Université Paul Sabatier

Internship report
L3 Parcours Spéciaux - Physique

Computer simulation (calcium ion) in the excitation-contraction coupling of skeletal muscle

By:
Damien Roncin
Internship completed in Universidad de Antioquia
From 1 March To 8 June 2017
Medellin, May 22, 2017
Supervised by Pr.Marco Giraldo

Acknowledgments

I would like to thank Prof.Marco Giraldo for welcoming me in his laboratory and being here every time I needed help. Thanks to all students who helped me during the internship: Daniel Mejía, the whole reading group, and physics students of the physics working room.

Presentation

Introducing the group of biophysics of University of Antioquia:
The University of Antioquia is one of the thirty-two public universities in Colombia. It is the oldest institute of Superior studies of Colombia, founded in 1803. In Colombia the university system is not the same as in France, every university offers courses for every career (science, art, communication, etc...).
I work in a research line created for a project about electrophysiology, it is composed by professor Marco Giraldo, Daniel Mejía Raigosa who's doing a thesis, and students from different backgrounds (medicine, engineering, biology, physics). This group has a weekly meeting to present articles, or part of the project.
My project is to simulate the diffusion of calcium ions into a mammal skeletal muscle cell. The project is performed with the Medical institute of University of Antioquia. The experimental part should have been done in this institute but unfortunately that was not possible. Consequently the whole project was done with data from articles and each one has its own experimental environment. Access to consistent baseline of data is difficult to find without making experiments on our own.

In this report we will first discuss of eletrophysiology's basics law and the structure of a skeletal muscle cell, which have been the start of the internship. After what we will study and simulate the behaviour of a membrane. The project has a big programming part, the report will also focus on technical issue of the simulation. To finish with the diffusion simulation of some examples and the calcium one.

Table of Contents

Part 1

Understanding the skeletal muscle cell, and basic electrophysiology laws

1.1 The excitation–contraction coupling (ECC) mechanism in skeletal muscle

Skeletal muscle fibre is composed by an external membrane with a part called Transverse-Tubule (T-Tube system), which is the part where interaction between intracellular and extracellular plasmas that interest us happens. This part of the membrane interacts by ionic calcium channels with the sarcoplasmic reticulum (SR). These ionic channels are composed by receptors DHPR protein on the external membrane, and RyR on the sarcoplasmic reticulum.The sarcoplasmic reticulum is a part of the muscular cell separated with the rest of the cell by a membrane. When a depolarisation happens along the external membrane, there is exchange of calcium ions between the T-tube system and sarcoplasmic reticulum. From the sarcoplasmic reticulum there is a release of Ca^{2+} in the myofibril (the contractile part of the muscle fibre with is myofilaments) where Ca^{2+} interacts with other chemical species (see the diagram). The myofibril is composed by the sarcomere part of the myofilament), which is the part which is contracted in contact with Ca^{2+}. After the contraction, Ca^{2+} ions leave from myofibril by SRCa^{2+} adenosine triphosphatase (SERCA) to the SR because of a mechanism where Na^{+} ions are involved. [2]

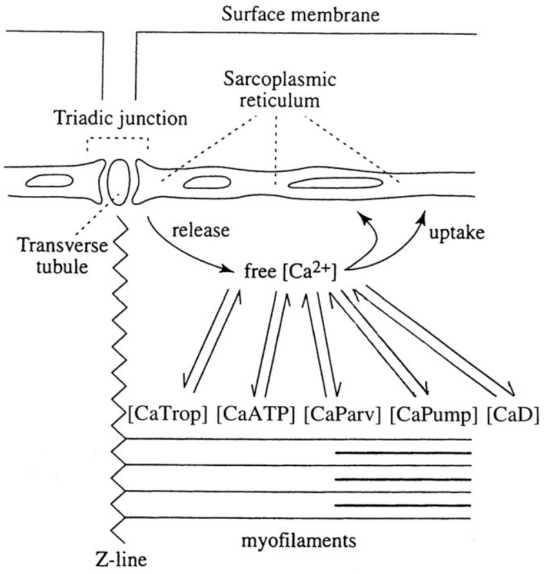


Figure 1.1: skeletal muscle cell diagram

In the diagram the free Ca^{2+} interacts with troponin, ATP, parvalbumin, and the SR Ca^{2+} pump. [1]

1.2 Basics laws of electrophysiology

In electrophysiology we will focus on two factors of our system which are the local ionic concentration and the electric field. There are some laws to understand the operating mechanisms of a muscle cell.

Ionic movement is due to an electric field and difference of concentration of each variety of ions. Which is described by the Nernst-Planck equation (NPE), which describes the flux of a certain ion like a sum of diffusion flux and drift flux (because of electric field).

$$J = -\mu(Z[C]\frac{\partial V}{\partial x} + \frac{KT}{q}\frac{\partial [C]}{\partial x})$$

Where J is the ionic flux ($\frac{mol}{sec*cm^2}$), μ is the mobility ($\frac{cm^2}{V*sec}$), Z is the valence,[C] the concentration, K is the Boltzmann constant ($\frac{joule}{kelvin}$), T is the temperature (Kelvin), q is the charge (c).
This equation can be seen as a current density:

$$I = J * ZF = -(uZ^2F[C]\frac{\partial V}{\partial x} + uZRT\frac{\partial [C]}{\partial x})$$

Where I is the current density($\frac{A}{cm^2}$), $u = \frac{\mu}{N_a}$ and F the Faraday constant ($\frac{c}{mol}$).
With this equation we obtain the Nernst equation which gives us the condition that the total current created by one type of ion across the membrane has to be zero.

$$I = -(uZ^2F[C]\frac{\partial V}{\partial x} + uZRT\frac{\partial [C]}{\partial x}) = 0 \quad => \int_{in}^{out}\frac{\partial V}{\partial x}dx = -\frac{RT}{ZF}\int_{in}^{out}\frac{\partial [C]}{[C]\partial x}dx$$

$$V_m = V_{in} - V_{out} = E_i = -\frac{RT}{ZF}ln(\frac{[C]_{out}}{[C]_{in}})$$

'Out' refers to the external part of the membrane, and 'in' the internal one; V_{in} is the potential in the cell. E_i is the Nerst potential for the i ion. This equation is an explicit expression of the equilibrium potential for each ion across the membrane (in concentration).
There is a model of the total equilibrium potential of the membrane where all of the most influential ions are involved, the Goldman-Hodgkin-Katz model (GHK):

$$V_{rest} = \frac{RT}{F}ln(\frac{P_k[K^+]_{out} + P_{Na}[Na^+]_{out} + P_{cl}[cl^-]_{in}}{P_k[K^+]_{in} + P_{Na}[Na^+]_{in} + P_{cl}[cl^-]_{out}})$$

Here P_k, P_{Na}, P_{cl} are the membrane permeability for the Potassium, Sodium and Chlorine ions pulled from the experimental data. As it can be seen the calcium ion concentration is not involved because it's negligible in comparison with the others concentration for the total membrane potential. [5]

1.2.1 Python computing of these laws

All codes are in the appendix 'Python notebook'.
First we create a data base with every experimental data needed in our model, like concentration of every ions, valence, permeability of the membrane. The data base is a comma-separated values (csv) file. This file is just data separated by a symbol, in my case it's just a space. To use it, a list of dictionaries is created. A dictionary in the python language is a collection of unordered values accessed by key rather than by index. Where every ion has his own dictionary.For example in the Fig1.2 "Valence" is a key and 2 is the value associated for the calcium dictionary. This structured data is used to calculate the rest potential (GHK model), create a vector with all Nerst Potential's ions. That can be used in the next part to simulate the membrane.

```
name valence Cin Cout permeabilidad
Ca++ 2 0.0001 2.5 0.7
K+ 1 140 5 1
Na+ 1 10 145 0.003
Cl- -1 4 110 0.1
```

```
[OrderedDict([('name', 'Ca++'), ('valence', '2'), ('Cin', '0.0001'), ('Cout', '2.5'), ('permeabilidad', '0.7')]),
OrderedDict([('name', 'K+'), ('valence', '1'), ('Cin', '140'), ('Cout', '5'), ('permeabilidad', '1')]), OrderedDic
t([('name', 'Na+'), ('valence', '1'), ('Cin', '10'), ('Cout', '145'), ('permeabilidad', '0.003')]), OrderedDict([(
'name', 'Cl-'), ('valence', '-1'), ('Cin', '4'), ('Cout', '110'), ('permeabilidad', '0.1')])]
```

