Numerical problems and agent-based models for a mass transfer course

Manohar Murthi, Lonnie D. Shea, Randall Q. Snurr[†]

Department of Chemical and Biological Engineering
Northwestern University Evanston, IL 60208

Abstract

Problems requiring numerical solutions of differential equations or the use of agent-based modeling are presented for use in a course on mass transfer. These problems were solved using the popular technical computing language MATLAB^{TM}. Students were introduced to MATLAB via a problem with an analytical solution. A more complex problem to which no analytical solution exists was then assigned. An example follows using agent-based simulation techniques to solve problems that are difficult to formulate in terms of differential equations. This example was used as a course project in which a kinetic Monte Carlo simulation was used by students to investigate progenitor cell invasion into a polymeric tissue engineering scaffold.

Keywords

Mass transfer, MATLAB, numerical problems, computer simulation, tissue engineering

Version: September 24, 2007

Submitted to *Chemical Engineering Education*

[†]Author to whom correspondence should be addressed (snurr@northwestern.edu)

Introduction

In the traditional undergraduate chemical engineering curriculum, the mathematical formulations of engineering problems are solved almost exclusively by analytical methods. A prototypical example is the analytical solution of differential equations in transport. However, many interesting - and not particularly exotic - problems cannot be solved analytically. In the past, limiting cases and approximations might have been used to circumvent this problem. But with the availability of fast computers and user-friendly software for numerical computation, we risk becoming irrelevant if we do not equip our students to use these new tools. Other disciplines are also concerned about this problem. For example, concerns about whether undergraduates are equipped to tackle problems of increasing complexity recently led the Mathematical Association of America to convene a working group of chemists to examine the connections between the undergraduate mathematics and chemistry curricula.[1]

The mass transfer course is an ideal one for including numerical solution methods for chemical engineers, because the students have already seen the same mathematical methods in their fluid mechanics and heat transfer courses. At Northwestern University, momentum, heat, and mass transport are taught separately in successive quarters in the third (junior) year. Fluid mechanics is taught first, along with the requisite vector analysis, followed by heat transfer. By the time mass transfer is encountered, the analogies to fluid and heat transfer can be used to speed the coverage of the core material, leaving more time for exploration of other topics, such as numerical methods or short group projects.

At Northwestern University's McCormick School of Engineering and Applied Science, the computer solution of numerical problems is introduced to first-year students through the Engineering First program using MATLAB™.[2] While FORTRAN 77 was often taught to chemical engineering students in the past, MATLAB now seems to be a favored introductory programming language in the United States. It has the advantage of incorporating a graphical user interface and good graphing capabilities. In addition, MATLAB boasts a range of ODE and PDE solvers that are of particular use for transport problems. With the widespread availability of such tools built into the software and on the internet, it is no longer necessary to make students write their own code to solve

these types of problems. Researchers in industry also typically operate by using numerical codes written and published by experts in the relevant area of numerical computation. The researcher is thus able to devote more time to solving problems in his or her own field of expertise. Therefore the ability to understand and use existing pieces of high-level code is of great practical importance.

Discrete particle- or agent-based simulations are rarely taught to undergraduates, but they are becoming increasingly commonplace in fields as diverse as materials science, transportation, and sociology.[3] One could argue that such skills are of more use to a chemical engineering graduate entering the workforce today than the knowledge of rigorous distillation column design. Stochastic simulation has become an integral part of the computational chemist's or biologist's toolbox, with a number of practitioners calling for its inclusion in the undergraduate curriculum.[4] As chemical engineers have also embraced molecular simulation in their research, some chemical engineering departments have added undergraduate courses in molecular simulation and theory, covering topics such as the estimation of thermochemical and reaction rate data and the prediction of phase equilibria and transport properties by molecular simulation.[5]

