Mathematical and numerical modelling of buckling in a spherical cell due to an osmotic shock

By Jake Bewick

DEN318 Third Year Project (Biomedical Engineering) April 2018

Supervisor: Dr Lorenzo Botto

School of Engineering and Materials Science

Engineering/Materials
Third Year Project
DEN318/MAT500

April 2018

Declaration

This report entitled:

Mathematical and numerical modelling of buckling in a spherical cell due to an osmotic shock

Was composed by me and is based on my own work. Where the work of the others has been used, it is fully acknowledged in the text and in captions to table illustrations. This report has not been submitted for any other qualification.

Name: Jake Bewick

Signed:

Date:

Mathematical and numerical modelling of buckling of a spherical cell due to an osmotic shock.

Jake Bewick

***Abstract*—The flux of water under an osmotic gradient may cause buckling in biological cells. In this report we model a single chondrocyte immersed in both a hypertonic and hypotonic medium as a multi-layer sphere. We implement the FTCS finite different method in MATLAB to solve the parabolic partial differential equation describing the diffusion of water around - and through - the cell. Our numerical result is then comprehensively validated - including a comparison with an existing semi-analytical solution (of which we find $<1\%$ difference in mean intracellular concentration after complete convergence). Further mathematical modelling allows us the simulate the expansion and contraction of the chondrocyte, as well as the buckling pressure acting on the membrane. Our proposed solution can - and has - been used to solve analogous problems of drug delivery and heat transfer.**

CONTENTS

I. List of Symbols

Symbol	Variable	Explanation
Ω	-	Complete modelling domain
Ω_i	-	Intracellular medium
Ω_m	-	Membrane
Ω_e	-	Extracellular matrix
r	-	Radius
R_1	`R_1`	Intracellular/membrane boundary
R_2	`R_2`	Membrane/extracellular boundary
R_∞	`R_inf`	Boundary of modelling domain
D	-	Diffusivity
D_i	`D_i`	Intracellular diffusivity
D_m	`D_m`	Membrane diffusivity
D_e	`D_e`	Extracellular diffusivity
c	-	Concentration
c_{in}	`C_in`	Concentration inside cell
c_{out}	`C_out`	Concentration outside cell
t	-	Time
T	`T`	Simulation time
θ	-	Polar angle
φ	-	Azimuthal angle
u	`u`	u substitution
ϕ	`phi`	ϕ substitution
c_{Ω_i}	-	Concentration in Ω_i
c_{Ω_m}	-	Concentration in Ω_m
c_{Ω_e}	-	Concentration in Ω_e
a_x	`ax`	Laplace of diffusion equation
$\bar{g}_1(s)$	`g_1`	Flux through R_1
$\bar{g}_2(s)$	`g_1`	Flux through R_2
μ_x	`u_x`	μ substitution with D_x
z_{2k-1}	`zk`	Caratheodroy-Fejer poles
f_{2k-1}	`ck`	Caratheodroy-Fejer residuals
q_{R_2}	-	Water flux through R_2
ρ	`rho`	Density of water
σ_b	-	Buckling stress
p_b	-	Buckling pressure
h	-	Thickness
E	`E`	Elastic modulus
ν	`pos`	Possion's ratio

II. Introduction

The uneven distribution of solute across a cell membrane induces a flux of water down the concentration gradient. This movement of water would normally cause cells to either expand or contract, but the activation of specific carrier proteins and channels allow cells to regulate their internal volume [1].

However, if the change in solute concentration is rapid, osmotic homeostasis cannot be maintained. This negatively affects the concentration, regulation and operation of intracellular macromolecules [2] which has led a recent aetiological study [3] to indicate that hyperosmotic stress contributes to a number of human diseases.

In response to an uncontrollably severe osmotic shock cells can even induce apoptosis in an effort of prevent hemolysis [4]. Catastrophic osmotic shock results in the complete failure of the mechanical integrity of the cell, causing either lysis or buckling [5]. For over half a decade this has been induced in cells to collect and study enzymes without destroying their viability [6].

Formulation of a mathematical model is critical to properly understanding - and controlling - osmotic shock. As buckling pressure is proportional to the elastic modulus of a spherical shell [7], working backwards we can use the mathematical model to determine the mechanical properties of a cell. Furthermore any derived solution can also easily be applied to problems of drug delivery [8] and heat transfer [9].

A. The Need for a Numerical Solution

For over 70 years analytical solutions have existed to solve diffusion through spherical shells [10]. So why is a numerical solution needed?

Unfortunately, older analytical solutions only consider a single spherical shell [10]. The diffusivity of water across the cell membrane is much lower than through other cellular regions [11], as such a more modern multi-layer model is required [8], [12], [13]. These analytical solutions are often very computationally expensive, either continuing [10] to rely on the truncation of an infinite series [13], or Monte Carlo methods [12].

All existing analytical solutions are also reliant on axisymmetric spherical geometry. Cells are capable of complex shapes, necessitating a numerical approach [14].

B. The Finite Difference Method

We will find a parabolic partial differential equation describes how the concentration of water varies as a function of time and radius. We opt to use finite differences to approximate the solution, using the first-order forward Euler method in time and the second-order central difference in space. This scheme is known as the Forward-Time Central-Space (FTCS) FDM.

The FTCS is an explicit method, meaning there is no need to solve a system of linear equations [14] It is therefore less computationally expensive than using an implicit scheme such as the Crank-Nicolson. However, the FTCS is only conditionally stable when applied to a parabolic PDE, a conditional check in the MATLAB code ensures stability.

III. Mathematical Formulation

A. Modelling a Cell

We first model a cell and the surrounding medium to be a closed ball, Ω, centred at $\boldsymbol{x}$ with a radius of R. We can formally state this as:

Let $R > 0$ and $\boldsymbol{x} \in \mathbb{R}^3$ where $\boldsymbol{x} = (x_1, x_2, x_3)$. The 3-dimensional closed ball centred at $\boldsymbol{x}$ with radius R denoted $\Omega(\boldsymbol{x}, R)$ is the set of points $\boldsymbol{y} \in \mathbb{R}^3$ such that:

$$\Omega(x, R) = \{\boldsymbol{y} \in \mathbb{R}^3 \ \ | \ ||\boldsymbol{y} - \boldsymbol{x}|| \leq R\} \tag{1}$$

This closed ball is divided into three domains: Ω_i, Ω_m and Ω_e, representing the intracellular medium, membrane and extracellular matrix respectively. We define radius R_1 to delineate the boundary between Ω_i and Ω_m, radius R_2 to mark the Ω_m/Ω_e boundary, and R_∞ to represent the radius, R of the original ball, Ω. We require that $0 < R_1 < R_2 \ll R_\infty$. Note R_∞ must be significantly larger than R_2 so that far away from the cell the concentration gradient is 0, we consider R_∞ semi-infinite. All radii and domains have been shown in Figure 1.

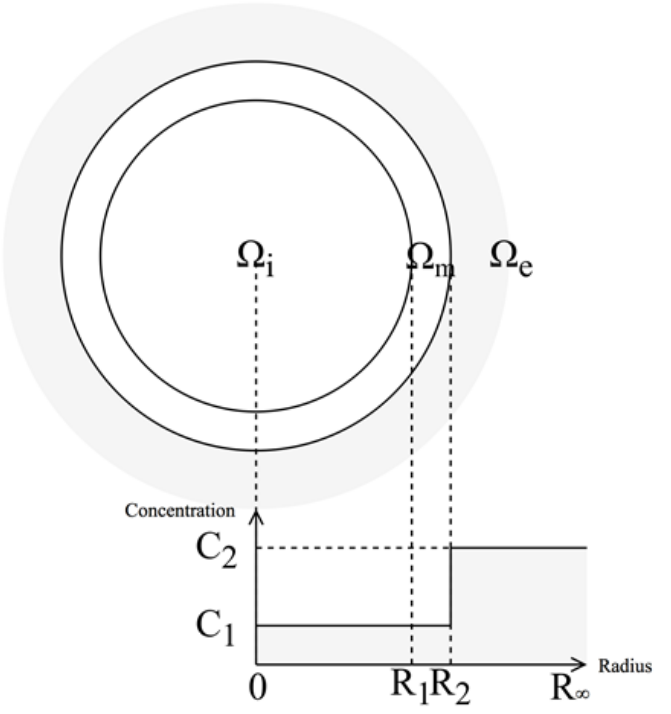

Fig. 1. Deconstructing a spherical cell into a multilayer sphere consisting of three contiguous domains.

