Modeling Blood Cell Concentration in a Dialysis Cartridge

by

Kathleen Haas

A Project Report

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Industrial Mathematics

April 2010

APPROVED:

Professor Burt S. Tilley, Project Advisor

Professor Bogdan Vernescu, Head of Department

Contents

List of Figures

1 Introduction

1.1 Kidneys and Dialysis

Kidneys are organs that, among other things, remove waste and excess fluid from a person's body. Healthy kidneys are able to remove from one's blood substances like urea and creatinine, which would cause illness if allowed to build up in one's body. They also pass excess fluid along to the bladder, which is then expelled as urine. Additionally, kidneys regulate blood pressure and the amount of substances like sodium and potassium in one's body. When a patient's kidneys fail, the patient may feel unwell, and since liquid cannot be removed from his body normally, the patient can suffer from swelling of the ankles, excess fluid in his lungs, and hypertension. [1] [5]

Kidney dialysis is a process that performs the work of kidneys that cannot function normally. In 1943, Willem Kolff first performed dialysis, developing a mechanism to remove urea from a patient by using cellophane as a filter, with blood on one side, and a solution called dialysate on the other. Urea passed through the filter by diffusion and into the dialysate. [4]

During the process of modern hemodialysis, blood from the patient's body is run through very thin hollow fibers made of a semi-permeable membrane inside a dialyzer, shown in Figure 1 [6]. Dialysate is also run through the dialyzer outside the hollow fibers, carrying substances such as sodium, potassium, calcium, magnesium, chloride, bicarbonate, and glucose. During this process, the filtered toxins and excess water travel out of the blood and through the membrane, by both diffusion and convection, and into the dialysate. The blood inside the fibers is run countercurrent to the dialysate in order to maximize the gradient of the concentration of solutes across the membrane. The concentration of these substances is prescribed so that concentrations of these substances in the patient's body are as normal as possible after treatment. [4]

The pores in the membrane are large enough to allow water and waste through, but small enough to prevent blood cells from passing through. The process itself usually lasts three to five hours,

and is usually performed two or three times per week. [5]

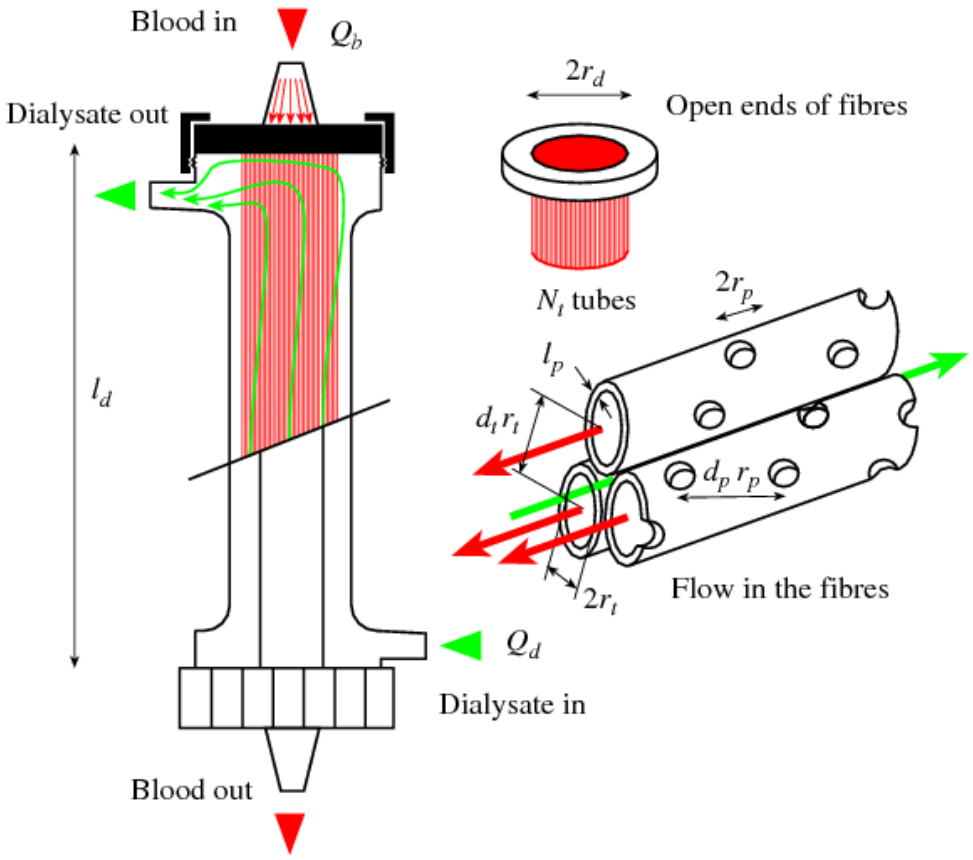

Figure 1: Example dialyzer through which blood and dialysate flow (left), and hollow fibers through which blood flows and around which dialysate flows (right)

After undergoing dialysis treatment, however, patients may experience a deficiency or an excess of solutes like sodium in their bloodstream. A deficiency of sodium can lead to low blood pressure, and an excess of sodium can cause fluid retention. [6]

1.2 Problem

In this report, we attempt to model the velocities and pressures of blood plasma and dialysate in a dialyzer, as well as the concentration of blood cells, as described in [6]. We look at the blood plasma in one hollow fiber and the dialysate flowing outside it. The hope is that knowledge of where blood cells are concentrated can give a better understanding of where ions are not lo-

cated. The ultimate goal is to model the flow of sodium during the process of dialysis, so that it can eventually be determined what rate of blood flow, rate of dialysate flow, amount of liquid removed, and concentration of solutes in the dialysate should be used in order to best treat a patient.

In the Section 2, we describe equations that model the velocities and pressures of blood and dialysate. We assume flow of blood plasma and dialysate to be Stokes flow, we assume that the pressure gradient varies only in the axial direction, and we model blood cell concentration as a continuous scalar field. The equations modeling velocities and pressures are nondimensionalized and solved, and the resulting velocity profiles are used in a nondimensionalized advection-diffusion equation that is used to model the blood cell concentration within the fibers inside the dialyzer. We found that varying the pressure in the dialysate moved the location at which the maximum blood cell concentration was achieved, and that the maximum blood cell concentration is directly proportional to the product of the blood cell Peclet number and the permeability of the membrane. In the Appendix, we describe the finite-difference methods used to numerically solve diffusion and advection-diffusion equations, namely Crank-Nicolson and Alternating Direction Implicit (ADI).

2 Model Formulation

Consider the configuration from Figure 2. For simplicity, we use Cartesian coordinates rather than cylindrical coordinates. Blood flows in the region bounded by $0 \leq x \leq l_d, 0 \leq z \leq 2r_t$ symmetrically about the line $z = r_t$. The dialysate flows in the region bounded by $0 \leq x \leq l_d, -2r_t \leq z \leq 0$ symmetrically about the line $z = -r_t$. Here, l_d is the length of a hollow fiber in the dialyzer, and r_t is the radius.

The velocities and pressures of the blood plasma and dialysate can be described by the Navier-Stokes equations:

$$\frac{\partial u_1}{\partial x} + \frac{\partial w_1}{\partial z} = 0, \quad 0 \leq z \leq r_t, \tag{1}$$

$$\rho_1\left(\frac{\partial u_1}{\partial t} + (\mathbf{u_1} \cdot \nabla)u_1\right) = -\frac{\partial p_1}{\partial x} + \mu_1 \nabla^2 u_1, \quad 0 \leq z \leq r_t, \tag{2}$$

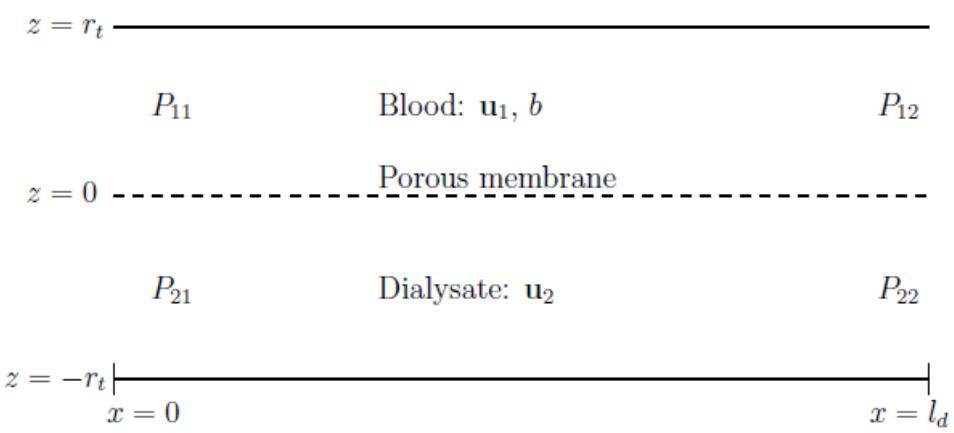

Figure 2: Domain of the problem, where the region $0 \leq x \leq l_d, 0 \leq z \leq r_t$ corresponds to the fiber through which blood flows, and the region where $0 \leq x \leq l_d, -r_t \leq z \leq 0$ corresponds to the area outside the fiber, where the dialysate flows

$$\rho_1(\frac{\partial w_1}{\partial t} + (\mathbf{u_1} \cdot \nabla)w_1) = -\frac{\partial p_1}{\partial z} + \mu_1 \nabla^2 w_1, \quad 0 \leq z \leq r_t, \tag{3}$$

$$\frac{\partial u_2}{\partial x} + \frac{\partial w_2}{\partial z} = 0, \quad -r_t \leq z \leq 0, \tag{4}$$

$$\rho_2(\frac{\partial u_2}{\partial t} + (\mathbf{u_2} \cdot \nabla)u_2) = -\frac{\partial p_2}{\partial x} + \mu_2 \nabla^2 u_2, \quad -r_t \leq z \leq 0, \tag{5}$$

$$\rho_2(\frac{\partial w_2}{\partial t} + (\mathbf{u_2} \cdot \nabla)w_2) = -\frac{\partial p_2}{\partial z} + \mu_2 \nabla^2 w_2, \quad -r_t \leq z \leq 0. \tag{6}$$

Here, $\mu_1, \rho_1, p_1, u_1,$ and w_1 are the effective dynamic viscosity, effective density, pressure, x-component of velocity, and z-component of velocity of blood, respectively. The variables $\mu_2, \rho_2, p_2, u_2,$ and w_2 are the effective dynamic viscosity, effective density, pressure, x-component of velocity, and z-component of velocity of dialysate, respectively. For simplicity, we assume that $\rho_1 = \rho_2$, $\mu_1 = \mu_2$, and are constant.

These equations are subject to the boundary conditions:

$$\frac{\partial u_1}{\partial z} = w_1 = 0, \quad z = r_t, \tag{7}$$

$$\frac{\partial u_2}{\partial z} = w_2 = 0, \quad z = -r_t, \tag{8}$$

$$u_1 = u_2 = 0, \quad z = 0, \tag{9}$$

Variable	Typical value
r_t	10^{-4}m
l_d	0.2 m
U_b	0.0018 m/s
μ	4×10^{-3}Pa $\cdot$ s
D	10^{-9}m^2/s

Table 1: Typical values of r_t, l_d, U_b, μ, D

$$w_1 = w_2 = \frac{k}{\mu}\frac{p_2 - p_1}{l_p}, \qquad z = 0. \tag{10}$$

where k is the permeability of the membrane and l_p is the thickness of the membrane. Additionally, the values of p_1 and p_2 are prescribed at the boundaries $x = 0, x = l_d$ to be the values $P_{11}, P_{12}, P_{21}, P_{22}$, as shown in Figure 2.

The concentration of blood cells, b, can be modelled by the following equation:

$$\frac{\partial b}{\partial t} + u_1\frac{\partial b}{\partial x} + w_1\frac{\partial b}{\partial z} = D\left(\frac{\partial^2 b}{\partial x^2} + \frac{\partial^2 b}{\partial z^2}\right), \qquad 0 \le z \le r_t, \tag{11}$$

where D is the diffusion coefficient of the blood cells. Equation 11 is subject to the boundary conditions of no flux along $z = 0$ and $z = r_t$, a known concentration along $x = 0$, and a concentration independent of x at $x = l_d$, i.e.,

$$\begin{aligned} w_1 b - D\frac{\partial b}{\partial z} &= 0, \qquad z = 0, z = r_t, \\ b &= f(z,t), \qquad x = 0, \\ \frac{\partial b}{\partial x} &= 0, \qquad x = l_d. \end{aligned} \tag{12}$$

Table 1 shows some typical values of r_t, l_d, U_b (an average velocity of the blood), μ, and D [6]. We apply the same scalings as in [6], where a tilde is used to denote a nondimensional variable:

$$x = l_d\tilde{x}\ ,\ z = r_t\tilde{z}\ ,\ \epsilon = \frac{r_t}{l_d}\ ,\ u = U_b\tilde{u}\ ,\ w = \epsilon U_b\tilde{w}\ ,\ p = \frac{\mu U_b}{r_t\epsilon}\tilde{p}\ ,\ b = b_0\tilde{b}\ ,\ t = \frac{r_t^2}{D}\tilde{t}. \tag{13}$$

These scalings, applied to (1) - (6), give:

$$\frac{\partial \tilde{u_1}}{\partial \tilde{x}} + \frac{\partial \tilde{w_1}}{\partial \tilde{z}} = 0, \tag{14}$$

$$\epsilon^2 \mathrm{Re}(\frac{\partial \tilde{u_1}}{\partial \tilde{t}} + (\tilde{\mathbf{u_1}} \cdot \nabla)\tilde{u_1}) = -\frac{\partial \tilde{p_1}}{\partial \tilde{x}} + \epsilon^2 \frac{\partial^2 \tilde{u_1}}{\partial \tilde{x}^2} + \frac{\partial^2 \tilde{u_1}}{\partial \tilde{z}^2}, \tag{15}$$

$$\epsilon^4 \mathrm{Re}(\frac{\partial \tilde{w_1}}{\partial \tilde{t}} + (\tilde{\mathbf{u_1}} \cdot \nabla)\tilde{w_1}) = -\frac{\partial \tilde{p_1}}{\partial \tilde{z}} + \epsilon^2 \left(\epsilon^2 \frac{\partial^2 \tilde{w_1}}{\partial \tilde{x}^2} + \frac{\partial^2 \tilde{w_1}}{\partial \tilde{z}^2} \right), \tag{16}$$

$$\frac{\partial \tilde{u_2}}{\partial \tilde{x}} + \frac{\partial \tilde{w_2}}{\partial \tilde{z}} = 0, \tag{17}$$

$$\epsilon^2 \mathrm{Re}(\frac{\partial \tilde{u_2}}{\partial \tilde{t}} + (\tilde{\mathbf{u_2}} \cdot \nabla)\tilde{u_2}) = -\frac{\partial \tilde{p_2}}{\partial \tilde{x}} + \epsilon^2 \frac{\partial^2 \tilde{u_2}}{\partial \tilde{x}^2} + \frac{\partial^2 \tilde{u_2}}{\partial \tilde{z}^2}, \tag{18}$$

$$\epsilon^4 \mathrm{Re}(\frac{\partial \tilde{w_2}}{\partial \tilde{t}} + (\tilde{\mathbf{u_2}} \cdot \nabla)\tilde{w_2}) = -\frac{\partial \tilde{p_2}}{\partial \tilde{z}} + \epsilon^2 \left(\epsilon^2 \frac{\partial^2 \tilde{w_2}}{\partial \tilde{x}^2} + \frac{\partial^2 \tilde{w_2}}{\partial \tilde{z}^2} \right), \tag{19}$$

where $\mathrm{Re} = \dfrac{\rho U_b l_d}{\mu}$ is the Reynolds number of the blood and dialysate flow.

