Statistical Properties and Simulation of Cellulose

A Major Qualifying Project Submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the Degree
in Bachelors of Science in
PHYSICS
by

Robert Rinearson

July, 2019

Project Advisor:
Professor Izabela Stroe

Abstract

Cellulose is one of the most abundant compounds found in nature. However, the environmental conditions the substance is placed in is of utmost importance to the structure of the cellulose, and must be controlled to ensure optimal strength and cohesiveness of the compound. Here, we developed a simple statistical physics model to simulate how thermodynamic properties affect microscopic cellulose. Partition functions to represent the glucose molecules in cellulose are constructed as the basis of the simulation. Depending on temperature, the type of hydrogen bonds holding the cellulose together dynamically change, until the point where the temperature becomes high enough to rupture all hydrogen bonds. The Specific Heat calculated in the simulation agrees well with cellulose's experimental Specific Heat value. Though only a simple model, the physics behind the simulation was able to reproduce experimental data with high accuracy and provides insights into the microscopic behavior of cellulose.

Acknowledgments

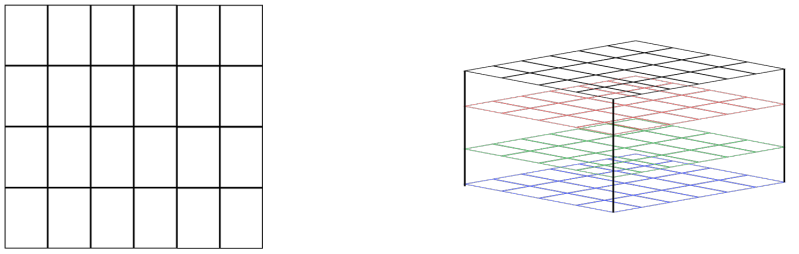
I would like to thank Dr. Hall, PhD for her unwavering support over the summer, partially expressed through delicious meals.

Importantly, I would like to give thanks my advisor, Professor Izabela Stroe, for her attentiveness to this project over the course of the summer.

Contents

1 Background

Cellulose is a compound found in many different substances, mostly because its abundance in organic plant cells. Cellulose grows to form microfibril which accumulate to form strands in substances such as cotton. Cotton is a cellulose substance that is able to withstand forces when woven without sheering and has tensile strength comparable to that of steel. Cotton is essential to the textile industry and the effects of thermodynamic properties on the substance are of great importance to the processing of the material. Many of the properties of cotton depend on pressure, temperature, and saturation of hydrogen-bonds in the substance, and dictate how the material is able to perform on the macro-scale; whether it is greatly rigid or not. The properties of cellulose that give it such great strength and rigidity lead to insights into the formation of rigid organic structures. The goal of this project is to derive a simple, yet meaningful model of cellulose through non-rigorous statistical physics and mathematics. The relatively simple, crystalline structure of the most abundant organic compound on Earth has gifted us with a multitude of uses the natural fibers cellulose creates. From the wood used in construction, to the cotton clothes in the closet, this compound permeates through the entire world. On the microscopic level, for this model, a sheet of cellulose will be modeled as a lattice, and is depicted in figure 1.



(a) lattice shape

(b) microfibril structure

Figure 1: Cellulose sheets stacking atop another.

Cellulose microfibrils are made from stacking multiple sheets atop each other. Microfibrils entangle en-mass to form the fine threads found in materials such as cotton.

Cotton used in the textile industry must be maintained at conditions which will not cause flaws in the final product, such as yarn and fabric. Throughout the processing of cotton, many precautions are taken on the macro scale, however, the cellulose that the cotton is produced from is rarely thought of. The importance of thermodynamics on the cellulose lattice is demonstrated on the

macro scale. The properties of cotton on the macro scale are affected by the sum of microscopic and atomic properties. When cotton grows, different pressures, temperatures, and humidity's affect the structure of the cotton fibers. Depending on the dryness of the soil, or how much the cotton was rained on, the structure of the cotton changes. The length of the fibers varies, the discoloration varies, the fineness of the fibers varies, as well as the uniformity of the fibers. All these properties play a large role in the processing of cotton into products. The properties of the natural cotton transfer into the properties of the spun cotton yarn which in turn transfers into the properties of woven products. Wrinkles in cotton clothing once they have been dried is caused by uneven distribution of saturation in the cotton, making that portion of the fabric shrink, that is why an iron uses steam to saturate the wrinkled garments to return them to their normal shape. In cotton yarn manufacturing, temperature and humidity are heavily controlled to ensure the maximum quality of product. Tensile strength of the cotton thread changes with the length of the fibers, as well as the saturation of hydrogen bonds, in addition to the cotton fibers becoming more or less permeable to air based on thermodynamic properties, which essentially defines how fine the fibers are. On the microscopic scale, microfibrils have variable lengths, largely depending availability of hydrogen bonds being able to form. The microfibrils have properties that, once they coalesce, show themselves in the properties of the cotton fibers.

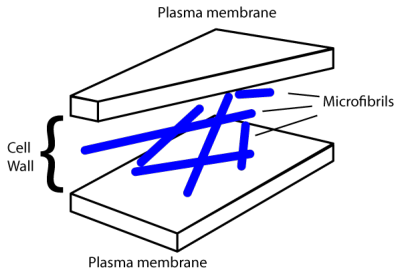


Figure 2: Microfibrils coalescing in a plant cell wall. Cotton is the most pure form in nature of cellulose, with 90 percent of the material being made up of these glucose chains.

