BACHELOR

Energy dissipation in breaking double networks

Slangen, T.M.

Award date:
2019

Technische Universiteit Eindhoven

Bachelor Final Project

Energy dissipation in breaking double networks

Supervisors Dr. W.G. Ellenbroek
Msc. A. Bose
Dr. N.B. Tito

June 19, 2019

Abstract

This report discusses a method of analyzing polymeric materials under stress, specifically partially pre-stretched double network elastomers. These networks are treated as following Gaussians statistics. From this, partition functions, entropy and free energy of these networks are derived. A method is developed which uses this to efficiently predict the process of stretching and breaking these networks, as well as compute the dissipation energy during such a process.
This simulation is then applied to two samples to illustrate it. The results show that the double network is significantly stronger than an equally sized single network, which is in line with previous research on this matter.

Contents

1 Introduction

Polymeric materials find widespread applications, from cheap plastic bottles to more solid building materials and from nylon in clothing to rubber in tires. Especially for elastomers, such as the rubber in the last example, understanding the behaviour of the material under stress is crucial. When developing new elastomers, rather than comparing options experimentally, a theoretical analysis can provide a faster and more cost-effective way to investigate their properties. [1][2] Additionally, parameters can be varied explicitly and a theoretical approach can help explore regions that are difficult to study experimentally.

So-called double networks where two (usually different) networks are interwoven and linked together have been applied successfully to hydro-gel materials [3][4] and could prove interesting for elastomers as well. [5] Developing such a network requires not only a choice of material. Its properties depend to a large extent on the interdependence of the different networks. Examples include having one network be more stretched out or more cross-linked than the other. Because of combinatorial explosion this yields too many options to easily analyze in an experimental setting. These types of networks are therefore of specific interest to investigate theoretically and will therefore be the main focus of this report.

The main goal of the project covered in this report was to develop an efficient method to compute the dissipation energy as a result of the breaking of bonds in a double polymer network. The method, which is discussed in section 3, uses Gaussian statistics to find the partition function and by extent free energy of networks. This statistical approach also provides a way to study the stretch of individual strands and can be used to derive a breaking condition for them. This will then allow us to write a simulation which goes through the breaking process of a (double) network step by step, computing dissipation energy along the way.
This method will also yield several additional results, covered in section 4, including theoretical stress-strain curves, as well as some insight into double networks.

2 Theory

This section will briefly cover the theory required to understand the method section.

2.1 Polymer networks as Gaussian chains

Subsection 2.1 is a summary of the matter covered in the book "Polymer Physics" by M. Rubinstein and R. Colby. [6]

2.1.1 The model

The model discussed in this projects will treat polymer networks as follows. The bridges making up the networks will be modeled as a chain of N freely connected segments, also called bonds, with an effective length b, called Kuhn length. For a polymer where the segments are freely connected, this Kuhn length is simply the expected length of one such segment. For a network where this does not hold, this Kuhn length allows us to treat the network as a freely joined network.
A point where chains of such segments cross is called a cross-linking point, the set of bonds connecting two cross-linking points to each other is referred to as a bridge or strand.

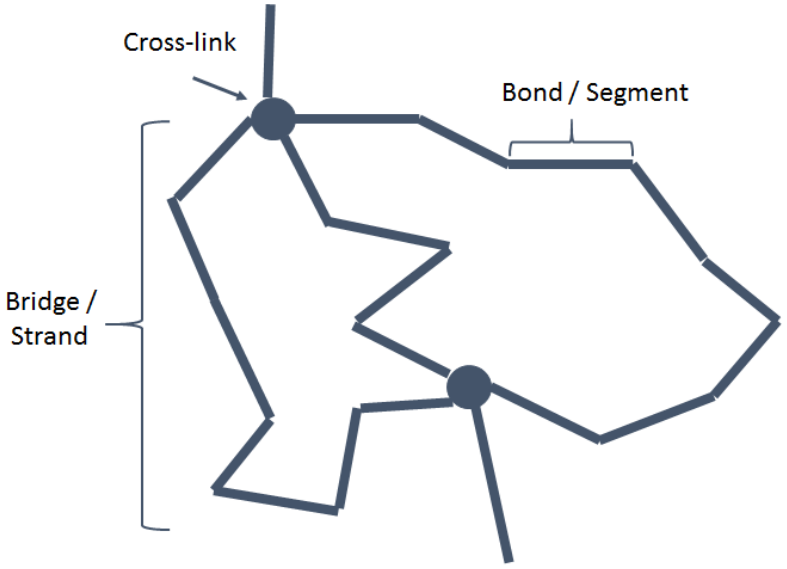


Figure 1: A picture illustrating the terminology

2.1.2 The average squared end-to-end length

For the expected squared end-to-end length of a bridge, the following holds.

$$< \vec{R}^2 >= \langle (\sum_{i=1}^{N} \vec{r_i}) \cdot (\sum_{j=1}^{N} \vec{r_j}) \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} < \vec{r_i} \cdot \vec{r_j} > \tag{1}$$

Since each segment has the same length, we have, with θ_{ij} the angle between the i-th and j-th segment,

$$< \vec{R}^2 >= \sum_{i=1}^{N} \sum_{j=1}^{N} < \vec{r_i} \cdot \vec{r_j} >= b^2 \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} < cos(\theta_{ij}) > \tag{2}$$

Since we are studying a freely joined network where are angles have equal probability, or a network equivalent to it, the terms in this sum all cancel out, except for when i and j are equal, in which case the angle is 0 and the cosine yields 1. This happens N times, so this yields

$$< \vec{R}^2 >= Nb^2 \tag{3}$$

2.1.3 The probability distribution

Our bridges are given by freely jointed bonds, and can therefore be modeled as a random walk. A distribution for the end-to-end distance of a one-dimensional random walk in the x-direction can be approximated by a Gaussian. In this case, it holds

$$P(x) = \frac{1}{\sqrt{2\pi < x^2 >}} \exp\left(\frac{-x^2}{2 < x^2 >}\right) \tag{4}$$

We know $< \vec{R}^2 >= Nb^2$. Since the X, Y and Z directions are equivalent and perpendicular, each of them should have an equal contribution. Therefore $< R_X^2 >= \frac{Nb^2}{3}$, and the same holds for the other two directions. This brings us to

$$P(R_X, N) = \sqrt{\frac{3}{2\pi N b^2}} \exp\left(\frac{-3R_X^2}{2Nb^2}\right) \tag{5}$$

