.2}$	xm	R8_G_xm
$m_{1.s}$		R8_G_xms
$m_{2.w}$		R8_G_xm2
V_w		R8_G_vw

APPENDIX 2

Flat Membrane: Dispersed Phase in Reservoir

Presented below is the Mathematica® program used to determine expressions for the coefficients A_{15}, A_{16}, A_{17}, and A_{18} for the "flat membrane: dispersed phase in reservoir" problem. It is followed by the Fortran code used to invert the solution numerically from the Laplace domain into the time domain. The reader is referred to Table 11 for relationships between some of the variables appearing in the text in Chapter 2 and those used in the Mathematica® and Fortran Programs.

```
Bcflat = Solve[{a1 Sqrt[delta]  == a2 Sqrt[delta],


      X (a1 exp[arg1] - a2 exp[-arg1]) ==
      (a3 exp[arg2] - a4 exp[-arg2]),

      a1 exp[arg1] + a2 exp[-arg1] - gamma/delta ==
      xm (a3 exp[arg2] + a4 exp[-arg2]),

      -Z (a3 exp[arg3]  - a4 exp[-arg3]) ==
        (a3 exp[arg3] + a4 exp[-arg3])},

      {a1, a2, a3, a4}] ;


A1 =a1 /. Bcflat;
A2 = a2 /. Bcflat;
A3 = a3 /. Bcflat;
A4 = a4 /. Bcflat;

FortranForm[A1]
List(gamma/(delta*exp(-arg1) + delta*exp(arg1)) +
delta*gamma*xm*
     -    (-(X*exp(-arg1)) +
     -    X*exp(arg1))*exp(arg2)*
     -    (-exp(-arg3) + Z*exp(-arg3))/
     -    ((delta*exp(-arg1) + delta*exp(arg1))*
     -    (-((-((delta*exp(-arg1) +
     -    delta*exp(arg1))*exp(arg2)) +
     -    delta*xm*
     -    (-(X*exp(-arg1)) + X*exp(arg1))*
     -    exp(arg2))*
     -    (-exp(-arg3) + Z*exp(-arg3))) +
     -    ((delta*exp(-arg1) + delta*exp(arg1))*
     -    exp(-arg2) +
     -    delta*xm*
     -    (-(X*exp(-arg1)) + X*exp(arg1))*
     -    exp(-arg2))*
     -    (-exp(arg3) - Z*exp(arg3)))) -
     -    delta*gamma*xm*
     -    (-(X*exp(-arg1)) + X*exp(arg1))*exp(-
     -    arg2)*
     -    (-exp(arg3) - Z*exp(arg3))/
     -    ((delta*exp(-arg1) + delta*exp(arg1))*
     -    (-((-((delta*exp(-arg1) +
     -    delta*exp(arg1))*exp(arg2)) +
     -    delta*xm*
     -    (-(X*exp(-arg1)) + X*exp(arg1))*
     -    exp(arg2))*
     -    (-exp(-arg3) + Z*exp(-arg3))) +
```

```
      -     ((delta*exp(-arg1) + delta*exp(arg1))*
      -     exp(-arg2) +
      -     delta*xm*
      -     (-(X*exp(-arg1)) + X*exp(arg1))*
      -     exp(-arg2))*
      -     (-exp(arg3) - Z*exp(arg3)))))

FortranForm[A2]
List(gamma/(delta*exp(-arg1) + delta*exp(arg1)) +
delta*gamma*xm*
      -     (-(X*exp(-arg1)) +
      -     X*exp(arg1))*exp(arg2)*
      -     (-exp(-arg3) + Z*exp(-arg3))/
      -     ((delta*exp(-arg1) + delta*exp(arg1))*
      -     (-((-((delta*exp(-arg1) +
      -     delta*exp(arg1))*exp(arg2)) +
      -     delta*xm*
      -     (-(X*exp(-arg1)) + X*exp(arg1))*
      -     exp(arg2))*
      -     (-exp(-arg3) + Z*exp(-arg3))) +
      -     ((delta*exp(-arg1) + delta*exp(arg1))*
      -     exp(-arg2) +
      -     delta*xm*
      -     (-(X*exp(-arg1)) + X*exp(arg1))*
      -     exp(-arg2))*
      -     (-exp(arg3) - Z*exp(arg3)))) -
      -     delta*gamma*xm*
      -     (-(X*exp(-arg1)) + X*exp(arg1))*exp(-arg2)*
      -     (-exp(arg3) - Z*exp(arg3))/
      -     ((delta*exp(-arg1) + delta*exp(arg1))*
      -     (-((-((delta*exp(-arg1) +
      -     delta*exp(arg1))*exp(arg2)) +
      -     delta*xm*
      -     (-(X*exp(-arg1)) + X*exp(arg1))*
      -     exp(arg2))*
      -     (-exp(-arg3) + Z*exp(-arg3))) +
      -     ((delta*exp(-arg1) + delta*exp(arg1))*
      -     exp(-arg2) +
      -     delta*xm*
      -     (-(X*exp(-arg1)) + X*exp(arg1))*
      -     exp(-arg2))*
      -     (-exp(arg3) - Z*exp(arg3)))))

FortranForm[A3]
List(gamma*(-(X*exp(-arg1)) + X*exp(arg1))*
      -     (-exp(-arg3) + Z*exp(-arg3))/
      -     (-((-((delta*exp(-arg1) + delta*exp(arg1))*
      -     exp(arg2)) +
      -     delta*xm*
      -     (-(X*exp(-arg1)) + X*exp(arg1))*
```

```
      -     exp(arg2))*
      -     (-exp(-arg3) + Z*exp(-arg3))) +
      -     ((delta*exp(-arg1) + delta*exp(arg1))*
      -     exp(-arg2) +
      -     delta*xm*
      -     (-(X*exp(-arg1)) + X*exp(arg1))*
      -     exp(-arg2))*(-exp(arg3) -
      -     Z*exp(arg3))))

FortranForm[A4]
List(-(gamma*(-(X*exp(-arg1)) + X*exp(arg1))*
      -     (-exp(arg3) - Z*exp(arg3))/
      -     (-((-((delta*exp(-arg1) +
      -     delta*exp(arg1))*
      -     exp(arg2)) +
      -     delta*xm*
      -     (-(X*exp(-arg1)) + X*exp(arg1))*
      -     exp(arg2))*
      -     (-exp(-arg3) + Z*exp(-arg3))) +
      -     ((delta*exp(-arg1) + delta*exp(arg1))*
      -     exp(-arg2) +
      -     delta*xm*
      -     (-(X*exp(-arg1)) + X*exp(arg1))*
      -     exp(-arg2))*
      -     (-exp(arg3) - Z*exp(arg3)))))
```

```
C ***********************************************************************
C * Program model_flatsusp provides release profiles for the flat
C * membrane - suspension in reservoir problem
C *
C * Stephanie Farrell
C * August 1995
C *
C * © Copyright 1995, 1996 Stephanie Farrell
C *
C * FILE: FLATSUSP.FOR → FLATSUSP.EXE
C ***********************************************************************
      program MODEL_FLATSUSP

C ***********************************************************************
C * Function definitions
C ***********************************************************************
      real*8 REALTIME
      real*8 REALCONC

      complex*16 C16_F_WALL

      external C16_F_WALL
      external WALL
      external CALCULATION
      external DINLAP
      external REALTIME
      external REALCONC

C ***********************************************************************
C * Main program.
C ***********************************************************************
      integer I_INDEX
      integer I_KMAX
      integer I_NOUT

      real*8 R8_ALPHA
      real*8 R8_EXP
      real*8 R8_FLOAT
      real*8 R8_RELERR

      real*8 R8_T(1)
      real*8 R8_FINV(1)

