Université d'Ottawa · University of Ottawa

A lattice model for the rupture kinetics of lipid bilayer membranes

by

Luc Fournier

A Thesis submitted to
the Faculty of Graduate and Postdoctoral studies
in partial fulfillment of the requirements for
the degree of Master in science in physics

University of Ottawa
Ottawa, Ontario
September 20, 2002

Table of contents

Abstract

We have constructed a model for the kinetics of rupture of membranes under tension, applying physical principles relevant to lipid bilayers held together by hydrophobic interactions. The membrane is characterized by the bulk compressibility (for expansion) K and the thickness $2h_t$ of the hydrophobic part of the bilayer. The model is a lattice model which incorporates stress relaxation, and considers the nucleation of pores at constant area, constant temperature, and constant particle number. The particle number is conserved by allowing multiple occupancy of the sites. A value for the rigidity of the phopholipid tails in the L_α liquid phase are found for saturated and unsaturated lipids, and long diblock copolymers. An equilibrium "phase diagram" is constructed as a function of temperature and strain with the pores total surface and distribution as the order parameters. With parameters relevant to saturated phosphatidylcholine (PC) lipid membranes, well defined regions of "no pores", "protopores (non-critical pores)", "rupture" are found. The model also reproduces recent results on super-thick membranes, and on membranes in presence of peptides. Free energy curves as a function of total pore surface are presented for various values of tension and temperature, and the fractal dimension of the pore edge is evaluated.

Sommaire

Nous avons construit un modèle pour étudier la cinétique de la rupture des bicouches lipidiques sous tensions, lesquelles sont soutenues par des liens hydrophobiques. Ces membranes sont définies principalement par leur compressibilité K et leur hauteur hydrophobique $2h_t$, qui représente l'épaisseur de la section hydrophobique de la membrane. Le modèle est discret (matricielle) et contient la relaxation sous l'effet de la nucléation de pores pour des membranes à surface constante, température constante, et à nombre de particules constant. Le nombre de particules est conservé en allouant une occupation multiple par site. Des valeurs pour la rigidité des queues lipidiques pour des bicouches dans la phase liquide L_α sont déterminées dans le cas de membranes saturées ou non saturées, et pour des membranes formées de longs co-polymères. Un diagramme de phase à l'équilibre en fonction de la température et de l'étirement fut réalisé. Avec des paramètres propres aux membranes PC saturées, les régions « sans pore », « protopores » (pores instables) et « rupture » sont trouvées. Le modèle reproduit aussi le scénario de rupture de membrane ultra-épaisse, et de membranes en présence de peptides. Des courbes d'énergie libre en fonction de la surface totale occupée par des pores sont aussi représentées pour diverses tensions. Finalement, la dimension fractale du contour des pores est calculée.

Remerciements

J'aimerais remercier tous ceux qui ont participé de près ou de loin à cette recherche, que ce soit pour leurs conseils pratiques, ou encore pour leur support.

Particulièrement, j'aimerais remercier ma mère, pour ses encouragements continus, ainsi que mon superviseur de thèse, Dr. Béla Joós, pour son apport scientifique, et sa constante disponibilité. Son expertise et son amour pour la science font de lui une très grande source d'inspiration non seulement pour cette recherche, mais aussi pour toute ma carrière scientifique à venir.

Pour leurs conseils en temps opportuns, je remercie : James Polson, Martin Zuckermann, David Boal, Michael Wortis et Dennis Disher.

Finalement, mes salutations les plus sincères vont à tout le personnel de soutien et le corps professoral de l'Université d'Ottawa.

1 Introduction

Fluid bilayer membranes made of lipids separate the contents of cells from their surroundings. The stability of those membranes under external perturbations is consequently of vital importance. In particular, if ruptured the functions of the cell will be disabled. Naturally, a red cell bilayer membrane may rupture to free hemoglobin, protein responsible for oxygen transportation in blood. This particular rupture, called blood cell hemolysis is achieved through thermal swelling of red cells [1]. Rupture or lysis can be achieved in a number of ways: mechanically, by suction through a pipette, electrically by electrocompressive forces [2–7], and by osmotic swelling of the cell [8–10]. With the rapid progress in microscopic manipulation techniques, pore formation can also serve a positive purpose in drug delivery and gene therapy [11–13]. The number of experimental studies of the mechanical properties of the cell is rapidly increasing and we only quoted a few examples of recent work [14].

Several models of membrane rupture have been developed over the years. Most of them derive from a model suggested by Litster a quarter of a century ago [15]. It explains the stability in terms of a surface energy and a pore edge energy. The model defines a critical pore size and an energy barrier for the creation of an irreversibly growing pore. It has been extended and applied by other groups in particular to electric breakdown situations [3, 16, 17]. Shillcock and Boal investigated the effect of temperature on membrane stability [18]. Temperature increases the entropy of the pore lowering its free energy. They show that even at zero tension, edge energy is required for stability.

In our work, we derive the energy of a finite size membrane under lateral expansion. We then consider pore creation as a thermally activated process [19]. There is an important entropic element implied in a nucleation process. The model developed is for a bilayer whose relevant physical quantities have their origin in the hydrophobicity of the lipid tails [20]. These quantities are the area compressibility (for expansion) K which results from the increased exposure of the hydrophobic tails as the membrane is stretched, and the edge tension λ, (or edge energy per unit length) which is the result of the exposure of these tails to water along the edge of the pore.

In the model, the bilayer is characterized by an energy per site, essentially one molecule in size. It incorporates stress relaxation as a pore is created and grows. Pore-pore interactions are automatically taken into account. An equilibrium phase diagram is constructed determining regions of no *pores*, non-critical pores (or *protopores*), and *rupture*. The phase diagram is in terms of the temperature and the area expansion of the membrane. The critical temperatures scale with the strength of the hydrophobic interaction which to first order is linked to the length of the tails of the lipids. This hydrophobic interaction is obtained from parameters appropriate to pure phosphatidylcholine (PC) [21, 22]. Our model applies to low strain rates since the model assumes that the membrane remains in mechanical equilibrium as it is stretched. We predict the rupture scenario for normal membranes, membranes made of unsaturated lipids, very thick membranes, and for membranes in presence of polar peptides. For these systems, our results are consistent with recent experiments. Finally, we add peptides in the membrane, which are able to lower the edge energy by sticking to the

edge of a pore. We observe long-lived pores present on the membrane for low area expansion, which is again in accord with experiments.

The next chapter is a general review of bilayer membanes. The following three develop the model. We first lay its physical foundations, and then present our Ising-like model, which is solved using Monte Carlo simulations. We continue with the results and finish with a discussion of the possible extensions of the model.

2 Bilayer membrane structure

2.1 Phospholipids

Phospolipids present in membranes consist in two fatty chains linked together at one end by a polar head group. In opposition to the tails, which are highly insoluble in water, the polar section of these lipids is highly soluble in water. Molecules with this hydrophobic-hydrophilic duality are also known as amphiphiles, or diblock-copolymers in the case of very large molecules. Each tail of a typical biomembrane molecule starts with an acyl group, followed by 12 to 24 pairs of hydrocarbon. These chains are then attached via an ester linkage to a three-carbon glyceride group, as shown in figure 1. Finally, the glyceride attaches via a phosphate linkage to one of several common polar head group.

Very crudely, a phospholipid, or just lipid from now on, can be thought of as a cylinder [20]. The height of the hydrophilic head region is roughly 0,5 nm, while the height of the tails fluctuates around 1,5 nm, depending on the number of hydrocarbons present in the chains. This corresponds to about 0,1 nm per C–C bond along the tail. The cross section area of the cylinder is about 0,6 nm^2. This area corresponds to the surface a lipid occupies in a bilayer membrane.

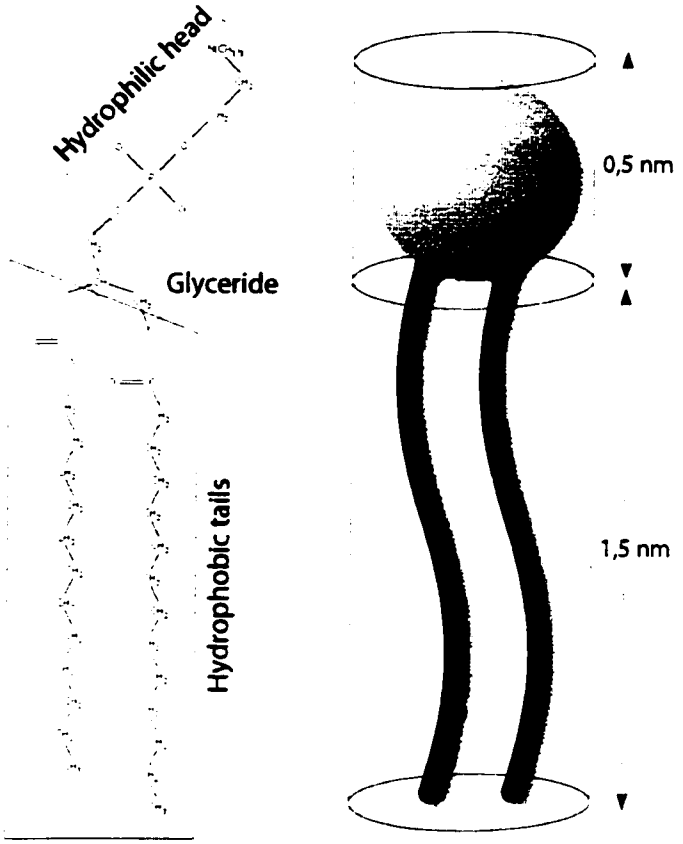


Figure 1: *Chemical composition and molecular shape of a typical phospholipid, DMPC = dimyristoyl-phosphatidylcholine. The shape of those lipids can be compared to a cylinder.*

Different experimental techniques have been used to study their physical properties, such as dimension. To illustrate the rich history of membrane study and to motivate the fact that work is still to being done, Nagle et al. [23] have regrouped different literature values of the cross section of DPPC in the same experimental conditions. Those values, shown in table 1, are associated with the experimental technique used to obtain them. The authors also point out the difficulty of creating theoretical models of membranes based on experimental results.

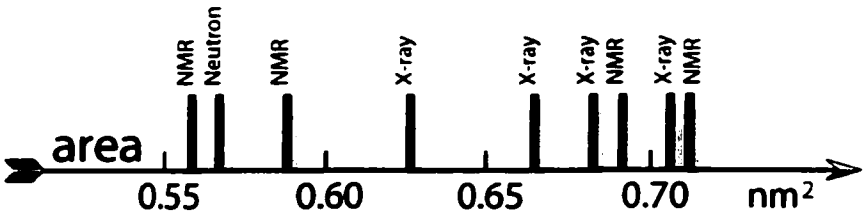


Table 1: *Cross section area of a lipid molecule in a bilayer membrane. There is a 30% fluctuation between the lowest and the highest measure. An average of 0,6 nm^2 will be considered thereafter.*

In biological membranes, phospholipid chains are normally saturated (only single bonds between carbons) or at the most mono-unsaturated (only one double bond). However, in some specific tissues, like the brain for instance, membranes rich in poly-unsaturated lipids are found [22]. As double covalent bonds are more rigid than single bonds, the tails of highly unsaturated lipids are less flexible than those of fully saturated. The degree of unsaturation has then an effect on the tension to rupture and bending stiffness of membranes [21], even though it has no effect on the membrane elasticity, as one would naively expect [22]. More specifically, lower tension to rupture were found to occur when two or more double bonds separated by a unique single bond (C=C–C=C) appear in one or both chains. Our model has an explanation for this invariance. Olbrich et al. [21] have also found that bilayers made of phosphatidylcholine (PC) lipids rich in unsaturated lipids are more permeable and much weaker than typical saturated or mono-unsaturated membranes. Unsaturation of phospholipids is therefore an important issue in membrane rupture.

2.2 Membrane structure

Phospholipid molecules are primarily held together by covalent bonds. With this in

mind, it might be thought that a lipid membrane is held together by covalent cross-links. In fact, this picture is wrong. Except for weak van der Waals bonds, phospholipids are not attracted to one another naturally.

On the other hand, bulk water is a network of loose hydrogen bonds, an interaction much stronger than Van der Waals interactions. In water, even if the formation of a hydrogen bond lowers the energy of the mixture, it may also lowers the entropy if the new configuration is not highly degenerate, such that the free energy ($E - TS$) of the system is increased. If a hydrophobic intruder is immersed in bulk water, water molecules can react in two different ways. They can either ignore the intruder and forego the hydrogen-bond formation, thus raising the energy E, or form a special hydrogen-bonded cage around the hydrophobic molecule, now lowering the entropy S. In both cases, the free energy of the system is increased. This is mainly why oil and water will phase separate when put together.

If a large quantity of phospholipids is immersed in pure water, it will self-assemble in a micellar, crystalline, bilayer or multi-layer structure, depending on the type of lipids used. Examples of such structures are shown in figure 2. Some of those structures are unique, but most of them are actually bonded bilayer membranes (figure 2c and 2d) [20], made out of cylindrical shaped lipids. The lipids forming micelles (figure 2a) are more conical in shape, whereas those forming cylindrical micelles (figure 2b) have geometries more apparent to a deep pie section [14]. Nevertheless, all those structures are such that no hydrophobic tails are exposed to water.

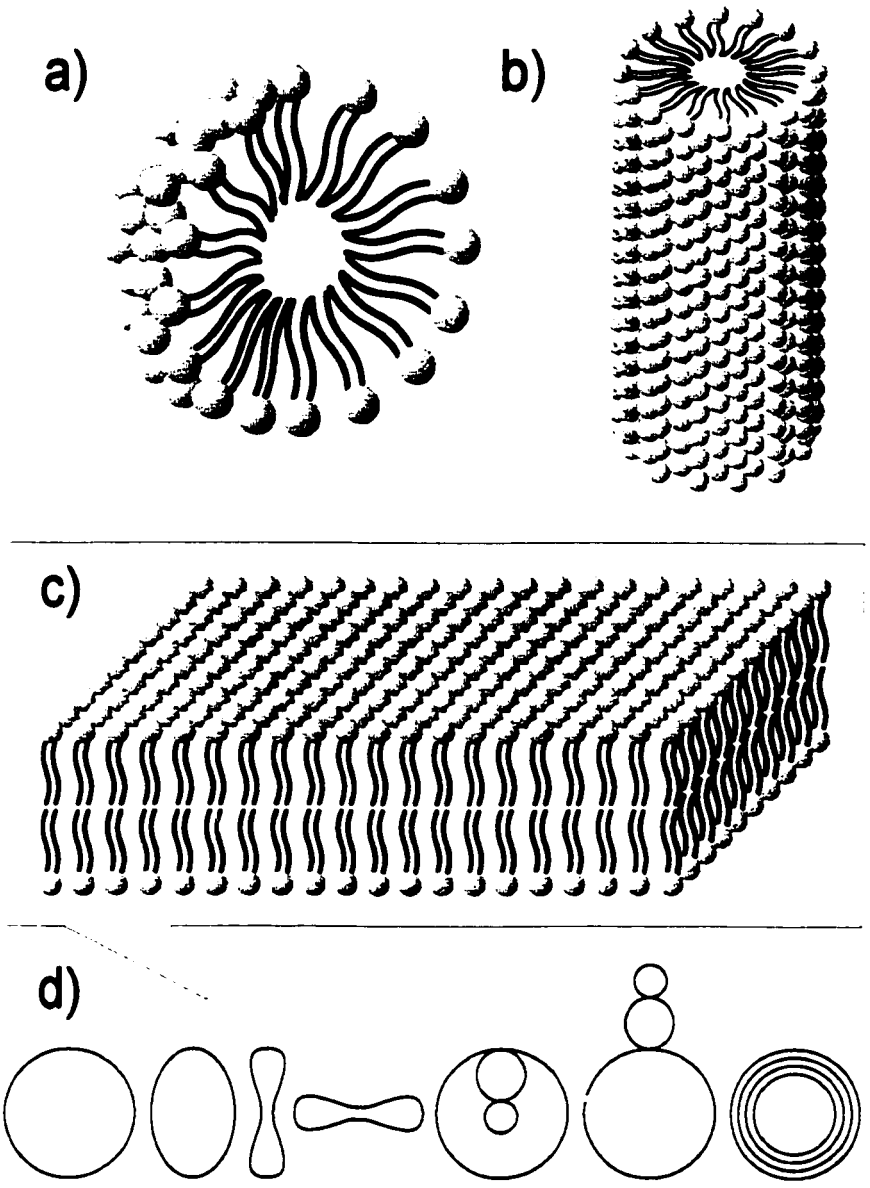


Figure 2: *Different phospholipid networks. Some structures are unique, like the spherical (a) and cylindrical (b) micelles. However, the bilayer membrane (c) can bind up to form various tridimensional shapes (d). All the forms illustrated in (d) have been produced in laboratories. They all have the vertical axis as an axis of symmetry.*

In a bilayer membrane, the tails are trapped in a sandwich between two planes of hydrophilic heads. Emphasis has to be put on this since there is no real interaction

between lipids. Nevertheless, one can associate a binding energy between them through the hydrophobicity of the tails. However, this hydrophobic binding energy per area, or hydrophobicity from now on, is not easily obtained experimentally. For phospholipids, a pure hydrophobicity can be obtained from the cost in free energy for letting a lipid swim freely in water. We say it is pure because this hydrophobicity is estimated from the real interactions between water and the different atoms of the hydrocarbon tail. This hydrophobic energy per surface area is known to be around $\sigma_{pure} = 40$ mN/m, or expressed in units of the room temperature ($T_{room} = 300$ K) thermal energy, $\sigma_{pure} = 10\ k_B T_{room}/nm^2$ [20, 24] ($k_B T_{room} = 4.14\times10^{-21}$ J). Since the lateral surface of the lipid tails is about 4 nm^2, the cost in free energy for letting a lipid float in bulk water is about 40 $k_B T_{room}$. This justifies the formation of bilayer membranes.

A water molecule can be crudely modeled as a sphere of radius $r = 0.1$nm. The energy cost for a water molecule entering the hydrophobic region of the membrane is roughly $4\pi r^2 \sigma_{pure} \cong k_B T_{room}$. Thus, water diffusion through the membrane is easily activated thermally. This assures membrane permeability. Since water is omnipresent in the membrane, the pure hydrophobicity alone cannot be used to obtain the interaction between the lipids of a hydrated membrane. Thus, an *apparent* hydrophobicity σ will have to be considered, which takes into account the permeability of the membrane.

In a water-oil mixture, phospholipids form a monolayer at the interface. The hydrophobic carbon tails face the oil, while the hydrophilic heads are in favorable contact with water. Monolayers can also form on a air-water interface.

Since there are no tangible interactions between lipids, biological membranes are normally in a liquid crystalline phase L_α. This phase is an important feature of most biological systems. In this phase, the lipids are partially oriented in a triangular network (6 closest neighbors), and the tails are flexible. It must be noted that the lipids are not fixed, but free to move. They can even, in the most extreme of cases, swap from one layer to the other, a phenomenon called flip-flop. However, the energy required for a lipid to flip-flop can be estimated to about 20 $k_B T_{room}$, which means that flip-flops are a rare, not to say improbable phenomenon [20]. Liquid, or fluid membranes are also compressible.

In laboratories, at lower temperatures, it is possible to create membranes in the straight gel phase L_β or the tilted gel phase $L_{\beta'}$ (figure 3). Phospholipids of the chlorine group (PC's) can perform the phase transition from $L_{\beta'}$ to L_α by passing through a ripple phase $P_{\beta'}$ [25]. The ripple phase $P_{\beta'}$ is characterized by considerable periodic undulations on the membrane. In this present work, however, only L_α fluid membranes will be considered.

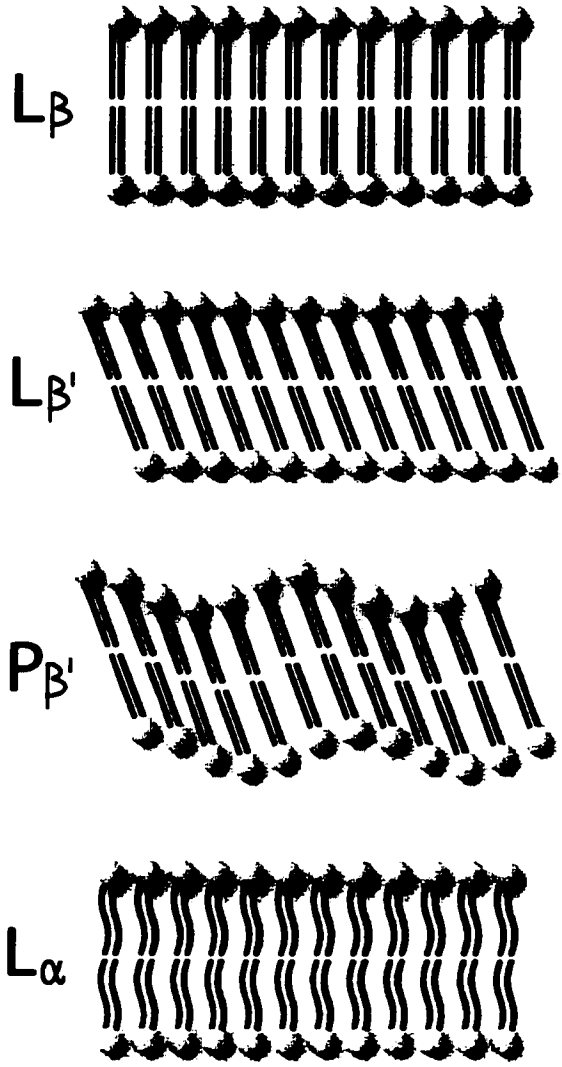


Figure 3: *Different phases of lipid bilayer membranes. At room temperature, lipids are normally in the L_α phase, where the membrane is behaving mostly like a liquid, and the tails are somehow flexible.*

2.3 Membrane Rupture

For the fluid bilayers, there is a significant degree of consensus on the order of magnitude of the quantities involved in membrane rupture. For PC bilayers, area expansion seems to be only 2% to 4% before rupture. The corresponding external tension per unit length

is of the order of 10^{-3} to 10^{-2} N/m [4, 21, 22, 26, 27]. It also seems that in liquid membranes, rupture occurs via the nucleation of holes [28]. More precisely, as the surface tension goes up, the membrane will first show weakness with the apparition of small unstable pores with life-times as short as a few nanoseconds. At a critical tension, a large stable pore opens, relaxing almost entirely the membrane (figure 4). This is rupture. Pore sizes can range from nano- to micro-meter in radius. Those observations are consistent with what is expected from structural considerations. Solid membranes as they go from brittle to ductile, experience rupture kinetics which evolve from processes dominated by crack formation to hole formation [28].

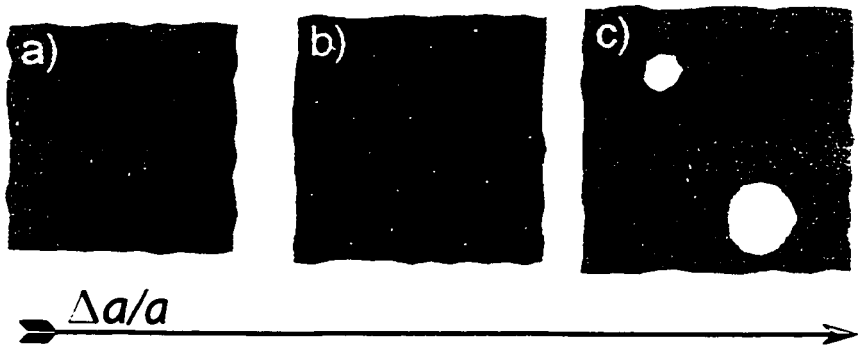


Figure 4: *Behaviour of a membrane for different expansions. At a critical tension, the membrane goes from tense to relax by a few large holes. Before this critical tension, a few small unstable pores, protopores, might be observed.*

As mentioned earlier, rupture can be achieved mechanically, and electrically by manipulations called electroporation. The latter is based on the fact that hydrocarbon chains in the middle of the membrane have a very high permeability. Therefore, the entire bilayer can act as a capacitor. If we apply a transverse electric field across the membrane, charges of opposite signs will accumulate on each side. This will create a

transversal positive pressure on the bilayer. Since the volume of the bilayer is essentially preserved, this pressure will be transformed into lateral tension (figure 5). In electroporation experiments, pores can last several microseconds [2–5, 29]. Using mechanical means, namely micropipette extrusion on vesicles, pores can be kept open for several seconds before resealing [30, 31].