The scaled boundary conditions are given by:

$$\frac{\partial \tilde{u_1}}{\partial \tilde{z}} = \tilde{w_1} = 0, \quad \tilde{z} = 1, \tag{20}$$

$$\frac{\partial \tilde{u_2}}{\partial \tilde{z}} = \tilde{w_2} = 0, \quad \tilde{z} = -1, \tag{21}$$

$$\tilde{u_1} = \tilde{u_2} = 0, \quad \tilde{z} = 0, \tag{22}$$

$$\tilde{w_1} = \tilde{w_2} = K(p_2 - p_1), \quad \tilde{z} = 0, \tag{23}$$

where $K = \dfrac{k}{r_t l_p \epsilon^2}$.

Applying the scalings to (11) gives:

$$\frac{D}{r_t^2}\frac{\partial \tilde{b}}{\partial \tilde{t}} + \frac{U_b}{l_d}\tilde{u_1}\frac{\partial \tilde{b}}{\partial \tilde{x}} + \frac{\epsilon U_b}{r_t}\tilde{w_1}\frac{\partial \tilde{b}}{\partial \tilde{z}} = D\left(\frac{1}{l_d^2}\frac{\partial^2 \tilde{b}}{\partial \tilde{x}^2} + \frac{1}{r_t^2}\frac{\partial^2 \tilde{b}}{\partial \tilde{z}^2} \right), \tag{24}$$

or equivalently,

$$\frac{\partial \tilde{b}}{\partial t} + \epsilon^2 \text{ Pe } \tilde{u_1}\frac{\partial \tilde{b}}{\partial \tilde{x}} + \epsilon^2 \text{ Pe } \tilde{w_1}\frac{\partial \tilde{b}}{\partial \tilde{z}} = \epsilon^2 \frac{\partial^2 \tilde{b}}{\partial \tilde{x}^2} + \frac{\partial^2 \tilde{b}}{\partial \tilde{z}^2}, \tag{25}$$

where $\text{Pe} = \dfrac{U_b l_d}{D}$ is the Peclet number. The scaled boundary conditions are:

$$\begin{aligned} \epsilon^2 \text{ Pe } \tilde{w}\tilde{b} &= \frac{\partial \tilde{b}}{\partial \tilde{z}}, \qquad \tilde{z} = 0, \tilde{z} = 1, \\ \tilde{b} &= f(z,t)/b_0, \qquad x = 0, \\ \frac{\partial \tilde{b}}{\partial \tilde{x}} &= 0, \qquad x = 1. \end{aligned} \tag{26}$$

From here on, we drop the tildes from our variables and assume that every variable is nondimensional.

Since $\epsilon \ll 1$ and $\text{Re} \sim O(1)$, (16) and (19) give us that

$$\frac{\partial p_1}{\partial z} = \frac{\partial p_2}{\partial z} = 0, \tag{27}$$

implying that pressure is a function only of x. Equations (15) and (18) also give that

$$-\frac{\partial p_1}{\partial x} + \frac{\partial^2 u_1}{\partial z^2} = -\frac{\partial p_2}{\partial x} + \frac{\partial^2 u_2}{\partial z^2} = 0. \tag{28}$$

This, combined with the boundary conditions for u, gives:

$$\begin{aligned} u_1 &= \frac{dp_1}{dx}\left(\frac{z^2}{2} - z\right), \qquad 0 \le z \le 1, \\ u_2 &= \frac{dp_2}{dx}\left(\frac{z^2}{2} + z\right), \qquad -1 \le z \le 0. \end{aligned} \tag{29}$$

Equations (14) and (17), along with (20) and (21), give:

$$\begin{aligned} w_1 &= \frac{d^2 p_1}{dx^2}\left(\frac{-z^3}{6} + \frac{z^2}{2} - \frac{1}{3}\right), \qquad 0 \le z \le 1, \\ w_2 &= \frac{d^2 p_2}{dx^2}\left(\frac{-z^3}{6} - \frac{z^2}{2} + \frac{1}{3}\right), \qquad -1 \le z \le 0. \end{aligned} \tag{30}$$

Applying (23) gives the equations for the pressures of the blood and dialysate as:

$$\begin{aligned} \frac{d^2p_1}{dx^2} &= 3K(p_1 - p_2), \quad 0 \leq z \leq 1, \\ \frac{d^2p_2}{dx^2} &= 3K(p_2 - p_1), \quad -1 \leq z \leq 0. \end{aligned} \tag{31}$$

The pressures, then, are given by:

$$\begin{aligned} p_1(x) &= Ae^{\sqrt{6K}x} + Be^{\sqrt{-6K}x} + ax + b, \\ p_2(x) &= Ae^{\sqrt{6K}x} + Be^{\sqrt{-6K}x} + ax + b, \end{aligned} \tag{32}$$

where the values of A, B, a, b are given by:

$$\begin{aligned} A &= -\frac{P_{11}e^{-\sqrt{6K}} - P_{21}e^{-\sqrt{6K}} + P_{22} - P_{12}}{2(e^{\sqrt{6K}} - e^{-\sqrt{6K}})}, \\ B &= -\frac{P_{11}e^{\sqrt{6K}} - P_{21}e^{\sqrt{6K}} + P_{22} - P_{12}}{2(e^{\sqrt{6K}} - e^{-\sqrt{6K}})}, \\ a &= \frac{P_{22} + P_{12} - P_{21} - P_{11}}{2}, \\ b &= \frac{P_{21} + P_{11}}{2}. \end{aligned} \tag{33}$$

We can now substitute the pressures from (32) into (29) and (30) in order to find the values of u_1, u_2, w_1, and w_2. Figure 3 shows an example of streamlines of the velocity of blood when $K = 0.01, P_{11} = 815, P_{12} = 796, P_{21} = 472$, and $P_{22} = 518$.

The values of these velocities can then be used in Equation 11 to find a solution for the blood cell concentration.

3 Results

Applying the velocities in Figure 3 to equation 11, we can find a steady state solution for the blood cell concentration by solving for the concentration at each timestep until the difference between the solution at time t_n and the solution at t_{n+1} is within some tolerance. A solution, given an initial concentration of 1 everywhere, and given that $Pe = 9000$, $\epsilon = 5 \times 10^{-4}$ and $K = 0.01$, is given in Figure 4.

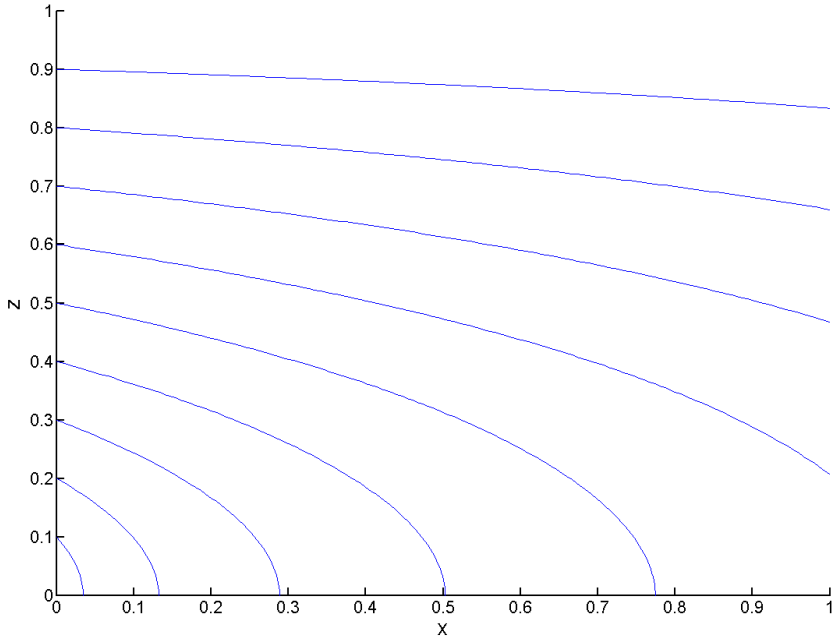

Figure 3: Streamlines of the velocity of blood when $K = 0.01, P_{11} = 815, P_{12} = 796, P_{21} = 472$, and $P_{22} = 518$

Figure 5 shows the value of the maximum blood cell concentration at steady state, b_{max}, within our domain, for varying values of Pe and K. This figure shows that when Pe $= 0$, or when $K = 0$, the solution remains constant at 1, and increasing Pe or K results in an increase in b_{max}. Figure 6 shows the value of b_{max} for varying values of PeK. This figure shows that b_{max} is directly proportional to the product of Pe and K. For all of these cases, b_{max} is achieved at the point $x = 1, z = 0$.

To change the locus of the maximum, we change the pressure in the dialysate. If $\Delta p = P_{22} - P_{21}$, Figure 7 shows the value of b_{max} for varying values of Δp, and Figure 8 shows the x-coordinate at which b_{max} is achieved, for varying values of Δp. The z-coordinate at which b_{max} is achieved remains at $z = 0$. As Δp increases, the maximum concentration decreases, and the locus of the maximum moves to the left.

Figure 9 shows a solution for the blood cell concentration when Pe=10000, $K = 0.01$, and $\Delta p = 450$. Here, by varying the pressure in the dialysate, the maximum concentration has been

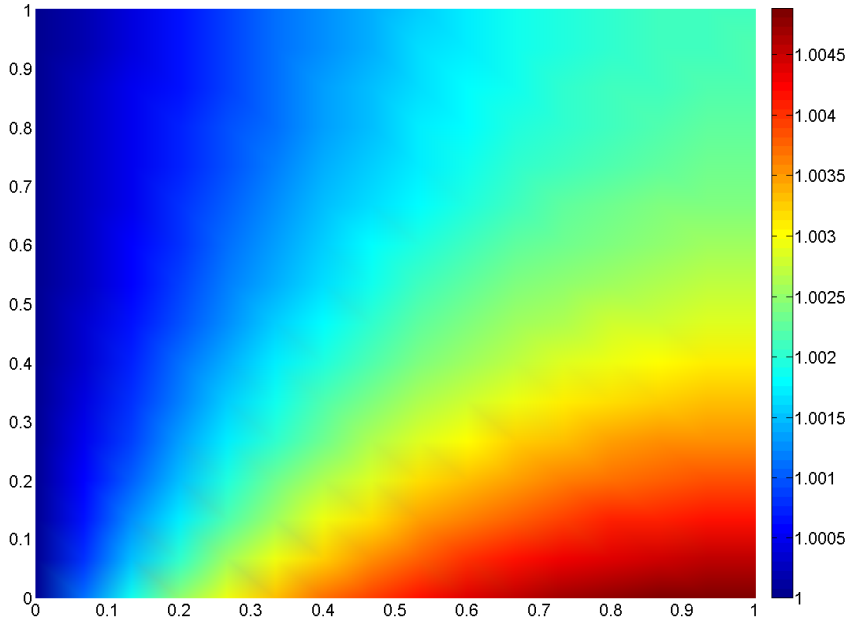

Figure 4: Blood cell concentration given that $Pe = 9000$, $\epsilon = 5 \times 10^{-4}$ and $K = 0.01$

moved from $x = 1$ to around $x = 0.3$.

Figure 10 shows the difference between p_1 and p_2 at the x-coordinate at which b_{max} is achieved, for varying values of Δp. A typical value for Δp is around 45. Here, we see that as Δp increases and moves further away from the typical value, the difference in pressure across the membrane decreases.

4 Conclusions

We were able to find solutions for blood cell concentrations, given values of Pe, $K, p_{11}, p_{12}, p_{21}, p_{22}$. We found that, holding pressures constant at typical values, $b_{max} = aPeK$, for some constant $a > 0$.