C ***********************************************************************
C * Data input parameters
C ***********************************************************************
      character TITLE*80
      character VAR_NAME*30

      integer I_NUM_TIMES

      real*8 R8_TIME_INTERVAL
      real*8 R8_INIT_CONC
      real*8 R8_G_a
      real*8 R8_G_b
      real*8 R8_G_d1
      real*8 R8_G_d2
      real*8 R8_G_xm
      real*8 R8_G_vw
      real*8 R8_G_xm2
      real*8 R8_G_r
```

```
      real*8 R8_G_Cs0_C10
      real*8 R8_G_phi
      real*8 R8_G_k
      real*8 R8_G_xms

      character EOF*80

      common /IN_PARMS/
     &  R8_G_a,
     &  R8_G_b,
     &  R8_G_d1,
     &  R8_G_d2,
     &  R8_G_xm,
     &  R8_G_vw,
     &  R8_G_xm2,
     &  R8_G_r,
     &  R8_G_Cs0_C10,
     &  R8_G_phi,
     &  R8_G_k,
     &  R8_G_xms

C ***********************************************************************
C * Read in data
C ***********************************************************************
      read *, TITLE
      read *, VAR_NAME, I_NUM_TIMES
      read *, VAR_NAME, R8_TIME_INTERVAL
      read *, VAR_NAME, R8_INIT_CONC

      read *, VAR_NAME, R8_G_a
      read *, VAR_NAME, R8_G_b
      read *, VAR_NAME, R8_G_d1
      read *, VAR_NAME, R8_G_d2
      read *, VAR_NAME, R8_G_xm
      read *, VAR_NAME, R8_G_vw
      read *, VAR_NAME, R8_G_xm2
      read *, VAR_NAME, R8_G_r
      read *, VAR_NAME, R8_G_Cs0_C10
      read *, VAR_NAME, R8_G_phi
      read *, VAR_NAME, R8_G_k
      read *, VAR_NAME, R8_G_xms
      read *, EOF

      if (EOF .NE. 'EOF') then
         print *,'Input file format is incorrect. Aborting.'
         Goto 1000
      endif

      print *, TITLE
      print *, 'I_NUM_TIMES      ,', I_NUM_TIMES
      print *, 'R8_TIME_INTERVAL ,', R8_TIME_INTERVAL
      print *, 'R8_INIT_CONC     ,', R8_INIT_CONC
      print *, 'R8_G_a           ,', R8_G_a
      print *, 'R8_G_b           ,', R8_G_b
      print *, 'R8_G_d1          ,', R8_G_d1
      print *, 'R8_G_d2          ,', R8_G_d2
      print *, 'R8_G_xm          ,', R8_G_xm
      print *, 'R8_G_vw          ,', R8_G_vw
      print *, 'R8_G_xm2         ,', R8_G_xm2
      print *, 'R8_G_r           ,', R8_G_r
      print *, 'R8_G_Cs0_C10     ,', R8_G_Cs0_C10
```

```
      print *, 'R8_G_phi          ,', R8_G_phi
      print *, 'R8_G_k            ,', R8_G_k
      print *, 'R8_G_xms          ,', R8_G_xms

C *********************************************************************
C *   Arguments for the IMSL subroutine DINLAP
C *********************************************************************
      R8_ALPHA = 0
      I_KMAX = 10000
      R8_RELERR = 5.0E-5

      print *,'THETA,U,REAL TIME,REAL CONC.'

      Do I_INDEX = I_NUM_TIMES, 1, -1
         R8_T(1) =  R8_TIME_INTERVAL * Float(I_INDEX)

         call DINLAP(C16_F_WALL,
     &               1,
     &               R8_T,
     &               R8_ALPHA,
     &               R8_RELERR,
     &               I_KMAX,
     &               R8_FINV)


         print '(E9.4, A, E9.4, A, F15.2, A, E15.4)',
     &          R8_T(1), ',',
     &          R8_FINV(1), ',',
     &          REALTIME(R8_T(1), R8_G_b, R8_G_d1), ',',
     &          REALCONC(R8_FINV(1), R8_INIT_CONC)
      end do

 1000 continue
      end

C *********************************************************************
C * User-supplied function to which the inverse Laplace transform
C * will be computed
C *********************************************************************
      complex*16 function C16_F_WALL(C16_s)

      intrinsic CDEXP

      complex*16 C16_s
      complex*16 C16_arg3
      complex*16 C16_a3
      complex*16 C16_a4

      call WALL(C16_s, C16_arg3, C16_a3, C16_a4)

      C16_F_WALL = (C16_a3 * CDEXP(C16_arg3) + C16_a4 *
     &    CDEXP(-C16_arg3))

      return
      end

C *********************************************************************
C *     Subroutine WALL evaluates a3 and a4 which are used in F_WALL
C *********************************************************************
      subroutine WALL(C16_s, C16_arg3, C16_a3, C16_a4)
```

```
      intrinsic CDSQRT
      intrinsic CDEXP

      complex*16 C16_s
      complex*16 C16_gamma
      complex*16 C16_delta
      complex*16 C16_arg1
      complex*16 C16_arg2
      complex*16 C16_arg3
      complex*16 C16_X
      complex*16 C16_Z
      complex*16 C16_a3
      complex*16 C16_a4

      real*8 R8_XM
      real*8 R8_XM2

      call CALCULATION(C16_s,
     &                 C16_arg1,
     &                 C16_arg2,
     &                 C16_arg3,
     &                 C16_X,
     &                 C16_Z,
     &                 R8_xm,
     &                 R8_xm2,
     &                 C16_gamma,
     &                 C16_delta)

C ***********************************************************************
C * Compute a3
C ***********************************************************************
      C16_a3 = (C16_gamma * (-(C16_X * CDEXP(-C16_arg1)) + C16_X *
     &    CDEXP(C16_arg1)) *
     &    (-CDEXP(-C16_arg3) + C16_Z * CDEXP(-C16_arg3)) /
     &   (-((-((C16_delta * CDEXP(-C16_arg1) + C16_delta *
     &     CDEXP(C16_arg1)) *
     &             CDEXP(C16_arg2)) +
     &          C16_delta * R8_xm *
     &          (-(C16_X * CDEXP(-C16_arg1)) + C16_X * CDEXP(C16_arg1)) *
     &           CDEXP(C16_arg2))*
     &         (-CDEXP(-C16_arg3) + C16_Z * CDEXP(-C16_arg3))) +
     &   ((C16_delta * CDEXP(-C16_arg1) + C16_delta * CDEXP(C16_arg1)) *
     &         CDEXP(-C16_arg2) +
     &        C16_delta * R8_xm *
     &         (-(C16_X * CDEXP(-C16_arg1)) + C16_X * CDEXP(C16_arg1)) *
     &         CDEXP(-C16_arg2)) * (-CDEXP(C16_arg3) - C16_Z *
     &    CDEXP(C16_arg3))))

C ***********************************************************************
C * Compute a4
