A Model for the Biodegradation of Tetrachloroethylene by a Mixed Aerobic-Anaerobic Culture Immobilized in Ca-alginate

by

Daniel W. Crouse

Submitted to the
Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the
Degree of

Master of Science in Civil
and Environmental Engineering

at the

Massachusetts Institute of Technology

January 1994

Signature of Author . . . . . .
Department of Civil and Environmental Engineering
January 14, 1994

Certified by . . . . . .
Assistant Professor Lee Krumholz
Thesis Supervisor

Accepted by . . . . . .
Joseph M. Sussman, Chairman
Departmental Committee on Graduate Studies

A Model for the Biodegradation of Tetrachloroethylene by a Mixed Aerobic-Anaerobic Culture Immobilized in Ca-alginate

by

Daniel W. Crouse

Submitted to the Department of Civil and Environmental Engineering on January 14, 1994 in partial fulfillment of the requirements for the Degree of Master of Science in Civil and Environmental Engineering

ABSTRACT

A variety of industrial chemicals are thought to be highly recalcitrant to biodegradation by a single group of either aerobic or anaerobic bacteria. The coupling of these two groups of microorganisms has become increasingly useful for the biodegradation of some of these organic compounds including tetrachloroethylene and carbon tetrachloride. One method of coupling these reactions is to immobilize microorganisms in Ca-alginate spheres where both aerobic and anaerobic zones exist due to mass transfer limitations and the biological utilization of oxygen. Aerobic bacteria can then thrive near the exterior where oxygen is abundant, while the anaerobes flourish in the interior oxygen-free zone.

A model is created to describe the biodegradation of two model compounds by a mixed culture immobilized in Ca-alginate spheres of 3 mm diameter. Two scenarios examine tetrachloroethylene biodegradation using different substrates in a mixed culture. In the third scenario the degradation of 4-chloro-2-nitrophenol using an immobilized pure culture with glucose as the anaerobic substrate is modeled. A computer program is used to solve the model and generate data describing effluent concentration from a bioreactor over time. The model is an explicit, finite-difference scheme using a time step of 0.1 seconds, with each bead discretized into ten hollow spheres. Fick's Law is used to describe the diffusion of chemicals into the spheres where the biodegradation is modeled using Monod kinetics.

This first two scenarios predict significant degradation of tetrachloroethylene (PCE) under aerobic conditions. In the first scenario the PCE concentration is reduced by 2 orders of magnitude after 140 hours, while in second scenario this occurred after only 24 hours. The DCE peak concentration for these two scenarios also occurred at these times. Steady state was not achieved in the first scenario after the 240 hour simulation, but scenario 2 achieved steady state after only 70 hours. The simulated data of Scenario 3 compares well to experimental data from Beunink and Rehm (1990) which shows complete degradation of CNP to CAP after 30 hours, and the subsequent mineralization of CAP. The concentration profiles of the contaminants, oxygen, and substrates within the beads of each scenario are also examined. The oxygen profile compares well to experimental data by Beunink et al. (1989) and the profile data for the substrates and contaminants are consistent with expectations.

Thesis Supervisor: Dr. Lee Krumholz
Title: Assistant Professor of Civil and Environmental Engineering

ACKNOWLEDGMENTS

The inspiration for this work came from Lee Krumholz whose in depth knowledge of microbiology made all this possible, as well as saving me many tedious hours in the library searching for information that he knew off the top of his head. His patience was also extremely valuable in dealing with a person who is somewhat less than easy-going.

Phil Gschwend and his research group also deserve thanks for their input. Phil probably saved me weeks of work in the beginning by setting me in the right direction when I didn't know which way was up. His thorough knowledge of chemistry was indeed valuable to this work. The Gschwend lab group offered insightful suggestions from an impartial perspective when I had begun to lose mine. I have them to thank for the existence of the Model Validation and Mass Balance section of this work.

Other invaluable help with numerical methods was offered by Eric Adams. When I had a hard time proving to others what I believed to be true, he showed me how to force others to believe. Indeed, there never would have been any numerics if Greg hadn't given me insight into C programming, and if MIT did not have such wonderful facilities; not the least of which is the Cray supercomputer which saved days of computation.

My wife Carolyn deserves credit here. She stuck with me through some rough times, especially in the beginning when I was wondering if I'd made some big mistakes.

Finally, my parents. I would not be here today had my parents not stressed the importance of excellence and living to one's potential. Somehow they managed to instill in me a motivation and desire to succeed without ever forcing me to accomplish. This set of values will forever serve me well.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

LITERATURE REVIEW

INTRODUCTION

Biological processes have been used to remediate many different contaminants produced by man. Secondary (biological) treatment is widely used today to reduce the organic content in the billions of gallons of sewage produced daily in this country.

A more recent development in the use of microorganisms involves bacterial degradation of xenobiotic compounds produced for use in agriculture, as industrial solvents, lubricants, and for a host of other applications. Unfortunately, during both the production and use of these chemicals, accidents and spills are inevitable. Once these compounds are released into the environment, they are immediately subject to attack by a host of microorganisms. In many cases, the microorganisms suited to the degradation of a particular compound are already present in the environment, however, the environmental conditions may not be suitable for the successful degradation of a particular chemical. For example, highly chlorinated PCBs typically only degrade under anaerobic conditions (Brown et al., 1987). If chemicals such as these are released into an aerobic environment, they can persist for decades. This is illustrated by the persistence of PCB contamination in the Hudson River, Lake Michigan, and many other areas (Harkness et al., 1993; Rapaport, 1988; Smith et al., 1990).

Persistent chemical contamination in the environment has been aggressively pursued by government regulators in the past decade through legislation such as Superfund, CERCLA, and RCRA. Much of the treatment at contaminated sites uses nonbiological processes, including incineration, immobilization, and landfilling. Bioremediation is increasing in popularity due to its low price relative to other options and the fact that contaminants are often completely mineralized into their innocuous inorganic constituents. Many of these biological processes must be applied *ex situ* as bacteria often require special culture conditions which are not easily reproduced in situ.

Ex situ biological treatment can usually be well controlled because microorganisms can be cultured for special degradation processes and contained in a bioreactor. The bioreactor controls many of the critical variables necessary for microorganisms to flourish, such as nutrient availability, oxygen supply, and pH.

An alternate and often less expensive method for using bacteria to carry out biological processes is *in situ* treatment. This type of treatment has the benefit that considerable expense can be saved not having to remove the contaminated material to facilitate treatment. Many studies have been conducted studying *in situ* treatment in an attempt to stimulate the indigenous microflora to degrade certain compounds. The studies usually describe the addition of supplemental minerals and growth substrates (MacDonald and Rittman, 1993). However, this type of treatment does not lend itself to rigorous control. Uncertainties often make interpretation of the results quite difficult These uncertainties are due to differences in microbial population, nonuniform distribution of supplements, migration of microbial populations, and the general heterogeneity and anisotropy commonly occurring in many subsurface environments.

Even given the complex conditions under which *in situ* bioremediation programs are conducted, techniques exist to effectively degrade many organic compounds. Land farming, bioventing, and biosparging are all techniques for stimulating microbial degradation of hydrocarbons and, in many cases, chlorinated aromatics *in situ*. These processes often involve supplying oxygen and nutrients to indigenous microflora that generally are naturally rate limited by either oxygen supply or nutrient availability. The contaminant is then degraded in a reaction where it becomes the electron donor and carbon source for microorganisms. Inorganic nutrients such as ammonium nitrate and an oxygen source such as air or hydrogen peroxide are commonly used to supplement microorganisms in these processes (Fredrickson et al., 1993).

CELL IMMOBILIZATION

A process that has potential uses for both *ex situ* and *in situ* bioremediation is the immobilization of microorganisms within a polymer matrix. This type of immobilization has been pursued during the past decade (Cheetham, 1980; Birnbaum et al., 1982). There is significant value in understanding the usefulness of preserving the viability of cells by entrapping them whole and undamaged. A process such as this was described by Nilsson et al. (1983). Immobilized bacteria have been shown to be effective in degrading many types of contaminants including chlorinated phenols (O'Reilly and Crawford, 1989; Portier and Fujisaki, 1986). There are both advantages and disadvantages to this type of technique. The use of immobilized cells permits the operation of bioreactors at flow rates that are independent of the growth rate of the microorganisms employed (Nunez and Lema, 1987). It could also allow *in situ* applications without allowing migration of the organisms. However, when microorganisms are immobilized in a matrix, a mass transfer problem is commonly encountered. This is due to the barrier that exists in the matrix itself. Common ways to immobilize cells include the use of a polymer matrix of polyurethane, Ca-alginate, or agar (Gosmann and Rehm, 1988; O'Reilly and Crawford, 1989; Nilsson et al., 1983). These matrices are normally formed into spheres of up to 5 mm in diameter containing a generally homogeneous concentration of cells (Beunink et al., 1989). Spherical particles also help facilitate column and reactor packing (Nilsson et al., 1983).

MASS TRANSFER LIMITATIONS

Mass transfer limitations become evident when one is concerned about the transport of substrates and nutrients into polymer matrices used to immobilize microorganisms. Since the matrix forms a semi-solid sphere, diffusion is the primary method of mass transfer. Of high importance is the diffusion of oxygen into the spheres. Beunink et al. (1989) studied the oxygen gradients in spherical Ca-alginate beads

containing entrapped whole cells of *Enterobacter cloacae* and found a steep oxygen gradient near the surface of the beads. This resulted in an anaerobic zone in the interior of the beads. The work of Chang and Moo-Young (1988) centered around estimating the oxygen penetration depth in Ca-alginate beads. Formulae were developed based on mass transfer resistances which were tailored according to specific parameters of the bioreactor, bead shape, and several other variables. From the data presented, it is apparent that mass transfer limitations, especially for oxygen, are of utmost importance in maintaining viable aerobes immobilized in a polymer matrix.

In order to properly design and engineer systems containing entrapped microorganisms in polymer matrices, the mass transfer limitations created by the matrix need to be understood. The phenomenon of mass transfer resistance in gels entrapping cells has been extensively studied (Sun et al., 1989; Korgel et al., 1992; Mehmetoglu, 1990; Longo et al., 1992). Most researchers center their attentions on describing the change in the diffusion coefficient of various chemicals based on the type of gel used and the cell concentration. Muhr and Blanshard (1982) give a comprehensive review of diffusion in gels including the governing equations and theory. However, when limiting the topic of diffusion in gels to that of biological importance and relevance, one can look to other sources of information such as the review presented by Westrin and Axelsson (1991). In this review, the theory behind diffusion in gels is developed and the corresponding equations are derived to obtain final formulae for the prediction of diffusion coefficients in gels entrapping microorganisms.

There are several variations in theory that lead to different equations governing diffusion in gels. All of the theories describe diffusion with Fick's law:

$$F = -D \frac{dC_G}{dx} \tag{1}$$

where F is the mass flux, D is the diffusion coefficient in pure gel, C_G is the solute concentration in the gel phase, and x is distance in the direction of diffusion. Another

equation can be written in terms of D_e, or the effective diffusion coefficient in gel containing microbial cells:

$$F = -D_e \frac{dC_L}{dx} \tag{2}$$

where F and x refer to the gel and C_L is the amount of solute per unit volume of the liquid void phase within the gel. Various theories attempt to obtain D_e from D and other parameters such as the polymer volume fraction within the bead, and the effective diffusion coefficient within each cell. (Chang and Moo-Young, 1988; Sun et al., 1989; Mehmetoglu, 1990). The effective diffusion coefficient in a polymer gel is less than the corresponding aqueous diffusion coefficient (D or D_{aq}). This is for two reasons. First, the polymer reduces the available volume (or area) to some fraction of the total; called the exclusion effect. Second, the obstruction effect is created when impermeable polymer molecules increase the path length which has to be traveled by a diffusing molecule.

Using the above definitions of diffusion coefficients, formulae are developed to predict the variance of diffusion coefficients with gel type and cell concentration. This can take two approaches: one is to address the cells as impermeable particles of a finite size that must be circumvented in a diffusion path, and the other is to allow the cells themselves to have a small diffusion coefficient unique to themselves which allows some diffusion to take place directly through the cells. Yan et al. (1989) develop a model that assumes the cells to be impermeable boundaries. The predictions made by their equations match their experimental data fairly well. However, there are many different equations developed by different researchers that fit certain data relatively well (e.g. Westrin and Axelsson 1991). In all the predictions however, the deviation between the aqueous diffusion coefficient and that in gels is less that one order of magnitude, and with cell volume fractions less than 0.3, the effective diffusion coefficient in gel containing microorganisms is nearly 75% of that in pure gel alone. This leads to the conclusion that the difference in diffusion coefficients between aqueous solutions and gel beads

containing entrapped cells is also less than one order of magnitude. In a study by Sun et al. (1989) which investigated the effect of cell density on the oxygen diffusion coefficient in calcium-alginate gel, the effective diffusion coefficients ranged from 86% (0 g dry cell/liter of gel) to 55% (170 g dry cells/liter gel) of the aqueous diffusion coefficient.

The interest in diffusion coefficients in gels is a result of the efforts of some researchers to create microenvironments in the form of gel beads for the existence and growth of microorganisms. In order for microorganisms to survive and flourish, there must be a constant supply of substrate and nutrients. The effective diffusion coefficients shed some light onto the availability of substrates, etc. to microorganisms entrapped in a polymer matrix. These matrices then become the habitat for microorganisms that carry out a certain biological degradation reaction of interest to the researcher.

COUPLED REACTION SYSTEMS

Research has also been conducted that exploits these mass transfer limitations by utilizing the anaerobic zone created by the depletion of oxygen in the inner radii of each bead. Beunink and Rehm (1988) studied the possibility of using the aerobic and anaerobic zones to carry out synchronous aerobic and anaerobic degradation. A model compound selected by these researchers was 1, 1, 1 - trichloro- 2, 2-bis(4-chlorophenyl) ethane (DDT). This compound was chosen due to its slow degradation in the environment and its accumulation in the food chain (Woodwell et al., 1971). The complete mineralization of this chemical by sequential cometabolic reactions is possible (Focht, 1972; Focht and Alexander 1970, 1971; Pfaender and Alexander, 1972, 1973). The cometabolic reactions necessary for the degradation of DDT are both reductive and oxidative (Guenzi and Beard, 1968). Beunink and Rehm showed that by generating both aerobic and anaerobic zones inside Ca-alginate beads containing entrapped viable cells, this sequential reaction could occur and the mineralization of DDT was possible. In their experiment 40% of the DDT added to their medium was degraded.

Another study by Beunink and Rehm (1990) investigated the use of this type of coupled aerobic and anaerobic system in the degradation of 4-chloro-2-nitrophenol. Ca-alginate was again used as the immobilization material. By co-immobilizing two strains of bacteria (*Enterobacter cloacae* and *Alcaligenes* sp.) and using glucose as the anaerobic substrate, they were able to show that the coupled reaction did occur with the resultant mineralization of 93% of the 4-chloro-2-nitrophenol in the culture. Thus, the mass transfer limitations that would otherwise hamper an oxidative process make possible a combined oxidative and reductive system that would normally require two separate bioreactors or two completely different environments.

Karube et al. (1980) were able to show methanogenesis under aerobic condition utilizing the mass transfer limitations caused by immobilizing cells in agar gel, polyacrylamide-gel, and collagen, with agar providing the highest rate of methane production. Using cells immobilized in agar and aerobically incubated at 37°C, methane production plateaued after 25 days and continued at steady-state throughout the 90 day experiment. Similarly, Kokufuta et al. (1988) showed both nitrification (an oxidative process) and denitrification (a reductive process) using cells immobilized in a polyelectrolyte complex. A co-immobolized culture of *Nitrosomonas europaea* and *Paracoccus denitrificans* was able to completely remove nitrogen (in the form of ammonia) from the experimental system in 150 hours under continuous aerobic conditions.

MODELLING

The above studies utilized the diffusion restrictions caused by immobilization material to create coupled aerobic and anaerobic reactions. However, in addition to equations governing the diffusion of materials into and out of the matrix, insight is needed into the extent to which biological transformations occur inside the matrix with respect to a certain chemical system. Semprini and McCarty (1991) present a nonsteady-

state model for estimating the growth of an indigenous microbial population in saturated porous media, resulting from the addition of an electron donor (primary substrate) and an electron acceptor. Their model presentation is similar to that given by Molz et al. (1986) and Borden and Bedient (1986), and includes basic microbial and physical processes that govern transport of chemicals in saturated porous media. Biological parameters include microbial growth and utilization of electron donor and acceptor.

The rates of microbial growth and decay are assumed to be functions of both electron donor and acceptor (Kissel et al., 1984; Molz et al., 1986; Borden and Bedient, 1986):

$$\frac{\partial X}{\partial t} = XkY\left(\frac{C_D}{K_{SD}+C_D}\right)\left(\frac{C_A}{K_{SA}+C_A}\right) - bX\left(\frac{C_A}{K_{SA}+C_A}\right) \tag{3}$$

where:

X	=	cell concentration (mg/l)
k	=	maximum donor utilization rate (g donor/g cells d)
Y	=	yield coefficient (g cells/g donor)
K_{SD}	=	donor half-saturation constant (mg donor/l)
K_{SA}	=	acceptor half-saturation constant (mg acceptor/l)
b	=	cell decay coefficient (day^{-1})
C_A	=	concentration of electron acceptor (mg/l)
C_D	=	concentration of electron donor (mg/l)

and the rates of utilization of the electron donor and acceptor are given by equations (4) and (5), respectively:

$$\frac{\partial C_D}{\partial t} = -kX\left(\frac{C_D}{K_{SD}+C_D}\right)\left(\frac{C_A}{K_{SA}+C_A}\right) \tag{4}$$

$$\frac{\partial C_A}{\partial t} = -kFX\left(\frac{C_D}{K_{SD}+C_D}\right)\left(\frac{C_A}{K_{SA}+C_A}\right) - d_c f_d bX\left(\frac{C_A}{K_{SA}+C_A}\right) \tag{5}$$

where:

F = stoichiometric ratio of electron acceptor to electron donor utilization for biomass synthesis (g acceptor/g donor)
d_c = cell decay oxygen demand (g O2/g cells)
f_d = fraction of cells that is biodegradable

The above equations are further developed in Semprini and McCarty (1992) to include cometabolic transformation kinetics where an enzyme such as methane monooxygenase whose production is stimulated by a certain chemical (e.g.; methane) fortuitously degrades other compounds present (e.g.; DCE) as well as the stimulating chemical. This results in equation (6) using a dual Monod expression reflecting the competition likely between the growth substrate and the non-growth substrate for the active site of the methane monooxygenase (MMO) enzyme which can influence the rate of biotransformation of the non-growth compound. As the concentration of the electron donor increases, there is a proportional decrease in the amount of non-growth substrate transformed.

$$\frac{\partial C_2}{\partial t} = -F_a X k_2 \left[\frac{C_2}{K_{S2} + C_2 + \frac{K_{S2} C_D}{K_{SD}}} \right] \left[\frac{C_A}{K_{SA} + C_A} \right] \qquad (6)$$

where:

C_2 = contaminant concentration (mg/l)
X = concentration of bacteria active toward cometabolic transformation of contaminant (mg/l)
k_2 = maximum utilization rate of cometabolism (mg contaminant/mg cell d)
K_{S2} = contaminant half-saturation coefficient (mg contaminant/l)
C_A = concentration of electron acceptor (mg/l)
C_D = concentration of electron donor (mg/l)
K_{SD} = donor half-saturation constant (mg donor/l)
K_{SA} = acceptor half-saturation constant (mg acceptor/l)

F_a in Equation (6) is used to describe the activation-deactivation of the cometabolic process. If the population is growing on the electron donor, that is, if:

$$\frac{dX}{dt} > 0 \qquad \text{then} \quad F_a = 1;$$

but, if

$$\frac{dX}{dt} < 0 \qquad \text{then } F_a \text{ decreases with time according to}$$

$$\frac{dF_a}{dt} = -b_d F_a \tag{7}$$

where

b_d = rate constant for a first-order deactivation process

Equation (6) is a dual Monod expression reflecting the competition likely between the electron donor and the non-growth substrate for the active site of the methane monooxygenase (MMO) enzyme which can influence the rate of biotransformation of the non-growth compound. As the concentration of the electron donor increases, there is a proportional decrease in the amount of non-growth substrate transformed. There is also a term for oxygen (the electron acceptor) since it also is required for aerobic cometabolism.

REFERENCES

1. Beunink J. and H.-J. Rehm. 1990. Coupled reductive and oxidative degradation of 4-chloro 2-nitrophenol by a co-immobilized mixed culture system. Appl. Microbiol. Biotechnol. 34:108-115.
2. Beunink, J., H. Baumgärtl, W. Zimelka, and H.-J. Rehm. 1989. Determination of oxygen gradients in single Ca-alginate beads by means of oxygen-microelectrodes. Experientia 45:1041-1047.
3. Beunink, J., and H.-J. Rehm. 1988. Synchronous anaerobic and aerobic degradation of DDT by an immobilized mixed culture system. Appl Microbiol. Biotechnol 29:72-80.
4. Birnbaum, S., P.-O. Larsson, and K. Mosbach. 1983. Immobilized cells. In: Schoten W (ed) Solid phase biochemistry. John Wiley, New York.
5. Borden, R. C. and P. B. Bedient. 1986. Transport of dissolved hydrocarbons influenced by oxygen-limited biodegradation. 1. Theoretcial development. Water Res. Res. 22n13:1973-1982.
6. Brown, J. F. Jr., D. L. Bedard, M. J. Brennan, J. C. Carnahan, H. Feng, and R. E. Wagner. 1987. Polychlorinated biphenyl dechlorination in aquatic sediments. Science 236:709.
7. Chang, Ho Nam, and M. Moo-Young. 1988. Estimation of oxygen penetration depth in immobilized cells. Appl. Microbiol. Biotechnol. 29:107-112.
8. Cheetham, P. S. J. 1980. Developments in the immobilization of microbial cells and their applications. In: Wiseman A (ed) Topics in enzyme and fermentation technology, vol 4. Ellis Horwood, Chichester, pp 189-238.
9. Focht, D. D. 1972. Microbial degradation of DDT metabolites to carbon dioxide, water and chloride. Bull. Environ. cont. Toxicol. 7:51-56.
10. Focht, D. D., and M. Alexander. 1971. Aerobic cometabolism of DDT analogues by *Hydrogenomonas* sp. J. Agric. Food Chem. 19:20-22.
11. Focht, D. D., and M. Alexander. 1970. DDT metabolites and analogs: ring fission by *Hydrogenomonas*. Science 170:91-92.
12. Fredrickson, J. K., H. Bolton Jr., and F. J. Brockman. 1993. In situ and on-site bioreclamation. Environ. Sci. Tech. 27:1711-1716.
13. Gosmann, B., and H.-J. Rehm. 1988. Influence of growth begavior and physiology of alginate-entrapped microorganisms on the oxygen consumption. Appl. Microbiol. Biotechnol. 29:554-559.
14. Guenzi, W. D., and W.E. Beard. 1968. Anaerobic conversion of DDT to DDD and aerobic stability of DDT is soil. Proc. Soil Sci. Soc. Americ. 32:522-524.
15. Harkness, M. R., J. B. McDermott, D. A. Abramowicz, J. J. Salvo, W. P. Flanagan, M. L. Stephens, F. J. Mondello, R. J. May, J. H. Lobos, K. M. Carroll, M. J. Brennan, A. A. Bracco, K. M. Fish, G. L. Warner, P. R. Wilson, D. K. Dietrich, D. T. Lin, C. B. Morgan, and W. L. Gately. 1993. In situ stimulation of aerobic PCB bidegradation in Hudson River sediments. Science 259:503-507.
16. Karube, I., S. Kuriyama, T. Matsunaga, and S. Suzuki. 1980. Methane progdcuction from wastewaters by immobilized methanogenic bacteria. Biotech. Bioeng. 22:847-857.
17. Kissel, J. C., P. L. McCarty, and R. L. Street. 1984. Numerical simulation of mixed-culture biofilm. J. of Envrionmental Engineering (ASCE). 110n2:393-411.
18. Kokufuta, M. Shimohashi, and I. Nakamura. 1988. Simultaneously occurring nitrification and denitrification under oxygen gradient by polyelectrolyte complex-coimmobilized *Nitrosomonas europaea* and *Paracoccus denitrificans* cells. Biotech. Bioeng. 31:382-384.

19. Korgel, B. A., A. Rotem, and H. G. Monbouquette. 1992. Effective diffusivity of galactose in Ca-alginate gels containing immobilized *Zymomonas mobilis*. Biotechnol. Prog. 8:111-117.
20. Longo, M. A., I. S. Novella, L. A. Garcia, and M. Diaz. 1992. Diffusion of proteases in calcium alginate beads. Enzyme Microb. Technol. 14:586-590.
21. MacDonald, J. A., and B. Rittmann. 1993. Performance standards for in situ bioremediation. Envir. Sci. Tech. 27:1974-1979.
22. Mehmetoglu, U. 1990. Effective diffusion coefficient of sucrose in calcium alginate gel. Enzyme Microb. Technol. 12:124-126.
23. Molz, F. J., M. A. Widdowson, and L. D. Benefield. 1986. Simulation of microbial growth dynamics coupled to nutrient and oxygen transport in porous media. Water Res. Res. 22n8:1207-1216.
24. Muhr, A. H., and M. V. Blanshard. 1982. Diffusion in gels. Polymer 23:1012-1026.
25. Nilsson, Kjell, et al. 1983. A General Method for ith Immobilization of Cells with Preserved Viability. Eur. J. Appl. Microbiol. Biotechnol. 17:319-326.
26. Nunez, M. J., and J. M. Lema. 1987. Cell immobilization: application to alcohol production. Enzyme Microb. Technol. 9:642-661.
27. O'Reilly, K. T., and R. L. Crawford. 1989. Degradation of Pentachlorophenol by Polyurethane-Immobilized *Flavobacterium* Cells. Appl. Environ. Microbiol. 55:2113-2118.
28. Pfaender, F.K., and M. Alexander. 1973. Effect of nutrient additions on the apparent cometabolism of DDT. J. Agric. Food Chem. 21:397-399.
29. Pfaender, F. K., and M. Alexander. 1972. Extensive microbial degradation of DDT in vitro and DDT metabolism by natural communities. J. Agric. Food Chem. 20:842-846.
30. Portier, R. J., and K. Fujisaki. 1986. Continuous Biodegradation and Detoxification of Chlorinated Phenols Using Immobilized Bacteria. Toxicity Assessment: An International Quarterly 1:501-513.
31. Rapaport, R. A., and S. J. Eisenreich. 1988. Historical atmospheric inputs of high molecular-weight chlorinated hydrocarbons to eastern North America. Environ. Sci. Technbol. 22:931-941.
32. Semprini, L., and P. L. McCarty. 1991. Comparison Between Model Simulations and Field Results for In-Situ biorestoration of Chlorinated Aliphatics: Part1. Biostimulation of Methanotrophic Bacteria. Ground Water 29n3:365-374.
33. Smith, P. L., R. A. Ragotzgie, A. W. Andren, and H. Harris. 1990. J. Estuary Rehabilitation: The Green Bay Story. p. 12-20.
34. Sun, Y., S. Furusaki, A. Yamauchi, and K. Ichimura. 1989. Diffusivity of oxygen into carriers entrapping whole cells. Biotech. Bioeng. 34:55-58.
35. Westrin, B. A. and A. Axelsson. 1991. Diffusion in gels containing immobilized cells: A critical review. Biotech. Bioeng. 38:439-446.
36. Woodwell, G. M., P. P. Craig, A. A. Johnson. 1971. DDT in the biosphere: where does it go? Science 74:1101-1107.