For typical values of the parameters, the maximum blood cell concentration occurs at the point $x = 1, z = 0$. We also found that by increasing the value of Δp in the dialysate, we can move

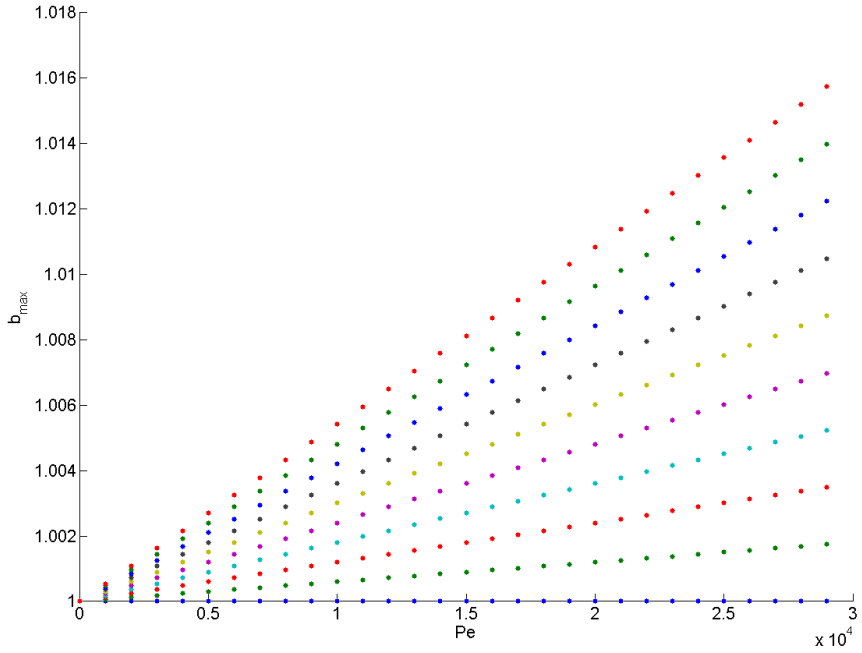

Figure 5: Maximum blood cell concentration vs. Pe. Different colored lines correspond to different values of K, ranging from 10^{-8} to 0.01, where lines of increasing slope correspond to higher values of K. This shows that when Pe $= 0$, or when $K = 0$, the solution remains constant at 1, and increasing Pe or K results in an increase in b_{max}.

the point at which the maximum blood cell concentration is achieved. However, control over this locus comes at a cost. Increasing the value of Δp results in a smaller pressure difference across the membrane, which results in less fluid being removed from the blood. Further study would be required to determine the best values of parameters like prsesure to best remove fluid from a patient and control the concentrations of substances like sodium to minimize adverse health effects.

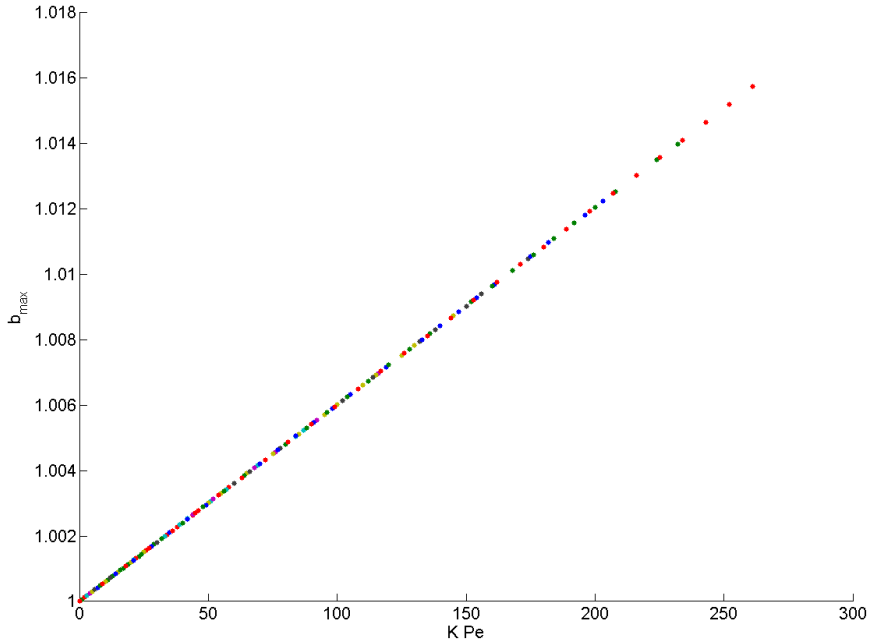

Figure 6: Maximum blood cell concentration vs. KPe, showing that $b_{max} = aKPe$ for some constant a.

Appendices

We now describe the methods of Alternating Direction Implicit (ADI) and Crank-Nicolson, and show how they can be used to solve the diffusion and advection-diffusion equations.

A ADI

ADI, or Alternating Direction Implicit, is a finite-difference method which can be used to solve the advection-diffusion equation. The method involves two steps for each timestep. One step is implicit in the x-direction and explicit in the z-direction, and one step is explicit in the x-direction and implicit in the z-direction. For example, we approximate the solution, $c(t, x, z)$ to the equation

$$\frac{\partial c}{\partial t} + u(z)\frac{\partial c}{\partial x} = \alpha\left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial z^2}\right), \tag{34}$$

where $u(z)$ is the solvent velocity, c is the solute concentration, and α is the diffusion coefficient. The second term on the left hand side (LHS) of Equation 34 is the advective term, and the term

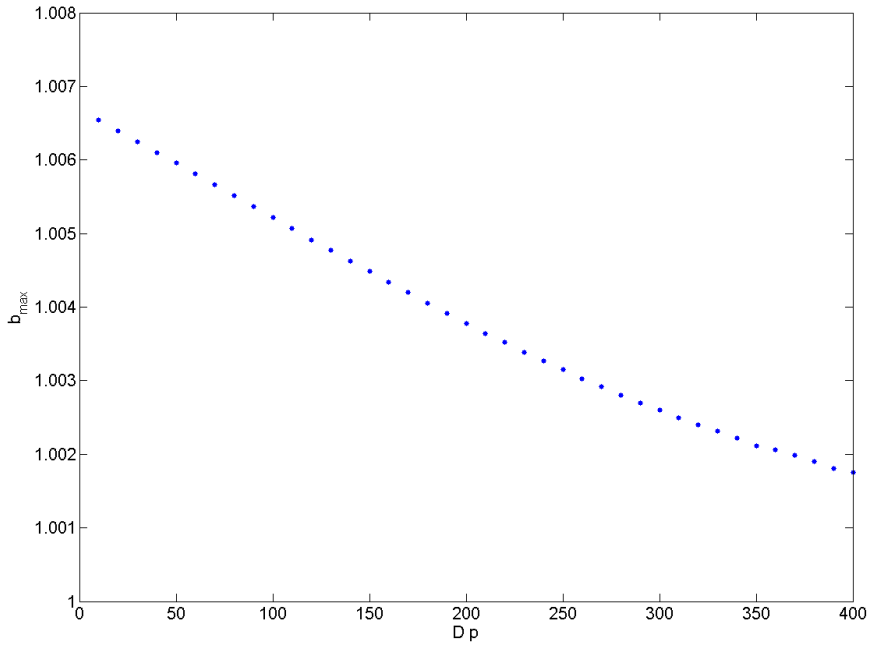

Figure 7: Maximum blood cell concentration vs. Δp, showing that b_{max} decreases and tends towards 1 as Δp increases.

on the right hand side (RHS) is the diffusion term. The following finite-difference approximation is used:

Step 1:

$$\frac{c_{i,j}^{n+1/2} - c_{i,j}^{n}}{\Delta t/2} + u(z)|_j \, \frac{c_{i,j}^{n+1/2} - c_{i-1,j}^{n+1/2}}{\Delta x} = \alpha \left[\frac{c_{i+1,j}^{n+1/2} - 2c_{i,j}^{n+1/2} + c_{i-1,j}^{n+1/2}}{\Delta x^2} + \frac{c_{i,j+1}^{n} - 2c_{i,j}^{n} + c_{i,j-1}^{n}}{\Delta z^2} \right], \tag{35}$$

Step 2:

$$\frac{c_{i,j}^{n+1} - c_{i,j}^{n+1/2}}{\Delta t/2} + u(z)|_j \, \frac{c_{i,j}^{n+1/2} - c_{i-1,j}^{n+1/2}}{\Delta x} = \alpha \left[\frac{c_{i+1,j}^{n+1/2} - 2c_{i,j}^{n+1/2} + c_{i-1,j}^{n+1/2}}{\Delta x^2} + \frac{c_{i,j+1}^{n+1} - 2c_{i,j}^{n+1} + c_{i,j-1}^{n+1}}{\Delta z^2} \right], \tag{36}$$

where $c_{i,j}^{n} = c(t_n, x_i, z_j)$, Δt is the timestep, Δx is the grid spacing in the x-direction, and Δz is the grid spacing in the z-direction. Moving all terms evaluated at time $t_{n+1/2} = t_n + \Delta t/2$ to the left and all terms evaluated at time t_n to the right, Step 1 becomes:

$$\frac{c_{i,j}^{n+1/2}}{\Delta t/2} + u(z)|_j \, \frac{c_{i,j}^{n+1/2} - c_{i-1,j}^{n+1/2}}{\Delta x} - \alpha \frac{c_{i+1,j}^{n+1/2} - 2c_{i,j}^{n+1/2} + c_{i-1,j}^{n+1/2}}{\Delta x^2} = \frac{c_{i,j}^{n}}{\Delta t/2} + \alpha \frac{c_{i,j+1}^{n} - 2c_{i,j}^{n} + c_{i,j-1}^{n}}{\Delta z^2} \tag{37}$$

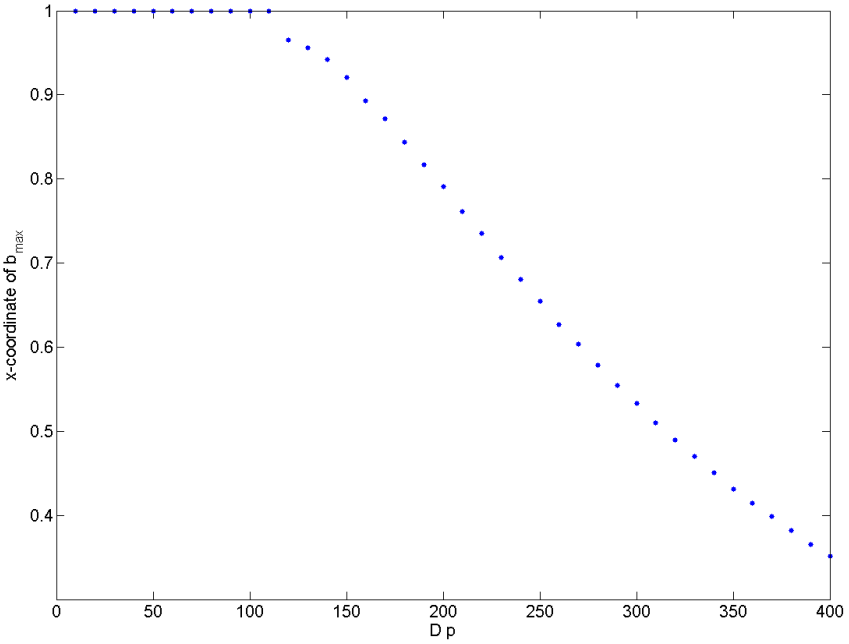

Figure 8: x-coordinate at which b_{max} is achieved, for varying values of Δp, showing that the locus b_{max} can be controlled by varying the dialysate pressure.

or equivalently,

$$\left(-\frac{u(z)|_j}{\Delta x}-\frac{\alpha}{\Delta x^2}\right)c_{i-1,j}^{n+1/2}+\left(\frac{1}{\Delta t/2}+\frac{2\alpha}{\Delta x^2}+\frac{u(z)|_j}{\Delta x}\right)c_{i,j}^{n+1/2}-\frac{\alpha}{\Delta x^2}c_{i+1,j}^{n+1/2}=$$

$$=\frac{\alpha}{\Delta z^2}c_{i,j-1}^{n}+\left(\frac{1}{\Delta t/2}-\frac{2\alpha}{\Delta z^2}\right)c_{i,j}^{n}+\frac{\alpha}{\Delta z^2}c_{i,j+1}^{n}$$

Similarly, Step 2 becomes:

$$\frac{c_{i,j}^{n+1}}{\Delta t/2}-\alpha\frac{c_{i,j+1}^{n+1}-2c_{i,j}^{n+1}+c_{i,j-1}^{n+1}}{\Delta z^2}=\frac{c_{i,j}^{n+1/2}}{\Delta t/2}-u(z)|_j\frac{c_{i,j}^{n+1/2}-c_{i-1,j}^{n+1/2}}{\Delta x}+\alpha\frac{c_{i+1,j}^{n+1/2}-2c_{i,j}^{n+1/2}+c_{i-1,j}^{n+1/2}}{\Delta x^2} \quad (38)$$

or equivalently,

$$-\frac{\alpha}{\Delta z^2}c_{i,j-1}^{n+1}+\left(\frac{1}{\Delta t/2}+\frac{2\alpha}{\Delta z^2}\right)c_{i,j}^{n+1}-\frac{\alpha}{\Delta z^2}c_{i,j+1}^{n+1}=\left(\frac{u(z)|_j}{\Delta x}+\frac{\alpha}{\Delta x^2}\right)c_{i-1,j}^{n+1/2}+$$

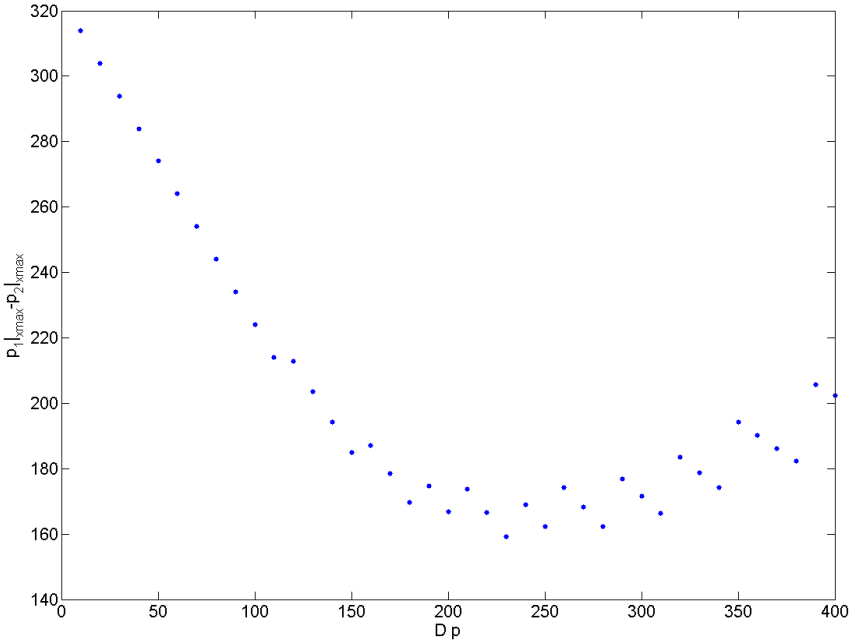

Figure 9: Blood cell concentration given that $Pe = 10000$, $\epsilon = 5 \times 10^{-4}$, $K = 0.01$, and $\Delta p = 450$. The locus of the maximum blood cell concentration has been moved from $x = 1$ to around $x = 0.3$.

$$+\left(\frac{1}{\Delta t/2} - \frac{2\alpha}{\Delta x^2} - \frac{u(z)|_j}{\Delta x}\right) c_{i,j}^{n+1/2} + \frac{\alpha}{\Delta x^2} c_{i+1,j}^{n+1/2}$$

Each step is performed by solving two systems of tridiagonal equations.