C ***********************************************************************
      C16_a4 = (-(C16_gamma * (-(C16_X * CDEXP(-C16_arg1)) + C16_X *
     &    CDEXP(C16_arg1)) *
     &      (-CDEXP(C16_arg3) - C16_Z * CDEXP(C16_arg3)) /
     &     (-((-((C16_delta * CDEXP(-C16_arg1) + C16_delta *
     &    CDEXP(C16_arg1)) *
     &               CDEXP(C16_arg2)) +
     &            C16_delta * R8_xm *
     &             (-(C16_X * CDEXP(-C16_arg1)) + C16_X *
     &    CDEXP(C16_arg1)) *
     &             CDEXP(C16_arg2)) *
```

```
     &           (-CDEXP(-C16_arg3) + C16_Z * CDEXP(-C16_arg3))) +
     &        ((C16_delta * CDEXP(-C16_arg1) + C16_delta *
     &    CDEXP(C16_arg1)) *
     &           CDEXP(-C16_arg2) +
     &          C16_delta * R8_xm *
     &         (-(C16_X * CDEXP(-C16_arg1)) + C16_X * CDEXP(C16_arg1)) *
     &           CDEXP(-C16_arg2))*
     &        (-CDEXP(C16_arg3) - C16_Z * CDEXP(C16_arg3)))))
      end

C ***********************************************************************
C * Subroutine CALCULATION computes everything necessary for the
C * evaluation of a3 and a4.
C ***********************************************************************
      subroutine CALCULATION (C16_s,
     &                        C16_arg1,
     &                        C16_arg2,
     &                        C16_arg3,
     &                        C16_X,
     &                        C16_Z,
     &                        R8_xm,
     &                        R8_xm2,
     &                        C16_gamma,
     &                        C16_delta)

C ***********************************************************************
C * Constants
C ***********************************************************************
      real*8 R8_pi
      parameter (R8_pi = 3.14159265359)

C***********************************************************************
C Commons
C***********************************************************************
      real*8 R8_G_a
      real*8 R8_G_b
      real*8 R8_G_d1
      real*8 R8_G_d2
      real*8 R8_G_xm
      real*8 R8_G_vw
      real*8 R8_G_xm2
      real*8 R8_G_r
      real*8 R8_G_Cs0_C10
      real*8 R8_G_phi
      real*8 R8_G_k
      real*8 R8_G_xms

      common /IN_PARMS/
     &  R8_G_a,
     &  R8_G_b,
     &  R8_G_d1,
     &  R8_G_d2,
     &  R8_G_xm,
     &  R8_G_vw,
     &  R8_G_xm2,
     &  R8_G_r,
     &  R8_G_Cs0_C10,
     &  R8_G_phi,
     &  R8_G_k,
     &  R8_G_xms
```

```
C ***********************************************************************
C * Variables
C ***********************************************************************
      complex*16 C16_s
      complex*16 C16_arg1
      complex*16 C16_arg2
      complex*16 C16_arg3
      complex*16 C16_X
      complex*16 C16_Z
      complex*16 C16_gamma
      complex*16 C16_delta

      real*8 R8_A
      real*8 R8_B
      real*8 R8_D1
      real*8 R8_D2
      real*8 R8_XM
      real*8 R8_XM2
      real*8 R8_VW
      real*8 R8_R
      real*8 R8_AREA
      real*8 R8_CS0_C10
      real*8 R8_PHI
      real*8 R8_XMS
      real*8 R8_V1
      real*8 R8_ALPHA
      real*8 R8_BETA
      real*8 R8_k

C ***********************************************************************
C * Computations
C ***********************************************************************
      R8_a      =  R8_G_a              ! Inside distance to membrane
      R8_b      =  R8_G_b              ! Outside distance
      R8_d1     =  R8_G_d1             ! Diffusivity in reservoir
      R8_d2     =  R8_G_d2             ! Effective membrane diffusivity
      R8_xm     =  R8_G_xm             ! Partition coefficient at A
      R8_xm2    =  R8_G_xm2            ! Partition coefficient at B
      R8_vw     =  R8_G_vw             ! External aqueous phase volume
      R8_R      =  R8_G_r              ! Radius of flat membrane
      R8_area   =  R8_pi * R8_R ** 2 ! Mass transfer area

      R8_Cs0_C10 = R8_G_Cs0_C10
      R8_phi     = R8_G_phi
      R8_k       = R8_G_k
      R8_xms     = R8_G_xms

      R8_v1 = R8_pi * R8_R ** 2 * R8_a
      R8_alpha = R8_phi / (1.0 - R8_phi)
      R8_beta = R8_k * R8_b ** 2 / (R8_v1 * R8_phi * R8_d1)
      C16_gamma = -1.0 - R8_alpha * R8_Cs0_C10 + R8_alpha * C16_s *
     &     R8_Cs0_C10 / (C16_s + R8_beta * R8_xms)
      C16_delta = C16_s * (1.0 + R8_beta * R8_alpha / (C16_s + R8_beta *
     &     R8_xms))
      C16_X = R8_d1 * CDSQRT(C16_delta) / (R8_d2 * CDSQRT(C16_s * R8_d1
     &     / R8_d2))
      C16_Z = R8_area * R8_b * R8_xm2 / (R8_vw * CDSQRT(C16_s * R8_d1 /
     &     R8_d2))

C ***********************************************************************
C *     Arguments
```

```
C ***********************************************************************
      C16_arg1 = R8_a * CDSQRT(C16_delta) / R8_b
      C16_arg2 = R8_a * CDSQRT(C16_s * R8_d1 / R8_d2) / R8_b
      C16_arg3 = CDSQRT(C16_s * R8_d1 / R8_d2)

      return
      end

C ***********************************************************************
C * Function REALTIME converts to real time.
C ***********************************************************************
      real*8 function REALTIME(R8_THETA, R8_b, R8_d1)

      real*8 R8_THETA
      real*8 R8_b
      real*8 R8_d1

      REALTIME = R8_THETA * (R8_b ** 2) / R8_d1

      return
      end

C ***********************************************************************
C * Function REALCONC converts to real concentration.
C ***********************************************************************
      real*8 function REALCONC(R8_U, R8_INIT_CONC)

      real*8 R8_U
      real*8 R8_INIT_CONC

      REALCONC = R8_U * R8_INIT_CONC

      return
      end
C ***********************************************************************
C * EOF
C ***********************************************************************
```

