University of California, San Diego

Hollywood vs. Reality:
Examining the Diffusion of Etorphine Following a Direct Injection

Melissa Hernandez
Murphy Mao
Raymond Wang
Tianyi Zhao

BENG 221
November 11, 2014

Table of Contents

1. Introduction

1.1 Background

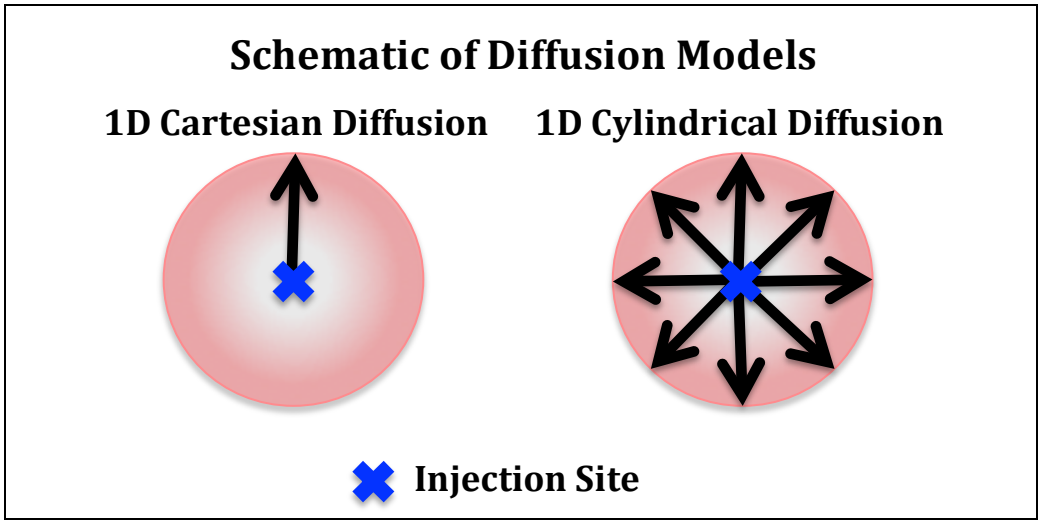
The acclaimed television series, *Dexter*, has captured the attention of a wide range of audiences. The series focuses on Dexter, a blood spatter analyst with the Miami Metro Homicide Department by day and a serial killer who kills other serial kills by night. We aimed to address how realistic the show is; more specifically, we wanted to investigate the drug Dexter uses to render his victims unconscious. In the series, the drug, known as etorphine or M99, has an instantaneous effect on its victims, but we wanted to verify the speed at which the drug diffuses through the body.

Etorphine is an analgesic with a potency that is 1000-3000 times stronger than morphine [1]. Consequently, etorphine is only legal for veterinary uses in large mammals, such as horses or elephants, for immobilization. In a study with stallions, 5.5 mg of etorphine were injected intravenously. For that particular dosage, the stallions took 55 ± 4 seconds to become immobilized (range of 20 to 185 seconds) [2]. For humans, the dose level is a fraction of that used for animals. In fact, a dosage of 0.03 mcg will be lethal for a human [3]. Since a single drop of veterinary-grade etorphine on the skin is enough to kill a human, packages of etorphine always come with a human antidote in the event of any accidents.

1.2 Problem Statement

For this problem, we aimed to model the diffusion of etorphine after intravenous administration to address the following problem statements:

- How quickly does etorphine diffuse throughout the body?
- How do 1-dimensional Cartesian and 1-dimensional cylindrical diffusion models compare for the rate of diffusion?
- Does "Dexter" accurately portray the rate at which etorphine takes effect on its victims?



The model for the 1D cartesian diffusion model and the 1D cylindrical diffusion model are shown on the previous page. For the 1D Cartesian diffusion model, the drug diffuses through the arterial wall in one single direction. However, with the 1D cylindrical diffusion model, the drug diffuses outward in the radial direction and through each point along the arterial wall.

Based on our diffusion models, we made several assumptions to simplify the analytical solutions for the 1D Cartesian and 1D cylindrical diffusion models. The assumptions were the same for both of the models, unless indicated otherwise.

- The thickness of the artery wall is negligible.
- The artery is a "perfect artery", meaning the shape is completely uniform, and the radius remains constant everywhere.
- There is no convection in the process (aka blood flow is ignored).
- The injection occurs at the center of the artery.
- The z-direction along the length of the artery is ignored. (Cylindrical case only)
- The angle in the artery is ignored since the artery is assumed to be perfectly symmetric. (Cylindrical case only)

Due to the incorporation of two different coordinate systems, the analytical solutions and the associated initial condition and boundary conditions for each case will be discussed separately.

2. Analytical Solutions

2.1 Constants

Before beginning to solve for the analytical solutions in both cases, constants were compiled to reflect the model being used for the problem statement. Based on the series *Dexter*, the constants were chosen for the carotid artery, since Dexter injects all of his victims in the neck. The constants are displayed in the table below.

Parameters	Values
Initial Concentration, $\mathbf{C_0}$	7.29×10^{-4} mol/L
Radius of the Artery, **a**	3.15 mm
Diffusivity, **D**	6.48×10^{-5} mm^2/s

The initial concentration, C_0, was determined based on two assumptions. First, by examining the scenes in which Dexter injects his victims, the volume of the injection was observed to be approximately 5 cm^3. Next, we assumed that Dexter used half of the lethal dose (0.015 mcg) of etorphine to immobilize his victims. Using the molecular weight of etorphine (411.53 g/mol) [4] and the assumed concentration, we calculated our value for C_0. For the radius of the artery, a, the diameter for the male carotid artery (6.5 mm) and the diameter for a female carotid artery (6.1 mm)

were averaged together [5]. Lastly, the diffusivity was computed using the Stokes-Einstein equation. To use this equation, we made the assumption that the radius of an etorphine molecule was 1 nm. All of the calculations for the constants can be found in Appendix A.

2.2 1D Cartesian Diffusion Model

To begin modeling our 1D Cartesian drug diffusion, we started with the governing diffusion equation below for Cartesian coordinates. For our equations, we substituted the x found in Cartesian coordinates for r.

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial r^2}$$

Before continuing with our governing equation, we needed to establish an initial condition and two boundary conditions. For our initial condition, we assumed a delta-dirac function multiplied by our initial concentration to represent the instantaneous effect of the injection on the concentration. The first boundary condition was determined to be a concentration of zero at the arterial walls since all of the drug will be immediately carried off. For the second boundary condition, the change in concentration, or the flux, at the center of the artery was zero since no diffusion will be occurring at the center. The conditions are listed below.

$$IC: C(r,0) = C_0\delta(r,t)$$
$$BC1: C(a,t) = 0$$
$$BC2: \frac{\partial C}{\partial t}(0,t) = 0$$

To solve our governing equation, it was necessary to use the method of separation of variables.

$$C(r,t) = \varphi(r)\cdot G(t)$$

Both of the variables, r and t, were separated and set equal to a constant.

$$\frac{\frac{d^2\varphi(r)}{dr^2}}{\varphi(r)} = \frac{1}{D}\frac{\left(\frac{dG(t)}{dt}\right)}{G(t)} = -\lambda$$

This allowed the original governing equation to be separated into a time-dependent part and a space dependent part. The time-dependent portion was solved first. The solution is shown below.

$$\frac{dG}{dt} = -D\lambda G$$
$$G(t) = G_0 e^{-D\lambda t}$$

Next, the space-dependent part needed to be solved beginning with the equation and boundary conditions on the following page.

$$\frac{d^2\varphi}{dr^2}+\lambda\varphi=0$$

$$BCs:\frac{\partial\varphi}{\partial r}(0)=0,\varphi(a)=0$$

To determine a solution, the values for the constant, λ, had to be examined. By checking the possible values for λ, it was determined that λ must be greater than zero in order to avoid having a trivial solution.

$$\lambda=0:\varphi(r)=Ar+B\Rightarrow A=B=0\Rightarrow Trivial$$

$$\lambda<0:\varphi(r)=Ae^{\sqrt{-\lambda}r}+Be^{-\sqrt{-\lambda}r}\Rightarrow A=B=0\Rightarrow Trivial$$

$$\lambda>0:\varphi(r)=A\cos(\sqrt{\lambda}r)+B\sin(\sqrt{\lambda}r)$$

With an equation for the space-dependent part, the boundary conditions needed to be incorporated, and the value for λ was determined.