The probability distributions of the other two directions are identical. Independence of these distributions means the 3-dimensional distribution is given by their product. Combined with $R_X^2 + R_Y^2 + R_Z^2 = \vec{R}^2$, this yields

$$P(\vec{R}, N) = \left(\frac{3}{2\pi b^2 N}\right)^{3/2} \exp\left(-\frac{3\vec{R}^2}{2b^2 N}\right) \tag{6}$$

2.2 Forces on bridges

The free energy of a strand is given by the Boltzmann constant k_B multiplied with the temperature T and the entropy of the strand, times -1. The entropy is the logarithm of the probability distribution. Therefore, we can use equation 6 from the previous section to find

$$F = F_0 + \frac{3k_BT}{2} \cdot \frac{\vec{R}^2}{N \cdot b^2} \tag{7}$$

F_0 is a constant that doesn't matter for the point of this section, so we re-scale the energy such that

$$F = \frac{3k_BT}{2} \cdot \frac{\vec{R}^2}{N \cdot b^2} \tag{8}$$

The force needed to stretch the strand is given by the derivative of the free energy with respect to $\vec{R}$, so

$$f = \frac{\delta F}{\delta \vec{R}} = 3k_BT \cdot \frac{\vec{R}}{N \cdot b^2} \tag{9}$$

Imposing a breaking point in terms of strand length is therefore equivalent with imposing a threshold force f_t. Squaring and rewriting equation 9 yields the break condition

$$\vec{R}^2 \geq \frac{f_t^2 \cdot (Nb^2)^2}{(3k_BT)^2} \tag{10}$$

3 Method

This section will expand upon the theory section, by discussing how the network can be written in terms of Gaussians and how these can be used not only to compute energy levels at different stages of breaking, but also to determine which parts of the network break first and the point at which they break.

3.1 The model

The network will be considered as a set of crosslinking points and the bridges in between. These crosslinking points exist both between the two generations and within each generation. The crosslinking points $x_1, ..., x_n$ will only be used implicitly and their location is not used or computed. The bridges, henceforth denoted by $\vec{r_1}, ..., \vec{r_m}$ will be assigned a length analogue to the one discussed in section 2.1.
Since the goal is to study stress, we require a model which allows for scaling of the size of the network in three dimensions. This can be accomplished by taking fixed points and then scaling the distance in between those points, or by placing the network in a periodic box and then changing the dimensions of this box. In this project, the latter was chosen, but the former leads to nearly identical computations.

3.2 From network to free energy

This section will describe how to obtain the free energy of a given network. This will be done by finding the entropy of the system. To this end, an integral form of the partition function will be obtained and solved.

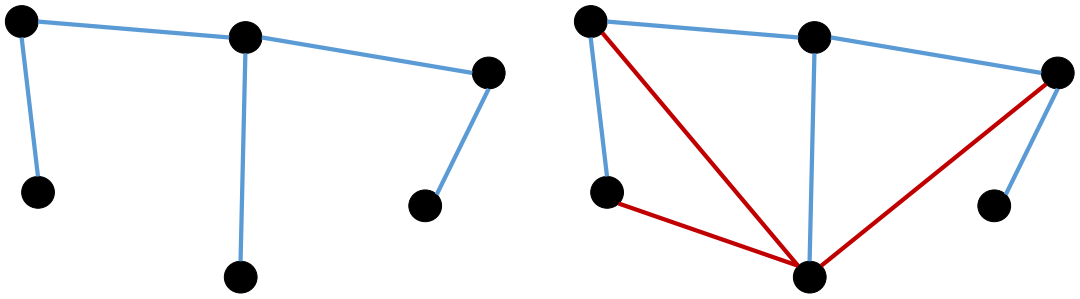
3.2.1 Requirements

In order to analyze networks, we need two things. The first is the set of bridge lengths. Formally, we also need the Kuhn lengths b, since these could be different for the two cross-linked networks, but as will become clear later in this section, b only appears in the formulae in the form of $b^2 \cdot N$. Therefore, the Kuhn length can be accounted for by re-scaling the bridge lengths.
The second thing we need is a set of constraints, which encapsulates the structure

of the network. These constraints are always of the form

$$\sum_{i=1}^{n} c_i \cdot \vec{r_i} = \vec{r_c} \tag{11}$$

with n the number of bridges in the network, c_i an integer, $-1, 0$ or 1. For so called loops, $\vec{r_c}$ is the zero-vector, whereas groups of bridges spanning across periodic borders will have a non-zero vector on the right hand side.
A system with m bridges and n cross-linking points need $n-1$ bridges to connect the points in the network and should therefore have $m-n+1$ constraints, as illustrated in the image below.



(a) A network with 6 nodes, therefore requiring 6-1=5 bridges to connect these nodes

(b) The other bridges cause loops, 8-(6-1)=3 in total

Figure 2

These constraints need not be unique, but every linearly independent set of $m-n+1$ correct constraints is equivalent. These constraints can be identified from a network in multiple ways, the ideal method depending on the available format of the network. In this project, the following was chosen.

3.2.2 Finding the constraints

This method requires one to know for each bridge $\vec{r_1}, ..., \vec{r_m}$ which points in the set $x_1, ..., x_n$ it connects, and whether this connection crosses a periodic boundary and if so which one. The constraints will be found by defining a new network.

Start at any point, say x_1. Now take any bridge connecting the point x_1. Define $\vec{\tilde{r}_1}$ as the chosen bridge, with the direction chosen from x_1 to the other point, and remove that bridge from the original bridge list. Now continue from the point $\vec{\tilde{r}_1}$ connected x_1 to and repeat the process.
If at any point a bridge connects from a point x_p to a point x_q already in the new network, a loop has been discovered. This new bridge is now instead defined as $\vec{\tilde{r}_n}$, or $\vec{\tilde{r}_{n+1}}, \vec{\tilde{r}_{n+2}}, ..$ the following times this happens. The left hand side of the constraint is now defined as $\vec{\tilde{r}_n}$ plus the path from x_1 to x_p, minus the path from x_1 to x_q.
Since the new network $\vec{\tilde{r}_1}, ..., \vec{\tilde{r}_{n-1}}$ we are creating is a one-directional tree, these paths are unique. The right hand side of these constraints are found by adding or subtracting the corresponding vector $\vec{r_c}$, see equation 11, for each bridge on the left hand side.
A problem that may arise is that there is no bridge left that is connected to the most recently added point. If this happens, the process is instead continued from any point already in the new network. Since the network being studied is connected, such a point can always be found.

The figure below shows the algorithm graphically.