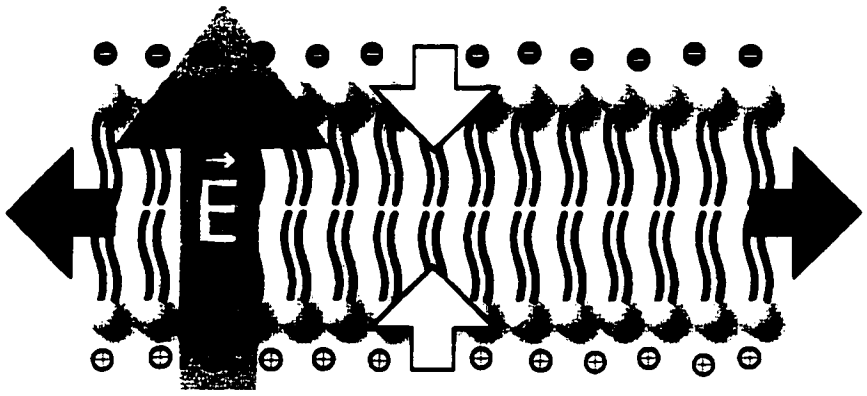


Figure 5: *Electrical breakdown of bilayer membranes. The accumulation of opposite charges on each side of the membrane induced by the external electrical field creates a transversal pressure. Since the volume of the membrane is mainly preserved, this pressure is converted in lateral forces.*

2.4 Proteins and peptides

Real biological membranes are more complex than pure lipid bilayers. A cell membrane, for example, has to be able to establish controlled communication between the cell and the surrounding. Proteins are able to give access to certain chemicals in, or out the cell [32]. Those proteins can be pictured like locked doors all around the membrane, and the corresponding chemical is someone with the key. In order to act as channels, these proteins have to be large.

Smaller proteins, peptides, can also create channels or even penetrate in a bilayer [33, 34]. The mechanism is although very different. Peptides are made from a small number of amino acids, about 3-5. They are therefore incapable to act as channels themselves. Their role will then be to activate hole formation on the membrane. For the purpose of this work, peptides can be classified in three categories: plain, amphiphilic and cationic. Plain peptides are not charged and not polarized. In normal circumstances, they don't interact with membranes.

Amphiphilic peptides, as their name suggest, also have a hydrophobic-philic duality. Unlike phospholipids, however, this polarity is not longitudinal, but rather "side to side". One entire side of the molecule is then hydrophobic, as the other is hydrophilic. If a protopore is created, an amphiphilic peptide can stick on the hydrophobic chains that are exposed to water. As the edge of the pore is now hydrophilic with the peptide well oriented, the energy of the pore drops significantly. With a few peptides along the edge, the little pore becomes prematurely stable, and can act as a channel. This channel, however, is not controlled, and any small molecules can go from in and out the membrane as those protopores are opened. This is generally not good for a cell, as it normally kills it by letting almost everything leave the cell. Some very strong peptides, like melittin present in bee venom, don't even need an existing protopore to create a channel in the membrane [34]. When a group of those peptides is near the membrane, they simply sink all together in to form a wrapped hole [35], as shown in figure 6. Some antibiotics also use this approach to neutralize bacteria [36].

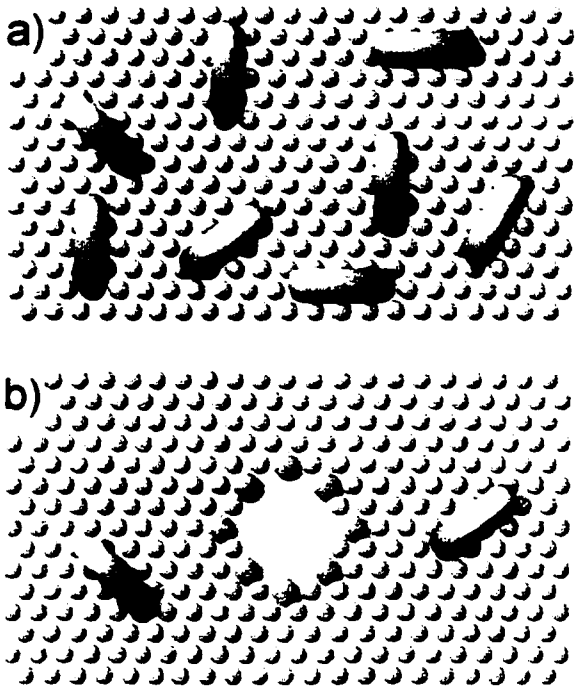


Figure 6: *Peptide insertion around a preformed hole. This normally unstable pore is now able to act as a chemical channel.*

Finally, cationic peptides are positively charged peptides and can penetrate a cell membrane. As they are charged, once through the membrane, the chemical balance of the cell is altered, and the cell dies.

3 Model for membrane expansion and rupture

3.1 Model introduction

Upon area increase, stress will build up in the membrane. As mentioned earlier, being a liquid the most likely stress release mechanism will be the formation of holes or pores. Shillcock and Boal, as mentioned in the introduction, investigated one aspect of temperature effects: the decrease in the free energy of the hole as its edges meander [26]. They considered a two-dimensional, self-avoiding, tethered manifold consisting of spherical beads (vertices) linked by flexible tethers (bonds). A hole was generated and its time evolution was studied by Monte Carlo simulation. But at finite temperature, bonds on the edge of the hole can be broken or reformed, and the latter can change shape and size. Shillcock and Boal find that even at zero tension, the tethers for stability require a minimum binding energy per unit area λ. Their estimate is $\lambda^* a = 1.24\ k_B T_{room}$, where a is the molecular cross section surface on the edge of the pore.

Another model considers thermal fluctuations of the membrane when it gains energy [37]. The normal modes appearing from the boundary conditions locate the regions of high curvature, more sensitive to rupture. Netz and Schick have also developed a mean-field theory of the fluid bilayers as stacks of diblock copolymers [38].

In this study, the interest is on developing a simple, yet complete model for the scenario of a full-scale membrane rupture. We consider the rupture as a thermally activated nucleation process through a model amenable to a Monte Carlo simulation. So for a given value of the expansion and temperature, there would be a distribution of pore

sizes and locations. The single hole analysis of Shillcock and Boal is therefore not suitable for this purpose.

However, like Shillcock and Boal, our work will lay its foundation on a model developed by Litster. As previously mentioned, this model explains membrane stability in terms of a surface energy and an edge energy. [15]. Under lateral stretching a membrane will, by the creation of pores, reduce its surface energy and increase its edge energy. The last increase is from exposing the hydrophobic lipid tails to water along the pore edges. But as long as the total edge energy created by the pores exceeds the gain in surface tension, the membrane will be stable. Litster was working on the limit of an infinite membrane, thus on an infinitely compressible membrane. In our finite size model, the surface energy will be expressed in terms of the membrane compressibility, as suggested by Israelachvili [24]. The following section is therefore an extension of this model to the case of a finite stretched membrane.

3.2 Pore creation

Let us imagine that a hole has appeared through thermal induced stress fluctuations. The question will be whether the gain in energy occasioned by the relaxation of the surfactants will counterbalance the energy loss through exposure of the tails of the molecule to the solvent. If this is the case, the hole will continue to grow into a large pore that will permit the system to relax almost entirely. For now, we assume the process to be purely planar. Whether fluctuations in the third dimension are important is an open question. At first thought they do not seem predominant. Assuming an elastic regime for

small expansions of the membrane on a lattice of relaxed size a_m, the energy associated with a stretching area of the membrane is given by

$$E_m = \frac{1}{2} K a_m \left(\frac{\Delta a}{a} \right)^2 \qquad (1)$$

where, K, the compressibility, is the curvature of the potential and $\Delta a / a$ is the strain of the membrane. For small expansion, the compressibility can be found experimentally from various techniques described previously to stretch the membrane. The apparent compressibility is the measure of the slope of the stress-strain curve of a membrane under a small tension, that is the slope of the curve

$$\tau = K \left(\frac{\Delta a}{a} \right), \qquad (2)$$

where τ is the surface tension. However, since those membranes are fluid, thermal transverse undulations are usually present. For small stretches, a fraction of the tension does not induce any strain, but rather brings back the membrane in its plane [39]. The real compressibility is in many cases also obtained by the slope of the stress-strain curve, but only once the undulations have been ironed out. For many lipids, this compressibility increases to about 243 mN/m when the bending motions are accounted for [22]. In table 2 we have grouped values of known compressibilities for common lipids.

Lipid	Tail length	Real K (mN/m)	Apparent K (mN/m)	Reference
DAPC	20	250±10	183±8	Rawicz et al., 2000
DGDG	23	—	160±7	Evans and Rawicz, 1990
DMPC	14	234±23	150±14	Rawicz et al., 2000
	29	—	145±10	Evans and Rawicz, 1990
	30	—	140	Koenig et al., 1997 [1]
DOPC	21	265±18	237±16	Rawicz et al., 2000
DPPC	18	235±17	209±14	Rawicz et al., 2000
DTPC	13	239±15	153±13	Rawicz et al., 2000
SOPC	15	—	190±20	Needham and Nunn, 1990 [1]
	18	235±14	208±10	Rawicz et al., 2000
	30	—	220	Koenig et al., 1997 [1]
Egg PC	—	—	170	Zhelev, 1998 [1]
Red cell	—	—	450	Evans and Waugh, 1977 [1]

Table 2: *Experimental results for the real and the apparent compressibility for different natural lipids with different lengths. The number of carbon atoms present in each tail represents the length of the lipids. In all cases (except red cell membrane), the membranes were synthetic and pure. The reference for each data is also included. Abbreviations:*

DAPC = diarachidonyl-phosphatidylcholine;
DGDG = diagalactosyl-diacylglycerol;
DMPC = dimyristoyl-phosphatidylcholine;
DOPC = dioleoyl-phosphatidylcholine;
DPPC = dipetroselinoleoyl-phosphatidylcholine;
DTPC = ditridecanoyl-phosphatidylcholine;
SOPC = 1-stearoyl-2-oleoyl-phosphatidylcholine;
egg PC = egg phosphatidylcholine.

[1] : *Values indirectly taken from David Boal's book* [14].

But if a hole is inserted, some of this stress will be relieved. Nevertheless, the edge of the new pore will now be exposed to water. As the tails are hydrophobic, this will result in an increase of the energy proportional to the perimeter of the pore. For simplicity, we will assume the pore to be circular. In the presence of a pore, the energy of a stretched

membrane then becomes

$$E_m(a_p) = 2\lambda\sqrt{\pi a_m}\left(\frac{a_p}{a_m}\right)^{1/2} + \frac{1}{2}Ka_m\left(\frac{\Delta a_m - a_p}{a_m}\right)^2, \quad (3)$$

where a_p is the area of the pore and Δa_m is the total expansion of the membrane. The parameters λ represents the edge tension (the effective edge energy per unit length). With the use of hydrophobicity, the edge tension can be rewritten as

$$\lambda = 2h_t\sigma, \quad (4)$$

where $2h_t$ is the hydrophobic thickness of the bilayer, and therefore h_t is the length of the lipid tails. The first term in equation 3 is the total edge energy, which represents the loss of energy on the perimeter of the circular hole. The second term, the surface energy, is the loss in energy resulting from the relaxation of the membrane induced by the hole, which augments the density of particles in the rest of the membrane

The energy barrier to rupture is given by $\lambda^2\pi / K(\Delta a / a)^{-1}$. In the assumption of large systems, this barrier is located at a pore area of $a_p = (\lambda a_m\sqrt{\pi} / K)^{2/3}$. When a_p exceeds that critical value, the pore will grow till the membrane reaches the minimum energy configuration near $a_p = \Delta a_m$. If we use an edge tension λ of the order of 10^{-7} mN [20, 30], this argument predicts a barrier height of the order of 10^4 k_BT_{room} for experimentally observed stretches, such as $\Delta a / a = 4\%$ and a membrane of total area 1 μm^2, typical for phospholipid membranes. Even if the entropic components were very significant as mentioned above, it is not sufficient to decrease significantly this large barrier. This

illustrates again that membrane values can be somehow misleading: reported experimental data of rupture do not lead to rupture! In our opinion, the line energy here is over-estimated since the pure hydrophobicity is used to evaluate it. The *real* hydrophobicity of the membrane should be considered here since water is already present in the membrane.

3.3 Compressibility, hydrophobicity and rigidity

Under expansion, more water will enter the membrane, augmenting the surface energy through the compressibility. The latter has then also its origin in hydrophobicity. Moreover, the rigidity of the acyl tails can control the amount of water that penetrates the membrane at a given stretch. There is therefore a very close relationship between these three parameters. We will closely follow here some of the arguments of Wortis and Evans [20] (themselves based partly on Israelachvili's book [24]), but with values of the physical quantities updated, when required, with the results shown in table 2.

To estimate K, the curvature of the surface energy potential near its minimum, we must understand the force balance that sets the area of the lipids a. The repulsive force between lipids is strong at short distances, but falls off rapidly as a increases [20]. Following, we shall model this repulsion by an arbitrarily potential of the form D/a, where D is a positive constant. On the other hand, the attractive part of the potential is the direct consequence of hydrophobicity. More water can enter in contact with the hydrocarbon chains as the lipids are separated from one another. Assuming a linear regime for small expansions of the cross section area of a lipid a, the energy per molecule can be written as $2\sigma\gamma(a - a_0)$, where $\Delta a = (a - a_0)$ is the relaxed or equilibrium area.

The constant γ is a geometrical factor that makes the entire term $2\gamma(a - a_0)$ represents the entire surface of a bilayer exposed to water under a stretching per molecule of $\Delta a = (a - a_0)$. The factor 2 is to account both sides of the bilayer, and γ is related to the rigidity of the tails, and therefore from now on, γ will be referred as the tail rigidity. A rigidity of 1 represents tails that are completely flexible since the expanded lateral surface Δa has all been completely covered by the chains. In parallel, a high rigidity represents very stiff chains, such that a small stretch can lead to a full exposure of the tails to water. To have a geometric picture of γ, lets imagine that the stretching Δa is represented by a circle, and the effective exposed surface by a cone, as shown in figure 7. The rigidity, which is by definition the ratio of the cone's surface over the projected surface, can be written as:

$$\gamma = \frac{h_e}{r}\left(\frac{\Delta a}{a} - \frac{n_h}{N}\right)^{-1/2}. \qquad (5)$$

where $r = \sqrt{a/\pi}$ is the radius of the lipids, and $h_e = h_t - h_{ne}$ the exposed part of the tails to water (and h_{ne} the non-exposed length, as discussed earlier). n_h/N and N are respectively the number of sites that are nucleated on the membrane and the total number of sites, such that n_h/N is the number density of pores in the membrane. The entire factor inside the parenthesis is therefore the effective stretch. Fixing the rigidity for a certain type of lipid gives us h_e and h_{ne} as a function of the effective stretch.

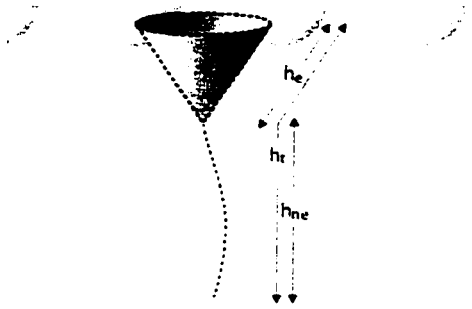


Figure 7: *Schematic representation of the total height of the lipid tails exposed for a amount of stress. Depending on the rigidity of the lipid tails, the total exposed surface to water is larger than the surface expansion.*

If we combine both the attractive and the repulsive part of the potential, we get a general expression for the energy of the membrane per particle:

$$U(a) = 2\sigma\gamma(a - a_0) + D(a^{-1}) + U_0 . \tag{6}$$

where U_0 sets the energy scale, not relevent in our case. The requirement $dU/da\big|_{a_0} = 0$ leads to the relationship $D = 2\sigma\gamma a_0^2$. It must be emphasized that σ here represents the interaction of the lipids when present in the membrane, and therefore refers to the *apparent* hydrophobicity, in opposition to the pure hydrophobicity σ_{pure} discussed earlier in section 2.2.

By comparing the curvature of this potential around the minimum with the surface energy of equation 1, we obtain

$$K = 4\sigma\gamma . \tag{7}$$

Thus, there is a very close relation between the compressibility, the hydrophobicity and the rigidity of the hydrocarbon tails.

In our simulations for typical membranes, we use the experimentally known values of compressibility K, and set the rigidity γ to obtain rupture near $\Delta a / a = 4\%$. With the use of equation 7, we shall demonstrate in section 5.2 that the apparent hydrophobicity is considerably smaller than the pure one, which ignores the water already present in the membrane. Using the pure hydrophobicity to calculate the line energy yields to an over-estimation of the latter.

4 The calculation

4.1 The Monte Carlo model

What we are trying to simulate is a thermally activated nucleation process in order to obtain an average configuration of the system for a given set of parameters. Those parameters in our case will be the number of lipids, the temperature, and the total surface. Even though a molecular dynamics simulation would be appropriate, a model amenable to a Monte Carlo calculation seems to be more convenient since we are interested in the global scenario of rupture. Molecular dynamics may not lead to a configuration representative of the equilibrium state. More simulation would then have to be done in order to obtain an average scenario. Moreover, for very large systems, molecular dynamics are very time consuming. And detailed analytic interaction potentials are not required for a Monte Carlo calculation, making it . Overall, the Metropolis algorithm seems to be more applicable to our thermally activated process.

The Metropolis algorithm is used to generate configurations in accordance to the probability given by the Boltzmann distribution. At equilibrium, the probability for the system to be in a given state i is:

$$P_i = \frac{\exp(-E_i / k_B T)}{Z}. \qquad (8)$$

where E is the energy of the system in the i^{th} state, and $k_B T$ the thermal energy, such that $\exp(-E_i/k_B T)$ is the Boltzmann factor. The partition function Z is the sum of all Boltzmann factors. It is a normalization factor. Configurations with a high probability

density have therefore a greater probability to be generated. It is good to notice that this algorithm will not tend to generate configurations with low energies, but rather with low *free* energies, since there are many states with the same energy when the entropy is high.

For ease of computation we consider a lattice model similar to the well-known Ising model for binary mixtures, and consider the equilibrium phase diagram of the system as a function of temperature and area expansion. This will reveal the expected scenarios of rupture as the membrane is expanded slowly, quasi-statically, so that the membrane remains in thermodynamic equilibrium at every step of the way. For the discrete case, the Metropolis algorithm can be resumed in three steps. First, we select a site at random. Second, we generate a trial solution by change the identity of the selected site. In our case, this will consist in transforming particles to holes and vice versa. Finally, we accept this change with probability

$$acc(i \rightarrow f) = \min(1, \exp[-\Delta E / k_B T]). \tag{9}$$

This acceptance rule is set to respect both the Boltzmann distribution and detailed balance. Detailed balance is a general conservation law for iterative processes which ensures that there is no net flow of energy from in or out the system. It must be pointed out however that the procedure described above is not unique. One could change the acceptance rule of equation (9) and modify the selection algorithm and still have a Boltzmann distribution with the respect of detailed balance. Such a variant will be used for the insertion of peptides in the membrane in section 6.2.

We will limit ourselves to two-dimensions, and to represent an isotropic liquid, we will

use a hexagonal lattice (6 nearest neighbors). We begin by presenting the standard binary mixture model (SBMM) as applied to our problem and then describe the modifications that we have made to it to incorporate the stress relaxation that occurs when a hole is created. In the SBMM, every site can be in either of two states, in our case "a" or "h", representing respectively sites occupied by a phospholipid or a hole. The site occupancy variable s_i, is equal to 1 if the site is an "a" site and 0 if it is a "h" site. The indices i and j run from 1 to N, where N is the total number of sites. Let $J_{aa}(r_{ij})$, or its compact form J_{ij}^{aa}, be the interaction between two lipids separated by a distance $r_{ij} = |\mathbf{R}_i - \mathbf{R}_j|$, where $\mathbf{R}_i$ is the position vector of site i. In a similar way, $J^{hh}(r_{ij})$ and $J^{ah}(r_{ij})$ are defined. The microscopic Hamiltonian H consists of the sum of all interactions:

$$H = \frac{1}{2}\sum_{ij}\left(J_{ij}^{aa}s_i s_j + J_{ij}^{hh}(1-s_i)(1-s_j) + J_{ij}^{ah}\left[s_i(1-s_j) + s_j(1-s_i)\right]\right). \qquad (10)$$

Expanding equation 10, we have, regrouping terms in power of s_i:

$$H = \frac{1}{2}\sum_{ij}\left(J_{ij}^{hh} + 2\left(J_{ij}^{ah} - J_{ij}^{hh}\right)s_i + \left(J_{ij}^{aa} + J_{ij}^{hh} - 2J_{ij}^{ah}\right)s_i s_j\right). \qquad (11)$$

In a conventional canonical binary mixture, one would eliminate the two first constant terms of equation 11 since the number of each type of particle is conserved. The effective potential then becomes $J_{ij} = J_{ij}^{aa} + J_{ij}^{hh} - 2J_{ij}^{ah}$. If J_{ij}^{ah} is greater than both J_{ij}^{aa} and J_{ij}^{hh}, the system will tend to phase separate.

In our binary mixture model we keep the number N of particles (the a sites) and the total

area constant. The number of holes (h), however, is subject to vary. Our model is neither the conventional canonical or grand canonical version of the binary mixture model. In the canonical ensemble, occupied (a) and empty (h) sites are interchanged to preserve the total number of particles. The conventional canonical model would be used if we were only interested in the phase separation of a system, which is not our case. In the grand canonical ensemble, the only dynamics that would be observed is the distribution of the holes. Neither model is suitable to incorporate the relaxation created by the appearance of a hole.

We start with the SBMM in the canonical ensemble with an initial total number of particles equal to the number of sites N. We now modify the model to impose the following constraints: the total area of the membrane stays constant as holes are created and the total number of lipids stays constant. These lead to the following change in the dynamics. Occupied sites, when holes are created, now contain more than one particle, actually $N/(N-n_h)$, where n_h is the number of hole sites at a given time. The occupancy variables s_i still only take the values 0 or 1, but the energy in the membrane has to consider all the particles present in the initial membrane. As a hole site appears, the density in the membrane increases. The nucleation of holes therefore relaxes the membrane and reduces the interparticle distance r_{ij}. This relaxation represents the surface energy and will affect the J_{ij}^{aa} term in equation 10.

On the other side, the energy interaction between holes and particles J_{ij}^{ah} represents the line energy of the system. Also due to the increase in density of lipids, there are more

particles on the edge than the actual number of sites. As we will discuss in section 4.3, this will also lead to a correction of the line energy.

In summary, even if the line energy and the surface energy are both a consequence of exposing the hydrophobic tails to water, it remains natural to have them behave independently, similarly to the pore creation in chapter 3.

4.2 The surface energy

With n_h hole sites, the interparticle distances have to be rescaled to the new values

$$\tilde{r}_{ij} = \left(\frac{N - n_h}{N} \right)^{1/2} r_{ij}. \tag{12}$$

where the $r_{ij} = |\mathbf{R}_i - \mathbf{R}_j|$ are the initial intermolecular distances. As discussed earlier, the nucleation of holes changes the density of the membrane and must not remove any lipids. Therefore, the surface energy will only be considered through the relaxation due to the change in the membrane density. Supposing that the energy of an unstretched membrane is zero, we have, using equation 3,

$$E_s = \frac{1}{2} K a_m \left(\frac{\Delta a}{a} - \frac{a_p}{a_m} \right)^2 = \frac{1}{2} K a_m \left(\frac{\Delta a}{a} - \frac{n_h}{N} \right)^2. \tag{13}$$

In comparison to equation 3, this surface energy does not result from a single circular pore. In equation 13, a_p refers to the surface of the membrane that has been nucleated. It should also be clear from this last equation that if the relative number of holes n_h / N completely relaxes the imposed stretch on the membrane $\Delta a / a$, the surface energy

vanishes, as expected.

4.3 The line energy

As previously mentioned, the edge energy represents the hydrophobic contacts along the perimeter of a pore, namely J_{ij}^{ah}. The edge energy therefore depends on the location of the holes on the lattice. More precisely, the sum of those interactions cannot be reduced to a simple form like the surface energy.

As we are working on a hexagonal lattice, J_{ij}^{ah} consists in the energy of exposing one sixth of the hydrophobic surface of a lipid to water, multiplied by two for the bilayer. Using the hydrophobicity, the hole-lipid interaction can be written as

$$J_{ij}^{ah} = \begin{cases} \frac{2}{3} h_{ne} \sigma \sqrt{\pi a} & \text{if } r_{ij} \text{ is 1}^{\text{st}} \text{ neighbour} \\ 0 & \text{otherwise} \end{cases} . \qquad (14)$$

where a is the cross section of the lipids, and $h_{ne} = h_t - h_e$ the hydrophobic height of those same lipids that are not exposed to water (and h_e the exposed height). The hydrophobic energy of the exposed height h_e is included in the surface energy.

Every time a hole is created, the density of the membrane is changed through the rescaling of equation 12. There are therefore more particles on the edge of a pore than the actual number of sites. Since we are working on a discrete lattice, a correction is to be added to the line energy to correct for number of particles present on the edge of a pore. The edge energy per unit length can be written as $J^{ah}/\tilde{r}$. With this in mind, we

can write the correct expression for the line energy as

$$E_{edge} = \frac{J^{ah}}{\tilde{r}} L^{ah} r , \tag{15}$$

where L^{ah} is the number of lipid-hole interactions (between the sites), and therefore the number of sites along the perimeter of all the pores in the membrane. Thus, $L^{ah}r$ represents the perimeter of the entire interface. The equivalent expression in the model for the edge energy is however:

$$E_{edge}^{model} = J^{ah} L^{ah} . \tag{16}$$

A factor of $r/\tilde{r}$ has to be applied to the edge energy of the model to have the right expression for the total edge energy. Using the inter-particle distance renormalization of equation 12, we can write

$$E_{edge} = \left(\frac{N}{N - n_h} \right)^{1/2} E_{edge}^{model} \tag{17}$$

or, in the expanded form,

$$E_{edge} = \left(\frac{N}{N - n_h} \right)^{1/2} \sum J^{ah} \left[s_i (1 - s_j) + s_j (1 - s_i) \right]. \tag{18}$$

For a small fraction of holes in the membrane, this correction is quite small.