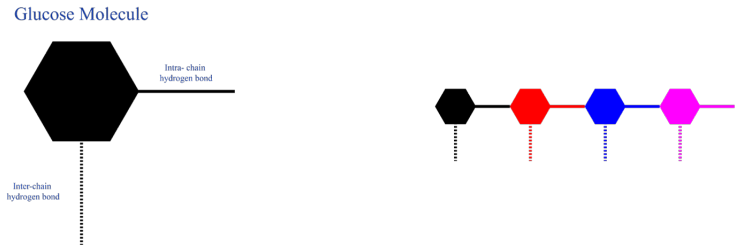
As temperature increases, cotton fibers begin to decompose and no longer represent a strand. This behavior is a result of microscopic changes in the material that cause a cascading reaction where the fiber can no longer maintain its structure. In contrast, certain temperatures and pressures allow for the most

tensile strength in a cotton fiber. Knowing the structure of a microfibril, which is formed by the cellulose lattice created by many glucose molecules, we will be able to predict microscopic properties of cellulose. Microscopic properties should be able to lend an explanation to the seemingly increased rigidity of the cellulose lattice.

2 Approach

2.1 Partition Function

This model is created from the sum of small components made of glucose molecules and their potential hydrogen bonds[1], with bonds that connect glucose to form chains being called "Intra-chain bonds" and the bonds that connect chains to other chains "Inter-chain bonds" (Intra and Inter bonds for short).



(a) Glucose Molecule (b) Cellulose chain

Figure 3: Glucose molecules form cellulose chains.

Glucose will play the roll of the networks vertices, while the hydrogen bonds will play the edges. The model is extended from one glucose molecule into multiple glucose molecules with intra-chain bonds, forming cellulose chains. Multiple chains are then connected to each other through inter-chain bonds, finally forming the sheet.

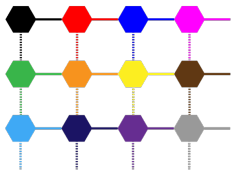


Figure 4: Cellulose Sheet.

All the hydrogen bonds involved have different states, strengths, and orientations due to the structure of the glucose molecule. This causes some complication for the model, diverging further from the simple DNA zipper problem [2]. The model must take into account the probability of each of these states occurring, both intra-bonds and inter-bonds[3]. This will be tackled by creating two separate partition functions, one for intra-bonds, and one for inter-bonds, and later taking the product in order to simplify the process.

Intra-bonds have 8 unique states, 6 different hydrogen bond orientations(2 single bonds, 2 bifurcated bonds, and 3 combinations)[4] and has 1 state where there is no hydrogen bond, with entropy being the only factor. From this, the partition function for the intra-bonds can be constructed as shown in equation 1 below:

$$Z_A = e^s + e^{\epsilon_a} + e^{\epsilon_b} + e^{\epsilon_c} + e^{\epsilon_d} + e^{\epsilon_a+\epsilon_d} + e^{\epsilon_b+\epsilon_d} + e^{\epsilon_c+\epsilon_d} \tag{1}$$

Inter-bonds have 5 unique states, with 4 different hydrogen bond orientations (2 single bonds, 1 bifurcated bond, and 1 combination)[4] and also has a free state. The inter-bond partition function is represented as

$$Z_E = e^s + e^{\epsilon_e} + e^{\epsilon_f} + e^{\epsilon_g} + e^{\epsilon_e+\epsilon_f} \tag{2}$$

Therefore, the complete partition function of a single glucose molecule would be where $G = Z_A Z_E$. Where the total number of states for a single glucose molecule is $\Omega = 40$. The partition function of a cellulose sheet with n glucose molecules would be approximately $S \approx G^n \approx {Z_A}^n {Z_E}^n$, where the total number of states for a sheet of n glucose molecules is $\Omega \approx 40^n$.

These are approximations to show the derivation of the simple sheet's partition function with extraneous hydrogen bonds on the right and bottom edges. The model takes into account the double counting hydrogen bonds on the right and bottom edges of the cellulose sheet. The model must depend on the length of the cellulose chain and the number of chains in a sheet to make it more accurate.

$$S = {Z_A}^{n-c} {Z_E}^{n-g} \tag{3}$$

The equation above is the basis for the simulation created in order to model the behavior of the cellulose lattice, and could model any lattice structure if the intra-bond and inter-bond partition functions are changed accordingly.

2.2 Entropy

In order to test the validity of the model, experimental data must be compared to the models predictions. The specific heat of a substance is a measurable quantity associated with the thermodynamic properties of the substance.

The Gibbs entropy, which calculates the entropy of systems with different probabilities for each of its states, can be calculated simply with the partition function for the lattice. The partition function allows for calculation of the

probabilities of different bonds, and with that probability the Gibbs entropy can be calculated. The Gibbs entropy can be written as:

$$S = -k_B \sum_i p_i ln\, p_i, \tag{4}$$

where the change in Gibbs entropy is simply equal to the difference of the Gibbs final entropy and the Gibbs initial entropy

The Gibbs entropy does not take into account the possible motion of the particles in the lattice, which is another source of entropy. In order to include the motion of the particles in the lattice, we will include in the simulation the motion of each particle as a system of four springs. This should account for the translational and vibrational energies. Intra and inter-bonds are modeled as springs with different spring constants depending on the strength of the bond.