Table 12. Nomenclature Equivalents for the "Hollow Fiber: Dispersed Phase in Reservoir" Problem.

Text	Mathematica®	Fortran
$\frac{a}{b}\sqrt{\delta}$	arg1	C16_arg1
$\frac{a}{b}\sqrt{\frac{D_1}{D_2}s}$	arg2	C16_arg2
$\sqrt{\frac{D_1}{D_2}s}$	arg3	C16_arg3
C_1^0		R8_G_INIT_CONC
$\frac{V_w}{\alpha_2 b m_{2,w}}\sqrt{\frac{sD_1}{D_2}}$	Z	C16_Z
$\sqrt{\frac{D_1\delta}{sD_2}}$	X	C16_X
$\frac{C_s^0}{C_1^0}$		R8_G_CS0_C10
a		R8_G_a
A_{15}	a1	
A_{16}	a2	
A_{17}	a3	C16_a3
A_{18}	a4	C16_a4
b		R8_G_b
δ	delta	C16_delta
D1	d1	R8_G_d1
D_2	d2	R8_G_d2
ϕ		R8_G_phi
γ	gamma	C16_gamma
k		R8_G_k
$m_{1,2}$	xm	R8_G_xm
$m_{1,s}$		R8_g_xms
$m_{2,w}$		R8_g_xm2
V_w		R8_g_vw