Solving problems numerically forces students to think about mathematics in a different light, in that it is no longer possible to simply learn the analytical solution. Some element of programming is invariably involved in solving equations numerically, forcing students to break the solution down into the simplest steps possible, which can in turn lead to a more concrete understanding of the mathematical relationships. In order for students to become comfortable with solving problems numerically, such problems must be incorporated throughout the curriculum. In this paper, we give examples of problems we posed to expose students to numerical solutions and simulations in the context of mass transfer. The following sequence of problems was assigned to increase the students' familiarity with the solution of ODE's using MATLAB. We started off with a diffusion problem that could be solved analytically and had the students compare the analytical solution to the numerical solution generated by a boundary value solver intrinsic to MATLAB. We also encouraged them to experiment with different values of the parameters fed to the boundary-value solver to see whether the quality of the numerical solution or the time taken to reach it were affected. We then posed a problem with reaction and diffusion that had no analytical solution. Students

were asked to vary the reaction rate constant until the concentration profiles in the system began to resemble those for the limiting case of reaction being much faster than diffusion. Finally, we presented students with a lattice simulation of cells invading a polymer tissue engineering scaffold and had them experiment with the different parameters controlling the rate of cellular invasion and the concentration profile of the invading cells in the polymer matrix.

All of the problems discussed in this paper can be found at the website

`http://zeolites.cqe.northwestern.edu/Publications/SupportingInfo/Download.html`.

Instructors can obtain MATLAB solutions by sending e-mail to snurr@northwestern.edu.

An introductory ODE problem with an analytical solution

To provide a fairly gentle reintroduction to MATLAB, we first asked students to determine the concentration profile and flux of a liquid, A, slowly evaporating into a gaseous mixture of A and B from a reservoir of pure A located at the bottom of a cylindrical tube. Since the evaporation of the liquid is slow, one may assume that the surface of the liquid is stationary. This example is covered in many texts and an analytical solution is straightforward. We used the text by Middleman,[6] where this problem is solved as Example 2.1.1. We asked the students to generate a numerical solution using MATLAB and compare this with the analytical solution. The steady-state equations to be solved are

$$\begin{aligned} \frac{dx}{dz} &= \frac{-(1-x)N}{CD} \\ \frac{dN}{dz} &= 0 \end{aligned} \tag{1}$$

where x is the mole fraction of A in the gas phase, N is the flux of A, C is the total concentration in the gas phase, and D is the diffusivity of gaseous A in B. The boundary conditions are that at $z = 0$, $x = x_0$ (determined from the vapor pressure of A at the liquid/gas interface), and at $z = L$ (end of the tube), $x = x_1$, the mole fraction of x in the gas mixture flowing past the tube.

MATLAB incorporates a two-point boundary value problem solver called bvp4c, which is useful in solving this problem. The bvp4c solver requires as input an initial guess to the solution in a data structure that consists of the initial mesh points in the domain of the solution and the values of the

initial guess for each mesh point. This data structure can be formed by another function intrinsic to MATLAB called bvpinit. The other inputs to bvp4c are the functions describing the system of ODE's and the boundary conditions. Our ODE function consisted of just the first of the above equations, with N as a parameter to be determined. Since most students had only encountered initial value ODE problems, we first discussed the differences between initial value and boundary value problems in class, pointing out that the latter can have no solutions or an infinite number of solutions. A hands-on introduction to MATLAB covered the use of file input and output, functions, function handles, data structures, and graphing. This was followed by fairly explicit guidance in the use of the boundary value solver functions. Students were also encouraged to try different initial guesses for the solution and the unknown parameter and to examine the effects on the validity of the solution and the time taken to reach it. Increasing the number of initial mesh points from 10 to 10,000 resulted in an increase in computation time from a few seconds to over a minute but did not affect the quality of the solution for this simple problem. However, certain initial guesses for the concentration profile resulted in numerical disasters, with the result that no solution could be found. The solution was completely insensitive to the initial guess for the unknown parameter, N. Finding that this parameter was in exact agreement with the analytical solution strengthened the students' confidence in their ability to obtain correct numerical solutions.

A more challenging boundary value problem

We followed by posing a problem without an analytical solution, in this case Middleman's Example 3.2.9. This example consists of steady-state binary diffusion through a stagnant liquid film in which a second order reaction is taking place. The equations to solve are

$$\begin{aligned} D_1 \frac{d^2 C_1}{dz^2} - k C_1 C_2 &= 0 \\ D_2 \frac{d^2 C_2}{dz^2} - k C_1 C_2 &= 0 \end{aligned} \tag{2}$$

At $z = 0$, C_1 is maintained at a constant value of C_{10}, and component 2 is not allowed to cross the plane, i.e., $dC_2/dz = 0$. At $z = L$, C_2 is maintained at a constant value of C_{2L}, and all of component 1 is required to react within the film so that $C_1 = 0$.