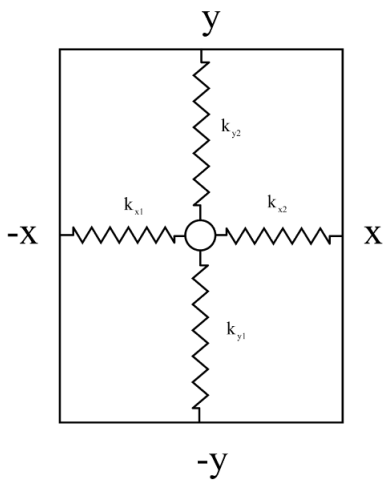


Figure 5: Hydrogen Bonds represented as springs with different spring constants in both x and y directions.

Random values of displacement from the center are chosen based on the strength of the springs constants, temperature, and a random value between -1 and 1.

$$x_{displacement} = \frac{1}{2} x_{max} P\Big(\frac{T}{T_{max}} + \frac{k_{x2} - k_{x1}}{k_{x2} + k_{x1}}\Big) \tag{5}$$

where the weighted spring constant factor is

$$\frac{k_{x2} - k_{x1}}{k_{x2} + k_{x1}} \tag{6}$$

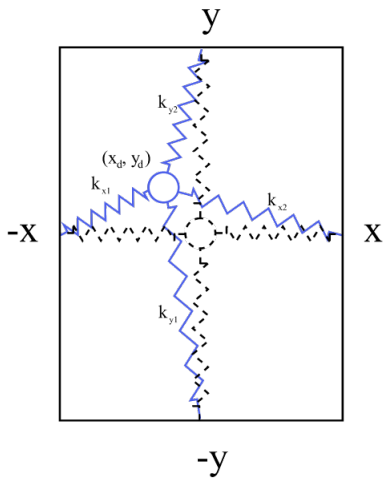


Figure 6: Representation of the glucose molecule displaced from the equilibrium position in the xy-plane. The elastic constants $k_{x1}, k_{x2}, k_{y1}, k_{y2}$ and the displacement (x, y) are indicated in the figure.

and the weighted temperature factor is

$$\frac{T}{T_{max}} \tag{7}$$

and P is the probability between -1 and 1. Similar equations are used for the y-axis. Using the Second Law of Thermodynamics for entropy

$$dS = \frac{\delta Q}{T} \tag{8}$$

and the First Law of Thermodynamics of a quasi-static system that states the following

$$dU = \delta Q - W \tag{9}$$

allow to calculate the change in heat. The equation can be manipulated such that we can express the work in terms of the force multiplied by the distance traveled.

$$\delta Q = dU + Fdr \tag{10}$$

Hooke's law states that the force enacted on an object by a spring is

$$F_n = k_n \Delta r_n \tag{11}$$

and the potential energy is

$$PE_n = \frac{1}{2} k_n {r_n}^2 \tag{12}$$

The spring constant value is calculated by taking a portion of the experimental hydrogen bond energy as the average vibrational energy. For instance, the force and internal energy for x_1 would be calculated by calculating the spring constant

$$\epsilon_{x1} = \frac{1}{x} \int_0^x PE dx \tag{13}$$

$$\epsilon_{x1} = \frac{1}{6} \frac{k_{k1} x^3}{x} \tag{14}$$

$$k_{k1} = \frac{6\epsilon_{x1}}{x^2} \tag{15}$$

and calculating the distance the spring has been stretched

$$\Delta r_{x1} = \sqrt{(x + x_d)^2 + {y_d}^2} - x \tag{16}$$

in order to find the force and potential energy

$$F_{x1} = k_{k1} \Delta r_{x1} \tag{17}$$

$$dU_{x1} = \frac{1}{2} k_{k1} {\Delta r_{x1}}^2 \tag{18}$$

For this model, the kinetic energy will be approximated as zero due to its generally small contribution to the internal energy of a crystalline microscopic system. Substituting these values into the first law equation leaves

$$\delta Q = \sum U_n + r \sum F_n, \tag{19}$$

per glucose molecule. The entropy of a single glucose molecule placed into the physical constraints of the cellulose lattice should lend a clue to the nature of the cellulose lattice and help to extrapolate a bigger picture. In addition from the total calculated entropy, a prediction of the specific heat can be made. The specific heat of the model can be compared to experimental data of the specific heat and heat capacity of cellulose.

$$dS = C \frac{dt}{T} \ (Heat\, Capacity), \tag{20}$$

$$C_p = \frac{C}{m} \ (Specific\, Heat) \tag{21}$$

where C is the Heat Capacity and C_p the Specific Heat, which are essentially derived the same, with slightly differing units based on a mass ratio.

2.3 Heat Capacity

The heat capacity of a substance can be calculated with

$$TdS = CdT \tag{22}$$

or

$$\delta Q = CdT \tag{23}$$

The integral can be taken across an initial temperature to a final temperature, and since the heat capacity is a function of temperature, it is the only variable.

$$\int_{Q_i}^{Q_f} \delta Q = C \int_{T_i}^{T_f} dT \tag{24}$$

$$Q_f - Q_i = C(T_f - T_i) \tag{25}$$

$$C = \frac{Q_f - Q_i}{T_f - T_i} \tag{26}$$

Where, due to the quantized nature of the computer simulation,

$$Q_i = \sum_{t=0}^{T_i} \delta Q_t \tag{27}$$

where

$$\delta Q = \sum U_n + r \sum F_n \tag{28}$$

In addition to the Gibbs entropy, through the previous equations for Heat Capacity and Specific Heat, the value of the Heat Capacity can be calculated, which is a function of temperature. From the Heat Capacity, the Specific Heat can be calculated, which can be compared to the experimental Specific Heat values of cellulose.