TABLE I
DOMAIN CONSTRAINS AND INITIAL CONDITIONS

Region	Domain	Constraints	Diffusivity	Concentration
Intracellular	Ω_i	$0 < r \leq R_1$	D_i	C_{in}
Membrane	Ω_m	$R_1 < r \leq R_2$	D_m	C_{in}
Extracellular	Ω_e	$R_2 < r \leq R_\infty$	D_e	C_{out}

Each domain has a distinct and uniform diffusivity and initial concentration assigned to it, all show in Table I.

Before continuing to derive the governing equations we first must state the assumptions that limit and define this model:

1) There exists a perfect contact between each domain. Membrane effects such as the partition coefficient are ignored [15].
2) The assigned concentration is uniform across any domain when $t = 0$s.
3) The assigned diffusivity is uniform across any domain at all times.
4) The flux of water is dominated by diffusion. Therefore the cell's attempts to restore osmotic homeostasis are negligible.
5) Both concentration and mass flux are conserved between adjacent layers at all times.

B. Numerical Solution

Fick's second law can be used to model diffusion within any one domain [15]:

$$\frac{\partial c}{\partial t} = D\nabla^2 c \tag{2}$$

Given our spherical geometry it is sensible to work in spherical coordinates. Therefore the expanded Laplace operator becomes:

$$\begin{aligned} D\nabla^2 c =& D\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial c}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial c}{\partial\theta}\right)\right. \\ &\left. + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 c}{\partial\varphi^2}\right] \end{aligned} \tag{3}$$

Recall we model the cell as spherically symmetrical, so that concentration does not change along the polar or azimuthal angles:

$$0 = \frac{\partial c}{\partial\theta} = \frac{\partial c}{\partial\varphi} \tag{4}$$

So that with the removal of the polar and azimuthal terms Equation 3 reduces to:

$$\begin{aligned} D\nabla^2 c =& D\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial c}{\partial r}\right) + \cancel{\frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial c}{\partial\theta}\right)}\right. \\ &\left. + \cancel{\frac{1}{r^2\sin^2\theta}\frac{\partial^2 c}{\partial\varphi^2}}\right] \end{aligned} \tag{5}$$

So that the partial differential equation we must solve within each region is:

$$\frac{\partial c}{\partial t} = D\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial c}{\partial r}\right)\right] \tag{6}$$

This is rearanged and stated mathematically as:

$$\left.\begin{aligned} \frac{\partial c}{\partial t} &= D\left(\frac{\partial^2 c}{\partial r^2} + \frac{2}{r}\cdot\frac{\partial c}{\partial r}\right) && \text{(7a)}\\ c = c_{in} \text{ and } D = D_i &\text{ when } 0 < r < R_1 && \text{(7b)}\\ c = c_{in} \text{ and } D = D_m &\text{ when } R_1 < r < R_2 && \text{(7c)}\\ c = c_{out} \text{ and } D = D_e &\text{ when } R_2 < r < R_\infty && \text{(7d)} \end{aligned}\right\}$$

IV. NUMERICAL DISCRETISATION

A. Analytical Solution

There is a convenient [16] substitution often used in literature to simplify Equation 7a. By substituting in $u = cr$ we find that Equation 7a reduces into a form analogous to the diffusion equation in one dimension [17].

$$\left.\begin{aligned} \frac{\partial u}{\partial t} &= D\frac{\partial^2 u}{\partial r^2} && \text{(8a)}\\ u = c_{in}r \text{ and } D = D_i &\text{ when } 0 < r < R_1 && \text{(8b)}\\ u = c_{in}r \text{ and } D = D_m &\text{ when } R_1 < r < R_2 && \text{(8c)}\\ u = c_{out}r \text{ and } D = D_e &\text{ when } R_2 < r < R_\infty && \text{(8d)} \end{aligned}\right\}$$

We solve this PDE using the Forward-Time Central-Space (FTCS) Finite Difference Method (FDM), ensuring we conserve both concentration and flux across the boundaries [18]. Equation 8a becomes:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = D\frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta r^2} \tag{9}$$

Where n represents a time step and i a spatial step. We rearrange and solve for u_i^{n+1}:

$$u_i^{n+1} = u_i^n + \phi\left(u_{i-1}^n - 2u_i^n + u_{i+1}^n\right) \tag{10}$$

Where:

$$\phi = \frac{D\Delta t}{\Delta r^2} \tag{11}$$

As previously mentioned the FTCS is only conditionally stable when $\phi \leq \frac{1}{2}$.

B. Numerical Solution

Carr and Pontrelli's semi-analytical solution [8] has been implemented in MATLAB and used to validate my numerical alternative. Put succinctly they consider concentration in each layer as a function of time and radius, using the Laplace transform to solve Equation 7a.

$$\begin{aligned} c_{\Omega_i}(r,t) =& \mathcal{L}^{-1}\{\bar{c}_{\Omega_i}(r,s)\} \\ c_{\Omega_m}(r,t) =& \mathcal{L}^{-1}\{\bar{c}_{\Omega_m}(r,s)\} \\ c_{\Omega_e}(r,t) =& \mathcal{L}^{-1}\{\bar{c}_{\Omega_e}(r,s)\} \end{aligned} \tag{12}$$

Where $c_\Omega(r,t)$ is the concentration in a specific region as a function of radius and time, and $\bar{c}_\Omega(r,s)$ is the concentration in a region as a function of radius and frequency. The Laplace transform then expands to:

$$\begin{aligned} \mathcal{L}^{-1}\{\bar{c}_{\Omega_i}(r,s)\} =& c_{in} + \mathcal{L}^{-1}\{a_1(r,s)\bar{g}_1(s)\} \\ \mathcal{L}^{-1}\{\bar{c}_{\Omega_m}(r,s)\} =& c_{in} + \mathcal{L}^{-1}\{a_2(r,s)\bar{g}_1(s)\} + \\ & \mathcal{L}^{-1}\{a_3(r,s)\bar{g}_2(s)\} \\ \mathcal{L}^{-1}\{\bar{c}_{\Omega_e}(r,s)\} =& c_{out} + \mathcal{L}^{-1}\{a_4(r,s)\bar{g}_2(s)\} \end{aligned} \tag{13}$$

Where $\bar{g}_1(s)$ and $\bar{g}_2(s)$ represent the mass flux through R_1 and R_2 respectively, and:

$$\begin{aligned}
a_1(r,s) &= -\frac{R_0^2 sinh(\mu_i(s)r)}{rD_0[cosh(\mu_i(s)R_0)\mu_i(s)R_0 - sinh(\mu_i(s)R_0)]} \\
a_2(r,s) &= \frac{\begin{array}{r} R_0^2[\mu_m(s)R_1 cosh(\mu_m(s)(r-R_1)) \\ + sinh(\mu_m(s)(r-R_1))] \end{array}}{\begin{array}{r} r[D_1\mu_m(s)\Delta R_1 cosh(\mu_m(s)\Delta R_1) \\ + (sR_1R_0 - D_1)sinh(\mu_m(s)\Delta R_1)] \end{array}} \\
a_3(r,s) &= -\frac{\begin{array}{r} R_1^2[\mu_m(s)R_0 cosh(\mu_m(s)(r-R_0)) \\ + sinh(\mu_m(s)(r-R_0))] \end{array}}{\begin{array}{r} r[D_1\mu_m(s)\Delta R_1 cosh(\mu_m(s)\Delta R_1) \\ + (sR_1R_0 - D_1)sinh(\mu_m(s)\Delta R_1)] \end{array}} \\
a_4(r,s) &= \frac{R_1^2 \exp(-\mu_e(s)(r-R_1))}{rD_2[1+\mu_e(s)R_1]}
\end{aligned} \tag{14}$$

Where:

$$\begin{aligned}
\Delta R_1 &= R_2 - R_1 \\
\mu_i &= \sqrt{s/D_i} \\
\mu_m &= \sqrt{s/D_m} \\
\mu_e &= \sqrt{s/D_e}
\end{aligned} \tag{15}$$