Hollow Fiber: Dispersed Phase in Reservoir

Presented below is the Mathematica® program used to determine expressions for the coefficients A_{19}, A_{21}, and A_{22} for the "hollow fiber: dispersed phase in reservoir" problem. It is followed by the Fortran code used to invert the solution numerically from the Laplace domain into the time domain. The reader is referred to Table 12 for relationships between variables used in the text in Chapter 2 and those used in the Mathematica® and Fortran programs.

```
const = Solve[{-gamma/delta + A1 I0arg1 ==

      xm (A3 I0arg2 + A4 K0arg2),
      X A1 I1arg1 ==
      A3 I1arg2 - A4 K1arg2,
      Z (A3 I0arg3 + A4 K0arg3) ==
          -(A3 I1arg3 - A4 K1arg3)},{A1, A3, A4}];
General::spell1:
   Possible spelling error: new symbol name "gamma"
     is similar to existing symbol "Gamma".
General::spell1:
   Possible spelling error: new symbol name "K0arg2"
     is similar to existing symbol "I0arg2".
General::spell1:
   Possible spelling error: new symbol name "K1arg2"
     is similar to existing symbol "I1arg2".
General::stop:
   Further output of General::spell1
     will be suppressed during this calculation.

A1 = A1 /. Const;
a3 = A3 /. Const;
a4 = A4 /. Const;

FortranForm[a1]
List(gamma/(delta*I0arg1) -
      -     gamma*I1arg1*K0arg2*X*xm*(I1arg3 +
      -     I0arg3*Z)/
      -     (I0arg1*((delta*I0arg1*K1arg2 +
      -     delta*I1arg1*K0arg2*X*xm)*
      -     (I1arg3 + I0arg3*Z) -
      -     (-(delta*I0arg1*I1arg2) +
      -     delta*I0arg2*I1arg1*X*xm)*
      -     (-K1arg3 + K0arg3*Z))) +
      -     gamma*I0arg2*I1arg1*X*xm*
      -     (-K1arg3 + K0arg3*Z)/
      -     (I0arg1*((delta*I0arg1*K1arg2 +
      -     delta*I1arg1*K0arg2*X*xm)*
      -     (I1arg3 + I0arg3*Z) -
      -     (-(delta*I0arg1*I1arg2) +
      -     delta*I0arg2*I1arg1*X*xm)*
      -     (-K1arg3 + K0arg3*Z))))

FortranForm[a3]
List(gamma*I1arg1*X*(-K1arg3 + K0arg3*Z)/
      -     ((delta*I0arg1*K1arg2 +
      -     delta*I1arg1*K0arg2*X*xm)*
      -     (I1arg3 + I0arg3*Z) -
      -     (-(delta*I0arg1*I1arg2) +
      -     delta*I0arg2*I1arg1*X*xm)*
```

```
      -     (-K1arg3 + K0arg3*Z)))

FortranForm[a4]
List(-(gamma*I1arg1*X*(I1arg3 + I0arg3*Z)/
      -     ((delta*I0arg1*K1arg2 +
      -     delta*I1arg1*K0arg2*X*xm)*
      -     (I1arg3 + I0arg3*Z) -
      -     (-(delta*I0arg1*I1arg2) +
      -     delta*I0arg2*I1arg1*X*xm)*
      -     (-K1arg3 + K0arg3*Z))))
```

```
C ***********************************************************************
C * Program model_mhfsusp provides release profiles for the hollow
C * fiber - suspension in reservoir problem.
C *
C * Stephanie Farrell
C * August 1995
C *
C * © Copyright 1995, 1996 Stephanie Farrell
C *
C * FILE: MHFSUSP.FOR → MHFSUSP.EXE
C ***********************************************************************
      program MODEL_MHFSUSP

C ***********************************************************************
C * Function definitions
C ***********************************************************************
      real*8 R8_REALTIME
      real*8 R8_REALCONC

      complex*16 C16_F_WALL

      external C16_F_WALL
      external WALL
      external CALCULATION
      external DINLAP
      external BESK0
      external BESI0
      external BESI1
      external BESK1
      external R8_REALTIME
      external R8_REALCONC

C ***********************************************************************
C * Main program.
C ***********************************************************************
      integer I_INDEX
      integer I_KMAX
      integer I_NOUT

      real*8 R8_ALPHA
      real*8 R8_RELERR

      real*8 R8_T(1)
      real*8 R8_FINV(1)

      real*8 R8_RADIUS
      common R8_RADIUS

C ***********************************************************************
C * Data input parameters
C ***********************************************************************
      character TITLE*80
      character VAR_NAME*30

      integer I_NUM_TIMES

      real*8 R8_TIME_INTERVAL
      real*8 R8_INIT_CONC
      real*8 R8_G_a
      real*8 R8_G_b
      real*8 R8_G_d1
```

```
      real*8 R8_G_d2
      real*8 R8_G_xm
      real*8 R8_G_vw
      real*8 R8_G_xm2
      real*8 R8_G_Cs0_C10
      real*8 R8_G_phi
      real*8 R8_G_k
      real*8 R8_G_xms

      character EOF*80

      common /IN_PARMS/
     &  R8_G_a,
     &  R8_G_b,
     &  R8_G_d1,
     &  R8_G_d2,
     &  R8_G_xm,
     &  R8_G_vw,
     &  R8_G_xm2,
     &  R8_G_Cs0_C10,
     &  R8_G_phi,
     &  R8_G_k,
     &  R8_G_xms

C ***************************************************************************
C * Read in data
C ***************************************************************************
      read *, TITLE
      read *, VAR_NAME, I_NUM_TIMES
      read *, VAR_NAME, R8_TIME_INTERVAL
      read *, VAR_NAME, R8_INIT_CONC

      read *, VAR_NAME, R8_G_a
      read *, VAR_NAME, R8_G_b
      read *, VAR_NAME, R8_G_d1
      read *, VAR_NAME, R8_G_d2
      read *, VAR_NAME, R8_G_xm
      read *, VAR_NAME, R8_G_vw
      read *, VAR_NAME, R8_G_xm2
      read *, VAR_NAME, R8_G_Cs0_C10
      read *, VAR_NAME, R8_G_phi
      read *, VAR_NAME, R8_G_k
      read *, VAR_NAME, R8_G_xms
      read *, EOF

      if (EOF .NE. 'EOF') then
         print *,'Input file format is incorrect. Aborting.'
         Goto 1000
      endif

      print *, TITLE
      print *, 'I_NUM_TIMES      ,', I_NUM_TIMES
      print *, 'R8_TIME_INTERVAL ,', R8_TIME_INTERVAL
      print *, 'R8_INIT_CONC     ,', R8_INIT_CONC
      print *, 'R8_G_a           ,', R8_G_a
      print *, 'R8_G_b           ,', R8_G_b
      print *, 'R8_G_d1          ,', R8_G_d1
      print *, 'R8_G_d2          ,', R8_G_d2
      print *, 'R8_G_xm          ,', R8_G_xm
      print *, 'R8_G_vw          ,', R8_G_vw
      print *, 'R8_G_xm2         ,', R8_G_xm2
```

```
      print *, 'R8_G_Cs0_C10      ,', R8_G_Cs0_C10
      print *, 'R8_G_phi          ,', R8_G_phi
      print *, 'R8_G_k            ,', R8_G_k
      print *, 'R8_G_xms          ,', R8_G_xms


C ***********************************************************************
C * User-supplied arguments for the IMSL subroutine DINLAP
C ***********************************************************************
      R8_ALPHA = 0
      I_KMAX = 10000
      R8_RELERR = 5.0E-5

      print *,'THETA,U,REAL TIME,REAL CONC.'
      Do I_INDEX = I_NUM_TIMES, 1, -1

          R8_T(1) = R8_TIME_INTERVAL * Float(I_INDEX)

          call DINLAP(C16_F_WALL,
     &                1,
     &                R8_T,
     &                R8_ALPHA,
     &                R8_RELERR,
     &                I_KMAX,
     &                R8_FINV)


          print '(E9.4, A, E9.4, A, F15.2, A, E15.4)',
     &           R8_T(1), ',',
     &           R8_FINV(1), ',',
     &           R8_REALTIME(R8_T(1), R8_G_b, R8_G_d1), ',',
     &           R8_REALCONC(R8_FINV(1), R8_INIT_CONC)
       end do

 1000 continue
      end

C ***********************************************************************
C * User-supplied function to which the inverse Laplace transform
C * will be computed
C ***********************************************************************