3 Simulation

3.1 Simulation Definitions

The method of testing this model is a computer simulation created in Python[5]. The first step in the program was to define the constants that are to be used throughout the simulation. The math, random, numpy, and pylab packages were used for ease of use[6]. The distance between glucose molecules in a chain is 5 angstrom while the distance between chains is defined as 8 angstrom [7]. The strength of a single hydrogen bond was defined as 5 kilocalories/mol [8].

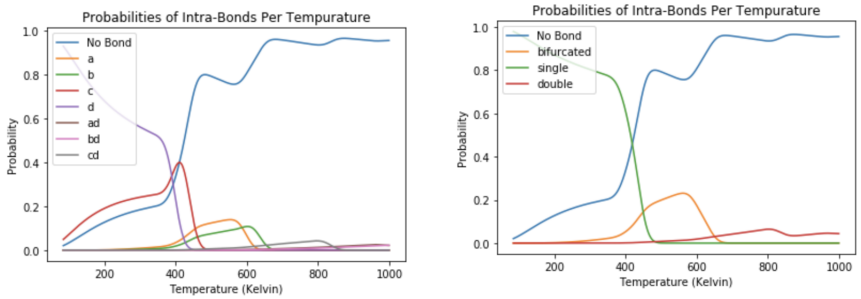
3.2 Computational Methods and Logic

The variation in the bond energies were used to calculate the values of the spring constants, and in the case of no hydrogen bond, the van der Waals force was used [9]. The first step is to define the partition function as a function of temperature. Once the partition function exists, it is possible to calculate an array of probabilities of each state that is defined. With a probability array, the probable energies are calculated in the simulation. When calculating entropy, the energies previously determined will be used. Adjustments can be made to the partition function due to the fact that one of the states exclusively depends on entropy (the free state). The Heat Capacity and Specific Heat can be approximated through a new method with the entropy, and taking into account the molar mass of a glucose molecule.

4 Results and Discussion

4.1 Theoretical Results and Figures

Due to the different types of hydrogen bonds, the glucose molecules are expected to have different "preferred" bonds at different temperatures, pressures, volume, chemical potential. The findings of the model predict these properties based on temperature, with a link to volume based on molecular distances between the molecules and the 3D nature of the microfibril structure.



(a) Probability of each Intra-bonds

(b) Probability of types of bonds

Figure 7: Probabilities of intra-bonds (as described in section 2) at varying temperatures. As expected, when the temperature increases the hydrogen bonds begin to rupture.

As the temperature rises, hydrogen bonds rupture depending on their energies. When the entropy state becomes more likely than a bond state, the bond state's probability decreases rapidly.

The Gibbs entropy behaves according to the probability calculations. The higher the temperature, the more likely fewer states are possible, until eventually the Gibbs entropy goes to zero with only one likely state.

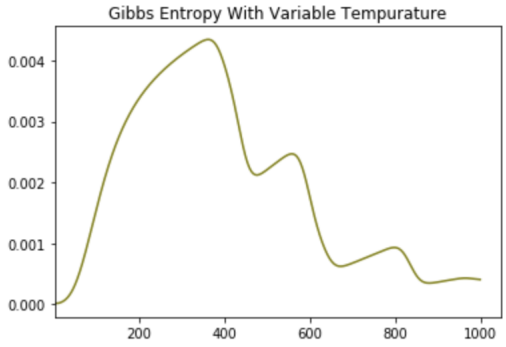


Figure 8: Gibbs Entropy decreases with increased temperature as less states become possible.

The thermodynamic change in entropy, however, is different. The change in entropy increases before leveling off and becoming, for the most part, constant.

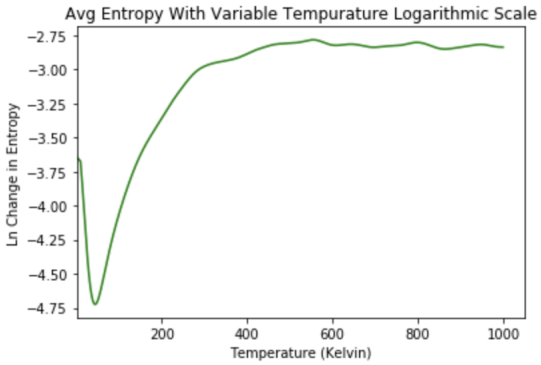


Figure 9: Average Change in Entropy represented on a logarithmic scale increases up to an average value of $23k_b$.

The model finds that the change in entropy on average is about $23k_b$ per glucose molecule with heat added per degree averaging at 0.0567 kilocalories per mole $^\circ K$. Taking into account that the molar mass of a glucose molecule is 0.180156 kilograms per mole, the Specific heat of the glucose that builds our cellulose can be calculated.

$$Q_{added} = mc_p\Delta T \tag{29}$$

$$0.0567[\frac{kcal}{mol\,^\circ K}] = 0.180156[\frac{kilograms}{mol}]c_p \tag{30}$$

$$c_p = \frac{0.0567}{0.180156}[\frac{kcal}{kilogram°K}] \tag{31}$$

$$c_p = 0.31498\ \frac{kcal}{kilogram°K} \tag{32}$$

$$c_p = 0.31498\ \frac{calories}{gram°C} \tag{33}$$

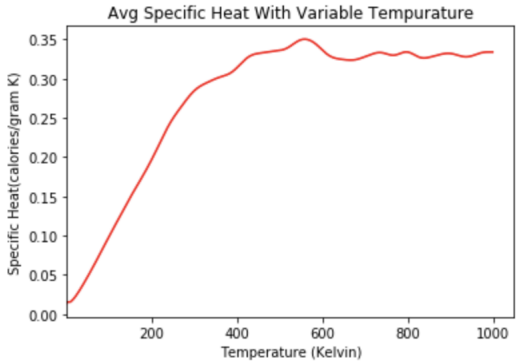


Figure 10: Specific Heat as a function of temperature. The theoretical value of 0.315 calories per gram is in good agreement with the experimental values [3].