Carr and Pontrelli use the following quadrature formula, which was suggested by Trefethen, Weideman and Schmelzer [19], to approximate the inverse Laplace transform of all Equations 13:

$$\mathcal{L}^{-1}\{a_x(r,s)\bar{g}_j(s)\} \approx -2\Re\left\{\sum_{k=1}^{N/2} f_{2k-1}\frac{a_x(r,s_k)\bar{g}_j(s_k)}{t}\right\} \tag{16}$$

Where:

- $\Re\{\cdot\}$ is the real part of a complex number
- N = 14 [8]
- z_{2k-1} are the poles found from the Caratheodroy-Fejer method
- f_{2k-1} are the residuals found from the Caratheodroy-Fejer method
- t is the time to evaluate at

However we must first solve for $\bar{g}_j(s_k)$ at $j = 0$ and $j = 1$ when $s_k = z_{2k-1}$. This is accomplished using Gaussian elimination to solve an equation of the form $\mathbf{Ax} = \mathbf{b}$, where:

$$\begin{aligned}
\mathbf{A} &= \begin{bmatrix} a_1(R_1,s_k) - a_2(R_1,s_k) & -a_3(R_1,s_k) \\ a_2(R_2,s_k) & a_3(R_2,s_k) - a_4(R_2,s_k) \end{bmatrix} \\
\mathbf{x} &= \begin{bmatrix} \bar{g}_1(s_k) \\ \bar{g}_2(s_k) \end{bmatrix} \\
\mathbf{b} &= \begin{bmatrix} (c_{in} - c_{in})/s_k \\ (c_{out} - c_{in})/s_k \end{bmatrix}
\end{aligned} \tag{17}$$

C. The Change in a Cell's Radius Over Time

Both the proposed numerical solution and Carr and Pontrelli's semi-analytical solution produce values for concentration, $c(r,t)$. Numerical differentiation allows us to then derive the concentration gradient, $\left(\frac{\partial c}{\partial r}\right)$. Ficks first law [15] then relates the concentration gradient to the mass flux, q through the membrane-extracellular boundary

$$q_{R_2} = 4\pi(R_2)^2 D\left(\frac{\partial c}{\partial r}\right)_{r=R_2} \tag{18}$$

Epstein and Plesset consider an analogous problem in *"On the Stability of Gas Bubbles in Liquid-Gas Solution"* and derive the mass flow through a spherical surface [16]:

$$q_{R_2} = 4\pi(R_2)^2 \rho\left(\frac{dR}{dt}\right) \tag{19}$$

Where R is the radius of the cell membrane ($R = R_2$ at time $t = 0$) and ρ is the density of water. Substituting Equations 18 and 19 together we get an equation explaining radial change as a function of the concentration gradient at the cell membrane:

$$\frac{dR}{dt} = \frac{D}{\rho}\left(\frac{\partial c}{\partial r}\right)_{r=R_2} \tag{20}$$

D. Buckling of a Spherical Shell

Love's classical theory of spherical shell buckling was the first attempt to analyse shell structures [20]. It is a linear model that, despite its drawbacks [21], can still be used to model buckling of our cell membrane. Where buckling stress is σ_b, the buckling pressure is p_b, radius is R and thickness, h buckling stress can then be defined by [7]:

$$\sigma_b = \frac{p_b R}{2h} \tag{21}$$

Love's theory then gives:

$$\sigma_b = \frac{Eh/R}{\sqrt{3(1-\nu^2)}} \tag{22}$$

Where E is the elastic modulus of the cell membrane and ν is its Poisson's ratio. We can combine Equations 21 and 22 to find that:

$$p_b = \frac{2E\left[\frac{h}{R}\right]^2}{\sqrt{3(1-\nu^2)}} \tag{23}$$

V. Computational Implementation

A. Initial Set-up

All programming was done in MATLAB. First we clear the workspace:

```
%% Clearing the Workspace

clc;
clear;
close all;
```

Then we can begin with the spatial discretisation. Start by assigning the three radii, recall that we require $0 < R_1 < R_2 \ll R_\infty$

```
%% Spatial Discretization

R_1   = 5e-6;
R_2   = 5.008e-6;
R_inf = 10e-6;
```

From there we assign the number of spatial steps, `nr`, from this we can calculate exactly how many step are required until reaching the boundaries `R_1` or `R_2`:

```
nr    = 50;
nr_R1 = round(nr*R_1/R_inf);
nr_R2 = round(nr*R_2/R_inf);
```

Finally we calculate spatial step size, `dr` and use that to create a vector of length `nr` of all spatial steps (note that this a linearly spaced vector, the consequences of this will be referred to later in the report):

```
dr    = R_inf/nr;
r_vec = [0:nr]'*dr;
```

Similarly we can now start on temporal disctetisation by defining the total simulation time, `T`, the number of temporal steps, `nt`, the temporal step size, `dt`, and a vector of all temporal steps, `t_vec` (as is the case with `r_vec`, `t_vec` is a linearly spaced vector):

```
T     = 1/10;
nt    = 100;
dt    = T/nt;
t_vec = [0:nt]'*dt;
```

Now we can start defining some general cell properties and boundary conditions, starting with the concentration, c at $t = 0$, `c_in` when $0 < r < R_2$ and `c_out` when $R_2 < r < R_\infty$:

```
%% Cell Properties

c_in  = 0;
c_out = 100;
```

We define the three diffusivities corresponding to the three regions and define the molar density of water:

```
D_i   = 1.38e-9;
D_m   = 2e-12;
D_e   = 1.2720e-9;
rho   = 0.01802;
```

Recall the FTCS is only conditionally stable when $\phi \leq \frac{1}{2}$. We calculate ϕ for each region and preform a stability check:

```
%% Check Stability

phi_i = (D_i*dt)/(dr*dr);
phi_m = (D_m*dt)/(dr*dr);
phi_e = (D_e*dt)/(dr*dr);

if phi_i>0.5 || phi_m>0.5 || phi_e>0.5
        error('Unstable')
end
```

B. Numerical Solution

Now all variables have been defined and stability has been ensured we can start implementing our finite difference scheme. We first preallocate a concentration matrix $(m \times n)$ where $m =$`nr` and $n =$`nt`. We input boundary conditions into the newly formed matrix:

```
%% FTCS

c_mat =ones(length(r_vec),length(t_vec));
c_mat(1:nr_R2,:) = c_in;
c_mat(nr_R2+1:end,:) = c_out;
```

Now we use `bsxfun` to make the $u = cr$ substitution mentioned earlier:

```
u_mat = bsxfun(@times,c_mat,r_vec);
```

The FTCS scheme has us solving Equation 10 at every element of the matrix `u_mat`. This is accomplished using nested loops, so that we solve along each spatial step before using Euler time stepping to advance forward.

```
for n=1:nt
    for i=2:nr_R1;
        u_mat(i,n+1) =u_mat(i,n) + phi_i
            *(u_mat(i-1,n)+u_mat(i+1,n)
            -2.*u_mat(i,n));
    for i=nr_R1:nr_R2;
        u_mat(i,n+1) =u_mat(i,n) + phi_m
            *(u_mat(i-1,n)+u_mat(i+1,n)
            -2.*u_mat(i,n));
    for i=nr_R2:nr;
        u_mat(i,n+1) =u_mat(i,n) + phi_e
            *(u_mat(i-1,n)+u_mat(i+1,n)
            -2.*u_mat(i,n));
    end
    end
    end
end
```

We once again use `bsxfun` to reverse the u-substitution, arriving at the matrix of concentration we desired.

```
numerical = bsxfun(@rdivide,u_mat,r_vec);
```

C. Semi-Analytical Solution

We now implement Carr and Pontrelli's semi-analytical solution so that we may validate our numerical solution. Start by defining the limit of the summation in Equation 16 where we take $N = 14$ [8]. We use function `cf` created by Trefethen, Weideman and Schmelzer [19] to calculate the poles `zk` and residuals `ck` using the Caratheodroy-Fejer method, which take the form of vectors with length N. We also define ΔR_1 from Equation 15.

```
%% Analytical Solution

N = 14;
[zk,ck] = cf(N);
sk = zk/T;
DR1 = R_2-R_1;
```

As previously mentioned, before implementing the quadrature formula (Equation 16) we must solve for $\bar{g}_1(s_k)$ and $\bar{g}_2(s_k)$ where $s_k = z_{2k-1}/T$ for $k = 1, 2..., N/2$. To do this we use Gaussian elimination to solve Equation 17. We start by opening a `for` loop to iterate through $k = 1, 2..., N/2$ and defining the variable `s`:

```
for k = 1:N/2
    s=sk(2*k-1);
```