Figure 1.2: From the csv file (up) we obtain the dictionaries (down)

Part 2

Membrane

2.1 Passive membrane

2.1.1 Theoretical model of a passive membrane

A passive membrane can be seen as an RC circuit, with the resistance (R_m) which is a total resistance of ionic channels, and C_m the capacity of the membrane. External and internal plasmas are conductors but the membrane is not, which is the definition of a capacitor. Our first model for a passive membrane simulation is a simple RC circuit. Where the membrane capacitance and the resistance data are from experiment (membrane capacitance = 1μF). But the rest potential(E_{rest}) must be considered too.

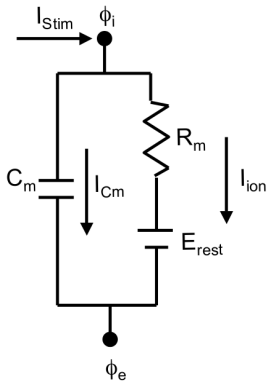


Figure 2.1: Analogy of a passive membrane

With Ohm's and Kirchhoff's laws, the circuit equation is composed by one ion current and a membrane capacity current. There is a time dependence:

$$I_{stim} = I_{Cm} + I_{ion} = C_m \frac{dV_m}{dt} + \frac{(V_m - E_{rest})}{R_m}$$

This equation is valid for the membrane current of each ion, but instead of the total rest potential of GHK model it has the Nerst one. The analytical result of this equation is:

$$V_{passive} = \begin{cases} (E_{rest} - V_\infty)e^{(-t/RC)} + V_\infty & 0 < t < T \\ (V_0 - E_{rest})e^{(-t-T)/RC} + E_{rest} & T < t \end{cases}$$

Where $V_\infty = I_{stim}R_m$, T is the time stimulation and $V_0 = V_{passive}(T)$. [11]

2.1.2 Simulation of a passive membrane

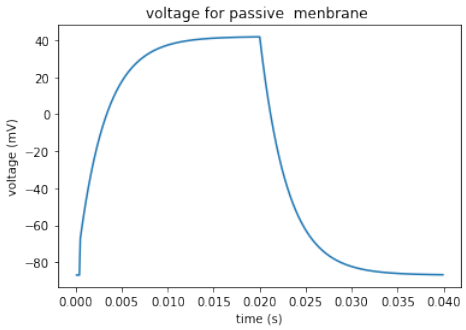


Figure 2.2: passive membrane potential

The passive membrane model has an analytical solution, this solution just has to be computed with a value of rest potential (GHK model) and a stimulation current (here $Istim = 1.4 * 10^{-2}$ A). The time stimulation choice (t=0.02s) for this simulation is enough to completely charge the membrane. Once the stimulation is over the membrane discharges to the rest potential value (computed in the part 1.2).

2.2 Active membrane

If the membrane potential (V_m) is higher than a threshold, the membrane behaviour is no longer linear, which is the main difference with the passive membrane model. Now we consider that the stimulation causes a bigger V_m than the threshold.

2.2.1 Hodgkin-Huxley model of an active membrane

The resistance of all ion channels of the passive membrane model is now splitted in a potassium, a sodium and a leak ion channels resistance. This is the Hodgkin Huxley model for an active membrane. Every ion channel has his own Nernst potential instead of a global rest potential, as we can see in Fig 2.3. The most influential ion channels in a membrane are the potassium and sodium ones, the leak resistance is the influence of the other ion channels on V_m.

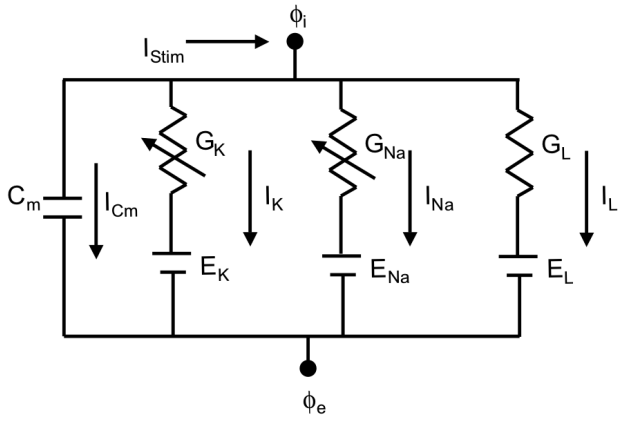


Figure 2.3: Active membrane equivalent circuit in the HH model

With I_{stim}, I_{Cm}, I_{Ions}, I_{Na}, I_K and I_{leak} as the respective currents for the stimulation, membrane capacity, the entire ion channels, sodium channel, potassium channel and the leak. These currents are ruled by:

$$I_{stim} = I_{Cm} + I_{Ions} = I_{Cm} + I_{Na} + I_K + I_{leak}$$

The biggest difference between an active and a passive membrane is that the ions resistance are no longer linear. The potassium and sodium resistance will depend on V_m.
The leakage current is linear : $I_L = g_l(V_m - E_l)$
Where g_l is the leak conductance, and E_l is the equivalent of the Nernst potential for the leakage current.
The other ionic current has a conductance which depends of V_m, The non-linear current has the following form: $I_{nl} = g_{nl}(V_m)(V_m - E_{nl})$
The non-linear conductance is an average of all ionic channels by area. As Ionic channel is a protein constituted by gates, which determines if ions can cross the membrane or not. Every gate of this protein has has its own purpose that determines if the ionic channel is open or not. The Hodgkin Huxley experiments had shown that every gate of an ionic channel can be described by functions that depend of V_m. This function is the probability for the gate to be open. Every non-linear channel has four gates, the ionic channel is considered open if all gates are open. When all gates are open then the

conductance of the ionic channel is the highest possible, conductance maximum is g_K for the potassium and g_{Na} for the sodium.

Potassium channel For this channel all gate functions are the same as for the n gates.

$$I_K = g_K(n^4)[V_m - E_K]$$

n gate function is related to V_m by α_n and β_n by the following differential equation. These functions are from HH experience on a squid axon, which is a neuron but we consider that the neurone and muscle membrane cell have the same properties. A gate function is a probability so its value is between 0 and 1.

$$\frac{dn}{dt} = \alpha_n(1-n) - \beta_n, \quad \beta_n = 0.125e^{(-(V_m - Vrest)/80}, \quad \alpha_n = 0.01\frac{10-(V_m-Vrest)}{e^{(1-0.1(V_m-Vrest)))}-1}$$

Sodium channel This channel has three gates functions m and one h, who has the same structure as n.

$$I_{Na} = g_K(m^3h)[V_m - E_K]$$

$$\frac{dm}{dt} = \alpha_m(1-m) - \beta_m, \qquad \beta_m = 4e^{(-V_m/18)}, \qquad \alpha_m = \frac{2.5-0.1(V_m-Vrest)}{e^{2.5-0.1(V_m-Vrest)}-1}$$

$$\frac{dh}{dt} = \alpha_h(1-h) - \beta_h, \qquad \beta_h = \frac{1}{e^{(30-(V_m-Vrest)))/10}+1}, \quad \alpha_h = 0.07e^{(V_m-Vrest)/20}$$