      complex*16 function C16_F_WALL(C16_s)

      intrinsic CDSQRT

      complex*16 C16_I0argWall
      complex*16 C16_K0argWall
      complex*16 C16_argWall

      complex*16 C16_s
      complex*16 C16_a1
      complex*16 C16_a3
      complex*16 C16_a4

      real*8 R8_D1
      real*8 R8_D2

      call WALL(C16_s, R8_d1, R8_d2, C16_a3, C16_a4)

      C16_argWall = CDSQRT(C16_s * R8_d1 / R8_d2)
```

```
      call BESI0(C16_argWall, C16_I0argWall)
      call BESK0(C16_argWall, C16_K0argWall)

      C16_F_WALL =  (C16_a3 * C16_I0argWall + C16_a4 * C16_K0argWall)

      return
      end

C ***********************************************************************
C * Subroutine BESI0 computes Bessel(I,0)
C ***********************************************************************
      subroutine BESI0(C16_X, C16_I0)

      complex*16 C16_X
      complex*16 C16_I0

      C16_I0 = 1.0 + C16_X**2 / 4.0 + C16_X**4 / 64.0 +
     &   C16_X**6  / 2304.0 + C16_X**8 / 147456.0 +
     &   C16_X**10 / 14745600.0 + C16_X**12 / 2123366400.0 +
     &   C16_X**14 / 416179814400.0 +
     &   C16_X**16 / 106542032486400.0 +
     &   C16_X**18 / 34519618525593600.0 +
     &   C16_X**20 / 13807847410237440000.0

      return
      end

C ***********************************************************************
C * Subroutine BESI1 computes Bessel(I,1)
C ***********************************************************************
      subroutine BESI1(C16_X, C16_I1)

      complex*16 C16_X
      complex*16 C16_I1

      C16_I1 = C16_X/2.0 + C16_X**3/16.0 +
     &     C16_X**5/384.0 + C16_X**7/18432.0 +
     &     C16_X**9/1474560.0 + C16_X**11/176947200.0 +
     &     C16_X**13/29727129600.0 +
     &     C16_X**15/6658877030400.0 +
     &     C16_X**17/1917756584755200.0 +
     &     C16_X**19/690392370511872000.0

      return
      end

C ***********************************************************************
C * Subroutine BESK0 computes Bessel(K,0)
C ***********************************************************************
      subroutine BESK0(C16_X, C16_K0)

      complex*16 C16_X
      complex*16 C16_XL
      complex*16 C16_K0

      intrinsic CDLOG

      C16_XL = LOG(1.0/2.0) + CDLOG(C16_X)

      C16_K0 = -0.577215 - LOG(1.0/2.0) +
     &    C16_X**20*((7381.0/1260.0 - 2*0.577215)/2761569482047488000.0
```

```
     &     + (-C16_XL)/13807847410237440000.0) +
     &     C16_X**18*((7129.0/1260.0 - 2*0.577215)/69039237051187200.0 +
     &     (-C16_XL)/34519618525593600.0) +
     &     C16_X**16*((761.0/140.0 - 2*0.577215)/213084064972800.0 +
     &     (-C16_XL)/106542032486400.0) +
     &     C16_X**14*((363.0/70.0 - 2*0.577215)/832359628800.0 +
     &     (-C16_XL)/416179814400.0) +
     &     C16_X**12*((49.0/10.0 - 2*0.577215)/4246732800.0 +
     &     (-C16_XL)/2123366400.0) +
     &     C16_X**10*((137.0/30.0 - 2*0.577215)/29491200.0 +
     &     (-C16_XL)/14745600.0) +
     &     C16_X**8*((25.0/6.0 - 2*0.577215)/294912.0 +
     &     (-C16_XL)/147456.0) +
     &     C16_X**6*((11.0/3.0 - 2*0.577215)/4608.0 + (-C16_XL)/2304.0) +
     &     C16_X**4*((3.0 - 2*0.577215)/128.0 + (-C16_XL)/64.0) +
     &     C16_X**2*((2.0 - 2*0.577215)/8.0 + (-C16_XL)/4.0) -
     &     CDLOG(C16_X)

      return
      end

C ***********************************************************************
C * Subroutine BESK1 computes Bessel(K,1)
C ***********************************************************************
      subroutine BESK1(C16_X, C16_K1)

      complex*16 C16_X
      complex*16 C16_XL
      complex*16 C16_K1

      intrinsic CDLOG

      C16_XL = LOG(1.0/2.0) + CDLOG(C16_X)

      C16_K1 = 1.0/C16_X + C16_X**17*((-6989.0/1260.0 + 2*0.577215)/
     &  3835513169510400.0 + C16_XL/1917756584755200.0) +
     &  C16_X**15*((-1487.0/280.0 + 2*0.577215)/13317754060800.0 +
     &  C16_XL/6658877030400.0) +
     &  C16_X**13*((-353.0/70.0 + 2*0.577215)/59454259200.0 +
     &  C16_XL/29727129600.0) +
     &  C16_X**11*((-71.0/15.0 + 2*0.577215)/353894400.0 +
     &  C16_XL/176947200.0) +
     &  C16_X**9*((-131.0/30.0 + 2*0.577215)/2949120.0 +
     &  C16_XL/1474560.0) +
     &  C16_X**7*((-47.0/12.0 + 2*0.577215)/36864.0 + C16_XL/18432.0) +
     &  C16_X**5*((-10.0/3.0 + 2*0.577215)/768.0 + C16_XL/384.0) +
     &  C16_X**3*((-5.0/2.0 + 2*0.577215)/32.0 + C16_XL/16.0) +
     &  C16_X*((-1.0 + 2*0.577215)/4.0 +  C16_XL/2.0)

      return
      end

C ***********************************************************************
C * Subroutine WALL evaluates a3 and a4 which are used in C16_F_WALL
C ***********************************************************************
      subroutine WALL(C16_s, R8_d1, R8_d2, C16_a3, C16_a4)

      complex*16 C16_s
      complex*16 C16_I0arg1
      complex*16 C16_I0arg2
      complex*16 C16_I0arg3
```

```
      complex*16 C16_I1arg1
      complex*16 C16_I1arg2
      complex*16 C16_I1arg3
      complex*16 C16_K0arg2
      complex*16 C16_K0arg3
      complex*16 C16_K1arg2
      complex*16 C16_K1arg3
      complex*16 C16_arg1
      complex*16 C16_arg2
      complex*16 C16_arg3
      complex*16 C16_Z
      complex*16 C16_a3
      complex*16 C16_a4
      complex*16 C16_gamma
      complex*16 C16_delta
      complex*16 C16_X

      real*8 R8_D1
      real*8 R8_D2
      real*8 R8_VW
      real*8 R8_V1
      real*8 R8_A
      real*8 R8_B
      real*8 R8_XM
      real*8 R8_XM2
      real*8 R8_AREA

C ***************************************************************************
C * Call CALCULATION to calculate everything necessary for
C * evaluation of a3 and a4
C ***************************************************************************
      call CALCULATION (C16_s,
     &                  C16_I0arg1,
     &                  C16_I0arg2,
     &                  C16_I0arg3,
     &                  C16_I1arg1,
     &                  C16_I1arg2,
     &                  C16_I1arg3,
     &                  C16_K0arg2,
     &                  C16_K0arg3,
     &                  C16_K1arg2,
     &                  C16_K1arg3,
     &                  C16_arg1,
     &                  C16_arg2,
     &                  C16_arg3,
     &                  R8_d1,
     &                  R8_d2,
     &                  R8_vw,
     &                  R8_v1,
     &                  R8_a,
     &                  R8_b,
     &                  C16_X,
     &                  C16_Z,
     &                  R8_xm,
     &                  R8_xm2,
     &                  R8_area,
     &                  C16_gamma,
     &                  C16_delta)

C ***************************************************************************
C * Compute a3
```

```
C ***********************************************************************
      C16_a3 = (C16_gamma * C16_I1arg1 * C16_X * (-C16_K1arg3 +
     &     C16_K0arg3 * C16_Z) /
     &   ((C16_delta * C16_I0arg1 * C16_K1arg2 +
     &       C16_delta * C16_I1arg1 * C16_K0arg2 * C16_X * R8_xm) *
     &     (C16_I1arg3 + C16_I0arg3 * C16_Z) -
     &     (-(C16_delta * C16_I0arg1 * C16_I1arg2) +
     &       C16_delta * C16_I0arg2 * C16_I1arg1 * C16_X * R8_xm) *