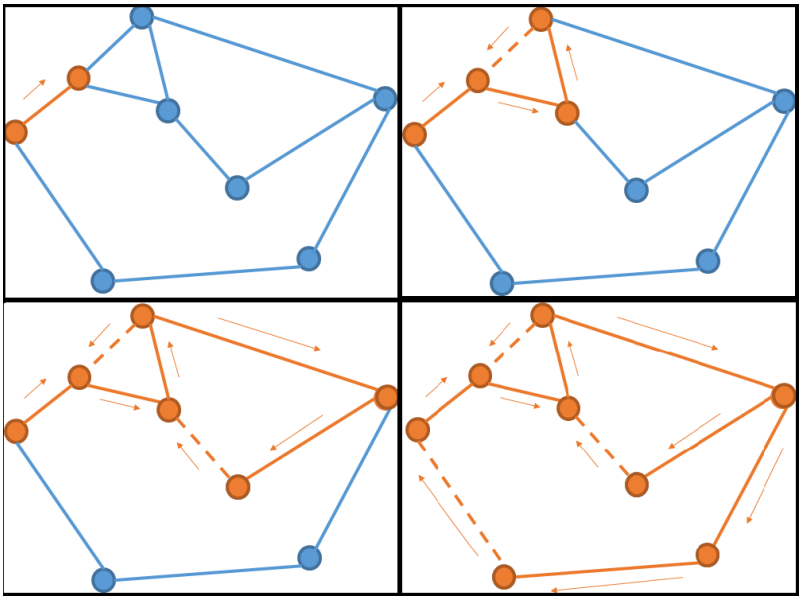


Figure 3: The constraint finding algorithm:
1. Adding a bridge to the tree
2. Discovering a loop
3. Continue after a loop
4. The end-result

The end product is a re-ordered set of bridges, in some cases mirrored, and a set of $m - n + 1$ linearly independent constraints.

3.2.3 The partition function

Defining the partition function
The method discussed in this subsection is based on notes from dr. Ellenbroek. [7] The script which is the end product of this project is an extension of his script which computed the free energy and expected location of individual bridges for two-dimensional networks.

Recall from section 2.1 that the probability distribution of the end-to-end vector $\vec{r}$ of a three dimensional bridge consisting of N Kuhn segments is given by

$$P(\vec{r}, N) = (\frac{3}{2\pi b^2 N})^{3/2} \exp{(-\frac{3\vec{r}^2}{2b^2 N})} \tag{12}$$

Henceforth, we will use the symbol K as propagator, with $K(\vec{r}, N) = P(\vec{r}, N)$. Its Fourier transform and reverse Fourier transform are

$$\begin{aligned} \tilde{K}(\vec{q}, N) &= \int K(\vec{r}, N) e^{-i\vec{q}\cdot\vec{r}} d\vec{r} = \exp{(-\frac{1}{4} b^2 N \vec{q}^2)} \\ K(\vec{r}, N) &= \frac{1}{(2\pi)^3} \int \tilde{K}(\vec{q}, N) e^{i\vec{q}\cdot\vec{r}} d\vec{q} \end{aligned} \tag{13}$$

If the bridges were independent, the partition function of the system would be defined as

$$Q = \int \Big[\prod_{i=1}^{n} K(\vec{r_i}, N_i) \Big] \Big[\prod_{i=1}^{n} d\vec{r_i} \Big] \tag{14}$$

However, the aforementioned loops present in the system cause dependence. This can be incorporated into the partition function by multiplying the interior of the integral with a delta function for each constraint. We will need the Fourier representations of these delta functions, which are given by

$$\delta_k(\vec{r}) = \frac{1}{(2\pi)^3} \int e^{-\vec{q_k}\cdot\vec{r}} d\vec{q_k} \tag{15}$$

Here, the vector $\vec{r}$ is the left hand side of equation 11 minus the right hand side. This brings us to the following partition function

$$\begin{aligned} Q = \frac{1}{(2\pi)^{3\cdot(m-n+1)}} \int \Big[\prod_{i=1}^{n} K(\vec{r_i}, N_i) \Big] \Big[\prod_{k=1}^{m-n+1} \exp{(-i\vec{q_k} \cdot (\sum_{i=1}^{n} \big[c_{i,k} \cdot \vec{r_i}\big] - \vec{r_{c,k}}))} \Big] \\ \Big[\prod_{i=1}^{n} d\vec{r_i} \Big] \Big[\prod_{k=1}^{m-n+1} d\vec{q_k} \Big] \end{aligned} \tag{16}$$

Solving the integral
First, we integrate over $\vec{r_1}, ...\vec{r_n}$, which yields

$$Q = \frac{1}{(2\pi)^{3\cdot(m-n+1)}} \cdot \int \Big[\prod_{i=1}^{n} \tilde{K}\big(\sum_{k=1}^{m-n+1} \big[c_{i,k}\cdot\vec{q_k}\big], N_i\big)\Big]\Big[\prod_{k=1}^{m-n+1} \big[\exp\left(i\vec{q_k}\cdot\vec{r_{c,k}}\right)\big]\Big]\Big[\prod_{k=1}^{m-n+1} d\vec{q_k}\Big] \tag{17}$$

Substituting equation 13 for $\tilde{K}$ leads to

$$Q = \frac{1}{(2\pi)^{3\cdot(m-n+1)}} \cdot \int \exp\Bigg(-\frac{b^2}{4}\sum_{i=1}^{n}\Big[N_i \sum_{k=1}^{m-n+1}[c_{i,k}\cdot\vec{q_k}]^2\Big] + \Big[\sum_{k=1}^{m-n+1}\big[i\vec{q_k}\cdot\vec{r_{c,k}}\big]\Big]\Bigg)\Big[\prod_{k=1}^{m-n+1} d\vec{q_k}\Big] \tag{18}$$

Define the $3k$-dimensional vectors $\vec{q} = (q_{1,x}, q_{1,y}, q_{1,z}, ..., q_{k,z})^\top$ and $\vec{r_c} = (r_{c,1,x}, q_{c,1,y}, q_{c,1,z}, ...r_{c,k,z})^\top$. This allows to write the partition sum into the following form

$$Q = \frac{1}{(2\pi)^{3\cdot(m-n+1)}} \cdot \int \exp\left(-\frac{1}{2}\vec{q}^\top \cdot \mathbf{A} \cdot \vec{q} + \vec{q}^\top \cdot i\vec{r_c}\right)\Big[\prod_{k=1}^{m-n+1} d\vec{q_k}\Big] \tag{19}$$

In the appendix it is discussed how this integral can be computed. The end-result is given by

$$Q = \frac{1}{(2\pi)^{3\cdot(m-n+1)}} \cdot \left(\frac{(2\pi)^{m-n+1}}{Det(\mathbf{A})}\right)^{3/2} \exp\left(-\frac{1}{2}\vec{r_c}^\top \mathbf{A^{-1}} \vec{r_c}\right) \tag{20}$$

The result
This result allows us to find the partition sum of any network by finding the matrix $\mathbf{A}$ and the vector $\vec{r_c}$, which takes negligible computation time, inverting $\mathbf{A}$, which will prove to be the most computation-heavy step, and substituting the results into equation 20.