INTRODUCTION

Organic solvents with varying degrees of chlorination have many industrial uses. Tetrachloroethylene (also known as perchloroethylene, PCE) is a solvent commonly used in dry cleaning. It has been estimated that 6.4E9 kg of PCE were synthesized and distributed in the United States over the period from 1945 to 1984 (~164 tons/year), with a significant fraction of this entering the ground because of improper handling and storage (Abelson, 1990). Due to such widespread contamination, remediation technology concerning chlorinated solvents is becoming increasingly important. One promising method of remediation involves the use of microorganisms to degrade, and ultimately mineralize, organic pollutants.

One type of bioremediation effective in degrading chlorinated compounds involves the use of coupled aerobic-anaerobic systems. In addition to effectively removing nitrogen and phosphorus from wastewater, coupled systems are also useful for degrading toxic chemicals such as PCE, hexachlorobenzene, and carbon tetrachloride which are recalcitrant under normal, aerobic conditions (Zitomer and Speece, 1993).

Another type of bioremediation involves immobilizing relevant microbial communities in gel beads that could then be placed into a contaminated area or maintained in a bioreactor. The immobilization material must be permeable to allow the immobilized cells access to oxygen, nutrients, and substrates, yet rigid enough to maintain cell immobilization. This can be accomplished by entrapping cells in a polymer matrix of agar, Ca-alginate, or some other gel media (Nilsson et al., 1983). The medium is then formed into spheres of up to 5 mm in diameter which contain a relatively uniform concentration of immobilized microorganisms.

Entrapping cells in polymer beads creates mass transfer limitations due to the presence of the polymer matrix. Diffusion becomes the main process by which oxygen, nutrients, and substrates are transported into the bead (Beunink, et al., 1989). Oxygen transport limitations often lead to an anaerobic zone within each polymer bead. This

anaerobic zone can be exploited to carry out synchronous aerobic and anaerobic degradation that often require two different bioreactors with completely different operating environments. Beunink and Rehm (1988) were able to show that by using the mass transfer limitations inherent in Ca-alginate entrapped whole cells, they were able to effectively (40% degradation) couple aerobic and anaerobic reactions to degrade DDT. Similarly, Beunink and Rehm (1990) where able to show mineralization of 4-chloro 2-nitrophenol by coupling reductive and oxidative reactions in a co-immobilized, mixed culture system.

It is the purpose of this study to develop a predictive model which describes the biological processes taking place within Ca-alginate beads entrapping whole, viable cells. We will then show that the mass transfer limitations for oxygen can be exploited to create a coupled reaction system with an aerobic reaction occurring near the oxygen rich surface of the beads, while the interior anaerobic zone allows anaerobic processes to occur. This technique could then be used to predict the effectiveness of a coupled aerobic-anaerobic reaction for the degradation of a particular chlorinated chemical. The transport of nutrients, substrates, and oxygen to cells immobilized in the beads will be addressed. The model will also account for microbial growth which allows for an increased rate of biodegradation over time. The processes for an individual bead will then be expanded to describe three bioreactor systems in which the influent stream is contaminated with tetrachloroethylene (Scenarios 1 and 2) or 4-chloro-2-nitrophenol (CNP) (Scenario 3). In the first two scenarios PCE is reductively dechlorinated with a substrate (acetate in Scenario 1 and phenol in Scenario 2) in the absence of oxygen to yield carbon dioxide, dichloroethylene (DCE), and hydrochloric acid (Scholz-Muramatsu et al., 1989; Vogel and McCarty, 1985; Krumholz, 1993; Barik et al., 1985). The DCE is then cometabolized with a substrate (methane in Scenario 1 and phenol in Scenario 2) in the presence of oxygen to form carbon dioxide and hydrochloric acid (Hopkins et al., 1993). In Scenario 3 CNP is reduced with glucose as the electron donor to form 4-chloro-2-

aminophenol (CAP), carbon dioxide, and water (Beunink and Rehm, 1990). The CAP further degrades aerobically to yield carbon dioxide, ammonia, hydrochloric acid, and water.

METHODS

SCENARIO DESCRIPTION

All three scenarios follow the same general principles and similar model equations. It is proposed that due to mass transfer limitations created by the gel, oxygen diffusion into the gel beads will be limited to a certain radius. This will create distinctly different environments within the same gel bead for survival of microorganisms. Near the outer radius of each bead where oxygen is present, an aerobic environment will exist. However, as oxygen diffuses into the bead, microorganisms living near the outer radii consume the oxygen faster than it can diffuse into the interior of the bead. This creates a high oxygen gradient near the surface of the bead, and an anaerobic zone in the interior of the bead. The creation of this type of coupled aerobic and anaerobic environment has been demonstrated by Gosmann and Rehm (1988) and Beunink et al. (1989).

The three scenarios presented here differ in the type of contaminant being degraded, as well as the substrates provided for the microorganisms. In addition, the first two scenarios use mixed cultures, while the third scenario uses only pure cultures.

Scenario 1

In the first scenario tetrachloroethylene is the contaminant with acetate and methane being provided as electron donors of the anaerobes and aerobes, respectively. It is proposed that tetrachloroethylene diffuses into the interior of the bead where it is anaerobically metabolized to cis-dichloroethylene (Vogel and McCarty, 1985). The cis-dichloroethylene is then cometabolized in the presence of methane under aerobic

conditions in the outer, aerobic zone to carbon dioxide and hydrochloric acid. The contaminant transformations are as follows:

anaerobic (Vogel and McCarty, 1985);

$$C_2H_4O_2 + 2C_2Cl_4 + 2H_2O \Rightarrow 2CO_2 + 2C_2H_2Cl_2 + 4HCl \quad \text{(E1)}$$

acetate + tetrachloroethylene + water ⇒

carbon dioxide + cis-dichloroethylene + hydrochloric acid

aerobic (Hopkins et al., 1993);

$$C_2H_2Cl_2 + 2O_2 \Rightarrow 2CO_2 + 2HCl \quad \text{(E2)}$$

cis-dichloroethylene + oxygen ⇒ carbon dioxide + hydrochloric acid

Other reactions are also occurring in the bead, including the oxidation of methane and acetate by aerobes in the aerobic zone of the beads. These reactions are shown in equations (E3) and (E4):

aerobic;

$$CH_4 + 2O_2 \Rightarrow CO_2 + 2H_2O \quad \text{(E3)}$$

methane + oxygen ⇒ carbon dioxide + water

$$C_2H_4O_2 + 2O_2 \Rightarrow 2CO_2 + 2H_2O \quad \text{(E4)}$$

acetate + oxygen ⇒ carbon dioxide + water

Equations (E1), (E3), and (E4) are the equations representing reactions used by the microorganisms for energy and growth. Equation (E2) is a cometabolic reaction utilizing

the methane monooxygenase enzyme and yields no energy to the microorganisms. Figure 1 shows a schematic of the bead processes in this scenario.

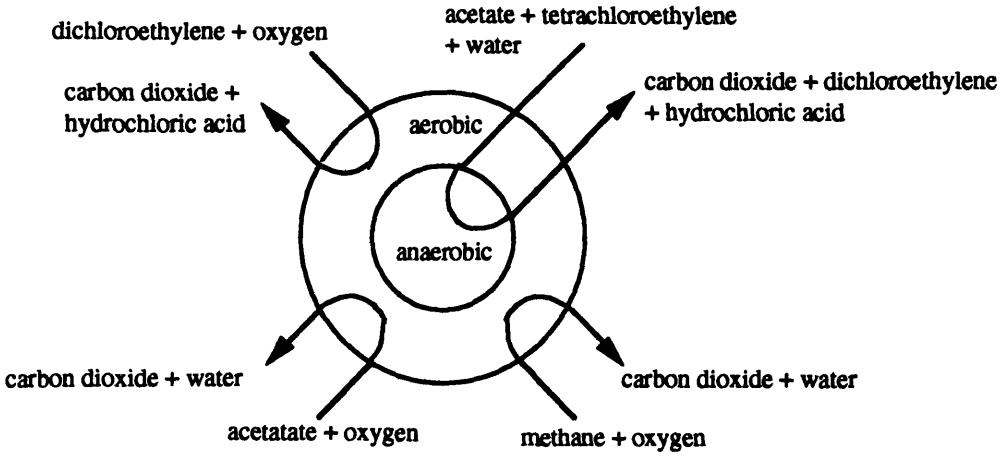

Figure 1: Transformations occurring in beads of Scenario 1.

Scenario 2

The second scenario is similar to the first with the exception that phenol is supplied as the common substrate to be utilized by both the aerobes and anaerobes. This simplifies the process by only having to follow the concentration of one substrate as opposed to both the acetate and methane above. The contaminant transformations occurring in the beads in this scenario are:

anaerobic (Barik et al., 1985);

$$C_6H_6O + 5H_2O \Rightarrow 3C_2H_4O_2 + 2H_2 \qquad \text{(E5)}$$

phenol + water ⇒ acetate + hydrogen

and (DiStefano, et al. 1992);

$$3C_2H_4O_2 + 2H_2 + 7C_2Cl_4 + 6H_2O \Rightarrow 7C_2H_2Cl_2 + 14HCl + 6CO_2 \qquad \text{(E6)}$$

acetate + hydrogen + tetrachloroethylene + water ⇒

dichloroethylene + hydrochloric acid + carbon dioxide

aerobic (Hopkins et al., 1993);

$$C_2H_2Cl_2 + 2O_2 \Rightarrow 2CO_2 + 2HCl \quad \text{(E7)}$$

dichloroethylene + oxygen $\Rightarrow$ carbon dioxide + hydrochloric acid

Another reaction occurring in this system is the growth on phenol by the aerobic bacteria:

$$C_6H_6O + 7O_2 \Rightarrow 6CO_2 + 3H_2O \quad \text{(E8)}$$

phenol + oxygen $\Rightarrow$ carbon dioxide + water

Equations (E6) and (E7) are the contaminant degradation reactions, but (E6) is also the reaction representing anaerobic growth. Similar to (E2), (E7) is a cometabolic reaction. Figure 2 shows a schematic of the bead processes in this scenario.

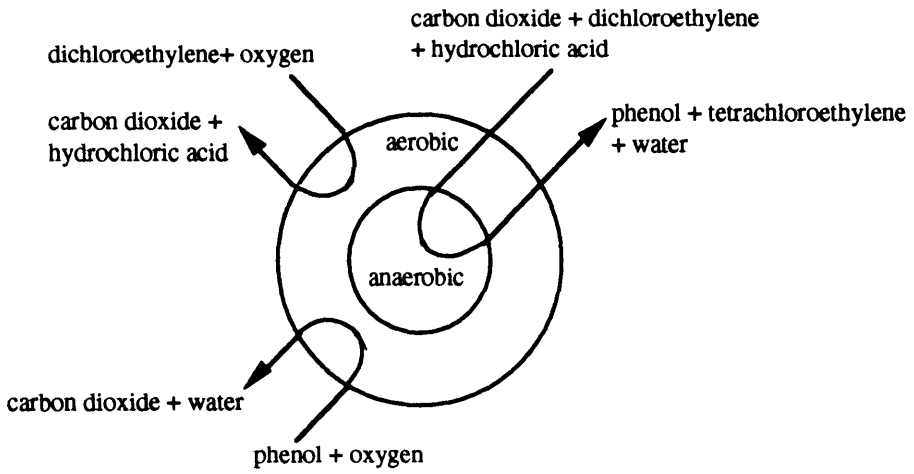

Figure 2: Transformations occurring in beads of Scenario 2.

Scenario 3

The third scenario has different substrates and contaminants than Scenarios 1 and 2. It involves the anaerobic transformation of 4-chloro-2-nitrophenol (CNP) to 4-chloro-2-aminophenol (CAP) using glucose as the anaerobic substrate. This model is taken from an experiment performed by Beunink and Rehm (1990). The reactions occurring in this system are as follows:

anaerobic;

$$4C_6H_4O_3NCl + C_6H_{12}O_6 \Rightarrow 4C_6H_6ONCl + 6CO_2 + 2H_2O \quad \text{(E9)}$$

CNP + glucose ⇒ CAP + carbon dioxide + water

aerobic;

$$C_6H_6ONCl + 6O_2 \Rightarrow 6CO_2 + NH_3 + HCl + H_2O \quad \text{(E10)}$$

CAP + oxygen ⇒ carbon dioxide + ammonia + hydrochloric acid + water

$$C_6H_{12}O_6 + 6O_2 \Rightarrow 6CO_2 + 6H_2O \quad \text{(E11)}$$

glucose + oxygen ⇒ carbon dioxide + water

Equations (E9) and (E10) are the growth reactions as well as the degradation reactions. Equation (E11) represents the aerobes growing on the anaerobic substrate, also yielding aerobic biomass. Figure 3 shows a schematic of the bead processes in this scenario.

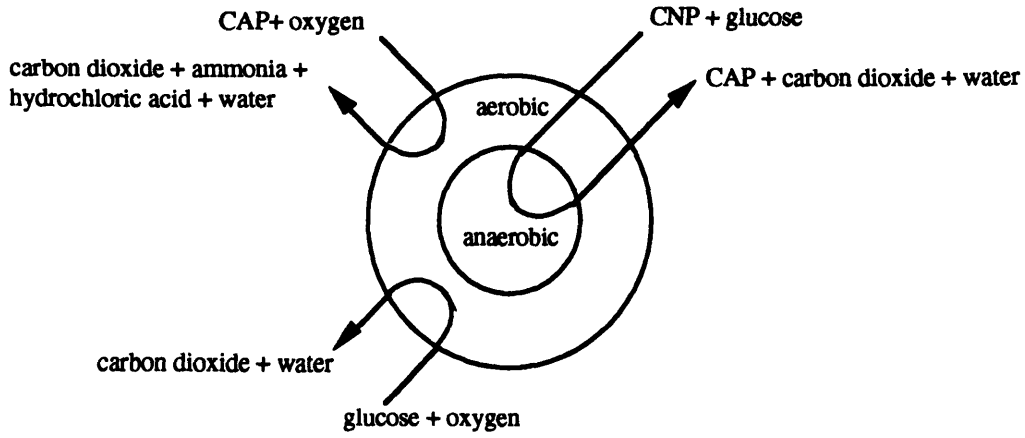

Figure 3: Transformations occurring in beads of Scenario 3.

MODEL DEVELOPMENT

Of critical importance to a system utilizing microorganisms entrapped in a polymer matrix are the mass transfer limitations caused by the matrix itself. In this simulation, the polymer immobilization matrix consists of Ca-alginate spheres with a diameter of 3.0 mm. These spheres are treated as semi-solid with molecular diffusion being the primary process involved in the transport of substrates, nutrients, oxygen, and contaminants into the spheres.

All three previously mentioned scenarios will be addressed. In each scenario, contaminated wastewater will be pumped through a theoretical bioreactor containing Ca-alginate beads with a co-immobilized mixed culture of both aerobic and anaerobic bacteria, carrying out a coupled aerobic-anaerobic reaction. This type of system has been reported experimentally in two studies by Beunink and Rehm (1988, 1990) where DDT and 4-chloro-2-nitrophenol were mineralized using a co-immobilized mixed culture in an aerobic environment. In another study (Kokufuta et al. 1988) both nitrification and denitrification were shown to occur under aerobic conditions. All three of these processes require both oxidative and reductive reactions.

Each scenario also builds upon the same model equations with slight variations with the assumption that the process responsible for transport of materials into each cell is molecular diffusion. To reflect this, the model transport equation will be the standard equation for diffusion into a sphere:

$$\frac{\partial C}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 D \frac{\partial C}{\partial r}\right) \tag{1}$$

where

C = concentration in the bead (moles/m^3)
t = time (sec)
r = radius (m)
D = molecular diffusion coefficient of a chemical in Ca-alginate gel (m^2/sec)

Equation (1) describes the diffusion of a chemical with molecular diffusion coefficient, D, into each bead with respect to time. Another term (B) will be added to equation (1) to reflect the utilization of different chemicals by immobilized microorganisms:

$$\frac{\partial C}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 D \frac{\partial C}{\partial r}\right) + B \tag{2}$$

where

B = term describing biological utilization within gel beads (moles/m^3 sec)

The biological utilization term for all three scenarios is also similar. Common to two of the three is a form reflecting cometabolism. Only the third involves the direct aerobic degradation of the contaminant by microorganisms. The contaminants in the first

two scenarios are broken down through an aerobic cometabolic process yielding no energy to the microorganisms.

The expression for B in equation (2) is developed below:

$$\frac{dX}{dt} = \mu X \qquad \text{from Brock and Madigan (1991)} \tag{3}$$

where

X = microbial biomass (cells)
μ = growth rate (sec^{-1})

and from Molz et al. (1986);

$$\mu = \mu_{max}\left[\frac{C_D}{K_D + C_D}\right]\left[\frac{C_A}{K_A + C_A}\right] \quad \text{for normal metabolism} \tag{4}$$

or, for metabolism with non-competitive inhibition caused by an electron donor such as phenol or toluene (Mathews and van Holde, 1990):

$$\mu = \mu_{max}\left[\frac{C_D}{(K_D + C_D)\left(1 + \frac{C_D}{K_I}\right)}\right]\left[\frac{C_A}{K_A + C_A}\right] \quad \text{for inhibited normal metabolism} \tag{5}$$

or, for cometabolism of both the electron donor and the non-energy yielding contaminant, after Semprini and McCarty (1992):

$$\mu = \mu_{max}\left[\frac{C_C}{K_C + C_C + \frac{K_c C_D}{K_D}}\right]\left[\frac{C_A}{K_A + C_A}\right] \tag{6}$$

or, combining these equations in a form for non-competitively inhibited cometabolism:

$$\mu = \mu_{max}\left[\frac{C_C}{\left(K_C + C_C + \frac{K_c C_D}{K_D}\right)\left(1+\frac{C_D}{K_I}\right)}\right]\left[\frac{C_A}{K_A + C_A}\right] \tag{7}$$

where

μ_{max} = maximum growth rate (sec^{-1})
C_D = concentration of the electron donor ($moles/m^3$)
K_D = electron donor half-saturation constant ($moles/m^3$)
C_A = concentration of electron acceptor ($moles/m^3$)
K_A = electron acceptor half-saturation constant ($moles/m^3$)
C_C = concentration of the contaminant being cometabolized ($moles/m^3$)
K_C = contaminant half-saturation constant ($moles/m^3$)
K_I = inhibition coefficient ($moles/m^3$)

Since the microbial growth is dependent on the electron donor, and assuming the only sink for the electron donor is utilization by microorganisms, it follows that the rate of growth of the biomass will be proportional to the rate of disappearance of the electron donor. This is shown mathematically in equation (8):

$$\frac{dX}{dt} = -y\frac{dM_D}{dt} \tag{8}$$

where

y = microbial yield on the electron donor (cells/mole)
M_D = mass of electron donor (moles)

Also, in a cometabolic reaction, the rate of degradation of the contaminant is proportional to the rate of degradation of the electron donor. This is shown in equation (9):

$$\frac{dM_D}{dt} = a\frac{dM_C}{dt} \tag{9}$$

where

a = proportionality constant representing the ratio of contaminant transformed relative to electron donor transformed (dimensionless)
M_C = mass of the contaminant (moles)

Now, combining equations (3) and (8):

$$\frac{dM_D}{dt} = -\frac{\mu X}{y} \tag{10}$$

combining equations (9) and (10):

$$\frac{dM_C}{dt} = -\frac{\mu X}{ay} \tag{11}$$

Substituting equation (6) into equation (11), and dividing by volume:

$$B_C = -\frac{\mu_{max} X}{ayV}\left[\frac{C_C}{K_C + C_C + \frac{K_C C_D}{K_D}}\right]\left[\frac{C_A}{K_A + C_A}\right] \tag{12}$$

where

V = bead volume of interest (m^3)
B_C = contaminant biological utilization term (moles/m^3 sec)

Finally, combining equations (2) and (12) yields an expression for the change in concentration of a contaminant undergoing cometabolism in a bead with respect to time:

$$\frac{\partial C_C}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 D_C \frac{\partial C_C}{\partial r}\right) - \frac{\mu_{max} X}{ayV}\left[\frac{C_C}{K_C + C_C + \frac{K_C C_D}{K_D}}\right]\left[\frac{C_A}{K_A + C_A}\right] \qquad (13)$$

where

D_C = diffusion coefficient of contaminant in gel (m^2/sec)

Similar expressions can be written using equations (4), (5), and (7) for the concentration of the electron donor with respect to time in the cases of normal metabolism, metabolism inhibited by the electron donor, and cometabolism inhibited by the electron donor:

$$\frac{\partial C_D}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 D_D \frac{\partial C_D}{\partial r}\right) - \frac{\mu_{max} X}{yV}\left[\frac{C_D}{K_D + C_D}\right]\left[\frac{C_A}{K_A + C_A}\right] \text{ normal metabolism} \qquad (14)$$

for inhibited metabolism;

$$\frac{\partial C_D}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 D_D \frac{\partial C_D}{\partial r}\right) - \frac{\mu_{max} X}{yV}\left[\frac{C_D}{(K_D + C_D)\left(1 + \frac{C_D}{K_I}\right)}\right]\left[\frac{C_A}{K_A + C_A}\right] \qquad (15)$$

for inhibited cometabolism;

$$\frac{\partial C_D}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 D_D \frac{\partial C_D}{\partial r}\right) - \frac{\mu_{max} X}{yV}\left[\frac{C_D}{\left(K_D + C_D + \frac{K_C C_D}{K_D}\right)\left(1 + \frac{C_D}{K_I}\right)}\right]\left[\frac{C_A}{K_A + C_A}\right] \qquad (16)$$

where

D_D = diffusion coefficient of electron donor in gel (m^2/sec)

Equations (13)-(16) are examples of the types of equations that form the basis for the model in all three scenarios. Likewise, equations of the same form can be written to describe all chemicals of interest (oxygen, nutrients, etc.) in the biological system; each having its own slightly different form and parameters.

NUMERICAL STRATEGY AND MODEL PARAMETERS

The differential equations shown above describe the processes occurring in one bead. These processes will now be extended to a finite number of beads (n) present in a bioreactor of volume V_r, with influent flow Q_{in} of concentration C_{in}. The only sink mechanism from the frame of reference of the bioreactor will be diffusion into the gel beads. The one-dimensional diffusion across the surface toward the center of each bead is given by Fick's Law:

$$F = -D\frac{\partial C}{\partial r} \tag{17}$$

where

F = flux of chemical across a boundary (moles/s m^2)
D = molecular diffusion coefficient in gel (m^2/s)
C = concentration of the chemical (moles/m^3)
r = radius of the bead (m)

A mass balance for the mass of a particular chemical in a well mixed bioreactor takes the following form:

$$\frac{dM}{dt} = Q_{in}C_{in} - Q_{out}C_{out} - FA \tag{18}$$

where

M = mass of the chemical in the reactor (moles)
A = boundary area over which diffusion is taking place (m^2)

and

$$Q_{in} = Q_{out}$$

Substituting equation (17) into equation (18), taking A as the surface area of n spherical gel beads of average radius, R, and dividing by the reactor volume yields equation (19) which is a differential equation representing the change in concentration of a chemical in the bioreactor with respect to time. The term $\partial C/\partial r$ on the right side of the equation refers to the chemical concentration gradient at the surface of the gel beads.

$$\frac{dC_{out}}{dt} = \frac{Q}{V_r}(C_{in} - C_{out}) - \frac{4\pi nR^2}{V_r} D \left.\frac{\partial C}{\partial r}\right|_R \tag{19}$$

In order to implement equation (19), a value needs to be determined for the surface bead concentration gradient. This is different for each of the scenarios described above. Making use of equations (13)-(16), the concentration of a particular chemical (substrate, contaminant, or oxygen) can be determined at a particular radius of each gel bead. The difference in concentration between the reactor and the outermost radius of each bead determines the gradient necessary for equation (19).