4.4 Programming and Analysis

To create and recover holes in the membrane up to an equilibrium point, we use the

Metropolis algorithm in the Monte Carlo simulation. The mass of a hole is zero, but the mass of the system or number of particles is conserved through the renormalization of Eq. 5. Although the occupation of sites changes as the simulation evolves, detailed balance is satisfied because each configuration of pores has a uniquely defined energy. This energy does not depend upon the path followed to reach it. The transition probability between a given initial and final configuration will also be unique and reversible. If we choose an occupied site with the same probability than an empty site, that is, if we select particles or holes according to their surface distribution; the usual acceptance rule shown in equation 9 can still be used. Finally, periodic boundary conditions are considered on the hexagonal lattice.

The approach is highly efficient and large systems can be studied, up to 10^7 particles. However, systems of 10^4 particles, which correspond to small vesicles, are normally sufficient. Usually within 1000 samplings per site, the equilibrium state is easily reached. Typical simulations for the construction of hysteresis plots, distribution of pores and fractal dimension for example can normally be run in 12 hours on a 600MHz Pentium® processor. For phase diagrams, however, simulation need to run at least 48 to offer good precision. The program is coded in Fortran 77, and is included in annex B.

5 Results

5.1 Dependence on the size of the system

As discussed earlier, bilayer membranes can form vesicles of a variety of shapes. In nature, however, they are normally present as spherical vesicles of radii between 1-10 μm [11-13]. It is an interesting issue to see the dependence of rupture on the system size, if any. It should be clear from section 4.2 that for a given relative stretch $\Delta a / a$, the surface energy per particle does not depend on the size of the lattice. On the other hand, for a given configuration of pores on large and small systems, the line energy per particle will be smaller on the large lattice since the line energy scales as the square root of the surface. However, in practice, large and small systems may not behave the same way when they rupture. If rupture can occur via the nucleation of several pores, large systems are more likely to have a larger quantity of pores. If the relaxation is distributed in several pores, of course, the line energy is increased.

As shown in figure 8, stretching to rupture almost stabilizes to about 3.9% for systems larger than 10^4 particles. For smaller systems, we observe a significant increase. As expected, this is mainly due to the increase in the number of pores at rupture, which raises the line energy (see figure 9). Furthermore, entropy increases for larger systems, which favours earlier rupture. Those plots are averages with a standard deviation taken from 10 runs. In is an interesting and opened question to know if multiple pore rupture is observed in biological membranes. It must be noted that the number of sites in our simulation corresponds to half the number of particles, since we are working on bilayer

membranes. From now on, we shall work on a lattice of 30301 sites (60602 lipids). This corresponds to a total membrane area of 18 μm^2. For a spherical vesicle, it corresponds to a radius of 1.5 μm, a small natural biological cell.

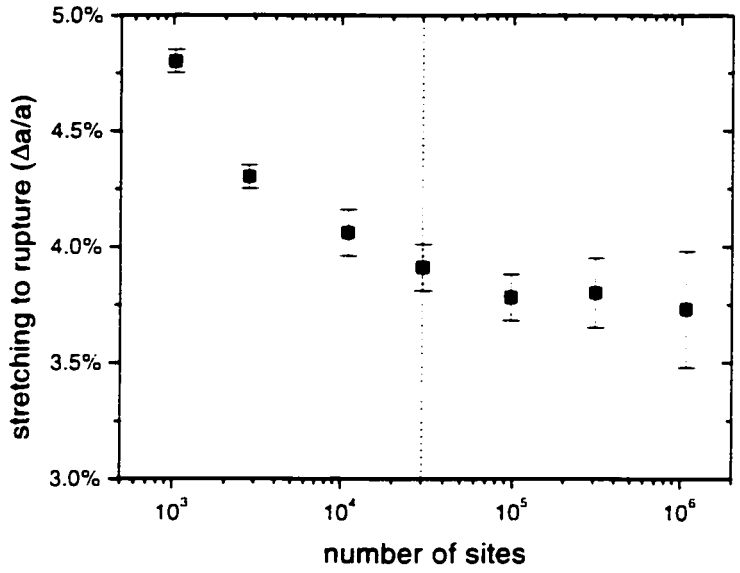


Figure 8: *The dependence of stretching to rupture on the size of the membrane. The number of sites is half the number of particles since we are working on a bilayer. This value stabilizes at expansions near 4% for systems larger than 10^4 sites. The dashed line indicates the size of the network used in the rest of the simulations.*

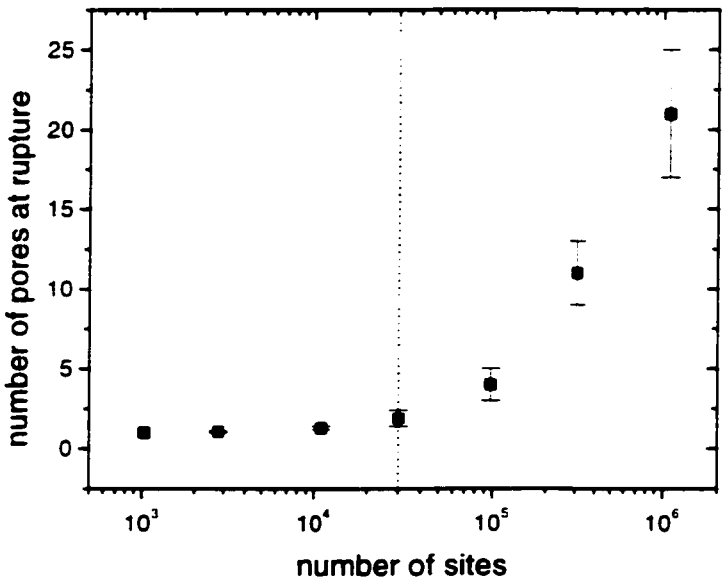


Figure 9: *The average number of stable pores in the system at rupture. The relation is linear (exponential on this log scale). The dashed line indicates the size of the network used in the rest of the simulations.*

The scenario of rupture also seems to be influenced by the size of the system. For average size systems, rupture is almost an explosion, whereas very large systems break smoothly (see figure 10). More precisely, rupture goes from a first order transition to a higher order. For systems of 30301 particles however, rupture remains clearly a first order transition. Again, this is due to the presence of many pores at rupture.

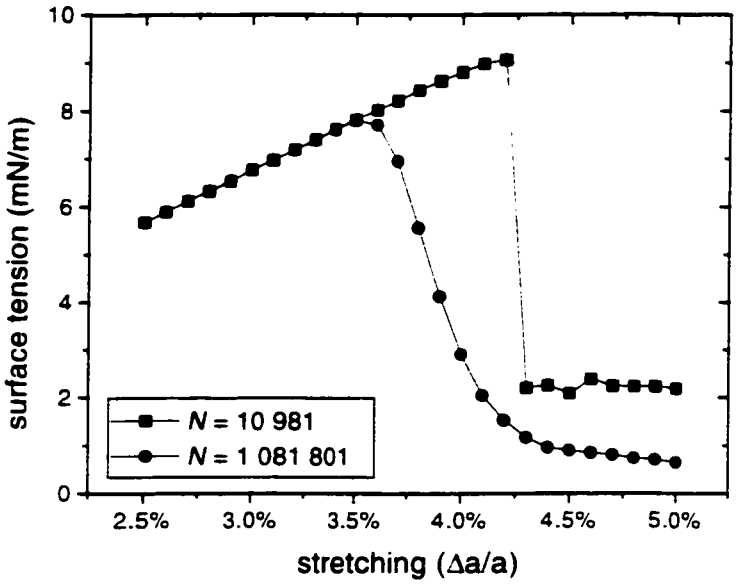


Figure 10: *Stress-strain curves of a small and a large system. Larger systems seem to break more smoothly, in comparison to smaller membranes, which literally explode. Membrane rupture for very large systems is therefore not a first-order transition.*

5.2 The rigidity and hydrophobicity of lipid tails

Depending on the rigidity of the tails of a lipid, more or less water may penetrate the membrane under stretching. For unsaturated lipids, one would then expect higher rigidities, since the tails are stiffer. Furthermore, if the tails are rigid, the membrane is also more permeable [22]. In this case, the apparent hydrophobicity of the membrane is lower, since the difference in energy from the lipid inside and outside the membrane is less important. For small variations of these parameters, we can see from equation 7 that a saturated and an unsaturated membrane made out of the same lipid group should have comparable compressibility. This was verified experimentally by Rawicz *et al.* [22] on their investigation of the effects of lipid unsaturations on membrane elasticity. This

group has found that unsaturation has no effect on compressibility. Nevertheless, unsaturation is still an important issue for membrane rupture, as unsaturated lipids are less flexible than saturated ones. Olbrich et al. [21] have discovered that rupture tension was only affected for bilayers made with lipids of two or more alternating cis-double bonds (C=C-C=C) in one or both chains. It is therefore natural to investigate the rigidity, the hydrophobocity, and therefore the rupture of unsaturated lipids. Three cases of unsaturated bilayers will be studied: lipids with one tail having two alternating cis-double bonds (1 pair), both tails having two alternating cis-double bonds (2 pairs) and both tails with three alternating cis-double bonds (4 pairs). All these unsaturated lipids also have 18 carbon atoms in each tail.

The line energy is directly proportional to the length of the lipid tails. It is therefore natural to expect higher tensions at rupture for longer chains. Membrane rupture of diblock copolymers membranes will also be studied.

Values of compressibility K are shown in table 3 for different types of PC bilayers that will be studied thereafter. In addition to those values, membrane core thickness $2h_t$, the permeability, the tension to rupture τ^*, and the stretching to rupture $\Delta a/a^*$ are also given. The typical membrane represents more or less membranes made out of the biological lipids shown in table 2. Unsaturated lipids may also be found in biological systems. The long tail diblock copolymers, however, were synthesized by Bermudez et al., and then used to build vesicle shape polymersomes [40]. The authors also point out that the compressibility of their copolymers is different from the usual 243 mN/m mainly because

the head group is different, and not because of the tail length.

Membrane	Name	Uns.	K (mN/m)	$2h_t$ (nm)	P (µm/s)	τ^* (mN/m)	$\Delta a/a^*$ (%)
Typical	(average)	0, 1	243±24	3,0±0,1	35±7	9±2	3,9±0,9 [1]
Unsaturated	SLPC:0/2	2	243±24	3,0±0,1	49±6	4,9±1,6	2,0±0,5 [1]
lipids	DLPC:2/2	4	243±24	3,0±0,1	91±24	5,1±1,0	2,1±0,4 [1]
	DLPC:3/3	6	243±24	3,0±0,1	146±24	3,1±1,0	1,2±0,4 [1]
Long tails	OE7	0	102±10	8±1	—	21±3 [1]	21±2
copolymers	OB9	0	102±10	11±1	—	28±4 [1]	28±3
	OB18	0	102±10	21±1	—	40±5 [1]	40±4

Table 3: *Compressibility and other known physical parameters of the different type of membrane studied. These parameters are respectively: the total number of cis-unsaturated bonds in the lipid tails, the compressibility, the permeability, the tension to rupture, and the strain to rupture. Values for Typical and unsaturated lipids are from Olbrich et al.* [21] *and Rawicz et al.* [22], *whereas the values for long tail diblock copolymers are taken from Bermudez et al.* [40]. *Abbreviations:*

SLPC:0/2 = 1-stearoyl-2-linoleoyl $_{\text{cis at 9, 12}}$ *-phosphatidylcholine;*
DLPC:2/2 = dilinoleoyl $_{\text{both cis at 9, 12}}$ *-phosphatidylcholine;*
DLPC:3/3 = dilinoleoyl $_{\text{both cis at 9, 12, 15}}$ *-phosphatidylcholine;*
OE7 = $ethylenoxide_{40}$*-*$ethylethylene_{37}$*;*
OB9= $ethylenoxide_{50}$*-*$butadiene_{55}$*;*
OB18= $ethylenoxide_{80}$*-*$butadiene_{125}$*.*

[1] : Values calculated with equation 4, since either the tension to rupture of the stress to rupture was given in the referred articles.

To obtain γ, and therefore σ, of the different membranes, we fit this parameter to have the corresponding stretching to rupture. For typical lipid bilayer membranes, we need a rigidity of $\gamma = 8 \pm 1$ to have rupture at 3,9% (see figure 11). From equation 7, this corresponds to a hydrophobicity of 7,6 mN/m.

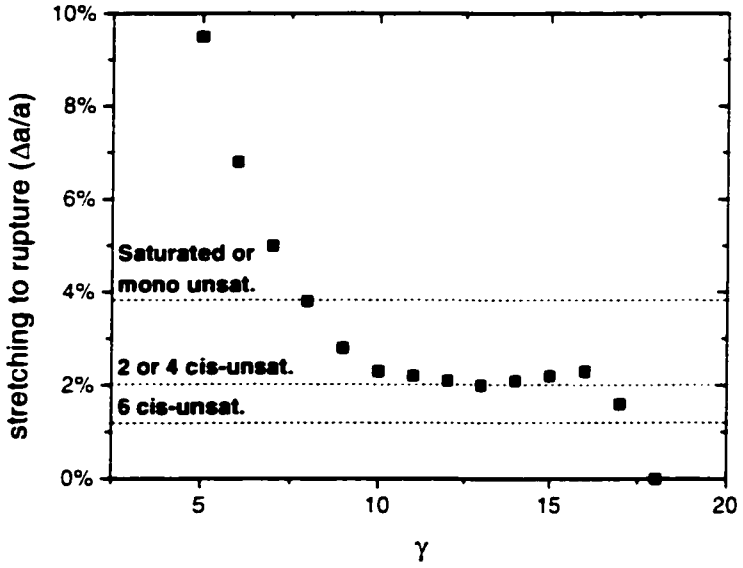


Figure 11: *Stretching to rupture for different tail rigidities for membranes with compressibility of 243 mN/m and membrane core thickness of 3,0 nm (see table 3). The dashed lines are the known expansion to rupture for membranes with different degrees of unsaturation.*

The plateau near 2% stretching in figure 11 for large γ's may seem abnormal at first sight. Even more, just before zero tension rupture, there is even a slight increase in the stretching to rupture. This effect is related to the criterion that defines rupture. Zero tension rupture means that the lipids are so stiff and the membrane so permeable that stability is not even assured. Just before this critical rigidity, the membrane is still not very stable, since the line energy per lipid is losing strength. This weak interaction between lipids raises the entropy of the system. For high rigidities, this gain in entropy will make the rupture scenario more comparable to a melting phenomenon than a pore creation relaxation. Since we have lost the ideal scenario of rupture which favours relaxation as opposed to the creation of line energy, rupture in terms of a few stable pores

is more difficult to achieve. The definition of rupture will be discussed in more detail in the next section.

Nevertheless, this plateau seems consistent with experimental observations. Unsaturated lipids with 2 and 4 total unsaturated bonds both have a stretching to rupture near 2% (see figure 11). This therefore means that even though they have the same stretching to rupture, it does not mean that they have the same tail rigidity. In fact, they should not.

For typical membranes, the apparent hydrophobicity is about 5 times smaller than the approximated value for the pure one (table 4). For highly unsaturated lipids, this ratio can be as high as 10 if we suppose that the pure hydrophobicity has not changed significantly. It would be wrong, however, to say that 80% of the membrane is flooded. This apparent hydrophobicity may also take into account other repulsive energies, such as the head-head repulsion. Moreover, σ_{pure} was approximated from geometrical simplifications, and might be smaller than 40 mN/m. Nevertheless, this noticeable difference between the pure and the apparent hydrophobicity indicates that membrane permeability is not a negligible effect.

For the long tails diblock copolymers studied by Bermudez et al., the rigidity of the two first long polymer tails (OE7 and OB9) is comparable with the rigidity of a typical lipid tail (figure 12 and 13). However, for OB18, γ is considerably higher than the previous tail rigidities (figure 14). A hypothesis can be that for very long polymers, like OB18, the tail entanglements are thought to contribute [40, 41]. This is equivalent, as a result in our simulations, to raising the rigidity of these long tails.

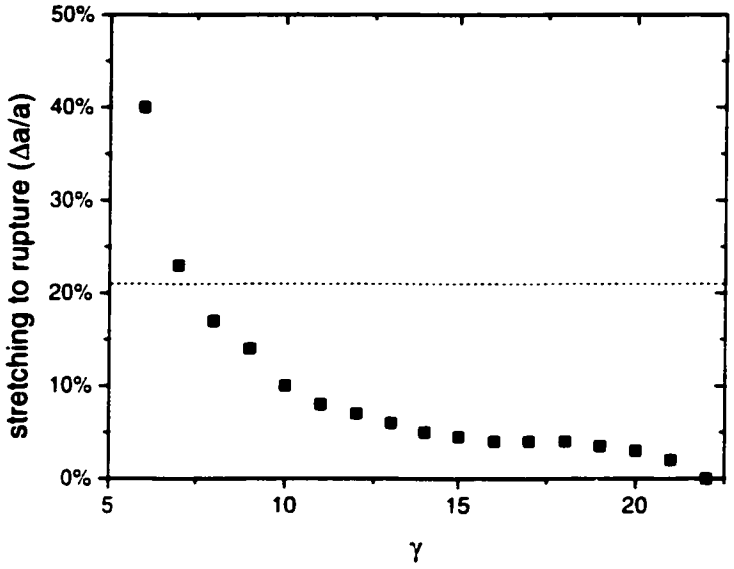


Figure 12: *Stretching to rupture for different tail rigidities of OE7. The dashed line (· · ·) is the known stretching to rupture for this copolymer.*

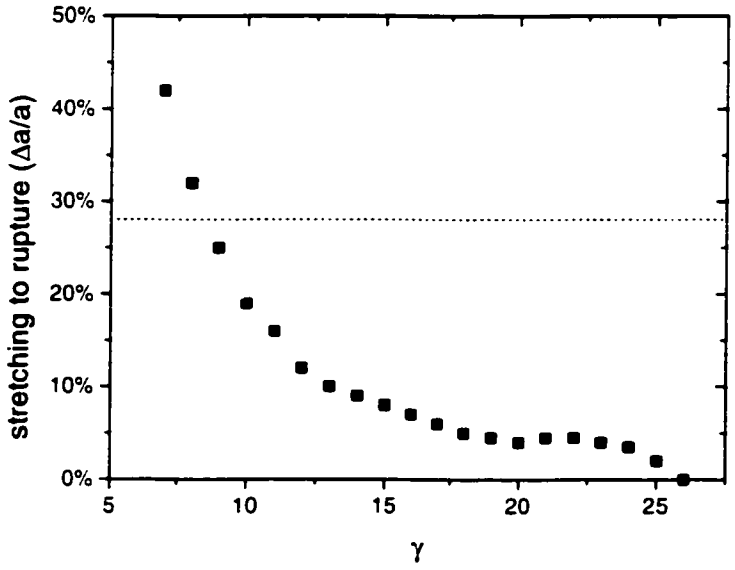


Figure 13: *Stretching to rupture for different tail rigidities of OB9. The dashed line (· · ·) is the known stretching to rupture for this copolymer.*

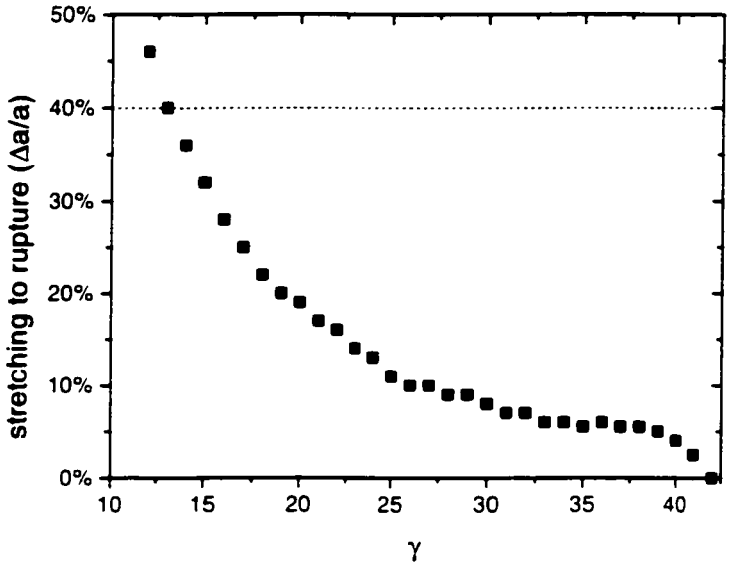


Figure 14: *Stretching to rupture for different tail rigidities of OB18. The dashed line (· · ·) is the known stretching to rupture for this copolymer.*

Membrane	Name	Uns.	γ	σ (mN/m)	σ_{pure} (mN/m)
Typical	(average)	0, 1	8,0±1,0	7,6	40
Unsaturated	SLPC:0/2	2	13,0±3,0	4,7	≈40
lipids	DLPC:2/2	4	13,0±3,0	4,7	≈40
	DLPC:3/3	6	17,5±0,5	3,8	≈40
Long tails	OE7	0	7,3±1,0	3,4	—
copolymers	OB9	0	8,6±1,0	2,9	—
	OB18	0	13,0±1,0	2,0	—

Table 4: *Tail rigidity and apparent hydrophobicity of the different lipids studied. The estimated value for pure hydrophobicity is also given as a reference.*

For the following sections, only typical biological membranes will be of interest. Therefore, γ shall be set to 8.

5.3 Phase diagram

A phase diagram was constructed to study the behavior of the membrane under surface and temperature variations. To make it biologically relevant, all the values on the diagram were approached in a quasitatic manner. Moreover, our system, offering uniform relaxation, is adapted to this scenario. Qualitatively, we can separate the membrane behavior in three regimes. The *stable* region is where none or a small fraction of the new surface was converted into holes to relax the system. *Unstable*, where relaxation is higher than a few percent but with a considerable tension remaining in the membrane. The main characteristic of this region is that the pores are short-lived and of sizes no larger than 10 particles or so. Finally *rupture*, where the membrane is about 80%-90% relaxed by only a few massive stable pores. In figure 15, the black points indicate the first order transition where the membrane tension is dropping. Even at high temperatures, this transition remains first order. This is in contrast to the continuous changes occurring between constant relative relaxation below rupture. For low temperatures and temperatures near T_{room}, rupture is clearly defined. We go directly from a high tension where less than 20% of the membrane is relaxed to an almost fully relaxed system (80% or more relaxation). Nevertheless, the limit where the total tension is released will always be considered the criterion for rupture.

For higher temperatures, a high fraction of holes is present in the membrane at very low stretches. Again, for high temperatures, this premature high density of holes is due to entropy. It is therefore almost improbable for these unstable holes to aggregate in a large unique pore. Only those large pores, which favor surface relaxation to line energy, are to

induce proper rupture, and reduce the tension. This explains the augmentation in the stretching to rupture at high temperatures, comparable to the plateau in figure 11. Finally at $T = 2.8\ T_{room}$, the membrane loses stability: stable pores are created without any tension applied. There is therefore a minimal line energy required to assure membrane stablility. Shillcock and Boal [18] have obtained a dimensionless value of $\beta\lambda r = 1.66$, where $\beta \equiv 1 / k_B T$, which is equivalent to a temperature of $T = 2.9 T_{room}$. Our results are then consistent with this minimal line tension required for stability. At room temperature, the presence of the regime offering 2% to 20% of unstable holes in the new surface seems reasonable, as protopores, or small non-stable pores have experimentally been observed at room temperature [4].