Finally we obtain the global current

$$I_{stim} = C_m(\frac{V_m}{dt}) + g_{Na}(m^3)h[V_m - E_{Na}] + g_K(n^4)[V_m - E_K] + g_{leak}[V_m - E_{leak}]$$

As we can see it's not a differential equation that we can't solve analytically, so we used a numerical solution.
[6] [11]

2.2.2 Simulation of an active membrane

For the simulation of the HH model we need to use a program that can numerically solve differential equations. The one which is used here is based on the Euler method to solve differential equations. It consist on taking a time step (dt) and with the initial conditions we calculate the other points of our function (y). It has the following form:

$$\begin{array}{ll} \frac{dy}{dt} = f(t,y)\,, & y_0 = y(t_0) \\ y_1 = y_0 + f(t_0,y_0)dt\,, & y_n = y_{n-1} + f(t_{n-1},y_{n-1})dt \end{array}$$

Where n is an integer that can be every "point time", which means that there is $n \leq N$, $N = \frac{T}{dt}$.
T is the global period where the function is calculated. To obtain a good approximation of our function we need to have a small time step.

In python we use the 'odeint' function which calls the function we have to resolve, initial conditions, 'time list' (number of point for a defined time), and object called by the function we have to resolve. In our case the function is:

$$\frac{V_m}{dt} = \frac{1}{C_m}(-I_{stim} + g_{Na}(m^3)h[V_m - E_{Na}] + g_K(n^4)[V_m - E_K] + g_{leak}[V_m - E_{leak}])$$

But this function calls others which are the gates functions (n, m and h). We have many ways to come up with the code for the HH model, we can create a function for each gate or a function with just the structure of a gate function for example. Here a function with the structure of the gates function has been created and it call the α, β, for the two gates. (see one in appendix python notebook section Active membrane). We obtain the result in Fig 2.4. The stimulus has a value of $I_{stim} = 0.03A$ and a duration time of $t_{stim} = 1ms$.
In the beginning of the simulation the membrane is at the rest state. The ionic currents are near to 0, there is just small ionic flux across the membrane.

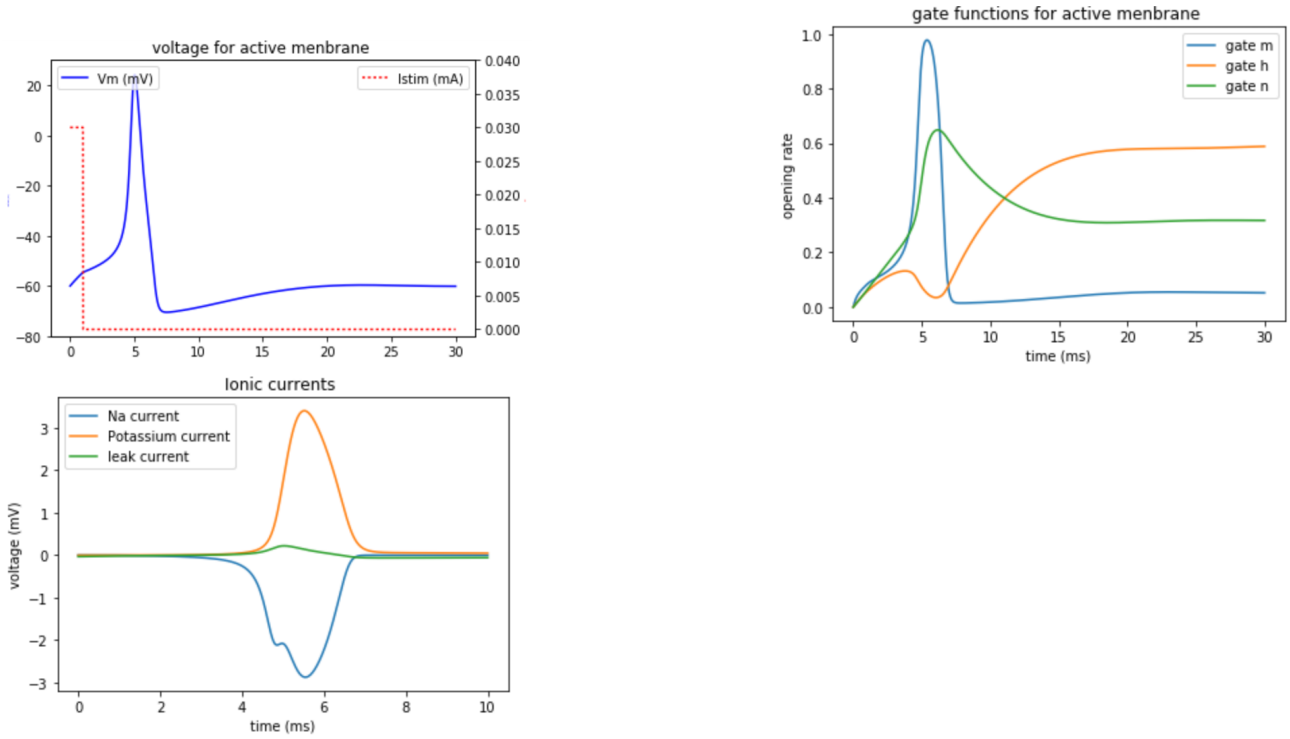


Figure 2.4: Active membrane potential and gate function associated

The next phase is the activation, there is a rapid change of Vm, this is the depolarization (before t=5ms). There is a negative sodium current which is a release of sodium into the cell, because the m gate is open, a lot more than the n gate (the time scale for the gate function graphic is different from the others). This effect is counterbalanced by the repolarization.
There is a positive increase of the potassium current, an amount of potassium flows out of the cell The membrane potential decreases rapidly because of the positive charge (K^+) flowing out.
Because the n gate doesn't go back directly to rest, the membrane potential value decreases more to the rest potential. This is called hyperpolarization. After that, there is a small depolarization until it goes back to rest.
All of these four states are called Action Potential. Note that in this model the initial gate function values are all zero, in fact it is not true, their ionic currents are at rest too. It is not so important because the Nerst potential of each ion is near to the resting membrane potential so the ionic current is not high.

Part 3

Population model and diffusion

3.1 Population Model

The diffusion of calcium in the cell depends on a diffusion term and a chemical reaction's one chemical reaction term that can be seen at first sight as a growth population model.

3.1.1 Continuous growth model

Continuous growth models with a single species involved must have the structure of the conservation equation for a population:

$$\frac{dN}{dt} = births - deaths + migration$$

The simplest model doesn't have a migration term, and births and deaths terms depend of N which is the population number. This is the Malthus model:

$$\frac{dN}{dt} = bN - dN => N(t) = N_0 e^{-(b-d)t}$$

Where b and d are positive constants (birth and death rate, respectively), and $N(0) = N_0$ initial condition. This is clearly not a realist model, if $b > d$ you have exponentially growth of the population and if $d > b$ the population dies out.
Another model is the Verhulst model which has a limit process:

$$\frac{dN}{dt} = rN(1 - \frac{N}{k})$$

with $rN(1 - \frac{N}{k})$ is the rate of birth and k the carrying capacity of the environment (k>0). The analytical solution is:

$$N(t) = \frac{N_0 K e^{rt}}{[K + N_0(e^{rt} - 1)}$$

For a long time it will tend to k. $\frac{1}{r}$ is a representative time-scale of response of the model.