     &      (-C16_K1arg3 + C16_K0arg3 * C16_Z)))

C ***********************************************************************
C * Compute a4
C ***********************************************************************
      C16_a4 = (-(C16_gamma * C16_I1arg1 * C16_X * (C16_I1arg3 +
     &    C16_I0arg3 * C16_Z) /
     &     ((C16_delta * C16_I0arg1 * C16_K1arg2 +
     &         C16_delta * C16_I1arg1 * C16_K0arg2 * C16_X * R8_xm) *
     &       (C16_I1arg3 + C16_I0arg3 * C16_Z) -
     &       (-(C16_delta * C16_I0arg1 * C16_I1arg2) +
     &         C16_delta * C16_I0arg2 * C16_I1arg1 * C16_X * R8_xm) *
     &        (-C16_K1arg3 + C16_K0arg3 * C16_Z))))

      end

C ***********************************************************************
C * Subroutine CALCULATION computes everything necessary for the
C * evaluation of a3, and a4.
C ***********************************************************************
      subroutine CALCULATION (C16_s,
     &                        C16_I0arg1,
     &                        C16_I0arg2,
     &                        C16_I0arg3,
     &                        C16_I1arg1,
     &                        C16_I1arg2,
     &                        C16_I1arg3,
     &                        C16_K0arg2,
     &                        C16_K0arg3,
     &                        C16_K1arg2,
     &                        C16_K1arg3,
     &                        C16_arg1,
     &                        C16_arg2,
     &                        C16_arg3,
     &                        R8_d1,
     &                        R8_d2,
     &                        R8_vw,
     &                        R8_v1,
     &                        R8_a,
     &                        R8_b,
     &                        C16_X,
     &                        C16_Z,
     &                        R8_xm,
     &                        R8_xm2,
     &                        R8_area,
     &                        C16_gamma,
     &                        C16_delta)

C ***********************************************************************
C * Constants
C ***********************************************************************
      real*8 R8_pi
      parameter (R8_pi = 3.14159265359)
```

```
C*****************************************************************************
C Commons
C*****************************************************************************
      real*8 R8_G_a
      real*8 R8_G_b
      real*8 R8_G_d1
      real*8 R8_G_d2
      real*8 R8_G_xm
      real*8 R8_G_vw
      real*8 R8_G_xm2
      real*8 R8_G_Cs0_C10
      real*8 R8_G_phi
      real*8 R8_G_k
      real*8 R8_G_xms

      common /IN_PARMS/
     &  R8_G_a,
     &  R8_G_b,
     &  R8_G_d1,
     &  R8_G_d2,
     &  R8_G_xm,
     &  R8_G_vw,
     &  R8_G_xm2,
     &  R8_G_Cs0_C10,
     &  R8_G_phi,
     &  R8_G_k,
     &  R8_G_xms

C *****************************************************************************
C * Variables
C *****************************************************************************
      complex*16 C16_s
      complex*16 C16_arg1
      complex*16 C16_arg2
      complex*16 C16_arg3
      complex*16 C16_I0arg1
      complex*16 C16_I0arg2
      complex*16 C16_I0arg3
      complex*16 C16_I1arg1
      complex*16 C16_I1arg2
      complex*16 C16_I1arg3
      complex*16 C16_K0arg2
      complex*16 C16_K0arg3
      complex*16 C16_K1arg2
      complex*16 C16_K1arg3
      complex*16 C16_Z
      complex*16 C16_a1
      complex*16 C16_a3
      complex*16 C16_a4
      complex*16 C16_gamma
      complex*16 C16_delta
      complex*16 C16_X

      real*8 R8_a
      real*8 R8_b
      real*8 R8_d1
      real*8 R8_d2
      real*8 R8_vw
      real*8 R8_area
      real*8 R8_xm2
```

```
      real*8 R8_phi
      real*8 R8_xm
      real*8 R8_xms
      real*8 R8_v1
      real*8 R8_alpha
      real*8 R8_beta
      real*8 R8_k
      real*8 R8_cs0_c10

C *****************************************************************
C * Computations
C *****************************************************************
      R8_a                = R8_G_a          ! Inside radius
      R8_b                = R8_G_b          ! Outside radius
      R8_d1               = R8_G_d1         ! Diffusivity in reservoir
      R8_d2               = R8_G_d2         ! Effective membrane diffusiv
      R8_xm               = R8_G_xm         ! Partition coefficient at A
      R8_vw               = R8_G_vw         ! External aqueous phase volume
      R8_xm2              = R8_G_xm2
      R8_Cs0_C10          = R8_G_Cs0_C10
      R8_phi              = R8_G_phi
      R8_k                = R8_G_k
      R8_xms              = R8_G_xms

      R8_area = 2.0 * R8_pi * R8_b                ! Outside area

      R8_v1 = R8_pi * R8_a**2
      R8_alpha = R8_phi / (1.0 - R8_phi)
      R8_beta = R8_k * R8_b**2 / (R8_v1 * R8_phi * R8_d1)
      C16_gamma = -1.0 - R8_alpha * R8_Cs0_C10 + R8_alpha * C16_s *
     &    R8_Cs0_C10 / (C16_s + R8_beta * R8_xms)
      C16_delta = C16_s * (1.0 + R8_beta * R8_alpha / (C16_s + R8_beta *
     &    R8_xms))

      C16_X = R8_d1 * CDSQRT(C16_delta) / (R8_d2 * CDSQRT(C16_s * R8_d1
     &    / R8_d2))
      C16_Z = R8_vw * CDSQRT(C16_s * R8_d1 / R8_d2) / (R8_area * R8_b *
     &    R8_xm2)

C *****************************************************************
C * Arguments of the Bessel functions that appear in the expressions
C * for a1, a3 and a4
C *****************************************************************
      C16_arg1 = R8_a * CDSQRT(C16_delta) / R8_b
      C16_arg2 = R8_a * CDSQRT(C16_s * R8_d1 / R8_d2) / R8_b
      C16_arg3 = CDSQRT(C16_s * R8_d1 / R8_d2)

C *****************************************************************
C * Evaluate the Bessel functions.
C *****************************************************************
      call BESI0(C16_arg1, C16_I0arg1)
      call BESI0(C16_arg2, C16_I0arg2)
      call BESI0(C16_arg3, C16_I0arg3)

      call BESI1(C16_arg1, C16_I1arg1)
      call BESI1(C16_arg2, C16_I1arg2)
      call BESI1(C16_arg3, C16_I1arg3)

      call BESK0(C16_arg2, C16_K0arg2)
      call BESK0(C16_arg3, C16_K0arg3)
```

```
      call BESK1(C16_arg2, C16_K1arg2)
      call BESK1(C16_arg3, C16_K1arg3)

      return
      end

C ***************************************************************************
C * Subroutine R8_REALTIME converts to real time.
C ***************************************************************************
      real*8 function R8_REALTIME(R8_THETA, R8_b, R8_d1)

      real*8 R8_THETA
      real*8 R8_b
      real*8 R8_d1

      R8_REALTIME = R8_THETA * (R8_b ** 2) / R8_d1

      return
      end

C ***************************************************************************
C * Subroutine REALCONC converts to real concentration.
C ***************************************************************************
      real*8 function R8_REALCONC(R8_U, R8_INIT_CONC)

      real*8 R8_U
      real*8 R8_INIT_CONC

      R8_REALCONC = R8_U * R8_INIT_CONC

      return
      end

C ***************************************************************************
C * EOF
C ***************************************************************************
```