In each scenario, the behavior of each chemical in a bead can be described by an equation of the form (13)-(16). However, the concentration of the chemical must be determined at different radii within each bead in order to account for diffusive transport and biological utilization. To solve these differential equations, an explicit finite-difference numerical scheme was used. Using this method, equation (19) can be written:

$$\frac{C_{out,i+1} - C_{out,i}}{\Delta t} = \frac{Q}{V_r}(C_{in,i} - C_{out,i}) - \frac{4\pi nR^2}{V_r} D \frac{C_{i,R} - C_{i,R-1}}{\Delta r} \tag{20}$$

where

i = time
j = radius

and the values for $C_{i,R}$ and $C_{i,R-1}$ are determined from equations of the form (13)-(16). An example of equation (16) written using finite difference and solving for $C_{i+1,j}$ is shown in equation (21):

$$C_{D_{i+1,j}} = \Delta t \left\{ D_D \left[\frac{C_{D_{i,j+1}} - 2C_{D_{i,j}} + C_{D_{i,j-1}}}{(\Delta r)^2} \right] + \frac{2D_D}{r_j} \left[\frac{C_{D_{i,j}} - C_{D_{i,j-1}}}{\Delta r} \right] - \right.$$

$$\left. \frac{\mu_{max} X_{i,j}}{yV} \left[\frac{C_{Di,j}}{\left(K_D + C_{Di,j} + \frac{K_C C_{Di,j}}{K_D} \right) \left(1 + \frac{C_{Di,j}}{K_I} \right)} \right] \left[\frac{C_{Ai,j}}{K_A + C_{Ai,j}} \right] \right\} + C_{Di,j} \qquad (21)$$

Using a program written in C, the concentration of a chemical was calculated at each time step at 10 different radii in one bead that was considered the average of all the beads. At each radius, the number of bacterial cells was also calculated. Thus at each time step there was a new chemical concentration and bacterial number. Each scenario required the calculation of substrate concentration, contaminant concentration, oxygen concentration, aerobic cell number, anaerobic cell number, and in Scenario 1, a second substrate concentration. The initial condition for the calculation was that the concentration of all chemicals at time t=0 was zero and the initial cell concentrations were a finite value. The two boundary conditions were at r=0 (the center of the bead), the gradient of each chemical was zero, and that the surface of the bead was equal to the concentration of the surrounding bulk reactor phase liquid concentration. The parameters used in the program were as follows:

Table 1: Parameters Used In Simulation

Parameter	Value Scenario 1	2	3	Description	Source*
DO2 (cm/s)	1.5×10^{-5}	1.5×10^{-5}	1.5×10^{-5}	Oxygen diffusion coef. in gel	†
DDCE (cm/s)	8.4×10^{-6}	8.4×10^{-6}	-	DCE diffusion coef. in gel	†
DPCE (cm/s)	6.58×10^{-6}	6.58×10^{-6}	-	PCE diffusion coef. in gel	†
DPHE (cm/s)	-	6.75×10^{-6}	-	Phenol diffusion coef. in gel	†
DSUB (cm/s)	1×10^{-5}	-	-	Acetate diffusion coef. in gel	†
DCH4 (cm/s)	2.1×10^{-5}	-	-	Methane diffusion coef. in gel	†
DCNP (cm/s)	-	-	5.5×10^{-6}	CNP diffusion coef. in gel	†
DCAP (cm/s)	-	-	5.7×10^{-6}	CAP diffusion coef. in gel	†
DGLU (cm/s)	-	-	5.4×10^{-6}	Glucose diffusion coef. in gel	†
KO2 (molar)	3.13×10^{-5}	3.13×10^{-5}	3.4×10^{-4}	Monod half-saturation constant for oxygen	30, 30, 11
KDCE (molar)	1.03×10^{-5}	1.03×10^{-5}	-	Monod half-saturation constant for DCE	29
KPCE (molar)	5×10^{-7}	5×10^{-7}	-	Monod half-saturation constant for PCE	15
KPHE (molar)	-	1.59×10^{-5}	-	Aerobic Monod half-saturation constant for phenol	19
KPHEA (molar)	-	5×10^{-4}	-	Anaerobic Monod half-saturation constant for phenol	12
KSUB (molar)	6×10^{-4}	-	-	Anaerobic Monod half-saturation constant for acetate	25, 13
KSUBA (molar)	1.6×10^{-4}	-	-	Aerobic Monod half-saturation constant for acetate	††
KCH4 (molar)	1.25×10^{-4}	-	-	Monod half-saturation constant for methane	30
KGLU (molar)	-	-	1.33×10^{-4}	Aerobic Monod half-saturation constant for glucose	††
KGLUA (molar)	-	-	1.33×10^{-4}	Anaerobic Monod half-saturation constant for glucose	4
KCNP (molar)	-	-	2.5×10^{-5}	Monod half-saturation constant for CNP	††
KCAP (molar)	-	-	2×10^{-4}	Monod half-saturation constant for CAP	††
KIO (molar)	1×10^{-7}	1×10^{-7}	8×10^{-6}	Aerobic inhibition constant	34
KIP (molar)	-	5.32×10^{-3}	-	Aerobic phenol inhib. constant	19
KIPA (molar)	-	1.86×10^{-2}	-	Anaerobic phenol inhib. const.	12
Ya (cells/mol)	6.25×10^{11}	3.00×10^{13}	2.1×10^{14}	Anaerobic yield	‡, ‡, 6
Y (cells/mol)	8×10^{12}	4.89×10^{13}	2.1×10^{14}	Aerobic yield	30, 19, ††
CAPyield (cells/mol)	-	-	4.89×10^{13}	Yield of CAP-oxygen reaction	††
DCEmur (1/s)	8.68×10^{-7}	1.45×10^{-5}	-	DCE maximum utilization rate	29, 18
CAPμmax (1/s)	-	-	9.6×10^{-6}	CAP maximum utilization rate	††
μmax (1/s)	1×10^{-5}	1.25×10^{-4}	7×10^{-5}	Maximum aerobic growth rate	30, 19, ††
anμmax (1/s)	4×10^{-6}	3.09×10^{-6}	2.1×10^{-5}	Max. anaerobic growth rate	††, 3, 6
cells/g dry wt	1×10^{12}	1×10^{12}	1×10^{12}	numb. of cells per gram dry wt	††

† taken as 70% of the aqueous diffusion coefficient which was calculated using the size of the molecule in an expression from Othmer and Thakar (1953) with coefficients modified by Hayduk and Laudie (1974)
†† estimate
‡ personal calculations comparing ΔG's of metabolic reactions for anaerobic yield with aerobic yield from literature
* if more than one reference is given, the order corresponds to the value given for Scenario 1, Scenario 2, and Scenario 3, respectively

The value of KIO in Table 1 refers to a term of the form:

$$\left(1+\frac{C_{oxygen_{i,j}}}{K_{IO}}\right)^{-1} \tag{22}$$

that was used in modeling anaerobic response to oxygen concentration. This form is that used in non-competitive inhibition of enzymatic reactions. It was added as a multiplicative term to anaerobic biological terms (B) of equations of form (2). KIO represents the oxygen concentration at which anaerobic activity becomes severely limited (Zehnder, 1988). Thus, with increasing oxygen concentration, anaerobic bacterial activity represented in B rapidly decreases.

RESULTS

A computer program was written in C to solve the equations shown above for each of the different scenarios. Scenarios 1 and 2 shared the same physical parameters while Scenario 3, since it was modeled after an experiment by Beunink and Rehm (1990) had physical parameters unique to itself. Table 2 shows the values of these parameters used in each simulation. Figure 4 is a schematic describing the setup of each simulation. It is important to note that the bead volume, number of beads, and cell densities used in Scenario 3 were only estimates since these parameters were not documented in the study.

Table 2: Parameters Describing the Physical Characteristics of the Reactor Setup in Each Scenario

	Scenario 1	Scenario 2	Scenario 3
Total Volume (l)	0.500	0.500	0.300
Liquid Volume (l)	0.401	0.401	0.278*
Bead Volume (l)	0.099	0.099	0.022*
Number of beads	7000	7000	5000*
Bead Radius (mm)	1.5	1.5	1.5
Init.Aerobic Cell Density (cells/ml)	1E8	1E8	5E8
Init. Anaerobic Cell Density (cells/ml)	1E8	1E8	2E8
Hyd. Ret. Time (d)	2	2	-

* estimate

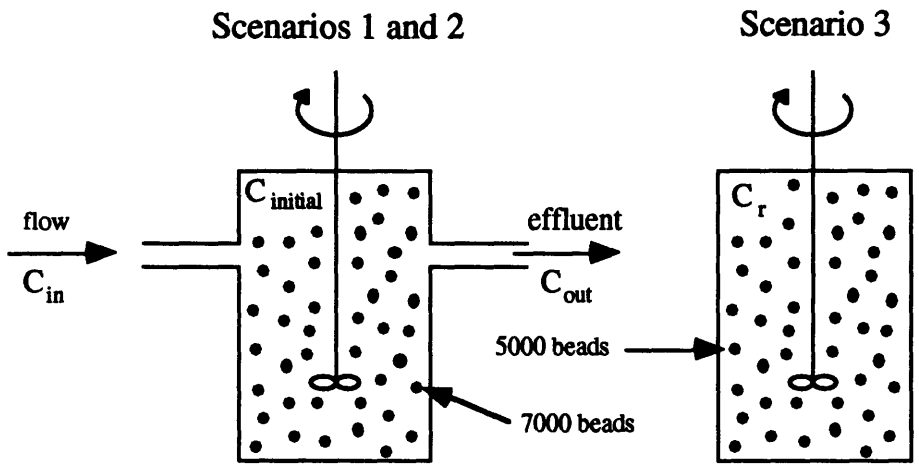

Figure 4: Schematics for computer simulations. Scenarios 1 and 2 consist of a continuous-flow bioreactor with influent concentration C_{in}, initial concentration $C_{initial}$, and effluent concentration C_{out}. Scenario 3 was a batch test with concentration in the reactor at a given time C_r.

Simulations

The following two figures show the effluent over time as predicted by the model.

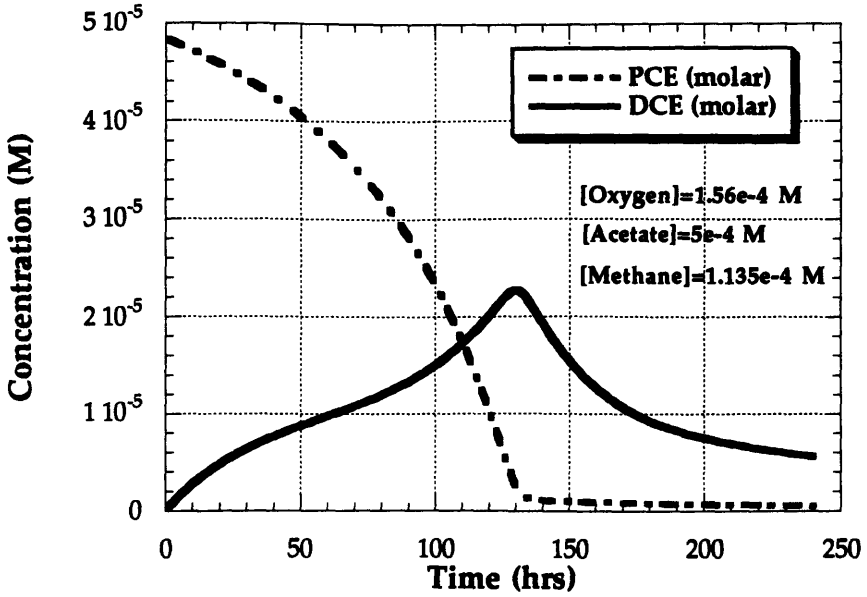

Figure 5: Plot of PCE and DCE concentration vs. time for Scenario 1. This scenario used methane and acetate as the electron donors for the aerobes and anaerobes, respectively. The initial concentration of tetrachloroethylene (PCE) in the reactor was 4.86×10^{-5} M (8 ppm) and the influent PCE concentration was 6.03×10^{-5} M (10 ppm). The dichloroethylene (DCE) concentration in the reactor initially and in the influent were both zero. The ratio of bulk liquid to gel bead volume in this simulation was 4:1. The figure shows the effluent concentration vs. time for this scenario using the additional input parameters shown in Table 1. For more information on the program used to create the simulation, see Appendix A.

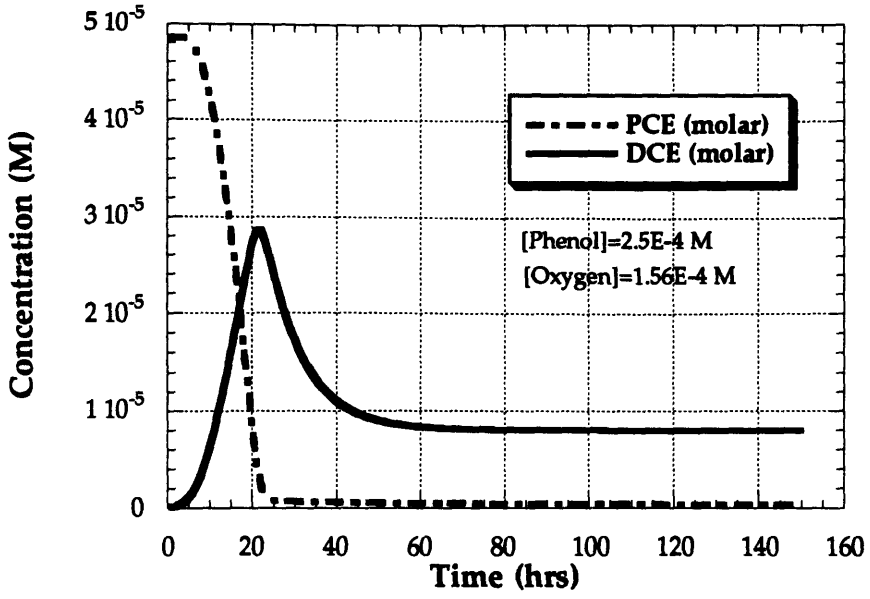

Figure 6: Plot of PCE and DCE concentration vs. time for Scenario 2. Scenario 2 used phenol as the co-substrate to be utilized by both the aerobes and anaerobes. The contaminant conditions were the same as in Scenario 1. The liquid to bead volume was the same as in Scenario 1, 4:1. This figure shows the effluent concentration vs. time for Scenario 2 using the additional input parameters shown in Table 1. For more information on the program written to create this simulation, see Appendix B.

Scenario 3

This scenario was different from the first two in that the simulation was not entirely theoretical. The goal of this scenario was to simulate an experiment conducted by Beunink and Rehm (1990). In this experiment, 4-chloro-2-nitrophenol (CNP) was added to a batch reactor with glucose as the anaerobic substrate. Using beads (as discussed in the first two scenarios) containing an aerobe, *Alcaligenes* sp. TK-2, and a facultative anaerobe *Enterobacter cloacae*, they were able to completely mineralize CNP. They showed that the only way this could happen was through a coupled aerobic and

anaerobic reaction whereby the CNP was anaerobically reduced to 4-chloro-2-aminophenol (CAP) which was then mineralized by the aerobes.

Although their experiments yielded some kinetic data regarding CNP reduction, very little data was presented dealing with CAP degradation. As a result, many of the parameters listed in Table 1 are only estimates. Another critical factor that was not specified was the ratio of bulk fluid volume to bead volume. The rate of the reactions vary directly with the number of beads in the reactor. Again, this parameter was estimated.

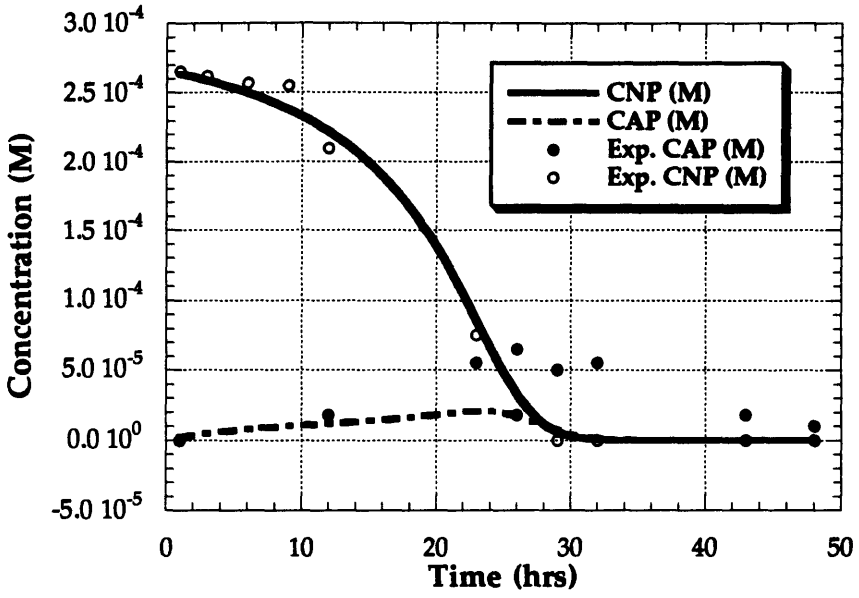

Figure 7: Plot of PCE and DCE concentration and experimental data vs. time for Scenario 3. The concentration of oxygen in the reactor was assumed to be 1.56E-4 M (~5 ppm). The experiment by Beunink and Rehm (1990)involved a 48 hour incubation of the co-immobilized bacteria in a batch reactor with 0.265 mM initial CNP concentration, no CAP, and 25 mM glucose as the substrate for both the aerobes and anaerobes with an additional 25 mM glucose added at t=23 h. During the 48 hours of incubation, 93% of the CNP was reduced. This figure shows the simulation results along with the experimental data. For more information on the program written to create this simulation, see Appendix C.

Model Validation

After each of the first two simulations was completed, an additional simulation was run as a check for the numerical code. In both cases this additional simulation had no aerobic activity. By not allowing aerobic activity to take place, the second simulation becomes a mass balance check for the computer code. The contaminant initially present in the reactor along with the subsequent contaminant inflow can only be transformed to its product in a 1:1 ratio with no mineralization taking place. In this case, the tetrachloroethylene initially present in the reactor in addition to that carried in the inflow can only be transformed to dichloroethylene, with no subsequent mineralization. Thus, the basic ethylene molecules are conserved.

The mass balance graph in each of the first two scenarios shows the concentration of the effluent in the validation (no aerobic degradation, and thus, no mass loss) simulations. The sum of the tetrachloroethylene and dichloroethylene concentrations is compared to the analytical solution of effluent from a reactor with a given initial concentration, $C_{initial}$; influent concentration, C_{in}; effluent concentration, C_{out}; volume, V; and flow, Q:

$$C_{out} = C_{in} + \left(C_{initial} - C_{in}\right) e^{-\frac{Q}{V}t} \tag{23}$$

This equation is valid for a well-mixed reactor of homogeneous concentration. The presence of the beads adds some uncertainty to the applicability of this equation to the present scenarios, due to the non-homogeneous concentration gradient that exists in each bead.

The following derivation will show that the above equation is a very close approximation of the solution for the present systems. The four equations below represent mass balances for chemical 1 in the bulk phase, chemical 2 in the bulk phase, chemical 1 in the beads, and chemical 2 in the beads, respectively.

$$V_{bulk}\frac{\partial C_{1,bulk}}{\partial t} = Q\left(C_{1,in} - C_{1,bulk}\right) - \frac{DA}{r}\left(C_{1,bulk} - C_{1,bd}\right) \tag{24}$$

$$V_{bulk}\frac{\partial C_{2,bulk}}{\partial t} = Q\left(C_{2,in} - C_{2,bulk}\right) - \frac{DA}{r}\left(C_{2,bulk} - C_{2,bd}\right) \tag{25}$$

$$V_{bd}\frac{\partial C_{1,bd}}{\partial t} = -kV_{bd}C_{1,bd} - \frac{DA}{r}\left(C_{1,bulk} - C_{1,bd}\right) \tag{26}$$

$$V_{bd}\frac{\partial C_{2,bd}}{\partial t} = kV_{bd}C_{2,bd} - \frac{DA}{r}\left(C_{2,bulk} - C_{2,bd}\right) \tag{27}$$

where

V_{bulk} = volume of the bulk free liquid (l)
V_{bd} = total volume of gel beads (l)
$C_{1,bulk}$ = concentration of chemical 1 in the bulk phase (moles/m^3)
$C_{2,bulk}$ = concentration of chemical 2 in the bulk phase (moles/m^3)
$C_{1,in}$ = concentration of chemical 1 in the influent (moles/m^3)
$C_{2,in}$ = concentration of chemical 2 in the influent (moles/m^3)
$C_{1,bd}$ = concentration of chemical 1 in the gel beads (moles/m^3)
$C_{2,bd}$ = concentration of chemical 2 in the gel beads (moles/m^3)
D = the common diffusion coefficient of both chemical 1 and 2 (m^2/sec)
r = average radius of a bead (m)
A = total surface area of all gel beads (m^2)
Q = flow through the reactor (m^3/s)

Equations (24)-(25) consist of an advection term and a sink term (diffusion into the beads from the bulk phase), while equations (26)-(27) consist of a reaction term and a source term (also, diffusion into the beads from the bulk phase). For the purposes of this derivation, the diffusion coefficients for chemicals 1 and 2 will be taken as the same value. Also, it is only through the biological reaction of chemical 1 that chemical 2 is formed, thus the equal and opposite reaction terms. Adding equations (24) and (25):

$$V_{bulk}\frac{\partial C_{T,bulk}}{\partial t} = Q\left(C_{1,in} - C_{T,bulk}\right) - \frac{DA}{r}\left(C_{T,bulk} - C_{T,bd}\right) \tag{28}$$

where

$C_{T,bulk}$ = total bulk concentration, or, $C_{1,bulk}+C_{2,bulk}$
$C_{T,bd}$ = total bead concentration, or, $C_{1,bd}+C_{2,bd}$

Now, adding equations (26) and (27):

$$V_{bd}\frac{\partial C_{T,bd}}{\partial t} = \frac{DA}{r}\left(C_{T,bulk} - C_{T,bd}\right) \tag{29}$$

Taking that the total mass of chemicals in the system, M_T, as $V_{out}C_{T,bulk}+V_{bd}C_{T,bd}$:

$$\frac{dM_T}{dt} = V_{bulk}\frac{dC_{T,bulk}}{dt} + V_{in}\frac{dC_{T,bd}}{dt} \tag{30}$$

Substituting equations (28) and (29) into (30):

$$\frac{dM_T}{dt} = Q\left(C_{T,in} - C_{T,bulk}\right) \tag{31}$$

and writing an expression for the average total concentration, $\overline{C}_T$:

$$\overline{C}_T = \frac{M_T}{V_{bulk} + V_{bd}} = \frac{M_T}{V_T} \tag{32}$$

Now, taking the derivative with respect to time of equation (32) and substituting equation (31) yields:

$$\frac{d\overline{C}_T}{dt} = \frac{dM_T}{dt}\frac{1}{V_T} = \frac{Q}{V_T}\left(C_{T,in} - C_{T,bulk}\right) \tag{33}$$

If equation (33) is integrated and a solution found if the initial reactor concentration of chemical 1 is $C_{initial}$, and chemical 2 is zero, the result is equation (23)

written in terms of the average total concentration. Also, if the influent concentration of chemical 2 is zero, then, $C_{T,in}=C_{1,in}=C_{in}$, and:

$$\overline{C}_T = C_{in} + \left(C_{initial} - C_{in}\right) e^{-\frac{Q}{V_T}t} \tag{34}$$

There is one important difference between equations (23) and (34). Equation (23) is an expression giving the effluent concentration represented by the concentration of the free liquid, C_{bulk}, while equation (34) expresses the effluent concentration as the average concentration of the bulk liquid and beads. Therefore, if there is a significant difference in the total concentration of chemicals 1 and 2 in the beads and in the bulk liquid, equation (34) and not (23) would have to be used to determine the effluent concentration from the reactor in the simulations. In order for the use of equation (23) to be valid in the simulations, C_{bulk} must very nearly approximate $\overline{C}_T$, which requires $C_{T,bd}$ be equal to $C_{T,bulk}$. Using the data generated from the simulations, it was shown that $C_{T,bd}$ and $C_{T,bulk}$ differed by less that one percent, thus providing the first verification of the applicability of equation (23) to these simulations.

The quantity $C_{T,bulk}-C_{T,bd}=\Delta C_T$ can also be shown to be small on the condition that the hydraulic retention time is much greater than the time for a chemical to diffuse to the center of a bead. Subtracting equation (29) from (28):

$$\frac{\partial}{\partial t}\left(C_{T,bulk} - C_{T,bd}\right) = \frac{Q}{V_T}\left(C_{in} - C_{T,bulk}\right) - \frac{2D}{r^2}\left(C_{T,bulk} - C_{T,bd}\right) \tag{35}$$

Now, substituting ΔC_T and dividing by $2D/r^2$:

$$\Delta C_T = \frac{\frac{Q}{V_T}\left(C_{in} - C_{T,bulk}\right)}{\frac{2D}{r^2}} - \frac{\frac{\partial \Delta C_T}{\partial t}}{\frac{2D}{r^2}} \tag{36}$$

Upon examination of equation (36) it can be determined that ΔC_T is small. The derivative term on the right side of the equation will be taken as negligible since the gradient can only change as fast as the bulk concentration changes, which is slow compared to diffusion. The hydraulic retention time for the first two simulations is 2 days, while the time for diffusion to occur into the center of each bead, given by r^2/D is on the order of 40 minutes. Therefore, diffusion is rapid compared to the change in the bulk concentration in the reactor even considering biological use of the chemicals. For this same reasoning, the first term in the equation is also small. Assuming that the concentration difference is small (<1M) the first term becomes the ratio of the inverses of the hydraulic detention time and the diffusion time. Since the value of this term is small (<2%) it shows qualitatively how ΔC_T should also be a small value, with a maximum on the order of a few percent. Due to this small error, equation (23) does not exactly describe the effluent from the reactor in the simulations, however, it is a very close approximation that will be used for the verification of the computer program written to solve the numerical scheme.

Determination of Mass Conservation

Figure 6 shows the mass balance check for the program used in Scenario 1. In this case, PCE was transformed to DCE, but any further reaction was prevented. This allows the PCE to DCE reaction with a subsequent accumulation of DCE. The curves for PCE and DCE effluent are added together to form a curve for total concentration (Total). This total concentration is then compared to concentration data points calculated from equation (23) (Analytical).

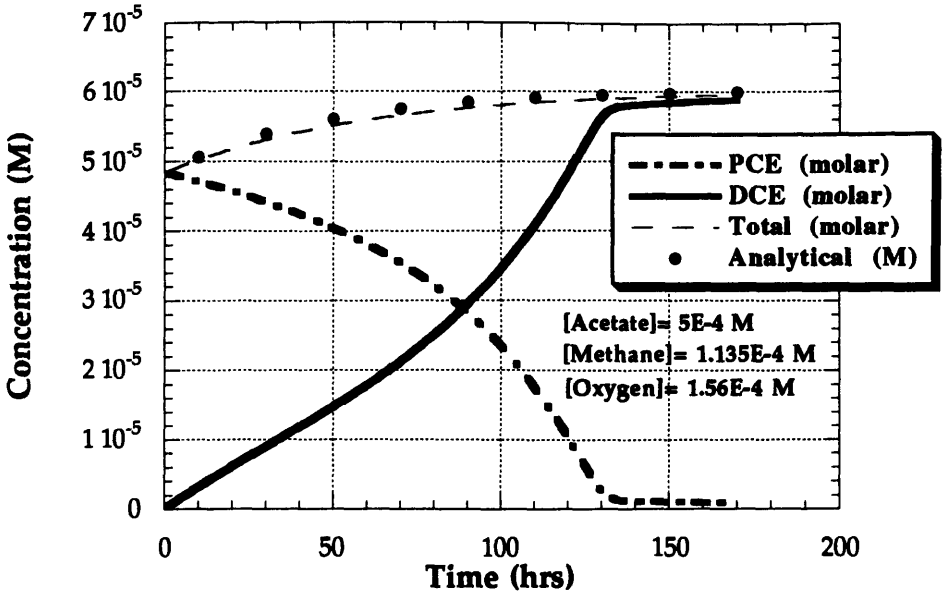

Figure 8: Scenario 1 mass balance comparing simulated data with an analytical solution. This graph shows the effluent concentration of the validation experiment for Scenario 1. The initial and influent concentrations of PCE, DCE, and substrates are all identical to those used in Scenario 1. However, in this case, there is no aerobic degradation, so DCE accumulates as it is produced from the degradation of PCE.