We now assign μ_i, μ_m and μ_e from Equation 15:

```
    u_i = sqrt(s/D_i);
    u_m = sqrt(s/D_m);
    u_e = sqrt(s/D_e);
```

Recall from Equation 15 that we must calculate the values of $a_1(r, s_k)$, $a_2(r, s_k)$ and $a_3(r, s_k)$ when $r = R_1$ and $s_k = z_{2k-1}/T$:

```
    r=R_1;
    a1r1=-(R_1^2*sinh(u_i*r))/(r*D_i*(
        cosh(u_i*R_1)*u_i*R_1-sinh(u_i*R_1
        )));
    a2r1=(R_1^2*(u_m*R_2*cosh(u_m*(r-R_2)
        )+sinh(u_m*(r-R_2))))/(r*(D_m*u_m*
        DR1*cosh(u_m*DR1)+(s*R_2*R_1-D1)*
        sinh(u_m*DR1)));
    a3r1=(-R_2^2*(u_m*R_1*cosh(u_m*(r-R_1
        ))+sinh(u_m*(r-R_1))))/(r*(D_e*u_m
        *DR1*cosh(u_m*DR1)+(s*R_2*R_1-D1)*
        sinh(u_m*DR1)));
```

Likewise we calculate the values of $a_2(r, s_k)$, $a_3(r, s_k)$ and $a_4(r, s_k)$ when $r = R_2$ and $s_k = z_{2k-1}/T$:

```
    r=R_2;
    a2r2=(R_1^2*(u_m*R_2*cosh(u_m*(r-R_2)
        )+sinh(u_m*(r-R_2))))/(r*(D1*u_m*
        DR1*cosh(u_m*DR1)+(s*R_2*R_1-D1)*
        sinh(u_m*DR1)));
    a3r2=(-R_2^2*(u_m*R_1*cosh(u_m*(r-R_1
        ))+sinh(u_m*(r-R_1))))/(r*(D1*u_m*
        DR1*cosh(u_m*DR1)+(s*R_2*R_1-D1)*
        sinh(u_m*DR1)));
    a4r2=(R_2^2*exp(-u_e*(r-R_2)))/(r*D2
        *(1+u_e*R_2));
```

We form the matrices **A** and **B** such that we prepare to use Gaussian elimination to solve a problem of the form $\mathbf{Ax} = \mathbf{b}$:

```
    A=[a1r1-a2r1,-a3r1;a2r2,a3r2-a4r2)];
    b=[(c_in-c_in)/s;(c_out-c_in)/s];
```

We use MATLAB's built-in solver for systems of linear equations to find $\bar{g}_1(s_k)$ and $\bar{g}_2(s_k)$ and we terminate the loop:

```
    x(:,k)=A\b;
end
g_1 = x(1,:);
g_2 = x(2,:);
```

We can now use the quadrature formula (Equation 16) to solve for concentration each each region independently. Starting with the intracellular medium we begin a spatial loop bounded from 0 to R_1:

```
for n = 1:nr_R1
```

We then start to loop for all values of $k = 1, 2...N/2$, once again defining `s`, `u_i`, `u_m` and `u_e` (see Equation 15):

```
    for k=1:N/2
        s=sk(2*k-1);
        u_i = sqrt(s/D0);
        u_m = sqrt(s/D1);
        u_e = sqrt(s/D2);
```

We now create a vector containing all terms of the quadrature formula (Equation 16) for $k = 1, 2...N/2$:

```
        quad0(k)=ck(2*k-1)*(((-(R_1^2*
            sinh(u_i*r_vec(n)))/(r_vec(n)*
            D0*(cosh(u_i*R_1)*u_i*R_1-sinh
            (u_i*R_1))))*(g_1(k)))/(T));
```

Simply ending the inner loop and completing the summation returns the value of the inverse Laplace transform:

```
    end
    total0 = sum(quad0);
```

So that we can generate a vector, length `nr`, containing the concentration at any given time, T.

```
    analytical(n) =c_in -2*real(total0);
end
```

Similarly we calculate concentration profiles for the membrane:

```
for n = nr_R1+1:nr_R2
    for k=1:N/2
        s=sk(2*k-1);
        u_i = sqrt(s/D0);
        u_m = sqrt(s/D1);
        u_e = sqrt(s/D2);
        quad1(k)=ck(2*k-1)*((((R_1^2*(u_m
            *R_2*cosh(u_m*(r_vec(n)-R_2))+
            sinh(u_m*(r_vec(n)-R_2))))/(
            r_vec(n)*(D1*u_m*DR1*cosh(u_m*
            DR1)+(s*R_2*R_1-D1)*sinh(u_m*
            DR1))))*(g_1(k)))/(T));
        quad2(k)=ck(2*k-1)*((((-R_2^2*(
            u_m*R_1*cosh(u_m*(r_vec(n)-R_1
            ))+sinh(u_m*(r_vec(n)-R_1))))
            /(r_vec(n)*(D1*u_m*DR1*cosh(
            u_m*DR1)+(s*R_2*R_1-D1)*sinh(
            u_m*DR1))))*(g_2(k)))/(T));
    end
    total1 = sum(quad1);
    total2 = sum(quad2);
    analytical(n) =c_in -2*real(total1)
        -2*real(total2);
end
```

And finally for the extracellular matrix:

```
for n = nr_R2:nr+1
    for k=1:N/2
        s=sk(2*k-1);
        u_i = sqrt(s/D0);
        u_m = sqrt(s/D1);
        u_e = sqrt(s/D2);
```

```
        quad3(k)=ck(2*k-1)*((((R_2^2*exp
            (-u_e*(r_vec(n)-R_2)))/(r_vec(
            n)*D2*(1+u_e*R_2)))*(g_2(k)))
            /(T));
    end
    total3 = sum(quad3);
    analytical(n) =c_out -2*real(total3);
end
```

D. Concentration Gradient, Radial Change, Flux and Buckling Pressure

Now that we know concentration as a function of time and radius we can easily calculate the concentration gradient using MATLAB's built in `gradient` function. Disregard `FX`, we create a concentration gradient matrix, `conc_grad` and then find the concentration gradient when $r = R_2$:

```
%% Concentration Gradient

[FX,conc_grad]=gradient(numerical);
conc_grad_R2 = conc_grad(nr_R2,:);
```

We use the concentration gradient at R_2 to calculate the radial change from Equation 20:

```
%% Radial Change

rad_change = D_m/rho*c_grad_R2;
```

The variable `rad_change` describes the increase or decrease in radius at each time step. To find the running change we calculate its cumulative sum and add it to the initial radius:

```
rad_change = R_2+cumsum(rad_change);
```

We calculate flux from the concentration gradient using Equation 18:

```
%% Flux

q=4*pi*R_2^2*D_m*conc_grad_R2;
```

We define the elastic modulus, `E`, the Possion's ratio, `pos`, and the membrane thickness, `thickness`:

```
%% Buckling Pressure

E=14e6;
pos=0.38;
thickness=R_2-R_1;
```

And calculate buckling from Equation 23:

```
buckling=(2*E*(thickness./rad_change).^2)
    /(sqrt(3*(1-pos^2)));
```

VI. Results and Discussion

We model a single chondrocyte suspended in saline solution such that it forms a spherical shape [22]. We now assign the realistic physical parameters found in Table II to our variables in MATLAB.

TABLE II
PHYSICAL PARAMETERS OF A CHONDROCYTE SUSPENDED IN A SALINE SOLUTION

Parameter	Value	Units	Variable	Source
R_1	5	μm	`R_1`	[23]
R_2	8	nm	`R_2`	[24]
R_∞	$5\times(R_1+R_2)$	μm	`R_inf`	-
D_i	1.36×10^{-9}	m^2s^{-1}	`D_i`	[25]
D_m	2.00×10^{-12}	m^2s^{-1}	`D_m`	[26]
D_e	2.27×10^{-9}	m^2s^{-1}	`D_e`	[27]
C_{in}	0	-	`c_in`	-
C_{out}	100	-	`c_out`	-

Note that R_∞ is a large multiple of $R_1 + R_2$ to fulfil the condition $R2 \ll R_\infty$. We can also observe that the diffusivity of water in the intracellular and extracellular domains are roughly the same, with D_i being 60% of D_e. The greatest impediment to the diffusive flux of water is the cell membrane, with D_m being about 0.15% of D_e. As a final note, for the sake of comparison C_{in} and C_{out} have been normalised.