The concentration profiles in the film for different rate constants are plotted in figure 1. While no analytical solution is possible, an approximation in the limit of fast reaction is possible. If the reaction is sufficiently fast and there is no excess of either species, the concentration of each species becomes essentially zero upon crossing a small zone known as the reaction front, in which almost all the reaction takes place. This behavior can be seen in the concentration profiles in figure 1 corresponding to the highest values of the reaction rate constant, k. The location of the reaction front, L_f can be found from the realization that, due to the stoichiometry of the reaction, the fluxes of the two components must be equal. Then

$$L_f = L/(1 + \frac{C_{2L}D_2}{C_{10}D_1}) \tag{3}$$

Since the bvp4c solver can only be used to solve systems of first order equations, the above system of equations and boundary conditions must be rewritten as such via the substitution $y_i = dC_i/dz$, which results in a system of four first-order equations to go with the four boundary conditions. The students could be expected to arrive at this realization on their own. They were asked to calculate the concentration profiles of the components in the film and find the value of the rate constant at which the reaction front approximation becomes valid. This exercise was repeated with different diffusivity values to emphasize that the behavior of the system is governed by the ratio of reaction rate constant to diffusivity, essentially the Thiele modulus.

Further numerical projects

The tools that students gained in the problems above could be readily extended to explore multicomponent diffusion, a topic that is often completely ignored even in graduate transport courses. For example, consider binary diffusion of species i and j through a zeolite membrane. The membrane can be considered a third, non-diffusing component. The fluxes of the two species with respect to the stationary membrane frame of reference can be written as[7]

$$\begin{aligned} J_i &= -D_{ii}\nabla C_i - D_{ij}\nabla C_j \\ J_j &= -D_{ji}\nabla C_i - D_{jj}\nabla C_j \end{aligned} \tag{4}$$