3.2.4 Free Energy

Now that we can compute the partition function of a given network, we can easily compute the free energy F.

$$F = -k_B T \cdot \ln Q \tag{21}$$

A system without constraints has partition sum $Q = 1$ and thus free energy $F = 0$, providing a suitable zero-energy reference point.

3.3 The effect of stress

The next step will be to bring stress into the system. This will be done by taking the dimensions of the periodic box around the network and varying them. There are a number of different ways to do this, for example

- Fix 2 dimensions, vary the third
- Vary 1 dimension explicitly, have the other two change in such a way the volume stays constant
- Vary all three dimensions at the same pace

The second possibility is the most realistic model for one-dimensional stretch, but is more difficult to compute then the other two. Therefore, we will instead study the third possibility, a model for stretch in all directions. It should be noted that both the first and the third option are specific forms of the more general

$$\begin{cases} X = X_0 + c_X \cdot f(t) \\ Y = Y_0 + c_Y \cdot f(t) \\ Z = Z_0 + c_Z \cdot f(t) \end{cases} \tag{22}$$

Computation-wise, they are fundamentally identical, but for simplicity's sake only the periodic cube with $X = Y = Z$ will be discussed.

3.3.1 The free energy

Recall equation 20. The matrix $\mathbf{A}$ depends only on the structure of the network and will not change when the system is stretched, which makes the term before the exponent a constant. The vector $\vec{r_c}$ on the other hand scales linearly with box size X. The interior of the exponent therefore scales quadratically with X. The free energy, which was defined as a constant multiplied with the logarithm of the partition function, will therefore be composed of a constant and a quadratic term.

3.3.2 Individual strands

The effects of stress can also be studied at strand-level. Stretching the network means stretching the individual bridges. After enough stress, bridges will start breaking and eventually the network falls apart.

Break condition
In section 2.2 we derived the break condition

$$\vec{R}^2 \geq \frac{f_t^2 \cdot (Nb^2)^2}{(3k_BT)^2} \tag{23}$$

Defining $f_{threshold} = \frac{f_t}{3k_BT}$ for simplicity and taking the expected value for $\vec{R}^2$, we derive the break condition

$$\left\langle r_x^2 + r_y^2 + r_z^3 \right\rangle \geq f_{threshold}^2 \cdot (Nb^2)^2 \tag{24}$$

Expected length
Now that we have the partition function, the average end-to-end distance $\left\langle r_x^2 + r_y^2 + r_z^3 \right\rangle$ can be computed fairly easily. First, compute the partition function of the network. Now, add the constraint $r_i = (x, y, z)^\top$, with r_i the strand we need the average end-to-end distance from, and re-compute the partition function. Divide this by the partition function of the entire system, which results in the probability distribution $P_i(x, y, z)$ for ri. This can be used for a number of things, but for now the most interesting use is computing the average end-to-end distance.

$$\left\langle r_{i,x}^2 + r_{i,y}^2 + r_{i,z}^3 \right\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x^2 + y^2 + z^2) \cdot P_i(x, y, z)dxdydz \tag{25}$$

In general, this does not result in a number, but a function depending on the stretched system size X. Solving equation 24 for X leads to a breaking point.

3.4 The initial simulation

We now have all the pieces required to write a simulation of the network, which will be discussed in this section. This initial simulation is quite inefficient, which is why it was improved afterwards. This newer simulation will be described in the next section, but to understand it, this one needs to be introduced first.

The input
As described in section 3.2.1, we need a list of constraints and a list of bridge lengths. By row-reducing the matrix containing the list of constraints, linearly dependent constraints are removed. The Kuhn length is set to 1 as it is incorporated in the bridge lengths. $f_{threshold}$ depends on the chosen material. Y and Z are set to be equal to X, which remains undefined. Initial system size X_0 is chosen.

3.4.1 The partition function

This part of the simulation was written by dr. Ellenbroek for two dimensions. It remains almost identical for three dimensions.
Recall the following equations from section 3.2.3

$$Q = \frac{1}{(2\pi)^{3\cdot(m-n+1)}} \cdot \int \exp\left(-\frac{b^2}{4} \sum_{i=1}^{n} \left[N_i \sum_{k=1}^{m-n+1} [c_{i,k} \cdot \vec{q_k}]^2 \right] + \left[\sum_{k=1}^{m-n+1} \left[i\vec{q_k} \cdot \vec{r_{c,k}} \right] \right] \right) \left[\prod_{k=1}^{m-n+1} d\vec{q_k} \right] \tag{26}$$

$$Q = \frac{1}{(2\pi)^{3\cdot(m-n+1)}} \cdot \left(\frac{(2\pi)^{m-n+1}}{Det(\mathbf{A})} \right)^{3/2} \exp\left(-\frac{1}{2} \vec{r_c}^{\top} \mathbf{A^{-1}} \vec{r_c}\right) \tag{27}$$

First, the interior of the integral from equation 26 is extracted from the constraints and lengths. The matrix $\mathbf{A}$ from equation 27 is now the Hessian matrix of the logarithm of this interior with respect to the vector $\vec{q}$. Dividing this interior by $\exp\left(-\frac{1}{2}\vec{q}^{\top} \cdot \mathbf{A} \cdot \vec{q}\right)$ and taking the gradient of its logarithm with respect to $\vec{q}$ leads to the vector r_c.
Now compute the determinant and invert the matrix and substitute the results in equation 27. The free energy of the network can now also be computed by taking the logarithm and multiplying with the Boltzmann constant k_B and temperature T.

3.4.2 The system check

In section 3.3.2 we discussed how to find the breaking point of a given strand by adding a constraint, finding the new partition function in the same way discussed in section 3.4.1, using this to find the probability distribution, integrating this to find the average end-to-end distance as a function of X and solving equation 24. In the case that there is no solution for this equation, the breaking point is set to infinity. This process is repeated for every bridge in the system, leading to a list of breaking points. The lowest breaking point and the corresponding strand are saved. In case of a draw, a strand is chosen at random since in reality these lengths vary over time so one of them is longer than the other at any point.