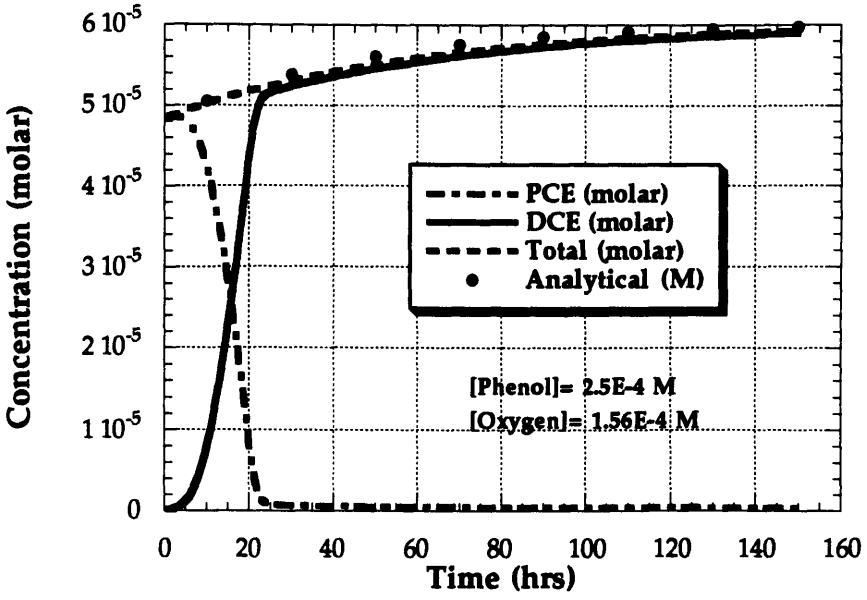

Figure 9: Scenario 2 mass balance comparing simulated data with an analytical solution. This graph shows the effluent concentration of the validation experiment for Scenario 2. The initial and influent concentrations of PCE, DCE, and phenol are all identical to those used in Scenario 2. However, in this case, there is no aerobic degradation, so DCE accumulates as it is produced from the degradation of PCE.

Chemical Gradients

More insight into these simulations can be gained by examining the chemical concentration gradients within the gel beads. Perhaps the most important concentration gradient is that of oxygen. The ability for an anaerobic process to take place on the interior of the bead requires that the majority of the oxygen be used near the outer radii of the beads. The oxygen concentration gradients for Scenarios 1, 2, and 3 are shown in Figure 10.

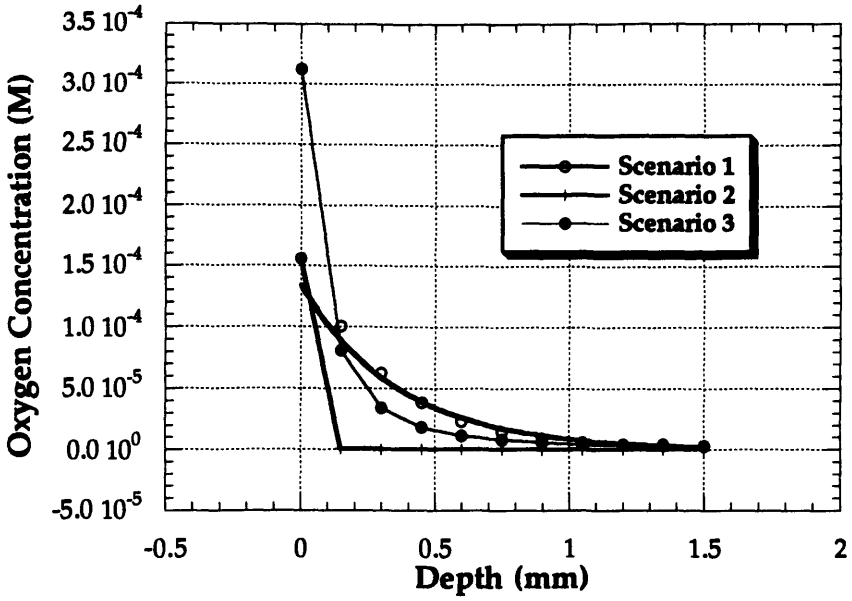

Figure 10: Oxygen concentration gradients within beads of Scenarios 1 and 2 after 150 hours of incubation, and Scenario 3 after 48 hours of incubation.

It is useful to compare the simulated oxygen gradients within each bead to that in beads that have been used in an experiment. Beunink et al. (1989) measured the oxygen gradients within 3 mm Ca-alginate beads using an oxygen microelectrode. The beads had an initial cell concentration of 2×10^8 cells/ml gel bead, and were suspended in aqueous medium containing glucose as a substrate, and a constant aeration rate of 1.7 vvm. The medium to bead ratio was 2:1. Figure 11 shows the concentration of oxygen in the beads at different radii.

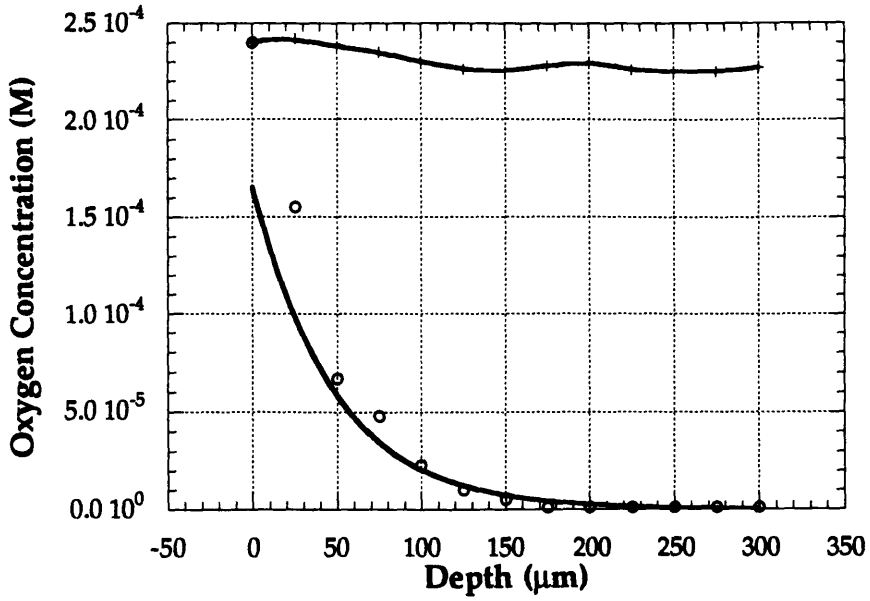

Figure 11: Oxygen concentration in the bead at different radii as measured by an oxygen microelectrode. The beads were incubated in a glucose medium with constant aeration (o-bead concentration after 6 hour incubation, + - oxygen concentration at time zero).

The fitted curve for Scenario 1 in Figure 10 and the curves in Figure 11 are exponentials. Due to an extremely high gradient near the surface of bead in Scenarios 2 and 3, an exponential does not well describe the shape. However, there are significant qualitative similarities between the simulated data of Scenario 1 and the experimental data presented by Beunink et al. (1989) shown in Figure 11.

The PCE concentration gradients in Scenarios 1 and 2 are shown in Figure 12. Since the PCE is degraded near the interior of the bead, the shapes displayed by the curves are reasonable. Near the outer radii of the beads the concentration is nearly equal to that in the bulk phase, but as the interior of the bead is approached, and oxygen

concentration becomes limited (as shown by Figure 10), the PCE concentration begins to decline exponentially.

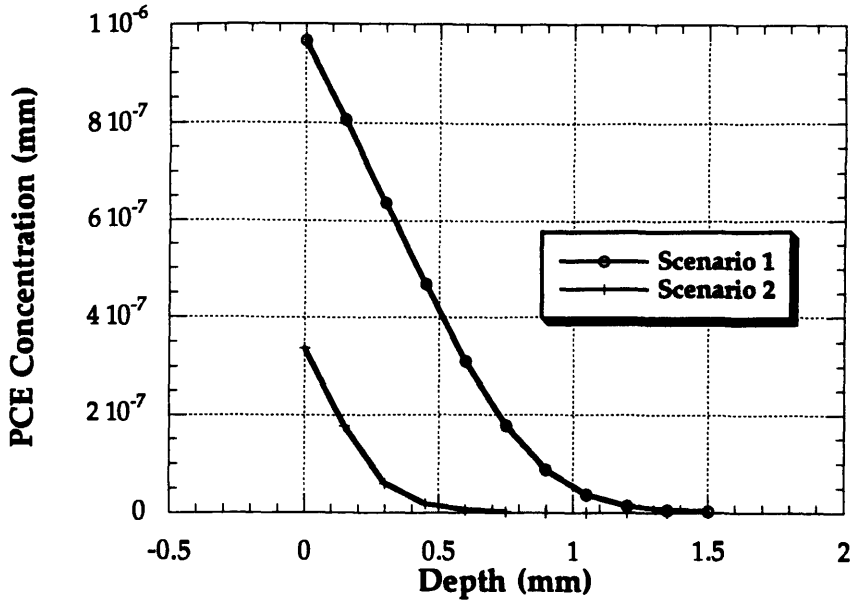

Figure 12: PCE concentration gradients in beads of Scenarios 1 and 2 after 150 hours of incubation.

The DCE concentration gradient, however, is not nearly as high as the PCE gradient, and in the opposite direction. This is because the DCE is being produced near the center of the bead where the anaerobic reaction is taking place. As the DCE diffuses outward in the bead it moves toward areas of higher oxygen concentration where it is mineralized. However, not all the DCE is degraded before it diffuses into the bulk phase. These DCE molecules are then mixed into the bulk liquid and are free to diffuse into the surface of any bead where they are degraded. Figure 13 shows the DCE concentration gradients in beads of Scenarios 1 and 2.

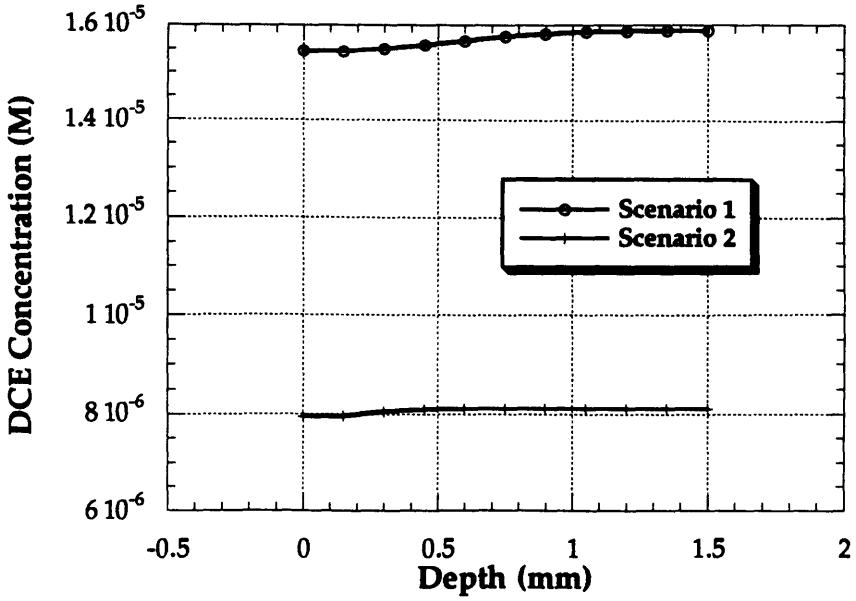

Figure 13: DCE concentration gradient in beads of Scenarios 1 and 2 after 150 hours of incubation.

In the first two scenarios the concentrations of the substrates within the beads remains relatively constant (see Figure 14) due to the fact that these concentrations are held constant in the bulk phase at levels much higher than the concentrations of the contaminants. The change in concentration of the substrates is insignificant compared to the concentration of the contaminant. The substrate concentration gradients for Scenario 1 after 150 hours of incubation are shown in Figure 14. In Scenario 2 the substrate concentration was 2.5×10^{-4} M throughout the bead after 150 hours.

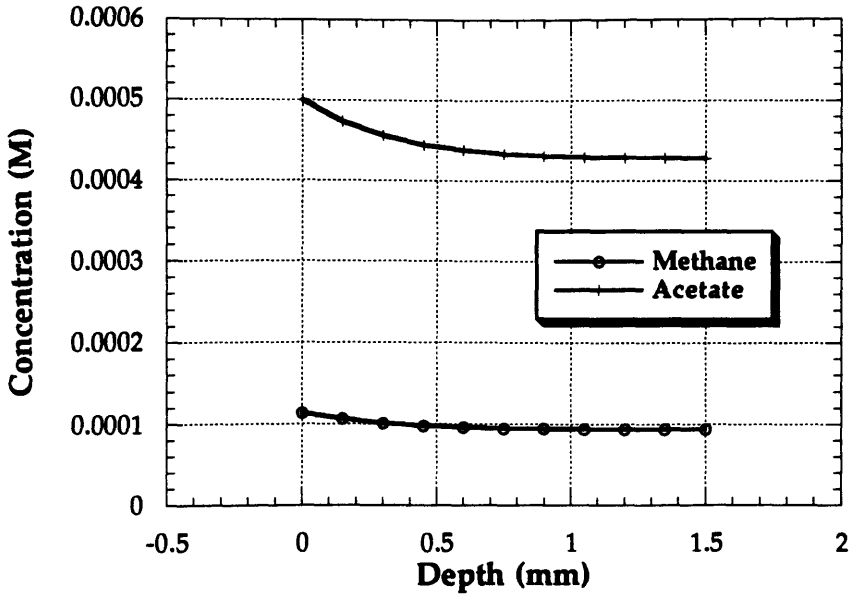

Figure 14: Methane and acetate concentrations plotted as a function of depth in beads of Scenario 1 after 150 hours of incubation.

The concentration gradients of Scenario 3 are less enlightening due to the fact that it was a batch test. The CNP and CAP gradients in the beads of this scenario were quite small and consisted of concentrations nearly ten orders of magnitude less than those shown in the previous two scenarios.

DISCUSSION

This study was undertaken in an attempt to show and predict the existence of an anaerobic zone in the interior of Ca-alginate gel beads immobilizing a mixed culture of microorganisms, and that this anaerobic zone could be exploited to couple oxidative and reductive reactions that are necessary for the biodegradation of certain chlorinated organic compounds. The results from the simulations predicts the existence of an

anaerobic zone within the gel beads. A model was developed which can be used once the appropriate parameters are determined. This model can be used to optimize a biodegradation study prior to actually carrying out the bioreactor experiment.

Figures 5 and 6 showing results for Scenarios 1 and 2, respectively, predict that it is possible to gain a significant reduction in the concentration of tetrachloroethylene which requires both an anaerobic and aerobic process for mineralization. Figure 6 shows that at near t=70 hrs in Scenario 2, steady state is approached. At steady state, however, there is still a significant amount of DCE that remains without further degradation by the aerobes. This is because of the inhibitive effects of phenol which has been modeled as a non-competitive inhibitor. Since the expression for the degradation of DCE depends on both the DCE concentration and the phenol concentration, as the DCE concentration gets small, the phenol, and hence the inhibitory effects of phenol begin to dominate the equation. The DCE concentration is not constant, rather the degradation slows considerably to the point where the decrease in concentration over time is too small to be clearly seen on the graph. The parameters governing the function of these simulations (e.g.; retention time, bulk oxygen concentration, number of beads, etc.) could be adjusted to optimize the mineralization of PCE.

Scenario 3 was created to test the prediction of the model with experimental data generated by Beunink and Rehm (1990). The results of this scenario in Figure 7 show that there is acceptable agreement between the simulation data and the experimental data. The CNP effluent concentration fits well, while the CAP concentrations remain lower and peak earlier than those shown by the experimental data. However, in light of the fact that many of the parameters had to be estimated, the agreement is acceptable.

The concentration gradients shown in Figures 10-14 are consistent with what is to be expected. The oxygen gradient in Scenario 1 shown in Figure 10 approaches an exponential with a high gradient near the surface. This is consistent with experimental data generated by Beunink et al. (1989); also shown in Figure 10. The predicted oxygen

concentration in Scenario 2 decreased much more rapidly than in Scenario 1 most likely due to the value of μ_{max} for phenol being higher than acetate.

The PCE concentration gradients shown in Figure 12 are also intuitive due to the consumption of PCE and production of DCE in the anaerobic zone of the beads which creates opposing gradients with the highest concentration of PCE near the surface of the bead and the highest concentration of DCE at the center of the bead. The DCE concentration gradients shown in Figure 13 approaches horizontal with very little change in concentration over the radius of the bead, although the simulated data for Scenario 1 shows a decreased concentration toward the outside of the bead, which is consistent with expectations.

Cometabolism

It is important to note here an important factor governing the cometabolic reactions in the first two scenarios. Since cometabolism depends on the presence of a growth substrate (electron donor) for the stimulation of production of the enzymes that can fortuitously degrade DCE and many other compounds, the concentration of the electron donor must always be significant compared to the concentration of the contaminant. As discussed previously, Semprini and McCarty (1992) used an expression that determined whether the microorganisms were increasing in number from growth on the electron donor to determine the cometabolic activity. If there was enough electron donor present that the net number of cells was increasing, the cometabolic process continued unimpeded. This reasoning requires one to assume a death rate. Death rate being constant, when the electron donor decreases in concentration to the point where the death rate exceeds the growth rate, and the bacterial population on whole begins to decrease in number, all cometabolic processes drop off rapidly.

This phenomenon is supported by field studies (Semprini, et al., 1990, 1991) and laboratory experience (Alvarez-Cohen and McCarty, 1991; Henry and Grbic-Galic, 1991). These studies observed that in the case of methane, as in Scenario 1, when the

growth substrate was removed, the methane monooxygenase enzyme was deactivated and all cometabolic reactions rapidly ceased. In this simulation, we chose not to use the death rate activation-deactivation of cometabolic activity because we believed that cell death at a constant rate was not likely to occur in our simulations. For the purposes of these simulations, cometabolic activity took place only if the electron donor concentration was greater than the contaminant concentration. If this was not the case, no cometabolism would take place. It did not impact these simulations since the electron donor was always far in excess of the contaminant.

Parameter Variation

Although a change in any parameter changes the output of the simulation, some parameters are more critical than others. The Monod half-saturation constants that appear in the denominator of equations of form (13)-(16) have only a relatively small influence on the shape of the effluent curve while parameters such as the maximum growth rate, microbial yield, and initial biomass have a greater effect on the simulation output. As either the maximum growth rate or initial biomass increases, the biodegradation becomes faster and peaks in the effluent graphs are attenuated. However, if the cell yield is increased, biodegradation slows peaks in the effluent graphs are accentuated. The bulk oxygen concentration also plays an important role in determining the resultant degradation reactions. More oxygen in the system leads to more aerobic degradation and less anaerobic degradation. Perhaps the most important parameter is KIO. Even one order of magnitude change in this parameter produces a large variation in the simulation. A higher value of KIO results in less anaerobic activity due to the inability of the anaerobes to function at the oxygen concentration given by KIO. This produces an effluent graph that nears horizontal, or shows no degradation until farther out in time. If KIO is decreased the anaerobic reaction moves very quickly and much of the contaminant is degraded in a very short time.

REFERENCES

1. Abelson, P. H. 1990. Inefficient remediation of ground-water pollution. Science 250:73.
2. Alvarez-Cohen, L., and P. L. McCarty. 1991. Product toxicity and cometabolic competitive inhibition modeling of chloroform and trichloroethylene transformation by methanotrophic resting cells. Appl. Environ. Microbiol. 57n4:1031-1037.
3. Barik, S., W. J. Brulla, and M. P. Bryant. 1985. PA-1, a versatile anaerobe obtained in pure culture, catabolized benzenoids and other compounds in syntrophy with hydrogenotrophs, and P-2 plus *Wolinella* sp. degrades benzenoids. Appl. Environ. Microbiol. 50n2:304-310.
4. Benthin, S., J. Nielsen, and J. Villadsen. 1992. Anomeric specificity of glucose uptake systems in *Lactococcus cremoris*, *Escherichia coli*, and *Saccharomyces cerevisiae*: mechanism, kinetics, and implications. Biotech. Bioeng. 40:137-146.
5. Beunink, J., H. Baumgärtl, W. Zimelka, and J. J. Rehm. 1989. Determination of oxygen gradients in single Ca-alginate beads by means of oxygen-microelectrodes. Experientia 45:1041-1047.
6. Beunink J. and H. J. Rehm. 1990. Coupled reductive and oxidative degradation of 4-chloro 2-nitrophenol by a co-immobilized mixed culture system. Appl. Microbiol. Biotechnol. 34:108-115.
7. Beunink J. and H. J. Rehm. 1988. Synchronous anaerobic and aerobic degradation of DDT by an immobilized mixed culture system. Appl. Microbiol. Biotechnol. 29:72-80.
8. Bouwer, E. J., B. E. Rittmann, and P. L. McCarty. 1981. Anaerobic degradation of halogenated 1- and 2-carbon organic compounds. Environ. Sci. Technol. 15:596-599.
9. Brock, T. D., and M. T. Madigan. Biology of microorganisms, Sixth Edition. Prentice-Hall Inc. 1991. ppg. 830-832.
10. DiStefano, T. D., J. M. Gossett, and S. H. Zinder. 1992. Hydrogen as an electron donor for Dechlorination of tetrachloroethene by an anaerobic mixed culture. Appl. Environ. Microbiol. 58n11:3622-3629.
11. Dorn, E., and H.-J. Knackmuss. 1978. Chemical and Biodegradability of Halogenated Aromatic Compounds. Biochem. J. 174:85-94.
12. Dwyer, D. F., M. L. Krumme, S. A. Boyd, and J. M. Tiedje. 1986. Kinetics of phenol biodegradation by an immobilized methanogenic consortium. Appl. Environ. Microbiol. 52n2:345-351.
13. Fox, P., and M. T. Suidan. 1990. Batch tests to determine activity distribution and kinetic parameters for acetate utilization in expanded-bed anaerobic reactors. Appl. Environ. Microbiol. 56n4:887-894.
14. Gosmann, B., and H. J. Rehm. 1988. Influence of growth behaviour and physiology of alginate-entrapped microorganisms on the oxygen consumption. Appl. Microbiol. Biotechnol. 29:554-559.
15. Gossett, James. personal communication. University of Oregon Department of Civil Engineering.
16. Hayduk, W., and H. Laudie. 1974. Prediction of diffusion coefficients for non-electrolytes in dilute aqueous solutions. J. AIChE. 20:611-615.
17. Henry, S. M. and D. Grbic-Galic. 1991. The effects of mineral media on trichloroethylene oxidation by aquifer methanotrophs. Microb. Ecol. 20:106-137.

18. Hopkins, G. D., L. Semprini, and P. L. McCarty. 1993. Microcosm and in situ field studies of enhanced biotransformation of trichloroethylene by phenol-utilizing microorganisms. Appl. Environ. Microbiol. 59n7:2277-2285.
19. Kotturi, G., C. W. Robinson, and W. E. Inniss. 1991. Phenol degradation by psychrotrophic strain of *Pseudomonas putida*. Appl. Microbiol. Biotechnol. 34:539-543.
20. Krumholz, L. 1993. Unpublished data.
21. Little C. D., A. V. Palumbo, S. E. Herbes, M. E. Lidstrom, R. L. Tyndall, and P. J. Gilmer. 1988. Trichloroethylene biodegradation by a methane-oxidizing bacterium. Appl. Environ. Microbiol. 54n4:951-956.
22. Mathews, C. K., and K. E. van Holde. 1990. Biochemistry. The Benjamin/Cummings Publishing Company, Inc.
23. Molz, F. F., M. A. Widdowson, and L. D. Benefield. 1986. Simulation of microbial growth dynamics coupled to nutrient and oxygen transport in porous media. Wat. Res. Res. 22n8:1207-1216.
24. Nilsson, K., S. Birnbaum, S. Flygare, L. Linse, U. Schröder, U. Jeppsson, P. Larsson, K. Mosbach, and P. Brodelius. 1983. A general method for the immobilization of cells with preserved viability. Eur. J. Appl. Microbiol. Biotechnol. 17:319-326.
25. Ohtsubo, S., K. Demizu, S. Kohno, I. Miura, T. Ogawa, and H. Fukuda. 1992. Comparison of acetate utilization among strains of an aceticlastic methanogen, *Methanothrix soehngenii.* Appl. Environ. Microbiol. 58n2:703-705.
26. Oldehhuis, R., R. L. J. M. Vink, D. B. Janssen, and b. Witholt. 1989. Degradation of chlorinated aliphatic hydrocarbons by *Methylosinus trichosporium* OB3b expressing soluble methan monooxygenase. Appl. Environ. Microbiol. 55n11:2819-2826.
27. Othmer, D. F., and M. S. Thakar. 1953. Correlating diffusion coefficients in liquids. Ind. Eng. Chem. 45:589-593.
28. Scholz-Muramatsu, H., R. Szewzyk, U. Szewzyk, and S. Gaiser. 1990. Tetrachloroethylene as electron acceptor for the anaerobic degradation of benzoate. FEMS Microbiol. Letters 66:81-86.
29. Semprini, L., and P. L. McCarty. 1992 Comparison between model simulations and field results for in-situ biorestoration of chlorinated aliphatics: Part 2. Cometabolic transformations. Ground Water 30n1:37-44.
30. Semprini, L., and P. L. McCarty. 1991. Comparison between model simulations and field results for *in-situ* biorestoration of chlorinated aliphatics: Part 1. Biostimulation of methanotrophic bacteria. Ground Water 29n3:365-374.
31. Semprini, L., P. V. Roberts, G. D. Hopkins, and P. L. McCarty. 1990. A field evaluation of in-situ biodegradation of chlorinated ethenes. Part 2, The results of biostimulation and biotransformation experiments. Ground Water 28:715-727.
32. Vogel, T. M., and P. L. McCarty. 1985. Biotransformation of tetrachloroethylene to trichloroethylene, dichloroethylene, vinyl chloride, and carbon dioxide under methanogenic conditions. Appl. Environ. Microbiol. 49n5:1080-1083.
33. Wackett, L. P., and D. T. Gibson. 1988. Degradation of trichloroethylene by toluene dioxygenase in whole-cell studies with *Pseudomonas putida* F1. Appl. Environ. Microbiol. 54n7:1703-1708.
34. Zehnder, A. J. B. 1988. The Biology of Anaerobic Microorganisms. Hogn Weley & Sons. p. 213-215.
35. Zitomer, D. H., and R. E. Speece. 1993. Sequential environments for enhanced biotransformation of aqueous contaminants. Environ. Sci. Tech. 27:227-244.