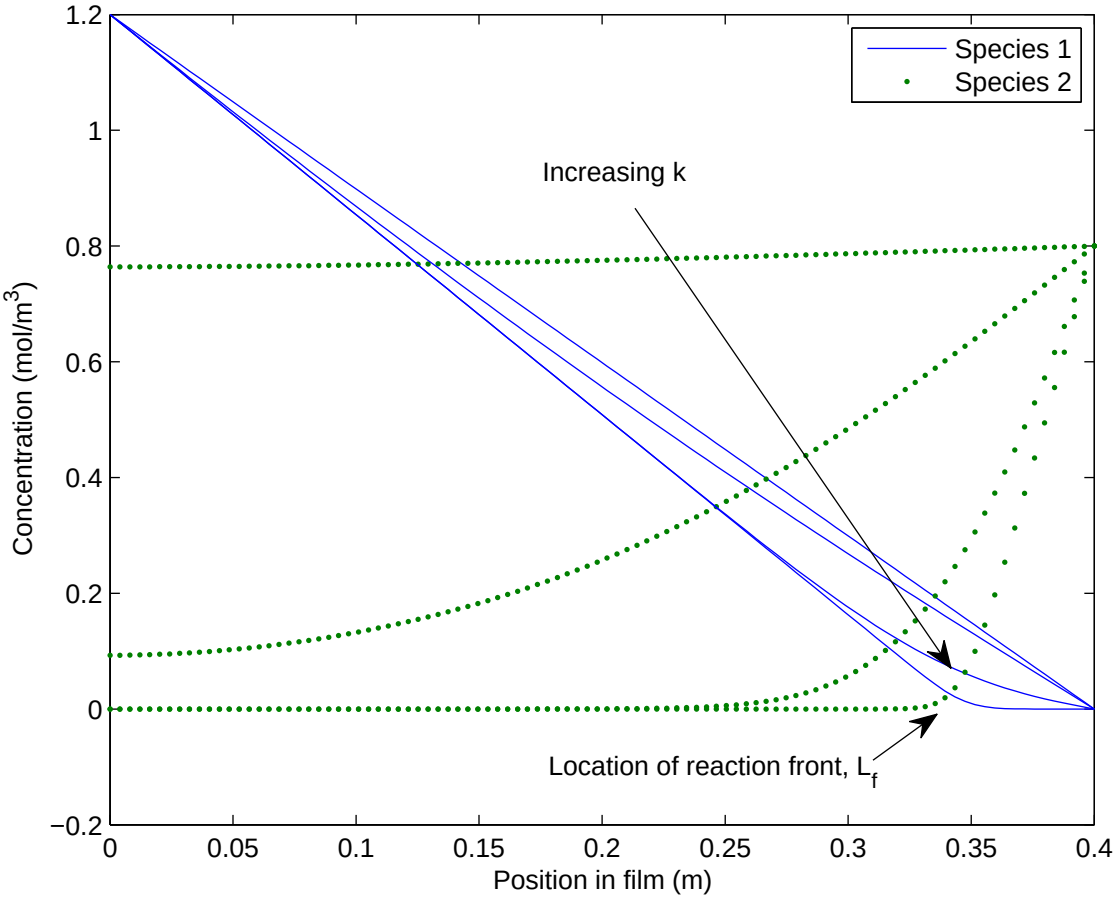

Figure 1: Binary diffusion and second-order reaction in a stagnant film. The concentration profiles of each species are shown for four different values of the reaction rate constant, with the diffusivity of each species held constant. If reaction is much faster than diffusion, almost all the reaction takes place at the reaction front, marked on the figure as L_f.

where J_i is the flux of component i, D_{ii} is its main-term diffusivity, and D_{ij} is its cross-term diffusivity. In general $D_{ij} \neq D_{ji}$, and for guest molecules in zeolites all of the diffusion coefficients may depend on the concentrations of both species. Sanborn and Snurr have reported the concentration-dependent diffusion coefficients for mixtures of methane and CF_4 in the zeolite faujasite, calculated from equilibrium molecular dynamics simulations and fitted to simple analytical functions of the concentrations.[8] They used MATLAB to solve equations 4 for several interesting sets of boundary conditions. For "co-diffusion" boundary conditions, both species have a higher concentration on one side of the membrane. For "counter-diffusion" boundary conditions, species i has a higher concentration on one side but species j has a higher concentration on the other side of the membrane. Normally, one would expect species i and j then to diffuse in opposite directions (counter-diffusion). Depending on the magnitudes of the main- versus cross-term diffusivities, however, it is possible that both species will diffuse in the same direction.[8] This illustrates that for molecules in tightly confined spaces, such as zeolite pores, the seemingly esoteric cross terms may, in fact, be important. As an interesting question, one may ask the students if this violates the laws of thermodynamics. (It does not, as both species still diffuse downhill in chemical potential, illustrating that chemical potential and not concentration is the real driving force for diffusion.)

Multicomponent diffusion is often ignored in courses because of the lack of access to the needed diffusion coefficients and because numerical solutions of the differential equations are usually required. This example shows that modern molecular simulations may provide access to difficult-to-measure multicomponent diffusivities. It also shows that the numerical solution of the differential equations is easily undertaken with widely available software such as MATLAB.

Agent-based modeling

Certain types of problems may be more easily modeled as computer algorithms than as differential equations. While this type of modeling is now commonplace, competing with and sometimes replacing equation-based approaches, chemical engineering students tend to be almost entirely unacquainted with it by the end of their undergraduate careers.[3] An intriguing and intuitive example of this type of modeling is a simulation of ants foraging for food,[9] which can be found

at the website of Northwestern University's Center for Connected Learning and Computer-Based Modeling,

`http://ccl.northwestern.edu/netlogo/models/Ants.`

This simulation concerns the manner in which a colony of ants finds food and transports it back to their anthole. It could easily be used as an in-class demonstration of a different mechanism for mass transfer, albeit one that would be extremely difficult to describe by differential equations.