Before proper analysis of the results we must first validate our provided numerical model. We propose two separate methods to ensure our numerical solution works as intended: first we observe if the simulation behaves as intuitively expected of a cell - validation through stimulation - and then we directly compare it to an existing analytical solution - validation through imitation.

A. Validation Through Stimulation

1) Establishing a Baseline: After entering the variables in Table II into MATLAB we find our numerical solution produces Figure 2:

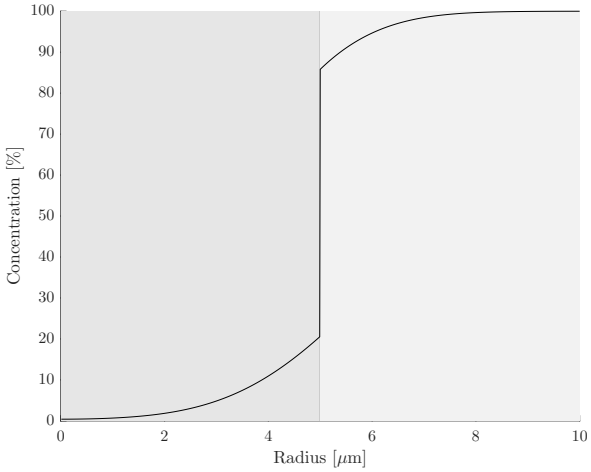

Fig. 2. The concentration in a cell at time $t = 1/1000$s after a net flux of water into the cell. The grey colouring represents the different domains, with intensity proportional to diffusivity, this convention will be used for the remained of the report. No observations will be made about the data - for now we only analyse how this baseline changes in response to altered variables. The data has been produced using the outlined numerical method.

2) Does increasing diffusivity increase water flux into the cell?: For our first experiment we begin by quadrupling the diffusivity of the intracellular medium and the matrix, D_i and D_m respectively. We know from Fick's first law that the diffusive flux of water, J, will be directly proportional to the diffusion coefficient [15]:

$$J = -D\nabla c \tag{24}$$

So that we expect the water to have penetrated further into the cell at any given time.

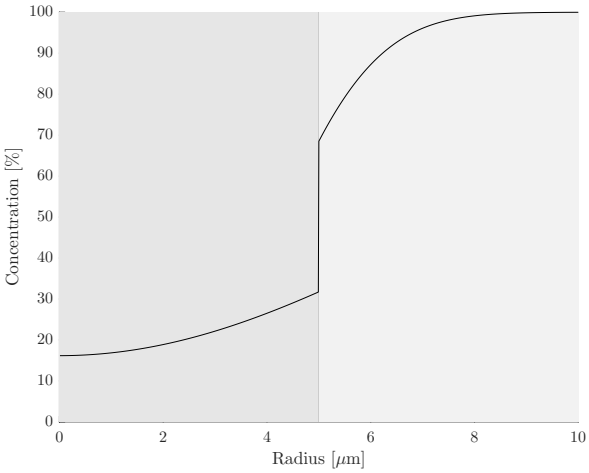

Fig. 3. Numerical simulation of concentration in a cell at time $t = 1/1000$s after a net flux of water into the cell. All variables set according to Table II, except D_i and D_m which have been quadrupled as compared to Figure 2.

Figure 3 clearly indicates this has happened: the concentration of water at the origin has increased from 0.41% to 16.16% and the mean water concentration in Ω_i has increased from 5.57% to 21.57%.

Even more interestingly we are able to note a change in the extracellular domain, Ω_e, even though no associated variable has been modified: the mean concentration of water has decreased from 97.32% to 93.69% due to a local depletion in water molecules at the cell membrane. This demonstrates that the concentration in one domain is not just dependent on its parameters, but also on the parameters of surrounding domains. This indicates our numerical solution correctly solves the three coupled equations that arise from having three contiguous domains, correctly conserving concentration and flux at each boundary.

3) Does the membrane actually have an affect on the simulation?: Recall that the cell membrane, despite being only 5μm thick (Table II), is the greatest impediment of water flux into the intracellular domain. This is because a cell's membrane has a water diffusivity 680 times smaller than the intracellular medium (Table II). So what would happen if this membrane didn't exist? We set $D_m = D_i$ and run the simulation:

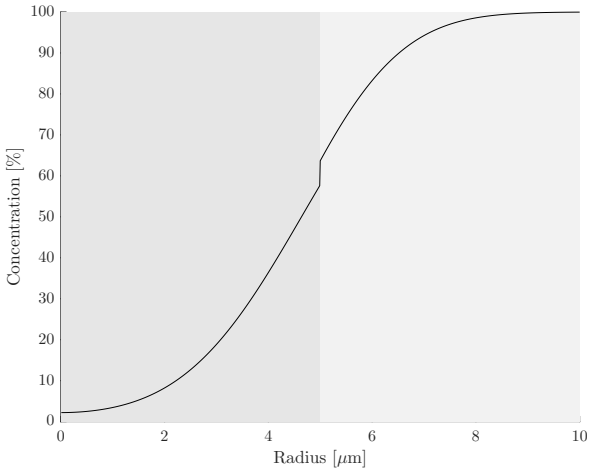

Fig. 4. Numerical simulation of concentration in a cell at time $t = 1/1000$s after a net flux of water into the cell. All variables set according to Table II, except now we have set $D_m = D_i$ so that the membrane effectively no longer exists.

Figure 4 shows mean intracellular concentration has increased from 5.57% in Figure 2 to 19.01%. The concentration is much higher in Ω_i nearer towards the membrane, as compared to Figure 3 which presents a much more flat concentration profile across the domain. This is exactly what we expect from the removal of the flux limiting membrane - water is able to more easily diffuse into the cell, but then is slowed by the lower diffusivity of the intracellular medium. The diffusion limiting domain has effectively switched from Ω_m to Ω_i.

Conversely if we make the membrane completely impermeable we should make any net water flux impossible. Does our solution capture this?

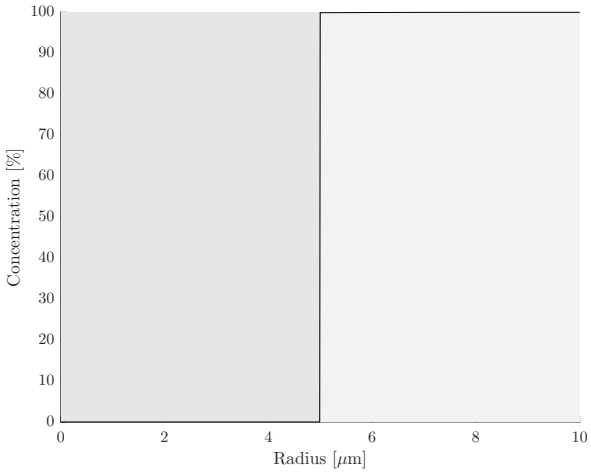

Fig. 5. Numerical simulation of concentration in a cell at time $t = 1/1000$s. All variables set according to Table II, except now D_m is 1000 times smaller than the original value, effectively making it impermeable. As such there is no net flux of water into the cell.

Unsurprisingly we find from Figure 5 that the intracellular concentration does not change from c_{in} and the extracellular concentration does not vary from c_{out}. There has been absolutely no movement of water across the now impermeable membrane.

4) Can we simulate other things?: Recall that earlier in the report we mentioned that any derived solution could also be applied to problems of heat transfer and drug delivery, we now present proof by way of modelling the dissolution of a coated spherical capsule. We set the simulation parameters according to Table III. This is the problem that appears in Carr and Pontrelli's report. [8].