Appendix A
C Code for Scenario 1

```
#include <stdio.h>
#define PI 3.14159265359
#define DO2 1.5e-5                  /* The D definitions refer to the
                                       corresponding diffusivities */

#define DCH4 2.1e-5
#define DSUB 1e-5
#define DPCE 6.58e-6
#define DDCE 8.4e-6
#define KPCE 5e-7                   /* The K definitions refer to the
                                       corresponding Monod constants */

#define KSUB 6e-4
#define KSUBA 1.6e-4                /* K for anaer. sub. util. by aerobes */
#define KCH4 1.25e-4
#define KO2 3.13e-5
#define KDCE 1.03e-5
#define KI 1e-7
#define anyield 6.25e11             /* anaerobic cell yield in cells per
                                       mole substrate */
#define aeryield 8e12               /* aerobic cell yield in cells per
                                       mole CH4 */
#define DCEmur 8.68e-7              /* DCE rx maximum util. rate */
#define test 0.5                    /* the numerics test factor (<1) */
#define CGDW 1e12                   /* number of cells per gram dry weight */


struct datastruct
{
  int numofbeads, time, simtime, divisions;
  long writeinterval;
  double radius, reacvol, conc02in, concsubin, deltat, DCEmmax,
         concCH4in, concPCEin, flow, deltar, anmmax, aermmax,
         initaercelldensty, initancelldensty, O2attime, CH4attime,
         subattime;
  double **DCEbdconc, **aerbiomass, **O2bdconc, **CH4bdconc,
         **PCEbdconc, **anbiomass, **subbdconc;
  double *PCEattime, *DCEattime;
  FILE *ofp, *ofp2, *ofp3;
} data;

float sqr (float x)
{
  return x*x;
}

float cub (float x)
{
  return x*x*x;
}

int inputdata (struct datastruct *dtptr)
{
  dtptr->radius = 1.5;                  /* Average radius of beads in mm */

  dtptr->divisions = 10;                /* Numb. of radii used in numerics */

  dtptr->numofbeads = 7000;             /* Number of beads in reactor */

  dtptr->anmmax = 4e-6;                 /* Anaer. doubling time in inv sec */

  dtptr->initancelldensty = 1e8;        /* In. anaer. cell dsty in cells/ml */

  dtptr->aermmax = 1e-5;                /* Aer. doubling time in inv. sec */

  dtptr->initaercelldensty = 1e8;       /* In. aer. cell dsty in cells/ml */

  dtptr->reacvol = 0.401;               /* Reactor fluid volume in liters */

  printf("Input the flow in l/s -> ");
  scanf("%lf", &dtptr->flow);           /* Flow in l/s */

  dtptr->deltat = 0.1;                  /* Timestep in sec  */

  printf("Input simtime -> ");          /* Simulation time in sec */
  scanf("%ld", &dtptr->simtime);

  printf("Input the write interval (sec) -> ");
  scanf("%ld", &dtptr->writeinterval);  /* Interval (in number of sec) at
                                           which results should be written
                                           to a file. e.g.; every 1 hr
                                           would require input of 3600 (sec) */

  printf("Input the constant oxygen concentration -> ");
  scanf("%lf", &dtptr->O2attime);       /* constant oxygen conc. in molar */

  printf("Input the constant methane concentration -> ");
  scanf("%lf", &dtptr->CH4attime);      /* constant methane conc. in molar */

  printf("Input the constant substrate concentration -> ");
  scanf("%lf", &dtptr->subattime);      /* constant substr. conc in molar */

  dtptr->concPCEin = 6.03e-5;           /* PCE influent conc. in molar */

  return 0;
}

int main (void)
{
  int inputdata (struct datastruct *);
  int operate  (struct datastruct *);
  struct datastruct *dtptr;

  dtptr = &data;
  inputdata (dtptr);
  operate(dtptr);

  return 0;
}

int operate (struct datastruct *dtptr)
{
  int chemfunctime (struct datastruct *dtptr);

/*  calculate the delta r from the given radius and number of divisions  */

  dtptr->deltar = dtptr->radius/dtptr->divisions;

/*  convert the timesteps into integer values for use in array  */

   dtptr->time = (int) (dtptr->simtime/dtptr->deltat);

   chemfunctime(dtptr);         /* called to simulate conc. in reactor
                                   and beads over time */

  return 0;
}
```

```
int writeoutput (struct datastruct *dtptr, int wrttime, int i)
{
  int j;
  static int static_x = 0;

  if (static_x == 0)
    {
      dtptr->ofp = fopen ("effluent", "w");
      fprintf(dtptr->ofp,"                BIOLOGICAL REACTOR SIMULATION RESULTS\n\n");
      fprintf(dtptr->ofp,"      Radius= %6.2f mm\n", dtptr->radius);
      fprintf(dtptr->ofp,"      Number of beads= %6d              Anaer. mmax= %.3e sec-1\n
",
              dtptr->numofbeads, dtptr->anmmax);
      fprintf(dtptr->ofp,"      Initial anaer. cell density= %.3e cells/ml\n",
              dtptr->initancelldensty);
      fprintf(dtptr->ofp,"      Aer. mmax= %.3e sec-1          In. aer. cell dnsty= %.3e c
ells/ml\n", dtptr->aermmax, dtptr->initaercelldensty);
      fprintf(dtptr->ofp,"      DCEmur= %.3e sec-1\n", DCEmur);
      fprintf(dtptr->ofp,"      Reactor volume= %8.3f l          Flow= %.3e l/s\n",
              dtptr->reacvol, dtptr->flow);
      fprintf(dtptr->ofp,"      Delta t= %5.3f sec                  Simulation time= %d se
c\n",
              dtptr->deltat, dtptr->simtime);
      fprintf(dtptr->ofp,"      Write interval= %6ld sec          Divisions= %d\n\n",
              dtptr->writeinterval, dtptr->divisions);
      fprintf(dtptr->ofp,"                      Constant Concentrations (molar)\n");
      fprintf(dtptr->ofp,"                     ---------------------------------\n\n");
      fprintf(dtptr->ofp,"                 Oxygen= %.3e          Methane= %.3e\n",
              dtptr->O2attime, dtptr->CH4attime);
      fprintf(dtptr->ofp,"              Substrate= %.3e\n", dtptr->subattime);
      fprintf(dtptr->ofp,"                 Influent  PCE= %.3e\n\n\n",
              dtptr->concPCEin);
      fprintf (dtptr->ofp,"                      Effluent Concentrations (molar)\n");
      fprintf (dtptr->ofp,"                     ---------------------------------\n\n");
      fprintf (dtptr->ofp,"  Time          PCE         DCE         Oxygen  ");
      fprintf (dtptr->ofp,"   Methane     Acetate  \n");
      fprintf (dtptr->ofp," ------       -----       -----       -------- ");
      fprintf (dtptr->ofp,"  ---------   --------- \n");
      fclose (dtptr->ofp);
      dtptr->ofp = fopen("effluent", "a");

      dtptr->ofp2 = fopen("bdconc", "w");
      fprintf (dtptr->ofp2,
               "\n\n                                Bead Concentration\n");
      fprintf (dtptr->ofp2,
               "                                  ---------------------\n\n");
      fprintf (dtptr->ofp2," Radius         PCE         DCE         Oxygen  ");
      fprintf (dtptr->ofp2,"   Methane     Acetate  \n");
      fprintf (dtptr->ofp2," ------       -----       -----       -------- ");
      fprintf (dtptr->ofp2,"  ---------   --------- \n");
      fclose (dtptr->ofp2);
      dtptr->ofp2 = fopen ("bdconc", "a");

      dtptr->ofp3 = fopen("cellnum", "w");
      fprintf(dtptr->ofp3,"                   Cell Numbers at each radii\n");
      fprintf(dtptr->ofp3,"                  ----------------------------\n\n\n");
      fprintf(dtptr->ofp3,"Radius          Aerobic          Anaerobic\n");
      fprintf(dtptr->ofp3,"------         ---------         ----------\n\n");
      fclose(dtptr->ofp3);
      dtptr->ofp3 = fopen ("cellnum", "a");
    }

  fprintf(dtptr->ofp,"%6d      %9.3e   %9.3e   %9.3e   %9.3e   %9.3e\n",
          wrttime, dtptr->PCEattime[i], dtptr->DCEattime[i],
          dtptr->O2attime, dtptr->CH4attime, dtptr->subattime);

  fprintf (dtptr->ofp2,"\n\nTime (sec) = %d\n", wrttime);

  for (j=0; j <= dtptr->divisions; j++)
    {
      fprintf(dtptr->ofp2,
              "%5d       %9.3e   %9.3e   %9.3e   %9.3e   %9.3e\n",
              j, dtptr->PCEbdconc[i][j], dtptr->DCEbdconc[i][j],
              dtptr->O2bdconc[i][j], dtptr->CH4bdconc[i][j],
              dtptr->subbdconc[i][j]);
    }

  fprintf (dtptr->ofp3,"\n\nTime (sec) = %d\n", wrttime);

  for (j=0; j <= dtptr->divisions; j++)
    {
      fprintf(dtptr->ofp3,
              "%5d          %9.3e          %9.3e\n",
              j, dtptr->aerbiomass[i][j], dtptr->anbiomass[i][j]);
    }

  ++static_x;
}


int initbdconczero (double **bdconc, int divisions)
{
    int j;

    for (j=0; j <= divisions; j++)
      {
        bdconc[0][j]=0;
        bdconc[1][j]=0;
      }                                 /* Initializes the bead concentration
                                           of the input chemical to zero at
                                           all radii*/
    return 0;
}


int initbiomass(double **aerbiomass, double **anbiomass, int divisions,
                double initaercelldensty, double initancelldensty,
                double *v)
{
  int j;

  for (j=0; j <= divisions; j++)
    {
      aerbiomass[0][j] = initaercelldensty*v[j]*1000; /* The *1000 converts
                                                        volume (calculated
                                                        in liters) to ml */
      anbiomass[0][j] = initancelldensty*v[j]*1000;
    }
}


double  doPCEbdconc (double deltat, double deltar, double antrm,
                     double **PCEbdconc, int i, int j)
{
  double x;

  /* calculates the PCE concentration at a given radius and time in a bead
     from both diffusion (x) and anaerobic utilization (antrm) */
```

```
  x = deltat*(DPCE*((PCEbdconc[i-1][j+1] - 2*PCEbdconc[i-1][j]+
                     PCEbdconc[i-1][j-1])*100/sqr(deltar)) +
                       2*DPCE*100/(j*deltar)*
                         ((PCEbdconc[i-1][j]-PCEbdconc[i-1][j-1])/deltar))
                           + PCEbdconc[i-1][j];
  if ( x-(2*antrm) >= 0 )  /* the two is a stoichiometric conversion factor */
    return x-(2*antrm);
  else
    return 0;

}


double dosubbdconc(double deltat, double deltar, double antrm,
                   double **subbdconc, int i, int j, double subaertrm)
{
  double x;

  /* calculates the substrate conc. at a given radius and time in a bead
     from diffusion (x), aerobic utilization (subaertrm), and anaerobic
     utilization (antrm) */

  x = deltat*(DSUB*((subbdconc[i-1][j+1] - 2*subbdconc[i-1][j]+
                     subbdconc[i-1][j-1])*100/sqr(deltar))+
                       2*DSUB*100/(j*deltar)*
                         ((subbdconc[i-1][j]-subbdconc[i-1][j-1])/deltar))+
                           subbdconc[i-1][j];
  if ( x-antrm-subaertrm >= 0 ) /* 1 coefficients are from stoichiometry  */
    return x-antrm-subaertrm;
  else
    return 0;
}


double doDCEbdconc(double deltat, double deltar, int i, int j, double
                   **DCEbdconc, double antrm, double **aerbiomass, double
                   **CH4bdconc, double v, double aermmax, double **O2bdconc,
                   double DCEaertrm)

{
  double x, ret;

  /* calculates the DCE conc. at a given radius and time in a bead
     from diffusion (x), aerobic utilization (DCEaertrm), and anaerobic
     generation (antrm) */

  x = deltat*(DDCE*((DCEbdconc[i-1][j+1] - 2*DCEbdconc[i-1][j]+
                     DCEbdconc[i-1][j-1])*100/sqr(deltar))+
                        2*DDCE*100/(j*deltar)*
                          ((DCEbdconc[i-1][j]-DCEbdconc[i-1][j-1])/deltar))+
                            DCEbdconc[i-1][j];

  if (CH4bdconc[i-1][j] > 0)
    ret = x -DCEaertrm *(2*antrm); /* the 1 and 2 coeffs. are from stoich. */
  else
    ret = x + (2*antrm);

  if (ret >= 0)
    return ret;
  else
    return 0;
}
```

```
double doO2bdconc (double deltat, double deltar, int i, int j, double
                   **O2bdconc, double aertrm, double subaertrm, double
                   DCEaertrm)
{
  double x;

  /* calculates the oxygen conc. at a given radius and time in a bead
     from diffusion (x), aerobic utilization (aertrm), cometabolic use
     (DCEaertrm), and aerobic utilization of anaerobic substrate (subaertrm) */

  x = deltat*(DO2*((O2bdconc[i-1][j+1] - 2*O2bdconc[i-1][j]+
                    O2bdconc[i-1][j-1])*100/sqr(deltar))+
                      2*DO2*100/(j*deltar)*
                        ((O2bdconc[i-1][j]-O2bdconc[i-1][j-1])/deltar))
                          + O2bdconc[i-1][j];


  if ( x-(2*aertrm)-(2*subaertrm)-(2*DCEaertrm) >= 0 )
    return x-(2*aertrm)-(2*subaertrm)-(2*DCEaertrm);  /* 2's are from stoich.*/
  else
    return 0;

}


double doCH4bdconc (double deltat, double deltar, int i, int j,
                    double **CH4bdconc, double aertrm)
{
  double x;

  /* calculates the methane conc. at a given radius and time in a bead
     from diffusion (x), and aerobic utilization (aertrm)*/

  x = deltat*(DCH4*((CH4bdconc[i-1][j+1] - 2*CH4bdconc[i-1][j]+
                     CH4bdconc[i-1][j-1])*100/sqr(deltar))+
                       2*DCH4*100/(j*deltar)*
                         ((CH4bdconc[i-1][j]-CH4bdconc[i-1][j-1])/deltar))
                           + CH4bdconc[i-1][j];
  if ( x-aertrm >= 0 )    /* the 1 coefficient is from stoichiometry */
    return x-aertrm;
  else
    return 0;
}


int initconc (struct datastruct *dtptr, double *v)
{
  double **dmatrix (int rows, int cols);
  double *dvector(int size);
  int div;

  div=dtptr->divisions;
  dtptr->PCEattime = dvector(3);   /* assigns a 4 member array to PCEattime */
  dtptr->DCEattime = dvector(3);
  dtptr->PCEattime[0] = dtptr->concPCEin;  /* init. PCE at t=0 in reac. */
  dtptr->DCEattime[0] = 0;

  dtptr->PCEbdconc = dmatrix (3, div+1);  /* assigns a 3x(div+1) array to
                                             PCEbdconc */
  initbdconczero(dtptr->PCEbdconc, dtptr->divisions);

  dtptr->anbiomass = dmatrix (3, div+1);
  dtptr->aerbiomass = dmatrix (3, div+1);
```

```
    initbiomass(dtptr->aerbiomass, dtptr->anbiomass, div, dtptr->
                initaercelldensty, dtptr->initancelldensty, v);

    dtptr->subbdconc = dmatrix (3, div+1);
    initbdconczero(dtptr->subbdconc, div);

    dtptr->DCEbdconc = dmatrix (3, div+1);
    initbdconczero(dtptr->DCEbdconc, div);

    dtptr->O2bdconc = dmatrix (3, div+1);
    initbdconczero(dtptr->O2bdconc, div);

    dtptr->CH4bdconc = dmatrix (3, div+1);
    initbdconczero(dtptr->CH4bdconc, div);

    printf("\nBeginning calculations.\n");
    return 0;
  }

int reacconc(struct datastruct *dtptr, int i, double k)
{
  double rv, flow, deltat, deltar;
  int div;


  rv= dtptr->reacvol;
  flow= dtptr->flow;
  div= dtptr->divisions;
  deltat= dtptr->deltat;
  deltar= dtptr->deltar;

  /*  This function calculates the concentration of the given chemical
      in the reactor at a given time */

  dtptr->PCEattime[i]=((flow/rv*
                       (dtptr->concPCEin - dtptr->PCEattime[i-1])-k*DPCE*
                       ((dtptr->PCEbdconc[i-1][div]-
                         dtptr->PCEbdconc[i-1]
                         [div-1]) / deltar))*
                       deltat+dtptr->PCEattime[i-1]);

  dtptr->DCEattime[i]=((flow/rv*
                        (0 - dtptr->DCEattime[i-1])-k*DDCE*
                        ((dtptr->DCEbdconc[i-1][div]-
                          dtptr->DCEbdconc[i-1]
                          [div-1]) / deltar))*
                        deltat+dtptr->DCEattime[i-1]);

  return 0;
}

double dosubaerb (double deltat, double v, double aermmax, double subbdconc,
                  double O2bdconc, double aerbiomass)
{
  double subaert;

  /*  This function calculates a term that allows the aerobes to degrade
      the anaerobic substrate  */

  subaert= deltat/v * aerbiomass*aermmax/aeryield*
    (O2bdconc/(KO2+O2bdconc))*(subbdconc/(KSUBA+subbdconc));

  return subaert;
}
```



```
double doaerb (double deltat, double v, double aermmax, double CH4bdconc,
               double O2bdconc, double aerbiomass)
{
  double aert;

  /* This function calculates the aerobic term in the mass balance
     equation at a given time and radius (volume) */

  aert= deltat/v * aerbiomass*aermmax/aeryield*
    (CH4bdconc/(KCH4+CH4bdconc))*(O2bdconc/(KO2+O2bdconc));

  return aert;
}

double doanb (double deltat, double v, double anmmax, double subbdconc,
              double PCEbdconc, double anbiomass, double O2bdconc)
{
  double ant;

  /* This function calculates the anaerobic term in the mass balance
     equation at a given time and radius (volume) */

  ant= deltat/v *anbiomass*anmmax/anyield*
    (PCEbdconc/(KPCE+PCEbdconc))*(subbdconc/(KSUB+subbdconc))*(1/(1+O2bdconc/KI));

  return ant;
}

double doDCEaerb (double CH4bdconc, double DCEbdconc, double aerbiomass,