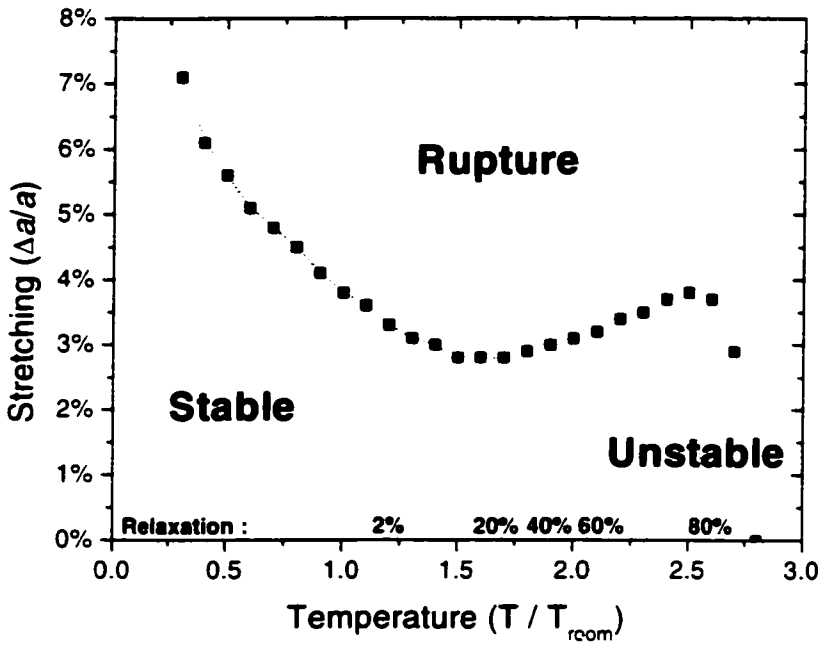


Figure 15: *Phase diagram of typical biological membranes showing the variation of the stretching to rupture for different temperatures. Hard points indicate the first-order transition to rupture, as the dashed lines are the curves of constant relative relaxation.*

5.4 Hysteresis

To illustrate the properties of the real dynamics, we show a single sample put through a full cycle of expansion and compression. This is done again in a quasi-static process, i.e., at each step the system is allowed to relax. Samples are made at $\frac{1}{2}T_{room}$, T_{room} and $2T_{room}$. Hysteresis of the stress-strain curve, the pore density, and the number of pores for different expansions will be produced to study the scenarios to rupture.

As shown in figure 16, hysteresis effects are very strong at low temperature. Rupture and healing occur through nucleation of one pore. At high temperature, hysteresis is reduced and there are many more pores present, before and after rupture (figure 18). At T_{room}, behavior is similar to the high temperature before rupture, but more comparable to low temperatures afterwards. A considerable number of pores are present in the membrane at first, but after rupture, the membrane presents only a small number of large non-fluctuating pores. In other words, at room temperature, the unstable protopores present in the membrane before tension vanish as soon as the system is able to relax through rupture. The intermediate protopore regime is very clearly observable in figures 16 and 17, when plotting the variation of the number of pores as a function of expansion. The protopores are noticeable through the spikes in those curves.

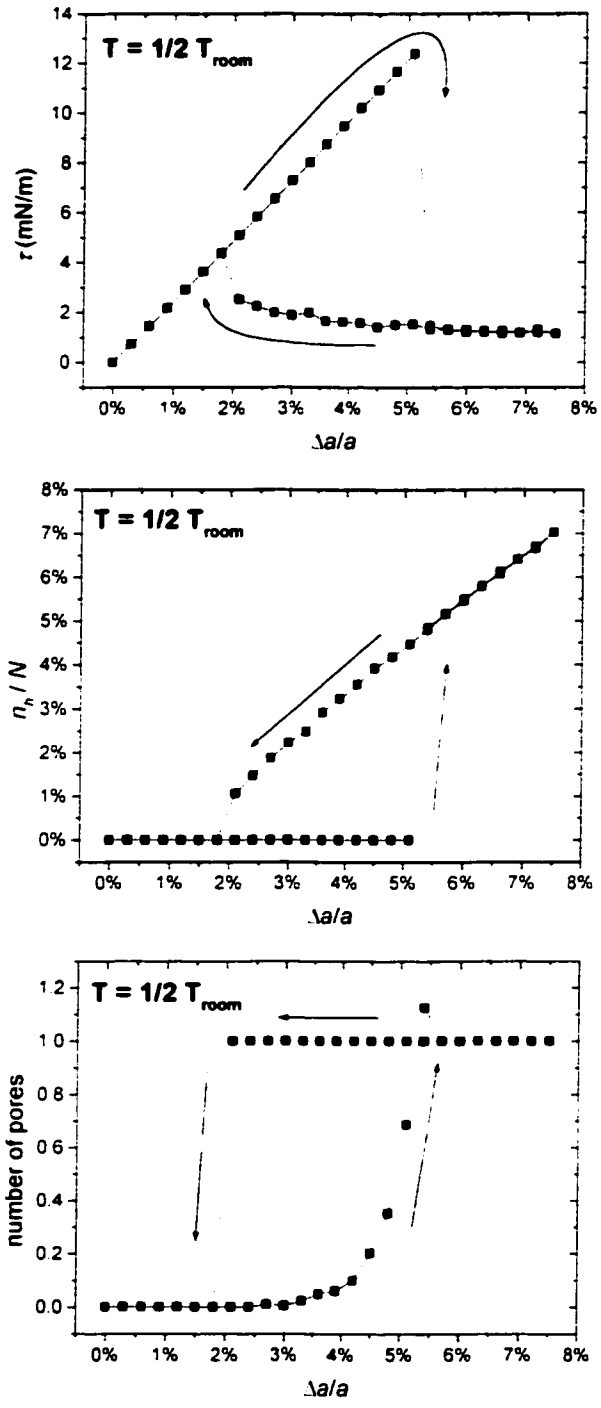


Figure 16: *Hysteresis of different quantities upon stretching at T = ½ T_{room}. Low temperature systems are characterized by a strong hysteresis, and by a low number of pores, before and after rupture.*

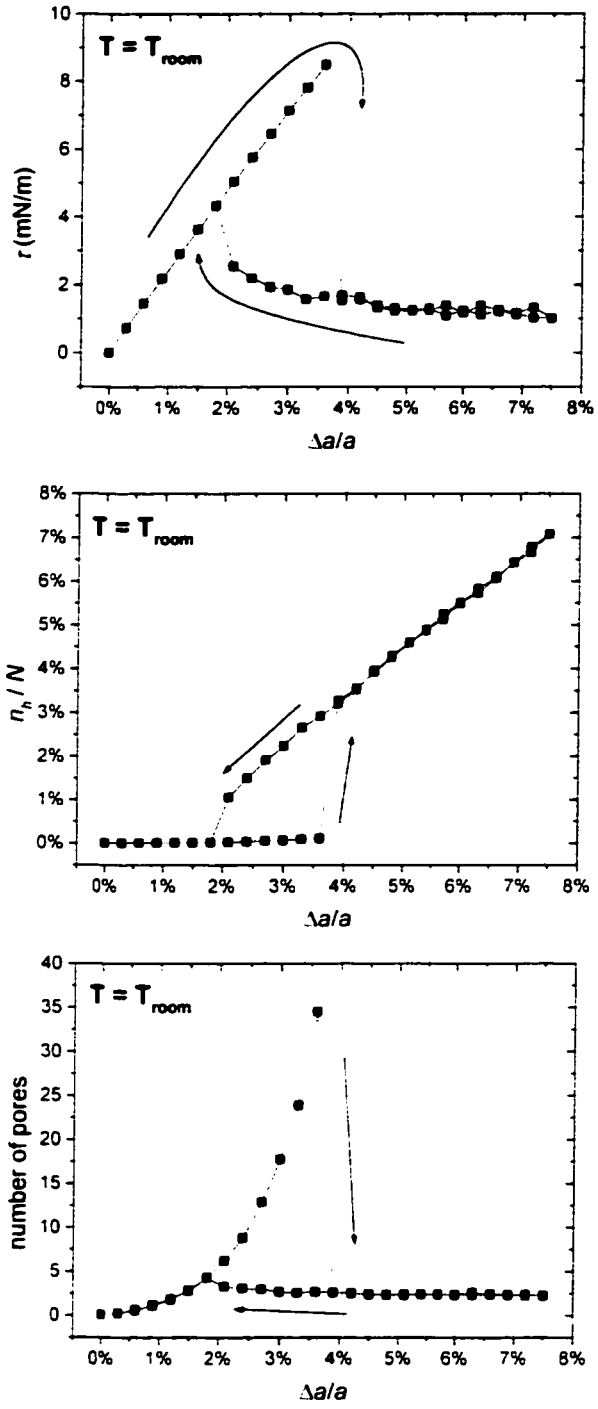


Figure 17: *Hysteresis of different quantities upon stretching at $T = T_{room}$. Room temperature systems behave like at high temperature for low tensions by having a considerable quantity of protopores present in the membrane. However, rupture scenario is similar to those of low temperature systems.*

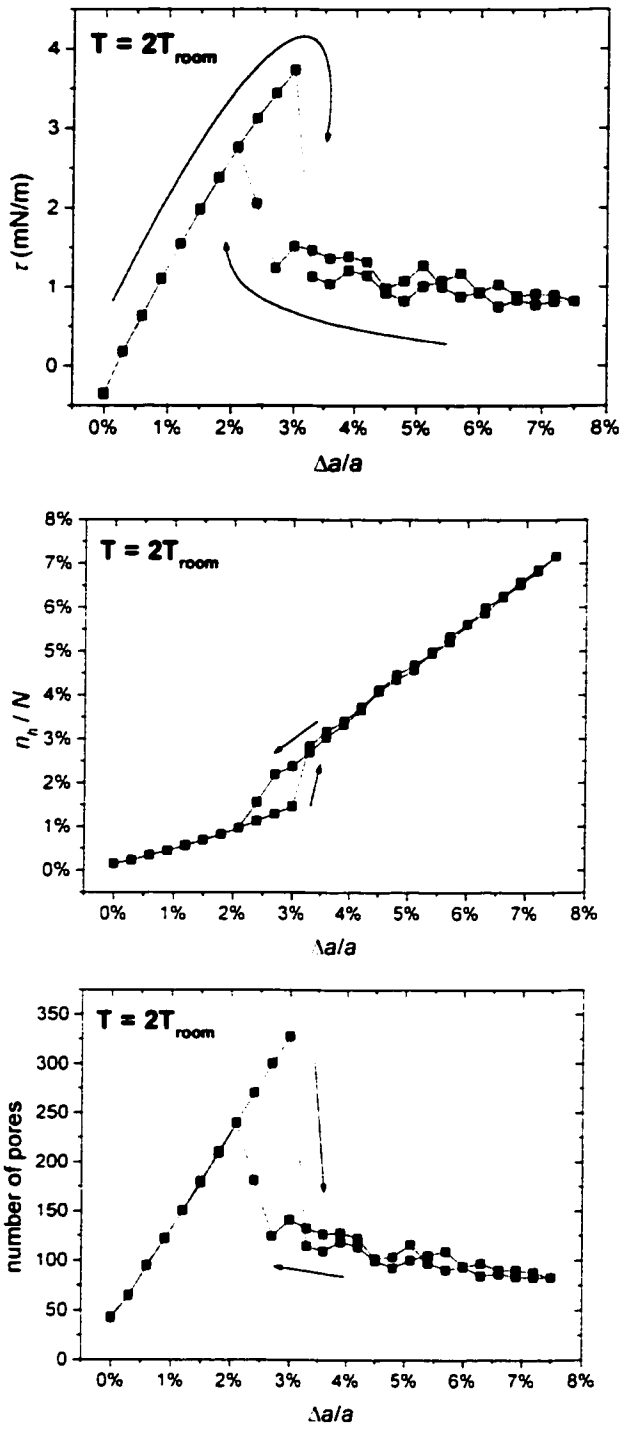


Figure 18: *Hysteresis of different quantities upon stretching at $T = 2T_{room}$. High temperature is highly entropic, and for all tensions, there is a considerable number of pores. Hysteresis effects are also less important.*

5.5 Pore size distribution and shape

To furthermore illustrate the difference in pore distribution between systems before and after rupture, we compare the pore size distribution of membranes with 2% and 6% strain, for various temperatures. On the other hand, the shape of a pore is estimated from the scaling relationship between the pore area and its perimeter. At different temperatures for a stretching of 3% (just before rupture), we break the membrane by inserting a hole at the origin. After the pore is fully opened and stable, we compute the fractal dimension of this pore to get the proportionality between the area of a pore and its circumference.

For the size distributions of the pores, it was done for systems at 3 different temperatures: T_{room}, $2T_{room}$ and even up to points where membrane stability is lost, $3T_{room}$. All the distributions are from snap-shots of the membrane at equilibrium for the given parameters. Again, equilibrium was reached in a quasitatic manner. An average distribution would not have been representative of the system. Nevertheless, these snap-shots are representative of the average sampling.

For the pore distributions of stable membranes shown in figure 19 and 20, rupture induces a drop in the number of small pores present in the system. This, of course, was expected, since a very large pore creates less line energy than a protopore for the same gain in surface energy.

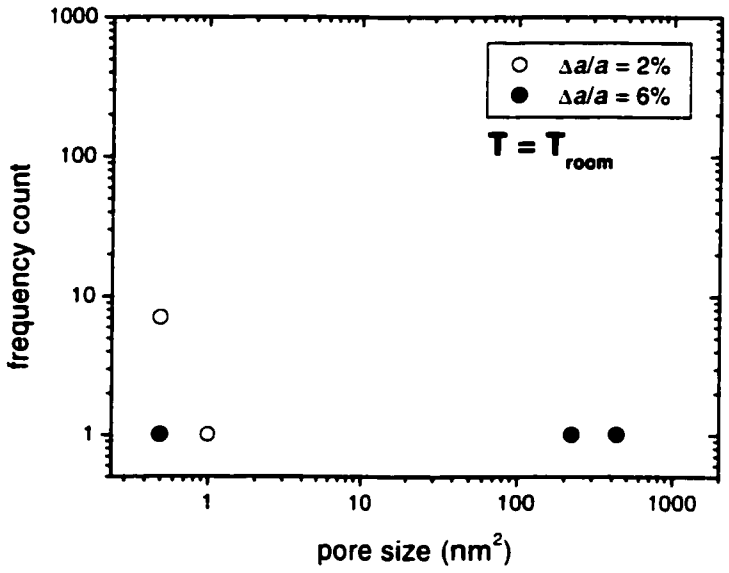


Figure 19: *Distribution of pores at room temperature before and after rupture. The presence of protopores is negligible when the membrane is ruptured.*

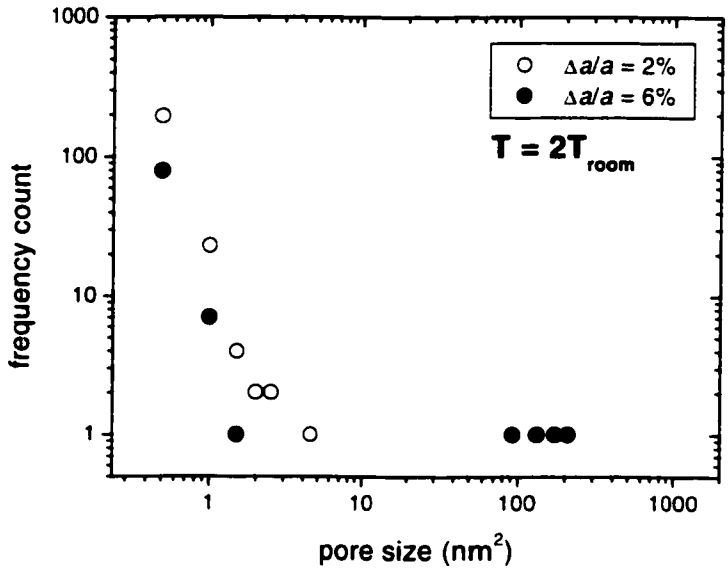


Figure 20: *Distribution of pores at $T = 2T_{room}$. Rupture is still characterized by the creation of large pores. However, a significant number of small pores are still present in the membrane after rupture.*

For unstable membranes, the appearance of larger pores in the system does not reduce the number of smaller ones (figure 21). These small pores, present at zero tension for high temperatures, are then stable due to the high entropy of the membrane. This illustrates from another perspective that a membrane at $3T_{room}$ is unstable.

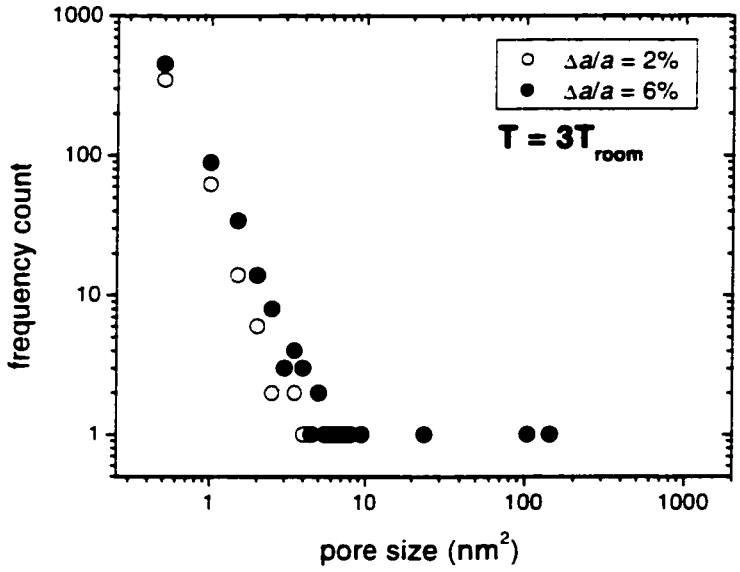


Figure 21: *Distribution of pores at $T = 3T_{room}$. Even though large pores are created at high strain, the number of small pores keeps increasing.*

From the results plotted in figure 22, we clearly see that the fractal dimension decreases as the temperature is raised. This is caused by increased irregularities along the edge of the pore. At the temperature $T = 2.8\ T_{room}$, where rupture occurs spontaneously at zero tension, one would expect the scaling relationship applicable to a self-avoiding ring [26]. In such geometries, the area of the pore should scale as $A = A_0\, l^{3/2}$, where l is the perimeter, and A_0 a proportionality factor, not relevant here. In this case, our model gives a scaling relationship of $A = A_0\, l^{1.54\pm0.10}$ in agreement with this argument. To find more

about the distribution and the shape of pores in the membrane. some snapshots are given in appendix A.

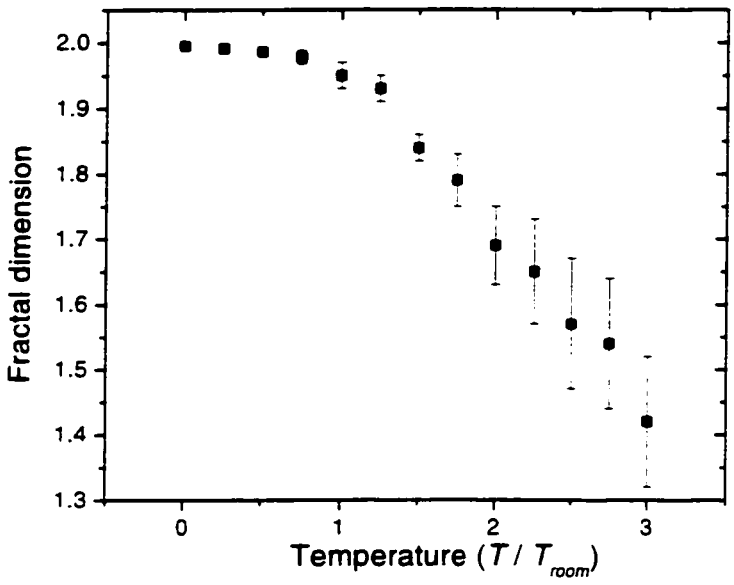


Figure 22: *The Fractal dimension of a pore in a stretched membrane. This quantity is a measure of the regularity of the shape of the pore and its edge. As the temperature is increased, the pore becomes irregular and spreads over a larger membrane surface.*

5.6 Free energy curves

Even though we have a good idea of the distribution of the pores for various equilibrium states. we cannot yet determine the critical pore size which leads to rupture. At room temperature. as showed in figure 23. the free energy is either minimal at a low concentration of very small protopores. or at rupture. This equilibrium depends on the applied stretch. Moreover. systems with expansions between 2% and 4% both have minima. Their equilibrium state is therefore dependent on the initial conditions. This bistability is easily observed in the hysteresis plots.

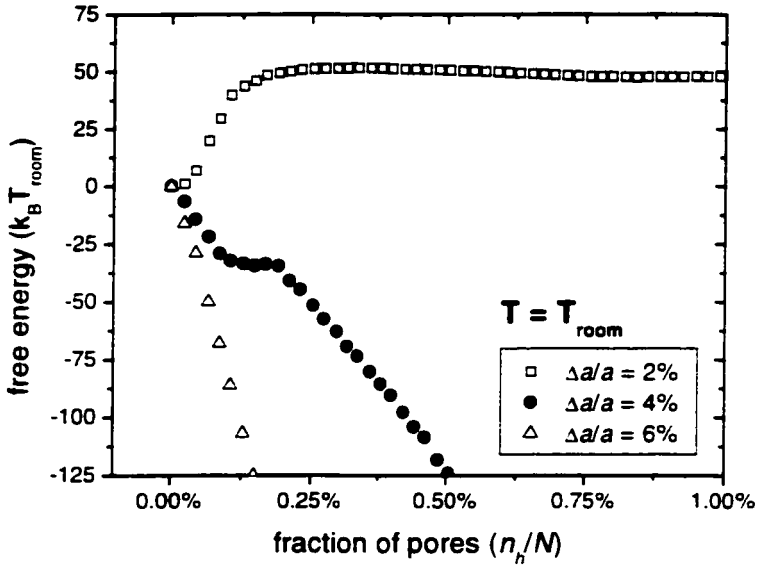


Figure 23: *Free energy plots for different expansions. At 4% and 6% strains, the system is invited to sit in its unique minimum at rupture. At 2%, the protopore regime now becomes the unique minimum. Stretches between 2% and 4% are then bistable, so that the equilibrium state depends on the initial conditions.*

The plot of the free energy F of the system as a function of the pore density cannot be obtained directly from the simulation, since the system will always stabilize around a density (n_h / N) that minimizes the free energy. One way to obtain this curve is by the technique known as umbrella sampling. It consists in adding to the initial Hamiltonian H a harmonic potential $W(n_h)$, known as a bias potential, centered at an arbitrary n_h*. We let the system evolve to get occupation probabilities P_{bias} around a given n_h*. This is repeated for a sequence of n_h* spanning the range of n_h's of interest. From the set of segments of P_{bias}, probabilities $P_0(n_h)$ for the unbiased system are obtained. These are linked together to generate the correct probability function $P(nh)$. The free energy F is

obtained directly from $F = \log(P)+C$.

The difference between the free energy curves plotted and the energy curve predicted by equation 3 both illustrated in figure 24 is the effect of entropy near $n_h = 0$. As discussed previously, without entropy, the barrier to rupture is physically prohibitive. Entropy can also shift the local minimum near zero pore size to a nonzero value, noted as the protopore regime. There is another shift near the global minimum, where the amphiphiles do not relax entirely when membrane rupture occurs: there is still a bit of tension. This effect, predicted by the theoretical model, is not entropic, but purely energetic, a contribution of the edge energy to the final equilibrium state.

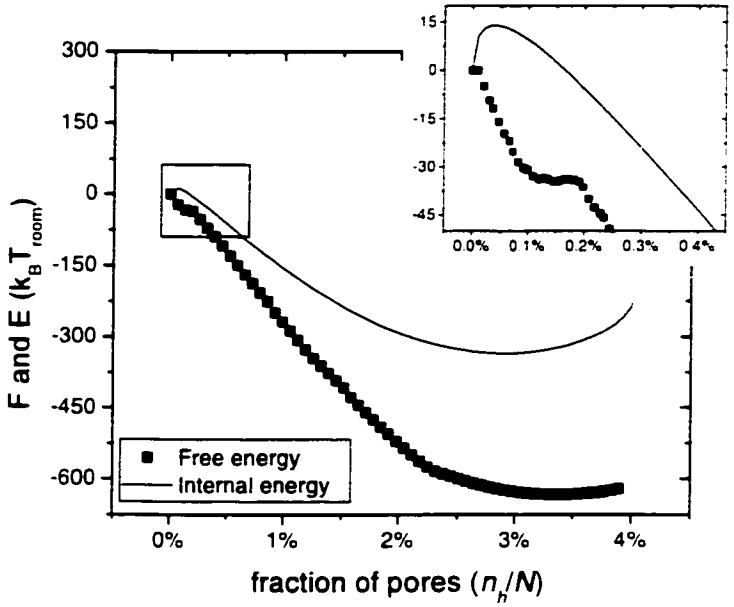


Figure 24: *Comparison of the free energy at T_{room} given by the umbrella sampling and the energy of a single hole rupture, predicted by the theoretical model. Both curves are for an applied stretch of 4%. Entropy is mainly responsible for membrane rupture.*

We can also see from the free energy curve that the critical pore size, which leads to rupture, is less than $n_h / N = 0.17\%$. For a lattice of 30301 sites, it corresponds to a total pore area of 30 nm^2. We must not forget that the total pore size in the free energy plots is likely to be distributed in many pores. Now from the energy curve representing an ideal single hole rupture, a pore has to be larger than $n_h / N = 0.04\%$ to grow stable. This area, which corresponds to a 7 nm^2 pore in our system, is smaller than 35 nm^2, as expected. The critical pore size is then between those two values. This is consistent with the pore sizes obtained in section 5.3 for stretches of 2% and 6% at room temperature. In figure 19, no pores were in this unstable range.