$$BC1:\frac{\partial\varphi}{\partial r}(r=0):-A\sin(0)+B\cos(0)=0$$

$$BC2:\varphi(r=a):A\cos(\sqrt{\lambda}a)+B\sin(\sqrt{\lambda}a)=0$$

$$\therefore B=0$$

$$A\cos(\sqrt{\lambda}a)=0\Rightarrow\cos(\sqrt{\lambda}a)=0$$

$$\sqrt{\lambda}=\left(\frac{(2n+1)\pi}{2a}\right),(n=0,1,2,...)$$

This resulted in a general solution for the space-dependent part. Using the original equation for the separation of variables, the time-dependent and space-dependent portions were combined to yield an equation for the concentration shown below.

$$C(r,t)=\sum_{n=0}^{\infty}A_n\cos\left(\frac{(2n+1)\pi}{2a}r\right)e^{-D\left(\frac{(2n+1)\pi}{2a}\right)^2t}$$

In order to determine the value of the coefficient, A_n, the initial condition needed to be used.

$$IC:C(r,0)=C_0\delta(r,t)$$

$$C(r,0)=\sum_{n=0}^{\infty}A_n\cos\left(\frac{(2n+1)\pi}{2a}r\right)=C_0\delta(r)$$

$$A_n=\frac{2}{a}\int_0^R C_0\delta(r)\cdot\varphi(r)dr=\frac{2}{a}\int_0^R C_0\delta(r)\cos\left(\frac{(2n+1)\pi}{2a}r\right)dr$$

Due to the presence of the delta-dirac function, the solution becomes a concentration per unit length. The value for the coefficient is shown below.

$$A_n=\frac{2}{a}C_0$$

By plugging the coefficient back into the general solution, a final solution for the 1D Cartesian diffusion model was determined for the concentration per unit length with respect to r and t, as shown on the following page.

$$C(r,t) = \sum_{n=0}^{\infty} \frac{2C_0}{a} \cos\left(\frac{(2n+1)\pi}{2a} r\right) e^{-D\left(\frac{(2n+1)\pi}{2a}\right)^2 t}$$

2.3 1D Cylindrical Diffusion Model

The equation below represents the governing equation for diffusion in a cylindrical model. Since we are ignoring the z-direction and assuming that the artery is symmetric, the concentration terms with respect to z and theta were ignored to simplify the problem.

$$\frac{1}{D}\frac{\partial C}{\partial t} = \frac{\partial^2 C}{\partial r^2} + \frac{1}{r}\frac{\delta C}{\partial r} + \frac{1}{r^2}\frac{\partial^2 C}{\partial \theta^2} + \frac{\partial^2 C}{\partial z^2}$$

$$\frac{1}{D}\frac{\partial C}{\partial t} = \frac{\partial^2 C}{\partial r^2} + \frac{1}{r}\frac{\delta C}{\partial r}$$

The same initial condition and boundary conditions used for the 1D Cartesian diffusion model were also used for the 1D cylindrical diffusion model.

$$IC: C(r,0) = C_0 \delta(r,t)$$

$$BC1: C(a,t) = 0$$

$$BC2: \frac{\partial C}{\partial t}(0,t) = 0$$

Also similar to before, the equation needed to be solved using the method of separation of variables, and the time-dependent part and space-dependent part were both set equal to a constant, λ. Different variables were used to differentiate the two models.

$$C(r,t) = R(r) \cdot \tau(t)$$

$$\frac{1}{D} R(r) \cdot \tau'(t) = R''(r)\tau(t) + \frac{1}{r} R'(r)$$

$$\frac{1}{D}\frac{\tau'}{\tau} = \frac{R''}{R} + \frac{1}{r}\frac{R'}{R} = -\lambda$$

The time-dependent portion was solved the same way as before, and yielded the same result.

$$\tau' = -D\lambda\tau$$

$$\tau(t) = c_1 e^{-D\lambda t}$$

However, the space-dependent part could not be solved the same way as before due to the 1/r multiple present in the equation below. In effect, the equation had to be converted to a Bessel function form.

$$R'' + \frac{1}{r}R' + \lambda R = 0$$

$$x = \sqrt{\lambda} r$$

$$R' = \frac{dy}{dx}\frac{dx}{dr} = \frac{dy}{dx}\frac{d}{dr}\left(\sqrt{\lambda} r\right) = \sqrt{\lambda}\frac{dy}{dx}$$

$$R'' = \lambda \frac{d^2 y}{dx^2}$$

The equation below represents the general form of the Bessel equation of the zeroth order.

$$x^2 \frac{d^2 y}{dx^2} + x\frac{dy}{dx} + x^2 y = 0$$

After taking the Frobenius Series of the equation above, the general form of the Bessel function was found, as shown below.

$$y = c_1 J_0(x) + c_2 Y_0(x)$$

Since $Y_0(x)$ goes to infinity at r=0, the value of c_2 must be zero, which makes the second term in the Bessel function go away. The result is shown in the equation below.

$$y = c_1 J_0(x)$$

The equation below is the Bessel function of the first kind, zero order.

$$J_0(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{2^{2n}(n!)^2}$$

The value of x was then plugged back into the equation to yield the following equation.

$$R(r) = c_1 J_0(x) = c_1 J_0\left(\sqrt{\lambda} r\right)$$

Similar to an equation with a sine or cosine, the eigenvalues were solved for by applying the boundary conditions and finding values that do not yield a trivial solution.

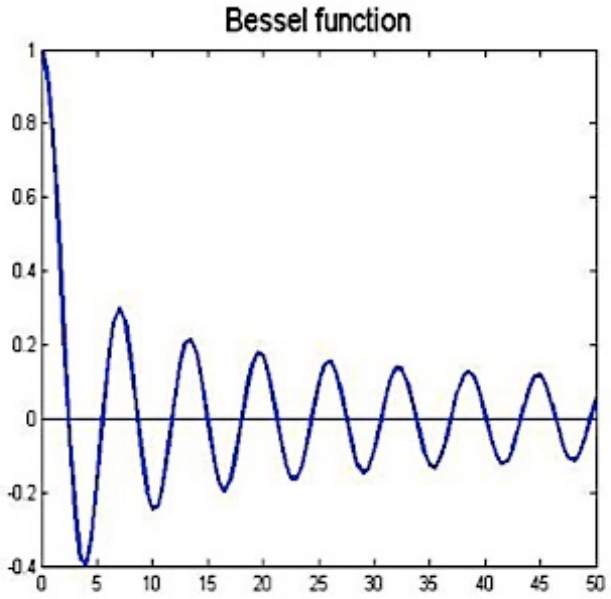
$$J_0 = \left(\sqrt{\lambda_n} r\right)$$

$$J_0 = \left(\sqrt{\lambda_n} a\right) = 0$$

Since there is no convenient standard notation for the eigenvalue, it is necessary to find the roots of the Bessel function using an analytical table or with Matlab. A graph of the Bessel function and its roots are shown on the following page.

$$\left(\sqrt{\lambda_n} a\right) = \left(roots_n\right)$$

$$\lambda_n = \left(\frac{(roots_n)}{a} \right)^2$$



By combining the time-dependent and space-dependent parts from the separation of variables, the general solution was determined.

$$C(r,t) = R(r) \cdot \tau(t) = \sum_{n=0}^{\infty} A_n J_0(\lambda_n r) e^{-\lambda D t}$$

Similar to the 1D Cartesian diffusion model, we still needed to solve for the coefficient. However, the Sturm-Liouville Theory was used instead of Fourier series. The initial condition was still applied, though.

$$\int_0^a r J_0\left(\sqrt{\lambda_n} r\right) J_0\left(\sqrt{\lambda_m} r\right) dr = 0, \; if \; n \neq m$$

$$If \;\; n = m, \int_0^a C_0 \delta(r) J_0\left(\sqrt{\lambda_n} r\right) J_0\left(\sqrt{\lambda_m} r\right) dr = A_n \int_0^a r J_0^2\left(\sqrt{\lambda_n} r\right) dr$$

By making a substitution with a Bessel function of the first kind, first order in the denominator, the coefficient was then solved for.