                  double O2bdconc, double deltat, double v)

{
  double DCEaertrm;

  if (CH4bdconc>DCEbdconc)  /* allows DCE degradation only if the methane
                               concentration is greater than the DCE conc. */
    DCEaertrm= deltat/v * aerbiomass*DCEmur/(96.9*CGDW)*
      (O2bdconc/(KO2+O2bdconc))*
        (DCEbdconc/(KDCE+DCEbdconc+(KDCE*CH4bdconc/KCH4)));
  else
    DCEaertrm= 0;

  return DCEaertrm;
}

int dozeroradius (struct datastruct *dtptr, int i, double v)
{
  double deltat, deltar, anb, aerb, subaerb, PCEx, subx, O2x, CH4x, DCEx, ans,
         DCEaertrm;

  deltat = dtptr->deltat;
  deltar = dtptr->deltar;

  /* This function calculates all the necessary parameters for the
     volume at zero radius in the bead (actually a sphere of radius
     0.5 deltar with origin at the center of each bead.  This function
     is needed because zero radius is a boundary condition and special
     equations apply that do not apply to the rest of the bead. */

  anb = doanb(deltat, v, dtptr->anmmax, dtptr->subbdconc[i-1][0],
              dtptr->PCEbdconc[i-1][0], dtptr->anbiomass[i-1][0],
              dtptr->O2bdconc[i-1][0]);
```

```
  aerb = doaerb(deltat, v, dtptr->aermmax, dtptr->CH4bdconc[i-1][0],
                dtptr->O2bdconc[i-1][0], dtptr->aerbiomass[i-1][0]);

  subaerb = dosubaerb(deltat, v, dtptr->aermmax, dtptr->subbdconc[i-1][0],
                      dtptr->O2bdconc[i-1][0], dtptr->aerbiomass[i-1][0]);

  DCEaertrm= doDCEaerb(dtptr->CH4bdconc[i-1][0], dtptr->
                       DCEbdconc[i-1][0], dtptr->aerbiomass
                       [i-1][0], dtptr->
O2bdconc[i-1][0], deltat, v);

  PCEx = deltat*(2*DPCE*(dtptr->PCEbdconc[i-1][1]-dtptr->PCEbdconc[i-1][0])*
                 100/sqr(deltar))+dtptr->PCEbdconc[i-1][0];

  if ( PCEx-2*anb >= 0 )
    dtptr->PCEbdconc[i][0] = PCEx-2*anb;
  else
    dtptr->PCEbdconc[i][0] = 0;



  subx = deltat*(2*DSUB*(dtptr->subbdconc[i-1][1]-dtptr->subbdconc[i-1][0])*
                 100/sqr(deltar))+dtptr->subbdconc[i-1][0];

  if ( subx-anb-subaerb >= 0 )
    dtptr->subbdconc[i][0] = subx-anb-subaerb;
  else
    dtptr->subbdconc[i][0] = 0;



  O2x = deltat*(2*DO2*(dtptr->O2bdconc[i-1][1]-dtptr->O2bdconc[i-1][0])*
                 100/sqr(deltar))+dtptr->O2bdconc[i-1][0];

  if ( O2x-(2*aerb)-(2*subaerb)-(2*DCEaertrm) >= 0 )
    dtptr->O2bdconc[i][0] = O2x-(2*aerb)-(2*subaerb)-(2*DCEaertrm);
  else
    dtptr->O2bdconc[i][0] = 0;



  CH4x = deltat*(2*DCH4*(dtptr->CH4bdconc[i-1][1]-dtptr->CH4bdconc[i-1][0])*
                 100/sqr(deltar))+dtptr->CH4bdconc[i-1][0];

  if ( CH4x-aerb >= 0 )
    dtptr->CH4bdconc[i][0] = CH4x-aerb;
  else
    dtptr->CH4bdconc[i][0] = 0;


  DCEx = deltat*(2*DDCE*(dtptr->DCEbdconc[i-1][1]-dtptr->DCEbdconc[i-1][0])*
                 100/sqr(deltar))+dtptr->DCEbdconc[i-1][0];

  ans = DCEx -DCEaertrm*(2*anb);

  if (ans >= 0)
    dtptr->DCEbdconc[i][0] = ans;
  else
    dtptr->DCEbdconc[i][0] = 0;


  dtptr->anbiomass[i][0] = dtptr->anmmax*(dtptr->subbdconc[i-1][0]/
                                          (KSUB+dtptr->subbdconc[i-1][0]))*
                                            (dtptr->PCEbdconc[i-1][0]/
                                                                    (KPCE+dtptr->PCEbdconc[i-1][0]))*
                                                                      dtptr->anbiomass[i-1][0]
                                                                        *deltat+dtptr->
                                                                          anbiomass[i-1][0];

  dtptr->aerbiomass[i][0] = (dtptr->aermmax*((dtptr->CH4bdconc[i-1][0]/
                                                                    (KCH4+dtptr->CH4bdconc[i-1][0]))*
                                  (dtptr->O2bdconc[i-1][0]/(KO2+dtptr->O2bdconc[i-1][0]))+
                                        (dtptr->subbdconc[i-1][0]/
                                                                    (KSUB+dtptr->subbdconc[i-1][0]))*
                                  (dtptr->O2bdconc[i-1][0]/(KO2+dtptr->O2bdconc[i-1][0])))*
                            dtptr->aerbiomass[i-1][0])*deltat + dtptr->aerbiomass[i-1][0];


  return 0;
}

int checkconsts(double deltat, double deltar)
{
  double PCEtest, PCEtest2, DCEtest, DCEtest2, CH4test, CH4test2,
         O2test, O2test2, subtest, subtest2;


  /*  This function checks to see if the constants in the diffusion
      parts of the mass balance equation are less that a certain value,
      as defined by <test> (defined at the top).  This is to ensure that
      certain conditions for the numerics are not violated, which would
      lead to a breakdown in the validity of the entire program.  */

  PCEtest = deltat*DPCE*100/sqr(deltar);
  PCEtest2 = deltat*2*DPCE*100/(deltar*deltar);
  if (PCEtest>test || PCEtest2>test)
    {
      printf("/nERROR:  PCE constant parameter out of range.\n");
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", PCEtest,
             PCEtest2, test);
      exit(-1);
    }

  DCEtest = deltat*DDCE*100/sqr(deltar);
  DCEtest2 = deltat*2*DDCE*100/(deltar*deltar);
  if (DCEtest>test || DCEtest2>test)
    {
      printf("\nERROR:  DCE constant parameter out of range.\n");
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", DCEtest,
             DCEtest2, test);
      exit(-1);
    }

  O2test = deltat*DO2*100/sqr(deltar);
  O2test2 = deltat*2*DO2*100/(deltar*deltar);
  if (O2test>test || O2test2>test)
    {
      printf("\nERROR:  O2 constant parameter out of range.\n");
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", O2test, O2test2,
             test);
      exit(-1);
    }

  CH4test = deltat*DCH4*100/sqr(deltar);
  CH4test2 = deltat*2*DCH4*100/(deltar*deltar);
  if (CH4test>test || CH4test2>test)
    {
      printf("\nERROR:  CH4 constant parameter out of range.\n");
```

```
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", CH4test,
            CH4test2, test);
      exit(-1);
    }

  subtest = deltat*DSUB*100/sqr(deltar);
  subtest2 = deltat*2*DSUB*100/(deltar*deltar);
  if (subtest>test || subtest2>test)
    {
      printf("\nERROR:  substrate  constant parameter out of range.\n");
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", subtest,
            subtest2, test);
      exit(-1);
    }
  return 0;
}

int dovartest (struct datastruct *dtptr, int i, int j, int static_y)
{
  double anPCEtest, ansubtest, aerCH4test, aerO2test, deltat, deltar,
         aersubO2test, aersubsubtest;

  /* This is similar to the checkconsts function above in that it checks
     certain expressions to ensure they are less that <test> defined at
     the top.  This function is called at every time step due to the fact
     that the expressions in question depend on time-varying concentrations,
     where those in checkconsts do not, and that function is called only
     once. */

  deltat = dtptr->deltat;
  deltar = dtptr->deltar;

  anPCEtest = deltat*dtptr->anbiomass[i-1][j]*1e6*dtptr->anmmax/
    (anyield*4/3*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->subbdconc[i-1][j]/(KSUB+dtptr->subbdconc[i-1][j]))/
        (KPCE+dtptr->PCEbdconc[i-1][j]))*(1/(1+dtptr->O2bdconc[i-1][j]));

  ansubtest = deltat*dtptr->anbiomass[i-1][j]*1e6*dtptr->anmmax/
    (anyield*4/3*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->PCEbdconc[i-1][j]/(KPCE+dtptr->PCEbdconc[i-1][j]))/
        (KSUB+dtptr->subbdconc[i-1][j]))*(1/(1+dtptr->O2bdconc[i-1][j]));

  if (anPCEtest > test || ansubtest > test)
    {
      printf("\nERROR:  anaerobic  parameter out of range.\n");
      printf("Indices - [%d][%d].\n", i-1, j);
      printf("PCEtest - %.2lf, subtest - %.2lf must be < %.2f.\n",
            anPCEtest, ansubtest, test);
      exit(-1);
    }

  aerCH4test = deltat*dtptr->aerbiomass[i-1][j]*1e6*dtptr->aermmax/
    (aeryield*4/3*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->O2bdconc[i-1][j]/(KO2+dtptr->O2bdconc[i-1][j]))/
        (KCH4+dtptr->CH4bdconc[i-1][j]);

  aerO2test = deltat*dtptr->aerbiomass[i-1][j]*1e6*dtptr->aermmax/
    (aeryield*4/3*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->CH4bdconc[i-1][j]/(KCH4+dtptr->CH4bdconc[i-1][j]))/
        (KO2+dtptr->O2bdconc[i-1][j]);

  if (aerCH4test > test || aerO2test > test)
    {
      printf("\nERROR:  aerobic  parameter out of range.\n");
      printf("Indices - [%d][%d].\n", i-1, j);
      printf("CH4test - %.2lf, O2test - %.2lf must be < %.2f.\n",
            aerCH4test, aerO2test, test);
      if (aerO2test>test)
        {
          printf("deltat= %.11f\n", deltat);
          printf("aerbiomass= %.3e\n", dtptr->aerbiomass[i-1][j]);
          printf("aermmax= %.3e\n", dtptr->aermmax);
          printf("aeryield= %.3e\n", aeryield);
          printf("v= %.3e\n",4*PI*
                (cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)));
          printf("CH4bdconc= %.3e\n", dtptr->CH4bdconc[i-1][j]);
          printf("KCH4= %.3e\n", KCH4);
          printf("O2bdconc= %.3e\n", dtptr->O2bdconc[i-1][j]);
          printf("KO2= %.3e\n", KO2);
        }
      exit(-1);
    }

  aersubO2test= deltat*dtptr->aerbiomass[i-1][j]*1e6*dtptr->aermmax/
    (aeryield*4/3*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->subbdconc[i-1][j]/(KSUBA+dtptr->subbdconc[i-1][j]))/
        (KO2+dtptr->O2bdconc[i-1][j]);

  aersubsubtest= deltat*dtptr->aerbiomass[i-1][j]*1e6*dtptr->aermmax/
    (aeryield*4/3*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->O2bdconc[i-1][j]/(KO2+dtptr->O2bdconc[i-1][j]))/
        (KSUBA+dtptr->subbdconc[i-1][j]);

  if (aersubO2test > test || aersubsubtest > test)
    {
      printf("\nERROR:  aerobic substrate parameter out of range.\n\n");
      printf("subO2test = %.3lf, subsubtest = %.3lf must be < %.3lf.\n",
            aersubO2test, aersubsubtest, test);
      printf("subbdconc = %.3e, O2bdconc = %.3e\n", dtptr->subbdconc[i-1][j],
            dtptr->O2bdconc[i-1][j]);
      printf("i = %d, j = %d\n", static_y, j);
      exit(-1);
    }
  return 0;

}


double *dov (double deltar, int div)
{
  int j;
  double *v, *dvector(int size);

  /* This function determines the volume of each anulus as defined by
     the radius and the number of divisions */

  v=dvector(div+1);

  for (j=0; j <= div; j++)
    {
      if (j==0)  v[j]= 4.0/3*PI*cub(deltar/2)/1e6;
      else
        v[j]= 4*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2))/1e6;

    }

  return v;
```

```
}

int baseline (struct datastruct *dtptr, int i)
{
  int j;

  /* This function resets the baseline at the end of each loop of three
     in chemfunctime.  It sets the concentrations, etc., at i=2 to those
     at i=0 for use during the next loop (starting at 1), so i-1=0. */

  for (j=dtptr->divisions; j>=0; --j)
    {
      dtptr->PCEbdconc[0][j] = dtptr->PCEbdconc[1][j];
      dtptr->DCEbdconc[0][j] = dtptr->DCEbdconc[1][j];
      dtptr->CH4bdconc[0][j] = dtptr->CH4bdconc[1][j];
      dtptr->O2bdconc[0][j] = dtptr->O2bdconc[1][j];
      dtptr->subbdconc[0][j] = dtptr->subbdconc[1][j];
      dtptr->aerbiomass[0][j] = dtptr->aerbiomass[1][j];
      dtptr->anbiomass[0][j] = dtptr->anbiomass[1][j];
    }
}

int chemfunctime (struct datastruct *dtptr)
{

   int i, j, c, wrtinterv, div, d, F;
   double k, aertrm, antrm, subaertrm, DCEaertrm, deltat, deltar, *v;
   static int static_y=0;

   /* This function calculates the concentration of the different chemicals
      of interest at each radius at every time. */

   wrtinterv=dtptr->writeinterval;
   div= dtptr->divisions;
   deltat= dtptr->deltat;
   deltar= dtptr->deltar;


   checkconst(dtptr->deltat, dtptr->deltar);

   k=(double) 4*PI*dtptr->numofbeads*sqr(dtptr->radius)/(10000*
                                                         dtptr->reacvol);

   v=dov(deltar, div);

   initconc(dtptr, v);


   for (c=1; c <= dtptr->time/2; c++)
     {
       for (i=1; i <= 2; i++)
         {
           ++static_y;

           dozeroradius(dtptr, i, v[0]);
           dtptr->PCEbdconc[i][div] = dtptr->PCEattime[i-1];
           dtptr->DCEbdconc[i][div] = dtptr->DCEattime[i-1];
           dtptr->subbdconc[i][div] = dtptr->subattime;
           dtptr->O2bdconc[i][div] = dtptr->O2attime;
           dtptr->CH4bdconc[i][div] = dtptr->CH4attime;

           j = div;
           dtptr->anbiomass[i][div] = dtptr->anmmax*
             (dtptr->subbdconc[i-1][j]/(KSUB+dtptr->subbdconc[i-1][j]))*
             (dtptr->PCEbdconc[i-1][j]/(KPCE+dtptr->PCEbdconc[i-1][j]))
           *dtptr->anbiomass[i-1][j]*deltat+dtptr->anbiomass[i-1][j];

dtptr->aerbiomass[i][div] = dtptr->aermmax*
  ((dtptr->CH4bdconc[i-1][j]/(KCH4+dtptr->CH4bdconc[i-1][j]))*
    (dtptr->O2bdconc[i-1][j]/(KO2+dtptr->O2bdconc[i-1][j]))*
   (dtptr->subbdconc[i-1][j]/(KSUB+dtptr->subbdconc[i-1][j]))*
    (dtptr->PCEbdconc[i-1][j]/(KPCE+dtptr->PCEbdconc[i-1][j])))
      *dtptr->aerbiomass[i-1][j]*deltat+
        dtptr->aerbiomass[i-1][j];

for (j= dtptr->divisions-1; j > 0; j--) /*calculates all concent.'s
                                          at each radius (j) at a
                                          given time (i)  */
  {
    aertrm= doaerb(deltat, v[j], dtptr->aermmax,
                   dtptr->CH4bdconc[i-1][j], dtptr->
                   O2bdconc[i-1][j], dtptr->aerbiomass[i-1][j]);

    antrm= doanb(deltat, v[j], dtptr->anmmax,
                 dtptr->subbdconc[i-1][j], dtptr->PCEbdconc[i-1][j]
                 ,dtptr->anbiomass[i-1][j],dtptr->O2bdconc[i-1][j]);

    subaertrm= dosubaerb(deltat, v[j], dtptr->aermmax,
                         dtptr->subbdconc[i-1][j], dtptr->
                         O2bdconc[i-1][j], dtptr->
                         aerbiomass[i-1][j]);

    DCEaertrm= doDCEaerb(dtptr->CH4bdconc[i-1][j], dtptr->
                         DCEbdconc[i-1][j], dtptr->aerbiomass
                         [i-1][j], dtptr->O2bdconc[i-1][j],
                         deltat, v[j]);

    dovartest(dtptr, i, j, static_y);

    dtptr->anbiomass[i][j] = dtptr->anmmax*
      (dtptr->subbdconc[i-1][j]/
       (KSUB+dtptr->subbdconc[i-1][j]))*
         (dtptr->PCEbdconc[i-1][j]/(KPCE+dtptr->PCEbdconc[i-1]
                                    [j]))*dtptr->anbiomass[i-1][j]
                                      *deltat + dtptr->
                                        anbiomass[i-1][j];

    dtptr->PCEbdconc[i][j]=doPCEbdconc(deltat, deltar, antrm,
                                       dtptr->PCEbdconc, i, j);

    dtptr->subbdconc[i][j]=dosubbdconc(deltat, deltar, antrm,
                                       dtptr->subbdconc, i, j,
                                       subaertrm);

    dtptr->DCEbdconc[i][j]=doDCEbdconc(deltat, deltar, i, j,
                                       dtptr->DCEbdconc, antrm,
                                       dtptr->aerbiomass,
                                       dtptr->CH4bdconc, v[j],
                                       dtptr->aermmax, dtptr->
                                       O2bdconc, DCEaertrm);


    dtptr->aerbiomass[i][j] = (dtptr->aermmax*
      (dtptr->CH4bdconc[i-1][j]/(KCH4+dtptr->CH4bdconc
                                 [i-1][j]))*
        (dtptr->O2bdconc[i-1][j]/
         (KO2+dtptr->O2bdconc[i-1][j])))
          *dtptr->aerbiomass[i-1][j]
```

```
                                                *deltat + dtptr->
                                                  aerbiomass[i-1][j];

                dtptr->O2bdconc[i][j]=doO2bdconc(deltat, deltar, i, j, dtptr->
                                                 O2bdconc, aertrm, subaertrm,
                                                 DCEaertrm);

                dtptr->CH4bdconc[i][j]=doCH4bdconc(deltat, deltar, i, j, dtptr->
                                                   CH4bdconc, aertrm);

              }                             /*  this closes for j  */
        /*  this calculates the chem.  conc. at a given time in the reactor  */
              reacconc (dtptr, i, k);

            if (i==2)
              {
                dtptr->PCEattime[0] = dtptr->PCEattime[i];
                dtptr->DCEattime[0] = dtptr->DCEattime[i];

                baseline(dtptr, i);

              }

            if ((int)(static_y)%(wrtinterv*10)==0)
              {
                d=i-1;
                writeoutput(dtptr, (int)(static_y/10), d);
                printf("Finished writing i = %d\n", (int) (static_y/10));
              }
          }                             /*  closes the for i  */
        }                               /*  closes the for c  */
      return 0;
    }
```