3.1.2 Reaction diffusion equation

For a surface (S) enclosing a volume (V), the conservation equation says that the rate of change of the amount of material in V is equal to the rate of flow of material across S into V plus the material created in V:

$$\frac{\partial}{\partial t} \int_V C(\vec{x}, t) dV = - \int \vec{J}.d\vec{S} + \int_V f dV$$

Where $\vec{J}$ is the flux transport, and f the source of material($C(\vec{x}, t)$). If $C(\vec{x}, t)$ is continuous we obtain :

$$\int_V [\frac{\partial C}{\partial t} + \nabla.\vec{J} - f(C(\vec{x}, t)] dV = 0$$

Here the volume doesn't have importance, just the integrand must be zero. We consider that the flux transport is $\vec{J} = -D\nabla C$ where D is the diffusion coefficient and C is the population concentration. So the form of our Diffusion equation will be:

$$\frac{\partial C}{\partial t} = f + D(\nabla^2 C)$$

This equation is valid just for a coefficient of diffusion (D) constant. In our case we will put a chemical reaction term instead of function f. However it is the structure of the diffusion of the calcium into our cell. The Nersnt-Planck equation for diffusion will not be used because into the cell the variance of potential is neglected compared to the variance of concentration. What we can't do across a membrane. If we consider that our material source function is the Verhulst model of birth and death, we have the Fisher-Kolmogoroff equation:

$$\frac{\partial C(x,y,t)}{\partial t} = rC(x,y,t)(1 - \frac{C(x,y,t)}{k}) + D(\nabla^2 C(x,y,t))$$

where k,r and D are constants. [10]

3.2 Finite-difference method (FD)

The FD is a numerical method used to solve differential equations systems. The method consists on splitting the continuous domain of the differential equations to a discrete finite domain where the resolution is an approximation from the Taylor's series expansion. First we need to discretize our space, then to define a finite difference approximation for operator used in diffusion and be careful about the boundary conditions, and finally apply it to our cases.

3.2.1 Discretization

To go from a continuous domain to the discrete one, we see our space as a matrix where the space is splitted into the number of points necessary to have a good approximation. From $r(x,y,t)$ we obtain $r(x_i, y_j, t_l)$ Where i, j and l are integer such that $0 \leq i \leq mx,\ 0 \leq j \leq my,\ 0 \leq l \leq mt$ with mx, my and mt the number of point for the discretization that will give us mt matrix of dimension mx times my. Here the simulation will just be in two dimensions plus the time. The distance between two points in our discretized space is $\delta x = \frac{Lx}{mx}$ with Lx is the length of the abscissa of the studied space. Same thing for ordinate $\delta y = \frac{Ly}{my}$ and the time step $\delta t = \frac{Lt}{mt}$. A coordinate $r(x_i, y_j, t_l)$ in the grid will be the coordinate $r(i\delta x,\ j\delta y,\ l\delta t)$ in the continuous space.

3.2.2 Partial difference equation

With the discontinuous space we had created, we can have approximate expressions for the classical derivative. First we can introduce some notation:

$$\begin{array}{ll} C^l_{i,j} = C(x,y,t)\,; & C^{l+1}_{i,j} = C(x,y,t+\delta t) \\ C^l_{i+1,j} = C(x+\delta x,y,t)\,; & C^l_{i,j+1} = C(x,y+\delta y,t) \\ C^l_{i-1,j} = C(x-\delta x,y,t)\,; & C^l_{i,j-1} = C(x,y-\delta y,t) \end{array}$$

With the Taylor's series we can obtain the following expression for derivatives.[9] [4]
-Time derivative :

$$C^{l+1}_{i,j} = C^l_{i,j} + \delta t\frac{\partial C^l_{i,j}}{\partial t} + O(\delta t^2)\ so\, we\, have\ => \frac{\partial C^l_{i,j}}{\partial t} = \frac{C^{l+1}_{i,j} - C^l_{i,j}}{\delta t} - \frac{O(\delta t^2)}{\delta t}$$

The development is only at the first order, and the leading term can be neglected.
-Space derivatives :

$$C^l_{i+1,j} = C^l_{i,j} + \delta x\frac{\partial C^l_{i,j}}{\partial x} + \frac{1}{2}(\delta x)^2\frac{\partial^2 C^l_{i,j}}{\partial x^2} + O(\partial x^3)$$

$$C^l_{i-1,j} = C^l_{i,j} - \delta x\frac{\partial C^l_{i,j}}{\partial x} + \frac{1}{2}(\delta x)^2\frac{\partial^2 C^l_{i,j}}{\partial x^2} + O(\partial x^3)$$

If we sum these two equation we obtain:

$$\frac{\partial C^l_{i,j}}{\partial x} = \frac{C^l_{i+1,j} - C^l_{i-1,j}}{2\delta x} - (O\delta x^3)$$

$$\frac{\partial^2 C^l_{i,j}}{\partial x^2} = \frac{C^l_{i+1,j} - C^l_{i-1,j} - 2C^l_{i,j}}{(\delta x)^2} - \frac{O(\partial x^3}{(\partial x)^2}$$

Note that every equation here works the same for the y derivative. The leading term can also be neglected.

Boundaries conditions The important technical point is that the finite difference method uses the two neighbouring points ($C_{i+1,j}$ and $C_{i-1,j}$) to calculate the middle point ($C_{i,j}$). A problem appear at the borders points. So we have to add in our grid an imaginary border to compute the real border points of the grid. In the codes of this project the grid representing the studied system is two steps smaller than the real grid. In fact the index points have the following form: $1 \leq i \leq mx - 1$, $1 \leq j \leq my - 1$. The Neumann boundary condition is really useful too, it means that the flux in the borders is zero.

$$\frac{\partial C_{x=0}}{\partial x} = \frac{\partial C_{x=Lx}}{\partial x} = 0 \;\Rightarrow\; \frac{C_{i+1} - C_i}{2\delta x} = 0 \;\Rightarrow\; C_{i+1} = C_{i-1}$$

The token solution for the grid is useful here too, we use the imaginary boundary to compute the Neumann condition.

3.2.3 Application of FD method to diffusion equation

See one in appendix "jupyter notebook" section Diffusion.
To understand how this method works we will begin with the following diffusion equation, which is the simplest:

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

One time discretized, and the error term neglected we obtain:

$$\frac{C^{l+1}_i - C^l_i}{\delta t} = D[\frac{C^l_{i+1} - C^l_{i-1} - 2C^l_i}{(\delta x)^2}]$$

To compute numerically we prefer this form:

$$C^{l+1}_i = C^l_i + D(\frac{\delta t}{(\delta x)^2})[C^l_{i+1} - C^l_{i-1} - 2C^l_i]$$

The method is the same when generalizing into two or three space dimensions. We just have about be careful to a stability condition which is: $(\frac{\delta t}{(\delta x)^2}) \leq \frac{1}{2}$. That means that the time step (δt) must be very small. [4] The size of our domain (Lx, Ly) and the time (Lt) will only depend on the physical term of the equation which is the diffusion coefficient, if $D \sim \frac{cm^2}{s}$ then Lx, Ly are centimetres and Lt is second.
Because the space is two-dimensional, the concentration of population or what we want to represent by C, is represented by another dimension space or colour, the choice here is the colour one. The colour scale must be well understated.
The simulation's physicals parameters are: Lx= 1.0 (the x domain size), Ly= 1.0 (the y domain size), Time= 100 (the integration time) and D=0.01 which is the coefficient of diffusion.
The simulation's numerical parameters are: NT = 10000 number of time steps, NX = 20 the number of grid points in x, NY= 20 the number of grid points in y, dt = Time/NT the time step. The initial condition is C[5,5]= 100, which means that the initial matrix representing our grid has the coordinate point $C^0_{5,5} = 100$, that the initial population.