4.2 Discussion

The model outputs $c_p = 0.315$ calories per gram Celcius, which, compared to cellulose' experimental data of between $c_p = 0.31$ and $c_p = 0.36$, is 87.5% accurate on the low end and 94% accurate if the average is taken. [10]

5 Model Short-comings and Future Analysis

The model presented here is a simplified one and inaccuracies are therefor expected. One short-coming of the model would be the lack of a third dimensional analysis in the model. The cellulose sheets do stack one on top of each other, but without the hydrogen bonds and instead utilizing the van der Waals forces completely. This means the glucose molecules have an entire other dimension to move around in and by then will increase entropy. Another short-coming of the model would be the simplified modeling of bonds as springs. While being somewhat accurate, on the extremes of bond contraction and rupturing temperatures it seems that this method would not tell all. In addition, the model was

also incomplete in the sense that it did not take into account the availability of hydrogen bonds and it was assumed the cellulose was saturated [11] [12].

Further analysis could be taken to measure the effect of other thermodynamic properties, such as pressure, volume, and chemical potential. Manipulation of multiple properties at once could shed light on interesting properties of the lattice structure. Some foundational work was completed for the other thermodynamic properties. While not fully developed, the simulation results are presented below.

5.1 Pressure

An analysis of the model's theoretical pressure would basically start with a quantified measurement of the pressure of each glucose molecule on the adjacent glucose molecules. Each glucose molecule has 4 "edges" with possibly unique energies for each one. A partition function of the unique state can be created from the energies in the original partition functions.

$$Z_A = e^s + e^{\epsilon_a} + e^{\epsilon_b} + e^{\epsilon_c} + e^{\epsilon_d} + e^{\epsilon_a+\epsilon_d} + e^{\epsilon_b+\epsilon_d} + e^{\epsilon_c+\epsilon_d} \tag{34}$$

$$Z_E = e^s + e^{\epsilon_e} + e^{\epsilon_f} + e^{\epsilon_g} + e^{\epsilon_e+\epsilon_f} \tag{35}$$

The statistical definition of pressure is defined

$$P = k_b T \frac{\partial ln Z}{\partial A}, \tag{36}$$

where P is the pressure, k is the Boltzmann constant , T is the temperature, Z is the partition function for the molecule, and A is the area. The addition of an adjacent molecule allows for a change in the system. The molecule has only one state. We can construct the partition function when realizing the total energy of the system is:

$$\epsilon_1 = \epsilon_A + \epsilon_B + \epsilon_C + \epsilon_D \tag{37}$$

thus

$$Z_1 = e^{\epsilon_1 \beta} \tag{38}$$

We can follow the same steps to get the energy and partition function for molecule 2.

$$\epsilon_2 = \epsilon_C + \epsilon_E + \epsilon_F + \epsilon_G \tag{39}$$

thus

$$Z_2 = e^{\epsilon_2 \beta} \tag{40}$$

Taking into account that the changes in the partition function are not infinitesimal and are quantized, due to the real number of vertices lets the pressure equation be transformed into:

$$P = k_b T \frac{\Delta ln Z}{\Delta A} \tag{41}$$

where,

$$\Delta A = \Delta x \Delta y \tag{42}$$

It can further be extrapolated to:

$$P_n = k_b T \frac{ln Z_n - ln Z_{n-1}}{\Delta x \Delta y} \tag{43}$$

Simplifying more, the equation becomes:

$$P_n = k_b T \frac{ln(\frac{Z_n}{Z_{n-1}})}{xy} \tag{44}$$

with

$$Z_n = e^{\epsilon_n \beta} \tag{45}$$

and

$$Z_{n-1} = e^{\epsilon_{n-1} \beta} \tag{46}$$

The equation can be further simplified. First, it must be noticed that the pressure only represents the area where either theres a change in x or a change in y, an infinitesimally small change in distance should look like $P\partial x = ...$ or $P\partial y = ...$ depending on the axes changing. But in the case of our quantized system, it looks like $P\Delta x = ...$ or $P\Delta y = ...$ In addition, a portion of the pressure function is

$$ln(Z_n) - ln(Z_{n-1}) \tag{47}$$

because the nature of a single state partition function, this simplifies to

$$\epsilon_n \beta - \epsilon_{n-1} \beta \tag{48}$$

where β is a common factor, thus

$$\frac{\epsilon_n - \epsilon_{n-1}}{k_b T} \tag{49}$$

If replaced back into the pressure equation,

$$P_n \Delta y = \frac{\epsilon_n - \epsilon_{n-1}}{\Delta x \Delta y} \tag{50}$$

$$P_{ny} = \frac{1}{\Delta x \Delta y} \frac{\Delta \epsilon_n}{\Delta y} \tag{51}$$

It becomes apparent that the equation is in the proper form.

$$P_{ny} = \frac{F_n}{\Delta A} \; or \; P_{nx} = \frac{F_n}{\Delta A} \tag{52}$$

Backtracking to an earlier form highlights the surprising lack of dependencies on temperature on the final equation, with only the change energy playing an important role in the equation with a constant x and y. However, the probabilities of the energy are still affected by the temperature.

$$P_{ny} = \frac{\Delta \epsilon_n}{\Delta x \Delta y^2} \tag{53}$$

The pressure of the molecule will be affected by molecules that are not adjacent to it, but the effect is inversely proportional to the distance from the square, and diminishes very rapidly. In this model we will only account for adjacent molecules.