The domain of $0 \leq x \leq L, 0 \leq z \leq H$ is divided into a uniform 2D grid, the values of which are contained in a matrix. For example, if there are N points in the x-direction and M points in the z-direction, then

$$c = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \cdots & \vdots \\ c_{M1} & c_{M2} & \cdots & c_{MN} \end{bmatrix} \tag{39}$$

If we impose the boundary conditions:

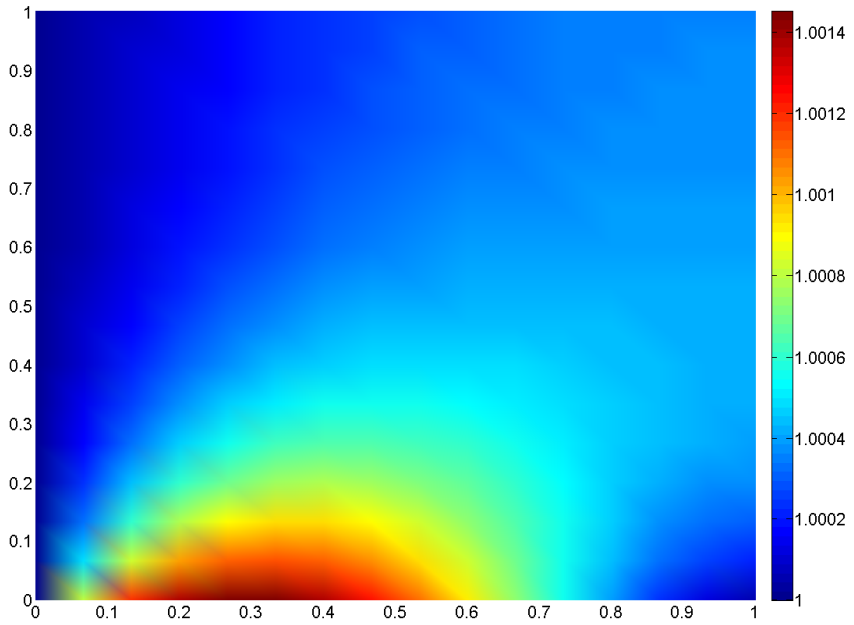

Figure 10: Difference between p_1 and p_2 at the x-coordinate at which b_{max} is achieved, for varying values of Δp. A typical value for Δp is 45. As Δp increases and moves further away from the typical value, the difference in pressure across the membrane decreases.

$$\begin{aligned} c_x &= 0 \qquad x = 0, 1 \\ c_z &= 0 \qquad z = 0, 1 \end{aligned} \tag{40}$$

we use image points to calculate derivatives at the boundaries:

$$\begin{aligned} \frac{c_{i2} - c_{i0}}{2\Delta x} &= 0 \Rightarrow c_{i2} = c_{i0} \\ \frac{c_{i,N+1} - c_{i,N-1}}{2\Delta x} &= 0 \Rightarrow c_{i,N+1} = c_{i,N-1} \\ \frac{c_{0j} - c_{2j}}{2\Delta z} &= 0 \Rightarrow c_{0j} = c_{2j} \\ \frac{c_{M-1,j} - c_{M+1,j}}{2\Delta z} &= 0 \Rightarrow c_{M-1,j} = c_{M+1,j} \end{aligned} \tag{41}$$

We can now use the approximations:

$$\begin{aligned}
\frac{\partial^2 c_{i1}}{\partial x^2} &= \frac{2c_{i2} - 2c_{i1}}{\Delta x^2} \\
\frac{\partial^2 c_{iN}}{\partial x^2} &= \frac{2c_{i,N-1} - 2c_{iN}}{\Delta x^2} \\
\frac{\partial c_{i1}}{\partial x} &= \frac{c_{i1} - c_{i0}}{2\Delta x} \\
\frac{\partial^2 c_{1j}}{\partial z^2} &= \frac{2c_{2j} - 2c_{1j}}{\Delta z^2} \\
\frac{\partial^2 c_{Mj}}{\partial z^2} &= \frac{2c_{M-1,j} - 2c_{Mj}}{\Delta z^2}
\end{aligned} \tag{42}$$

To take derivatives in the x-direction at time n, the rows of c are concatenated, and the matrices D_x^2 and D_x are multiplied by the resulting vector, c^n, to obtain the second and first derivatives in x, respectively. The first derivative is found using upwinding. [2]

$$D_x^2 \cdot c^n = \begin{bmatrix}
-2 & 2 & & & & & & & \\
1 & -2 & 1 & & & & & & \\
 & & \ddots & & & & & & \\
 & & 1 & -2 & 1 & & & & \\
 & & & 2 & -2 & & & & \\
 & & & & & -2 & 2 & & \\
 & & & & & & \ddots & & \\
 & & & & & & & 2 & -2
\end{bmatrix} \begin{bmatrix} c_{11} \\ c_{12} \\ \vdots \\ c_{1N-1} \\ c_{1N} \\ c_{21} \\ \vdots \\ c_{MN} \end{bmatrix} \tag{43}$$

$$D_x \cdot c^n = \begin{bmatrix} 1 & -1 & & & & & & & \\ -1 & 1 & & & & & & & \\ & & \ddots & & & & & & \\ & & & -1 & 1 & & & & \\ & & & 1 & -1 & & & & \\ & & & & & 1 & -1 & & \\ & & & & & & & \ddots & \\ & & & & & & & -1 & 1 \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{12} \\ \vdots \\ c_{1N-1} \\ c_{1N} \\ c_{21} \\ \vdots \\ c_{MN} \end{bmatrix} \tag{44}$$

To take derivatives in the z-direction at time n, the columns of c are concatenated, and the matrix D_z^2 is multiplied by the resulting vector, $c^{n\prime}$, to obtain the second derivative in z.

$$D_z^2 \cdot c^{n\prime} = \begin{bmatrix} -2 & 2 & & & & & & & \\ 1 & -2 & 1 & & & & & & \\ & & \ddots & & & & & & \\ & & 1 & -2 & 1 & & & & \\ & & & 2 & -2 & & & & \\ & & & & & -2 & 2 & & \\ & & & & & & & \ddots & \\ & & & & & & & 2 & -2 \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{21} \\ \vdots \\ c_{M-1\ 1} \\ c_{M1} \\ c_{12} \\ \vdots \\ c_{MN} \end{bmatrix} \tag{45}$$

U is a diagonal matrix containing, in the appropriate locations, the values of $u(z)$ from equation (34). If a vector v is of the form of c^n, the operation v' denotes permutation of v's entries into the form of $c^{n\prime}$. If a vector v is of the form of $c^{n\prime}$, the operation v' denotes permutation of v's entries into the form of c^n. U' is a matrix whose entries on the diagonal have been permuted so that the vector containing the diagonal elements are of the form of $c^{n\prime}$. Steps 1 and 2 of the ADI method are performed by solving the following equations [2]:

$$(I + U\frac{\Delta t}{2}D_x - \frac{\alpha\Delta t}{2}D_x^2)c^{n+1/2} = c^n + \frac{\alpha\Delta t}{2}(D_z^2 c^{n\prime})' \tag{46}$$

$$(I - \frac{\alpha\Delta t}{2}D_z^2)c^{n+1\prime} = c^{n+1/2\prime} + \frac{\alpha\Delta t}{2}(D_x^2 c^{n+1/2})' - \frac{\Delta t}{2}U'(D_x c^{n+1/2})' \tag{47}$$

B Crank Nicolson

Crank-Nicolson is a finite-difference scheme that can be used to solve the advection-diffusion equation. For example, to approximate the solution, $c(t, x, z)$ to Equation 34, the following finite-difference approximation is used, using upwinding for the advective term:

$$\begin{aligned}\frac{c_{ij}^{n+1} - c_{ij}^n}{\Delta t} + \frac{u(z)|j}{2}(\frac{c_{ij}^{n+1} - c_{i-1j}^{n+1}}{\Delta x} + \frac{c_{ij}^n - c_{i-1j}^n}{\Delta x}) = \frac{\alpha}{2}(\frac{c_{i+1j}^{n+1} - 2c_{ij}^{n+1} + c_{i-1j}^{n+1}}{\Delta x^2} + \\ \frac{c_{ij+1}^{n+1} - 2c_{ij}^{n+1} + c_{ij-1}^{n+1}}{\Delta z^2} + \frac{c_{i+1j}^n - 2c_{ij}^n + c_{i-1j}^n}{\Delta x^2} + \frac{c_{ij+1}^n - 2c_{ij}^n + c_{ij-1}^n}{\Delta z^2})\end{aligned} \tag{48}$$

Equivalently,

$$\begin{aligned}c_{ij}^{n+1} + \frac{u(z)|j\Delta t}{2}\frac{c_{ij}^{n+1} - c_{i-1j}^{n+1}}{\Delta x} - \frac{\alpha\Delta t}{2}(\frac{c_{i+1j}^{n+1} - 2c_{ij}^{n+1} + c_{i-1j}^{n+1}}{\Delta x^2} + \frac{c_{ij+1}^{n+1} - 2c_{ij}^{n+1} + c_{ij-1}^{n+1}}{\Delta z^2}) = \\ c_{ij}^n - \frac{u(z)|j\Delta t}{2}\frac{c_{ij}^n - c_{i-1j}^n}{\Delta x} + \frac{\alpha\Delta t}{2}(\frac{c_{i+1j}^n - 2c_{ij}^n + c_{i-1j}^n}{\Delta x^2} + \frac{c_{ij+1}^n - 2c_{ij}^n + c_{ij-1}^n}{\Delta z^2})\end{aligned} \tag{49}$$

where $c_{i,j}^n = c(t_n, z_i, x_j)$, Δt is the timestep, Δx is the grid spacing in the x-direction, and Δz is the grid spacing in the z-direction. The domain of $0 \leq x \leq L, 0 \leq z \leq H$ is divided into a uniform 2D grid, the values of which are contained in a matrix. For example, if there are N points in the x-direction and M points in the z-direction, then

$$c = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \cdots & \vdots \\ c_{M1} & c_{M2} & \cdots & c_{MN} \end{bmatrix} \tag{50}$$

To take derivatives at time n, the rows of c are concatenated, and the matrices D_x^2, D_x, and D_z^2

are multiplied by the resulting vector, c^n, to obtain the second and first derivatives in x, and the second derivative in z, respectively. The boundary conditions and use of image points are identical to those used in the previous section.

$$D_x^2 \cdot c^n = \begin{bmatrix} -2 & 2 & & & & & & & & \\ 1 & -2 & 1 & & & & & & & \\ & \ddots & & & & & & & & \\ & & 1 & -2 & 1 & & & & & \\ & & & 2 & -2 & & & & & \\ & & & & & -2 & 2 & & & \\ & & & & & & \ddots & & & \\ & & & & & & & 2 & -2 \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{12} \\ \vdots \\ c_{1N-1} \\ c_{1N} \\ c_{21} \\ \vdots \\ c_{MN} \end{bmatrix} \tag{51}$$

$$D_x \cdot c^n = \begin{bmatrix} 1 & -1 & & & & & & & & \\ -1 & 1 & & & & & & & & \\ & \ddots & & & & & & & & \\ & & & -1 & 1 & & & & & \\ & & & 1 & -1 & & & & & \\ & & & & & 1 & -1 & & & \\ & & & & & & \ddots & & & \\ & & & & & & & -1 & 1 \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{12} \\ \vdots \\ c_{1N-1} \\ c_{1N} \\ c_{21} \\ \vdots \\ c_{MN} \end{bmatrix} \tag{52}$$

$$D_z^2 \cdot c^n = \begin{bmatrix} -2 & 0 & \cdots & 0 & 2 & \cdots & & & & \\ 0 & -2 & 0 & \cdots & 0 & 2 & \cdots & & & \\ & & & \ddots & & & & & & \\ 1 & 0 & \cdots & 0 & -2 & 0 & \cdots & 0 & 1 & \cdots \\ & & & & & \ddots & & & & \\ & & & & \cdots & 2 & 0 & \cdots & 0 & -2 \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{12} \\ \vdots \\ c_{1N-1} \\ c_{1N} \\ c_{21} \\ \vdots \\ c_{MN} \end{bmatrix} \tag{53}$$

U is a diagonal matrix containing, in the appropriate locations, the values of $u(z)$ from equation (34). The Crank-Nicolson method is performed by solving the following equation [2]:

$$(I + U\frac{\Delta t}{2}D_x - \frac{\alpha \Delta t}{2}(D_x^2 + D_y^2))c^{n+1} = (I - U\frac{\Delta t}{2}D_x + \frac{\alpha \Delta t}{2}(D_x^2 + D_y^2))c^n \tag{54}$$

C 1D Diffusion

We now test both the Crank-Nicolson and ADI methods on the 1D and 2D diffusion and advection-diffusion equations. We solve for the analytical solutions of these equations using separation of variables and compare them with our numerical solutions. We show that as Δt goes to zero, the truncation error goes to zero in a way that is consistent with the method used.