TABLE III
PHYSICAL PARAMETERS OF A DISSOLVING DRUG

Parameter	Value	Units	Variable	Source
R_1	1.5	mm	`R_1`	[8]
R_2	1.7	mm	`R_2`	[8]
R_∞	$5\times(R_1+R_2)$	μm	`R_inf`	-
D_i	30×10^{-11}	m^2s^{-1}	`D_i`	[8]
D_m	5×10^{-11}	m^2s^{-1}	`D_m`	[8]
D_e	30×10^{-11}	m^2s^{-1}	`D_e`	[8]
C_{in}	0	-	`c_in`	-
C_{out}	1	-	`c_out`	-

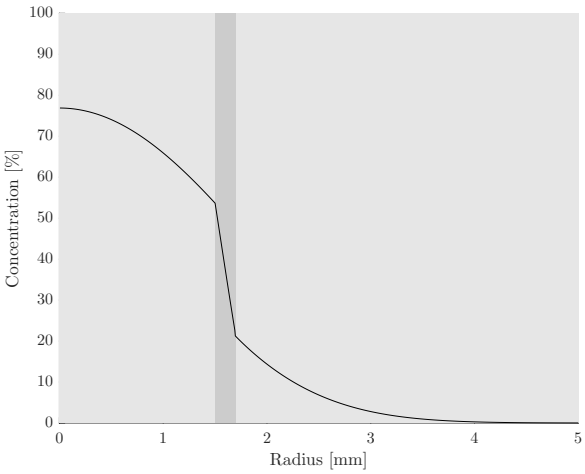

Fig. 6. Numerical simulation of drug dissolution from a coated spherical capsule after 30 minutes. All variables set according to Table III.

From Figure 6 we correctly observe a net flux of solute away from the drug. The thicker *"membrane"* is a good demonstration of how a low diffusivity layer is able to rapidly slow down diffusion, with concentration dropping from 53.66% to just 21.25% over the space of just 2mm. It becomes readily apparent that altering drug coating diffusivity is an effective way of controlling the release of a drug [8].

Running this simulation also proves that our model is capable of handling parameters of varying magnitude. Cells are simulated with dimensions on the order of individual microns, with time measured in thousandths of a second, conversely drug diffusion must deal with millimetres and minutes.

B. Validation Through Imitation

1) Mesh Convergence: For a robust validation of our numerical solution we now compare it against an existing analytical solution. Carr and Pontrelli present their solution in the context of drug delivery [8], so we will continue to simulate the dissolution of a spherical coated capsule with identical parameters to that which they use in their report (see Table III for the complete list).

Up until now no consideration has been given to the mesh width, both spatial and temporal. Mesh size determines the accuracy of our numerical solution, and so we increase element density until mesh convergence has been achieved. We start by setting the number of time steps, `nt`, equal to 10000, and then increment the number of spatial steps, `nr`, until convergence is reached - defined by noting no further change in the mean intracellular concentration despite further mesh refinement. We create a MATLAB script to iterate through the different step sizes and log the mean intracellular concentration - this is then used to produce Figure 7.

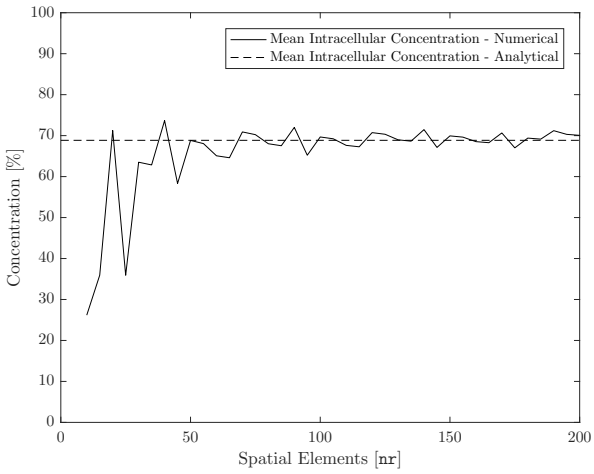

Fig. 7. Mean solute concentration in the domain Ω_i after 30 minutes of drug dissolution. Parameters set according to Table III, see Figure 6 for the concentration profile when mesh convergence has been reached.

Note from Figure 7 that the analytical solution is not dependent on mesh width and remains constant. For the numerical solution intracellular concentration increases with mesh density up until $\mathtt{nr} = 80$. The converged numerical solution has a mean concentration of 70.07% compared to 69.15% for the analytical result, an impressive difference of $< 1\%$. We also note that the numerical data oscillates about this value, this may be due to how the mesh elements are distributed across each domain: the number of elements until `R_1` is defined by the variable `nr_R1`. As `R_1` does not always divide evenly into the total number of elements `nr` we must round it to the closest integer:

```
nr_R1 = round(nr*R_1/R_inf)
```

This rounding introduces an error that is most likely responsible for the variation. The Future Work section will address ways to reduce or prevent this.

C. Concentration

Recall the first objective of our numerical code is to produce the variable `numerical`, a matrix $(m \times n)$ of concentration at each spatial step and time step (such that $m =$`nr` and $n =$`nt`). We are able to plot this entire matrix using the MATLAB function `meshplot`:

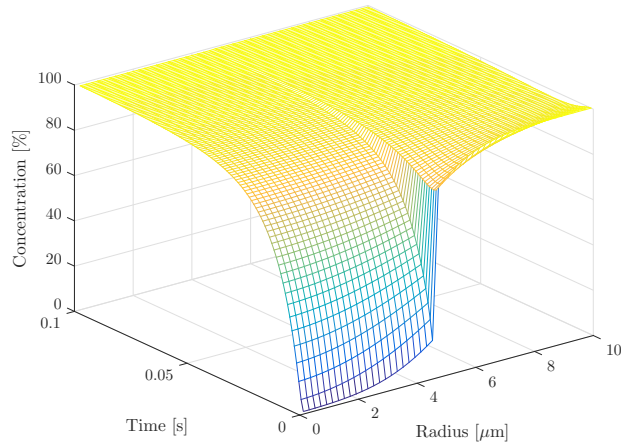

Fig. 8. Mesh plot of the concentration of water within a cell placed in a hypertonic solution. Time increases in increments of 1ms, terminating when $t = 1/10$s. Spatial steps increase in increments of 0.2μm. All simulation parameter set according to II. Radius is only plotted up to 10μm to produce a more readable graph.

The mesh plot - Figure 8 is unintuitive and difficult to read, but does have benefits that warrant its inclusion in the report. It clearly depicts the discretisation used for our numerical solution. Note for now that the meshing is uniform both in space and time - this has significant drawbacks which we address later in the report. Figure 8 also helps us understand how the concentration profile (Figure 2) was derived. If you cut a slice along the time axis at any given point you produce a graph showing how concentration varies with radius at that given time. If we take the slice when $t = 1\mu$s we produce Figure 2. Likewise, if you want to see exactly how concentration varies with time at any given radius we take a slice normal to the radial axis.

D. Mass Flux, Radial Change and Buckling Pressure

Now that we have successfully generated and then validated the concentration matrix (`numerical`) we can easily manipulate it to calculate the water flux into the membrane (Equation 18), the radial change of the cell (Equation 20) and ultimately the buckling pressure (Equation 23) acting on the chondrocyte.

All three have been calculated for a cell immersed in both a hypertonic and hypotonic medium. We set the elastic modulus equal to 14 kPa [28] Poisson's ratio equal to 0.38 [29], both realistic physical parameters for a chondrocyte. For the hypertonic simulation we set $c_{in} = 100\text{mol}\cdot\text{m}^{-3}$ and $c_{out} = 0\text{mol}\cdot\text{m}^{-3}$. For the hypotonic model we reverse these values. Both scenarios have been simulated and plotted on the following page.

Figure 9 correctly suggests that a cell shrinks when surrounded by a hypertonic medium, but it incorrect suggests a negative radius, even when we have already stated r must always be positive (see Equation 1). These values can safely be disregarded however as our simulation also predicts buckling occurring at $t = 1\mu$s (from Figure 11), before the radius turns negative.

Our hypotonic simulation also correctly predicts swelling due to the net influx of water (Figure 12), eventually leading to a sixfold increase in cell size. This in turn causes a large decrease in buckling pressure until equilibrium is reached (Figure 13).

For both hypertonic and hypotonic conditions we know that net water flux should be zero when the concentration gradient is zero (Fick's first law [15]). This has been correctly modelled as shown in Figures 10 and 13.

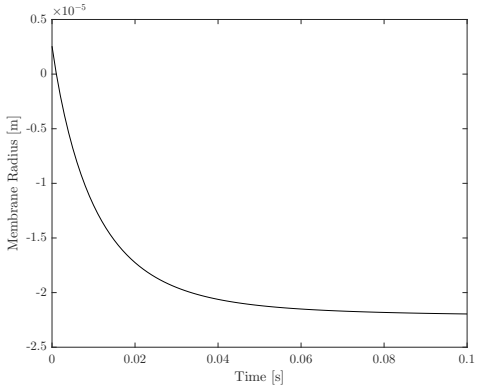

Fig. 9. Numerical simulation of the radial change of a chondrocyte in a hypertonic medium. The cell shrinks as water diffuses out from the cell. Recall we defined that the radius must always be positive, so values below 0 should be neglected.