3.4.3 Breaking a strand

The effect of a breaking strand on the network is the removal of a single loop. By row-reducing the set of constraints with the breaking strand as first variable, the set of constraints rewritten in such a way the breaking strand appears exactly once.

The constraint it appears in is removed and the rest of them form the new set.

3.4.4 The iteration

We now have a new network. The partition function is computed again. This corresponds to giving the network time to find a new equilibrium position. We not iterate the process, by performing the system check again, breaking a new strand etc. etc. until the breaking point of each remaining strand is infinity. In practice, this happens when the X-dependence is eliminated from the system.

3.4.5 The result

A flow chart of the complete simulation is as follows

The Initial Simulation
1. Find constraints and bridge lengths of the network
2. Find matrix A and vector r_c using these constraints
3. Derive the partition function and corresponding free energy. If the latter has no dependence on box dimension X, end simulation
4. Start with strand r_i
a) Add the constraint r_i = (x,y,z)
b) Find the new matrix A_i and vector $r_{c,i}$
c) Find the new partition function
d) Use this to find $<r_i^2>$
e) Solve the breaking condition with respect to box dimension X for a breaking point
Repeat for all strands
5. Choose the strand that breaks first, rewrite the set of constraints such that this strand appears exactly once
6. Remove this constraint, store the breaking point and free energy
7. Go back to step 2 with these new constraints and repeat

Figure 4: A flow chart of the initial simulation

This process yields the following end results

- A network that does not cross the periodic borders anymore
- A list of broken strands and the order in which they break
- A piece-wise free energy function

The latter is defined as the free energy of the initial network for X from X_0 to the first breaking point, the free energy of the network after the first strand breaks from breaking point one to breaking point two, etc. At each such interval, it is quadratic. After each interval, it takes a drop and its quadratic coefficient goes down. This drop in energy is the dissipation energy and the lower coefficient manifests itself a a less tough network in practice.

3.5 The final simulation

The previously discussed simulation worked, but was quite slow and thus not scaleable towards large networks. The problem was that at each iteration of the system for each strand r_i a new matrix $\mathbf{A_i}$ was defined and inverted, a quite time-intensive step, since it is a $3 \cdot k + 3$ dimensional matrix, with k the number of constraints, which for the quite small networks discussed in the results section already led to a matrix with over 100 rows and columns. It was simply unfeasible to perform this inversion step many times per iteration.

3.5.1 The solution

Ideally, this inversion should only have to be performed a single time per iteration. The matrix $\mathbf{A_i}$ is almost identical for each strand, with only the final three rows and columns being different. The idea behind the solution was to, instead of each time entering these last three rows and then inverting, to place indicators S_i in these spots, "almost" invert the matrix, save this result, then simply fill in the indicators for each strand and finish the inversion.

3.5.2 The interim result

This was achieved by taking $\mathbf{A}$ with the added indicator rows, taking an identity matrix of the same size and attaching its columns to the $\mathbf{A}$. If one were to row-reduce this matrix the result would be an identity matrix with the added columns of $\mathbf{A}^{-1}$. This is not a very useful result however, since the large number of unknown indicators slow down the row-reduction process, because throughout the way most spots in the matrix contain complicated equations with many of the indicators.
Instead, the bottom three rows are removed before row-reducing. Now, the row-reduction steps only have to deal with the indicators on the final columns, significantly speeding up the process. The result is a $3k \times 3k$ identity matrix, followed by

three columns containing cross-terms with indicators, and then $3k+3$ more columns containing the interim result.

This step can be performed about three times faster through the following trick. Because of the way $\mathbf{A}$ was constructed, row one, four, seven etc. are all of the form $(a_1, 0, 0, a_2, 0, 0...)$. This is because for example $\mathbf{A_{12}}$ is constructed by taking the partial derivative with respect to $q_{1,x}$, leaving an x-dependent term and then with respect to $q_{1,y}$, yielding zero. Furthermore, row two and three are shifted versions of row one, row five and six are shifted versions of row four and so forth.
Therefore, the row reduction can be done by defining the $k \times k$ matrix $\mathbf{A}^*$ with

$$\mathbf{A^*_{i,j}} = \mathbf{A_{1+3i,1+3j}} \tag{28}$$

Perform the half-inversion on this matrix instead, then place two zero columns after each column in the result and place two rows below each row, identical but shifted one and two to the right. This result is the same as the one obtained by row-reducing the $3k \times (6k+3)$ matrix, but is obtained faster.

3.5.3 Filling in the indicators

The indicator columns of the original matrix looked as follows.

S_1	0	0
0	S_1	0
0	0	S_1
$\vdots$	$\vdots$	$\vdots$
0	S_{k+1}	0
0	0	S_{k+1}

Table 1: The indicator columns

Look back at the way $\mathbf{A_i}$ was constructed, by finding the Hessian of the logarithm of the interior of equation 26. Since it is the only part containing any quadratic terms, it suffices to take the Hessian of

$$-\frac{b^2}{4}\sum_{j=1}^{n}\left[N_i\sum_{l=1}^{k}[c_{j,l}\cdot\vec{q}_l]^2\right] \tag{29}$$

where $c_{j,l}$ was the derivative of constraint l with respect to r_j and b has been taken equal to 1.

S_{k+1} is now simply $\sum_{j=1}^{n}\left[\frac{N_j}{2}\cdot c_{j,k+1}^2\right]$. Constraint $k+1$ is the newly added constraint $\vec{r_i}=(x,y,z)$, which results in $S_{k+1}=N_i/2$.

For $m\in 1,...,k$, $S_m=\sum_{j=1}^{n}\left[\frac{N_j}{2}\cdot c_{j,k+1}\cdot c_{j,m}\right]$. Once again, given what we know of constraint $k+1$, this results in $S_m=c_{i,m}\cdot N_i/2$.