$$A_n = \frac{\int_0^a C_0 \delta(r) J_0\left(\sqrt{\lambda_n} r\right) dr}{\int_0^a r J_0^2\left(\sqrt{\lambda_n} r\right) dr}$$

$$\int_0^a r J_0^2\left(\sqrt{\lambda_n} r\right) dr = \frac{a^2}{2} \cdot J_1^2\left(\sqrt{\lambda_n} a\right)$$

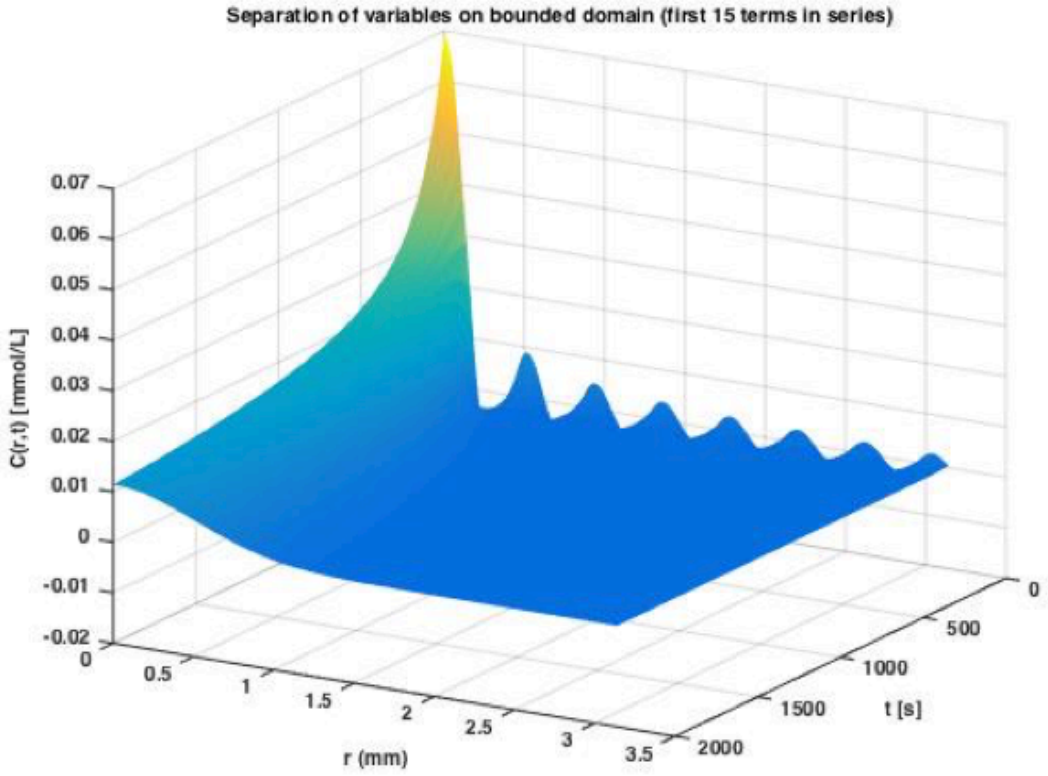
$$A_n = \frac{2C_0}{a^2 J_1^2\left(\sqrt{\lambda_n}a\right)}$$

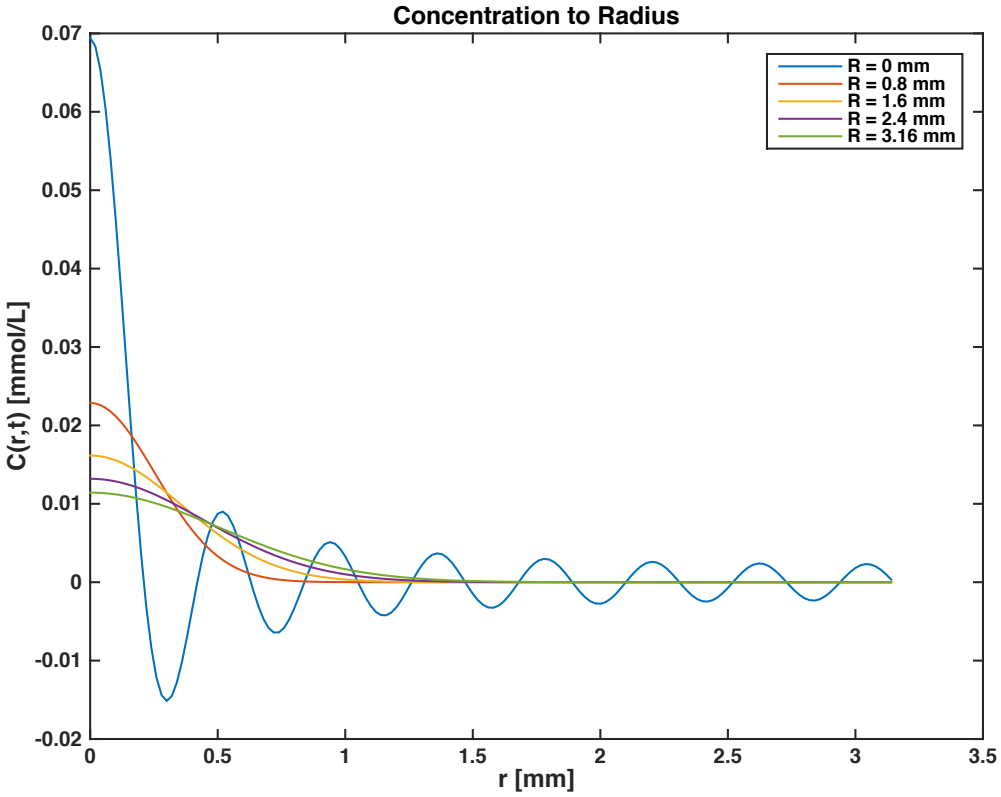
The final solution for the 1D cylindrical diffusion model was ultimately determined once the coefficient was plugged into the general solution. Due to the initial condition containing a delta-dirac function, the final solution is a concentration per unit area.

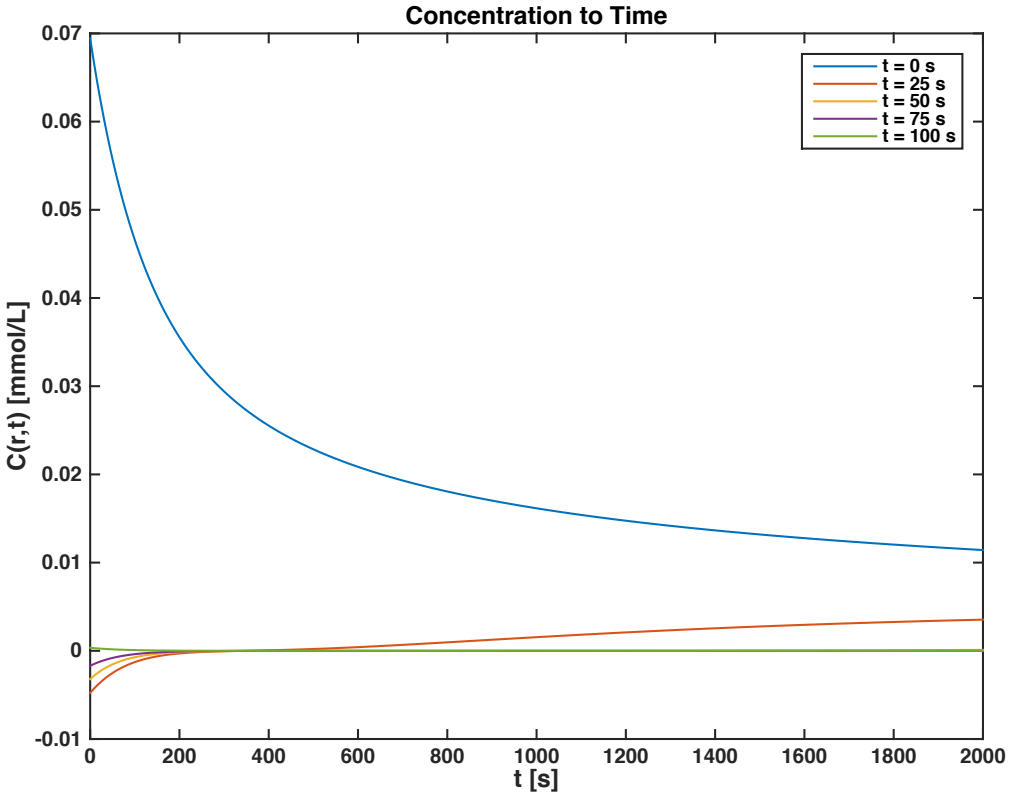
$$C(r,t) = \sum_{n=0}^{\infty} \frac{2C_0}{a^2\left[J_1^2\left(\sqrt{\lambda_n}a\right)\right]} \cdot J_0\left(\lambda_n r\right) e^{-\lambda_n D t}, \quad \lambda_n = \left(\frac{\left(roots_n\right)}{a}\right)^2$$

2.4 Matlab Results

Once the analytical solutions were derived for the 1D Cartesian diffusion model and the 1D cylindrical diffusion model, surface plots for both of the final solutions were generated in Matlab. Both of the solutions used the same constant values for the initial concentration, C_0, the radius of the artery, a, and the diffusivity, D. The Matlab code for both analytical models can be found in Appendix B.