APPENDIX 3

Following is a sample calculation for a single data point taken from Figure 27 (The release profile of benzoic acid from a nylon hollow fiber with water-filled pores. Benzoic acid was initially present in a concentration of 100 mg/ml; the fiber length was 12.0 cm. Experimental parameters, taken directly from the data file read by the Fortran simulation program, are shown below.

R8_INIT_CONC	100.0000
R8_G_a	0.0300
R8_G_b	0.0500
R8_G_d1	8.6000e-6
R8_G_d2	9.0000e-6
R8_G_xm	88.0000
R8_G_vw	100.0000
R8_G_xm2	1.0000

The concentration of benzoic acid in the surrounding water bath is measured by HPLC, and found to be 0.031 mg/ml after 95 minutes. The total amount of agent released per unit fiber length (M_t/l)is then found:

$$\frac{M_t}{l} = \frac{C_w V_w}{l} \qquad \frac{M_t}{l} = \frac{\left(0.031 \frac{mg}{ml}\right)(100\, ml)}{12.0\, cm} = 0.258 \frac{mg}{cm}$$

On output, the Fortran program gives the concentration of the agent per unit fiber length at the outer surface of the membrane ($\frac{C_w|_b}{l}$), at the time points specified on input. From this it is simple to calculate the total amount of agent released. The concentration of agent per unit fiber length in the

water bath is related to the concentration per unit length at the outer surface of the membrane by the distribution coefficient $m_{2,w}$. At 75 minutes, the concentration per unit length predicted by the model is 0.243 mg/cm

$$\frac{C_w}{l} = \frac{C_2|_b}{l\, m_{2,w}} \qquad \frac{C_w}{l} = \frac{0.00243\ \frac{mg}{ml\ cm}}{1.0} = 0.00243\frac{mg}{ml\ cm}$$

The amount released per unit fiber length at time t, M_t/l is related to C_w/l by

$$\frac{M_t}{l} = \left(\frac{C_w}{l}\right)V_w \qquad \frac{M_t}{l} = \left(0.00243\frac{mg}{ml\ cm}\right)(100\ ml) = 0.243\frac{mg}{cm}$$

REFERENCES

1 R. Langer, "New methods of drug delivery," *Science*, 249 (1990), 1527.

2 A. Zaffaroni, "Novel drug delivery device", United States Patent, 3,993,073 (1976).

3 J. Urquhart, K. Chandrasekaran, J. Shaw, "Bandage for transdermally administering scopolamine to prevent nausea," United States Patent, 4,031,894 (1977).

4 J. Urquhart, K. Chandrasekaran, J. Shaw, "Method and therapeutic system for administering scopolamine transdermally," United States Patent, 4,262,003 (1981).

5 G. Flynn and R. Smith, "Membrane diffusion III: influence of solvent composition and permeant solubility on membrane transport," *J. Pharm. Sci..*, 61, 1 (1972) 61.

6 F. Theeuwes, R. Gale, and R. Baker, "Transference: A comprehensive parameter governing permeation of solutes through membranes," *J. Membrane Sci.*, 1 (1976), 3.

7 R. Dunn, D. Lewis, and L. Beck, "Fibrous polymer for the delivery of contraceptive steroids to the female reproductive tract," in: D. H. Lewis, (Ed.), *Controlled Release of Pesticides and Pharmaceuticals*, Plenum Press, New York, 1981, p. 125.

8 M. Eenink, J. Feinjen, J. Olijslager, J. Albers, J. Riecke and P. Greidanus, "Biodegradable hollow fibers for the controlled release of hormones," *J. Control. Rel.*, 6 (1987) 225.

9 G. Wong, V. C. Stent, and R. A. Stanley, "Supported liquid membranes for peptide separation", presented at ICOM, Heidelberg, Germany, 1993.

10 M. Mezei, *Liposomes and the Skin*, in A. Florence, H. Patel and G. Gregoriadis, *Liposomes in Drug Delivery*, Hardwood Academic Publishers, Langhorne, PA, (1993).

11 D. Lasic, Liposomes, *American Scientist*, 80 (1985) 20.

12 D. Lasic and D. Papahadjopoulos, Liposomes Revisited, Science, 267 (1995), 1275.

REFERENCES
(Continued)

13 D. Lasic and Y. Barenholz, *Handbook of Nonmedical Applications of Liposomes,* CRC Press, Boca Raton, I-IV(1996).

14 S. Penner and S. Sherman, "Heat flow through composite cylinders," *J. Chem. Phys.* 15, No.8 (1947), 569.

15 J. Jaeger, "Heat conduction in composite circular cylinders," *Phil. Mag.* 32, No. 213 (1941), 324.

16 C. Tranter, "Heat flow in an infinite medium heated by a cylinder," *Phil. Mag,* 38 (1947): 131.

17 R. Barrer, "Formal theory of diffusion through membranes", in H. Hopfenberg (Ed.), *Permeability of Plastic Films and Coatings*, Plenum Publishing, New York, 1974, pp. 113-123.

18 F. De Hoog, J. Knight, and A. Stokes, "An improved method for numerical inversion of Laplace transforms," *SIAM J. Sci. Stat. Comput.* 3 No.3 (1982): 357.

19 C. Varelas, D. Dixon and C. Steiner, "Mathematical Model of Mass Transport through Dispersed-Phase Polymer Networks," *AIChE J.,* 41 (April 1995): 805-811.

20 T.Papadopoulos, Personal Correspondence, Stevens Institute of Technology, Hoboken, NJ, 1992.

21 R. Prasad and K. Sirkar, "Dispersion-free solvent extraction with microporous hollow-fiber modules," *AIChE J.,* 34 (1988) 177.

22 R. Prasad and K. K. Sirkar, "Solvent extraction with microporous hydrophilic and composite membranes", *AIChE J.,* 33, No.7 (1987) 1057.

23 C. K. Colton, K. A. Smith, E. W. Merrill and P. C. Farrell, "Permeability studies with cellulosic membranes," *J. Biomed. Mater. Res.,* 5 (1971) 459.

24 R. A. Moss and S. Bhattacharya, "Transverse membrane asymmetry in model phospholipid bilayers: NBD-phosphatidylethanolamine and the separation of flip from flop", *J. Amer. Chem. Soc.*, 117, No.33, (1995), 8688.

REFERENCES
(Continued)

25 S. Bhattacharya, Personal Correspondence, Rutgers University, Piscataway, NJ, 1996.

26 P. Shanbhag, "Kinetic studies of 2-phase ozonation of organic pollutants in wastewater", M.E. thesis, Stevens Institute of Technology, (1992).

27 E. L. Cussler, *Diffusion Mass Transfer in Fluid Systems,* Cambridge University Press: New York, 1984, p. 27.

28 R. Bhave and K. Sirkar, "Gas permeation and separation by aqueous membranfieles immobilized across the whole thickness or in a thin section of hydrophobic microporous Celgard® films," *J. Membrane Sci.* 27 (1986) 225.

29 R. Bhave and K. Sirkar, "Gas permeation and separation with aqueous membranes immobilized in microporous hydrophobic hollow fibers," *ACS Symposium Series,* 347 (1987) 138.

30 B. Kim and P. Hariott, "Critical entry pressure for liquids in hydrophobic membranes," *J. Coll. Inter. Sci.,* 115 (1987) 1.

31 R. Reid, J. Prausnitz, and T. Sherwood, *Properties of Gases and Liquids,* McGraw Hill, New York, 1977.