3.5.4 The final simulation

A diagram of the final simulation is given here, similar to figure 4 for the initial simulation

The Improved Simulation
1. Find constraints and bridge lengths of the network
2. Find matrix A and vector r_c using these constraints. Also find the Interim Matrix
3. Derive the partition function and corresponding free energy. If the latter has no dependence on box dimension X, end simulation
4. Start with strand r_i
a) Fill in the indicators in the Interim Matrix
b) Finish row reduction
c) Find the new partition function
d) Use this to find $\langle r_i^2 \rangle$
e) Solve the breaking condition with respect to box dimension X for a breaking point
Repeat for all strands
5. Choose the strand that breaks first, rewrite the set of constraints such that this strand appears exactly once
6. Remove this constraint, store the breaking point and free energy
7. Go back to step 2 with these new constraints and repeat

Figure 5: A diagram of the initial simulation

The final simulation is now nearly identical to the previous one, but at each iteration step we not only determine the complete matrix $\mathbf{A}$, but also the interim matrix. Then, instead of adding a constraint and inverting a large matrix $\mathbf{A_i}$ for each strand, we take the interim matrix and fill in the indicators. We then finish the row reduction process to find $\mathbf{A_i^{-1}}$. $\mathbf{A_i}$ and by extent $Det(\mathbf{A_i})$ can also be obtained easily from $\mathbf{A}$ and the indicators. Finally, the linear vector $\vec{r_{c,i}}$ is simply the vector $\vec{r_c}$ of the network, with (x,y,z) added at the end.
These substitutions take up a trivial amount of computation time, making the total time spent per iteration roughly equal to the time required to find the breaking

point of two strands in the old simulation.

4 Results

Whilst the main goal of this project was to develop the method, a few networks were analyzed to show how the simulation works. These networks were provided by A. Bose, who created them using LAMMPS software. First, a set of random walkers in a periodic cube create a network, and then a second set of walkers creates a second generation network. Cross-links exist between the networks and also within these networks. These crosslinking points and the bridges in between were analyzed in the way described in section 3.

4.1 The networks

4.1.1 The generation

To get some form of comparison, the two samples were given slightly different attributes. The dimension of the initial cube was set to 7 for both, so the walkers moved on a $7 \times 7 \times 7$ grid. For both networks, there were 2 walkers in the first generation and 5 in the second. The first sample has identical parameters for the first and second generation. It therefore models a network similar to a single generation network. For the second network, the first generation walkers were set to have step size 2. The result was two samples with a similar amount of cross-linking points, bonds and bridges. The different step size in the second network leads to a network that models a double network polymer for which one of the networks is stretched before creating the polymer.

4.1.2 The input

After putting the network through the script discussed in section 3.2.2, they can be compared a bit. The first network, where both generations have step size 1, has 23 constraints and 54 bridges which define the network. The second network has 35 constraints and 75 bridges. When comparing these, the fact that the second network is larger will have to be taken into account.
The initial free energy of the networks are $148.6kT$ and $313.5kT$ respectively, with kT the Boltzmann constant multiplied with the temperature. The quadratic coefficients in the free energy function, see section 3.3.1, are $0.543kT/x$ and $2.49kT/x$. Here, x is the length in between two points in the periodic cube.

4.2 The breaking process

The simulation was run, with the Kuhn length b set to 1 and $f_{threshold}$ to $10 \cdot x^{-2}$. The first network broke down in 4 steps, the last of these happening at a stretch of $50.21 \cdot x$. After these 4 broken strands a network remained, but didn't cross the periodic borders anymore and thus collapsed. The second network broke down in a over 10 steps, the final one happening at a stretch of $123.35 \cdot x$. Graphs of the free energy are shown below.

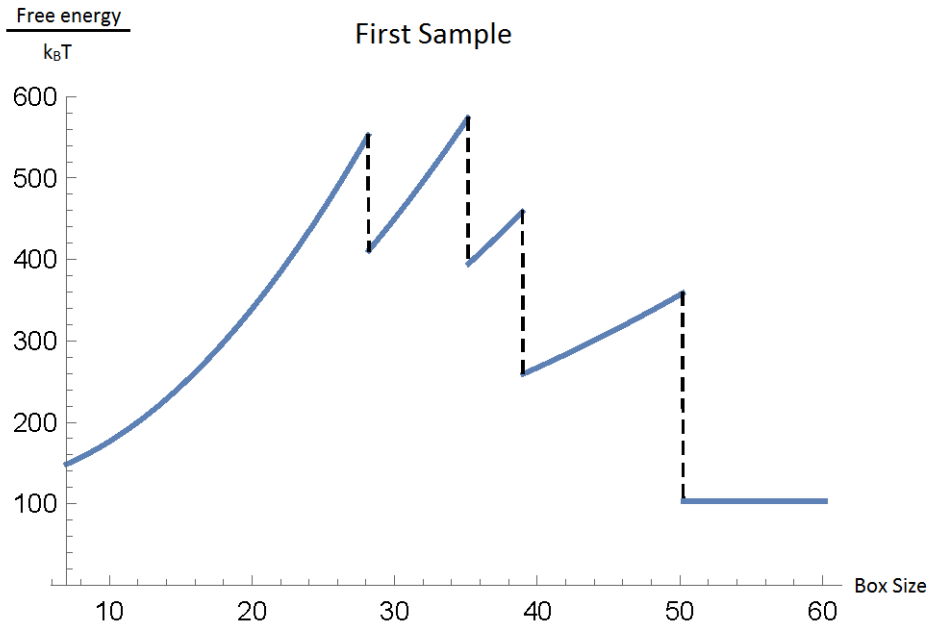


Figure 6: Free energy whilst breaking the first network

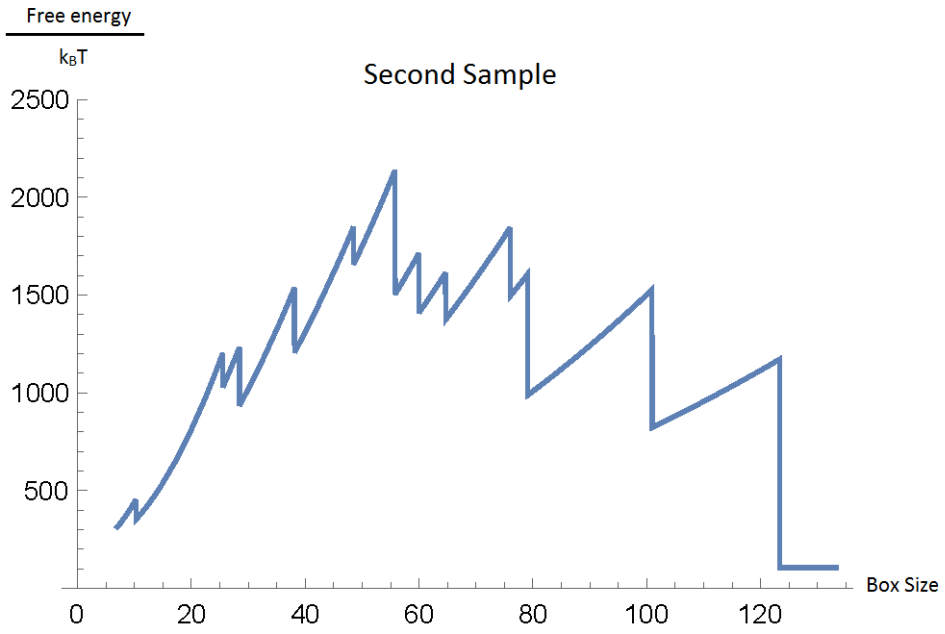
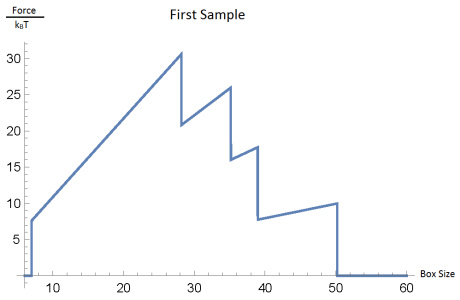