6 Hydrophilic edges

6.1 The healing on the edge of a pore

When a pore reaches a certain size, the lipids on the edge of a pore can form an arc shape to produce a beveled edge (figure 25). This way, water molecules are less forced to sit against a hydrophobic tail. This process is sometimes known as the healing of the edge of the pore, or as the creation of a lipid hydrophobic edge. Since the line energy of the system is affected, it seams natural that bending of lipids can affect the rupture scenario of the system. However, from geometrical considerations, the minimal pore size needed for healing is larger than the critical size needed for rupture. In other words, a pore only heals when it has grown stable. The healing on the edge of a pore has therefore no considerable effects on membrane rupture. In addition to this, the energy difference between a hydrophobic and beveled edge is known to be small [20].

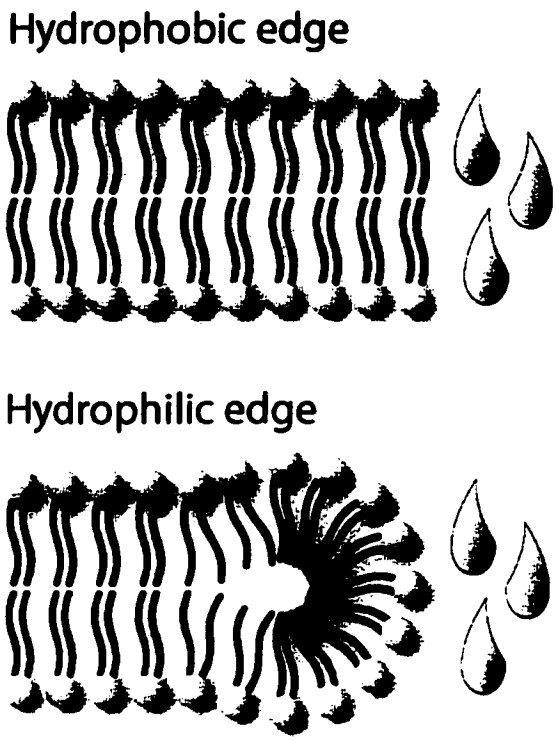


Figure 25: *Hydrophobic edge versus hydrophilic edge. When the lipids form an arc shape around an edge, there are less contact between the tails and water.*

To estimate the minimal pore size needed for healing, we consider the volume change of the tails through their deformation around a hydrophilic edge and look for the pore size which keeps the total volume of the lipids unchanged. To leading order, the membrane is not compressible. In order to be able to form an arc around a hole, the tail end of the cylindrical representation of the lipids chains must be compressed into an oval. However, the compression of the lipids in the edge plane can be counterbalanced by an expansion in the membrane plane, or the pore plane.

If the volume of the lipid tails relative to the head volume is to be smaller than the one normally allowed in a flat bilayer membrane, the exposure of the tails to water is reduced.

This case represents the healing of a large pore and is favorable. We are to believe, however, that the beveled edge will get bigger as the pore grows. Nevertheless, if the volume of the tail is augmented, more water can enter in contact with the tails. The healing of a small pore, which gives more expansion to the lateral surface of the lipid, is therefore unlikely. We estimate the smallest size of a healed pore by calculating the radius of the pore which preserves the total volume of the lipids.

From our cylindrical representation of a lipid, the volume of a "squeezed" lipid on a healed edge can be expressed as

$$V_{edge} = \int_0^h \pi \left(r_0 + \frac{\delta x}{h} z \right) \left(r_0 - \frac{\delta y}{h} z \right) dz , \qquad (19)$$

where δx is the total change in the lipid radius in the x direction, and δy the radius compression in the arc plane, as shown in figure 26. The z axis is defined to be the axis of the lipid. r_0 and h are respectively the radius and the length of the lipid chains cylindrical representation. We estimate the critical size of a pore eligible to healing as one where the volume of the acyl chains is unchanged. This sets the volume of the squeezed lipids given by equation 19 to $\pi r_0^2 h$. If we assume only small deformations of the lipid tails, it is trivial to see that $\delta x = \delta y$. Consequently, the concave curvature of the pore is, to the first order, equal to the convex curvature of the arc. In other words, the edge healing will be favorable for pores of radii larger than the radius of the arc. Unfortunately, no experimental data is yet available for the radius of a beveled edge, although we know that it must be larger than the height of a lipid. The minimal radius of a healed pore is then larger than 2nm. This corresponds to a circular pore larger than $13nm^2$, which

corresponds to a surface of about 25 lipids.

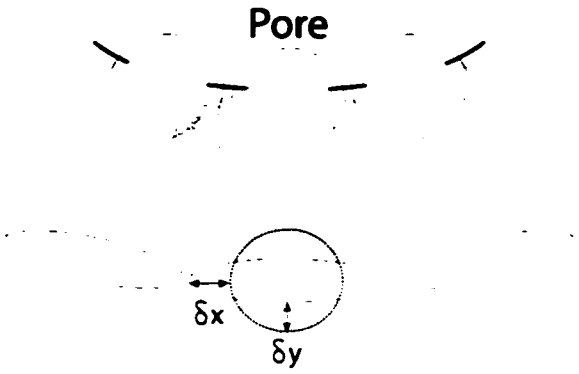


Figure 26: *Volume deformation of a lipid around a healed edge. The tails are expanded in the membrane plane, but compressed in the arc plane.*

For the typical membranes studied previously, a pore of that size is likely to have already induced rupture. Since the healing on the edge of a pore only happens when rupture has occurred, it has no major influence on the scenario of membrane rupture. For larger systems, however, pores of area smaller than the critical size may also heal. Nevertheless, we are to believe that the work done by the membrane to open such pores is to be of greater importance than the healing effects. Different models have been attempted to include the healing on the edge of a pore in the discrete Monte Carlo method, but as expected, no major differences were noticed.

6.2 Peptide insertion

Another way for the membrane to reduce its hydrophobicity along the exposed edges of a pore is through the insertion of amphiphilic peptides, as discussed in chapter 2. Moreover, peptides can stick to the edge of very small unstable holes in the membrane and stabilize them. With peptides wrapping a protopore, it becomes more stable since the line energy is lowered. Amphiphilic peptides are then to play an important role in membrane rupture, in opposition to the lipid healing on the edge of a pore.

When inserted in the membrane lattice, a peptide is going in a site previously occupied by a lipid. The missing lipid is then distributed in the membrane through the renormalization of equation 12. This change of identity of a site can be performed in so called semi-grand canonical ensembles, with a chemical potential μ associated with their insertion, or removal. When outside the membrane, the energy for an excluded peptide is set to be zero. We consider them to be non-interacting particles floating over the membrane, like a free gas. However, once in the membrane, they can interact with one another, since they are likely to get close together on the perimeter of a pore.

The internal energy of a peptide present in the membrane depends on its orientation. One of its sides is hydrophobic, while the other is hydrophilic. Modeling the evolution of a pore circled by peptides is complicated because pores keep deforming while shrinking or growing. To simplify the problem, we only consider 6 possible orientations, which correspond to the 6 nearest neighbor sites. The first-neighbor interaction energies will also be divided into 3 main cases. There are the positive and negative interactions between a lipid and a peptide, depending on the orientation of the later, and the

peptide-peptide interaction. To establish these energies, we consider the two halves of the peptide, namely the hydrophobic and hydrophilic sides. On the hexagonal lattice, 3 sites are then to face each side of the molecule. A negative (favorable) bonding α is associated between a lipid site facing the hydrophobic side of the molecule, or between a hole facing the hydrophilic side. In a parallel manner, β represents the positive interaction between a lipid site facing the hydrophilic side of the molecule, or between a hole facing the hydrophobic side. Finally, the interaction between two first-neighbor peptides is set to χ. Finally, 2 cases are made impossible: peptides with the hydrophobic side directly facing a hole, or with the hydrophilic side facing a lipid, or another peptide. Figure 27 illustrates all these possible cases.

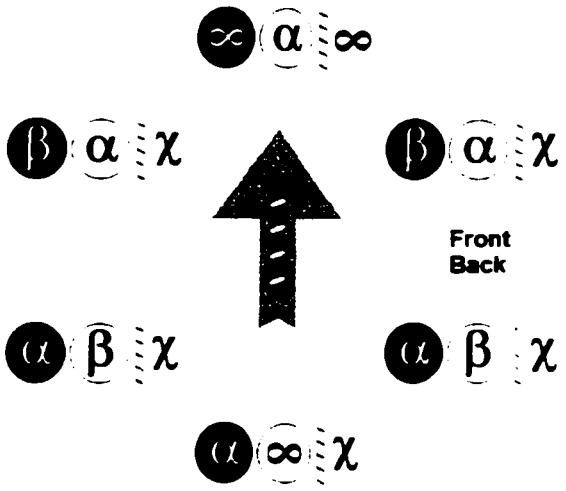


Figure 27: *Top view of the different possible interactions between an oriented lipid (arrow) and its 6 neighbours. Black circles represent lipids, white circles empty sites, and spirals peptides. α, β and χ are the possible interaction energies. The impossible cases are noted by an infinite bonding energy.*

Four parameters are then associated with the energy of a peptide, namely α, β, χ, and μ. It should be noted that all these interactions are relative to the typical interactions between a lipid and the same surrounding. From physical considerations, we will reduce the number of independent parameters to 2. First, the interaction between peptides χ will be set to have a stoichiometry larger than 1 lipid for 1 peptide along the edge, as it has been generally observed for lipids [35]. In other words, χ will be strong enough to assure that no peptides get directly in contact. For the chemical potential, it is such that a peptide along a straight edge can barely escape the membrane at room temperature. A site along a straight edge is defined by two holes and four lipids as a neighborhood. If a peptide is promptly aligned along this edge, we can write the expression

$$\Delta E_{pep} = \Delta U_{pep} - \mu \Delta N = 5\alpha + \beta - \mu = -k_B T_{room} \tag{19}$$

to estimate the chemical potential, where $k_B T_{room}$ is the thermal energy at room temperature. By choosing $\alpha = -1\ k_B T_{room}$, $\beta = 1\ k_B T_{room}$ to set the chemical potential to $\mu = -3\ k_B T_{room}$. This choice of α and β seems to give to the energy of a peptide a reasonable dependence on the orientation. If only one hole is facing the peptide with all other sites as lipids, $\Delta E_{pep} = 1\ k_B T_{room}$; with 2 holes facing the peptide, $\Delta E_{pep} = -1\ k_B T_{room}$; and 3 holes, $\Delta E_{pep} = -3\ k_B T_{room}$. And finally, for peptide interactions, $\chi = 3\ k_B T_{room}$ seems sufficient to avoid most contacts between peptides, without altering the dynamics.

This energy is not included in J^{ih} and the corrections associated with it. They are separate since normally a hole has first to be nucleated before peptides can heal it. Although the occupancy of the sites around a lipid is greater than one, only one particle

on each site can interact with a peptide. This is also why peptide insertion was separated from the correction of equation 12.

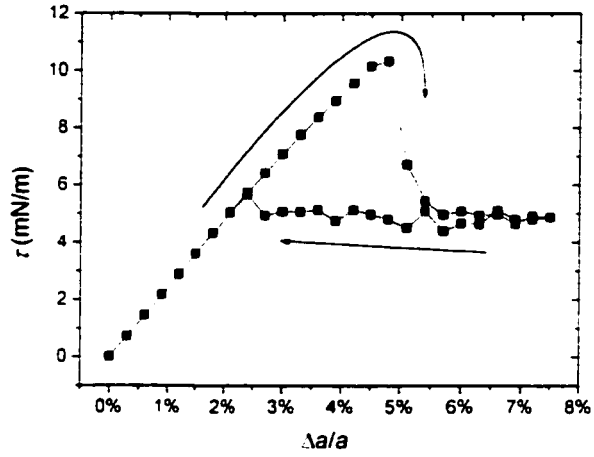
A Metropolis-type algorithm will again be used to create and annihilate orientations. However, the Rosenbluth factor will be used to generate the different trial orientations when a peptide gets inserted [42]. Instead of choosing totally randomly the orientation of the peptide at insertion time, this factor favors the probability of choosing a low energy trial configuration. However, once selected, a correction is added to the Metropolis test to make sure detailed balance is obeyed. This factor is to be used for systems where the particles have great liberty, but physically, only a few movements are favorable.

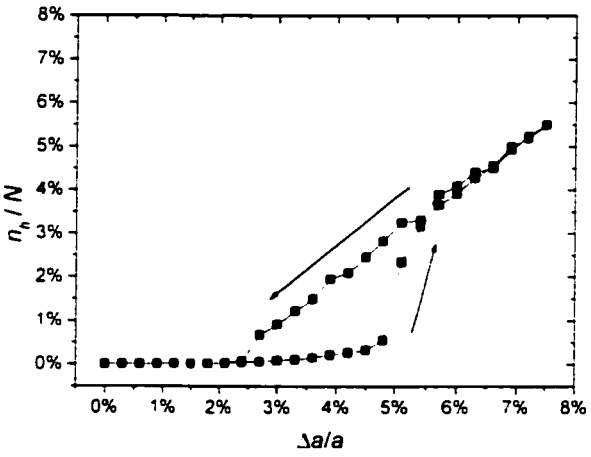
Peptides are also allowed to move when inserted in the membrane. This freedom is given to them since a pore surrounded by peptides must have the opportunity to evolve. Therefore, when the membrane has the opportunity to nucleate a site occupied by a peptide, the latter will be shifted backwards, such that the larger pore is still surrounded by the same lipid. This is physically the case. Reciprocally, if a hole facing a peptide is to be revived, the peptide will move forward, following the evolution of the pore.

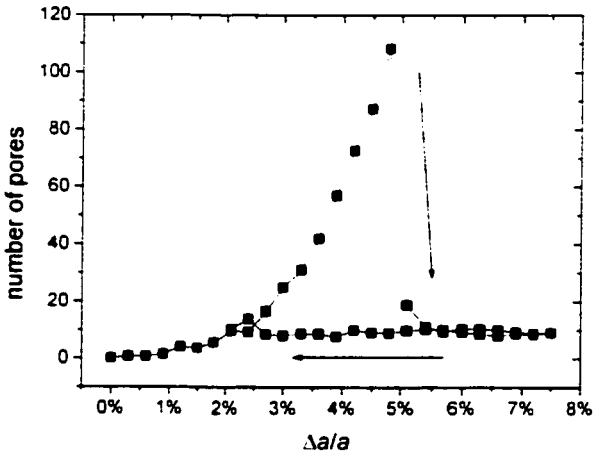
To compare the behavior of a membrane in presence of peptides, hysteresis plots have been made to mainly study the strain reaction to stress of the new system. Pore size distribution is also studied.

From the hysteresis plots illustrated in figure 28, we can see that the membrane is partially relaxed very early by the occurrence of a large number of pores. The number of pores present in the membrane is about 5 times larger than the number of pores in a

typical membrane.







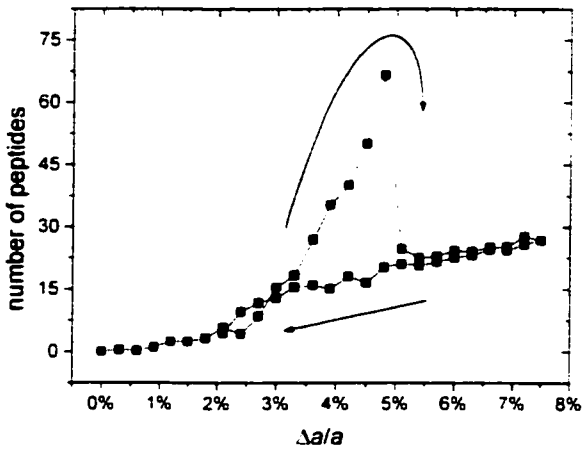


Figure 28: *Different hysteresis upon stretching for membranes in presence of amphiphilic peptides, for a membrane at T_{room}. The system is characterized by a higher concentration of pores before and after rupture. This high number of pores also inhibits the system of relaxing entirely.*

Having a larger quantity of pores does not appear in itself to be significant. However, some of these pores are larger in size (figure 29). They can therefore act as channels, as it is biologically thought [35]. With this in mind, one would think that since the formation of larger pores is favored, membrane rupture would be activated for lower tensions. Nevertheless, the larger total number of pores counterbalances this. In fact, rupture for membranes in presence of peptides occurs for expansions that are a little higher than the one of typical membranes. With peptides, membrane rupture occurs at about 5% expansions.

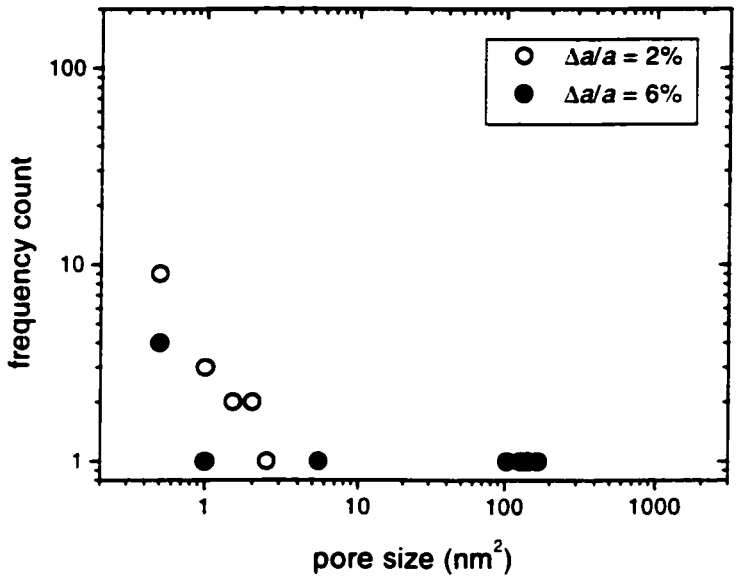


Figure 29: *Distribution of pores for a membrane at T_{room} in presence of amphiphilic peptides before and after rupture. The membrane in such environment is characterized by a larger number of pores. Those pores are larger in size before rupture (the actual chemical channels), but smaller at rupture.*

7 Conclusion

Bilayer membranes are an intrinsic part of all living species. With rapid evolution of biotechnology, membranes can now be artificially created, and thereafter even modified. However, even today, some physical properties of lipid bilayer membranes are still part of active research, as they are not well understood. The global scenario of membrane rupture is just an example of such a property.

In spite of their complex molecular structure, lipid bilayer membranes offer an ideal model system, as hydrophobicity is responsible for both the line energy and the bulk compressibility (for extension). We have developed the first model for the rupture of a membrane held together by hydrophobic forces, which includes the nucleation and growth of pores. Our minimal model explains current results on saturated and unsaturated PC lipid bilayers and thicker artificial bilayers made of diblock copolymers for the tension and area expansion at rupture. We also explain the behavior of a membrane in presence of amphiphilic peptides. The stability results of Shillcock and Boal [18] for unstressed membranes have also been reproduced. The model requires little computer time allowing the handling of real size vesicles. Entropic and nucleation effects are included naturally.

The structural integrity of a lipid bilayer is considerably affected by its composition: Biological membrane structure can be fairly complex. Different types of lipids, cholesterols, and proteins will play an important role in determining membrane stability. The model has the potential of handling more of these inclusions.

References

[1] D.S. Dimitrov and R.K. Jain, Biochim. Biophys. Acta. 779, 437 (1984).

[2] W. Harbich and W. Helfrich, Z. Naturforsch. 34A, 1063-1065 (1979).

[3] I.G. Abidor, V.B. Arakelyan, L.V.Chernomodik, Y.A. Chizmadzhev, V.F. Patushenko, and M.R. Tarasevich, Biolectrochem. Bioenerg. 6, 37-52 (1979).

[4] D. Needham, and R.M. Hochmuth, Biophys. J. 55, 1001-1009 (1989).

[5] C. Wilhelm, M. Winterhalter, U. Zimmermann, and R. Benz, Biophys. J. 64, 121-128 (1993).

[6] M. Winterhalter, Colloids and Surfaces A 149, 161-169 (1999).

[7] S.Y. Ho and G.S. Mittal, Crit. Rev. Biotech. 16, 349-362 (1996).

[8] A. Ertel, A.G. Marangoni, J. Marsh, F.R. Hallett, and J.M. Wood, Biophys. J., 64, 426-434 (1993).

[9] B.L.-S. Mui, P.R. Cullis, , E.A. Evans, T.D. Madden, Biophys. J., 64, 443-453 (1993).

[10] C. Taupin, M. Dvolaitzky, and C. Sauterey, Biochemistry 14, 4771-4775 (1975).

[11] J.S. Remy, A. Kichler, V. Mordvinov, F. Schuber, and J.P. Behr, Proc. Nat. Acad. Sci. (U.S.A.) 92, 1744 (1995).

[12] F.D. Ledley, Hum. Gene Ther. 6, 1129 (1995).

[13] N.M. Correa and Z.A. Schelly, Langmuir 14, 5802 (1998).

[14] D. Boal, *Mechanics of the Cell* (Cambridge Univ. Press, Cambridge, 2001).

[15] J.D. Litster, Phys. Lett. A 53, 193 (1975).

[16] A. Barnett and J.C. Weaver, Biolectrochem. Bioenerg. 25, 163 (1991).

[17] S.A. Freeman, M.A. Wang, and J.C. Weaver, Biophys. J. 67, 42 (1994).

[18] J.C. Shillcock and D. H. Boal, Biophys. J. 71, 317 (1996).

[19] E. Evans and F. Ludwig, J. Physics: Condens. Matter. 12, 315 (2000).

[20] M. Wortis and E. Evans, Physics in Canada 53, 281 (1997).

[21] K. Olbrich, W. Rawicz, D. Needham, and E. Evans, Biophys. J. 79, 321 (2000).

[22] W. Rawicz, K.C. Olbrich, T. McIntosh, D. Needham, and E. Evans, Biophys. J. 79, 328 (2000).

[23] J.F. Nagle, S. Tristram-Nagle, Biochim. Biophys. Acta. 1469, 159 (2000).

[24] J.N. Israelachvili, *Intermolecular and Surface Forces: With Applications to Colloidal and Biological Systems* (Academic Press, London, 1985) Chap. 16.

[25] B.D. Gaulin and J. Katsaras, Physics in Canada 53, 247 (1997).

[26] J. Wolfe, M.F. Dowgert, and P.L. Steponkus, J. Membrane Biol. 86, 127 (1985).

[27] E.A. Evans and D. Needham, J. Phys. Chem., 91, 4219 (1987).

[28] Z. Zhou and B. Joós, Phys. Rev. B. 56, 2997 (1997).

[29] J. Akinlaja and F. Sachs, Biophys. J. 75, 247 (1998).

[30] D.V. Zhelev and D. Needham, Biochim. Biophys. Acta. 1147, 89 (1993).

[31] O. Sandre, L. Moreaux, and F. Brochard-Wyart, Proc. Nat. Acad. Sci. (U.S.A.) 96, 10591 (1996).

[32] M. Paetzel and N.C.J. Strynadka, CSBMCB/SCBBMC Bulletin, 60 (2001).

[33] C. Escrive and M. Laguerre, Biochim. Biophys. Acta. 1503, 63 (2001).

[34] K. Piers, and R.E.W. Hancock., Molec. Microbiol. 12, 951 (1994).

[35] M.J. Zusckermann and T. Heimburg, Biophys. J. 81, 2458 (2001).

[36] R.E.W. Hancock, and D.S.Chapple, Antimicrob. Agents. Chemother. 43, 1317 (1999).

[37] P. Sens and A. Safran, Euro. Phys. Lett. 43 (1995).

[38] R.R. Netz and M. Schick, Phys. Rev. E 53, 3875 (1996).

[39] E. Evans and W. Rawicz, Phys. Rev. Lett. 64, 2094 (1990).

[40] H. Bermudez, A.K. Brannan, D.A. Hammer, F.S. Bates, and D.E. Discher, arXiv:cond-mat/0110088 v2 (2002).

[41] B.M. Discher, Y.-Y.Won, D.S. Ege, J.C.-M Lee, F.S. Bates, D.E. Discher, D.A. Hammer, Science 284, 1143 (1999).

[42] D. Frenkel, B. Smit, *Understanding molecular simulation: from algorithms to applications* (Academic Press, San Diego, 1996).

Appendix A, photos and films of our simulations

As an informal study of the scenario of membrane rupture, films of the membranes where produced. Those films show the position of the holes in the membrane for a given time interval. At rupture, about 4% of the 30301 particles are nucleated, that is about 1200 holes for each picture in the film. Normal viewers were unable to support that. In addition, differences had to be made between peptides and holes.

A small viewer was then created in Microsoft ® Visual Basic 6.0. From an ASCII file containing the positions of the holes at each time step, it is able to represent them on a 2D grid, representative of the hexagonal lattice. An adjustable frequency of slide change is then to decide the speed of the movie. The movie can thereafter be stopped, paused, and replayed. When paused, snapshots of the membrane can be saved in a bitmap format (BMP). Coordinates of the holes and peptides are also given.

Snapshots at different temperatures for plain membranes, and at T_{room} for membranes in presence of peptides are given here. They were taken for expansions near 2% and 6%, that is in all cases before and after rupture. The sizes of these lattices are 10981 sites, about 3 times smaller that the systems used for most of the previous results. The viewer could have supported larger files. However, with 10981 particles, for 1000 screen shots, data files are as large as 5 megabytes.