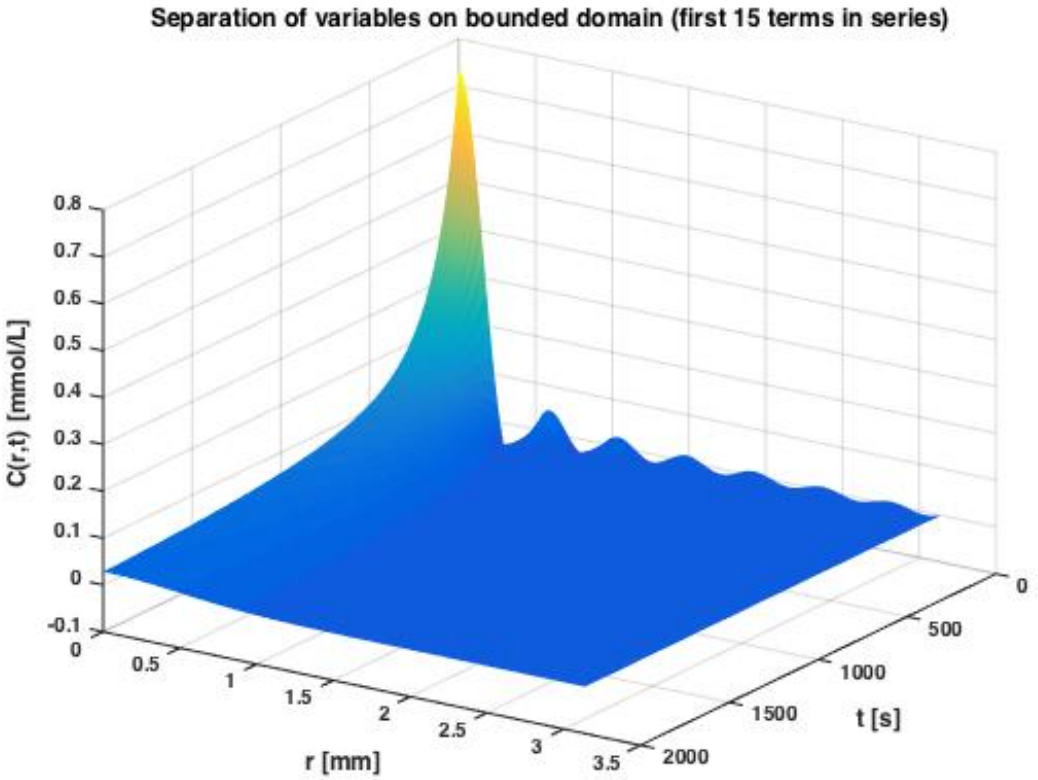


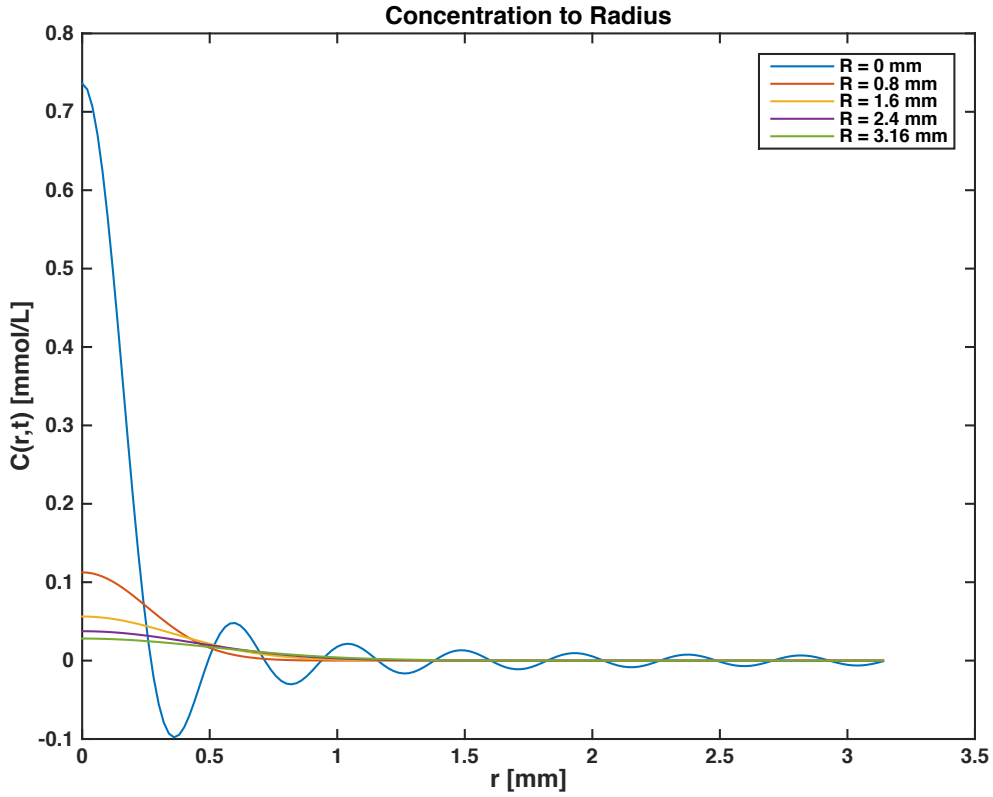

Concentration to Radius
R = 0 mm
R = 0.8 mm
R = 1.6 mm
R = 2.4 mm
R = 3.16 mm
C(r,t) [mmol/L]
r [mm]
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
-0.01
-0.02
0
0.5
1
1.5
2
2.5
3
3.5

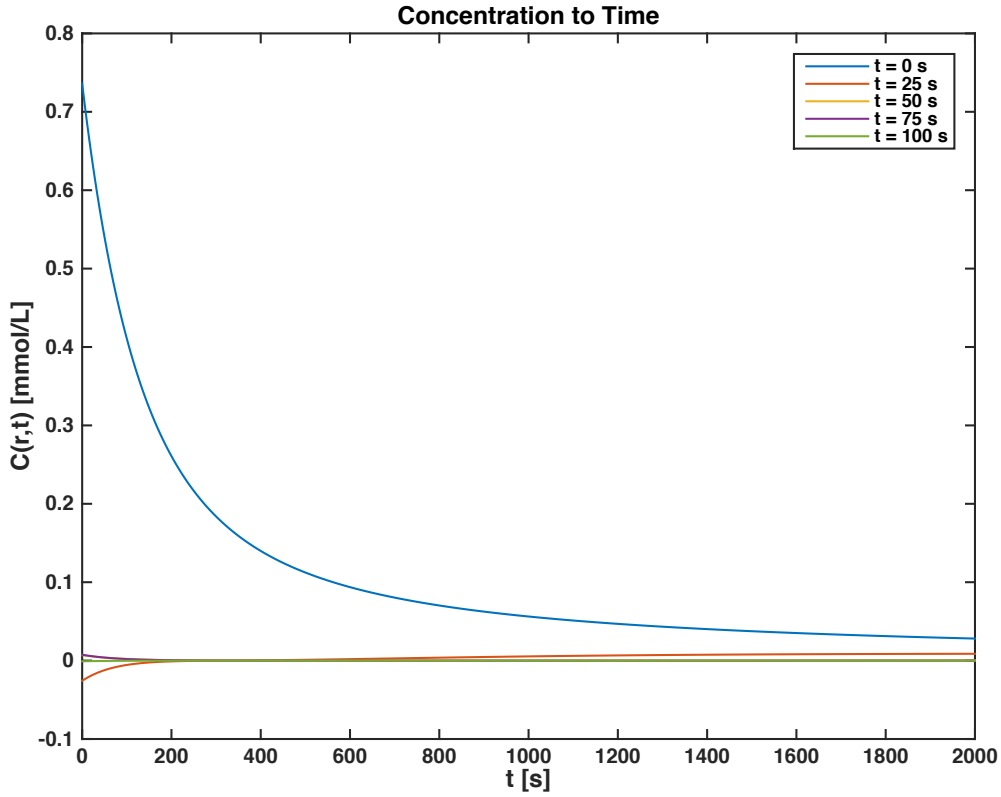

Concentration to Time
t = 0 s
t = 25 s
t = 50 s
t = 75 s
t = 100 s
C(r,t) [mmol/L]
t [s]
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
-0.01
0
200
400
600
800
1000
1200
1400
1600
1800
2000