Figure 7: Free energy whilst breaking the second network

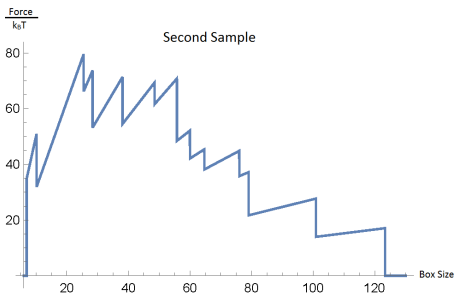
The drops in free energy at each of the breaking points is the energy dissipation corresponding to that break. The total energy dissipation is given by 777·kT and 5017·kT for the two networks respectively.

4.2.1 Forces

By taking the derivative of the free energy to the stretch, the relation between force and stretch can be found.



(a) The first sample



(b) The second sample

Physically, these graphs can be used to find the results of a constant-force experiment. For any given force on the vertical axis the resulting stretch is given by the left-most point where this force is found. The highest peak represents the force where a constant-force experiment destroys the network completely.

5 Conclusion

Recall that the main goal of this project was to develop a method capable of analysing the breaking process of a double network. A secondary goal was to apply this method to a few examples in order to draw some comparisons. This section will therefore briefly interpret the results given in the previous section before going into potential improvements on the method.

5.1 On the networks

The results from section 4.2 show the second network is significantly stronger. More force needs to be applied to stretch the network, more strands need to be broken before the network fails and the total stretch before the network breaks down is over two times as high.
The free energy of the second network for any given stretch is significantly higher, and the peak free energy is nearly four times as high. Furthermore, more energy dissipates during the network breakdown.
Part of this has to be attributed to the larger amount of bonds in the second network, approximately 20% more. Even so, the difference is striking. It was speculated earlier in this report, based on previous research, that this can be caused by the second, more stretched out generation of the second network breaking to a large extent before the first one, thus preserving the overall structure longer. This conclusion is also supported by the force graphs.
Whether this is actually the case can be verified by checking to which generation each broken strand belongs to. This is not currently possible with this code.

5.2 On the simulation

Though results can be found, there are a few details where the simulation can be improved.
Bridges connecting crosslinking points to themselves are not recognized and not properly analyzed. For bridges that do not cross the periodic borders this is not an issue, as these bridges never break. Bridges that do cross these borders have to be found and accounted for manually.
In terms of computation speed, a network of about 80 strands could be analyzed without much issue in a few hours on a not too powerful laptop. A few networks with over 100 strands took too long to analyze, though with better hardware better

results can of course be found.
There are some other small problems, which are given in the next paragraph.

5.3 Extensions

There is quite a large amount of possible extensions for this model, both fixes and "true" extensions.
As mentioned in section 5.2, bridges that connect cross-linking points to themselves have to be identified. Since these bridges don't interfere with the other bridges, their breaking point, which is infinite for such bridges that don't cross the periodic border, can be identified at the start, which eliminates unnecessary computations later on.
For results interpretation, it can be relevant to know to what generation each strand belongs to. A simple extension to the R script can provide this function. This would also allow re-scaling of a generation without the need to generate a completely new network.
Currently, the only stretch being studied is equal stretch in all directions. The model also works for stretch where in each direction the stretch scales linearly with time. A number of different stretches can be interesting subjects to study as well, for example stretch where the total volume is kept constant.
A fundamentally different example would be stretch where not the stretch in meters is increased over time, but one where a constant force is applied to the material and the force on individual strands follows from that.

5.3.1 Variance

This model is built upon the assumption that the networks stabilize completely before stretch is continued. Furthermore, it is assumed that the strand which is weakest on average breaks first. In reality, the structure of the network is continually fluctuating. To take this into account, the variance in strand length can be incorporated.
Since the probability distribution of each strand in each direction is already being derived in order to find the expected length, this distribution can be used without much additional computation to find more properties of each strand.
One possible way to do this is to break a strand if its expected length and a fraction, say 10% of the standard deviation of this length added together add up to its

breaking point.

5.4 Discussion

First of all, whilst it is nice to see the results on the two studied networks check out with the expected behaviour, this does not suffice as proper evidence in favour of this hypothesis. Two networks, which were both generated through a random process and can differ in a large number of aspects, do not form nearly a large enough sample size. These results were merely included to illustrate the simulation. Now that the script is finished, more networks can be analyzed in order to reach a well founded conclusion.
Still, the apparent validity of these results are an indication the simulation works correctly. Cross-checking these results with different methods of analysis or possibly even real experiments should go a long way in terms of testing the model. Unfortunately, there was no time in this project left to perform these tests and thus the sensibility of the results is the only indication of correctness currently available.

Appendix

Gaussian integrals

In this section, the solution to a multivariate Gaussian integral will be discussed. The goal is to find a solution to the following integral.

$$\int \exp\left(-\frac{1}{2}\vec{q}^{\top}\cdot\mathbf{A}\cdot\vec{q}+\vec{q}^{\top}\cdot\vec{b}\right)\Big[\prod_{k=1}^{n}d\vec{q_k}\Big] \tag{30}$$

$\mathbf{A}$ is a symmetric nxn matrix, $\vec{b}$ an n-dimensional vector.