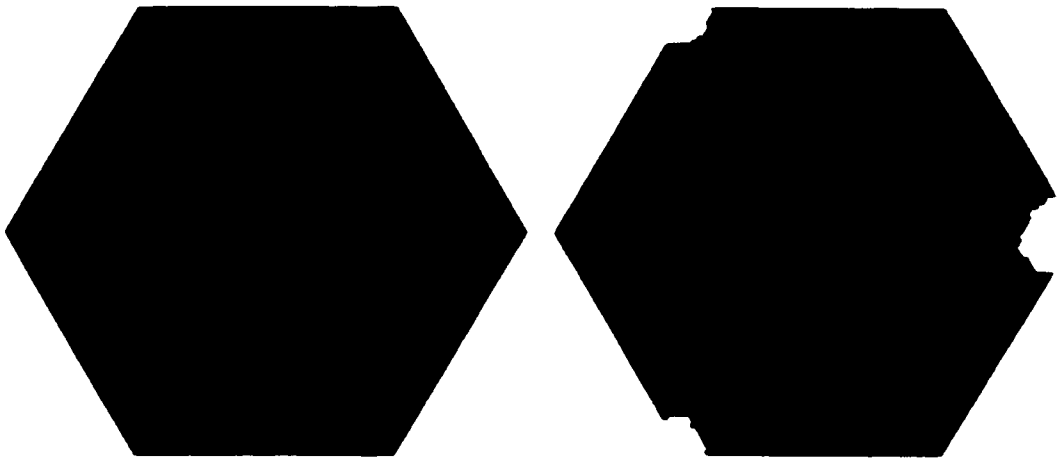


Figure A1: *Typical snapshots of membranes at ½ T_{room} before rupture (left) and after rupture (right). This trivial case is illustrated to show the shape of the hexagonal lattice, and also the periodic conditions. The network on the right contains only 1 pore.*

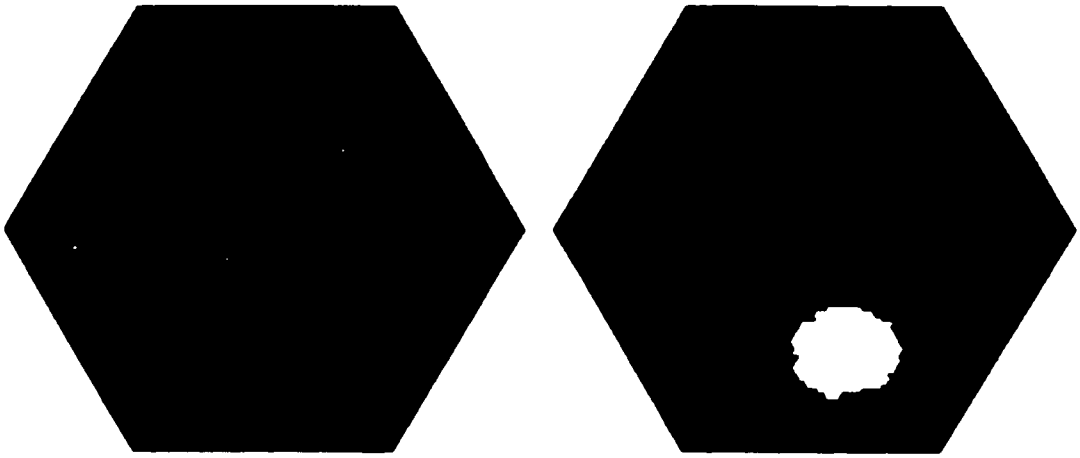


Figure A2: *Typical snapshots of membranes at T_{room} before rupture (left) and after rupture (right). Some protopores might be present before rupture, but they are more unlikely to appear when the membrane ruptured.*

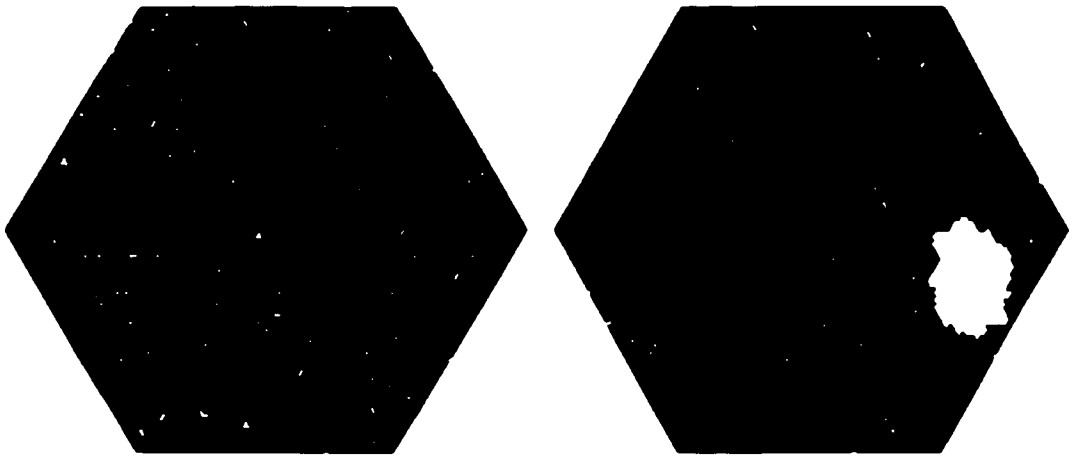


Figure A3: *Typical snapshots of membranes at $2T_{rupture}$ before rupture (left) and after rupture (right). Entropy makes the number of protopores much more significant.*

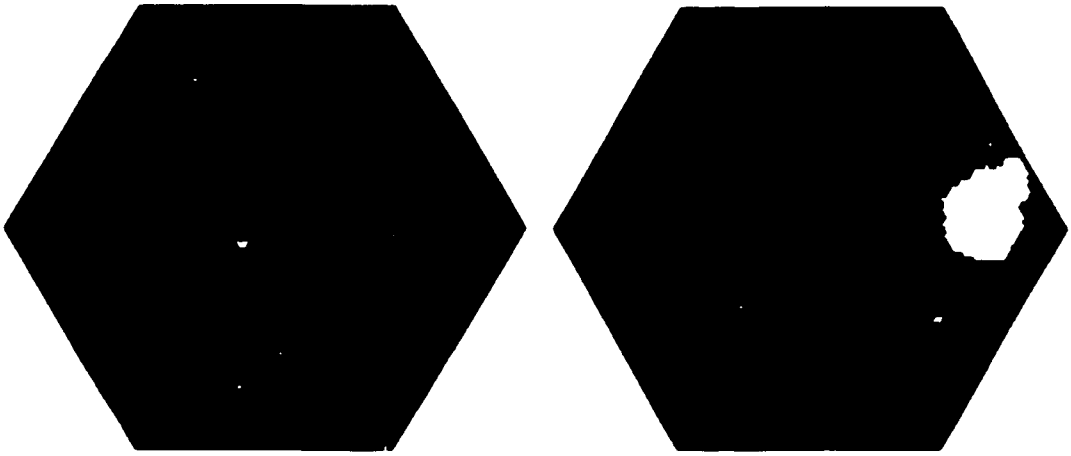


Figure A4: *Typical snapshots of membranes in an environment with peptides before (left) and after rupture (right). Channels can open at very low tensions.*

Full movie of the corresponding pictures are obtainable through ftp at

Host name: **"ftp.science.uottawa.ca"**
User name: **select or write "Anonymous"**
Password: **none required**
Pathway: **"/pub/membranes"**

From there, download the viewer called "film.exe" (by the way, the viewer is in French. Sorry for any inconvenience). Each movie with extension ".mem" can be opened from this viewer. When opening a file, expect a little wait (few seconds). Thereafter, the program will ask: "Voulez-vous transformer les données en représentation hexagonale?". Answer yes, as all the simulations are on a hexagonal lattice. From there, press "play" and enjoy.