Crank-Nicolson was used to solve the 1D diffusion equation

$$\frac{\partial c}{\partial t} = \alpha \frac{\partial^2 c}{\partial x^2} \text{ for } 0 \leq x \leq 1, \tag{55}$$

with Neumann boundary conditions

$$\frac{\partial c}{\partial x} = 0 \text{ on } x = 0, 1. \tag{56}$$

To verify the numerical solutions of this equation, we compared the L^2 norm of the analytical and

numerical solutions over time. To find the analytical solution, we applied separation of variables:

$$c(t,x) = T(t)X(x) \tag{57}$$

$$T'X = \alpha T X'' \tag{58}$$

$$\frac{T'}{T} = \alpha \frac{X''}{X} = -\lambda \tag{59}$$

This gives two separate equations,

$$T' + \lambda T = 0 \tag{60}$$

$$\alpha X'' + \lambda X = 0 \tag{61}$$

For simplicity, assume $\alpha = 1$. Equation becomes:

$$X(x) = a\cos(x\sqrt{\lambda}) + b\sin(x\sqrt{\lambda}) \tag{62}$$

Applying the boundary conditions, we find:

$$b = 0 \tag{63}$$

$$\lambda = (n\pi)^2, n = 1, 2, ... \tag{64}$$

$$\tag{65}$$

A solution to (55) is:

$$c(t,x) = ae^{-(n\pi)^2 t}\cos(n\pi x) \tag{66}$$

Setting α to 1, and given the initial conditions of $c(0,x) = \cos(n\pi x)$, the L^2 norm of the analytical solution is

$$|c(t,x)|_2 = \sqrt{\int_0^1 c^2(t,x)dx} = \sqrt{\frac{1}{2}}e^{-(n\pi)^2 t} \tag{67}$$

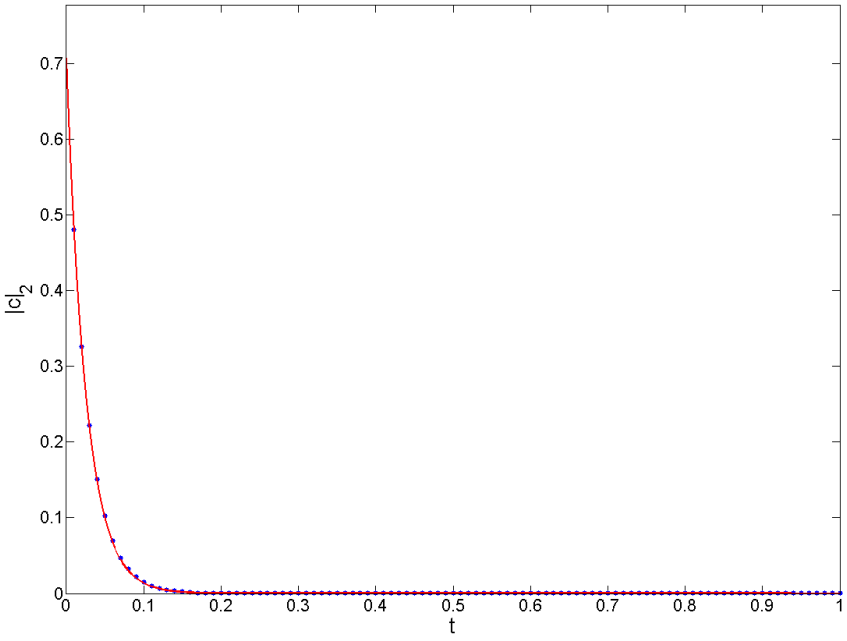

Figure 11: Comparison of numerical and analytical solutions of 1D diffusion equation using Crank-Nicolson, where dots are numerical solutions and solid lines are analytical solutions. The fact that the dots lie on the solid line suggests that the numerical and analytical solutions are in agreement.

Figure 11 shows a comparison of the numerical and analytical solutions. The fact that the dots lie on the solid line suggests that the numerical and analytical solutions are in agreement. In order to verify this, though, we look at the truncation error of the solution. If $c(x,t)$ is the numerical solution and $u(x,t)$ is the analytical soltuion, Figure 12 shows the value of $-\log_{10}||c|_2 - |u|_2|$. Because the method used is $O(h^2) + O(\Delta t^2)$, we expect a truncation error of the form

$$K\frac{(n\pi)^6(\Delta t)^3}{3!} \tag{68}$$

In this case, we have set $n = 2$ and $\Delta t = 0.01$, and the figure shows that the value of our truncation error is within that error. Figure 13 shows that same value at time $t = 1$ for varying values of Δt, and demonstrates that the error goes to zero as Δt goes to zero. Since the method is $O(\Delta t^2) + O(h^2)$, the global error at time $t = 1$ is $O(\Delta t) + O(h^2/\Delta t)$. If $h^2/\Delta t$ is small, which is true when Δt is large, then the global error is linear in Δt, which is consistent with Figure 13.

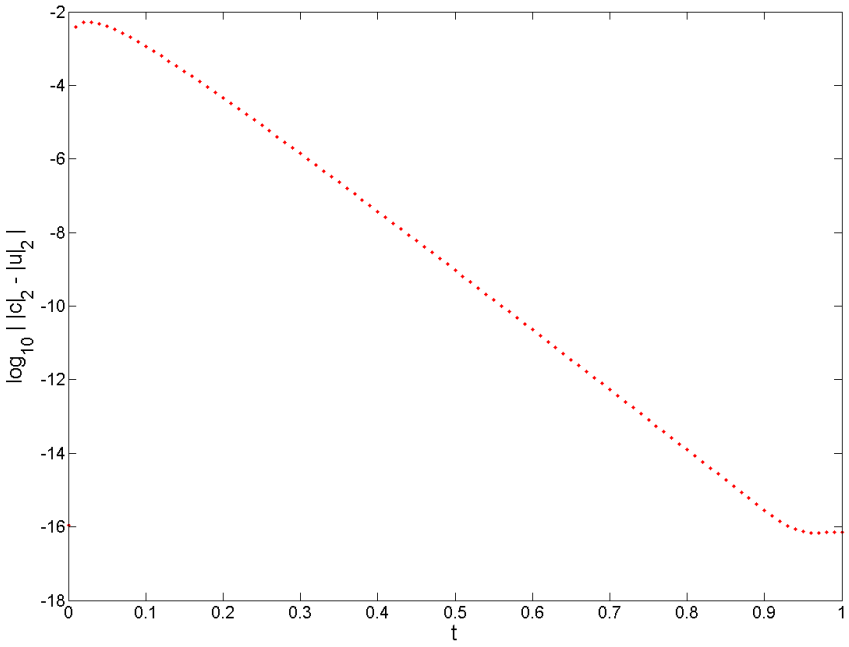

Figure 12: Truncation error for 1D diffusion equation using Crank-Nicolson, where $\Delta t = 0.01$. The truncation error is within expected values of the error for this method.

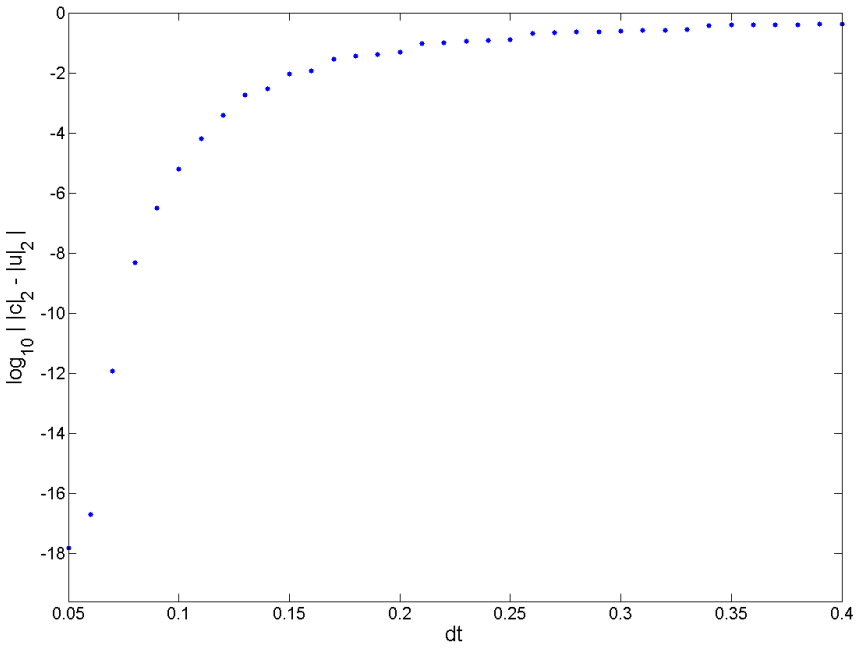

Figure 13: Truncation error for 1D diffusion equation using Crank-Nicolson at time $t = 1$. The error goes to zero as Δt goes to zero in a way that is consistent with this method.

D 2D Diffusion

Crank-Nicolson and ADI were used to solve the 2D diffusion equation

$$\frac{\partial c}{\partial t} = \alpha \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) \text{ for } 0 \leq x \leq 1, 0 \leq z \leq 1, \tag{69}$$

with Neumann boundary conditions

$$\begin{aligned} \frac{\partial c}{\partial x} &= 0 \text{ on } x = 0, 1, \\ \frac{\partial c}{\partial z} &= 0 \text{ on } z = 0, 1. \end{aligned} \tag{70}$$

To verify the numerical solutions of this equation, we compared the L^2 norm of the analytical and numerical solutions over time. To find the analytical solution, we apply separation of variables:

$$c(t, x, z) = T(t)\phi(x, z) \tag{71}$$

$$T'\phi = \alpha T \nabla^2 \phi \tag{72}$$

$$\frac{T'}{T} = \alpha \frac{\nabla^2 \phi}{\phi} = -\lambda \tag{73}$$

$$\tag{74}$$

This gives two separate equations,

$$T' + \lambda T = 0 \tag{75}$$

$$\alpha \nabla^2 \phi + \lambda \phi = 0 \tag{76}$$

Applying separation of variables again to Equation 76:

$$\phi(x,z) = X(x)Z(z) \tag{77}$$

$$\alpha X''Z + \alpha X Z'' + \lambda X Z = 0 \tag{78}$$

$$\frac{\alpha X''}{X} + \frac{\alpha Z''}{Z} + \lambda = 0 \tag{79}$$

$$\frac{-\alpha Z'' - \lambda Z}{Z} = \frac{\alpha X''}{X} = -k \tag{80}$$

This gives two separate equations,

$$\alpha X'' + kX = 0 \tag{81}$$

$$\alpha Z'' + (\lambda - k)z = 0 \tag{82}$$

For simplicity, assume $\alpha = 1$. Equation 81 becomes:

$$X(x) = a\cos(x\sqrt{k}) + b\sin(x\sqrt{k}) \tag{83}$$

Applying the boundary conditions, we find:

$$b = 0 \tag{84}$$

$$k = (n\pi)^2, n = 1, 2, ... \tag{85}$$

Equation 82 becomes:

$$Z(z) = p\cos(z\sqrt{\lambda - k}) + q\sin(z\sqrt{\lambda - k}) \tag{86}$$

Applying the boundary conditions, we find:

$$q = 0 \tag{87}$$

$$\lambda - k = (m\pi)^2, m = 1, 2, ... \tag{88}$$

$$\lambda = (n\pi)^2 + (m\pi)^2$$

A solution to Equation 69 is:

$$c(t,x,z) = Ae^{-((n\pi)^2+(m\pi)^2)t}\cos(n\pi x)\cos(m\pi z) \tag{90}$$

Setting α to 1, and given the initial conditions of $c(x,0) = \cos(2\pi x)\cos(\pi y)$, the L^2 norm of the analytical solution is

$$|c(t,x,z)|_2 = \sqrt{\int_0^1 \int_0^1 c^2(t,x,z)dxdz} = \frac{1}{2}e^{-5\pi t} \tag{91}$$

Figure 14 shows a comparison of the numerical and analytical solutions using Crank-Nicolson and ADI. The fact that the dots lie on the solid line suggests that the numerical and analytical solutions are in agreement. In order to verify this, though, we look at the truncation error of the solution. If $c(x,z,t)$ is the numerical solution and $u(x,z,t)$ is the analytical solution, Figure 15 shows the value of $-\log_{10}||c|_2 - |u|_2|$ for Crank-Nicolson and ADI. Because the method used is $O(h^2) + O(\Delta t^2)$, we expect a truncation error of the form

$$K\frac{((n\pi)^2 + (m\pi)^2)^3(\Delta t)^3}{3!} \tag{92}$$

In this case, we have set $n = 2$, $m = 1$, and $\Delta t = 0.01$, and the figure shows that the value of our truncation error is within that error. Figure 16 shows that same value at time $t = 1$ for varying values of Δt, and demonstrates that the error goes to zero as Δt goes to zero. Since the method is $O(\Delta t^2) + O(h^2)$, the global error at time $t = 1$ is $O(\Delta t) + O(h^2/\Delta t)$. If $h^2/\Delta t$ is small, which is true when Δt is large, then the global error is linear in Δt, which is consistent with Figure 16.