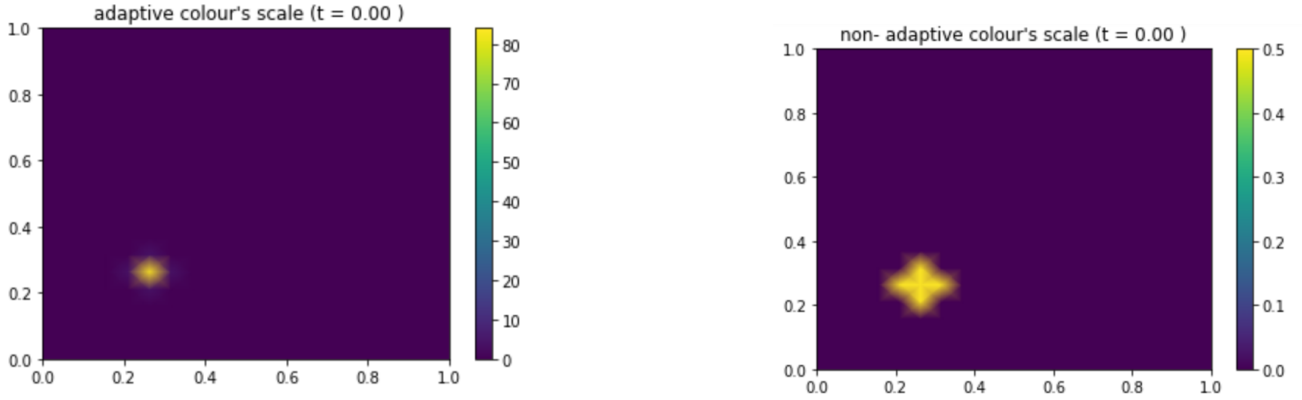


Figure 3.1: representation of diffusion simulation at a δt after the initial time

Figure 3.2: representation of diffusion simulation after a long time

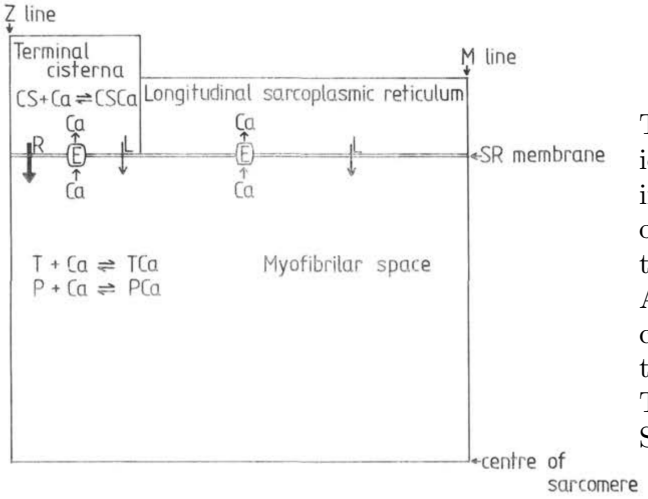
In the adaptive colour's scale image of Fig 3.2 we can think that the diffusion is not terminated, but the simulation stars in just one population's point equal to 100 and the scale of this image is 10^{-5} magnitude order. This is a homogeneous distribution population state, as we can see with the other colour's scale.

Fisher-Kolmogoroff equation Now we apply this method to the Fisher-Kolmogoroff equation with the Neuman border condition. In fact we just have to change the diffusion equation in adding the birth and death term. When we compute this equation, we have Nt matrix, what give us an animation, in a paper report it's not possible to put it.

Part 4

Simulation of calcium diffusion into a muscle cell

With all techniques explained overhead it's possible to simulate the diffusion of calcium in a skeletal muscle cell. To do it we use the Cannell and Allen model [3], the version of Holly which is a bit different with more mathematical model [7] and some ideas of Jiang article [8]. For example in the Holly article they don't talk precisely about initials or boundary conditions, but it is done in the Jiang article.



The calcium (Ca) is present in TC, either as free ions, or bounded to calsequestrin (CS). Activating the fibre, calcium is released (R) into the myofibrilar space. There, calcium diffuses and binds to the T-sites (T), the P-sites (P) and the Ca-ATPase (E). The enzyme Ca-ATPase provides calcium reuptake to the SR. Calcium diffuses through the longitudinal sarcoplasmic reticulum. There is also a constant leak of calcium from the SR (L). [7]

Figure 4.1: Digram of the Holly's Model

This model is for a frog muscle cell, but we will take this structure for our first simulation. The structure of a muscle fibre is a cylinder, with an angular symmetry. So we just have to compute a two dimensional space a big picture in our heads of the three dimensional diffusion. The axial symmetry is around the centre of the sarcomere.

Here the simulation of the myofirilar space is an Euclidean space (easier for the FD method). The diffusion of the calcium into the myofibrilar space is the only one we discuss here.

Diffusion of calcium ions in myofibrilar space

As we can see in the part 1.1, the calcium not only diffuses into the myofibrilar space, but there are interactions with chemicals species involved too. We just considered the troponin (T-sites) and parvalbumin (P-sites) as in the figure 4.1.

The reactions are chemical, with kinetic rate value of molecules bounding (K_{on} , K_{off}) that are given by the Callen and Allen experiments [3].

$$\mathrm{Ca} + \mathrm{P} \underset{Kp_{off}}{\overset{Kp_{on}}{\rightleftharpoons}} \mathrm{PCa}\,, \quad \mathrm{Ca} + \mathrm{T} \underset{Kt_{off}}{\overset{Kt_{on}}{\rightleftharpoons}} \mathrm{TCa}$$

This chemical reactions can be translated into a differential equation because of the kinetics terms.

Which give:

$$\frac{\partial Ca}{\partial t} = Kp_{off}[PCa] + Kt_{off}[TCa] - Kp_{on}[P][Ca] - Kt_{on}[T][Ca]$$

With Kp_{on} , Kp_{off} , Kt_{on} , Kt_{off} respectively the kinetic rate of creation PCa, Ca and P, TCa, T and Ca. This will be the birth and death rate term of our diffusion expression (Fisher-Kolmogoroff equation, Part 3.1.2). So the entire diffusion term into the myofibriliar space is:

$$\frac{\partial Ca}{\partial t} = Kp_{off}[PCa] + Kt_{off}[TCa] - Kp_{on}[P][Ca] - Kt_{on}[T][Ca] + D(\bigtriangledown^2[Ca])$$

It's necessary to discretize it to use the finite difference method. We also have to create a matrix for every chemical species that will interact one with another.

$$\frac{[Ca]_{i,j}^{l+1} - [Ca]_{i,j}^{l}}{\delta t} = K_{Toff}[TCa]_{i,j}^{l} - K_{Ton}[Ca]_{i,j}^{l}[TCa]_{i,j}^{l} + K_{Poff}[PCa]_{i,j}^{l} - K_{Pon}[Ca]_{i,j}^{l}[PCa]_{i,j}^{l}$$

$$+D(\frac{([Ca]_{i-1,j}^{l} - 2[Ca]_{i,j}^{l} + [Ca]_{i+1,j}^{l})}{(\delta x)^2} + \frac{([Ca]_{i,j-1}^{l} - 2[Ca]_{i,j}^{l} + [Ca]_{i,j+1}^{l})}{(\delta y)^2})$$

The others species matrix has to update too.

$$\frac{[TCa]_{i,j}^{l+1} - [TCa]_{i,j}^{l}}{\delta t} = K_{Ton}[Ca]_{i,j}^{l}[T]_{i,j}^{l} - K_{Toff}[TCa]_{i,j}^{l}, \qquad \frac{[T]_{i,j}^{l+1} - [T]_{i,j}^{l}}{\delta t} = K_{Toff}[TCa]_{i,j}^{l} - K_{Ton}[Ca]_{i,j}^{l}[T]_{i,j}^{l}$$

$$\frac{[P]_{i,j}^{l+1} - [P]_{i,j}^{l}}{\delta t} = K_{Poff}[TCa]_{i,j}^{l} - K_{Pon}[Ca]_{i,j}^{l}[P]_{i,j}^{l}, \qquad \frac{[PCa]_{i,j}^{l+1} - [PCa]_{i,j}^{l}}{\delta t} = K_{Pon}[Ca]_{i,j}^{l}[P]_{i,j}^{l} - K_{Poff}[TCa]_{i,j}^{l}$$