Pressures will also be quantized. The amount of different pressures possible would be equal to

$$N_p = ((Number\,Of\;X\;States)^2 (Number\,Of\,Y\;States)^2)^2 \tag{54}$$

An observation of the equation aligns with the prediction of how the lattice would behave. At exceedingly low temperatures, where only one state is probable, or at exceedingly high temperatures where all states are equally probable (or in the case of cellulose, all hydrogen bonds are broken) the pressure is seen to reduce to:

$$where\; Z_n => Z_{n-1}, \tag{55}$$

$$P \propto ln \frac{Z_n}{Z_{n-1}} => ln(1) => 0 \tag{56}$$

or

$$where\; \epsilon_n => \epsilon_{n-1}, \tag{57}$$

$$P \propto \epsilon_n - \epsilon_{n-1} => 0 \tag{58}$$

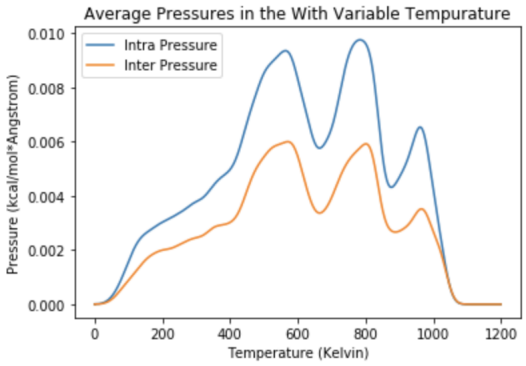


Figure 11: Average Pressure Between Glucose Molecules.

The pressure increases steadily with temperature until bonds begin to break. As certain bonds become preferable, the pressure fluctuates and upon all bonds rupturing and the entropy state is all that remains, the pressure returns to zero.

5.2 Crystallinity

The crystallinity is another interesting product of the lattice structure of cellulose [13] [14]. The crystallinity is dependent of the rigid structure and what percentage of the lattice is amorphous. The crystallinity can be roughly calculated by

$$N_i = Number\ of\ molecules\ with\ bonds\ considered\ tomakeitcrystalline \tag{59}$$

$$N_t = Number\ of\ molecules \tag{60}$$

$$\%C = \frac{N_i}{N_t} \tag{61}$$

or

$$\%C = 1 - P_a \tag{62}$$

where P_a is the probability of amorphous nature, and

$$P_a = \Pi_1^i\ P_{amorphous\ bond\ i} \tag{63}$$

For example, if an amorphous molecule were to be defined as having no intra-bonds, the crystallinity would look like,

$$P_a = {P_{amorphous\ intra}}^2 \tag{64}$$

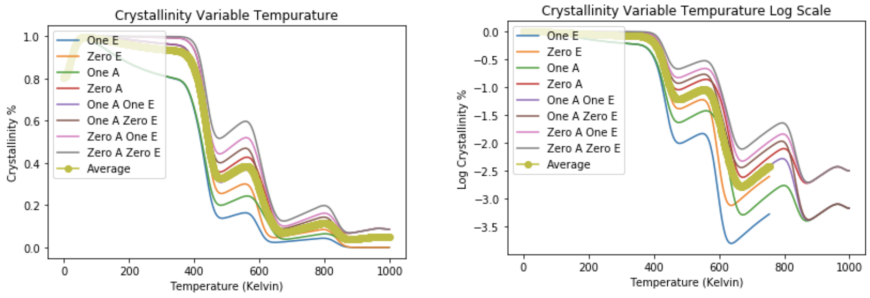
or an amorphous molecule with one missing intra-bond and one missing inter-bond,

$$P_a = P_{amorphous\ intra} \times P_{amorphous\ inter} \tag{65}$$

Thus,

$$\%C = 1 - P_a \tag{66}$$

While not able to fully analyze the model, an early analysis of the model's results are shown in the graphs below.



(a) Normal Scale

(b) Log Scale

Figure 12: Crystallinity with variable temperature.

A formal definition of the crystallinity still needs to be fully developed for the model, but conceptually the model above would be a good place to start.

6 Conclusion

The simple statistical model of the cellulose lattice with variable temperatures was able to predict the physical properties of the compound quite accurately. The relatively simple use of bond energies and entropy calculations are in good agreement with the experimental data. There is much to expand on this model, but a strong foundation has been laid for further research into the cellulose lattice model. The model has shown to be accurate in predicting and modeling some of the physical interactions of cellulose with thermodynamic properties. Limiting factors to this model include the physical nature of springs in contrast to the actual nature of inter-molecular bonds, the 3D nature of the actual material, along with a lack of interactivity between multiple thermodynamic properties. Overestimation of the internal energy and underestimation of the effect of the sheet stacking nature of cellulose could also be sources of problems. Nevertheless, this simple model provides a viable approximation for such a complex compound and can be further developed to predict optimal environments for the processing of substances and products made of cellulose.