The three plots shown on the previous pages represent the results for the 1D Cartesian diffusion model. As seen in the surface plot and the plot of concentration to time, the rate at which etorphine diffuses through the arterial wall does not resemble the time it takes to immobilize a stallion or the time it takes for Dexter's victims to become unconscious. In fact at the end of 2,000 seconds, the concentration has yet to reach zero. However, when examining the plot of concentration to radius, the concentration drops off quite quickly and reaches zero when the radius is approximately equal to 1.5 mm.

The next three plots on the following pages are for the 1D cylindrical diffusion model. Similar to the 1D Cartesian diffusion model, the rate at which etorphine diffuses through the arterial wall does not match the values found for a stallion or the details in Dexter. At 2,000 seconds, the concentration of etorphine has not completely reached zero, but it is closer to zero compared to the 1D Cartesian diffusion case. Also similar to before, the plot of concentration to radius revealed that the concentration of etorphine has reached zero when the radius is approximately equal to 1 mm, which is a smaller radius when compared to the Cartesian model.

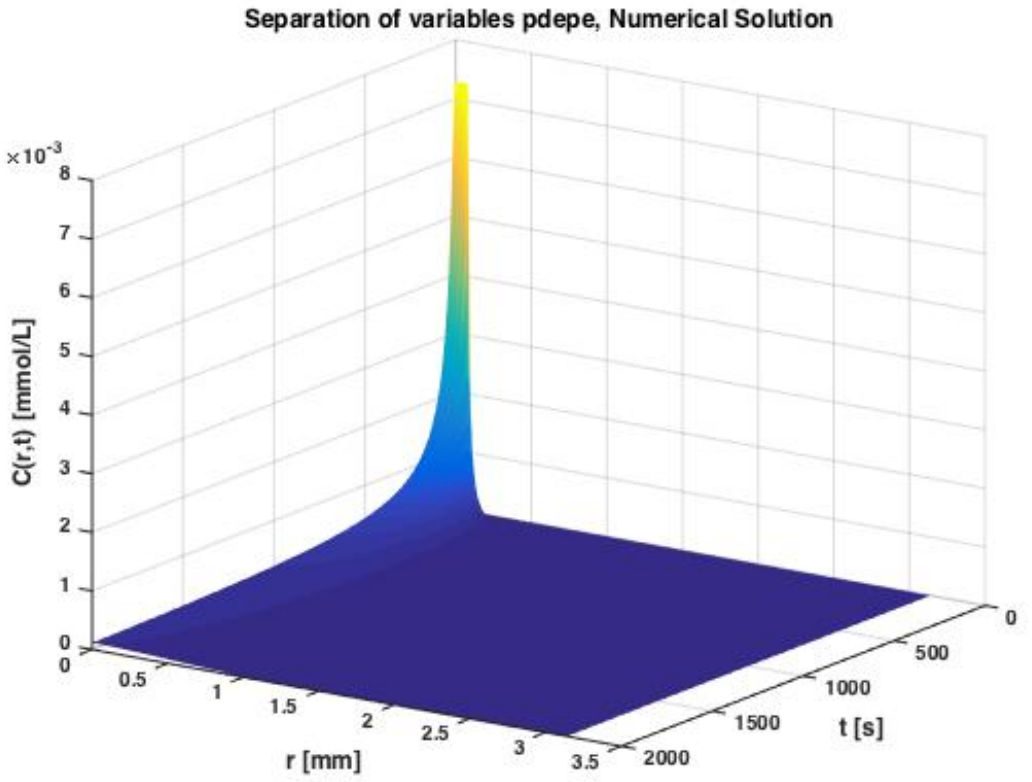


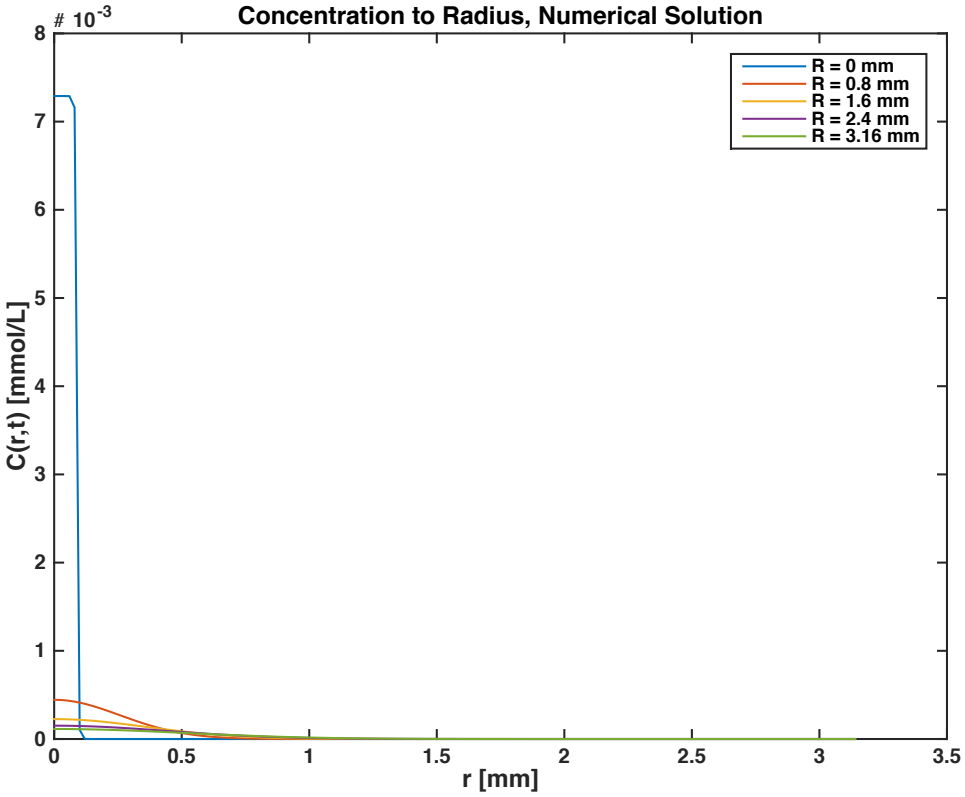

Concentration to Radius
R = 0 mm
R = 0.8 mm
R = 1.6 mm
R = 2.4 mm
R = 3.16 mm
C(r,t) [mmol/L]
r [mm]
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
0.5
1
1.5
2
2.5
3
3.5