Borders and initial conditions We consider that the Calcium can't cross the membrane, for that the Neumman border condition is used and that the neighbours membrane points have the resting Calcium concentration. [8] Holly's model [7] give the following data:

$$[T]_{total} = [T] + [TCa] = 240\mu mol.l^{-1}, \quad [P]_{total} = [P] + [PCa] = 1000\mu mol.l^{-1}, \quad [Ca]_{MSrest} = 0.06\mu mol.l^{-1}$$

$$K_{Ton} = 0,12\mu mol.ms^{-1},\ K_{Toff} = 0,12ms^{-1},\ K_{Pon} = 0,25\mu mol.ms^{-1},\ K_{Poff} = 0,001ms^{-1}$$

We consider that the initial cell's concentrations are in an equilibrium state, what means: $\frac{d[T]}{dt} = \frac{d[TCa]}{dt} = \frac{d[P]}{dt} = \frac{d[PCa]}{dt} = 0$. With this hypothesis we can calculate all initials concentrations of our system:

$$\frac{d[TCa]_0}{dt} = K_{Ton}[Ca]_0[TCa]_0 - K_{Toff}[TCa]_0 = 0 \ => \qquad [T]_0 = \frac{K_{Toff}[TCa]_0}{K_{Ton}[Ca]_0}$$

$$[T]_{total} = [T]_0 + [TCa]_0 = [TCa]_0(1 + \frac{K_{Toff}}{K_{Ton}[Ca]_0}) \ => \quad [TCa]_0 = \frac{K_{Ton}[Ca]_0[T]_{total}}{K_{Ton}[Ca]_0 + K_{Toff}}$$

We compute all initial concentrations and obtain:

$$[T]_0 = 222.2\mu mol,\ [TCa]_0 = 17.8\mu mol,\ [P]_0 = 995.0\mu mol,\ [PCa] = 5.0\mu mol$$

In this simulation we consider that the T-sites and the p-sites are homogeneously distributed, so the initials matrix has the same value in every point.

We also know that the myofibriliar's space takes up to 90 % of the sarcomere which has a radius $Ly = 0,5\mu m$ and a length $Lx = 1,1\mu m$. Because the TC and SR are at the outer localisation of the radius, the dimensions of MS are $Ly = 0,45\mu m$ and $Lx = 1,1\mu m$.
The parameters choice here are the same number of point to discretize the two spaces dimensions ($Nx = Ny = 20$) it's easier to have square matrix, with this split it's possible to determine the amount of chemicals in every point of the matrix in dividing the upper conversion by 4000 the number of points by matrix. The time simulation is $6ms$ split in $NT = 100000$ time steps. The step is very small, so the computing of the code is long, but if there is a smaller time step we will have computing problem. This simulation begins with a calcium point with a value of $Ca^0_{15,5} = 0.2\mu m$.

The last image of the Fig 4.2(Ca(t=6)) of the simulation shows us a homogeneous distribution of Calcium because of the colour scale. The concentration of the P and T sites has changed too, but not

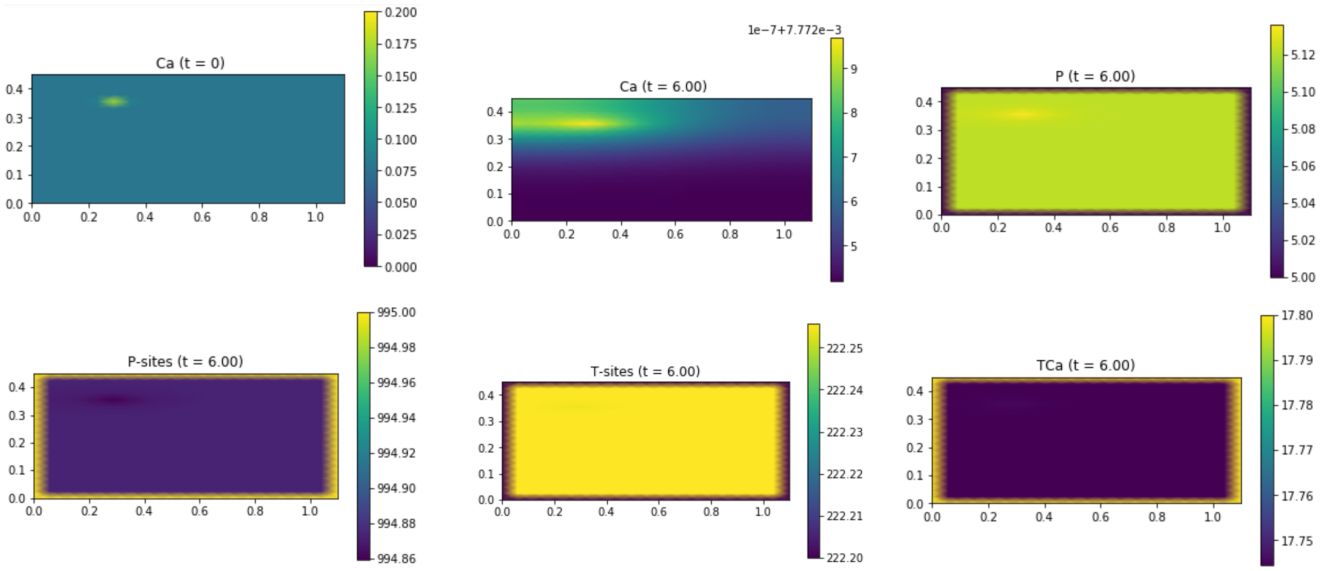


Figure 4.2: Representation of calcium diffusion into the myofibriliar space, and athors chemicals involved

in the border points because the calcium concentration in the border points doesn't change. With this simulation we can have an idea of the time needed for the calcium to be homogeneously distributed in the myofibrilar space, 6 ms is enough. The concentration of other species doesn't change significantly because the initials concentrations are high compared with the calcium initials concentration.

Conclusion

I started the internship without any knowledge in Biophysics; the first two weeks were about understanding the cell structure and the electrophysiology laws by reading articles and books. After that I began programming, first a data base and some basic programs. With series of articles of Hodgkin and Huxley and books of electrophysiology [6] [11] [5] I simulated the electrical behaviour of a membrane. To do so, I had to understand the complex model of Hodgkin and Huxley, and to resolve differential equations involved in its models. I had to use a numerical method (the one in the report is Euler's method). The second part was the diffusion of the calcium, for that I had to know some models of diffusion. The finite difference was used to simulate different diffusion models. The Holly model of the calcium diffusion into the myofibriliar space of a frog muscle cell. Programming can be really useful to understand better a system, but we have to be careful of all the data we use and how we represent it. A big part of the work is to create the right code, without it, nothing can be simulated, but the simulation is useless if the data we put into it is not correctly presented.
All the work I have done during the internship won't be used by the biophysics group exactly as it is, but the code structures can be helpful for different projects.

This Internship has taught me a lot. I was in contact with a team of researchers, and I know what a daily researcher's life looks like. It was the first time I spent a long time studying a specific subject, understanding the articles and following the scientific history of my subject. I discovered a lot of computing and programming tools, which will be really useful for the rest of my studies, and who knows, my future life as a researcher. Working in a biologic experiment was something new for me.
Also being in contact with another culture is also a very eye-opening experience. The exercise to speak to people in Spanish, read articles in English, and still thinking in French was highly challenging.