References

[1] Zhibo Ding, Ying Guan, Yongjun Zhang, and XX Zhu. Layer-by-layer multilayer films linked with reversible boronate ester bonds with glucose-sensitivity under physiological conditions. *Soft Matter*, 5(11):2302–2309, 2009.

[2] Frederick Reif. *Fundamentals of statistical and thermal physics*. Waveland Press, 2009.

[3] William D Jones. Conquering the carbon-hydrogen bond. *Science*, 287 (5460):1942–1943, 2000.

[4] Alfred Joaquim Stamm et al. Wood and cellulose science. *Wood and cellulose science.*, 1964.

[5] Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N Groves, Bjørk Hammer, Cory Hargus, et al. The atomic simulation environment—a python library for working with atoms. *Journal of Physics: Condensed Matter*, 29(27):273002, 2017.

[6] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90, 2007.

[7] Teppei Suzuki. The hydration of glucose: the local configurations in sugar–water hydrogen bonds. *Physical Chemistry Chemical Physics*, 10(1):96–105, 2008.

[8] Hugo C Hamaker. The london—van der waals attraction between spherical particles. *physica*, 4(10):1058–1072, 1937.

[9] ID Page. Van der waals interactions.

[10] T Hatakeyama, K Nakamura, and H Hatakeyama. Studies on heat capacity of cellulose and lignin by differential scanning calorimetry. *Polymer*, 23(12): 1801–1804, 1982.

[11] EN Baker and RE Hubbard. Hydrogen bonding in globular proteins. *Progress in biophysics and molecular biology*, 44(2):97–179, 1984.

[12] Zygmunt S Derewenda, Linda Lee, and Urszula Derewenda. The occurence of c–h··· o hydrogen bonds in proteins. *Journal of molecular biology*, 252 (2):248–262, 1995.

[13] LGJMA Segal, JJ Creely, AE Martin Jr, and CM Conrad. An empirical method for estimating the degree of crystallinity of native cellulose using the x-ray diffractometer. *Textile research journal*, 29(10):786–794, 1959.

[14] Sunkyu Park, John O Baker, Michael E Himmel, Philip A Parilla, and David K Johnson. Cellulose crystallinity index: measurement techniques and their impact on interpreting cellulase performance. *Biotechnology for biofuels*, 3(1):10, 2010.

7 Appendix A: Code

```
1
2  #-----------------------------------------
3  #--------------CELLULOSE MODEL-------------
4  #-----------------------------------------
5  #Robert Rinearson 2019
6
7  from mpl_toolkits.mplot3d import axes3d
8  import matplotlib.pyplot as plt
9  from scipy.interpolate import make_interp_spline, BSpline
10 from scipy.interpolate import spline
11 import scipy.interpolate as interpolate
12 from scipy.ndimage.filters import gaussian_filter1d
13 import scipy.ndimage
14
15
16 import math
17 import numpy as np
18 import random
19 import pylab
20
21 %matplotlib inline
22
23
24 chainLength = 10
25 chains = 7
26 angstrom = 1*(10**(-10))
27 x = angstrom * 5
28 y = angstrom * 8
29 xang = 5
30 yang = 8
31 mass = .180156  #kg/mol
32
33 hBondCaloriesPerMol = 5 #kilocalories
34 vanderwaalsCalories = 1 #kilocaleriespermol
35 k = 0.0019872041 #KiloCalories / mol
36 tMax = 300;
37
38 latticeSquares = (chainLength-1)*(chains-1)
39
40 s = 5.66
41 a = 1.5*hBondCaloriesPerMol
42 b = 1.6*hBondCaloriesPerMol
43 c = 1.1*hBondCaloriesPerMol
44 d = 1*hBondCaloriesPerMol
45 e = 1.1*hBondCaloriesPerMol
46 f = 1*hBondCaloriesPerMol
47 g = 1.5*hBondCaloriesPerMol
48 ad = a + d
49 bd = b + d
50 cd = c + d
51 ef = e + f
52
53
54 #------ ARRAYS
55
```

```
intraEnergies = [s, a, b, c, d, ad, bd, cd]
interEnergies = [s, e, f, g, ef]

#---------- METHODS

#------------------------------
#----- PARTITION FUNCTION -----
#------------------------------

def intraPartition(temp):
    partition = 0
    for i in range(len(intraEnergies)):
        if intraEnergies[i] <= k*temp*6.33 and i > 0:
            state = 0
            partition += state
        else:
            state = np.exp((-intraEnergies[i])/(k*temp))
            partition += state
    return partition

def interPartition(temp):
    partition = 0
    for i in range(len(interEnergies)):
        if interEnergies[i] <= k*temp*6.33 and i > 0:
            state = 0
            partition += state
        else:
            state = np.exp((-interEnergies[i])/(k*temp))
            partition += state
    return partition


#-------------------------
#----- PROBABILITIES -----
#-------------------------

def intraProbabilities(temp):
    probabilities = []
    probs = np.zeros(len(intraEnergies))
    partition = intraPartition(temp)
    for i in range(len(intraEnergies)):
        if intraEnergies[i] <= k*temp*6.33 and i > 0:
            probability = 0
            probs[i] = probability
        else:
            probability = np.exp((-intraEnergies[i])/(k*temp))/
    partition
            probs[i] = probability
    return probs

def interProbabilities(temp):
    probabilities = []
    probs = np.zeros(len(interEnergies))
    partition = interPartition(temp)
    for i in range(len(interEnergies)):
        if interEnergies[i] <= k*temp*6.33 and i > 0:
            probability = 0
```

```
            probs[i] = probability
        else:
            probability = np.exp((-interEnergies[i])/(k*temp))/
    partition
            probs[i] = probability
    return probs


#--------------------
#----- ENERGIES -----
#--------------------

def intraEnergy(temp):
    probabilities = intraProbabilities(temp)
    rand = random.random()
    val = 0
    for i in range(len(intraEnergies)):
        if rand <= (val + probabilities[i]):
            return intraEnergies[i]
        val += probabilities[i]

def interEnergy(temp):
    probabilities = interProbabilities(temp)
    rand = random.random()
    val = 0
    for i in range(len(interEnergies)):
        if rand <= (val + probabilities[i]):
            return interEnergies[i]
        val += probabilities[i]

def avgIntraEnergy(temp, times):
    totalE = 0
    for i in range(times):
        totalE += intraEnergy(temp)
    avgE = totalE / times
    return avgE

def avgInterEnergy(temp, times):
    totalE = 0
    for i in range(times):
        totalE += interEnergy(temp)
    avgE = totalE / times
    return avgE

#-------------------------
#----- GIBBS ENTROPY -----
#-------------------------


def edgeEntropy(probability):
    if probability != 0:
        entropy = -k * probability * np.log(probability)
        return entropy
    else:
        return 0