Appendix B
C Code for Scenario 2

```
#include <stdio.h>
#define PI 3.14159265359
#define DO2 1.5e-5                  /* The D definitions refer to the
                                       corresponding diffusivities */

#define DPHE 6.75e-6
#define DPCE 6.58e-6
#define DDCE 8.4e-6
#define KPCE 5.0e-7                 /* The K definitions refer to the
                                       corresponding Monod constants */

#define KPHE 1.59e-5
#define KPHEA 5e-4
#define KO2 3.13e-5
#define KDCE 1.03e-5
#define KI 1e-7                     /* the oxygen concentration above which
                                       anaerobes cease to function */

#define KIP 5.31e-3
#define KIPA 1.86e-2                /* anaerobic phenol inhib. satur. cnst*/
#define anyield 3e13                /* anaerobic cell yield in cells per
                                       mole phenol */
#define aeryield 4.89e13            /* aerobic cell yield in cells per
                                       mole phenol */
#define DCEmur 1.45e-5              /* DCE rx maximum utiliz. rate */
#define test 0.5                    /* the numerics test factor (<1) */
#define CGDW 1e12                   /* number of cells per gram dry weight */


struct datastruct
{
  int numofbeads, time, simtime, divisions;
  long writeinterval;
  double radius, reacvol, deltat,
         concPCEin, flow, deltar, anmmax, aermmax, DCEmmax,
         initaercelldensty, initancelldensty, O2attime, PHEattime;
  double **DCEbdconc, **aerbiomass, **O2bdconc,
         **PCEbdconc, **anbiomass, **PHEbdconc;
  double *PCEattime, *DCEattime;
  FILE *ofp, *ofp2, *ofp3;
} data;

float sqr (float x)
{
  return x*x;
}

float cub (float x)
{
  return x*x*x;
}

int inputdata (struct datastruct *dtptr)
{
  dtptr->radius = 1.5;              /* Average radius of beads in mm */

  dtptr->divisions = 10;            /* Numb. of radii used in numerics */

  dtptr->numofbeads = 7000;         /* Number of beads in reactor */

  dtptr->anmmax = 3.09e-6;          /* Anaer. doubling time in inv sec */

  dtptr->initancelldensty = 1e8;    /* In. anaer. cell dsty in cells/ml */

  dtptr->aermmax = 1.25e-4 ;        /* Aer. doubling time in inv. sec */

  dtptr->initaercelldensty = 1e8;   /* In. aer. cell dsty in cells/ml */

  dtptr->reacvol = 0.401;           /* Reactor fluid volume in liters */

  printf("Input the flow in l/s -> ");
  scanf("%lf", &dtptr->flow);       /* Flow in l/s */

  dtptr->deltat = 0.1;              /* Timestep in sec  */

  printf("Input simtime -> ");      /* Simulation time in sec */
  scanf("%ld", &dtptr->simtime);

  printf("Input the write interval (sec) -> ");
  scanf("%ld", &dtptr->writeinterval);  /* Interval (in number of sec) at
                                           which results should be written
                                           to a file. e.g.; every 1 hr
                                           would require input of 3600 (sec) */
  printf("Input the constant Oxygen concentration -> ");
  scanf("%lf", &dtptr->O2attime);   /* Oxyg. influent conc. in molar */

  printf("Input the constant phenol concentration -> ");
  scanf("%lf", &dtptr->PHEattime);  /* Phenol influent conc. in molar */

  dtptr->concPCEin = 6.03e-5;       /* PCE influent conc. in molar */

  return 0;
}

int main (void)
{
  int inputdata (struct datastruct *);
  int operate  (struct datastruct *);
  struct datastruct *dtptr;

  dtptr = &data;
  inputdata (dtptr);
  operate(dtptr);

  return 0;
}

int operate (struct datastruct *dtptr)
{
  int chemfunctime (struct datastruct *dtptr);

/*  calculate the delta r from the given radius and number of divisions  */

  dtptr->deltar = dtptr->radius/dtptr->divisions;

/*  convert the timesteps into integer values for use in array  */

   dtptr->time = (int) (dtptr->simtime/dtptr->deltat);

   chemfunctime(dtptr);         /* called to simulate conc. in reactor
                                   and beads over time */

  return 0;
}


int writeoutput (struct datastruct *dtptr, int wrttime, int i)
{
  int j;
  static int static x = 0;
```

```
  if (static_x == 0)
    {
      dtptr->ofp = fopen ("effluent", "w");
      fprintf(dtptr->ofp,"                BIOLOGICAL REACTOR SIMULATION RESULTS\n\n");
      fprintf(dtptr->ofp,"     Radius= %6.2f mm\n", dtptr->radius);
      fprintf(dtptr->ofp,"     Number of beads= %6d            Anaer. mmax= %.3e sec-1\n
",
              dtptr->numofbeads, dtptr->anmmax);
      fprintf(dtptr->ofp,"     Initial anaer. cell density= %.3e cells/ml\n",
              dtptr->initancelldensty);
      fprintf(dtptr->ofp,"     Aer. mmax= %.3e sec-1          In. aer. cell dnsty= %.3e c
ells/ml\n", dtptr->aermmax, dtptr->initaercelldensty);
      fprintf(dtptr->ofp,"     Reactor volume= %8.3f l          Flow= %.3e l/s\n",
              dtptr->reacvol, dtptr->flow);
      fprintf(dtptr->ofp,"     Retention time = %.3e days\n", dtptr->reacvol/dtptr->flow
/86400);
      fprintf(dtptr->ofp,"     Delta t= %5.3f sec                  Simulation time= %d se
c\n",
              dtptr->deltat, dtptr->simtime);
      fprintf(dtptr->ofp,"     Write interval= %6ld sec         Divisions= %d\n\n",
              dtptr->writeinterval, dtptr->divisions);
      fprintf(dtptr->ofp,"                          Concentrations (molar)\n");
      fprintf(dtptr->ofp,"                          -----------------------\n\n");
      fprintf(dtptr->ofp,"      Constant Oxygen= %.3e\n",dtptr->O2attime);
      fprintf(dtptr->ofp,"      PCE= %.3e                          Constant Phenol= %.3e\n\
n\n",
              dtptr->concPCEin, dtptr->PHEattime);
      fprintf (dtptr->ofp,"                   Effluent Concentrations (molar)\n");
      fprintf (dtptr->ofp,"                   ---------------------------------\n\n");
      fprintf (dtptr->ofp," Time          PCE          DCE          Oxygen  ");
      fprintf (dtptr->ofp,"     Phenol  \n");
      fprintf (dtptr->ofp," ------        -----        -----        -------- ");
      fprintf (dtptr->ofp,"    -------- \n");
      fclose (dtptr->ofp);
      dtptr->ofp = fopen("effluent", "a");

      dtptr->ofp2 = fopen("bdconc", "w");
      fprintf (dtptr->ofp2,
               "\n\n                              Bead Concentration\n");
      fprintf (dtptr->ofp2,
               "                              --------------------\n\n");
      fprintf (dtptr->ofp2," Radius          PCE          DCE          Oxygen  ");
      fprintf (dtptr->ofp2,"    Phenol  \n");
      fprintf (dtptr->ofp2," ------        -----        -----        -------- ");
      fprintf (dtptr->ofp2,"  -------- \n");
      fclose (dtptr->ofp2);
      dtptr->ofp2 = fopen ("bdconc", "a");

      dtptr->ofp3 = fopen("cellnum", "w");
      fprintf(dtptr->ofp3,"                   Cell Numbers at each radii\n");
      fprintf(dtptr->ofp3,"                   ---------------------------\n\n\n");
      fprintf(dtptr->ofp3,"Radius          Aerobic          Anaerobic\n");
      fprintf(dtptr->ofp3,"------          ---------        -----------\n\n");
      fclose(dtptr->ofp3);
      dtptr->ofp3 = fopen ("cellnum", "a");
    }

  fprintf(dtptr->ofp,"%6d      %9.3e    %9.3e    %9.3e    %9.3e\n",
          wrttime, dtptr->PCEattime[i], dtptr->DCEattime[i],
          dtptr->O2attime, dtptr->PHEattime);

  fprintf (dtptr->ofp2,"\n\nTime (sec) = %d\n", wrttime);

  for (j=0; j <= dtptr->divisions; j++)
    {
      fprintf(dtptr->ofp2,
              "%5d        %9.3e   %9.3e   %9.3e   %9.3e\n",
              j, dtptr->PCEbdconc[i][j], dtptr->DCEbdconc[i][j],
              dtptr->O2bdconc[i][j], dtptr->PHEbdconc[i][j]);
    }

  fprintf (dtptr->ofp3,"\n\nTime (sec) = %d\n", wrttime);

  for (j=0; j <= dtptr->divisions; j++)
    {
      fprintf(dtptr->ofp3,
              "%5d          %9.3e          %9.3e\n",
              j, dtptr->aerbiomass[i][j], dtptr->anbiomass[i][j]);
    }


  ++static x;
}


int initbdconczero (double **bdconc, int divisions)
{
    int j;

    for (j=0; j <= divisions; j++)
      {
        bdconc[0][j]=0;
        bdconc[1][j]=0;
      }                                  /* Initializes the bead concentration
                                            of the input chemical to zero at
                                            all radii*/
    return 0;
}


int initbiomass(double **aerbiomass, double **anbiomass, int divisions,
                double initaercelldensty, double initancelldensty,
                double *v)
{
  int j;

  for (j=0; j <= divisions; j++)
    {
      aerbiomass[0][j] = initaercelldensty*v[j]*1000; /* The *1000 converts
                                                        volume (calculated
                                                        in liters, to ml */
      anbiomass[0][j] = initancelldensty*v[j]*1000;
    }
}


double  doPCEbdconc (double deltat, double deltar, double antrm,
                     double **PCEbdconc, int i, int j)
{
  double x;

  x = deltat*(DPCE*((PCEbdconc[i-1][j+1] - 2*PCEbdconc[i-1][j]+
                     PCEbdconc[i-1][j-1])*100/sqr(deltar)) +
                       2*DPCE*100/(j*deltar)*
                         ((PCEbdconc[i-1][j]-PCEbdconc[i-1][j-1])/deltar))
                           + PCEbdconc[i-1][j];
  if (x-(7*antrm) >= 0)   /* This expression ensures that the returned value
                             of conc. is not <0, which is a meaningless result */
```

```
    return x-(7*antrm); /* the 7 is from stoichiometry */
  else
    return 0;

}


double doPHEbdconc(double deltat, double deltar, double antrm,
                   double **PHEbdconc, int i, int j, double aertrm)
{
  double x;

  x = deltat*(DPHE*((PHEbdconc[i-1][j+1] - 2*PHEbdconc[i-1][j]+
                     PHEbdconc[i-1][j-1])*100/sqr(deltar))+
                       2*DPHE*100/(j*deltar)*
                         ((PHEbdconc[i-1][j]-PHEbdconc[i-1][j-1])/deltar))+
                           PHEbdconc[i-1][j];
  if ( x-antrm-aertrm >= 0 ) /* the 1 coef's are from stoichiometry */
    return x-antrm-aertrm;
  else
    return 0;
}


double doDCEbdconc(double deltat, double deltar, int i, int j, double
                   **DCEbdconc, double antrm, double **aerbiomass, double
                   **PHEbdconc, double v, double aermmax, double **O2bdconc,
                   double DCEaertrm)

{
  double x, ret;

  x = deltat*(DDCE*((DCEbdconc[i-1][j+1] - 2*DCEbdconc[i-1][j]+
                     DCEbdconc[i-1][j-1])*100/sqr(deltar))+
                        2*DDCE*100/(j*deltar)*
                          ((DCEbdconc[i-1][j]-DCEbdconc[i-1][j-1])/deltar))+
                            DCEbdconc[i-1][j];



  if (PHEbdconc[i-1][j] > 0)
    ret = x -DCEaertrm +(7*antrm);  /* the 1 and 7 are from stoichiometry */
  else
    ret = x + (7*antrm);

  if (ret >= 0)
    return ret;
  else
    return 0;
}


double doO2bdconc (double deltat, double deltar, int i, int j, double
                   **O2bdconc, double aertrm, double DCEaertrm, int static_y,
                   struct datastruct *dtptr)
{
  double x, me;

  x = deltat*(DO2*((O2bdconc[i-1][j+1] - 2*O2bdconc[i-1][j]+
                    O2bdconc[i-1][j-1])*100/sqr(deltar))+
                      2*DO2*100/(j*deltar)*
                        ((O2bdconc[i-1][j]-O2bdconc[i-1][j-1])/deltar))
                          + O2bdconc[i-1][j];
```

```
  if ( x-(7*aertrm)-(2*DCEaertrm) >= 0 )  /* the 2 and 7 are from stoichiom. */
    return x-(7*aertrm)-(2*DCEaertrm);
  else
    return 0;

}

int initconc (struct datastruct *dtptr, double *v)

{
   double **dmatrix (int rows, int cols);
   double *dvector(int size);
   int div;

   div=dtptr->divisions;
   dtptr->PCEattime = dvector(3);   /* assigns a 4 member array to PCEattime */
   dtptr->DCEattime = dvector(3);
   dtptr->PCEattime[0] = dtptr->concPCEin;    /* init. PCE at t=0 in reac.*/
   dtptr->DCEattime[0] = 0;

   dtptr->PCEbdconc = dmatrix (3, div+1);  /* assigns a 3x(div+1) array to
                                              PCEbdconc */
   initbdconczero(dtptr->PCEbdconc, dtptr->divisions);

   dtptr->anbiomass = dmatrix (3, div+1);
   dtptr->aerbiomass = dmatrix (3, div+1);
   initbiomass(dtptr->aerbiomass, dtptr->anbiomass, div, dtptr->
               initaercelldensty, dtptr->initancelldensty, v);

   dtptr->PHEbdconc = dmatrix (3, div+1);
   initbdconczero(dtptr->PHEbdconc, div);

   dtptr->DCEbdconc = dmatrix (3, div+1);
   initbdconczero(dtptr->DCEbdconc, div);

   dtptr->O2bdconc = dmatrix (3, div+1);
   initbdconczero(dtptr->O2bdconc, div);

   printf("\nMemory allocation complete.  Beginning calculations.\n");
   return 0;
 }

int reacconc(struct datastruct *dtptr, int i, double k)
{
  double rv, flow, deltat, deltar;
  int div;

  rv= dtptr->reacvol;
  flow= dtptr->flow;
  div= dtptr->divisions;
  deltat= dtptr->deltat;
  deltar= dtptr->deltar;

  /*  This function calculates the concentration of the given chemical
      in the reactor at a given time */

  dtptr->PCEattime[i]=((flow/rv*
                        (dtptr->concPCEin - dtptr->PCEattime[i-1])-k*DPCE*
                        ((dtptr->PCEbdconc[i-1][div]-
                          dtptr->PCEbdconc[i-1]
                          [div-1]) / deltar))*
                       deltat+dtptr->PCEattime[i-1]);
```

```
  dtptr->DCEattime[i]=((flow/rv*
                       (0 - dtptr->DCEattime[i-1])-k*DDCE*
                       ((dtptr->DCEbdconc[i-1][div]-
                         dtptr->DCEbdconc[i-1]
                         [div-1]) / deltar))*
                      deltat+dtptr->DCEattime[i-1]);

  return 0;
}

double doaerb (double deltat, double v, double aermmax, double PHEbdconc,
               double O2bdconc, double aerbiomass)
{
  double aert;

  /* This function calculates the aerobic term in the mass balance
     equation at a given time and radius (volume) */

  aert= deltat/v * aerbiomass*aermmax/aeryield*
    (PHEbdconc/((KPHE+PHEbdconc)*(1+PHEbdconc/KIP))*(O2bdconc/(KO2+O2bdconc)));

  return aert;
}

double doanb (double deltat, double v, double anmmax, double PHEbdconc,
              double PCEbdconc, double anbiomass, double O2bdconc)
{
  double ant;

  /* This function calculates the anaerobic term in the mass balance
     equation at a given time and radius (volume) */


  ant= deltat/v *anbiomass*anmmax/anyield*
    (PCEbdconc/(KPCE+PCEbdconc))*(PHEbdconc/((KPHEA+PHEbdconc)*
                                             (1+PHEbdconc/KIPA)))*
                                               (1/(1+O2bdconc/KI));

  return ant;
}

double doDCEaerb (double PHEbdconc, double DCEbdconc, double aerbiomass,
                  double O2bdconc, double deltat, double v)

{
  double DCEaertrm;

  if (PHEbdconc>DCEbdconc)
    DCEaertrm= deltat/v * aerbiomass*DCEmur/(96.9*CGDW)*
      (O2bdconc/(KO2+O2bdconc))*
        (DCEbdconc/((KDCE+DCEbdconc+(KDCE*PHEbdconc/KPHE))*(1+PHEbdconc/KIP)));
  else
    DCEaertrm= 0;

  return DCEaertrm;
}

int dozeroradius (struct datastruct *dtptr, int i, double v)
{
  double deltat, deltar, anb, aerb, DCEaertrm, PCEx, PHEx, O2x, DCEx, ans;

  deltat = dtptr->deltat;
  deltar = dtptr->deltar;
```

```
/* This function calculates all the necessary parameters for the
   volume at zero radius in the bead (actually a sphere of radius
   0.5 deltar with origin at the center of each bead.  This function
   is needed because zero radius is a boundary condition and special
   equations apply that do not apply to the rest of the bead. */

anb = doanb(deltat, v, dtptr->anmmax, dtptr->PHEbdconc[1-1][0],
            dtptr->PCEbdconc[1-1][0], dtptr->anbiomass[1-1][0],
            dtptr->O2bdconc[1-1][0]);

aerb = doaerb(deltat, v, dtptr->aermmax, dtptr->PHEbdconc[1-1][0],
              dtptr->O2bdconc[1-1][0], dtptr->aerbiomass[1-1][0]);

DCEaertrm= doDCEaerb(dtptr->PHEbdconc[1-1][0], dtptr->
                                    DCEbdconc[1-1][0], dtptr->aerbiomass
                                    [1-1][0], dtptr->O2bdconc[1-1][0],
                                    deltat, v);

PCEx = deltat*(2*DPCE*(dtptr->PCEbdconc[1-1][1]-dtptr->PCEbdconc[1-1][0])*
               100/sqr(deltar))+dtptr->PCEbdconc[1-1][0];

if ( PCEx-(7*anb) >= 0 )
  dtptr->PCEbdconc[1][0] = PCEx-(7*anb);
else
  dtptr->PCEbdconc[1][0] = 0;


PHEx = deltat*(2*DPHE*(dtptr->PHEbdconc[1-1][1]-dtptr->PHEbdconc[1-1][0])*
               100/sqr(deltar))+dtptr->PHEbdconc[1-1][0];

if ( PHEx-anb-aerb >= 0 )
  dtptr->PHEbdconc[1][0] = PHEx-anb-aerb;
else
  dtptr->PHEbdconc[1][0] = 0;


O2x = deltat*(2*DO2*(dtptr->O2bdconc[1-1][1]-dtptr->O2bdconc[1-1][0])*
               100/sqr(deltar))+dtptr->O2bdconc[1-1][0];

if ( O2x-(7*aerb)-(2*DCEaertrm) >= 0 )
  dtptr->O2bdconc[1][0] = O2x-(7*aerb)-(2*DCEaertrm);
else
  dtptr->O2bdconc[1][0] = 0;


DCEx = deltat*(2*DDCE*(dtptr->DCEbdconc[1-1][1]-dtptr->DCEbdconc[1-1][0])*
               100/sqr(deltar))+dtptr->DCEbdconc[1-1][0];

ans = DCEx - DCEaertrm +(7*anb);

if (ans >= 0)
  dtptr->DCEbdconc[1][0] = ans;
else
  dtptr->DCEbdconc[1][0] = 0;


dtptr->anbiomass[1][0] = dtptr->anmmax*(dtptr->PHEbdconc[1-1][0]/
                                        (KPHEA+dtptr->PHEbdconc[1-1][0]))*
                                          (dtptr->PCEbdconc[1-1][0]/
                                            (KPCE+dtptr->PCEbdconc[1-1][0]))*
                                              dtptr->anbiomass[1-1][0]
```

```
                                                *deltat*dtptr->
                                                  anbiomass[i-1][0];

  dtptr->aerbiomass[i][0] = (dtptr->aermmax*(dtptr->PHEbdconc[i-1][0]/
                                                (KPHE+dtptr->PHEbdconc[i-1][0]))*
                    (dtptr->O2bdconc[i-1][0]/(KO2+dtptr->O2bdconc[i-1][0]))*
                dtptr->aerbiomass[i-1][0])*deltat + dtptr->aerbiomass[i-1][0];


  return 0;
}

int checkconsts(double deltat, double deltar)
{
  double PCEtest, PCEtest2, DCEtest, DCEtest2, O2test, O2test2,
         PHEtest, PHEtest2;


  /*  This function checks to see if the constants in the diffusion
      parts of the mass balance equation are less that a certain value,
      as defined by <test> (defined at the top.  This is to ensure that
      certain conditions for the numerics are not violated, which would
      lead to a breakdown in the validity of the entire program.  */

  PCEtest = deltat*DPCE*100/sqr(deltar);
  PCEtest2 = deltat*2*DPCE*100/(deltar*deltar);
  if (PCEtest>test || PCEtest2>test)
    {
      printf("/nERROR:  PCE constant parameter out of range.\n");
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", PCEtest,
             PCEtest2, test);
      exit(-1);
    }

  DCEtest = deltat*DDCE*100/sqr(deltar);
  DCEtest2 = deltat*2*DDCE*100/(deltar*deltar);
  if (DCEtest>test || DCEtest2>test)
    {
      printf("\nERROR:  DCE constant parameter out of range.\n");
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", DCEtest,
             DCEtest2, test);
      exit(-1);
    }

  O2test = deltat*DO2*100/sqr(deltar);
  O2test2 = deltat*2*DO2*100/(deltar*deltar);
  if (O2test>test || O2test2>test)
    {
      printf("\nERROR:  O2 constant parameter out of range.\n");
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", O2test, O2test2,
             test);
      exit(-1);
    }

  PHEtest = deltat*DPHE*100/sqr(deltar);
  PHEtest2 = deltat*2*DPHE*100/(deltar*deltar);
  if (PHEtest>test || PHEtest2>test)
    {
      printf("\nERROR:  phenol  constant parameter out of range.\n");
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", PHEtest,
             PHEtest2, test);
      exit(-1);
    }
  return 0;
}

int dovartest (struct datastruct *dtptr, int i, int j, int static y)
{
  double anPCEtest, anPHEtest, aerPHEtest, aerO2test, deltat, deltar;

  /* This is similar to the checkconsts function above in that it checks
     certain expressions to ensure they are less that <test> defined at
     the top.  This function is called at every time step due to the fact
     that the expressions in question depend on time-varying concentrations,
     where those in checkconsts do not, and that function is called only
     once. */

  deltat = dtptr->deltat;
  deltar = dtptr->deltar;

  anPCEtest = deltat*dtptr->anbiomass[i-1][j]*1e6*dtptr->anmmax/
    (anyield*4/3*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->PHEbdconc[i-1][j]/((KPHEA+dtptr->PHEbdconc[i-1][j])*
                                    (1+dtptr->PHEbdconc[i-1][j]/KIPA)))/
        (KPCE+dtptr->PCEbdconc[i-1][j])*(1/(1+dtptr->O2bdconc[i-1][j]/KI));

  anPHEtest = deltat*dtptr->anbiomass[i-1][j]*1e6*dtptr->anmmax/
    (anyield*4/3*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->PCEbdconc[i-1][j]/(KPCE+dtptr->PCEbdconc[i-1][j]))/
        ((KPHEA+dtptr->PHEbdconc[i-1][j])*(1+dtptr->PHEbdconc[i-1][j]/KIPA))*
          (1/(1+dtptr->O2bdconc[i-1][j]/KI));

  if (anPCEtest > test || anPHEtest > test)
    {
      printf("\nERROR:  anaerobic parameter out of range.\n");
      printf("Indices - [%d][%d].\n", i-1, j);
      printf("PCEtest - %.2lf, PHEtest - %.2lf must be < %.2f.\n",
             anPCEtest, anPHEtest, test);
      exit(-1);
    }

  aerPHEtest = deltat*dtptr->aerbiomass[i-1][j]*1e6*dtptr->aermmax/
    (aeryield*4/3*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->O2bdconc[i-1][j]/(KO2+dtptr->O2bdconc[i-1][j]))/
        (KPHE+dtptr->PHEbdconc[i-1][j]);

  aerO2test = deltat*dtptr->aerbiomass[i-1][j]*1e6*dtptr->aermmax/
    (aeryield*4/3*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->PHEbdconc[i-1][j]/(KPHE+dtptr->PHEbdconc[i-1][j]))/
        (KO2+dtptr->O2bdconc[i-1][j]);

  if (aerPHEtest > test || aerO2test > test)
    {
      printf("\nERROR:  aerobic parameter out of range.\n");
      printf("Indices - [%d][%d].\n", i-1, j);
      printf("PHEtest - %.3lf, O2test - %.3lf must be < %.3f.\n",
             aerPHEtest, aerO2test, test);
      if (aerO2test>test)
        {
          printf("deltat= %.11f\n", deltat);
          printf("aerbiomass= %.3e\n", dtptr->aerbiomass[i-1][j]);
          printf("aermmax= %.3e\n", dtptr->aermmax);
          printf("aeryield= %.3e\n", aeryield);
          printf("v= %.3e\n",4*PI*
                 (cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)));
          printf("PHEbdconc= %.3e\n", dtptr->PHEbdconc[i-1][j]);
          printf("KPHE= %.3e\n", KPHE);
          printf("O2bdconc= %.3e\n", dtptr->O2bdconc[i-1][j]);
```

```
          printf("KO2= %.3e\n", KO2);
        }
      exit(-1);
    }

  return 0;


}


double *dov (double deltar, int div)
{
  int j;
  double *v, *dvector(int size);

  /* This function determines the volume of each anulus as defined by
     the radius and the number of divisions */

  v=dvector(div+1);

  for (j=0; j <= div; j++)
    {
      if (j==0)  v[j]= 4.0/3*PI*cub(deltar)/1e6;
      else
        v[j]= 4.0/3*PI*(cub(j*deltar)-cub(j*deltar-deltar))/1e6;

    }

  return v;
}

int baseline (struct datastruct *dtptr, int i)
{
  int j;

  /* This function resets the baseline at the end of each loop of three
     in chemfunctime.  It sets the concentrations, etc., at i=2 to those
     at i=0 for use during the next loop (starting at 1), so i-1=0. */

  for (j=dtptr->divisions; j>=0; --j)
    {
      dtptr->PCEbdconc[0][j] = dtptr->PCEbdconc[1][j];
      dtptr->DCEbdconc[0][j] = dtptr->DCEbdconc[1][j];
      dtptr->O2bdconc[0][j] = dtptr->O2bdconc[1][j];
      dtptr->PHEbdconc[0][j] = dtptr->PHEbdconc[1][j];
      dtptr->aerblomass[0][j] = dtptr->aerblomass[1][j];
      dtptr->anblomass[0][j] = dtptr->anblomass[1][j];
    }
}

int chemfunctime (struct datastruct *dtptr)
{

   int i, j, c, wrtinterv, div, d, F, s;
   double k, aertrm, antrm, DCEaertrm, deltat, deltar, *v;
   static int static_y=0;

   /* This function calculates the concentration of the different chemicals
      of interest at each radius at every time. */

   wrtinterv=dtptr->writeinterval;
   div= dtptr->divisions;
   deltat= dtptr->deltat;
   deltar= dtptr->deltar;


   checkconsts(dtptr->deltat, dtptr->deltar);

   k=(double) 4*PI*dtptr->numofbeads*sqr(dtptr->radius)/(10000*
                                                        dtptr->reacvol);


   v=dov(deltar, div);

   initconc(dtptr, v);

   printf("writeinterval = %ld\n", dtptr->writeinterval);
   printf("flow = %.3e\n", dtptr->flow);
   printf("simtime = %d\n", dtptr->simtime);
   printf("PHE conc. = %.3e\n", dtptr->PHEattime);

   for (c=1; c <= dtptr->time/2; c++)
     {
       for (i=1; i <= 2; i++)
         {
           /*static y;

           dozeroradius(dtptr, 1, v[0]);
           dtptr->PCEbdconc[i][div] = dtptr->PCEattime[i-1];
           dtptr->DCEbdconc[i][div] = dtptr->DCEattime[i-1];
           dtptr->PHEbdconc[i][div] = dtptr->PHEattime;
           dtptr->O2bdconc[i][div] = dtptr->O2attime;

           j = div;
           dtptr->anblomass[i][div] = dtptr->anmmax*
             (dtptr->PHEbdconc[i-1][j]/(KPHE+dtptr->PHEbdconc[i-1][j]))*
               (dtptr->PCEbdconc[i-1][j]/(KPCE+dtptr->PCEbdconc[i-1][j]))
              *dtptr->anblomass[i-1][j]*deltat+dtptr->anblomass[i-1][j];

           dtptr->aerblomass[i][div] = dtptr->aermmax*
             (dtptr->PHEbdconc[i-1][j]/(KPHE+dtptr->PHEbdconc[i-1][j]))*
               (dtptr->O2bdconc[i-1][j]/(KO2+dtptr->O2bdconc[i-1][j]))
                 *dtptr->aerblomass[i-1][j]*deltat+
                   dtptr->aerblomass[i-1][j];

           for (j= dtptr->divisions-1; j > 0; j--) /*calculates all concent.'s
                                                     at each radius (j) at a
                                                     given time (i)  */
             {
               aertrm= doaerb(deltat, v[j], dtptr->aermmax,
                              dtptr->PHEbdconc[i-1][j], dtptr->
                              O2bdconc[i-1][j], dtptr->aerblomass[i-1][j]);

               if (static_y==1198 && j==3)
                 {
                   printf("aertrm = %.1e\n", aertrm);
                   printf("v[j] = %.3e\n", v[j]);
                   printf("aermmax = %.3e\n", dtptr->aermmax);
                   printf("PHEbdconc = %.3e\n", dtptr->PHEbdconc[i-1][j]);
                   printf("O2bdconc = %.3e\n", dtptr->O2bdconc[i-1][j]);
                   printf("aerblomass = %.3e\n", dtptr->aerblomass[i-1][j]);
                 }

               antrm= doanb(deltat, v[j], dtptr->anmmax,
                            dtptr->PHEbdconc[i-1][j], dtptr->PCEbdconc[i-1][j]
                            ,dtptr->anblomass[i-1][j],dtptr->O2bdconc[i-1][j]);

               DCEaertrm= doDCEaerb(dtptr->PHEbdconc[i-1][j], dtptr->
```

```
                        DCEbdconc[i-1][j], dtptr->aerbiomass
                        [i-1][j], dtptr->O2bdconc[i-1][j],
                        deltat, v[j]);

        dovartest(dtptr, i, j, static_y);

        dtptr->anbiomass[i][j] = dtptr->anmmax*
          (dtptr->PHEbdconc[i-1][j]/
           (KPHE+dtptr->PHEbdconc[i-1][j]))*
              (dtptr->PCEbdconc[i-1][j]/(KPCE+dtptr->PCEbdconc[i-1]
                                         [j]))*dtptr->anbiomass[i-1][j]
                                           *deltat + dtptr->
                                             anbiomass[i-1][j];

        dtptr->PCEbdconc[i][j]=doPCEbdconc(deltat, deltar, antrm,
                                           dtptr->PCEbdconc, i, j);

        dtptr->PHEbdconc[i][j]=doPHEbdconc(deltat, deltar, antrm,
                                           dtptr->PHEbdconc, i, j,
                                           aertrm);

        dtptr->DCEbdconc[i][j]=doDCEbdconc(deltat, deltar, i, j,
                                           dtptr->DCEbdconc, antrm,
                                           dtptr->aerbiomass,
                                           dtptr->PHEbdconc, v[j],
                                           dtptr->aermmax, dtptr->
                                           O2bdconc, DCEaertrm);


        dtptr->aerbiomass[i][j] = (dtptr->aermmax*
          (dtptr->PHEbdconc[i-1][j]/(KPHE+dtptr->PHEbdconc
                                     [i-1][j]))*
                                       (dtptr->O2bdconc[i-1][j]/
                                        (KO2+dtptr->O2bdconc[i-1][j]))
                                         *dtptr->aerbiomass[i-1][j])
                                           *deltat + dtptr->
                                             aerbiomass[i-1][j];

        dtptr->O2bdconc[i][j]=doO2bdconc(deltat, deltar, i, j, dtptr->
                                         O2bdconc, aertrm, DCEaertrm,
                                         static_y, dtptr);


      }                          /*  this closes for }  */
/*  this calculates the chem.  conc. at a given time in the reactor  */
        reacconc (dtptr, i, k);

      if (i==2)
        {
          dtptr->PCEattime[0] = dtptr->PCEattime[1];
          dtptr->DCEattime[0] = dtptr->DCEattime[1];

          baseline(dtptr, i);

        }


      if ((int)(static_y)%(wrtinterv*10)==0)
        {
          d= i-1;
          writeoutput(dtptr, (int)(static_y/10), d);
          printf("Finished writing i = %d\n", (int) (static_y/10));
        }
    }                            /*  closes the for i  */
  }                              /*  closes the for c  */
  return 0;
}
```