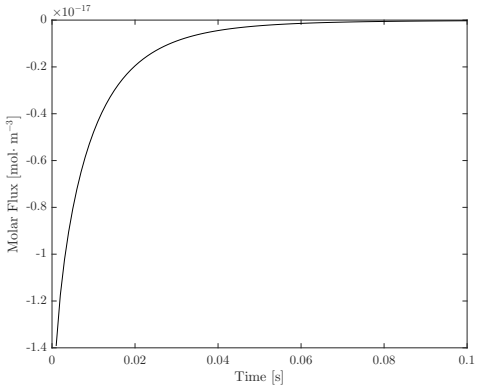

Fig. 10. Net molar flux of water out from a chondrocyte in a hypertonic medium. When equilibrium is reached there is no net water flux.

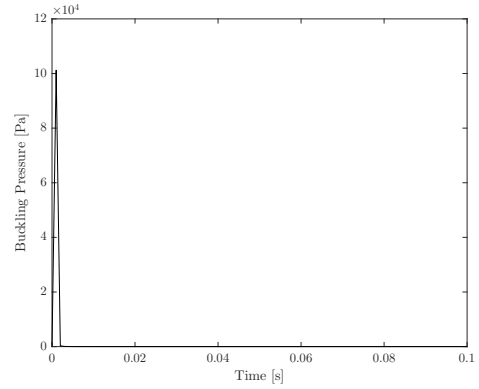

Fig. 11. Buckling pressure on a chondrocyte membrane following a net efflux of water due to hypertonic conditions.

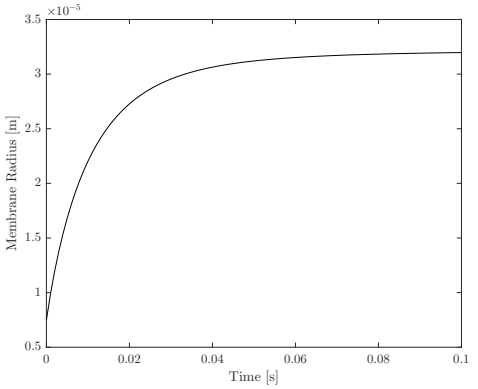

Fig. 12. Numerical simulation of the radial change of a chondrocyte in a hypotonic medium. The cell grows as water diffuses into the cell. When equilibrium has been reached the cell has expanded to 32μm, over a sixfold increase.

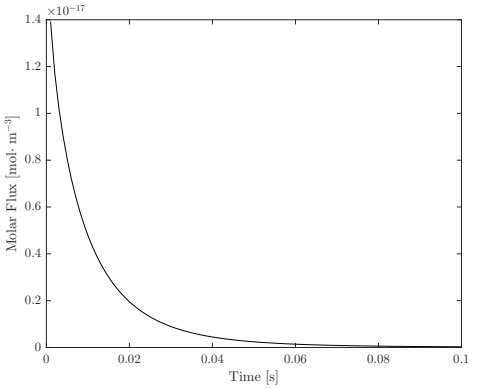

Fig. 13. Net molar flux of water into a chondrocyte in a hypotonic medium. When equilibrium is reached there is no net water flux.

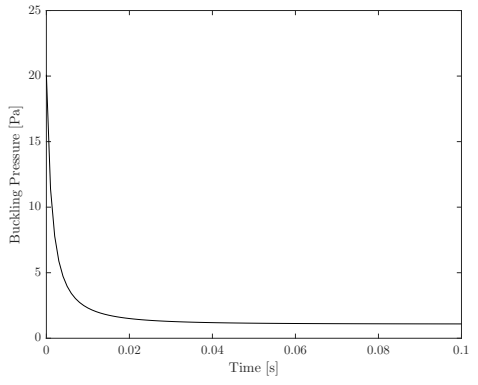

Fig. 14. Buckling pressure on a chondrocyte membrane following a net influx of water due to hypotonic conditions.

VII. Future Work

A. Improve Mesh Quality

Recall from Figure 8 that our numerical discretisation uses a uniformly spaced mesh. This is an incredibly wasteful practice - mesh density should be increased in regions where there is a higher concentration gradient. To visualise this we can actually use the MATLAB function `gradient` to calculate and subsequently plot the concentration gradient of Figure 8:

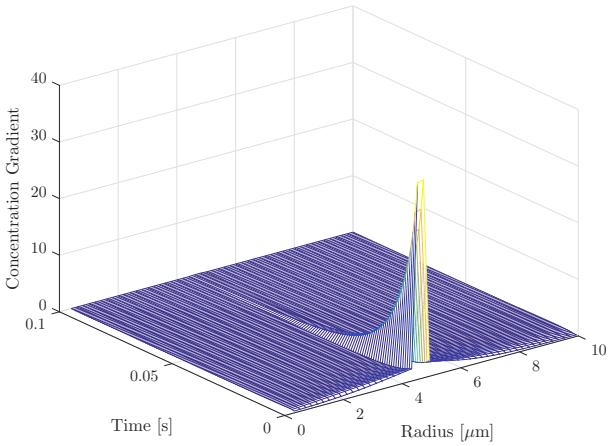

Fig. 15. Concentration gradient of a chondrocyte in a hypertonic medium. All variables set according to Table II. See Figure 8 for the equivalent concentration profile.

It becomes apparent that the greatest changes in water concentration happen near the membrane and earlier in time, a finer mesh should be used here to better resolve important detail. Far away from R_2 and as the simulation advances the concentration gradient approaches zero, a much wider mesh must be used in these areas to reduce computational cost. We offer two possible ways to program a mesh with non-uniform element size.

The first solution is to create a function to weight the vectors `r_vec` and `t_vec` higher nearer R_2 and $t = 0$ respectively. One suggested way to achieve this would be to run the provided numerical solution with uniform mesh width. From this find the concentration gradient of the concentration matrix (see Figure 15). Remap this to so that the minimum value is 0 and the maximum value is 1 and use this to create a weighting function to be applied to the space vector `r_vec` and the time vector `t_vec`.

The second solution is much easier to implement: create the meshing for each domain independently such that the user can manually specify mesh width in each domain. This has the added advantage of guaranteeing one element point falls exactly on the boundary, so there is no need to use the `round` function, which was suspected to cause the random fluctuation seen in Figure 7.

B. Expand the Code to Allow for Any Number of Domains

Some cells such as chondrocytes [30] are capable of making major structural and chemical changes to their surrounding medium over time. It stands to reason that this newly established pericellular matrix (PCM) would have different diffusion properties than the unaffected medium. Already some attempts to simulate chondrocytes model the PCM as a distinct domain [23]. If an extension is made to our existing code we would be able to model any arbitrary number of domains, each with distinct diffusivities and boundary conditions.

C. Move to an Implicit Finite Difference Method

Recall that FTCS is only stable when:

$$\phi = \frac{D\Delta t}{\Delta r^2} \leq \frac{1}{2} \tag{25}$$

Depending on the chosen material's diffusivity the ideal choice of Δt and Δr might not always be practical, while an unconditionally stable implicit scheme would still be workable. Additionally the FTCS is only first-order in time and second-order in space - leading to slower convergence as compared to an exclusively second-order scheme. The second-order, unconditionally stable Crank-Nicolson [31] might be a better choice as a FDM scheme if computational speed is less of an issue.