In this simulation, each ant encountering a large piece of food deposits a chemical trail as it transports some of that food back to the nest. The chemical trail evaporates and diffuses over time. Ants can sniff out the chemical trail and follow its gradient uphill. A similar mechanism using a "nest scent" is used to return to the nest once food has been found. The chemical trails are reinforced by repeated traversal, which induces more ants to chip away at the food source until it is completely consumed. Once this occurs, the chemical trail begins to dissipate and the colony consequently forages elsewhere. The colony generally exploits food sources closest to the nest before foraging further afield since the evaporation and diffusion of the chemical retards the formation of stable chemical trails to more distant food sources.

The ant simulation described above is an example of agent-based modeling based on the cellular automaton paradigm. Cellular automaton simulations are used to model complex systems comprised of interacting autonomous agents. Agent behaviors are modeled explicitly, using a range of behavioral models and representation schemes at appropriate levels of detail. The crux of agent-based modeling and simulation is that agents only interact and exchange locally available information with other agents in their immediate vicinity. What constitutes an agent's "immediate vicinity" varies depending on the type of system being simulated. For example, neighbors may be spatially close for a simulation in continuous space, occupying adjacent grid cells in a lattice simulation, or connected nodes in a simulation of a network. Generally, an agent's set of neighbors changes rapidly as a simulation proceeds. The problem of identifying an agent's neighbors can dominate the computational expense of such a simulation, particularly as the number of agents increases. Different algorithms for neighbor-searching vary dramatically in performance depending on the topology of the neighbor interaction, the language used to program the search, and the

platform on which the calculation is run.[10] For example, Mathematica™, with its high-level list processing functions and cellular automata package, can run these simulations extremely efficiently. However, there is a substantially steeper learning curve to programming in Mathematica due to the variety of programming paradigms it supports, and since our students already had acquaintance with MATLAB, we decided to implement an agent-based simulation project in MATLAB.

We developed an agent-based simulation project relevant to chemical engineering based on the cellular automaton simulations of Shea and Pasquini[11] of cells invading a polymer tissue engineering scaffold representative of those implanted in a wound or regenerating tissue. The scaffold, which serves to maintain a space conducive to tissue formation, contains an interconnected open pore structure which can either be seeded with progenitor cells or be infiltrated by such cells from the surrounding tissue. The progenitor cells within the scaffold migrate, proliferate, and differentiate to form a functional tissue which must eventually be integrated with the host. Though tissue engineering is currently a very active area of academic and industrial research, this was our students' first encounter with the field.

The scaffold consists of impermeable polymer walls which define a pore structure. These walls are modeled as evenly spaced planes in the x-, y- , and z-directions, which form cubic cavities called macropores. There are randomly placed holes called micropores in the macropore walls. Cells move between macropores through these micropores. (Note that the terms "macropore" and "micropore" as used in tissue engineering do not correspond to the standard IUPAC definitions.)

A lattice model is used to simulate the motion of cells in this scaffold, with the lattice spacing being equal to the cell size. Thus each lattice site can only be occupied by a single cell. The macropore walls are several tens of lattice spacings apart and a single lattice spacing in thickness, and the micropores in our simulations are a single lattice spacing in size. Cells can occupy vacant sites in this model pore network and migrate or reproduce if space is available. Each cell must, however, remain in some physical contact with the solid support of the walls, either directly or through adhesion to other cells which are in contact with the walls. This model is used to examine how cells penetrate from the external surface of the scaffold to its interior, ultimately filling the void space as they reproduce.

We used a simulation cell consisting of three macropores in the x- and y-directions and five macropores in the z-direction. The cells are initially placed at the bottom x-y plane, with periodic boundary conditions being used in the x- and y-directions. The z-direction points inward into the scaffold, with the cells initially filling the z = 0 plane and migrating in the positive z-direction. Several two-dimensional slices of this three-dimensional lattice are shown together in figure 2.