References

[1] Stephen M. Baylor and Stephen Hollingworth. Calcium indicators and calcium signalling in skeletal muscle fibres during excitation-contraction coupling. *Progress in Biophysics and Molecular Biology*, 105(3):162–179, 2011.

[2] Juan C. Calderón, Pura Bolaños, and Carlo Caputo. The excitation-contraction coupling mechanism in skeletal muscle. *Biophysical Reviews*, 6(1):133–160, 2014.

[3] Cannell and D. G. ALLEN. Model of calcium movements during. *Biophysical Journal*, 45(May), 1984.

[4] J. Crank. The mathematics of diffusion. *Oxford University Press*, page 414, 1975.

[5] Samuel Miao-Sin Wu Daniel Johnston. *Foundations of Cellular Neurophysiology*. the MIT press, 1994.

[6] a L Hodgkin and a F Huxley. Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo. *The Journal of physiology*, 116(4):449–472, 1952.

[7] M. Hollý and J. Poledna. Model of calcium diffusion, binding and membrane transport in the sarcomere of frog skeletal muscle. *General physiology and biophysics*, 8(6):539–553, 1989.

[8] Y H Jiang, M G Klein, and M F Schneider. Numerical simulation of Ca2+ "sparks" in skeletal muscle. *Biophysical journal*, 77(5):2333–2357, 1999.

[9] H Lewyt. On the Partial Difference Equations of Mathematical Physics. *Mathematische Annalen*, (March), 1928.

[10] Murray. *Mathematical Biology: I. An Introduction, Third Edition*. Springer, 2001.

[11] Joseph V. Tranquillo. *Quantitative Neurophysiology*, volume 3. 2009.

Appendix A

Python notebook

All of the programming I have done are also on Github, at: https://github.com/damienRONCIN

With a csv fichier create a dico list (eatch line)

In [ ]:
```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from pylab import *
import csv

def listdico (mon_fichier_csv):
    listdico=list()
    csvfile = open(mon_fichier_csv,"r")
    reader = csv.DictReader(csvfile , delimiter=' ')
    for item in reader:
        listdico.append(item)
    return listdico
```

Vrest=E from GHK Model

In [ ]:
```
#datas for a mammalian cell
T=310 #kelvin
R=1.98*4.2 #V*C/°K-mol
F=96.480 #C/mol
def Vrest (list_dico):
    E = ((R*T)/F)*np.log((float(list_dico[1]['permeabilidad'])*float(list_dico[1]['Cout']) \
                          +float(list_dico[2]['permeabilidad'])*float(list_dico[2]['Cout']) \
                          +float(list_dico[3]['permeabilidad'])*float(list_dico[3]['Cin'])) \
                         /(float(list_dico[1]['permeabilidad'])*float(list_dico[1]['Cin']) \
                           +float(list_dico[2]['permeabilidad'])*float(list_dico[2]['Cin']) \
                           +float(list_dico[3]['permeabilidad'])*float(list_dico[3]['Cout'])))
    return (E)
```

Passive Menbrane model:

In [ ]:
```
def Vpassive ( Istimulation, timestimulation, timeexperience ):
    E= Vrest( listdico('Ion.csv') )
    t=0
    time = []
    Vmen = []
    Vinf = ( float(Istimulation)*float(Rm))
    while t < timeexperience :
        tbegin=0.0005
        t=t + 0.0001
        if t < tbegin:
            Vmen.append(E)
            time.append(t)

        if tbegin<= t < timestimulation :
            Vm = (float(E)-float(Vinf))*((np.exp((-t)/(float(Rm)*float(Cm)))))+float(Vinf);
            time.append(t);
            Vmen.append(Vm);

        if  timestimulation< t <  timeexperience :
            Vm1 = (float(Vm)-E)*(np.exp((-t+timestimulation)/(float(Rm)*float(Cm))))+E;
            time.append(t);
            Vmen.append(Vm1);

    return (time, Vmen)

results = Vpassive (Istimulation, 0.02, 0.04)
t = results[0]
V= results[1]
```

Active menbrane

In [ ]:
```
# -*- coding: utf-8 -*-
from __future__ import division
from scipy import *
from pylab import *
h0=0
n0=0
m0=0
V0=-60
Istim=0.03#mA
Cm = 0.01
gNa = 1.20
gK = 0.36
gL = 0.003
Ilist=list()
vNa = 55.15
vK =-72.14
vL =-49.42
#Gates functions
def An(V):
    return (0.01*(V+50.0))/(1.0-np.exp(-(V+50.0)/10.0))
def Bn(V):
    return 0.125*np.exp(-(V+60.0)/80.0)
def Am(V):
    return (0.1*(V+35.0))/(1-np.exp(-(V+35.0)/10.0))
def Bm(V):
    return 4.0*np.exp(-(V+60.0)/18.0)
def Ah(V):
    return 0.07*np.exp(-(V+60.0)/20.0)
def Bh(V):
    return 1.0/(1.0+np.exp(-(V+30.0)/10.0))
def dmdt(V,m,Am,Bm):
    return Am(V)*(1-m)-Bm(V)*m
def dndt(V,n,An,Bn):
    return An(V)*(1-n)-Bn(V)*n
def dhdt(V,h,Ah,Bh):
    return Ah(V)*(1-h)-Bh(V)*h

# Simulate the time stimulation
def FIstim(t,tstim,Istim):
    if t < tstim:
        I=Istim
    else :
        I=0
    return I
#the Fonction is what we have to resolve to know VM:
def Fonction(y,t,FIstim,Am,Bm,An,Bn,Ah,Bh,tstim,Istim):
    #V , m, h , n = y
    V = y[0]
    m = y[1]
    h = y[2]
    n = y[3]
    dmdt= Am(V)*(1-y[1])-Bm(V)*y[1]
    dhdt= Ah(V)*(1-y[2])-Bh(V)*y[2]
    dndt= An(V)*(1-y[3])-Bn(V)*y[3]

    return [(1/Cm)*(FIstim(t,tstim,Istim)-gNa*(np.power(m,3))*h*(V-vNa) \
                    -gK*(np.power(n,4))*(V-vK)-gL*(V-vL)),dmdt,dhdt,dndt ]

# The python function to resolve ordinary differential equations
y0=[V0,m0,h0,n0]
tstim=1
time1=np.linspace(0,30,10000)
Vm1=odeint(Fonction,y0,time1,args=(FIstim,Am,Bm,An,Bn,Ah,Bh,tstim,Istim))
VM , m , h , n = Vm1[:,0] , Vm1[:,1] , Vm1[:,2] ,  Vm1[:,3]

#differents currants values
Ina=gNa*(m**3)*h*(VM-vNa)
Ik=gK*(np.power(n,4))*(VM-vK)
Il=gL*(VM-vL)
for i in time1 :
    if i < tstim:
        I=Istim
        Ilist.append(I)
    else :
        I=0
        Ilist.append(I)
```

Diffusion

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

# PHYSICAL PARAMETERS
Lx = 1.0   #Domain size x
Ly = 1.0   #Domain size y
Time = 100#Integration time
D=0.01
r=0.1
k=1

def laplacian(Z):
    Zxless = Z[0:-2,1:-1]
    Zyless = Z[1:-1,0:-2]
    Zxplus = Z[2:,1:-1]
    Zyplus = Z[1:-1,2:]
    Zcenter = Z[1:-1,1:-1]
    return ((Zxless+ Zxplus - 2 * Zcenter)/ dx**2 ) + (Zyless + Zyplus -2* Zcenter) / dy**2
# NUMERICAL PARAMETERS

NT = 10000   #Number of time steps
NX = 20        #Number of grid points in x
NY = 20        #Number of grid points in y
dt = Time/NT   #Grid step (time)
dx = Lx/(NX) #Grid step in x (space)
dy = Ly/(NY) #Grid step in y (space)

xx = np.linspace(0,Lx,NX)
yy = np.linspace(0,Ly,NY)