The Crank-Nicolson approximates Equation 8a differently from the FTCS:

$$\frac{\partial u}{\partial t} \cong \frac{u_i^{n+1} - u_i^n}{\Delta t} \tag{26}$$

and:

$$D\frac{\partial^2 u}{\partial r^2} \cong \frac{u}{2}\left[\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta r)^2} + \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta r)^2}\right] \tag{27}$$

So that rearranging produces:

$$-\phi u_{i+1}^{n+1} + 2(1+\phi)u_i^{n+1} - \phi u_{i-1}^{n+1} = \phi u_{i+1}^n + 2(1-\phi)u_i^n + \phi u_{i-1}^n \tag{28}$$

Fortunately we find that this equation can be expressed as a tridiagonal matrix of the form:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i \tag{29}$$

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & 0 & 0 & 0 & a_N & b_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{N-1} \\ d_N \end{bmatrix} \tag{30}$$

Using the coefficients:

$$a_i = -\frac{D}{2\Delta r^2}, \tag{31}$$

$$b_i = \frac{1}{\Delta t} + \frac{D}{\Delta r^2}, \tag{32}$$

$$c_i = -\frac{D}{2\Delta r^2} \tag{33}$$

$$d_i = a_i u_{i-1}^{n-1} + \left(\frac{1}{\Delta t} + a_i + c_i\right) u_i^{n-1} + c_i u_{i+1}^{n-1} \tag{34}$$

Luckily the use of a tridiagonal matrix allows us to apply the incredibly efficient Thomas algorithm [32] - otherwise known as the tridiagonal matrix algorithm, greatly reducing computational cost compared to what is typically expected of systems of linear equations. Even with the Thomas algorithm an explicit method will still be faster for any given mesh width, however we can expect the second-order Crank-Nicolson to converge faster, facilitating use of a coarser mesh and possible even lower simulation time.

VIII. Conclusion

The aim of this report has been to propose a numerical solution able to solve problems of diffusion in a multi-layer sphere, we have achieved this by using the FTSC FDM to solve Fick's second law in three contiguous domains, conserving concentration and flux through each surface. To this end we have developed a MATLAB script and used it to model water influx or efflux from a chondrocyte membrane. Further development of our solution has allowed us to simulate the expansion and contraction of the chondrocyte, as well as the buckling pressure on the membrane. We suggest it may be possible to now work backwards - using our proposed model to theoretically predict the mechanical properties of a biological cell.

The numerical simulation of spherical geometries has numerous applications - ranging from multi-phase flows to heat transfer. During validation we've proved that we can simulate the analogous problem of drug dissolution. Further development of the proposed model has the potential to decrease convergence time through the use of a second-order scheme while also improving mesh refinement by switching away from a uniform element size.

References

[1] J. Danziger and M. L. Zeidel, "Osmotic homeostasis," *Clinical Journal of the American Society of Nephrology*, pp. CJN–10 741 013, 2014.

[2] J. D. Finan and F. Guilak, "The effects of osmotic stress on the structure and function of the cell nucleus," *Journal of cellular biochemistry*, vol. 109, no. 3, pp. 460–467, 2010.

[3] C. Brocker, D. C. Thompson, and V. Vasiliou, "The role of hyperosmotic stress in inflammation and disease," *Biomolecular concepts*, vol. 3, no. 4, pp. 345–364, 2012.

[4] K. Lang, P. Lang, C. Bauer, C. Duranton, T. Wieder, S. Huber, and F. Lang, "Mechanisms of suicidal erythrocyte death," *Cellular Physiology and Biochemistry*, vol. 15, no. 5, pp. 195–202, 2005.

[5] S. Razin, "Osmotic lysis of mycoplasma," *Microbiology*, vol. 33, no. 3, pp. 471–475, 1963.

[6] H. C. Neu and L. A. Heppel, "The release of enzymes from kwherichia coli by osmotic shock and during the formation of spheroplasts," *J. biol. Chem*, vol. 240, no. 9, 1965.

[7] S. P. Timoshenko, *Theory of Elastic Stability, by S. Timoshenko...* McGraw-Hill Book Company, Incorporated, 1936.

[8] E. J. Carr and G. Pontrelli, "Modelling mass diffusion for a multi-layer sphere immersed in a semi-infinite medium: application to drug delivery," *arXiv preprint arXiv:1801.05136*, 2018.

[9] H. S. Carslaw and J. C. Jaeger, "Conduction of heat in solids," *Oxford: Clarendon Press, 1959, 2nd ed.*, 1959.

[10] R. Barrer, "Xciii. diffusion in spherical shells, and a new method of measuring the thermal diffusivity constant," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 35, no. 251, pp. 802–811, 1944.

[11] W. Stein, *Transport and diffusion across cell membranes*. Elsevier, 2012.

[12] A. Hadjitheodorou and G. Kalosakas, "Analytical and numerical study of diffusion-controlled drug release from composite spherical matrices," *Materials Science and Engineering: C*, vol. 42, pp. 681–690, 2014.

[13] B. Kaoui, M. Lauricella, and G. Pontrelli, "Mechanistic modelling of drug release from multi-layer capsules," *Computers in biology and medicine*, vol. 93, pp. 149–157, 2018.

[14] S. C. Chapra and R. P. Canale, *Numerical methods for engineers*. McGraw-Hill New York, 1998, vol. 2.

[15] E. L. Cussler, *Diffusion: mass transfer in fluid systems*. Cambridge university press, 2009.

[16] P. S. Epstein and M. S. Plesset, "On the stability of gas bubbles in liquid-gas solutions," *The Journal of Chemical Physics*, vol. 18, no. 11, pp. 1505–1509, 1950.

[17] J. Crank, *The mathematics of diffusion*. Oxford university press, 1979.

[18] R. Hickson, S. I. Barry, G. N. Mercer, and H. Sidhu, "Finite difference schemes for multilayer diffusion," *Mathematical and Computer Modelling*, vol. 54, no. 1-2, pp. 210–220, 2011.

[19] L. N. Trefethen, J. A. C. Weideman, and T. Schmelzer, "Talbot quadratures and rational approximations," *BIT Numerical Mathematics*, vol. 46, no. 3, pp. 653–670, 2006.

[20] A. K. Noor, "Bibliography of monographs and surveys on shells," *Applied Mechanics Reviews;(United States)*, vol. 43, no. 9, 1990.

[21] T. V. Karman, "The buckling of spherical shells by external pressure," *Journal of the Aeronautical Sciences*, vol. 7, no. 2, pp. 43–50, 1939.

[22] B. D. Ratner, A. S. Hoffman, F. J. Schoen, and J. E. Lemons, *Biomaterials science: an introduction to materials in medicine*. Elsevier, 2004.

[23] E. Kim, F. Guilak, and M. A. Haider, "The dynamic mechanical environment of the chondrocyte: a biphasic finite element model of cell-matrix interactions under cyclic compressive loading," *Journal of biomechanical engineering*, vol. 130, no. 6, p. 061009, 2008.

[24] R. Hine *et al.*, *The facts on file dictionary of biology*. Infobase Publishing, 2009.

[25] D. Burstein, M. L. Gray, A. L. Hartman, R. Gipe, and B. D. Foy, "Diffusion of small solutes in cartilage as measured by nuclear magnetic resonance (nmr) spectroscopy and imaging," *Journal of orthopaedic research*, vol. 11, no. 4, pp. 465–478, 1993.

[26] D. Dick, "The permeability coefficient of water in the cell membrane and the diffusion coefficient in the cell interior," *Journal of theoretical biology*, vol. 7, no. 3, pp. 504–531, 1964.

[27] L. Longsworth, "The mutual diffusion of light and heavy water," *The Journal of Physical Chemistry*, vol. 64, no. 12, pp. 1914–1917, 1960.

[28] B. V. Nguyen, Q. G. Wang, N. J. Kuiper, A. J. El Haj, C. R. Thomas, and Z. Zhang, "Biomechanical properties of single chondrocytes and chondrons determined by micromanipulation and finite-element modelling," *Journal of the Royal Society Interface*, vol. 7, no. 53, pp. 1723–1733, 2010.

[29] W. R. Trickey, F. P. Baaijens, T. A. Laursen, L. G. Alexopoulos, and F. Guilak, "Determination of the poisson's ratio of the cell: recovery properties of chondrocytes after release from complete micropipette aspiration," *Journal of biomechanics*, vol. 39, no. 1, pp. 78–87, 2006.

[30] J. Buckwalter and H. Mankin, "Articular cartilage: tissue design and chondrocyte-matrix interactions." *Instructional course lectures*, vol. 47, pp. 477–486, 1998.

[31] J. Crank and P. Nicolson, "A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type," *Advances in Computational Mathematics*, vol. 6, no. 1, pp. 207–226, 1996.

[32] T. Young and M. J. Mohlenkamp, "Introduction to numerical methods and matlab programming for engineers," *Department of Mathematics, Ohio University*, 2014.

Acknowledgements

Thank you to my supervisor - Dr Lorenzo Botto for his advice and guidance during my third year project.