The mean square displacement for cellular migration is given by

$$\langle d^2 \rangle = nS^2Pt \tag{5}$$

where $\langle d^2 \rangle$ is the mean square displacement, n is the number of dimensions, S is the root mean square speed, and t is the time. P is the persistence time, i.e. the time for which the direction of the moving cell remains constant. Variations in P have been shown to have little impact on system behavior.[12] P was chosen to be the same as the simulation time step. Setting the displacement, d, equal to the lattice spacing, l, allows the calculation of the time step from the above equation:

$$t_{step} = \sqrt{l^2/3S^2} \tag{6}$$

At each time step, each cell moves to one of the adjacent empty lattice sites. The choice of site is determined by comparing the probability of migration to each neighboring site to a random number drawn from the uniform distribution between 0 and 1. The probability of migration to a particular site is dependent on the number of cell-cell and cell-wall interactions that the cell will experience in that site and the relative importance of these interactions. This probability is normalized by the sum of migration probabilities to all the sites neighboring the starting site at that time step:

$$\begin{aligned} P_{i,j} &= A_j / \sum_{nn} A_k \\ A_k &= nc * cp + nw \end{aligned} \tag{7}$$

where $P_{i,j}$ is the probability of moving from site i to site j, nc is the number of neighboring sites occupied by cells, nw is the number of neighboring sites occupied by polymer wall, cp is the ratio of cell-cell cohesivity to cell-polymer adhesivity, and $\sum_{nn} A_k$ is the sum of A_k over all sites that are nearest neighbors to site i.