E 1D Advection-Diffusion

Crank-Nicolson was used to solve the 1D advection-diffusion equation,

$$\frac{\partial c}{\partial t} + U\frac{\partial c}{\partial x} = \alpha\frac{\partial^2 c}{\partial x^2} \text{ for } 0 \le x \le 1, \tag{93}$$

with Neumann boundary conditions

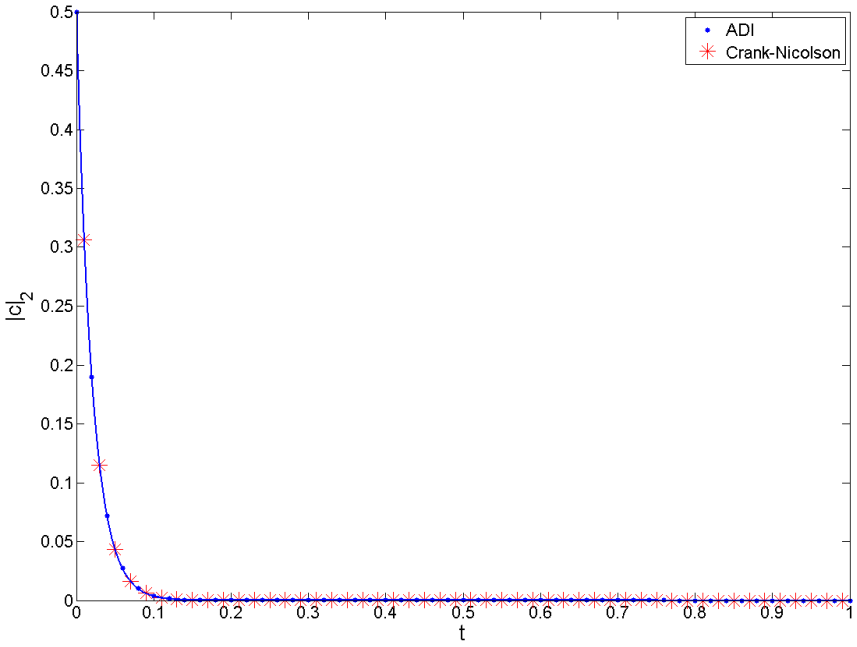

Figure 14: Comparison of numerical and analytical solutions of 2D diffusion equation using Crank-Nicolson and ADI, where dots are numerical solutions and solid lines are analytical solutions. The fact that the dots lie on the solid line suggests that the numerical and analytical solutions are in agreement.

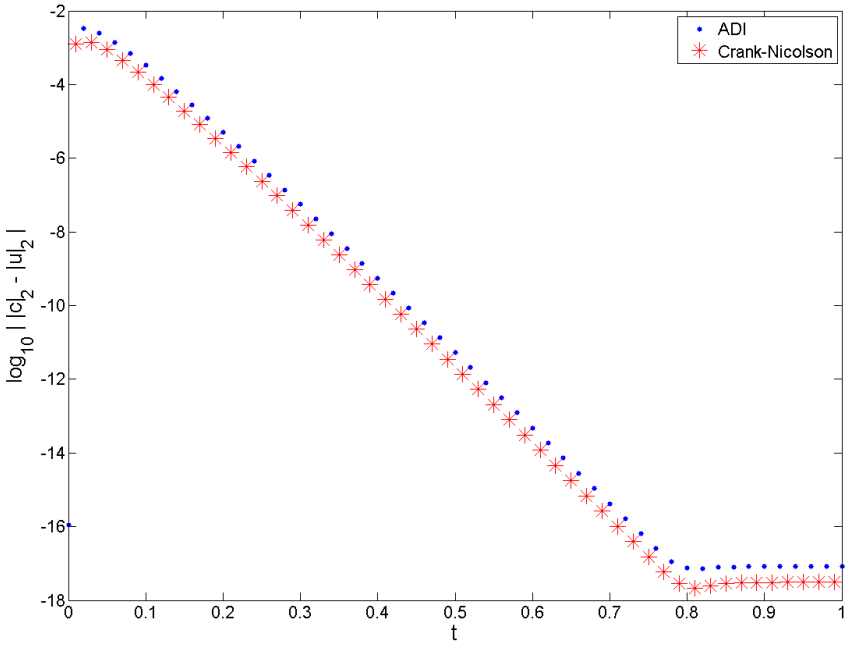

Figure 15: Truncation error for 2D diffusion equation using Crank-Nicolson and ADI, where $\Delta t = 0.01$. The truncation error is within expected values of the error for these methods.

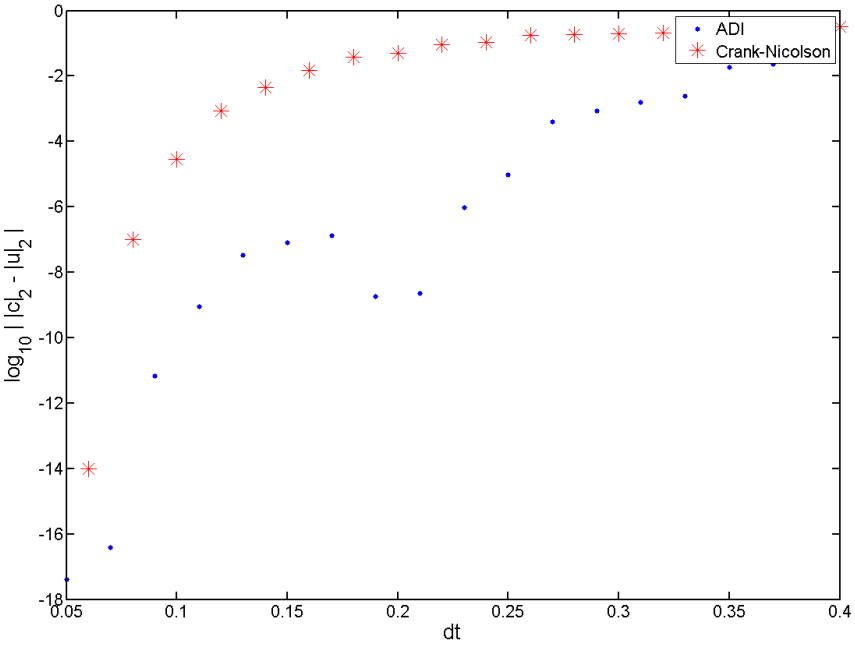

Figure 16: Truncation error for 2D diffusion equation using Crank-Nicolson and ADI at time $t = 1$. The error goes to zero as Δt goes to zero in a way that is consistent with these methods.

$$\frac{\partial c}{\partial x} = 0 \text{ on } x = 0, 1. \tag{94}$$

To verify the numerical solutions of this equation, we compared the L^2 norm of the analytical and numerical solutions over time. To find the analytical solution, we apply separation of variables:

$$c(t, x) = T(t)X(x) \tag{95}$$

$$T'\phi + UTX' = \alpha TX'' \tag{96}$$

$$\frac{T'}{T} + U\frac{X'}{X} = \alpha\frac{X''}{X} \tag{97}$$

$$\frac{T'}{T} = \frac{\alpha X'' - UX'}{X} = -\lambda \tag{98}$$

This gives two separate equations,

$$T' + \lambda T = 0 \tag{99}$$

$$\alpha X'' - UX' + \lambda X = 0 \tag{100}$$

For simplicity, assume $\alpha = U = 1$. Equation 119 becomes:

$$X(x) = ae^{x/2}\cos(x\sqrt{\lambda - 1/4}) + be^{x/2}\sin(x\sqrt{\lambda - 1/4}) \tag{101}$$

Applying the boundary conditions, we find:

$$\lambda = (n\pi)^2 + \frac{1}{4}, n = 1, 2, ... \tag{102}$$

$$a = -2bn\pi \tag{103}$$

A solution to Equation 93 is:

$$c(t, x) = be^{-((n\pi)^2 + \frac{1}{4})t}(-2ne^{x/2}\cos(n\pi x) + e^{x/2}\sin(n\pi x)) \tag{104}$$

Setting α and U to 1, and given the initial conditions of $c(x, 0) = -4\pi e^{x/2}\cos(2\pi x) + e^{x/2}\sin(2\pi x)$, the L^2 norm of the analytical solution is

$$|c(t, x)|_2 = \sqrt{\int_0^1 c^2(t, x)dx} = \sqrt{\frac{256e\pi^4 + 80e\pi^2 - 256\pi^4 - 80\pi^2}{2(1 + 16\pi^2)}}e^{-((n\pi)^2 + 1/4)t} \tag{105}$$

Figure 17 shows a comparison of the numerical and analytical solutions. The fact that the dots lie on the solid line suggests that the numerical and analytical solutions are in agreement. In order to verify this, though, we look at the truncation error of the solution. If $c(x, t)$ is the numerical solution and $u(x, t)$ is the analytical soltuion, Figure 18 shows the value of $-\log_{10}||c|_2 - |u|_2|$. Because the method used is $O(h) + O(\Delta t)$, we expect a truncation error of the form

$$K\frac{((n\pi)^2 + 1/4)^2(\Delta t)^2}{2!} \tag{106}$$

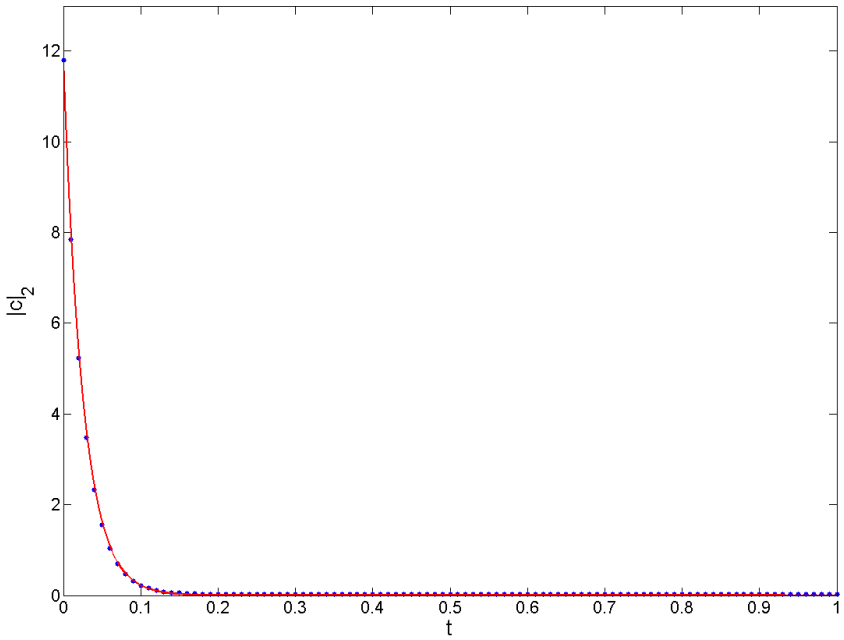

Figure 17: Comparison of numerical and analytical solutions of 1D advection-diffusion equation using Crank-Nicolson, where dots are numerical solutions and solid lines are analytical solutions. The fact that the dots lie on the solid line suggests that the numerical and analytical solutions are in agreement.

In this case, we have set $n = 2$ and $\Delta t = 0.01$, and the figure shows that the value of our truncation error is within that error. Figure 19 shows that same value at time $t = 1$ for varying values of Δt. Since the method is $O(\Delta t) + O(h)$, the global error at time $t = 1$ is $O(1) + O(h/\Delta t)$. If $h/\Delta t$ is small, which is true when Δt is large, then the global error should be roughly constant, which is consistent with Figure 19.

F 2D Advection-Diffusion

Crank-Nicolson and ADI were used to solve the 2D advection-diffusion equation,

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = \alpha\left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial z^2}\right) \text{ for } 0 \le x \le 1, 0 \le z \le 1, \tag{107}$$

with Neumann boundary conditions

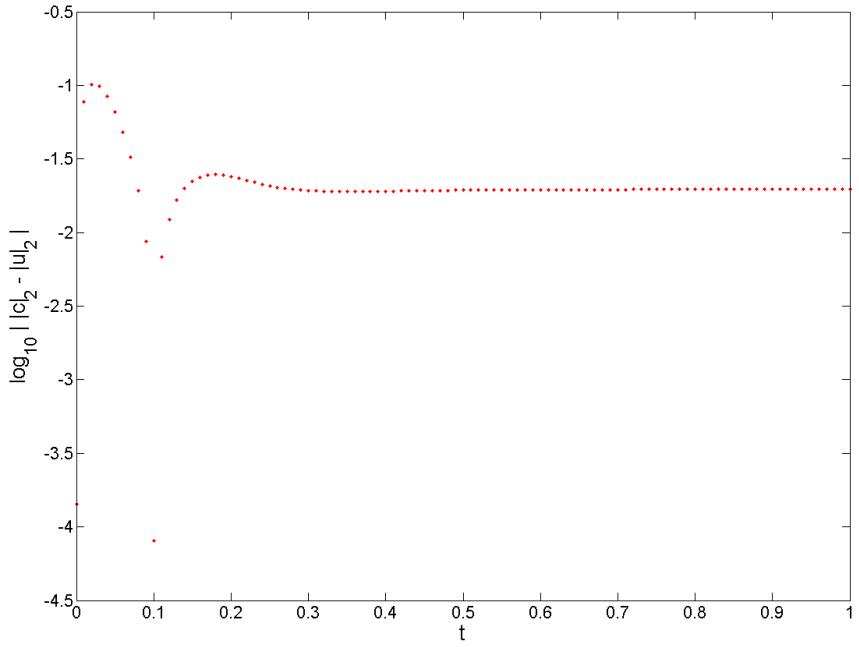

Figure 18: Truncation error for 1D advection-diffusion equation using Crank-Nicolson, where $\Delta t = 0.01$. The truncation error is within expected values of the error for this method.