The quadratic part

We start by deriving a solution to

$$\int \exp\left(-\frac{1}{2}\vec{q}^{\top}\cdot\mathbf{A}\cdot\vec{q}\right)\Big[\prod_{k=1}^{n}d\vec{q_k}\Big] \tag{31}$$

We know

$$\int \exp\left(-\frac{1}{2}a\cdot x^2\right)dx=\sqrt{\frac{2\pi}{a}} \tag{32}$$

And by extent

$$\int \exp\left(-\frac{1}{2}\sum_{i=1}^{n}a_i\vec{q_i}^2\right)\Big[\prod_{k=1}^{n}d\vec{q_k}\Big]=\frac{(2\pi)^{n/2}}{\sqrt{\prod_{i=1}^{n}a_i}} \tag{33}$$

By diagonalizing the matrix $\mathbf{A}$ to the form $\mathbf{A}=\mathbf{S}^{\top}\cdot\mathbf{\Lambda}\cdot\mathbf{S}$, which is possible due to the symmetry of A, equation 31 can be rewritten as

$$\int \exp\left(-\frac{1}{2}\vec{q}^{\top}\cdot\mathbf{S}^{\top}\cdot\mathbf{\Lambda}\cdot\mathbf{S}\cdot\vec{q}\right)\Big[\prod_{k=1}^{n}d\vec{q_k}\Big] \tag{34}$$

Now, defining $\vec{x}=\mathbf{S}\cdot\vec{q}$ yields

$$\int \exp\left(-\frac{1}{2}\vec{x}^{\top}\cdot\mathbf{\Lambda}\cdot\vec{x}\right)\Big[\prod_{k=1}^{n}d\vec{x_k}\Big]\cdot\frac{1}{|Det(\mathbf{S})|} \tag{35}$$

$\mathbf{\Lambda}$ is a diagonal matrix, so this equation is of the form of equation 33, so this yields

$$\frac{1}{|Det(\mathbf{S})|}\cdot\frac{(2\pi)^{n/2}}{\sqrt{\prod_{i=1}^{n}\Lambda_{ii}}}=\frac{(2\pi)^{n/2}}{\sqrt{Det(\Lambda)\cdot Det(\mathbf{S})^2}} \tag{36}$$

Using $\mathbf{A} = \mathbf{S}^\top \cdot \mathbf{\Lambda} \cdot \mathbf{S}$, we know $Det(\mathbf{A}) = Det(\Lambda) \cdot Det(\mathbf{S})^2$, so

$$\int \exp\left(-\frac{1}{2}\vec{q}^\top \cdot \mathbf{A} \cdot \vec{q}\right)\Big[\prod_{k=1}^{n} d\vec{q_k}\Big] = \frac{(2\pi)^{n/2}}{\sqrt{Det(A)}} \tag{37}$$

for any symmetric A.

The linear part

Going back to our original problem, we have

$$\int \exp\left(-\frac{1}{2}\vec{q}^\top \cdot \mathbf{A} \cdot \vec{q} + \vec{q}^\top \cdot \vec{b}\right)\Big[\prod_{k=1}^{n} d\vec{q_k}\Big] \tag{38}$$

Which we can now rewrite as

$$\int \exp\left(-\frac{1}{2}\vec{x}^\top \cdot \mathbf{\Lambda} \cdot \vec{x} + \vec{x}^\top \cdot \mathbf{S}^{-1\top} \cdot \vec{b}\right)\Big[\prod_{k=1}^{n} d\vec{x_k}\Big] \cdot \frac{1}{|Det(\mathbf{S})|} \tag{39}$$

This is equal to

$$\int \exp\left(-\frac{1}{2}\sum_{i=1}^{n}\left[\mathbf{\Lambda}_{ii} \cdot x_i^2 - 2(\mathbf{S}^{-1\top} \cdot \vec{b})_i\right) \cdot x_i\right]\Big[\prod_{k=1}^{n} d\vec{x_k}\Big] \cdot \frac{1}{|Det(\mathbf{S})|} \tag{40}$$

Completing the squares and using $\mathbf{S}^{-1} = \mathbf{S}^\top$ yields

$$\int \exp\left(-\frac{1}{2}\sum_{i=1}^{n}\left[\mathbf{\Lambda}_{ii}\left(x_i - \frac{(\mathbf{S}\cdot\vec{b})_i}{\mathbf{\Lambda}_{ii}}\right)^2 - \left(\frac{(\mathbf{S}\cdot\vec{b})_i}{\mathbf{\Lambda}_{ii}}\right)^2\right]\Big[\prod_{k=1}^{n} d\vec{x_k}\Big] \cdot \frac{1}{|Det(\mathbf{S})|} \tag{41}$$

The part with the quadratic x-part we are now able to solve, the other square is a constant, so can be taken out of the integral. The above equation therefore reduces to

$$\frac{(2\pi)^{n/2}}{\sqrt{Det(A)}} \cdot \exp\left(\frac{1}{2}\sum_{i=1}^{n}\left(\frac{(\mathbf{S}\cdot\vec{b})_i}{\mathbf{\Lambda}_{ii}}\right)^2\right) \tag{42}$$

Rewrite this as

$$\frac{(2\pi)^{n/2}}{\sqrt{Det(A)}} \cdot \exp\left(\frac{1}{2}\sum_{i=1}^{n}\left(\vec{b}^\top \cdot \mathbf{S}_i^\top \cdot \frac{1}{\mathbf{\Lambda_{ii}}} \cdot \mathbf{S}_i \cdot \vec{b}\right)\right) \tag{43}$$

Which yields the final result

$$\frac{(2\pi)^{n/2}}{\sqrt{Det(A)}} \cdot \exp\left(\frac{1}{2}\vec{b}^\top \cdot \mathbf{A}^{-1} \cdot \vec{b}\right) \tag{44}$$

References

[1] R Everaers and K Kremer. Modeling of polymer networks. 2001.

[2] Etienne Ducrot, Yulan Chen, Markus Bulters, Rint P Sijbesma, and Costantino Creton. Toughening elastomers with sacrificial bonds and watching them break. *Science*, 344(6180):186–189, 2014.

[3] Yuji Higuchi, Keisuke Saito, Takamasa Sakai, Jian Ping Gong, and Momoji Kubo. Fracture process of double-network gels by coarse-grained molecular dynamics simulation. *Macromolecules*, 51(8):3075–3087, 2018.

[4] Ecaterina Stela Dragan. Design and applications of interpenetrating polymer network hydrogels. a review. *Chemical Engineering Journal*, 243:572–590, 2014.

[5] Pierre Millereau, Etienne Ducrot, Jess M Clough, Meredith E Wiseman, Hugh R Brown, Rint P Sijbesma, and Costantino Creton. Mechanics of elastomeric molecular composites. *Proceedings of the National Academy of Sciences*, 115(37):9110–9115, 2018.

[6] Michael Rubinstein, Ralph H Colby, et al. *Polymer physics*, volume 23. Oxford university press New York, 2003.

[7] W. Ellenbroek. Exact partition function for gaussian networks, 2017.