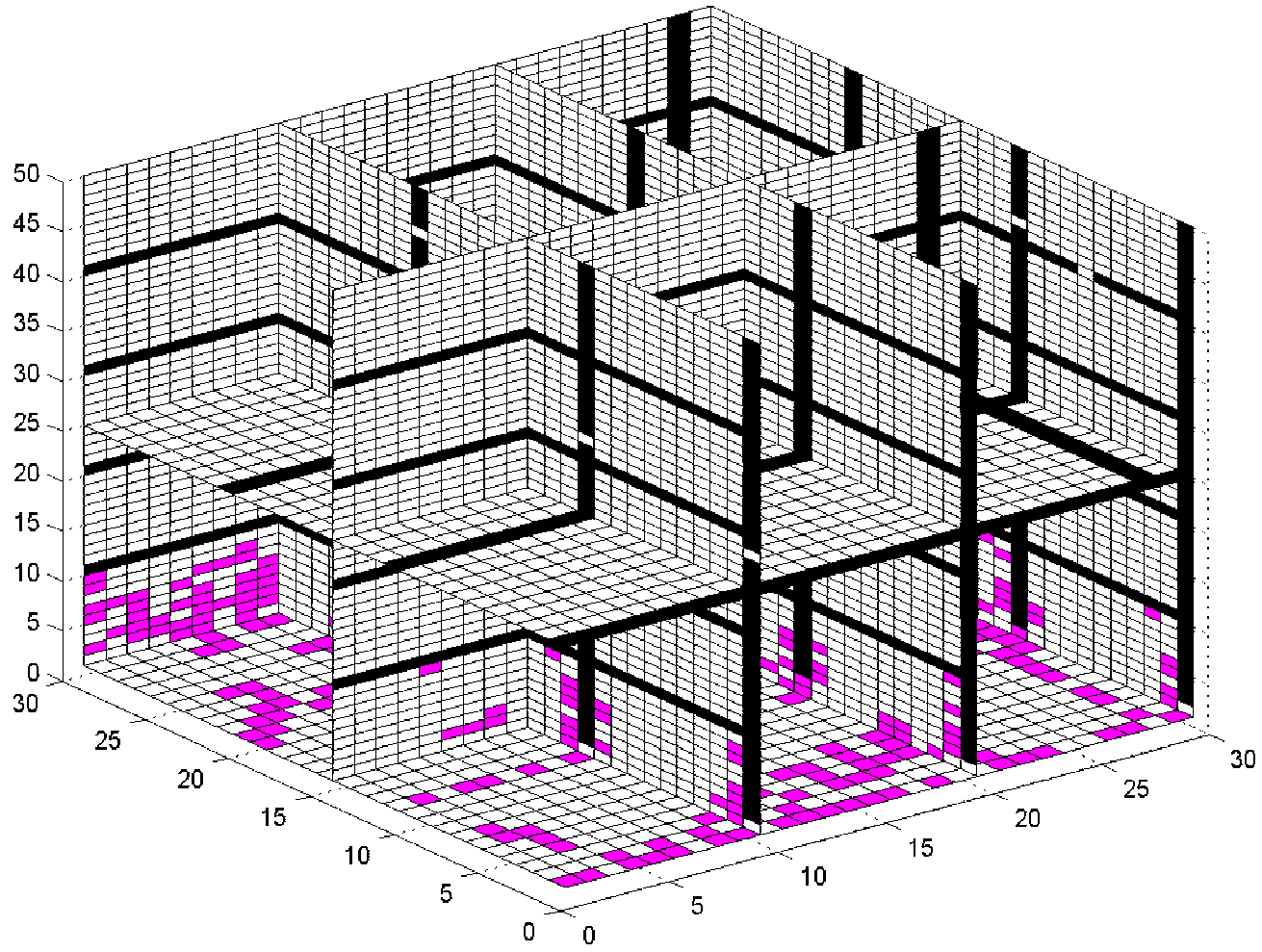

Figure 2: 2-D slices of a 3-D polymeric tissue engineering scaffold. The scaffold is placed within some healing tissue and serves to maintain a space conducive to the growth of new tissue from progenitor cells. The polymeric walls form macropores which are connected by randomly placed micropores. The scaffold is shown divided into lattice sites. Empty lattice sites in the scaffold are white, sites occupied by polymer walls are black, and sites occupied by cells are purple (gray). The micropores can be seen as holes in the black macropore walls. In this simulation the progenitor cells are initially present at the bottom of the scaffold (z = 0 plane), and migrate vertically into it, in the positive z-direction. A selection of horizontal and vertical slices through the scaffold are shown.

In addition to migrating, the cells can proliferate. Each cell is initially set to proliferate at a time randomly drawn from a normal distribution. At each time step, a counter indicating the cell's time to proliferation is decreased until it becomes negative, indicating that the cell should now divide. If space is available in one of the adjacent lattice sites, the cell will divide and the resulting new cell will occupy that site. The counters for both cells are again set randomly from the normal distribution of proliferation times.

The cells infiltrating the scaffold must maintain some physical connection to solid support, so no cluster of cells can be completely disconnected from the wall. Moves which isolate a cluster of cells from the walls must be disallowed. Therefore, before a move is accepted, one must check that every cell neighbouring the one about to be moved has some connection to the walls and that the cell in its new location will have some connection to the walls. This is done by a depth-first search (DFS) algorithm[13] adapted from that of Kevin Murphy, found at

`http://www.cs.ubc.ca/~murphyk/Software/.`

The DFS is the most computationally expensive part of the program.

It is instructive to compare the lattice model formulation of this "diffusion and reaction" problem with an attempt to cast it as a system of differential equations. The physical condition that no cell or cluster of cells can float unconnected to the solid support of the walls is particularly difficult to describe in the language of differential equations.

Since such a project involves a fair amount of coding effort, we provided the MATLAB code to the students. We asked them to run it for various values of macropore spacing, micropore fraction, and cp and to examine how these factors affect the time taken for cells to reach the top of the lattice and how the concentration profile of cells in the pore space varies. At low cp, the cells tend to penetrate into the lattice by crawling up the walls and leaving the centers of the macropores relatively empty. At higher values of cp, however, the increased preference for other cells over polymer walls results in a fairly uniform front of cells penetrating through the lattice. In this latter case, the cell proliferation rate determines the progress of the front, rather than the rate of migration along the walls. All of this is only possible if the micropore fraction is high enough to permit percolation from one end of the lattice to the other. Above this critical value, the micropore

fraction has minimal impact on the rate of cellular invasion.

For a reasonably large system of a few hundred lattice sites in each direction, the naive neighbor-searching algorithm we implemented (in which all cells are examined as possible neighbors) runs into recursion limits during the depth-first search. We wanted the students to run simulations for several different sets of parameters in a relatively short time, so we initially disabled the depth-first search, decreasing the run times to a minute or less even for large systems. Of course, this also meant that groups of cells could detach from the walls, unlike the work of Shea and Pasquini. Since we wanted the students to examine how the code actually worked, we refrained from explaining the physical meaning of *cp*, instead requiring the students to infer its meaning from the way in which it is used in the code. We also had the students modify the code to make the function call to the depth-first search routine and run simulations within a single small macropore, which took a few minutes per simulation. Students were additionally questioned about ways in which to make the full simulation more efficient by using a better neighbor-searching algorithm. Given sufficient time, the implementation of a grid-cell based neighbor-searching algorithm could be a useful programming exercise.