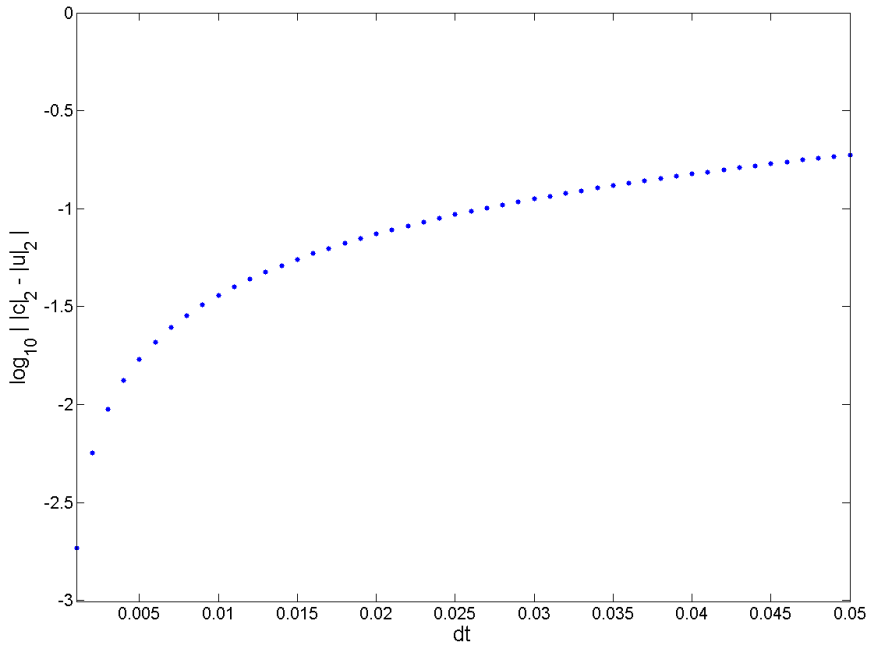

Figure 19: Truncation error for 1D advection-diffusion equation using Crank-Nicolson at time $t = 1$. The error goes to zero as Δt goes to zero in a way that is consistent with this method.

$$\begin{aligned} \frac{\partial c}{\partial x} &= 0 \text{ on } x = 0, 1, \\ \frac{\partial c}{\partial z} &= 0 \text{ on } z = 0, 1. \end{aligned} \tag{108}$$

To verify the numerical solutions of this equation, we compared the L^2 norm of the analytical and numerical solutions over time. In order to find the analytical solution, we apply separation of variables:

$$c(x, z, t) = T(t)\phi(x, z) \tag{109}$$

$$T'\phi + UT\frac{\partial \phi}{\partial x} = \alpha T \nabla^2 \phi \tag{110}$$

$$\frac{T'}{T} + U\frac{\frac{\partial \phi}{\partial x}}{\phi} = \alpha \frac{\nabla^2 \phi}{\phi} \tag{111}$$

$$\frac{T'}{T} = \frac{\alpha \nabla^2 \phi - U\frac{\partial \phi}{\partial x}}{\phi} = -\lambda \tag{112}$$

This gives two separate equations,

$$T' + \lambda T = 0 \tag{113}$$

$$\alpha \nabla^2 \phi - U\frac{\partial \phi}{\partial x} + \lambda \phi = 0 \tag{114}$$

Applying separation of variables again to Equation 114:

$$\phi(x, z) = X(x)Z(z) \tag{115}$$

$$\alpha X''Z + \alpha X Z'' - UX'Z + \lambda XZ = 0 \tag{116}$$

$$\frac{\alpha X''}{X} + \frac{\alpha Z''}{Z} - \frac{UX'}{X} + \lambda = 0 \tag{117}$$

$$\frac{-\alpha Z'' - \lambda Z}{Z} = \frac{\alpha X'' - UX'}{X} = -k \tag{118}$$

This gives two separate equations,

$$\alpha X'' - UX' + kX = 0 \tag{119}$$

$$\alpha Z'' + (\lambda - k)z = 0 \tag{120}$$

For simplicity, assume $\alpha = U = 1$. Equation 119 becomes:

$$X(x) = ae^{x/2}\cos(x\sqrt{k-1/4}) + be^{x/2}\sin(x\sqrt{k-1/4}) \tag{121}$$

Applying the boundary conditions, we find:

$$k = (n\pi)^2 + \frac{1}{4}, n = 1, 2, ... \tag{122}$$

$$a = -2bn\pi \tag{123}$$

Equation 120 becomes:

$$Z(z) = p\cos(z\sqrt{\lambda - k}) + q\sin(z\sqrt{\lambda - k}) \tag{124}$$

Applying the boundary conditions, we find:

$$z = p\cos(z\sqrt{\lambda - k}) \tag{125}$$

$$\lambda - k = (m\pi)^2, m = 1, 2, ... \tag{126}$$

$$\lambda = (n\pi)^2 + (m\pi)^2 + \frac{1}{4}$$

A solution to Equation 107 is:

$$c(x, z, t) = e^{-((n\pi)^2+(m\pi)^2+\frac{1}{4})t}(-2bne^{x/2}\cos(n\pi x) + be^{x/2}\sin(n\pi x))\cos(m\pi z) \tag{128}$$

Setting α and U to 1, and given the initial conditions of

$$c(x, z, 0) = (-4\pi e^{x/2}\cos(2\pi x) + e^{x/2}\sin(2\pi x))\cos(2\pi z), \tag{129}$$

the L^2 norm of the analytical solution is

$$|c(t, x, z)|_2 = \sqrt{\int_0^1 \int_0^1 c^2(t, x, z)dxdz} = \sqrt{\frac{256e\pi^4 + 80e\pi^2 - 256\pi^4 - 80\pi^2}{4(1 + 16\pi^2)}}e^{-((n\pi)^2+(m\pi)^2+1/4)t} \tag{130}$$

Figure 20 shows a comparison of the numerical and analytical solutions using Crank-Nicolson and

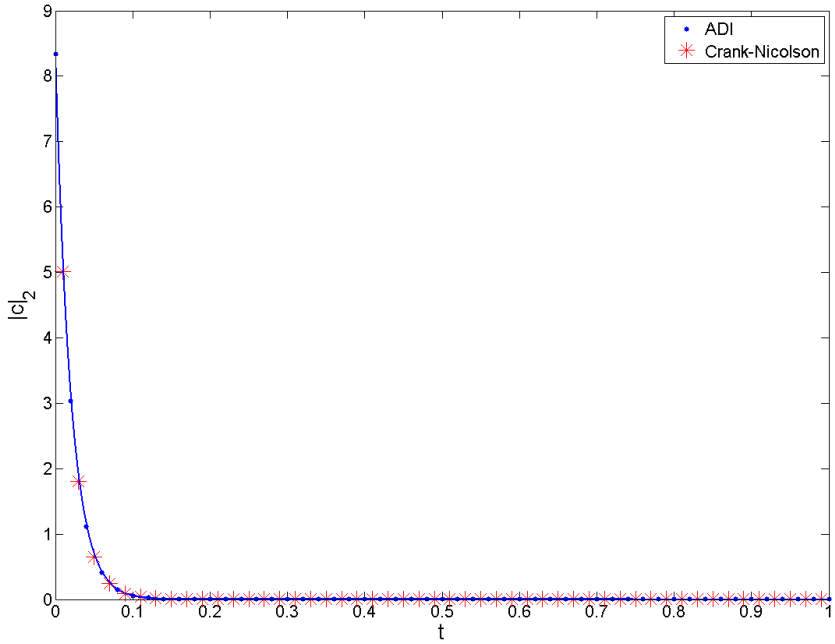

Figure 20: Comparison of numerical and analytical solutions of 2D advection-diffusion equation using Crank-Nicolson and ADI, where dots are numerical solutions and solid lines are analytical solutions. The fact that the dots lie on the solid line suggests that the numerical and analytical solutions are in agreement.

ADI. The fact that the dots lie on the solid line suggests that the numerical and analytical solutions are in agreement. In order to verify this, though, we look at the truncation error of the solution. If $c(x, z, t)$ is the numerical solution and $u(x, z, t)$ is the analytical solution, Figures 21 shows the value of $-\log_{10}||c|_2 - |u|_2|$ for Crank-Nicolson and ADI. Because the method used is $O(h)+O(\Delta t)$, we expect a truncation error of the form

$$K\frac{((n\pi)^2 + (m\pi)^2 + 1/4)^2(\Delta t)^2}{2!} \tag{131}$$

In this case, we have set $n = 2$, $m = 1$, and $\Delta t = 0.01$, and the figure shows that the value of our truncation error is within that error. Figure 22 shows that same value at time $t = 1$ for varying values of Δt. Since the method is $O(\Delta t) + O(h)$, the global error at time $t = 1$ is $O(1) + O(h/\Delta t)$. If $h/\Delta t$ is small, which is true when Δt is large, then the global error should be roughly constant, which is consistent with Figure 22.

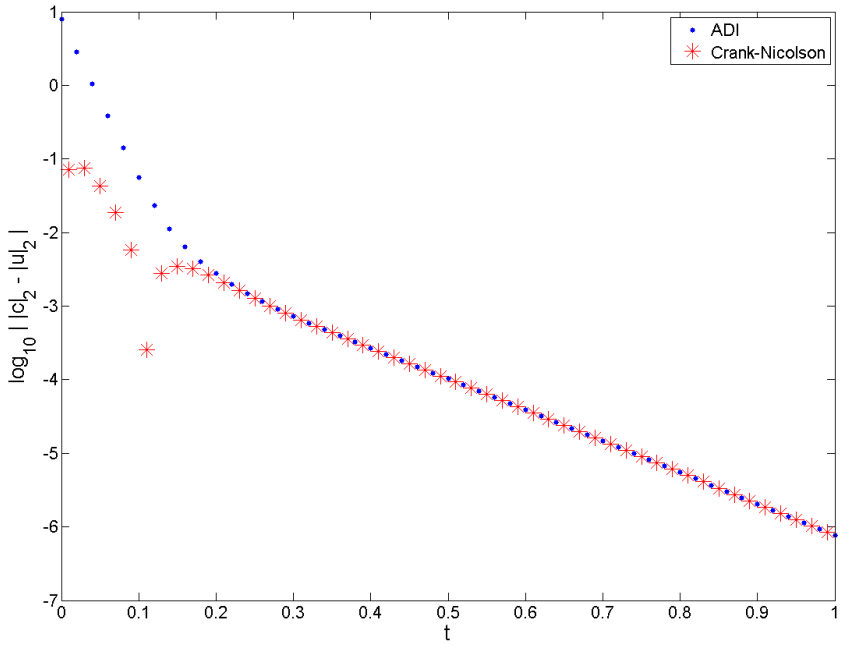

Figure 21: Truncation error for 2D advection-diffusion equation using Crank-Nicolson and ADI, where $\Delta t = 0.01$. The truncation error is within expected values of the error for these methods.

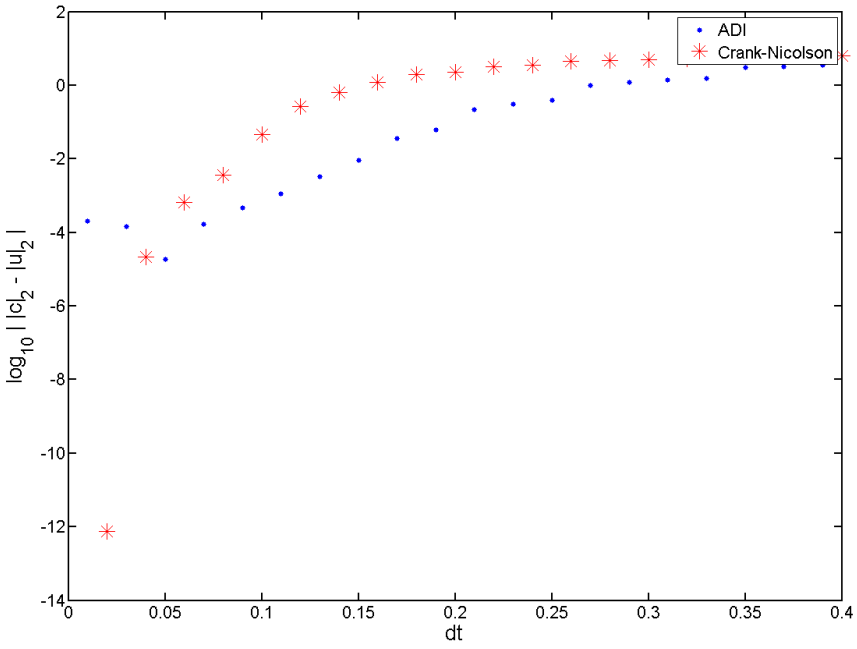

Figure 22: Truncation error for 2D advection-diffusion equation using Crank-Nicolson and ADI at time $t = 1$. The error goes to zero as Δt goes to zero in a way that is consistent with these methods.