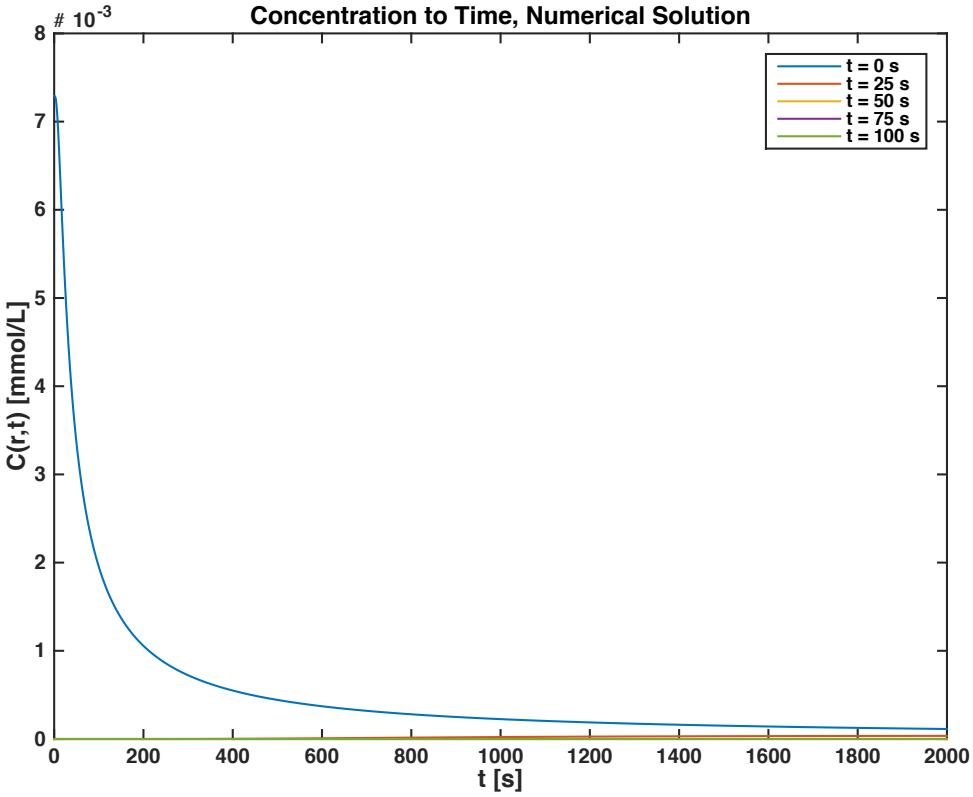

Concentration to Time
t = 0 s
t = 25 s
t = 50 s
t = 75 s
t = 100 s
C(r,t) [mmol/L]
t [s]
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
200
400
600
800
1000
1200
1400
1600
1800
2000

3. Matlab pdepe Numerical Solution

The numerical solution for both the 1D Cartesian diffusion model and the 1D cylindrical diffusion model were identical since the variables entered into the pdepe Matlab function do not change between each model. The Matlab code can be found in Appendix B. Based on the results from the pdepe Matlab function, etorphine diffuses out of the arterial wall in approximately 2,000 seconds, or about 30 minutes. This time does not match the time it takes to immobilize a stallion, and the time does not reflect the events in the show *Dexter*. In the show, Dexter's victims become immediately unconscious upon injection, but these results do not demonstrate the same instantaneous effect of the drug. Although, the majority of the drug diffuses rapidly, the dosage may not be enough to immobilize an individual immediately. Similar to the analytical models for both cases, the plot of concentration to radius showed that the concentration of etorphine is effectively zero when the radius is approximately 0.1 mm. With similar results in the analytical and numerical solutions, it is likely that the slow diffusion of etorphine through the artery wall and limited radial diffusion would indicate the presence of another force acting on the drug which is ultimately dominating the simple diffusion.




Concentration to Radius, Numerical Solution
# 10^-3
R = 0 mm
R = 0.8 mm
R = 1.6 mm
R = 2.4 mm
R = 3.16 mm
C(r,t) [mmol/L]
r [mm]
0
1
2
3
4
5
6
7
8
0
0.5
1
1.5
2
2.5
3
3.5


Concentration to Time, Numerical Solution
# 10^-3
t = 0 s
t = 25 s
t = 50 s
t = 75 s
t = 100 s
C(r,t) [mmol/L]
t [s]
0
1
2
3
4
5
6
7
8
0
200
400
600
800
1000
1200
1400
1600
1800
2000

4. Discussion

4.1 Comparison of Analytical Models

To begin analyzing the results, we first focused on comparing the 1D Cartesian diffusion model and the 1D cylindrical diffusion model. Using the first 15 series for both of the surface plots, the overall shape and behavior of both plots were the same. However, the rate of diffusion of etorphine was slightly different in the two cases. For the 1D cylindrical diffusion model, the etorphine diffused slightly faster over a smaller radius value compared to the 1D Cartesian diffusion model. This difference is mostly likely due to the fact that a cylindrical model is a better representation of the system we were attempting to model. However, with both cases, the initial concentrations did not match the constant we had originally chosen. This result is probably due to the method being used to plot the analytical solutions since the chosen number of terms in the series has a large effect on the initial concentration value.

4.2 Comparison of Analytical and Numerical Models

After comparing the two analytical models, we wanted to use the pdepe function in Matlab as a numerical model for another comparison. Overall, the behavior of the analytical solutions and the numerical solutions were fairly similar since the bulk of the etorphine diffused quickly but the remainder took a while longer to completely diffuse out of the arterial wall. The main difference in the two sets of solutions was the presence of oscillations in the analytical models due to trying to fit an impulse with a finite cosine and Bessel function. Additionally, this result can be explained by having to choose the number of terms in the series since a larger number of terms in the series would result in plots that are identical to the pdepe numerical solution. Overall, though, the analytical models were very similar to the numerical solution.

5. Future Investigation

For simplicity, we had made several assumptions for both of our 1D Cartesian and 1D cylindrical model. However, in the real physiological world, we would need to consider some key features related to our assumptions. First, we had assumed that our artery was a perfect cylinder in shape. This is not true, and the natural artery rarely has a perfect shape. In this case, the radius that we had used would change throughout our system. Therefore, in order to make our model more accurate, we would need to make the shape of our artery more irregular, and change in diffusion with respect to the radius value would need to be modeled to understand the overall diffusion pattern. Second, we did not consider convection in our model, and this is an essential process that needed to be taken into account when dealing with any drug transfer and delivery inside the artery. For our models (both Cartesian and

cylindrical), we assumed that diffusion was the dominating force causing etorphine to diffuse out of the arterial walls. In reality, though, the transport of the drug in the artery is largely due to the bulk motion of the blood flow. For large arteries, convection is the dominating term when investigating drug delivery. However, diffusion becomes the dominant term when looking at drug delivery in microarteries. In effect, to make our model more realistic, we would absolutely have to include the convection term since it would be unrealistic to assume that Dexter is able to inject his victims in their microarteries every single time. In addition, we neglected the thickness of the artery. Physiologically, the thickness could have affected the rate of diffusion and the whole pattern of etorphine diffusion through the wall. Furthermore, just like that the artery is rarely perfect in shape, the injection of the drug would vary depending on each situation. It would impossible for Dexter to control the injection site only at the center of the artery. This variation of the injection site would prevent us from excluding the theta component of the cylindrical model since the diffusion pattern would no longer by symmetrical. However, by including the angular component, we would have a more realistic and concise diffusion model of the drug. For future analysis, the inclusion of the convection term, the transport along the z-direction (the direction of blood flow), the consideration of the artery wall's thickness and the variation of the injection starting site, can all provide a much more accurate model of the diffusion of etorphine.

6. Conclusion

Although it is entertaining to investigate the degree of factual science in a TV series, understanding the diffusion behavior of a drug can be the difference between life and death for a patient. For anesthesiologists, their knowledge of the diffusion of anesthetics is crucial to ensure that a patient is unconscious for an entire procedure but can also recover once the procedure is completed. If an anesthesiologist failed to understand the drug's behavior, a patient could wake up in the middle of a surgery or possibly even die from too high of a dosage. In the case of *Dexter*, though, we can conclude that etorphine likely does not have an instantaneous effect on its victims as portrayed in the show, and it is possible that Dexter would be administering a lethal dose to his patients. Additionally, Dexter would have to administer the antidote for etorphine in order to wake up his victims before finally killing them.