Thus, though the students were saved much of the grunt work involved in writing the code, they had to become fairly familiar with its workings to receive full credit. The code is fairly well commented except for the deliberate omissions regarding *cp* and the routine in which the depth-first search is called. Most students were able to do this assignment without much assistance after being provided with the above details on the physical system and the overview of the code contained in the README file.

Student response

Despite the grumbling that ensues when students are prodded out of their comfort zones, most of them were able to complete these assignments without a great deal of difficulty. In general, we noticed that the qualitative and open-ended questions proved more taxing than actually solving the differential equations or running the agent-based simulations. Most of the students were interested in biological applications but had not encountered many at this stage of their education. This,

coupled with the novelty of agent-based modeling, added to interest in the tissue engineering example. Though the students would have learned more through coding the entire lattice model themselves, we feel reasonably satisfied that they have been exposed to using and modifying a relatively large piece of code, an accomplishment in itself, in the short span of a couple of weeks. If one wished to use this example in a course with more programming, the project could be extended by having students implement a more efficient neighbor searching algorithm or modify the code further to reproduce some of Shea and Pasquini's other results,[11] such as those for multiple cell types invading the scaffold or for cells initially being seeded throughout the scaffold.

Acknowledgment

This work was partially supported by the National Science Foundation (CTS-0302428).

Biographical sketches

Manohar Murthi earned his BS degree in chemical engineering from the University of California, Berkeley and his MS and PhD degrees from Northwestern University. He is currently employed as a quantitative researcher by a proprietary trading firm.

Lonnie D. Shea is Associate Professor of Chemical and Biological Engineering at Northwestern University. He holds BS and MS degrees in chemical engineering from Case Western Reserve University and a PhD in chemical engineering and scientific computing from the University of Michigan. His research laboratory focuses on the combination of biomaterials and gene/drug delivery for regenerative medicine.

Randall Q. Snurr is Professor of Chemical and Biological Engineering at Northwestern University. He holds BSE and PhD degrees in chemical engineering from the University of Pennsylvania and the University of California, Berkeley, respectively. His research interests include molecular simulation, development of new materials, diffusion in nanoporous materials, adsorption, catalysis, and energy storage.

References

[1] N. C. Craig *J. Chem. Educ.*, **2001**, *78*, 582.

[2] S.H. Carr, Engineering First at Northwestern University; In *Proceedings of the International Conference on Curricular Change in Engineering Education*, 2001.

[3] J. M. Ottino, *AIChE J.*, **2005**, *51*, 1840.

[4] M. M. Francl, *Ann. Rep. Comp. Chem.*, **2005**, *51*, 1840.

[5] R. M. Baldwin, J.F. Ely, J.D. Way, S.R. Daniel, *Chem. Eng. Educ.*, **2000**, *34*, 162.

[6] S. Middleman, *An Introduction to Mass and Heat Transfer*, John Wiley and Sons, New York, 1998.

[7] D.N. Theodorou, R.Q. Snurr, A.T. Bell; Molecular dynamics and diffusion in microporous materials; In G. Alberti and T. Bein, Ed., *Solid-state Supramolecular Chemistry: Two- and Three-Dimensional Networks*, Vol. 7 of Comprehensive Supramolecular Chemistry, Chapter 18, pages 507-548 Pergamon, Oxford, 1996.

[8] M.J. Sanborn, R.Q. Snurr, *AIChE J.*, **2001**, *47*, 2032.

[9] U. Willensky, *NetLogo Ants Model*, Center for Connected Learning and Computer-Based Modeling, Northwestern University, 1998.

[10] D. Knuth, *The Art of Computer Programming Volume 3: Sorting and Searching*, Addison Wesley, Reading, MA, 1998.

[11] L.D. Shea, B. Pasquini, unpublished results.

[12] Y. Lee, S. Kouvroukoglou, L.V. McIntire, K.A. Zygourakis, *Biophysics J.*, **1995**, *69*, 1248.

[13] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 2nd Edition, MIT Press and McGraw-Hill, Cambridge, MA, 2001.