Appendix C
C Code for Scenario 3

```
#include <stdio.h>
#define PI 3.14159265359
#define DO2 1.5e-5                  /* The D definitions refer to the
                                       corresponding diffusivities */

#define DGLU 5.4e-6
#define DCNP 5.5e-6
#define DCAP 5.7e-6
#define KCNP 2.5e-5                 /* The K definitions refer to the
                                       corresponding Monod constants */

#define KGLU 1.33e-4
#define KGLUA 1.33e-4               /* anaerobic glucose saturation constant*/
#define KO2 3.4e-4
#define KCAP 2e-4
#define KI 8e-6                     /* the oxygen concentration above which
                                       anaerobes cease to function */
#define anyield 2.1e14              /* anaerobic cell yield in cells per
                                       mole glucose */
#define aeryield 2.1e14             /* aerobic cell yield in cells per
                                       mole glucose */

#define CAPyield 4.89e13
#define test 0.5                    /* the numerics test factor (<1) */


struct datastruct
{
  int numofbeads, time, simtime, divisions;
  long writeinterval;
  double radius, reacvol, deltat,
         initCNP, initGLU, deltar, anmmax, aermmax, CAPmmax,
         initaercelldensty, initancelldensty, O2attime;
  double **CAPbdconc, **aerbiomass, **O2bdconc,
         **CNPbdconc, **anbiomass, **GLUbdconc;
  double *CNPattime, *CAPattime, *GLUattime;
  FILE *ofp, *ofp2, *ofp3;
} data;

float sqr (float x)
{
  return x*x;
}

float cub (float x)
{
  return x*x*x;
}

int inputdata (struct datastruct *dtptr)
{
  dtptr->radius = 1.5;                  /* Average radius of beads in mm */

  dtptr->divisions = 10;                /* Numb. of radii used in numerics */

  printf("Input the number of beads -> ");
  scanf("%d", &dtptr->numofbeads);          /* Number of beads in reactor */

  dtptr->anmmax = 2.14e-5;              /* Anaer. doubling time in inv sec */

  dtptr->initancelldensty = 2e8;        /* In. anaer. cell dsty in cells/ml */

  dtptr->aermmax = 7e-5;                /* Aer. doubling time in inv. sec */

  dtptr->CAPmmax = 9.6e-6;             /* Max grth rate for CAP-O2 rx */

  dtptr->initaercelldensty = 5e8;       /* In. aer. cell dsty in cells/ml */

  dtptr->reacvol = 0.300-4.0/3*PI*cub(dtptr->radius/10)*dtptr->numofbeads/1000;
                    /* Reactor fluid volume in liters */

  dtptr->deltat = 0.1;                  /* Timestep in sec  */

  printf("Input simtime -> ");          /* Simulation time in sec */
  scanf("%ld", &dtptr->simtime);

  printf("Input the write interval (sec) -> ");
  scanf("%ld", &dtptr->writeinterval);  /* Interval (in number of sec) at
                                           which results should be written
                                           to a file. e.g.; every 1 hr
                                           would require input of 3600 (sec) */
  printf("Input the constant Oxygen concentration -> ");
  scanf("%lf", &dtptr->O2attime);       /* Oxyg. influent conc. in molar */

  dtptr->initGLU = 7.5e-3/dtptr->reacvol;    /* Glucose init. conc. in molar */

  dtptr->initCNP = 7.95e-5/dtptr->reacvol;     /* CNP init. conc. in molar */

  return 0;
}

int main (void)
{
  int inputdata (struct datastruct *);
  int operate  (struct datastruct *);
  struct datastruct *dtptr;

  dtptr = &data;
  inputdata (dtptr);
  operate(dtptr);

  return 0;
}

int operate (struct datastruct *dtptr)
{
  int chemfunctime (struct datastruct *dtptr);

/*  calculate the delta r from the given radius and number of divisions  */

  dtptr->deltar = dtptr->radius/dtptr->divisions;

/*  convert the timesteps into integer values for use in array  */

   dtptr->time   (int) (dtptr->simtime/dtptr->deltat);

   chemfunctime(dtptr);         /* called to simulate conc. in reactor
                                   and beads over time */

  return 0;
}


int writeoutput (struct datastruct *dtptr, int wrttime, int i)
{
  int j;
  static int static_x = 0;

  if (static_x == 0)
    {
      dtptr->ofp = fopen ("reacconc", "w");
```

```
      fprintf(dtptr->ofp,"                BIOLOGICAL REACTOR SIMULATION RESULTS\n\n");
      fprintf(dtptr->ofp,"     Radius= %6.2f mm\n", dtptr->radius);
      fprintf(dtptr->ofp,"     Number of beads= %6d            Anaer. mmax= %.3e sec-1\n
",
              dtptr->numofbeads, dtptr->anmmax);
      fprintf(dtptr->ofp,"     Initial anaer. cell density= %.3e cells/ml\n",
              dtptr->initancelldensty);
      fprintf(dtptr->ofp,"     Aer. mmax= %.3e sec-1         In. aer. cell dnsty= %.3e c
ells/ml\n", dtptr->aermmax, dtptr->initaercelldensty);
      fprintf(dtptr->ofp,"     Reactor volume= %8.3f l\n",
              dtptr->reacvol);
      fprintf(dtptr->ofp,"     Delta t= %5.3f sec                Simulation time= %d se
c\n",
              dtptr->deltat, dtptr->simtime);
      fprintf(dtptr->ofp,"     Write interval= %6ld sec         Divisions= %d\n\n",
              dtptr->writeinterval, dtptr->divisions);
      fprintf(dtptr->ofp,"                          Concentrations (molar)\n");
      fprintf(dtptr->ofp,"                         ------------------------\n\n");
      fprintf(dtptr->ofp,"     Constant Oxygen= %.3e\n",dtptr->O2attime);
      fprintf(dtptr->ofp,"     Initial CNP= %.3e                Initial Glucose= %.3e\n
\n\n",
              dtptr->initCNP, dtptr->initGLU);
      fprintf (dtptr->ofp,"                  Effluent Concentrations (molar)\n");
      fprintf (dtptr->ofp,"                 ---------------------------------\n\n");
      fprintf (dtptr->ofp," Time         CNP          CAP          Oxygen  ");
      fprintf (dtptr->ofp,"     Glucose  \n");
      fprintf (dtptr->ofp," ------       -----        -----        -------- ");
      fprintf (dtptr->ofp,"     -------- \n");
      fclose (dtptr->ofp);
      dtptr->ofp = fopen("reacconc", "a");

      dtptr->ofp2 = fopen("bdconc", "w");
      fprintf (dtptr->ofp2,
               "\n\n                              Bead Concentration\n");
      fprintf (dtptr->ofp2,
               "                         --------------------\n\n");
      fprintf (dtptr->ofp2," Radius         CNP         CAP         Oxygen  ");
      fprintf (dtptr->ofp2,"    Glucose  \n");
      fprintf (dtptr->ofp2," ------        -----       -----       -------- ");
      fprintf (dtptr->ofp2,"  -------- \n");
      fclose (dtptr->ofp2);
      dtptr->ofp2 = fopen ("bdconc", "a");

      dtptr->ofp3 = fopen("cellnum", "w");
      fprintf(dtptr->ofp3,"                  Cell Numbers at each radii\n");
      fprintf(dtptr->ofp3,"                 ----------------------------\n\n\n");
      fprintf(dtptr->ofp3,"Radius          Aerobic          Anaerobic\n");
      fprintf(dtptr->ofp3,"------         ---------         -----------\n\n");
      fclose(dtptr->ofp3);
      dtptr->ofp3 = fopen ("cellnum", "a");

    }

  fprintf(dtptr->ofp,"%6d      %9.3e    %9.3e    %9.3e    %9.3e\n",
          wrttime, dtptr->CNPattime[i], dtptr->CAPattime[i],
          dtptr->O2attime, dtptr->GLUattime[i]);

  fprintf (dtptr->ofp2,"\n\nTime (sec) = %d\n", wrttime);

  for (j=0; j <= dtptr->divisions; j++)
    {
      fprintf(dtptr->ofp2,
              "%5d       %9.3e   %9.3e   %9.3e   %9.3e  %9.3e\n",
              j, dtptr->CNPbdconc[i][j], dtptr->CAPbdconc[i][j],
              dtptr->O2bdconc[i][j], dtptr->GLUbdconc[i][j],
              dtptr->anbiomass[i][j]);
    }

  fprintf (dtptr->ofp3,"\n\nTime (sec) = %d\n", wrttime);

  for (j=0; j <= dtptr->divisions; j++)
    {
      fprintf(dtptr->ofp3,
              "%5d          %9.3e          %9.3e\n",
              j, dtptr->aerbiomass[i][j], dtptr->anbiomass[i][j]);
    }

  **static x;
}


int initbdconczero (double **bdconc, int divisions)
{
  int j;

  for (j=0; j <= divisions; j++)
    {
      bdconc[0][j]=0;
      bdconc[1][j]=0;
    }                                  /* Initializes the bead concentration
                                          of the input chemical to zero at
                                          all radii*/
  return 0;
}


int initbiomass(double **aerbiomass, double **anbiomass, int divisions,
                double initaercelldensty, double initancelldensty,
                double *v)
{
  int j;

  for (j=0; j <= divisions; j++)
    {
      aerbiomass[0][j] = initaercelldensty*v[j]*1000; /* The *1000 converts
                                                        volume (calculated
                                                        in liters, to ml */

      anbiomass[0][j] = initancelldensty*v[j]*1000;
    }
}


double  doCNPbdconc (double deltat, double deltar, double antrm,
                     double **CNPbdconc, int i, int j)
{
  double x;

  x = deltat*(DCNP*((CNPbdconc[i-1][j+1] - 2*CNPbdconc[i-1][j]+
                     CNPbdconc[i-1][j-1])*100/sqr(deltar)) +
                       2*DCNP*100/(j*deltar)*
                         ((CNPbdconc[i-1][j]-CNPbdconc[i-1][j-1])/deltar))
                           + CNPbdconc[i-1][j];
  if (x-(4*antrm) >= 0)  /* This expression ensures that the returned value
                            of conc. is not <0, which is a meaningless result */
    return x-(4*antrm);  /* the 4 in this expression is from stoichiometry*/
  else
    return 0;
```

```
}


double doGLUbdconc(double deltat, double deltar, double antrm,
                   double **GLUbdconc, int i, int j, double aertrm)
{
  double x;

  x = deltat*(DGLU*((GLUbdconc[i-1][j+1] - 2*GLUbdconc[i-1][j]+
                   GLUbdconc[i-1][j-1])*100/sqr(deltar))+
                     2*DGLU*100/(j*deltar)*
                       ((GLUbdconc[i-1][j]-GLUbdconc[i-1][j-1])/deltar))+
                         GLUbdconc[i-1][j];
  if ( x-antrm-aertrm >= 0 ) /*the 1 coeff's in this expression are from
                               stoichiometry */
    return x-antrm;
  else
    return 0;
}


double doCAPbdconc(double deltat, double deltar, int i, int j, double
                   **CAPbdconc, double antrm, double **aerbiomass, double
                   **GLUbdconc, double v, double aermmax, double **O2bdconc,
                   double CAPaertrm)

{
  double x, ret;

  x = deltat*(DCAP*((CAPbdconc[i-1][j+1] - 2*CAPbdconc[i-1][j]+
                   CAPbdconc[i-1][j-1])*100/sqr(deltar))+
                      2*DCAP*100/(j*deltar)*
                        ((CAPbdconc[i-1][j]-CAPbdconc[i-1][j-1])/deltar))+
                          CAPbdconc[i-1][j];



  if (GLUbdconc[i-1][j] > 0)
    ret = x -CAPaertrm *(4*antrm);
  else
    ret = x + (4*antrm);  /* the 4 in this expression is from stoichiometry */

  if (ret >= 0)
    return ret;
  else
    return 0;
}


double doO2bdconc (double deltat, double deltar, int i, int j, double
                   **O2bdconc, double aertrm, double CAPaertrm, int static_y,
                   struct datastruct *dtptr)
{
  double x, mu;

  x = deltat*(DO2*((O2bdconc[i-1][j+1] - 2*O2bdconc[i-1][j]+
                   O2bdconc[i-1][j-1])*100/sqr(deltar))+
                     2*DO2*100/(j*deltar)*
                       ((O2bdconc[i-1][j]-O2bdconc[i-1][j-1])/deltar))
                         + O2bdconc[i-1][j];

  if ( x-(6*aertrm)-(6*CAPaertrm) >= 0 ) /* the 6's in this expression are
                                            from stoichiometry */
    return x-(6*aertrm)-(6*CAPaertrm);
  else
    return 0;

}

int initconc (struct datastruct *dtptr, double *v)

{
   double **dmatrix (int rows, int cols);
   double *dvector(int size);
   int div;

   div=dtptr->divisions;
   dtptr->CNPattime = dvector(3);   /* assigns a 3 member array to CNPattime */
   dtptr->CNPattime[0] = dtptr->initCNP; /* sets the intial concentration of
                                            CNP in the reactor */
   dtptr->CAPattime = dvector(3);
   dtptr->GLUattime = dvector(3);
   dtptr->GLUattime[0] = dtptr->initGLU; /* sets the initial concentration of
                                            glucose in the reactor */

   dtptr->CNPbdconc = dmatrix (3, div+1);  /* assigns a 3x(div+1) array to
                                              CNPbdconc */
   initbdconczero(dtptr->CNPbdconc, dtptr->divisions);

   dtptr->anbiomass = dmatrix (3, div+1);
   dtptr->aerbiomass = dmatrix (3, div+1);
   initbiomass(dtptr->aerbiomass, dtptr->anbiomass, div, dtptr->
               initaercelldensty, dtptr->initancelldensty, v);

   dtptr->GLUbdconc = dmatrix (3, div+1);
   initbdconczero(dtptr->GLUbdconc, div);

   dtptr->CAPbdconc = dmatrix (3, div+1);
   initbdconczero(dtptr->CAPbdconc, div);

   dtptr->O2bdconc = dmatrix (3, div+1);
   initbdconczero(dtptr->O2bdconc, div);

   printf("\nBeginning calculations.\n");
   return 0;
 }

int reacconc(struct datastruct *dtptr, int i, double k)
{
  double rv, flow, deltat, deltar, nb;
  int div;
  static int static_q=0;

  nb= dtptr->numofbeads;
  rv= dtptr->reacvol;
  div= dtptr->divisions;
  deltat= dtptr->deltat;
  deltar= dtptr->deltar;

  /*  This function calculates the concentration of the given chemical
      in the reactor at a given time */

  dtptr->CNPattime[i]= -4.0*PI*DCNP*sqr(dtptr->radius)*nb/
    (10000*dtptr->reacvol)*((dtptr->CNPbdconc[i-1][div]-dtptr->
                             CNPbdconc[i-1][div-1])/dtptr->deltar)*
                               deltat+dtptr->CNPattime[i-1];
```

```
  dtptr->CAPattime[i]= -4.0*PI*DCAP*sqr(dtptr->radius)*nb/
    (10000*dtptr->reacvol)*((dtptr->CAPbdconc[i-1][div]-dtptr->
                             CAPbdconc[i-1][div-1])/dtptr->deltar)*
                               deltat+dtptr->CAPattime[i-1];

  dtptr->GLUattime[i]= -4.0*PI*DGLU*sqr(dtptr->radius)*nb/
    (10000*dtptr->reacvol)*((dtptr->GLUbdconc[i-1][div]-dtptr->
                             GLUbdconc[i-1][div-1])/dtptr->deltar)*
                               deltat+dtptr->GLUattime[i-1];

  /* The following expression adds 25 mM glucose to the reactor at time=
     23 hrs as was done in the experiment by Beunink and Rehm */

  if (static_q==828000) dtptr->GLUattime[i] = dtptr->GLUattime[i] + 25e-3;

  ++static_q;

  return 0;
}

double doaerb (double deltat, double v, double aermmax, double GLUbdconc,
               double O2bdconc, double aerbiomass)
{
  double aert;

  /* This function calculates the aerobic term in the mass balance
     equation at a given time and radius (volume) */

  aert= deltat/v * aerbiomass*aermmax/aeryield*
    (GLUbdconc/(KGLU+GLUbdconc))*(O2bdconc/(KO2+O2bdconc));

  return aert;
}

double doCAPaert (double deltat, double v, double CAPmmax, double O2bdconc,
                  double CAPbdconc, double aerbiomass)
{
  double CAPaert;

  CAPaert= deltat/v * aerbiomass*CAPmmax/CAPyield*
    (O2bdconc/(KO2+O2bdconc))*(CAPbdconc/(KCAP+CAPbdconc));

  return CAPaert;
}

double doanb (double deltat, double v, double anmmax, double GLUbdconc,
              double CNPbdconc, double anbiomass, double O2bdconc)
{
  double ant;

  /* This function calculates the anaerobic term in the mass balance
     equation at a given time and radius (volume) */

  ant= deltat/v *anbiomass*anmmax/anyield*
    (CNPbdconc/(KCNP+CNPbdconc))*(GLUbdconc/((KGLUA+GLUbdconc)*
                                             (1+O2bdconc/KI)));

  return ant;
}

int dozeroradius (struct datastruct *dtptr, int i, double v)
{
  double deltat, deltar, anb, aerb, CAPaertrm, CNPx, GLUx, O2x, CAPx, ans;

  deltat = dtptr->deltat;
  deltar = dtptr->deltar;

  /* This function calculates all the necessary parameters for the
     volume at zero radius in the bead (actually a sphere of radius
     0.5 deltar with origin at the center of each bead.  This function
     is needed because zero radius is a boundary condition and special
     equations apply that do not apply to the rest of the bead. */

  anb = doanb(deltat, v, dtptr->anmmax, dtptr->GLUbdconc[i-1][0],
              dtptr->CNPbdconc[i-1][0], dtptr->anbiomass[i-1][0],
              dtptr->O2bdconc[i-1][0]);

  aerb = doaerb(deltat, v, dtptr->aermmax, dtptr->GLUbdconc[i-1][0],
                dtptr->O2bdconc[i-1][0], dtptr->aerbiomass[i-1][0]);

  CAPaertrm = doCAPaert(deltat, v, dtptr->CAPmmax, dtptr->O2bdconc[i-1][0],
                        dtptr->CAPbdconc[i-1][0], dtptr->aerbiomass[i-1][0]);

  CNPx = deltat*(2*DCNP*(dtptr->CNPbdconc[i-1][1]-dtptr->CNPbdconc[i-1][0])*
                 100/sqr(deltar))+dtptr->CNPbdconc[i-1][0];

  if ( CNPx-(4*anb) >= 0 )
    dtptr->CNPbdconc[i][0] = CNPx-(4*anb);
  else
    dtptr->CNPbdconc[i][0] = 0;


  GLUx = deltat*(2*DGLU*(dtptr->GLUbdconc[i-1][1]-dtptr->GLUbdconc[i-1][0])*
                 100/sqr(deltar))+dtptr->GLUbdconc[i-1][0];

  if ( GLUx-anb >= 0 )
    dtptr->GLUbdconc[i][0] = GLUx-anb-aerb;
  else
    dtptr->GLUbdconc[i][0] = 0;


  O2x = deltat*(2*DO2*(dtptr->O2bdconc[i-1][1]-dtptr->O2bdconc[i-1][0])*
                100/sqr(deltar))+dtptr->O2bdconc[i-1][0];

  if ( O2x-(6*aerb)-(6*CAPaertrm) >= 0 )
    dtptr->O2bdconc[i][0] = O2x-(6*aerb)-(6*CAPaertrm);
  else
    dtptr->O2bdconc[i][0] = 0;


  CAPx = deltat*(2*DCAP*(dtptr->CAPbdconc[i-1][1]-dtptr->CAPbdconc[i-1][0])*
                 100/sqr(deltar))+dtptr->CAPbdconc[i-1][0];

  ans = CAPx - CAPaertrm +(4*anb);

  if (ans >= 0)
    dtptr->CAPbdconc[i][0] = ans;
  else
    dtptr->CAPbdconc[i][0] = 0;


  dtptr->anbiomass[i][0] = dtptr->anmmax*(dtptr->GLUbdconc[i-1][0]/
                                          (KGLUA+dtptr->GLUbdconc[i-1][0]))*
                             (dtptr->CNPbdconc[i-1][0]/
                               (KCNP+dtptr->CNPbdconc[i-1][0]))*
```

```
                                                dtptr->anbiomass[i-1][0]
                                                  *deltat+dtptr->
                                                    anbiomass[i-1][0];

  dtptr->aerbiomass[i][0] = (dtptr->aermmax*(dtptr->GLUbdconc[i-1][0]/
                                              (KGLU+dtptr->GLUbdconc[i-1][0]))*
                            (dtptr->O2bdconc[i-1][0]/(KO2+dtptr->O2bdconc[i-1][0]))
                                  +dtptr->CAPmmax*
                                    (dtptr->GLUbdconc[i-1][0]/
                                     (KGLU+dtptr->GLUbdconc[i-1][0]))*
                            (dtptr->CAPbdconc[i-1][0]/
                             (KCAP+dtptr->CAPbdconc[i-1][0]))*
                          dtptr->aerbiomass[i-1][0]*deltat + dtptr->aerbiomass[i-1][0];


  return 0;
}

int checkconsts(double deltat, double deltar)
{
  double CNPtest, CNPtest2, CAPtest, CAPtest2, O2test, O2test2,
         GLUtest, GLUtest2;


  /*  This function checks to see if the constants in the diffusion
      parts of the mass balance equation are less that a certain value,
      as defined by <test> (defined at the top.  This is to ensure that
      certain conditions for the numerics are not violated, which would
      lead to a breakdown in the validity of the entire program.  */

  CNPtest = deltat*DCNP*100/sqr(deltar);
  CNPtest2 = deltat*2*DCNP*100/(deltar*deltar);
  if (CNPtest>test || CNPtest2>test)
    {
      printf("/nERROR:  CNP constant parameter out of range.\n");
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", CNPtest,
             CNPtest2, test);
      exit(-1);
    }

  CAPtest = deltat*DCAP*100/sqr(deltar);
  CAPtest2 = deltat*2*DCAP*100/(deltar*deltar);
  if (CAPtest>test || CAPtest2>test)
    {
      printf("\nERROR:  CAP constant parameter out of range.\n");
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", CAPtest,
             CAPtest2, test);
      exit(-1);
    }

  O2test = deltat*DO2*100/sqr(deltar);
  O2test2 = deltat*2*DO2*100/(deltar*deltar);
  if (O2test>test || O2test2>test)
    {
      printf("\nERROR:  O2 constant parameter out of range.\n");
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", O2test, O2test2,
             test);
      exit(-1);
    }

  GLUtest = deltat*DGLU*100/sqr(deltar);
  GLUtest2 = deltat*2*DGLU*100/(deltar*deltar);
  if (GLUtest>test || GLUtest2>test)
    {
      printf("\nERROR:  glucose  constant parameter out of range.\n");
      printf("Coefficients - %.11f  %.11f must be < %.2f.\n", GLUtest,
             GLUtest2, test);
      exit(-1);
    }
  return 0;
}

int dovartest (struct datastruct *dtptr, int i, int j, int static_y)
{
  double anCNPtest, anGLUtest, aerGLUtest, aerO2test, deltat, deltar;

  /* This is similar to the checkconsts function above in that it checks
     certain expressions to ensure they are less that <test> defined at
     the top.  This function is called at every time step due to the fact
     that the expressions in question depend on time-varying concentrations,
     where those in checkconsts do not, and that function is called only
     once. */

  deltat = dtptr->deltat;
  deltar = dtptr->deltar;

  anCNPtest = deltat*dtptr->anbiomass[i-1][j]*1e6*dtptr->anmmax/
    (anyield*4*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->GLUbdconc[i-1][j]/(KGLUA+dtptr->GLUbdconc[i-1][j]))/
        (KCNP+dtptr->CNPbdconc[i-1][j])*(1/(1+dtptr->O2bdconc[i-1][j]/KI));

  anGLUtest = deltat*dtptr->anbiomass[i-1][j]*1e6*dtptr->anmmax/
    (anyield*4*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->CNPbdconc[i-1][j]/(KCNP+dtptr->CNPbdconc[i-1][j]))/
        (KGLUA+dtptr->GLUbdconc[i-1][j])*(1/(1+dtptr->O2bdconc[i-1][j]/KI));

  if (anCNPtest > test || anGLUtest > test)
    {
      printf("\nERROR:  anaerobic parameter out of range.\n");
      printf("Indices - [%d][%d].\n", i-1, j);
      printf("CNPtest - %.2lf, GLUtest - %.2lf must be < %.2f.\n",
             anCNPtest, anGLUtest, test);
      exit(-1);
    }

  aerGLUtest = deltat*dtptr->aerbiomass[i-1][j]*1e6*dtptr->aermmax/
    (aeryield*4*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->O2bdconc[i-1][j]/(KO2+dtptr->O2bdconc[i-1][j]))/
        (KGLU+dtptr->GLUbdconc[i-1][j]);

  aerO2test = deltat*dtptr->aerbiomass[i-1][j]*1e6*dtptr->aermmax/
    (aeryield*4*PI*(cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)))*
      (dtptr->GLUbdconc[i-1][j]/(KGLU+dtptr->GLUbdconc[i-1][j]))/
        (KO2+dtptr->O2bdconc[i-1][j]);

  if (aerGLUtest > test || aerO2test > test)
    {
      printf("\nERROR:  aerobic parameter out of range.\n");
      printf("Indices - [%d][%d].\n", i-1, j);
      printf("GLUtest - %.3lf, O2test - %.3lf must be < %.3f.\n",
             aerGLUtest, aerO2test, test);
      if (aerO2test>test)
        {
          printf("deltat= %.11f\n", deltat);
          printf("aerbiomass= %.3e\n", dtptr->aerbiomass[i-1][j]);
          printf("aermmax= %.3e\n", dtptr->aermmax);
          printf("aeryield= %.3e\n", aeryield);
          printf("v= %.3e\n",4*PI*
```

```
                (cub(j*deltar+deltar/2)-cub(j*deltar-deltar/2)));
        printf("GLUbdconc= %.3e\n", dtptr->GLUbdconc[i-1][j]);
        printf("KGLU= %.3e\n", KGLU);
        printf("O2bdconc= %.3e\n", dtptr->O2bdconc[i-1][j]);
        printf("KO2= %.3e\n", KO2);
      }
    exit(-1);
  }

 return 0;


}


double *dov (double deltar, int div)
{
  int j;
  double *v, *dvector(int size);

  /* This function determines the volume of each anulus as defined by
     the radius and the number of divisions */

  v=dvector(div+1);

  for (j=0; j <= div; j++)
    {
      if (j==0)   v[j]= 4.0/3.0*PI*cub(deltar)/1e6;
      else
        v[j]= 4.0/3.0*PI*(cub(j*deltar+deltar)-cub(j*deltar-deltar))/1e6;
    }

  return v;
}

int baseline (struct datastruct *dtptr, int i)
{
  int j;

  /* This function resets the baseline at the end of each loop of three
     in chemfunctime.  It sets the concentrations, etc., at i-2 to those
     at i=0 for use during the next loop (starting at 1), so i-1=0. */

  for (j=dtptr->divisions; j>=0; --j)
    {
      dtptr->CNPbdconc[0][j] = dtptr->CNPbdconc[1][j];
      dtptr->CAPbdconc[0][j] = dtptr->CAPbdconc[1][j];
      dtptr->O2bdconc[0][j] = dtptr->O2bdconc[1][j];
      dtptr->GLUbdconc[0][j] = dtptr->GLUbdconc[1][j];
      dtptr->aerbiomass[0][j] = dtptr->aerbiomass[1][j];
      dtptr->anbiomass[0][j] = dtptr->anbiomass[1][j];
    }
}

int chemfunctime (struct datastruct *dtptr)
{

   int i, j, c, wrtinterv, div, d, F, s;
   double k, aertrm, antrm, CAPaertrm, deltat, deltar, *v;
   static int static_y=0;

   /* This function calculates the concentration of the different chemicals
      of interest at each radius at every time. */
```

```
wrtinterv=dtptr->writeinterval;
div= dtptr->divisions;
deltat= dtptr->deltat;
deltar= dtptr->deltar;


checkconsts(dtptr->deltat, dtptr->deltar);

k=(double) 4*PI*dtptr->numofbeads*sqr(dtptr->radius)/(10000*
                                                      dtptr->reacvol);

v=dov(deltar, div);

initconc(dtptr, v);

for (c=1; c <= dtptr->time/2; c++)
  {
    for (i=1; i <= 2; i++)
      {
        ++static_y;

        dozeroradius(dtptr, i, v[0]);
        dtptr->CNPbdconc[i][div] = dtptr->CNPattime[i-1];
        dtptr->CAPbdconc[i][div] = dtptr->CAPattime[i-1];
        dtptr->GLUbdconc[i][div] = dtptr->GLUattime[i-1];
        dtptr->O2bdconc[i][div] = dtptr->O2attime;

        j = div;
        dtptr->anbiomass[i][div] = dtptr->anmmax*
          (dtptr->GLUbdconc[i-1][j]/(KGLU+dtptr->GLUbdconc[i-1][j]))*
            (dtptr->CNPbdconc[i-1][j]/(KCNP+dtptr->CNPbdconc[i-1][j]))
           *dtptr->anbiomass[i-1][j]*deltat+dtptr->anbiomass[i-1][j];

        dtptr->aerbiomass[i][div] = (dtptr->aermmax*
          (dtptr->GLUbdconc[i-1][j]/(KGLU+dtptr->GLUbdconc[i-1][j]))*
            (dtptr->O2bdconc[i-1][j]/(KO2+dtptr->O2bdconc[i-1][j]))+
              dtptr->CAPmmax*(dtptr->GLUbdconc[i-1][j]/
                              (KGLU+dtptr->GLUbdconc[i-1][j]))*
            (dtptr->CAPbdconc[i-1][j]/(KCAP+dtptr->CAPbdconc[i-1][j])))
              *dtptr->aerbiomass[i-1][j]*deltat+
                dtptr->aerbiomass[i-1][j];

        for (j= dtptr->divisions-1; j > 0; j--) /*calculates all concent.'s
                                                  at each radius (j) at a
                                                  given time (i)  */
          {
            aertrm= doaerb(deltat, v[j], dtptr->aermmax,
                           dtptr->GLUbdconc[i-1][j], dtptr->
                           O2bdconc[i-1][j], dtptr->aerbiomass[i-1][j]);

            CAPaertrm= doCAPaert(deltat, v[j], dtptr->CAPmmax, dtptr->
                                 O2bdconc[i-1][j], dtptr->
                                 CAPbdconc[i-1][j], dtptr->
                                 aerbiomass[i-1][j]);

            antrm= doanb(deltat, v[j], dtptr->anmmax,
                         dtptr->GLUbdconc[i-1][j], dtptr->CNPbdconc[i-1][j]
                         ,dtptr->anbiomass[i-1][j],dtptr->O2bdconc[i-1][j]);

            dovartest(dtptr, i, j, static_y);

            dtptr->anbiomass[i][j] = dtptr->anmmax*
              (dtptr->GLUbdconc[i-1][j]/
               (KGLU+dtptr->GLUbdconc[i-1][j]))*
```

```
                (dtptr->CNPbdconc[i-1][j]/(KCNP+dtptr->CNPbdconc[i-1]
                                        [j]))*dtptr->anbiomass[i-1][j]
                                          *deltat + dtptr->
                                            anbiomass[i-1][j];

            dtptr->CNPbdconc[i][j]=doCNPbdconc(deltat, deltar, antrm,
                                        dtptr->CNPbdconc, i, j);

            dtptr->GLUbdconc[i][j]=doGLUbdconc(deltat, deltar, antrm,
                                        dtptr->GLUbdconc, i, j,
                                        aertrm);

            dtptr->CAPbdconc[i][j]=doCAPbdconc(deltat, deltar, i, j,
                                        dtptr->CAPbdconc, antrm,
                                        dtptr->aerbiomass,
                                        dtptr->GLUbdconc, v[j],
                                        dtptr->aermmax, dtptr->
                                        O2bdconc, CAPaertrm);


            dtptr->aerbiomass[i][j] = (dtptr->aermmax*
              (dtptr->GLUbdconc[i-1][j]/
               (KGLU+dtptr->GLUbdconc[i-1][j]))*(dtptr->O2bdconc
                                          [i-1][j]/
                                          (KO2+dtptr->
                                           O2bdconc[i-1][j]))*
                                    dtptr->CAPmmax*
                                    (dtptr->GLUbdconc[i-1][j]/
                                     (KGLUA+dtptr->GLUbdconc[i-1][j]))*
                                    (dtptr->CAPbdconc[i-1][j]/
                                     (KCAP+dtptr->CAPbdconc[i-1][j]))*
              dtptr->aerbiomass[i-1][j]*deltat + dtptr->aerbiomass[i-1][j];

            dtptr->O2bdconc[i][j]=doO2bdconc(deltat, deltar, i, j, dtptr->
                                        O2bdconc, aertrm, CAPaertrm,
                                        static y, dtptr);


          }                          /*  this closes for }  */
      /*  this calculates the chem.  conc. at a given time in the reactor  */
          reacconc (dtptr, i, k);

        if (i==2)
          {
            dtptr->CNPattime[0] = dtptr->CNPattime[1];
            dtptr->CAPattime[0] = dtptr->CAPattime[1];
            dtptr->GLUattime[0] = dtptr->GLUattime[1];

            baseline(dtptr, i);

          }


        if ((int)(static_y)%(wrtinterv*10)==0)
          {
            d= i-1;
            writeoutput(dtptr, (int)(static_y/10), d);
            printf("Finished writing i = %d\n", (int) (static_y/10));
          }
      }                          /*  closes the for i  */
    }                            /*  closes the for c  */
  return 0;
}
```