### MAIN PROGRAM ###
Ninit=100
N = np.zeros((NX,NY))
N_min, N_max = 0, 0.005*Ninit
N[5,5]= Ninit

# Main loop
for n in range(0,NT):
    deltaN = laplacian(N)
    # We take the values of N inside the grid.
    Nc = N[1:-1,1:-1]
    # We update the variables.
    #Simple Diffusion equation
    N[1:-1,1:-1] = Nc+ dt*(D*deltaN )
    #the Fisher-Kolmogoroff equation
    #N[1:-1,1:-1] =  Nc + dt * (r*Nc*(1-(Nc/k))+ D*deltaN )
    #Border condition nneumann condition is a egality and no one
    #who take the value of the other, so we do in 2 times
    N[0,:] = N[2,:]
    N[-1,:] = N[-3,:]
    N[:,0] = N[:,2]
    N[:,-1] =  N[:,-3]
    N[:,-3]= N[:,-1]
    N[:,2]= N[:,0]
    N[-3,:]=N[-1,:]
    N[2,:]= N[0,:]
```

Model from Holly 1989 of Calcium diffusion

In [8]:
```
#time and space
Lx,Ly=1.1, 0.45  #µm
Ny,Nx =20, 20
dy,dx  =Ly/Ny, Lx/Nx # space step
D=0.7 #µm*ms(-1)
k=1
NT=100000
T = 6# total time (ms)
dt= T/NT
n = int(T/dt)
xx = np.linspace(0,Lx,Nx,endpoint=True)
yy = np.linspace(0,Ly,Ny,endpoint=True)
#colour stuff
Caspeak=0.2
Ca_min, Ca_max= 0.08, Caspeak
T_min, T_max= 0, 222.2
TCa_min, TCa_max= 0, 100
PCa_min, PCa_max= 0, 100
P_min, P_max= 0, 995
# kinetics factors data from Holly paper 1989
kton,ktoff =0.12, 0.12 #µmol-1 l ms-1 and ms-1 for KToff
kpon,kpoff=0.25,0.001  #µmol-1 l ms-1 and ms-1 for KPoff

#All pieces matrices
Ca = T = TCa = P  = PCa = np.zeros((Nx, Ny))
# All spiecies start with a uniform concentration exept Ca, but all species are not homogeniusly tistribut
T[:,:],TCa[:,:] =222.2 ,17.8
P[:,:],PCa[:,:]=995, 5
Ca[:,:]= 0.08 # concentration at rest in MS µmol l-1
Ca[15,5]=Caspeak

def laplacian(Z):
    Zxless = Z[0:-2,1:-1]
    Zyless = Z[1:-1,0:-2]
    Zxplus = Z[2:,1:-1]
    Zyplus = Z[1:-1,2:]
    Zcenter = Z[1:-1,1:-1]
    return ((Zxless+ Zxplus - 2 * Zcenter)/ dx**2 ) + (Zyless + Zyplus -2* Zcenter) / dy**2
def neuman(N):
    N[0,:] = N[2,:]
    N[-1,:] = N[-3,:]
    N[:,0] = N[:,2]
    N[:,-1] =  N[:,-3]
    N[2,:]=N[0,:]
    N[-3,:]=N[-1,:]
    N[:,2]=N[:,0]
    N[:,-3]= N[:,-1]
    # Neumann conditions: derivatives at the edges
    # are null.
    return N
# We simulate the PDE with the finite difference method.
for i in range(n):
    # We compute the Laplacian of N
    deltaCa = laplacian(Ca)
    # We take the values of N inside the grid.
    Cai = Ca[1:-1,1:-1]
    Ti=T[1:-1,1:-1]
    TCai=TCa[1:-1,1:-1]
    Pi=P[1:-1,1:-1]
    PCai=PCa[1:-1,1:-1]

    # We update the variables.
    Ca[1:-1,1:-1] =  Cai + dt * (ktoff*TCa[1:-1,1:-1]-kton*Ca[1:-1,1:-1]*T[1:-1,1:-1]\
                                 +kpoff*PCa[1:-1,1:-1]-kpon*Ca[1:-1,1:-1]*P[1:-1,1:-1]+ D*deltaCa )
    T[1:-1,1:-1]=Ti+dt*(ktoff*TCa[1:-1,1:-1]-kton*Ca[1:-1,1:-1]*Ti)
    TCa[1:-1,1:-1]=TCai+dt*(kton*Ca[1:-1,1:-1]*Ti-ktoff*TCai)
    P[1:-1,1:-1]=Pi+dt*(kpoff*PCa[1:-1,1:-1]-kpon*Ca[1:-1,1:-1]*Pi)
    PCa[1:-1,1:-1]=PCai+dt*(kpon*Ca[1:-1,1:-1]*Pi-kpoff*PCai)
    #n neuman condition for calcium, the bonderies points  have the rest value:
    Ca[:,0]=Ca[:,-1]=Ca[0,:]=Ca[-1,:]=0.08
    neuman(Ca)
```

Appendix B

Résumé/Summary

Résumé Le projet de mon stage concerne la biophysique et plus précisément l'électrophysiologie. Le but étant de modéliser la diffusion des ions calcium au sein d'une cellule musculaire de mammifère (les expérimentations n'ayant pas pu se faire la modélisation n'est pas spécifique au rat). La première partie de du projet consiste à se familiariser avec la structure précise et les interactions présentes dans ce système. Une cellule musculaire se compose de reversoirs d'ion et de partie contractile chacune séparée par des membranes.
Les membranes sont composées de partie perméable et de canaux ioniques, la modélisation peut se faire par une analogie avec un circuit RC. Seulement le comportement de la résistance est complexe, chaque espèce chimique a son propre canal ionique qui est une résistance de notre modèle. Le comportement de chaque résistance dépend du potentiel membranaire non linéairement, l'équation n'est pas résolvable analytiquement il faut donc le faire numériquement.
La seconde partie du projet concerne la diffusion, ainsi dans un premier temps il convient d'introduire quelques systèmes comprenant de la diffusion, tel que quelques uns comprenant un terme de variation de population (ici de concentration). Pour en suite appliquer ceci a des cellules. Étant donné que les cellules sont des systèmes continus, il s'agit de discrétiser ces espaces pour les modéliser par la suite à l'aide de la méthode des différences finîes. Il faut porter un intérêt particulier aux valeurs utilisées pour les simulations, et la représentations de ces dernières car certaines peuvent être trompeuses.
Tout le travail que j'ai accomplis le long de ce stage est pour une grande partie de l'informatique, qui ne sera pas utilisé par laboratoire de recherche directement mais qui peut s'avérer utile pour différents projets.

Summary The project of my internship is about biophysics and more specifically, electrophysiology. For the purposes of modelling the ion calcium diffusion into mammalian muscular cell (the experiment couldn't be done so the model is not specifically about the rat). The first part of the project consists on being more familiar with the precise structure and interaction involved in our system. A muscular cell is made of an ion reservoir and contracted parts that are separated by membranes.
Membranes are composed by a permeable part and ionic channels. The modelling can be done by an analogy with a RC circuit. But the resistance behaviour is complex, every chemical species has their own ionic channel which is a resistance in our analogy. The behaviour of each resistance is not linearly dependant on the total membrane potential potential not linearly, this equation is not resolvable analytically so we solvable it numerically.
The second part of the project is about diffusion. In the beginning we introduced some systems with diffusion and variation of population term (here we talked about concentration). For the next phase we applied this to a cell system. Because cells are continuous systems we had to discretize the spaces for the model with the finite difference method. It's crucial to be careful about the values used for the simulations and their representations because this can be misleading.
The biggest part of the work I have done in this internship is for informatics, which will not be used directly by the laboratory but it can be useful for others projects.