7. References

1. Bentley KW, Hardy DG. Novel analgesics and molecular rearrangements in the morphine-thebaine group. 3. Alcohols of the 6,14-endo-ethenotetrahydrooripavine series and derived analogs of N-allylnormorphine and -norcodeine". *J. Am. Chem. Soc.* **89** (13): 3281–92, 1967.
2. Plotka ED, Seal US, Eagle TC, Asa CS, Tester JR, Siniff DB. Rapid reversible immobilization of feral stallions using etorphine hydrochloride, xylazine hydrochloride and atrophin sulfate. J Wildl Dis. 23:471-478, 1987.
3. Riviere JE, Papich MG. Veterinary Pharmacology and Therapeutics. John Wiley & Sons, 2013, pg. 323.
4. Wikipedia. Etorphine. http://en.wikipedia.org/wiki/Etorphine#cite_note-pmid6042764-1.
5. Krejza J, Arkuszewski M, Kasner S, Weigele J, Ustymowicz A, Hurst R, Cucchiara B, Messe S. Carotid artery diameter in men and women and the relation to body and neck size. Stroke. 37:1103-1105, 2006.

8. Appendix

Appendix A: Calculations for Constants

- Initial Concentration, C_0

$$V = 5\ cm^3$$
$$Dose = 0.015\ mcg$$
$$Concentration = \frac{0.015\ mcg}{5\ cm^3} = 3\times10^{-6}\ g/L$$
$$MW = 411.53\ \ g/mol$$
$$C_0 = \frac{Concentration}{MW} = \frac{3\times10^{-6}\ g/L}{411.53\ g/mol}$$
$$C_0 = 7.29\times10^{-9}\ mol/L$$

- Radius of artery, a

$$a = \frac{1}{2}\left(\frac{r_{carotid,male} + r_{carotid,female}}{2}\right)$$
$$a = \frac{1}{2}\left(\frac{6.5\ mm + 6.1\ mm}{2}\right)$$
$$a = 3.15\ mm = 0.00315\ m$$

- Diffusivity, D (Stokes-Einstein equation)

$$D = \frac{k_B T}{6\pi\eta r}$$
$$k_B = 1.38\times10^{-23}\ J/K$$
$$T = 310\ K$$
$$\eta = 3.5\times10^{-3}\ Pa\cdot s$$
$$r = 1\times10^{-9}\ m$$
$$D = 6.48\times10^{-11}\ m^2/s = 6.48\times10^{-5}\ mm^2/s$$

Appendix B: Matlab Code

- 1D Cartesian Diffusion Analytical Solution

```
global D
D = 6.48*10^-5;
a = 3.15;

dx = 0.02; % step size in x dimension
dt = 1;  % step size in t dimension
xmesh = 0:dx:a; % domain in x; L/2 = 1
tmesh = 0:dt:2000;  % domain in t
nx = length(xmesh); % number of points in x dimension
nt = length(tmesh); % number of points in t dimension
ns = 15;


% solution on bounded domain using separation of variables
sol_sep = zeros(nt, nx);
for n = 0:ns-1
    k = (2*n+1)*pi/2;
    C = 2*7.29*10^-3; %2C0 %mol/cm^3
    sol_sep = sol_sep + exp(-D*((k/a)^2)*tmesh)' *
cos((k/a)*xmesh)*(C/a);
end


figure(1)
set(surf(tmesh,xmesh,sol_sep'),'linestyle','none')
title(['Separation of variables on bounded domain (first ',
num2str(ns), ' terms in series)'])
xlabel('t [s]')
ylabel('r (mm)')
zlabel('C(r,t) [mmol/L]')
set(gca,'FontSize',10,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',10,'fontWeight','bold')

figure(2)
plot(xmesh, sol_sep(1,:),xmesh, sol_sep(round(nt*(1/4)),:),xmesh,
sol_sep(round(nt*(1/2)),:),xmesh, sol_sep(round(nt*(3/4)),:),xmesh,
sol_sep(nt,:))
title('Concentration to Radius')
xlabel('r [mm]')
ylabel('C(r,t) [mmol/L]')
legend('R = 0 mm','R = 0.8 mm','R = 1.6 mm','R = 2.4 mm','R = 3.16 mm')
set(gca,'FontSize',10,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')

figure(3)
plot(tmesh,
sol_sep(:,1)',tmesh,sol_sep(:,round(nx*(1/4)))',tmesh,sol_sep(:,round(n
x*(1/2)))',tmesh,sol_sep(:,round(nx*(3/4)))',tmesh,sol_sep(:,nx)')
title('Concentration to Time')
xlabel('t [s]')
ylabel('C(r,t) [mmol/L]')
```

```
legend('t = 0 s','t = 25 s','t = 50 s','t = 75 s','t = 100 s')
title('Concentration to Time')
set(gca,'FontSize',10,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
```