G MATLAB code

The file solveB.m finds a solution for the blood cell concentration given an initial condition, K, Pe, $\epsilon, \Delta t$, and h.

```
% solveB.m
% Returns solution for blood cell concentration

function [bfinal, t, avgConc, flag, p1, p2] = case1(b0, params, tols)

    K = params(1);
    epsilon = params(2);
    Pe = params(3);
    dp = params(4);

    l_d = 0.2;
    D = 10^-9;
    U_b = D*Pe/l_d;

    dt = tols(1);
    endtime = tols(2);
    h = tols(3);

    ncols = 1/h+1; nrows = 1/h+1;

    [u1 w1 p1 p2] = getVelPress(K,h,dp);
    avgConc = [];

    u = matToVect(u1(1:nrows,:));
    w = matToVect(w1(1:nrows,:));

    uMat = diag(u);
    wMat = diag(w);

    solutionV = b0;

    D2z = zeros(ncols*nrows);
    Dz = D2z;
    Dx = D2z;

    % Matrix to take first derivative in z-direction
    temp = zeros(nrows);
    for j = 1:nrows
        for k = 1:nrows
```

```
        if j-k == 1
            temp(j,k) = 1/(2*h);
        elseif k-j == 1
            temp(j,k) = -1/(2*h);
        end
    end
end
Dz = kron(temp,eye(ncols));

for j = 1:ncols
    Dz(j,:) = zeros(1,ncols*nrows);
    Dz(j,j) = epsilon^2*U_b*l_d/D*w(j);
end
for j = ncols*nrows-ncols+1:ncols*nrows
    Dz(j,:) = zeros(1,ncols*nrows);
    Dz(j,j) = epsilon^2*U_b*l_d/D*w(j);
end
for j = 1:ncols:ncols*nrows
    Dz(j,:) = zeros(1,ncols*nrows);
end

% Matrix to take second derivative in z-direction
temp = zeros(nrows);
for j = 1:nrows
    for k = 1:nrows
        if j==k
            temp(j,k) = -2/h^2;
        elseif abs(j-k) == 1
            temp(j,k) = 1/h^2;
        end
    end
end
D2z = kron(temp,eye(ncols));

for j = 1:ncols
    D2z(j,:) = zeros(1,ncols*nrows);
    D2z(j,j) = (-2+2*h*epsilon^2*U_b*l_d/D*w(j))/h^2;
    D2z(j,j+ncols) = 2/h^2;
end
for j = ncols*nrows-ncols+1:ncols*nrows
    D2z(j,:) = zeros(1,ncols*nrows);
    D2z(j,j) = (-2-2*h*epsilon^2*U_b*l_d/D*w(j))/h^2;
    D2z(j,j-ncols) = 2/h^2;
end
for j = 1:ncols:ncols*nrows
    D2z(j,:) = zeros(1,ncols*nrows);
end
```

```
% Matrix to take first derivative in x-direction
temp = zeros(ncols);
for j = 2:ncols-1
    for k = 1:ncols
        if j-k==1
            temp(j,k) = -1;
        elseif j==k
            temp(j,k) = 1;
        end
    end
end

Dx = kron(eye(nrows),temp);
Dx = Dx ./ (2*h);

% Matrix to take second derivative in x-direction
temp = zeros(ncols);
for j = 2:ncols-1
    for k = 1:ncols
        if j == k
            temp(j,k) = -2;
        elseif abs(j-k) == 1
            temp(j,k) = 1;
        end
    end
end
temp(ncols,ncols-1) = 2;
temp(ncols,ncols) = -2;
D2x = kron(eye(nrows),temp);
D2x = D2x ./ h^2;

[X,Y] = meshgrid(0:h:1,1:-h:0);

for t = dt:dt:endtime
    avgConc = [avgConc mean(solutionV)];

    I = eye(nrows*ncols);
    a = (I + dt/2*epsilon^2*U_b*l_d/D*uMat*Dx...
     + dt/2*epsilon^2*U_b*l_d/D*wMat*Dz - dt/2*D2x - dt/2*D2z);
    b = (I - dt/2*epsilon^2*U_b*l_d/D*uMat*Dx...
     - dt/2*epsilon^2*U_b*l_d/D*wMat*Dz + dt/2*D2x + dt/2*D2z)*solutionV;

    tempMat = vectToMat(solutionV-a\b,nrows,ncols);
    if sqrt(trapz(0:h:1-h,trapz(0:h:1,tempMat(1:nrows,1:ncols-1).^2))) < dt^4
```

```
            break
        end

        solutionV = a\b;
    end

    if t == endtime
        flag = 1;
    elseif min(solutionV) < 0
        flag = 2;
    elseif max(solutionV) > 1e3
        flag = 3;
    else
        flag = 0;
    end
    bfinal = solutionV;

end
```

The file getVelPress.m gives the velocity of blood and the pressure of blood and dialysate, given K, h, and Δp.

```
% getVelPress.m
% Returns velocity of blood and pressures of blood and dialysate

function [u1 w1 p1 p2] = getVelocities(K,h,dp)

    % Set values of pressures at x = 0, x = 1
    p11 = 815;
    p12 = 796;
    p21 = 472;
    p22 = 472+dp;

    A = -1/2*(p11*exp(-6^(1/2)*K^(1/2))-p21*exp(-6^(1/2)*K^(1/2))...
     +p22-p12)/(exp(6^(1/2)*K^(1/2))-exp(-6^(1/2)*K^(1/2)));
    B = 1/2*(p22+exp(6^(1/2)*K^(1/2))*p11-exp(6^(1/2)*K^(1/2))...
     *p21-p12)/(exp(6^(1/2)*K^(1/2))-exp(-6^(1/2)*K^(1/2)));
    a = 1/2*p22+1/2*p12-1/2*p21-1/2*p11;
    b = 1/2*p21+1/2*p11;

    x = 0:h:1;
    z = -1:h:1;
```

```
    % Solve for pressures of blood and dialysate
    p1 = A*exp(sqrt(6*K).*x)+B*exp(-sqrt(6*K).*x)+a.*x+b;
    p2 = a.*x+b-A*exp(sqrt(6*K).*x)-B*exp(-sqrt(6*K).*x);


    u1 = zeros(length(z),length(x));
    w1 = u1;

    % Solve for velocities of blood
    for i = 1:length(z)
        for j = 1:length(x)
            x2 = h*(j-1);
            z2 = 1-h*(i-1);
            u1(i,j) = (A*sqrt(6*K)*exp(sqrt(6*K)*x2)...
             -B*sqrt(6*K)*exp(-sqrt(6*K)*x2)+a)*(z2^2/2-z2);
            w1(i,j) = (A*6*K*exp(sqrt(6*K)*x2)...
              + B*6*K*exp(-sqrt(6*K)*x2))*(-z2^3/6+z2^2/2-1/3);
        end
    end

end
```

The files varyKPe.m and varyDp.m find solutions for blood cell concentration while varying the values of Pe, K, and Δp.

```
% varyKPe.m
% Finds solution for blood cell concentration
% for varying values of K, Pe

clear;

h = 1/15;
ncols = 1/h+1; nrows = 1/h+1;
b0 = zeros(ncols*nrows,1);

% Set initial conditions
count = 1;
z = 1;
for j = 1:nrows
    x = 0;
    for k = 1:ncols
        b0(count) = 1;
        count = count + 1;
```

```
        x = x + h;
    end
    z = z - h;
end

dp = 518-472;
k0 = 1e-8;
dk = 1e-3;
k1 = 1e-2;

Pe0 = 1;
dPe = 1000;
Pe1 = 30000;

epsilon = 5e-4;
dt = 0.01;
tfinal = 100;

% Find solution for blood cell concentration
% for varying values of K and Pe
for kloop = k0:dk:k1
    for Peloop = Pe0:dPe:Pe1
        params = [kloop epsilon Peloop dp];
        tols = [dt tfinal h];
        [bfinal t avgConc flag] = solveB(b0, params, tols);
        max(bfinal)

        savefile = ['Case1K' num2str(kloop) 'Pe' num2str(Peloop) '.mat'];
        save(savefile, 'bfinal', 't', 'avgConc', 'flag');
    end
end

************************************************

% varyDp.m
% Finds solution for blood cell concentration for varying values of Dp

clear

h = 1/15;
ncols = 1/h+1; nrows = 1/h+1;
b0 = zeros(ncols*nrows,1);

% Set initial conditions
count = 1;
z = 1;
for j = 1:nrows
```

```
    x = 0;
    for k = 1:ncols
        b0(count) = 1;
        count = count + 1;
        x = x + h;
    end
    z = z - h;
end

k = 0.01;
Pe = 10000;
epsilon = 5e-4;

dt = 0.01;
tfinal = 100;

% Find solution for blood cell concentration
% for varying values of Dp
for dploop = 10:10:500
    params = [k epsilon Pe dploop];
    tols = [dt tfinal h];
    [bfinal t avgConc flag p1 p2] = solveB(b0, params, tols);
    max(bfinal)

    savefile = ['Case2dp' num2str(dploop) '.mat'];
    save(savefile, 'bfinal', 't', 'avgConc', 'flag', 'p1', 'p2');
end
```

The files analyzeVaryKPe.m and analyzeVaryDp.m read the files generated by varyKPe.m and varyDp.m and generate the plots in Section 3.

```
% analyzeVaryKPe.m

nrows = 16;
ncols = 16;
k0 = 1e-8;
dk = 1e-3;
k1 = 1e-2;
Pe0 = 1;
dPe = 1000;
Pe1 = 30000;
```

```
% Read files and plot bmax for varying values of Pe
count = 1;
for kloop = k0:dk:k1
    colors = [
         0         0    1.0000;
         0    0.5000         0;
    1.0000         0         0;
         0    0.7500    0.7500;
    0.7500         0    0.7500;
    0.7500    0.7500         0;
    0.2500    0.2500    0.2500;
         0         0    1.0000;
         0    0.5000         0;
    1.0000         0         0;
         0    0.7500    0.7500;
    0.7500         0    0.7500;
    0.7500    0.7500         0;
    0.2500    0.2500    0.2500];

    for Peloop = Pe0:dPe:Pe1
        loadfile = ['Case1K' num2str(kloop) 'Pe' num2str(Peloop) '.mat'];
        load(loadfile);
        figure(1)
        hold on
        plot(Peloop,max(bfinal),'.','MarkerEdgeColor',colors(count,:));
        xlabel('Pe');
        ylabel('b_m_a_x')
    end
    count = count + 1;
end
```

The file diff2DADI.m finds a solution to the 2D diffusion equation using the method of ADI.

```
% diff2D_ADI.m
% Solves the 2D diffusion equation using ADI

clear

dt = 0.01;
L = 1; H = 1;
h = 1/15;
N = L/h+1; M = H/h+1;
m = 1; n = 2;
```

```
endtime = 1;

alpha = 1;

c = zeros(M*N,1);

% Set initial conditions
count = 1;
y = H;
for j = 1:M
    x = 0;
    for k = 1:N
        c(count) = cos(n*pi*x/L)*cos(m*pi*y/H);
        count = count + 1;
        x = x + h;
    end
    y = y - h;
end

newC = vectToMat(c,M,N);

% Matrix to take second derivative in x-direction
temp = zeros(N);
for j = 1:N
    for k = 1:N
        if j == k
            temp(j,k) = -2/(h^2);
        elseif abs(j-k) == 1
            temp(j,k) = 1/(h^2);
        end
    end
end
temp(1,2) = 2/(h^2);
temp(N,N-1) = 2/(h^2);
D2x = kron(eye(M),temp);

% Matrix to take second derivative in z-direction
temp = zeros(M);
for j = 1:M
    for k = 1:M
        if j == k
            temp(j,k) = -2/(h^2);
        elseif abs(j-k) == 1
            temp(j,k) = 1/(h^2);
        end
    end
end
```

```
temp(1,2) = 2/(h^2);
temp(M,M-1) = 2/(h^2);
D2y = kron(eye(N),temp);

amps = zeros(length(0:dt:endtime),1);
amps_0 = 0.5;
trunc = amps;
times2 = 0:0.001:endtime;
decay = (m^2+n^2)*pi^2;
exact = amps_0*exp(-decay.*times2);

% Find L^2 norm of solution, truncation error, and solution
% at the next timestep
count = 1;
for time = 0:dt:endtime
    amps(count) = sqrt(trapz(0:h:L,trapz(0:h:H,newC.^2)));
    trunc(count) = log10(abs(0.5*exp(-decay*time)...
     - sqrt(trapz(0:h:L,trapz(0:h:H,newC.^2)))  ));

    I = eye(M*N);
    mat1 = permutation(c,M,N,1);
    mat2 = mat1';
    cperm = mat1*c;

    ctemp = (I-alpha*dt/2*D2x)\( c+alpha*dt/2*mat2*D2y*mat1*c );
    ctempperm = mat1*ctemp;
    c = (I-alpha*dt/2*D2y)\(ctempperm +alpha*dt/2*mat1*D2x*mat2*ctempperm);
    c = mat2*c;
    newC = vectToMat(c,M,N);

    count = count + 1;
end
```

References

[1] Bailey, G. L. (1972). Hemodialysis: Principles and practice. Academic Press.

[2] Pozrikidis, C. (1997). Introduction to theoretical and computational fluid dynamics. Oxford University Press,

[3] S. R. Wickramasinghe, B. Han, J. D. Garcia, and R. Specht. (2005). Microporous membrane blood oxygenators. AIChE Journal, 51(2), 656-670.

[4] Saltzman, M. W. (2009). Biomedical engineering: Bridging medicine and technology. Cambridge University Press.

[5] Stein, A. (2006). Kidney failure explained (2nd ed.) Class Publishing.

[6] S. Waikar, B.S. Tilley, C.S. Bohun, C. Breward, R. Droumeva, I. Griffiths, M. Hennessy, H. Huang, M. Kloosterman, A. Pan, C. Raymond, D. Schwendeman, J. Siggers, M. Tindall, J. Wattis, J. Whiteley, R. Wittaker, JF Williams, Sodium Flux during Haemodialysis, Proceedings of the OCCAM-Fields-MITACS Biomedical Problem Solving Workshop, 2009.