Appendix B, the Monte Carlo program

```
C  ********** Modele d'Ising pour la rupture des membranes **********
C  **********           Algorithme de Metropolis           **********
C  **********             Structure hexagonale             **********

C  Luc Fournier
C  lfournie.science.uottawa.ca

      Program Membrane
      IMPLICIT NONE

C  ***************************************************************
C  ***************************** PREALABLES **********************
C  ***************************************************************

C  DECLARATION DES VARIABLES/FICHIERS
c    l'energie est seulement calculee lorsqu il y a une liaison
c    trous-parti.  nous calculerons donc prealablement l'energie
c    si un trou seulement se creerait et pour le reste, si un trou
c    se forme, l'energie sera donc celle ci moins la contribution
c    des trous environnants
c
c    Nous aurons la fonction n qui nous calculera
c    combien de niveau separe deux points et les fcts haut, bas qui
c    seront utiles dans les boucles pour faire une grille exagonale
c
c    Le potentiel est est du type lennard jones mais la section
c    attractive est en 1/r^4 comme le veut la litterature. A cet effet
c    les rayons stables son fixe a une distance de 3^(1/8) qui est le
c    minimum d'energie selon ce potentiel

c  La premiere ligne definit des variables communes
c  n    =      fonction retournant le no de niveau separant 2 sites
c  no   =      nombre de site totale (surfactant)
c  dV(n)=      No de site de niveau n autour dun site : membrane
parfaite
c  L(n) =      No de site de niveau n autour dun site en particulier
c  V(n) =      Contient le no total de site de tel distance ds membrane
c  Vess =      Nombre de voisin pour essayer une configuration dessaie
c  R(n) =      Rayon correspondant a tel niveau
c  P(no) =     Pour les probabilites, garde le compte, PROB BIAISE
c  W =         Pour le umbrella sampling, facteur multipliant P biaise
c  avgW =      Moyenne de W (qui est en fait 1/W)
c  VP   =      VRAI probabilite
c  bias  =     Calcule la fonction qui biaise F, lenergie libre pour US
c  event =     Nombre devenement realise
c  gamma =     Facteur multipliant les R initialement stables
c  lambda =    no trou REEL, Ilambda
c  Rstable =   Le rayon le plus stable pour le potentiel donne
c  notrou =    no de trou present, noess pour un essai, postrou =
compteur
c  noniveau = nombre de niveau quon considere dans la simulation
c  haut/bas = limites inf et sup des compteurs pour une membrane exa
```

```
c  occupation est la grille de travail et trou la pos des trous dedant
c  r1, r2, ran??? sont pour les nombres aleatoires

      INCLUDE 'parametres.h'
      INCLUDE 'variables.h'

      INTEGER t, finloop, seed, tag, diagrame, loop
      INTEGER Npore,Ntrou,Npep,Ndist(no/20),a(no/20) ! pour Igamma
(DIST)
      INTEGER P(0:5000), taille                      ! pour lambda (US)
      INTEGER Gauche, Droite, nophoto

      REAL avgpore, avgtrou, avgpep, avgdist(no/20)  ! pour gamma
      REAL VP, avgW, W, E, F, S, couture
      COMMON taille
      INCLUDE 'initialisation.h'

C  INITIALISATION
c  Initialisation du random generator RANDOM LUXURY
      seed = 9457273
      CALL RLUXGO(3,seed,0,0)

c  Appeller la sousroutine initialisatrice pour
c  voir si on debute ou si on continue
      Call IO()

C  ***********************************************************************
C  ********************** SECTION PRINCIPALE **********************
C  ***********************************************************************

C  Cas special pour faire des films de la membrane

      IF (usage.EQ.100) then
        W = gamma
        nophoto = 20

        DO loop = 1, 100
          gamma = W * float(loop) /100.0

          DO t = 1, total

          CALL metropolis()
          CALL metrop()

C         Faire un film de la membrane, nophoto photos a chaque gamma

          If ((mod(t, total/nophoto).EQ.0).AND.(t.NE.0)) THEN
            OPEN(77, FILE='resultats/'//filename//'.mem',
ACCESS='APPEND')
            WRITE(77,*) notrou+nopep, ',' ,gamma
              DO i = 1, notrou
              WRITE(77,*) trou(i, 1), ',' ,trou(i, 2), ', 1'
              END DO
              DO i = 1, nopep
              WRITE(77,*) peptide(i, 1), ',' ,peptide(i, 2), ', 2'
              END DO
```

```
            CLOSE(77)
          END IF

          END DO
        END DO
      END IF

C  AUCUNE ITERATION SPECIALE (DISTRIBUTION DES PORES)
C  ******************************************************************

      IF (usage.EQ.0) THEN

      OPEN(8, FILE='resultats/'//filename//'.data', ACCESS='APPEND')
      DO i = 1, no/20
        avgdist(i) = 0
      END DO

      INCLUDE 'initialisation.h'
C       INITIALISATION DES PROBABILITES
        DO i = 1, no/20
          Ndist(i) = 0
        END DO
        Npore = 0
        Ntrou = 0
        taille = 0

C       ITERATION DANS LE TEMPS, TOTAL ITERATIONS

c       On elimine en masse de transiants!
        W = gamma
        DO i = 1, 5
          gamma = W * float(i) / 5.0

          DO t = 1, total/10
            CALL metropolis()
            CALL metrop()
          END DO

        END DO
        gamma = W

        DO t = 1, total

          CALL metropolis()
          CALL metrop()

C         Etablir 1000 echantillons avec le dernier tier
          IF ((mod(t,total/2000).eq.0).AND.(t.GT.total/2)) THEN
            Ntrou = Ntrou + notrou
            Npore = Npore + nopore(a)
            DO i = 1, no/20
              Ndist(i) = Ndist(i) + a(i)
            END DO
            taille = taille + 1
          END IF
```

```
          END DO

          avgtrou = float(Ntrou)/float(taille)/float(no)
          avgpore = float(Npore)/float(taille)

          DO i = 1, no/20
            avgdist(i) = avgdist(i) +
     *                 float(Ndist(i))/taille/avgpore
          END DO

          DO i = 1, no/20
            write(8,*) i, avgdist(i), a(i)
          END DO
          CLOSE(8)

        END IF

C   ITERATION SUR LAMBDA (UMBRELLA SAMPLING)
C   ****************************************************************

C   LA VARIABLE LAMBDA EST ICI DIMENSIONNALISE!
C   ON Y VA PAR BONDS DE DIX TOUJOURS, DONC LE CENTRE EST LAMBDA+5

C       ON COMMENCE PAR ITERER SUR LES DIFFERENTS LAMBDA
C       Les variables sont reinitialises car lambda change

        IF (usage.EQ.1) THEN

        couture = 0.0
        DO 10 Ilambda = debut, fin, 10

        OPEN(8, FILE='resultats/'//filename//'.data', ACCESS='APPEND')
          lambda = Ilambda + 5
          phase = 0
          taille = -1
          CALL prob(P)

C         ITERATION DANS LE TEMPS, TOTAL' ITERATIONS


C         ELIMINATION DES TRANSIANTS (25%)

          i = 0
          usage = 2
          DO WHILE ((i.LT.total/2).AND.(float(notrou).LT.lambda))
            i = i + 1
            CALL metropolis()
            CALL metrop()
          END DO
          usage = 1

C         CALCUL DU BIAISAGE (25%)

          DO t = 1, total/2
          call metropolis()
          CALL metrop()
```

```
          CALL prob(P)
          END DO

          gauche = 0
          droite = 0
          DO i = 1, 20
            IF (lambda-i.GT.0) THEN
              gauche = gauche + P(lambda - i)
              droite = droite + P(lambda + i)
            END IF
          END DO
          phase = phase + 3*(gauche - droite)*2 / taille
          taille = -1
          Call prob(P)

C         CALCUL FINAL AVEC BIAISAGE (50%)

          DO t = 1, total
          call metropolis()
          CALL metrop()
          CALL prob(P)
          END DO

C         ON A MAINTENANT UNE BONNE DISTRIBUTION STATISTIQUE ENTRE
C         (LAMBDA-5) ET (LAMBDA+5) ON FAIT DONC LE CALCUL DE F
C         ENTRES CES VALEURS POUR UMBRELLA SAMPLING

c         On commence par calculer la moyenne de 1/W
c         On suppose que la membrane ne sera jamais plus de 50% vide

          AvgW = 0.0
          gauche = lambda - 10
          droite = lambda + 10
          DO i = gauche, droite
            IF (i.GE.0) THEN
              avgW = avgW + P(i)*(exp(Bias(i)/kT)/float(taille))
            END IF
          END DO

c         Calcul de F autrour de la region dinteret et on sauvegarde
c         ces valeurs pour une intervale de 10

          DO 14 i = lambda-5, lambda+5
            W = exp(Bias(i)/kT)
            VP = float(P(i))/float(taille) * W / avgW
            F = -kt*log(VP)
            IF ((abs(F).LT.100000.0).AND.(i.EQ.lambda-5)) couture=F-
couture
            write(8,*) i, float(i)/float(no), F  - couture
14        CONTINUE
          IF (abs(F).LT.100000.0) couture = F - couture

        CLOSE(8)
10      CONTINUE
        END IF
```

```
C  ITERATION SUR GAMMA (HYSTERESE DES TROUS/PORES)
C  ****************************************************************

       IF (usage.EQ.2) THEN

       DO Igamma = debut, 2*fin-debut
       OPEN(8, FILE='resultats/'//filename//'.data', ACCESS='APPEND')

         If (Igamma.LE.fin) gamma=step*float(igamma)
         If (Igamma.GT.fin) gamma=step*float(2*fin-igamma)

C        INITIALISATION DES PROBABILITES
         Npore = 0
         Ntrou = 0
         Npep = 0
         taille = 0

C        ITERATION DANS LE TEMPS, TOTAL ITERATIONS

         DO t = 1, total

           CALL metropolis()
           CALL metrop()

C          Etablir 5000 echantillons avec la derniere moitie
           IF ((mod(t,total/10000).eq.0).AND.(t.GT.total/2)) THEN
             Ntrou = Ntrou + notrou
             Npore = Npore + nopore(a)
             Npep = Npep + nopep
             taille = taille + 1
           END IF

         END DO

         avgtrou = float(Ntrou)/float(taille)/float(no)
         avgpep = float(Npep)/float(taille)/float(no)
         avgpore = float(Npore)/float(taille)

         write(8,*) gamma, avgtrou, avgpep, avgpore

       CLOSE(8)

       END DO
       END IF


C  ITERATION SUR ALPHA (CHANGEMENT DE LENERGIE DE LIGNE)
C  ****************************************************************

       IF (usage.EQ.3) THEN

       DO Ialpha = debut, fin

         INCLUDE 'initialisation.h'
```

```
         alpha = step*float(Ialpha)
c       OPEN(8, FILE='resultats/'//filename//'.data', ACCESS='APPEND')
         write(*,*) alpha
         avgtrou = 0.0
         avgpore = 0.0
         gamma = 0.005

C        ITERATION DES GAMMA DANS LE TEMPS

         DO WHILE ((avgtrou.LE.(0.75*gamma))
     * .AND.(gamma.LT.0.05))

C          INITIALISATION DES PROBABILITES

           gamma = gamma + 0.005
           avgtrou = 0.0
           Ntrou = 0
           taille = 0

           DO t = 1, total

             CALL metropolis()
             CALL metrop()

C            Etablir 100 echantillons avec la derniere moitie
             IF ((mod(t,total/200).eq.0).AND.(t.GT.total/2)) THEN
               Ntrou = Ntrou + notrou
               taille = taille + 1
             END IF

           END DO

           avgtrou = float(Ntrou)/float(taille)/float(no)
           write(*,*) gamma, avgtrou

         END DO

c       CLOSE(8)

       END DO
       END IF


C  ITERATION SUR kT (dimension fractale)
C  ***********************************************************************

       IF (usage.EQ.4) THEN

c      DO 22 IkT = debut, fin
       OPEN(8, FILE='resultats/'//filename//'.data', ACCESS='APPEND')
       INCLUDE 'initialisation.h'

c        kT = step*float(IkT)
         kT = 2.2
         DO h = -10 , 10
           DO i = bas(h,10-max),haut(h,10-max)
```

```
              occupation(h,i) = 0
              notrou = notrou+1
              trou(notrou,1) = h
              trou(notrou,2) = i
            END DO
          END DO

          DO t = 1, total
            CALL metropolis()
            CALL metrop()
c           CALL metrospin()
          END DO

c         CALL impression(notrou)

          DO g = 1, 40
            gauche = (2*g+1)**2-(g*(g+1))
            droite = 0

            DO h = -g , g
              DO i = bas(h,g-max),haut(h,g-max)
                if (occupation(h,i).NE.1) droite = droite + 1
              END DO
            END DO
            write(8,*) kT, gauche, droite

          END DO

        CLOSE(8)
c22     CONTINUE
        END IF


C   ITERATION SUR kT-GAMMA (diagrame de phase)
C   ******************************************************************

        IF (usage.EQ.5) THEN

        DO IkT = debut, fin
        INCLUDE 'initialisation.h'

          kT = step*float(IkT)

          DO Igamma = debut2, fin2

            gamma = step2*float(Igamma)
            OPEN(8, FILE='resultats/'//filename//'.data',
     *           ACCESS='APPEND')

            IF ((avgtrou.GT.(0.7*gamma)).AND.(gamma.GT.(0.02))
     *         .AND.(Igamma.GT.debut2)) THEN

            WRITE(8,*) kT, gamma, avgtrou, avgpore
            CLOSE(8)

            ELSE
```

```
C          INITIALISATION DES PROBABILITES
           Ntrou = 0
           Npore = 0
           taille = 0

C          ITERATION DANS LE TEMPS, TOTAL' ITERATIONS
C          Les premiers 20% serviront de transiants

           DO t = 1, total

             CALL metropolis()
             CALL metrop()

C            Etablir probs avec 5000 echantillons
             IF ((mod(t,total/10000).EQ.0).AND.(t.GT.total/2)) THEN
               Ntrou = Ntrou + notrou
               Npore = Npore + nopore(a)
               taille = taille + 1
             END IF

           END DO

           avgtrou = float(Ntrou)/float(taille)/float(no)
           avgpore = float(Npore)/float(taille)

           WRITE(8,*) kT, gamma, avgtrou, avgpore
           CLOSE(8)
           END IF
         END DO
       END DO
       END IF

       CLOSE(7)
       STOP 'Voila'
       END


C  ***********************************************************************
C  *************** VARIABLES GLOBALES ET PARAMETRES ***************
C  ***********************************************************************


C  CONTIENT TOUS LES PARAMETRES UTILES AUX
C  AU PROGRAMME PLUS QUELQUES CONSTANTES
C  Ces valeurs ne changent pas dans lexecution
C  CONSTANTES UNIVERSELLES
      INTEGER aucun, plein, proteine
      PARAMETER (aucun = 0)
      PARAMETER (plein = 1)
C  CONSTANTES SPECIFIQUES
C  nmax doit etre max+noniveau
      INTEGER max, no, total, noniveau, nopro
      REAL Eavant, Earriere, Epeptide
      PARAMETER (max = 10)          !rayon de la membrane
      PARAMETER (no = (2*max+1)**2-(max*(max+1))) !no sites
```

```
      PARAMETER (total = 1000000)  !iterations pour stabilisation
      PARAMETER (noniveau = 2)     !niveaux que l'on considere
      PARAMETER (Eavant = -0.05)
      PARAMETER (Earriere = 0.05)
      PARAMETER (Epeptide = 0.1)
C  FONCTIONS DU PROGRAMME
      INTEGER nopore, n, bas, haut, dirX, dirY
      INTEGER envirolegal, spinlegal
       REAL bias, J, spinJ, enviroJ
C la fin


C  VARIABLES GLOBALES CARACTERISTIQUES AU PROGRAMME
C  Ce sont ces variables qui seront sauvegardees

C  VARIABLES SAUVEGARDE (COMMUNES)
      INTEGER occupation
     *  (-max-noniveau:max+noniveau, -max-noniveau:max+noniveau)
      INTEGER orientation
     *  (-max-noniveau:max+noniveau, -max-noniveau:max+noniveau)
      INTEGER trou(2*max**2,2), peptide(max**4,2)
      INTEGER L1, notrou, nopep
      INTEGER Igamma, Ilambda, IkT, Ialpha, Imu
      INTEGER debut, fin, debut2, fin2
      REAL step, step2

      COMMON occupation, orientation, trou, peptide
      COMMON L1, notrou, nopep, step, step2
      COMMON Ilambda, Igamma, IkT, Ialpha, Imu
      COMMON debut, fin, debut2, fin2

C  VARIABLES UTILES UN PEU PARTOUT (COMMUNES MAIS PAS SAUVEGARDE)
C  phase y est car utile dans main et bias
      INTEGER lambda, usage, phase
      REAL gamma, kT, alpha, mu
      CHARACTER*8 filename

      COMMON lambda, gamma, alpha, usage, phase, kT, mu
      COMMON filename

C  VARIABLES DE LOOP (PAS COMMUNES)
      INTEGER g, h, i
      REAL ran

C la fin


C  ***************************************************************
C  ************************** SOUROUTINES ************************
C  ***************************************************************

C     FONCTION CALCULANT LA FONCTION BIAISEE
C     POUR UMBRELLA SAMPLING
      FUNCTION bias(loc)
      IMPLICIT NONE
      INCLUDE 'parametres.h'
```

```
      INCLUDE 'variables.h'
      INTEGER loc
        bias = 0.06*float(loc - lambda - phase)**2 - 12.0

      RETURN
      END


C  SOUROUTINE CHANGEANT LES SPINS
c  Cette routine genere donc une configuration de spin

      SUBROUTINE changespin(x0, y0, dir)
      IMPLICIT NONE
      INCLUDE 'parametres.h'
      INCLUDE 'variables.h'

      INTEGER dir, x0, y0, x, y

C     ON CHANGE LORIENTATION DU SITE AINSI QUE 3 NIVEAUX
C     TOUT AUTOUR DE LA GRILLE POUR COND PERIODIQUES

      orientation(x0,y0) = dir

      x = x0 + max
      y = y0 + max + 1
      IF ((y.LE.haut(x,noniveau)).AND.(y.GE.bas(x,noniveau))
     *   .AND.(ABS(x).LE.(max+noniveau))) orientation(x,y)=dir

      x = x0 - max
      y = y0 - max - 1
      IF ((y.LE.haut(x,noniveau)).AND.(y.GE.bas(x,noniveau))
     *   .AND.(ABS(x).LE.(max+noniveau))) orientation(x,y)=dir

      x = x0 + 2*max + 1
      y = y0 - max
      IF ((y.LE.haut(x,noniveau)).AND.(y.GE.bas(x,noniveau))
     *   .AND.(ABS(x).LE.(max+noniveau))) orientation(x,y)=dir

      x = x0 - 2*max - 1
      y = y0 + max
      IF ((y.LE.haut(x,noniveau)).AND.(y.GE.bas(x,noniveau))
     *   .AND.(ABS(x).LE.(max+noniveau))) orientation(x,y)=dir

      x = x0 + max + 1
      y = y0 - 2*max - 1
      IF ((y.LE.haut(x,noniveau)).AND.(y.GE.bas(x,noniveau))
     *   .AND.(ABS(x).LE.(max+noniveau))) orientation(x,y)=dir

      x = x0 - max - 1
      y = y0 + 2*max + 1
      IF ((y.LE.haut(x,noniveau)).AND.(y.GE.bas(x,noniveau))
     *   .AND.(ABS(x).LE.(max+noniveau))) orientation(x,y)=dir

        RETURN
        END
```

```
C  FONCTION RETOURNANT LA DISTANCE LA DISTANCE A PARCOURIR
C  EN X (EN Y)  POUR UNE ORIENTATION DONNEE

      FUNCTION dirX(direction)
      IMPLICIT NONE
      INTEGER dirX, direction

        IF ((direction.EQ.2).OR.(direction.EQ.7)) THEN
          dirX = 1
        ELSEIF ((direction.EQ.4).OR.(direction.EQ.5)) THEN
          dirX = -1
        ELSE
          dirX = 0
        END IF

      RETURN
      END


      FUNCTION dirY(direction)
      IMPLICIT NONE
      INTEGER dirY, direction

        IF ((direction.EQ.3).OR.(direction.EQ.4)) THEN
          dirY = 1
        ELSEIF ((direction.EQ.6).OR.(direction.EQ.7)) THEN
          dirY = -1
        ELSE
          dirY = 0
        END IF

      RETURN
      END


C     SOUSROUTINE POUR TRAITER UNE PILE
      SUBROUTINE ENQUEUE(pile, grille, finish, visite, x, y)
      IMPLICIT NONE
      INCLUDE 'parametres.h'
      INCLUDE 'variables.h'
      INTEGER x, y, finish, visite
      INTEGER grille(-max:max,-max:max)
      INTEGER pile(max**2,2)
        IF ((occupation(x,y).EQ.0).AND.(grille(x,y).EQ.0)) THEN
        IF (abs(x).LE.max) THEN
        IF ((y.LE.haut(x,0)).AND.(y.GE.bas(x,0))) THEN
          finish = finish + 1
          pile(finish,1) = x
          pile(finish,2) = y
          grille(x,y) = 1
          visite = visite + 1
        END IF
        END IF
        END IF
      RETURN
```

```
      END


C  FONCTION RETOURNANT LA VALEUR DE LENERGIE ENVIRONNANT UN SPIN
c  retourne la diference de E entre le un site plein (oriente ou
c  non) et un site vide.  Utile pour Metropolis.

       FUNCTION enviroJ(x0, y0)
       IMPLICIT NONE
       INCLUDE 'parametres.h'
       INCLUDE 'variables.h'

       INTEGER x0, y0, v2, v3, v4, v5, v6, v7
       INTEGER lav, lar, voisin
       REAL enviro

       v2 = orientation(x0+1,y0)
       v3 = orientation(x0,y0+1)
       v4 = orientation(x0-1,y0+1)
       v5 = orientation(x0-1,y0)
       v6 = orientation(x0,y0-1)
       v7 = orientation(x0+1,y0-1)

       lav = 0
       lar = 0

c      ON SUPPOSE AUSSI QUE ENVIROLEGAL A DEJA VERIFIER SI POSSIBLE
c      ON REGARDE LA DIFFERENCE ENTRE PLEIN ET VIDE
c      SANS LES SPINS INCLUS (TROP MELANGEANT AVEC LES SWAPS)
c      ON COMPTE LENERGIE GAGNE SI MAINTENANT UN TROU

c      PREMIERE LIGNE DU IF REGARDE LES FLECHES FUYANTES
c      ET ELSIF PREND LE RESTE DES ORIENTATIONS

       enviro = 0.0

       IF ((v2.EQ.7).OR.(v2.EQ.3)) THEN
         lar = lar + 1
       ELSEIF (v2.NE.aucun) THEN
         lav = lav + 1
       END IF

       IF ((v3.EQ.2).OR.(v3.EQ.4)) THEN
         lar = lar + 1
       ELSEIF (v3.NE.aucun) THEN
         lav = lav + 1
       END IF

       IF ((v4.EQ.3).OR.(v4.EQ.5)) THEN
         lar = lar + 1
       ELSEIF (v4.NE.aucun) THEN
         lav = lav + 1
       END IF

       IF ((v5.EQ.4).OR.(v5.EQ.6)) THEN
         lar = lar + 1
```

```
        ELSEIF (v5.NE.aucun) THEN
          lav = lav + 1
        END IF

        IF ((v6.EQ.5).OR.(v6.EQ.7)) THEN
          lar = lar + 1
        ELSEIF (v6.NE.aucun) THEN
          lav = lav + 1
        END IF

        IF ((v7.EQ.6).OR.(v7.EQ.2)) THEN
          lar = lar + 1
        ELSEIF (v7.NE.aucun) THEN
          lav = lav + 1
        END IF

        enviroJ = Eavant*float(lav) + Earriere*float(lar)

        RETURN
        END


C  FONCTION DISANT SI OUI(1), NON(0), OU SI UN SWAP(2-7) DOIT SE FAIRE
c  Cette fonction est utile seulement dans metropolis dans le cas
c  de linversion de polarite.

        FUNCTION envirolegal(x0,y0,xS,yS)
        IMPLICIT NONE
        INCLUDE 'parametres.h'
        INCLUDE 'variables.h'

        INTEGER x0, y0, xS, yS, x, y, d, dg, dd, fleche, swap

c       ON SUPPOSE QUE CE NEST PAS POSSIBLE
        swap = 0

c       ON REGARDE SI ON A A FAIRE AVEC UNE PARTICULE QUI DISPARAIT
c       OU APPARAIT.

c       SI OCCUPATION = VIDE, TROU EN DEVENIR!!!
        IF (occupation(x0,y0).EQ.aucun) THEN

c       ON DOIT REGARDER SI IL NY A PAS DE FLECHES FUYANTES
        IF ((orientation(x0+1,y0).NE.2).AND.
     *(orientation(x0,y0+1).NE.3).AND.
     *(orientation(x0-1,y0+1).NE.4).AND.
     *(orientation(x0-1,y0).NE.5).AND.
     *(orientation(x0,y0-1).NE.6).AND.
     *(orientation(x0+1,y0-1).NE.7)) THEN

c       ON REGARDE SI PROTEINE EST PRESENTE ET SI ELLE PEUT BOUGER
c       SI PROTEINE, ELLE DOIT BOUGER ET NE DOIT PAS EN ECRASER UNE AUTRE

        d = orientation(x0,y0)
        x = x0-dirX(d)
        y = y0-dirY(d)
```

```
      IF (d.NE.aucun) THEN

        IF ((spinlegal(x,y,d).EQ.1).AND.
     *     (orientation(x,y).EQ.AUCUN)) THEN
          swap = d + 3
          IF (swap.GT.7) swap = swap - 6
        END IF

      ELSE
        swap = 1
      END IF

      END IF

c     AUTREMENT CEST UN TROU QUI REVIENT A LA VIE
      ELSE

c     ON LES VERIFIE INDIVIDUELLEMENT CAS SWAP PEUT SAUVER LA SITUATION
c     ON NE FAIT PAS NON PLUS UNE SUITE REGULIERE POUR NE PAS AVOIR
c     DE CORRELATION (chaque cas individuel car il sagit dune
recherche)

        fleche = 0

        IF (orientation(x0-1,y0).EQ.2) THEN
          IF (spinlegal(X0,Y0,2).EQ.1) swap = 5
          fleche = fleche + 1
        END IF

        IF (orientation(x0+1,y0-1).EQ.4) THEN
          IF (spinlegal(X0,Y0,4).EQ.1) swap = 7
          fleche = fleche + 1
        END IF

        IF (orientation(x0,y0+1).EQ.6) THEN
          IF (spinlegal(X0,Y0,6).EQ.1) swap = 3
          fleche = fleche + 1
        END IF

        IF (orientation(x0+1,y0).EQ.5) THEN
          IF (spinlegal(X0,Y0,5).EQ.1) swap = 2
          fleche = fleche + 1
        END IF

        IF (orientation(x0-1,y0+1).EQ.7) THEN
          IF (spinlegal(X0,Y0,7).EQ.1) swap = 4
          fleche = fleche + 1
        END IF

        IF (orientation(x0,y0-1).EQ.3) THEN
          IF (spinlegal(X0,Y0,3).EQ.1) swap = 6
          fleche = fleche + 1
        END IF

c       VERIFIE SI TROP DE FLECHE POINTE SUR LE SITE
```

```
          IF (fleche.GT.1) swap = 0
          IF (fleche.EQ.0) swap = 1

        END IF

c       CALUL DE LEMPLACEMENT DU SWAP ET
c       VEFIFIE SI ON EST PAS HORS DE LA GRILLE

        xS = x0 + dirX(swap)
        yS = y0 + dirY(swap)

        IF ((yS.GT.haut(xS,0)).OR.(yS.LT.bas(xS,0)).OR.
       *(ABS(xS).GT.max)) swap = 0

        envirolegal = swap

        RETURN
        END


C   FONCTIONS CALCULANTS LA LIMITE DE LA GRILLE EXAGONALE
c   Fonctions uniquement concu pour les boucles
c   Fonctions qui transforment rien, inclus seulement parametres

        FUNCTION bas(x, bord)
        IMPLICIT NONE
        INCLUDE 'parametres.h'
          Integer x, bord
          IF (x.le.0) THEN
            bas = -(max+bord) - x
          ELSE
            bas = -(max+bord)
          ENDIF
        RETURN
        END

          FUNCTION haut(x, bord)
        IMPLICIT NONE
        INCLUDE 'parametres.h'
          Integer x, bord
          IF (x.le.0) THEN
            haut = max+bord
          ELSE
            haut = max+bord - x
          ENDIF
        RETURN
        END



C       SOUSROUTINE IMPRIMANT LA MATRICE D OCCUPATION
        SUBROUTINE impression(marque)
        IMPLICIT NONE
        INCLUDE 'parametres.h'
        INCLUDE 'variables.h'
          INTEGER x, y, t, lieu
```

```
      CHARACTER*50 ligne
      Integer marque
c     OPEN(7, FILE='resultats/'//filename//'.grid', ACCESS='APPEND')

      write(*,*) 'MARQUE', marque
      DO 120 y = 11,-11, -1
        lieu = abs(y)
        ligne = ' '
        DO 130 x=bas(y,1),haut(y,1)

        IF (occupation(x,y).EQ.0) THEN
           ligne=ligne(1:lieu) // '  '

           IF (orientation(x,y).NE.0)
     *          write(*,*) 'ERREUR, spin sans particule a',x,',',y

        ELSEIF (occupation(x,y).EQ.1) THEN
          IF (orientation(x,y).EQ.0)
     *          ligne = ligne(1:lieu) // ' #'

          IF ((orientation(x,y).EQ.2).OR.(orientation(x,y).EQ.5))
     *          ligne = ligne(1:lieu) // ' -'

          IF ((orientation(x,y).EQ.3).OR.(orientation(x,y).EQ.6))
     *          ligne = ligne(1:lieu) // ' /'

          IF ((orientation(x,y).EQ.4).OR.(orientation(x,y).EQ.7))
     *          ligne = ligne(1:lieu) // ' \\'

        ELSE
            write(*,*) 'ERREUR, mauvaise occupation a', x, ',', y,
     *       '.  Retourne la valeur',  occupation(x,y)

        END IF
              lieu = lieu + 2

130     CONTINUE
      write(*,*) ligne

120   CONTINUE
      write(*,*) ' '

c     CLOSE(7)
      RETURN
      END


C  SOUS ROUTINE SERVANT AUX I/O DU PROGRAMME
C  ORGANISER POUR POUVOIR STOPER ET CONTINUER

      SUBROUTINE IO(option)
      IMPLICIT NONE
      INCLUDE 'parametres.h'
      INCLUDE 'variables.h'
      INTEGER suite, option, x, y
```

```
      OPEN(10, FILE='valeurs.initiales')
      Read(10,*)
      Read(10,*)
      Read(10,*)
      Read(10,*)
      Read(10,*)
      Read(10,*)
      Read(10,*) usage
      Read(10,*)
      Read(10,*)
      Read(10,*)
      Read(10,*)
      Read(10,*)
      Read(10,*) debut, fin
      Read(10,*) debut2, fin2
      Read(10,*)
      Read(10,*) step
      Read(10,*) step2
      Read(10,*)
      Read(10,*) gamma, alpha, kT
      Read(10,*)
      Read(10,*) filename

      RETURN
      END



C     FONCTION CALCULANT L ENERGIE D INTERACTION

      Function J(dnh)
      IMPLICIT NONE
      INCLUDE 'parametres.h'
      INCLUDE 'variables.h'
      Integer Voisin(noniveau), dnh
      Real np, nh, nhSD
      Real epsilon, dE, lgn, dgamma, esp, shc

C     LA LINEARISATION SERA PRISE AU MILIEU DE LINTERVALE
      np = float(no)
      nh = notrou + float(dnh)/2.0
      nhSD = nh/np

C     POUR RENORMALISATION DE LA DENSITE DUN SITE AVEC DECALAGE
C     ALPHA EST SEPARE EN 1+EPSILON POUR DECALAGE ET ENERGIE LIGNE
C     Le facteur de correction ajoute des liens
C     Le facteur de particule par site en retire davantage
c     EQUATION : E = Ei*(N/(N-n)) ou alpha = (N/(N-n))