- 1D Cylindrical Diffusion Analytical Solution

```
function x = zerobess(funstr,nu,m)
%ZEROBESS Zeros of Bessel functions/derivatives of 1st and 2nd kind.
%   X = ZEROBESS('J',NU,M) finds M positive zeros of the Bessel
function
%   of the 1st kind, J_nu(X). The order NU is a real number in the
range
%   0 to 1e7. Default is NU = 0 and M = 5.
%
%   ZEROBESS('Y',NU,M) - Zeros of the Bessel function the 2nd kind,
Y_nu.
%   ZEROBESS('DJ',NU,M) - Zeros of the derivative of J_nu.
%   ZEROBESS('DY',NU,M) - Zeros of the derivative of Y_nu.
%
%   Examples
%       zerobess('J')
%       zerobess('Y',pi,10)
%       zerobess('DJ',1e-6)
%       zerobess('DY',1e6)
%       tic, x = zerobess('J',25,1e4); toc
%
%   See also BESSELJ, BESSELY

%   2010-02-10  Original ZEROBESS.
%   2010-11-09  Zeros of derivatives added.
%   2011-06-22  Cosmetic changes.
%   2011-12-02  More accurate for zerobess('DJ',nu,1), nu < 0.005.

%   Author: Jonas Lundgren <splinefit@gmail.com> 2010

if nargin < 1 || isempty(funstr), funstr = 'J'; end
if nargin < 2 || isempty(nu), nu = 0; end
if nargin < 3 || isempty(m), m = 5; end

% Check FUNSTR
if ~ischar(funstr)
    message = 'Function name FUNSTR must be a string.';
    error('ZEROBESS:NoString',message)
else
    funstr = upper(funstr);
end

% Check NU
if numel(nu) ~= 1 || ~isfinite(nu) || ~isreal(nu) || nu < 0 || nu > 1e7
    message = 'Order NU must be a real number between 0 and 1e7.';
    error('ZEROBESS:InvalidNU',message)
end

% Check M
if numel(m) ~= 1 || ~isfinite(m) || ~isreal(m) || m < 1 || fix(m) < m
```

```
    message = 'M must be a positive integer.';
    error('ZEROBESS:InvalidM',message)
end

% Set function handle and coefficients
switch funstr
    case 'J'
        fun = @besselj;
        deriv = 0;
        c0 = [  0.1701 -0.6563  1.0355  1.8558
                0.1608 -1.0189  3.1348  3.2447
               -0.2005 -1.2542  5.7249  4.3817 ];
    case 'Y'
        fun = @bessely;
        deriv = 0;
        c0 = [ -0.0474 -0.2300  0.2409  0.9316
                0.2329 -0.8871  2.0165  2.5963
                0.0013 -1.1220  4.3729  3.8342 ];
    case 'DJ'
        fun = @besselj;
        deriv = 1;
        c0 = [ -0.3601 -0.0233  0.0247  0.8087
                0.2232 -0.9275  1.9605  2.5781
                0.0543 -1.2050  4.3450  3.8258 ];
    case 'DY'
        fun = @bessely;
        deriv = 1;
        c0 = [  0.0116 -0.5561  0.9224  1.8212
                0.1766 -1.0713  3.0923  3.2329
               -0.1650 -1.3116  5.6956  4.3752 ];
    otherwise
        message = '''%s'' is not a valid bessel function.';
        error('ZEROBESS:InvalidName',message,funstr)
end

% Initiate zeros
x = zeros(m,1);

% Initial guess for the first three zeros
x0 = nu + c0*(nu+1).^[-1 -2/3 -1/3 1/3]';

% Exceptional case
n = 0;
if strcmp(funstr,'DJ') && nu < 0.8
    % Correction of initial guess
    d0 = [2, 3/2, -1/6, 53/576, -2059/34560, 86183/2073600];
    x0(1) = sqrt(d0*nu.^(1:6)');
    if nu < 0.005
        % Skip Newton-Raphson and use asymptotic value
        x(1) = x0(1);
        n = 1;
    end
end

% Newton-Raphson iterations
x(n+1:3) = newton(fun,deriv,nu,x0(n+1:3));
```

```
n = 3;  % Number of zeros computed
j = 2;  % Number of zeros to compute next
errtol = 0.005;  % Relative error tolerance for initial guess

% Remaining zeros
while n < m

    % Upper bound for j
    j = min(j,m-n);

    % Predict the spacing dx between zeros
    r = diff(x(n-2:n)) - pi;
    if r(1)*r(2) > 0 && r(1)/r(2) > 1
        p = log(r(1)/r(2))/log(1-1/(n-1));
        dx = pi + r(2)*exp(p*log(1+(1:j)/(n-1)))';
    else
        dx = repmat(pi,j,1);
    end

    % Initial guess for zeros
    x0 = x(n) + cumsum(dx);

    % Newton-Raphson iterations
    x(n+1:n+j) = newton(fun,deriv,nu,x0);
    n = n + j;

    % Check zeros
    if ~chkzeros(nu,x(n-j-1:n),deriv)
        message = 'Bad zeros encountered. NU and/or M may be too
large.';
        error('ZEROBESS:BadZeros',message)
    end

    % Relative error
    err = (x(n-j+1:n) - x0)./diff(x(n-j:n));

    % Number of zeros to compute next
    if max(abs(err)) < errtol
        j = 2*j;
    else
        j = 2*find(abs(err) >= errtol,1);
    end

end

% Return
x = x(1:m);


%-------------------------------------------------------------------------
----
function x = newton(fun,deriv,nu,x0)
% Newton-Raphson for Bessel function FUN or FUN' with initial guess X0
x = x0;
c = 8;
for t = 1:10
```

```
    % Newton-Raphson step
    f = fun(nu,x);
    g = nu*f./x;
    df = fun(nu-1,x) - g;
    if deriv == 0
        h = -f./df;
    else
        ddf = (nu*g - df)./x - f;
        h = -df./ddf;
    end
    x = x + h;
    % Convergence criteria
    if all(abs(h) < c*eps(x))
        break
    end
    % Relax convergence criteria
    if t >= 7
        c = 2*c;
    end
end
% Check convergence
if t == 10
    warning('ZEROBESS:Newton','No convergence for Newton-Raphson.')
end
% fprintf('%2d%8d\n',t,numel(x))


%-------------------------------------------------------------------------
----
function s = chkzeros(nu,x,deriv)
% Check sequence of Bessel function zeros
dx = diff(x);
% The spacing dx should decrease except for nu < 0.5
ddx = diff(dx);
if nu < 0.5 && deriv == 0
    ddx = -ddx;
end
% Criteria for acceptable zeros
s = isreal(x) && x(1) > 0 && all(dx > 3) && all(ddx < 16*eps(x(2:end-
1)));



function test(ns)
% Homogeneous PDE: Linear (1-D) Diffusion
%
% ns: number of terms in the infinite series
%

% diffusion constant
global D
D = 6.48*10^-5;
a = 3.15;
C = 7.29*10^-3;

% Find roots of bessel function
J0 = zerobess('J',0,100); %function found online to calculate roots of
```

```
%Bessel function
%http://www.mathworks.com/matlabcentral/fileexchange/26639-zerobess

% domain
dx = 0.02; % step size in x dimension
dt = 1;  % step size in t dimension
xmesh = 0:dx:a; % domain in x; L/2 = 1
tmesh = 0:dt:2000;  % domain in t
nx = length(xmesh); % number of points in x dimension
nt = length(tmesh); % number of points in t dimension
ns = 15;

% solution on bounded domain using separation of variables
sol_sep = zeros(nt, nx);
for n = 1:ns-1
    k = (J0(n)/(a))^2; % L = 2
    An = (2*C)/((a^2)*(besselj(1,sqrt(k)*a))^2);
    sol_sep = sol_sep + exp(-D*k*tmesh)' * An *
besselj(0,sqrt(k)*xmesh);
end

figure(1)
set(surf(tmesh,xmesh,sol_sep'),'linestyle','none')
title(['Separation of variables on bounded domain (first ',
num2str(ns), ' terms in series)'])
xlabel('t [s]')
ylabel('r [mm]')
zlabel('C(r,t) [mmol/L]')
set(gca,'FontSize',10,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')

figure(2)
plot(xmesh, sol_sep(1,:),xmesh, sol_sep(round(nt*(1/4)),:),xmesh,
sol_sep(round(nt*(1/2)),:),xmesh, sol_sep(round(nt*(3/4)),:),xmesh,
sol_sep(nt,:))
title('Concentration to Radius')
xlabel('r [mm]')
ylabel('C(r,t) [mmol/L]')
legend('R = 0 mm','R = 0.8 mm','R = 1.6 mm','R = 2.4 mm','R = 3.16 mm')
set(gca,'FontSize',10,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')

figure(3)
plot(tmesh,
sol_sep(:,1)',tmesh,sol_sep(:,round(nx*(1/4)))',tmesh,sol_sep(:,round(n
x*(1/2)))',tmesh,sol_sep(:,round(nx*(3/4)))',tmesh,sol_sep(:,nx)')
title('Concentration to Time')
xlabel('t [s]')
ylabel('C(r,t) [mmol/L]')

legend('t = 0 s','t = 25 s','t = 50 s','t = 75 s','t = 100 s')
title('Concentration to Time')
set(gca,'FontSize',10,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
```

- Pdepe Numerical Solution

```
function lindiff
% diffusion constant
global D
D = 6.48*10^-5;
L = 3.15;

% domain
xmesh = 0:0.02:L; % domain in x
tmesh = 0:0.1:2000;  % domain in t
lx = length(xmesh);
lt = length(tmesh);

% solution using Matlab's built in "pdepe"
sol_pdepe = pdepe(1,@pdefun,@ic,@bc,xmesh,tmesh);

figure(1)
set(surf(tmesh,xmesh,sol_pdepe'),'Linestyle', 'None')
zlabel('C(r,t)')
title('Separation of variables pdepe, Numerical Solution')
xlabel('t [s]')
ylabel('r [mm]')
zlabel('C(r,t) [mmol/L]')
set(gca,'FontSize',10,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')

figure(2)
plot(xmesh,
sol_pdepe(2,:),xmesh,sol_pdepe(round(lt*(1/4)),:),xmesh,sol_pdepe(round
(lt*(1/2)),:),xmesh,sol_pdepe(round(lt*(3/4)),:),xmesh,sol_pdepe(lt,:))
;
title('Concentration to Radius, Numerical Solution')
xlabel('r [mm]')
ylabel('C(r,t) [mmol/L]')
legend('R = 0 mm','R = 0.8 mm','R = 1.6 mm','R = 2.4 mm','R = 3.16 mm')
set(gca,'FontSize',10,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')

figure(3)
plot(tmesh, sol_pdepe(:,2)',tmesh, sol_pdepe(:,round(lx*(1/4)))',tmesh,
sol_pdepe(:,round(lx*(1/2)))',tmesh,
sol_pdepe(:,round(lx*(3/4)))',tmesh, sol_pdepe(:,lx)')
title('Concentration to Time, Numerical Solution')
xlabel('t [s]')
ylabel('C(r,t) [mmol/L]')
legend('t = 0 s','t = 25 s','t = 50 s','t = 75 s','t = 100 s')
set(gca,'FontSize',10,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')

% function definitions for pdepe:
% --------------------------------------------------------------

function [c, f, s] = pdefun(x, t, u, DuDx)
% PDE coefficients functions
D = 6.48*10^-5;
c = 1;
f = D * DuDx; % diffusion
```

```
s = 0; % homogeneous, no driving term

% -------------------------------------------------------------

function u0 = ic(x)
% Initial conditions function

u0 = zeros(1,length(x));
c0 = 7.29*10^-3;
for i = 1:length(x)
    if (x(i) < 0.1) && (x(i) > -0.2)
        u0(i) = c0;
    end
end  % initial conditions

% -------------------------------------------------------------

function [pl, ql, pr, qr] = bc(xl, ul, xr, ur, t)
% Boundary conditions function

pl = ul; % zero value left boundary condition
ql = 0;  % no flux left boundary condition
pr = 0; % zero value right boundary condition
qr = 1;  % no flux right boundary condition
```