c                alpha = 1+epsilon/2-> epsilon = n/(N-n)

      epsilon = nh/(np-nh)

C     CHANGEMENT DE LENERGIE DE SURFACE
c     constitue une linearisation locale du potentiel quad
c     dE = 1/2*K*(a0*N)*(2dn/N*(nh/N-da/a))
```

```
      dE = 0.58*50 * float(dnh) * (nhSD-gamma)

C     CHANGEMENT DE LENERGIE DE LIGNE + CORRECTION
c     la correction de lenergie de ligne consiste
c     le 1/6 vient du fait que shc est lenergie de 6 liens
      esp = gamma-nhSD
      if (esp.LT.0.0) esp = 0

C     voir note pour le fameux 2L-6 pour le nombre de liens
C     vient du fait que un lien avec un trou ne doit pas compter

      shc = 0.1667*(108.0/alpha - 28.0*sqrt(esp))
      lgn =(1.0-epsilon/2)*float(2*L1-6)
      dE  = dE + (1.0+epsilon)*dnh*lgn*shc

C     UMBRELLA SAMPLING (si applicable)
      IF (usage.EQ.1) dE = dE + Bias(nh)

      J = dE

      RETURN
      END


C     ALGORITHME DE METROPOLIS
      SUBROUTINE Metropolis()
      IMPLICIT NONE
      INCLUDE 'parametres.h'
      INCLUDE 'variables.h'

      INTEGER x0, y0, xS, yS, x, y, dir
      INTEGER niveau, dn, pos, swap, occupe
      INTEGER debt, finl
      REAL dE, dS, S

99    CALL RANLUX(ran,1)
      x0 = (ran*2*(max+1)) - (max+1)
      CALL RANLUX(ran,1)
      y0 = (ran*2*(max+1)) - (max+1)
      IF ((y0.GT.haut(x0,0)).OR.(y0.LT.bas(x0,0))) GOTO 99

        debt = 0
        DO h = -max, max
          DO i = bas(h,0),haut(h,0)
            if (orientation(h,i).NE.0) debt = debt + 1
          END DO
        END DO


c     ON REGARDE PRELIMINAIREMENT LEFFET SUR LES SPINS
c     POSSIBLE, IMPOSSIBLE OU ENCORE UN SWAP PEUT SIMPOSER.

c     ON INVERSE TEMPORAIREMENT LA POLARITE DU SPIN CIBLE
c     POUR EN VOIR LEFFET SUR LE VOISINAGE

      occupe = occupation(x0,y0)
```

```
      occupation(x0,y0) = 1 - occupe
      swap = ENVIROLEGAL(x0,y0,xS,yS)
      occupation(x0,y0) = occupe

      IF (swap.GT.1) THEN

        IF (occupe.NE.aucun) THEN
          dir = orientation(x0,y0)
          dS = enviroJ(x0,y0) - spinJ(x0,y0,dir)
          orientation(x0,y0) = 0
          occupation(x0,y0) = aucun
          dS = dS + spinJ(xS,yS,dir)
          orientation(x0,y0) = dir
          occupation(x0,y0) = plein
        ELSE
          dir = orientation(xS,yS)
          dS = -spinJ(xS,yS,dir)
          orientation(xS,yS) = 0
          occupation(x0,y0) = plein
          dS = dS + spinJ(x0,y0,dir) - enviroJ(x0,y0)
          orientation(xS,yS) = dir
          occupation(x0,y0) = aucun
        END IF

      ELSEIF (swap.EQ.1) THEN
        dS = (1 - 2*occupe) * enviroJ(x0,y0)
      END IF

      IF (swap.NE.0) THEN

c     INVERSER TROU/BILLE ET EVALUER LE CHANGEMENT D ENERGIE
c     On connait linteraction dun site sur une membrane parfaite
c     on soustrait les interactions absantes des voisins qui sont
c     en fait des trou.  Cette energie sera soit additionne dans le
c     cas ou on rajoute un pore, soustrait si on le retire

      L1 = occupation(x0+1, y0  ) + occupation(x0+1, y0-1) +
     *     occupation(x0  , y0+1) + occupation(x0  , y0-1) +
     *     occupation(x0-1, y0+1) + occupation(x0-1, y0  )

C     ON TENTE DE RETIRER (AJOUTER) CE NOMBRE DE LIEN DU SYSTEME
C     ET AINSI LE NOMBRE DE TROUS, CE QUI CHANGE LENERGIE DE dE
c     La fonction 2x-1 transforme 0->(-1) et 1->(1)

      dn = 2*occupe-1
      dE = J(dn) + dS

C     ******************** METROPOLIS ********************

      CALL RANLUX(ran, 1)
      IF ((dE.LT.(0.0)).OR.(exp(-dE/kT).GT.ran)) THEN

C       AJUSTER LE NOMBRE DE TROUS
        notrou = notrou + dn

C       REMPLIR LES CONDITIONS PERIODIQUES AINSI QUE REAJUSTER LE
```

```
C       NOMBRE DE TROUS

        occupe = 1-occupe
        occupation(x0,y0) = occupe

C       MODIFIER LA MATRICE AVEC CONDITIONS PERIODIQUES

        x = x0 + max
        y = y0 + max + 1
        IF ((y.LE.haut(x,noniveau)).AND.(y.GE.bas(x,noniveau))
     *  .AND.(ABS(x).LE.(max+noniveau))) occupation(x,y) = occupe

        x = x0 - max
        y = y0 - max - 1
        IF ((y.LE.haut(x,noniveau)).AND.(y.GE.bas(x,noniveau))
     *  .AND.(ABS(x).LE.(max+noniveau))) occupation(x,y) = occupe

        x = x0 + 2*max + 1
        y = y0 - max
        IF ((y.LE.haut(x,noniveau)).AND.(y.GE.bas(x,noniveau))
     *  .AND.(ABS(x).LE.(max+noniveau))) occupation(x,y) = occupe

        x = x0 - 2*max - 1
        y = y0 + max
        IF ((y.LE.haut(x,noniveau)).AND.(y.GE.bas(x,noniveau))
     *  .AND.(ABS(x).LE.(max+noniveau))) occupation(x,y) = occupe

        x = x0 + max + 1
        y = y0 - 2*max - 1
        IF ((y.LE.haut(x,noniveau)).AND.(y.GE.bas(x,noniveau))
     *  .AND.(ABS(x).LE.(max+noniveau))) occupation(x,y) = occupe

        x = x0 - max - 1
        y = y0 + 2*max + 1
        IF ((y.LE.haut(x,noniveau)).AND.(y.GE.bas(x,noniveau))
     *  .AND.(ABS(x).LE.(max+noniveau))) occupation(x,y) = occupe

C       CAS OU UN NOUVEAU TROU APPARAIT
        IF (occupe.eq.aucun) THEN

          trou(notrou,1) = x0
          trou(notrou,2) = y0

C         ON ORGANISE LENVIRONNEMENT DES SPINS EN CONSEQUENCE
C         ET ON AJUSTE LA LISTE DE SPIN (AU BESOIN)

          IF ((swap).GT.1) THEN
            CALL changespin(xS,yS,dir)
            if (orientation(x0,y0).EQ.0) pause 'finalement??'
            CALL changespin(x0,y0,0)

            h = 1
            DO WHILE (((x0.NE.peptide(h,1)).OR.(y0.NE.peptide(h,2))))
              h = h + 1
            END DO
```

```
            peptide(h,1) = xS
            peptide(h,2) = yS

          END IF

C       CAS OU UN SITE REVIENT A LA VIE
        ELSE

c         ON CHERCHE LE TROU A RETIRER DE LA LISTE

          h = 1
          DO WHILE ((x0.NE.trou(h,1)).OR.(y0.NE.trou(h,2)))
            h = h + 1
          END DO
          trou(h,1) = trou(notrou+1,1)
          trou(h,2) = trou(notrou+1,2)

C         ON ORGANISE LENVIRONNEMENT DES SPINS EN CONSEQUENCE
          IF ((swap).GT.1) THEN
            CALL changespin(x0,y0,dir)
            if (orientation(xS,yS).EQ.0) pause 'finalement??'
            CALL changespin(xS,yS,0)

            h = 1
            DO WHILE (((xS.NE.peptide(h,1)).OR.(yS.NE.peptide(h,2))))
              h = h + 1
            END DO
            if (h.GT.nopep) write(*,*) 'ouuuuuuuuuups2'
            peptide(h,1) = x0
            peptide(h,2) = y0
          END IF

        END  IF

      END IF

      END IF

        finl = 0
        DO h = -max, max
          DO i = bas(h,0),haut(h,0)
            if (orientation(h,i).NE.0) finl = finl + 1
          END DO
        END DO

        if (finl.NE.nopep) then
          call impression(notrou)
          write(*,*) debt, finl, nopep, swap
          pause 'metropolis'
        end if


      RETURN
      END
```

```
C  SOUROUTINE INSERANT OU RETIRANT LES PEPTIDES
C  ON SE SERT DES SPIN(QQC) ET DE S

       SUBROUTINE metrop()
       IMPLICIT NONE
       INCLUDE 'parametres.h'
       INCLUDE 'variables.h'

       INTEGER spinleg(0:7), x0,y0, x,y, oold,onew, dN, tag
       REAL expJ(0:7), wnew, wold, cumulw, wess, w
       REAL S, dS, expdE

C      ON SE CHOISIT UN SITE OCCUPE
99     CALL RANLUX(ran,1)
       x0 = (ran*2*(max+1)) - (max+1)
       CALL RANLUX(ran,1)
       y0 = (ran*2*(max+1)) - (max+1)
       IF ((y0.GT.haut(x0,0)).OR.(y0.LT.bas(x0,0))) GOTO 99
       IF (occupation(x0,y0).EQ.aucun) GOTO 99


C      ON TENTE DINSERER OU RETIRER UN PEPTIDE
C      SI ON LINSERE, ON PREND UNE DIRECTION BIAISEE
C      SELON -Unerstanding Simulations-

       oold = orientation(x0, y0)
       IF (oold.EQ.0) THEN

       dN = 1
       DO onew = 0, 7
       If (onew.EQ.1) THEN
         spinleg(1) = 0
         expJ(1) = 0.0
       ELSE
         spinleg(onew) = spinlegal(x0,y0,onew)
         IF (spinleg(onew).EQ.1)
     *     expJ(onew)=1.0*exp((-1.0)*spinJ(x0,y0,onew)/kT)
       END IF
       END DO

C      CALCULER LE FACTEUR DE ROSENBULTH (W) DE LA NOUVELLE
C      CONFIGURATION POUR BIAISER LE CHOIX DES PROBABILITES
C      si w = 0, cest quaucune autre config nest possible

       tag = 0
       wnew = 0.0
       DO onew = 0, 7
       IF (onew.NE.1) THEN
         IF ((oold.NE.onew).AND.(spinleg(onew).EQ.1)) THEN
           wnew = wnew + expJ(onew)
           tag = 1
         END IF
       END IF
       END DO

       IF (tag.NE.1) RETURN
```

```
C     CHOISIR UNE DES ORIENTATIONS AVEC PROB P(i)=w(i)/sum(w)
c     fondee sur -understanding molecular simulations-
c     h sera lorientation dessaie

      CALL RANLUX(ran,1)
      wess = ran*wnew
      cumulw = 0.0
      onew = 1

      DO WHILE (cumulw.LT.wess)
        onew = onew + 1
        DO WHILE ((oold.EQ.onew).OR.(spinleg(onew).EQ.0))
          onew = onew + 1
        END DO
        cumulw = cumulw + expJ(onew)
      END DO

C     CALCULER LE W DE LANCIENNE CONFIGURATION POUR ACC RULE
      wold = wnew - expJ(onew) + exp(0.0)
      w = wnew/wold

      expdE = w*exp(-mu/kT)

      ELSE

      dN = -1
      w = 1
      onew = 0
      expdE = exp(-(-SPINJ(x0, y0, oold)-mu)/kT)

      END IF

C     CHANGEMENT DENERGIE ASSOCIE A CE CHANGEMENT DE SPIN

c     dS = SPINJ(x0, y0, onew)-SPINJ(x0, y0, oold)+dN*mu

C     *******  METROPOLIS POUR INSERER LE PEPTIDE  *******

      CALL RANLUX(ran, 1)
987   IF ((expdE).GT.ran) THEN

C       ON AJOUTE/RETIRE UN PEPTIDE (SPIN) SUR LA MATRICE
        nopep = nopep + dN
        CALL changespin(x0,y0,onew)

C       ON AJOUTE/RETIRE UN PEPTIDE (SPIN) DE LA LISTE
        IF (dN.EQ.1) THEN
          peptide(nopep, 1) = x0
          peptide(nopep, 2) = y0
        ELSE
          h = 1
          DO WHILE ((x0.NE.peptide(h,1)).OR.(y0.NE.peptide(h,2)))
            h = h + 1
          END DO
```

```
          peptide(h,1) = peptide(nopep+1,1)
          peptide(h,2) = peptide(nopep+1,2)
        END IF

      END IF

      RETURN
      END


C  FONCTION N0US CALCULANTS LE NIVEAU DE DISTANCE
c    La premiere condition performe un chagement de
c    variable si necessaire, voir notes pour plus de
c    detail sur la methode de calcul
      FUNCTION n(dx, dy)
      Integer i, n, nn, dx, dy, ax, ay, sum, dif, small
        nn = 0
        ax = abs(dx)
        ay = abs(dy)
              IF ((dx*dy).lt.0) THEN
          IF (ax.lt.ay) THEN
            small = ax
          ELSE
            small = ay
          ENDIF
          ax = abs(ax-ay)
          ay = small
        ENDIF
        sum = ax + ay
        dif = abs(ax - ay)
        DO 234 i = 1, sum
234     nn = nn + i/2 + 1
        n = nn - (sum-dif)/2
      RETURN
      END


C  FONCTION CALCULANT LE NO DE PORE DANS UNE MEMBRANE

      FUNCTION nopore()
      IMPLICIT NONE
      INCLUDE 'parametres.h'
      INCLUDE 'variables.h'
      INTEGER visite, x, y, x0, y0
      INTEGER grille(-max:max,-max:max)
      INTEGER pile(max**2, 2)
      INTEGER start, finish
        nopore = 0
        visite = 0
        DO 541 x0 = -max, max
          DO 542 y0 = -max, max
            grille(x0, y0) = 0
542       CONTINUE
541     CONTINUE
```

```
C       CE WHILE EST POUR REUSSIR A VOIR TOUS LES TROUS
        do 550 while (visite.LT.notrou)

C         SI PAS TOUS VU, ON TROUVE UN TROU QUI NA PAS ETE VISITE
          do 551 i = 1, notrou
            x0 = trou(i,1)
            y0 = trou(i,2)
            if (grille(x0,y0).EQ.0) goto 552
551       CONTINUE

C         ON LAJOUTE A LA VISITE ET ON A UN NOUVEAU PORE
552       visite = visite + 1
          nopore = nopore + 1
          grille(x0,y0) = 1
          Pile(1,1) = x0
          Pile(1,2) = y0
          start = 1
          finish = 1

C         ON FOUILLE ALENTOUR POUR CONNAITRE LES TROUS CONNEXES
          do 553 WHILE (start.LE.finish)

             x = pile(start,1) + 1
             y = pile(start,2)
             CALL enqueue(pile,grille,finish,visite,x,y)

             x = pile(start,1) + 1
             y = pile(start,2) - 1
             CALL enqueue(pile,grille,finish,visite,x,y)

             x = pile(start,1)
             y = pile(start,2) + 1
             CALL enqueue(pile,grille,finish,visite,x,y)

             x = pile(start,1) - 1
             y = pile(start,2)
             CALL enqueue(pile,grille,finish,visite,x,y)

             x = pile(start,1) - 1
             y = pile(start,2) + 1
             CALL enqueue(pile,grille,finish,visite,x,y)

             x = pile(start,1)
             y = pile(start,2) - 1
             CALL enqueue(pile,grille,finish,visite,x,y)

             start = start + 1

553        CONTINUE
550       CONTINUE
      RETURN
      END


C  SOUS ROUTINE PRENANT NOTE DES PROBABILITES DANS UN TABLEAU INDEXE
C  Attention : pour le umbrella sampling
```

```
        SUBROUTINE prob(P)
      IMPLICIT NONE
      INCLUDE 'parametres.h'
      INCLUDE 'variables.h'
      INTEGER P(0:5000), taille
        COMMON taille
          IF (taille.EQ.(-1)) THEN
            taille = 0
            DO 222 i = 0, 5000
222           P(i) = 0
        ELSE
          taille = taille + 1
            P(notrou) = P(notrou) + 1
        END IF
      RETURN
      END


C  FONCTION CALCULANT LE NO ALEATOIRE (MARC PEPIN)
      SUBROUTINE RANLUX(RVEC,LENV)
C         Subtract-and-borrow random number generator proposed by
C         Marsaglia and Zaman, implemented by F. James with the name
C         RCARRY in 1991, and later improved by Martin Luescher
C         in 1993 to produce "Luxury Pseudorandom Numbers".
C     Fortran 77 coded by F. James, 1993
C
C       references:
C  M. Luscher, Computer Physics Communications  79 (1994) 100
C  F. James, Computer Physics Communications 79 (1994) 111
C
C   LUXURY LEVELS.
C   ------ ------      The available luxury levels are:
C
C  level 0  (p=24): equivalent to the original RCARRY of Marsaglia
C           and Zaman, very long period, but fails many tests.
C  level 1  (p=48): considerable improvement in quality over level 0,
C           now passes the gap test, but still fails spectral test.
C  level 2  (p=97): passes all known tests, but theoretically still
C           defective.
C  level 3  (p=223): DEFAULT VALUE.  Any theoretically possible
C           correlations have very small chance of being observed.
C  level 4  (p=389): highest possible luxury, all 24 bits chaotic.
C
C!!! +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C!!!  Calling sequences for RANLUX:                                  ++
C!!!      CALL RANLUX (RVEC, LEN)   returns a vector RVEC of LEN     ++
C!!!                   32-bit random floating point numbers between  ++
C!!!                   zero (not included) and one (also not incl.). ++
C!!!      CALL RLUXGO(LUX,INT,K1,K2) initializes the generator from  ++
C!!!                one 32-bit integer INT and sets Luxury Level LUX  ++
C!!!                which is integer between zero and MAXLEV, or if   ++
C!!!                LUX .GT. 24, it sets p=LUX directly.  K1 and K2   ++
C!!!                should be set to zero unless restarting at a break++
C!!!                point given by output of RLUXAT (see RLUXAT).     ++
C!!!      CALL RLUXAT(LUX,INT,K1,K2) gets the values of four integers++
```

```
C!!!               which can be used to restart the RANLUX generator ++
C!!!               at the current point by calling RLUXGO.  K1 and K2++
C!!!               specify how many numbers were generated since the ++
C!!!               initialization with LUX and INT.  The restarting  ++
C!!!               skips over  K1+K2*E9   numbers, so it can be long.++
C!!!   A more efficient but less convenient way of restarting is by: ++
C!!!      CALL RLUXIN(ISVEC)    restarts the generator from vector   ++
C!!!                    ISVEC of 25 32-bit integers (see RLUXUT)      ++
C!!!      CALL RLUXUT(ISVEC)    outputs the current values of the 25 ++
C!!!                  32-bit integer seeds, to be used for restarting ++
C!!!      ISVEC must be dimensioned 25 in the calling program         ++
C!!! ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      DIMENSION RVEC(LENV)
      DIMENSION SEEDS(24), ISEEDS(24), ISDEXT(25)
      PARAMETER (MAXLEV=4, LXDFLT=3)
      DIMENSION NDSKIP(0:MAXLEV)
      DIMENSION NEXT(24)
      PARAMETER (TWOP12=4096., IGIGA=1000000000,JSDFLT=314159265)
      PARAMETER (ITWO24=2**24, ICONS=2147483563)
      SAVE NOTYET, I24, J24, CARRY, SEEDS, TWOM24, TWOM12, LUXLEV
      SAVE NSKIP, NDSKIP, IN24, NEXT, KOUNT, MKOUNT, INSEED
      INTEGER LUXLEV
      LOGICAL NOTYET
      DATA NOTYET, LUXLEV, IN24, KOUNT, MKOUNT /.TRUE., LXDFLT, 0,0,0/
      DATA I24,J24,CARRY/24,10,0./
C                                 default
C  Luxury Level   0     1     2   *3*    4
      DATA NDSKIP/0,   24,   73,  199,  365 /
Corresponds to p=24    48    97   223   389
C     time factor 1     2     3     6    10   on slow workstation
C                 1   1.5     2     3     5   on fast mainframe
C
C  NOTYET is .TRUE. if no initialization has been performed yet.
C              Default Initialization by Multiplicative Congruential
      IF (NOTYET) THEN
         NOTYET = .FALSE.
         JSEED = JSDFLT
         INSEED = JSEED
         WRITE(6,'(A,I12)') ' RANLUX DEFAULT INITIALIZATION: ',JSEED
         LUXLEV = LXDFLT
         NSKIP = NDSKIP(LUXLEV)
         LP = NSKIP + 24
         IN24 = 0
         KOUNT = 0
         MKOUNT = 0
         WRITE(6,'(A,I2,A,I4)')  ' RANLUX DEFAULT LUXURY LEVEL =  ',
     +        LUXLEV,'      p =',LP
         TWOM24 = 1.
         DO 25 I= 1, 24
            TWOM24 = TWOM24 * 0.5
         K = JSEED/53668
         JSEED = 40014*(JSEED-K*53668) -K*12211
         IF (JSEED .LT. 0)  JSEED = JSEED+ICONS
         ISEEDS(I) = MOD(JSEED,ITWO24)
   25    CONTINUE
         TWOM12 = TWOM24 * 4096.
```

```
         DO 50 I= 1,24
         SEEDS(I) = REAL(ISEEDS(I))*TWOM24
         NEXT(I) = I-1
   50    CONTINUE
         NEXT(1) = 24
         I24 = 24
         J24 = 10
         CARRY = 0.
         IF (SEEDS(24) .EQ. 0.) CARRY = TWOM24
      ENDIF
C
C          The Generator proper: "Subtract-with-borrow",
C          as proposed by Marsaglia and Zaman,
C          Florida State University, March, 1989
C
      DO 100 IVEC= 1, LENV
      UNI = SEEDS(J24) - SEEDS(I24) - CARRY
      IF (UNI .LT. 0.)  THEN
         UNI = UNI + 1.0
         CARRY = TWOM24
      ELSE
         CARRY = 0.
      ENDIF
      SEEDS(I24) = UNI
      I24 = NEXT(I24)
      J24 = NEXT(J24)
      RVEC(IVEC) = UNI
C  small numbers (with less than 12 "significant" bits) are "padded".
      IF (UNI .LT. TWOM12)  THEN
         RVEC(IVEC) = RVEC(IVEC) + TWOM24*SEEDS(J24)
C        and zero is forbidden in case someone takes a logarithm
         IF (RVEC(IVEC) .EQ. 0.)  RVEC(IVEC) = TWOM24*TWOM24
      ENDIF
C        Skipping to luxury.  As proposed by Martin Luscher.
      IN24 = IN24 + 1
      IF (IN24 .EQ. 24)  THEN
         IN24 = 0
         KOUNT = KOUNT + NSKIP
         DO 90 ISK= 1, NSKIP
         UNI = SEEDS(J24) - SEEDS(I24) - CARRY
         IF (UNI .LT. 0.)  THEN
            UNI = UNI + 1.0
            CARRY = TWOM24
         ELSE
            CARRY = 0.
         ENDIF
         SEEDS(I24) = UNI
         I24 = NEXT(I24)
         J24 = NEXT(J24)
   90    CONTINUE
      ENDIF
  100 CONTINUE
      KOUNT = KOUNT + LENV
      IF (KOUNT .GE. IGIGA)  THEN
         MKOUNT = MKOUNT + 1
         KOUNT = KOUNT - IGIGA
```

```
      ENDIF
      RETURN
C
C           Entry to input and float integer seeds from previous run
      ENTRY RLUXIN(ISDEXT)
         TWOM24 = 1.
         DO 195 I= 1, 24
         NEXT(I) = I-1
  195    TWOM24 = TWOM24 * 0.5
         NEXT(1) = 24
         TWOM12 = TWOM24 * 4096.
      WRITE(6,'(A)') ' FULL INITIALIZATION OF RANLUX WITH 25 INTEGERS:'
      WRITE(6,'(5X,5I12)') ISDEXT
      DO 200 I= 1, 24
      SEEDS(I) = REAL(ISDEXT(I))*TWOM24
  200 CONTINUE
      CARRY = 0.
      IF (ISDEXT(25) .LT. 0)  CARRY = TWOM24
      ISD = IABS(ISDEXT(25))
      I24 = MOD(ISD,100)
      ISD = ISD/100
      J24 = MOD(ISD,100)
      ISD = ISD/100
      IN24 = MOD(ISD,100)
      ISD = ISD/100
      LUXLEV = ISD
        IF (LUXLEV .LE. MAXLEV) THEN
          NSKIP = NDSKIP(LUXLEV)
          WRITE (6,'(A,I2)') ' RANLUX LUXURY LEVEL SET BY RLUXIN TO: ',
     +                        LUXLEV
        ELSE  IF (LUXLEV .GE. 24) THEN
          NSKIP = LUXLEV - 24
          WRITE (6,'(A,I5)') ' RANLUX P-VALUE SET BY RLUXIN TO:',LUXLEV
        ELSE
          NSKIP = NDSKIP(MAXLEV)
          WRITE (6,'(A,I5)') ' RANLUX ILLEGAL LUXURY RLUXIN: ',LUXLEV
          LUXLEV = MAXLEV
        ENDIF
      INSEED = -1
      RETURN
C
C                     Entry to ouput seeds as integers
      ENTRY RLUXUT(ISDEXT)
      DO 300 I= 1, 24
         ISDEXT(I) = INT(SEEDS(I)*TWOP12*TWOP12)
  300 CONTINUE
      ISDEXT(25) = I24 + 100*J24 + 10000*IN24 + 1000000*LUXLEV
      IF (CARRY .GT. 0.)  ISDEXT(25) = -ISDEXT(25)
      RETURN
C
C                     Entry to output the "convenient" restart point
      ENTRY RLUXAT(LOUT,INOUT,K1,K2)
      LOUT = LUXLEV
      INOUT = INSEED
      K1 = KOUNT
      K2 = MKOUNT
```

```
      RETURN
C
C                     Entry to initialize from one or three integers
      ENTRY RLUXGO(LUX,INS,K1,K2)
         IF (LUX .LT. 0) THEN
            LUXLEV = LXDFLT
         ELSE IF (LUX .LE. MAXLEV) THEN
            LUXLEV = LUX
         ELSE IF (LUX .LT. 24 .OR. LUX .GT. 2000) THEN
            LUXLEV = MAXLEV
            WRITE (6,'(A,I7)') ' RANLUX ILLEGAL LUXURY RLUXGO: ',LUX
         ELSE
            LUXLEV = LUX
            DO 310 ILX= 0, MAXLEV
              IF (LUX .EQ. NDSKIP(ILX)+24)  LUXLEV = ILX
  310       CONTINUE
         ENDIF
      IF (LUXLEV .LE. MAXLEV)  THEN
         NSKIP = NDSKIP(LUXLEV)
c         WRITE(6,'(A,I2,A,I4)') ' RANLUX LUXURY LEVEL SET BY RLUXGO
:',
c     +        LUXLEV,'     P=', NSKIP+24
      ELSE
          NSKIP = LUXLEV - 24
          WRITE (6,'(A,I5)') ' RANLUX P-VALUE SET BY RLUXGO TO:',LUXLEV
      ENDIF
      IN24 = 0
      IF (INS .LT. 0)  WRITE (6,'(A)')
     +    ' Illegal initialization by RLUXGO, negative input seed'
      IF (INS .GT. 0)  THEN
        JSEED = INS
c        WRITE(6,'(A,3I12)') ' RANLUX INITIALIZED BY RLUXGO FROM
SEEDS',
c     +      JSEED, K1,K2
      ELSE
        JSEED = JSDFLT
        WRITE(6,'(A)')' RANLUX INITIALIZED BY RLUXGO FROM DEFAULT SEED'
      ENDIF
      INSEED = JSEED
      NOTYET = .FALSE.
      TWOM24 = 1.
         DO 325 I= 1, 24
           TWOM24 = TWOM24 * 0.5
         K = JSEED/53668
         JSEED = 40014*(JSEED-K*53668) -K*12211
         IF (JSEED .LT. 0)  JSEED = JSEED+ICONS
         ISEEDS(I) = MOD(JSEED,ITWO24)
  325    CONTINUE
      TWOM12 = TWOM24 * 4096.
         DO 350 I= 1,24
         SEEDS(I) = REAL(ISEEDS(I))*TWOM24
         NEXT(I) = I-1
  350    CONTINUE
      NEXT(1) = 24
      I24 = 24
      J24 = 10
```

```
      CARRY = 0.
      IF (SEEDS(24) .EQ. 0.) CARRY = TWOM24
C        If restarting at a break point, skip K1 + IGIGA*K2
C        Note that this is the number of numbers delivered to
C        the user PLUS the number skipped (if luxury .GT. 0).
      KOUNT = K1
      MKOUNT = K2
      IF (K1+K2 .NE. 0)  THEN
        DO 500 IOUTER= 1, K2+1
          INNER = IGIGA
          IF (IOUTER .EQ. K2+1)  INNER = K1
          DO 450 ISK= 1, INNER
            UNI = SEEDS(J24) - SEEDS(I24) - CARRY
            IF (UNI .LT. 0.)  THEN
               UNI = UNI + 1.0
               CARRY = TWOM24
            ELSE
               CARRY = 0.
            ENDIF
            SEEDS(I24) = UNI
            I24 = NEXT(I24)
            J24 = NEXT(J24)
  450     CONTINUE
  500   CONTINUE
C         Get the right value of IN24 by direct calculation
        IN24 = MOD(KOUNT, NSKIP+24)
        IF (MKOUNT .GT. 0)  THEN
           IZIP = MOD(IGIGA, NSKIP+24)
           IZIP2 = MKOUNT*IZIP + IN24
           IN24 = MOD(IZIP2, NSKIP+24)
        ENDIF
C       Now IN24 had better be between zero and 23 inclusive
        IF (IN24 .GT. 23) THEN
           WRITE (6,'(A/A,3I11,A,I5)')
     +     '  Error in RESTARTING with RLUXGO:','  The values', INS,
     +     K1, K2, ' cannot occur at luxury level', LUXLEV
           IN24 = 0
        ENDIF
      ENDIF
      RETURN
      END


C  FONCTION RETOURNANT LA VALEUR DE LENERGIE DE SPIN DUN SITE
c  base sur le meme algorithme que legalspin sauf que ce sont
c  les trois sites avant plutot que arriere qui interviennent

        FUNCTION spinJ(x0, y0, direction)
        IMPLICIT NONE
        INCLUDE 'parametres.h'
        INCLUDE 'variables.h'

        INTEGER x0, y0, direction, avant, d

c       POUR CHACUNE DES ORIENTATIONS POSSIBLES, ON COMPTE LE NO
c       DE TROU SUR LES 3 SITES AVANT DE LA DIRECTION DONNEE
```

```
          IF (direction.EQ.aucun) THEN
            spinJ = 0

          ELSEIF (direction.EQ.2) THEN
            avant = occupation(x0+d,y0-d)+occupation(x0+d,y0)+
     *              occupation(x0,y0+d)
            spinJ = 0.3*float(avant) - 0.3

          ELSEIF (direction.EQ.3) THEN
            avant = occupation(x0+d,y0)+occupation(x0,y0+d)+
     *              occupation(x0-d,y0+d)
            spinJ = 0.3*float(avant) - 0.3

          ELSEIF (direction.EQ.4) THEN
            avant = occupation(x0,y0+d)+occupation(x0-d,y0+d)+
     *              occupation(x0-d,y0)
            spinJ = 0.3*float(avant) - 0.3

          ELSEIF (direction.EQ.5) THEN
            avant = occupation(x0-d,y0+d)+occupation(x0-d,y0)+
     *              occupation(x0,y0-d)
            spinJ = 0.3*float(avant) - 0.3

          ELSEIF (direction.EQ.6) THEN
            avant = occupation(x0-d,y0)+occupation(x0,y0-d)+
     *              occupation(x0+d,y0-d)
            spinJ = 0.3*float(avant) - 0.3

          ELSEIF (direction.EQ.7) THEN
            avant = occupation(x0,y0-d)+occupation(x0+d,y0-d)+
     *              occupation(x0+d,y0)
            spinJ = 0.3*float(avant) - 0.3

          ELSEIF (direction.EQ.proteine) THEN

C         PREMIER NIVEAU
          L(1) = occupation(x0+1, y0  ) + occupation(x0+1, y0-1) +
     *           occupation(x0  , y0+1) + occupation(x0  , y0-1) +
     *           occupation(x0-1, y0+1) + occupation(x0-1, y0  )

C         DEUXIEME NIVEAU
          L(2) = occupation(x0+2, y0-1) + occupation(x0+1, y0+1) +
     *           occupation(x0+1, y0-2) + occupation(x0-1, y0+2) +
     *           occupation(x0-1, y0-1) + occupation(x0-2, y0+1)

C         TROISIEME NIVEAU
          L(3) = occupation(x0+2, y0  ) + occupation(x0+2, y0-2) +
     *           occupation(x0  , y0+2) + occupation(x0  , y0-2) +
     *           occupation(x0-2, y0+2) + occupation(x0-2, y0  )

C         Energie proteine-lipide (1tier de 3J)
C         Si ilya des trous presents, lenergie monte pas, donc favorable
          spinJ = 0.1*L(1) + 0.05*L(2) + 0.02*L(3)

          END IF
```

```
      RETURN
      END


C  FONCTION DISANT SI OUI(1) OU NON(0) UNE INSERTION DE
C  PEPTIDE EST POSSIBLE


      FUNCTION spinlegal(x0, y0, direct)
      IMPLICIT NONE
      INCLUDE 'parametres.h'
      INCLUDE 'variables.h'

      INTEGER x0, y0, direct, legalspin

c     ON SUPPOSE QUE CE NEST PAS POSSIBLE
      legalspin = 0

c     PAS BESOIN DE SAVOIR SI DES FLECHES POINTENT SUR CE SITE CAR
c     METRO VIA ENVIROLEGAL NE PERMET PAS A UN LIPIDE DOCCUPER
c     CE SITE : VOIR SI LIPIDE SEUL (SANS PEPTIDE DEJA PRESENT)

      IF (occupation(x0,y0).NE.aucun) THEN

c     ON REGARDE PRIMO SI LA DIRECTION POINTE SUR UN TROU ET PAS DE
c     TROU SUR LE SITE ARRIERE DU PEPTIDE (DEPEND DES DIRECTIONS)

        IF (direct.EQ.aucun) THEN
          legalspin = 1
       ELSEIF ((occupation(x0+dirX(direct),y0+dirY(direct)).EQ.0).AND.
     *(occupation(x0-dirX(direct),y0-dirY(direct)).EQ.1)) THEN
          legalspin = 1
        END IF

      END IF

      spinlegal = legalspin

      RETURN
      END
```