```

```
def entropyConstArray(temp):
    intraProbs = intraProbabilities(temp)
    interProbs = interProbabilities(temp)

    entropy = 0
    for i in range(len(intraProbs)):
        entropy += edgeEntropy(intraProbs[i])

    for j in range(len(interProbs)):
        entropy += edgeEntropy(interProbs[j])
    return entropy


def entropyMultiple(temp):
    intraProbs = intraProbabilities(temp)
    interProbs = interProbabilities(temp)
    entropy = 0

    for g in range(len(intraProbs)):
        for h in range(len(interProbs)):
            entropy += edgeEntropy(intraProbs[g]*intraProbs[h])

    return entropy


#----------------------------
#----- PHYSICAL ENTROPY -----
#----------------------------

def physicalEntropy(temp):
    intraEnergy1 = intraEnergy(temp)
    intraEnergy2 = intraEnergy(temp)
    interEnergy1 = interEnergy(temp)
    interEnergy2 = interEnergy(temp)

    kx1 = 6 * intraEnergy1 / (x**2)
    kx2 = 6 * intraEnergy2 / (x**2)
    ky1 = 6 * interEnergy1 / (y**2)
    ky2 = 6 * interEnergy2 / (y**2)

    kxReduced = (kx2 - kx1)/(kx2 + kx1)
    kyReduced = (ky2 - ky1)/(ky2 + ky1)
    randX = random.uniform(-1, 1)
    randY = random.uniform(-1, 1)

    if temp > tMax:
        temp = tMax
    tWeight = temp/tMax

    xDisplacement = (1/2)*(x)*randX*(kxReduced + tWeight)
    yDisplacement = (1/2)*(y)*randY*(kyReduced + tWeight)
    rDisplacement = (xDisplacement**2 + yDisplacement**2)**(1/2)

    x1Displacement = (((x + xDisplacement)**2) + yDisplacement**2)
    **(1/2) - x
    x2Displacement = (((x - xDisplacement)**2) + yDisplacement**2)
    **(1/2) - x
```

```
    y1Displacement = (((y + yDisplacement)**2)+ xDisplacement**2)
    **(1/2) - y
    y2Displacement = (((y - yDisplacement)**2)+ xDisplacement**2)
    **(1/2) - y

    fx1 = -kx1*(x1Displacement)
    fx2 = kx2*(x2Displacement)
    fy1 = -ky1*(y1Displacement)
    fy2 = ky2*(y2Displacement)
    fTotal = ((fx1 + fx2)**2 + (fy1 + fy2)**2)**(1/2)
    ux1 = (1/2) * kx1 * (x1Displacement**2)
    ux2 = (1/2) * kx2 * (x2Displacement**2)
    uy1 = (1/2) * ky1 * (y1Displacement**2)
    uy2 = (1/2) * ky2 * (y2Displacement**2)

    uTotal = abs(ux1) + abs(ux2) + abs(uy1) + abs(uy2)

    dQ = uTotal + fTotal * rDisplacement
    dS = dQ/temp
    return dS


def avgPhysicalEntropy(temp, times):
    total = 0
    for i in range(times):
        total += physicalEntropy(temp)
    return total/times


#-------------------------
#----- HEAT CAPACITY -----
#-------------------------

def heatCap(tempInitial, tempFinal):
    qTotal = 0
    for i in range(tempInitial, tempFinal):
        gibbs2 = entropyConstArray(tempFinal)
        gibbs1 = entropyConstArray(tempInitial)
        changeEntropy = gibbs2-gibbs1
        qTotal += (abs(avgPhysicalEntropy(i, 50)) + tempFinal*
    changeEntropy)
    heatCap = qTotal / (tempFinal - tempInitial)
    return heatCap


def avgHeatCap(tempInitial, tempFinal, reps):
    totalHeatCap = 0
    for i in range(reps):
        totalHeatCap += heatCap(tempInitial, tempFinal)
    return totalHeatCap/reps


#--------------------
#----- PRESSURE -----
#--------------------

def pressureX(temp):
```

```
277     #VERTEX 1
278     intraEnergy1 = intraEnergy(temp)
279     intraEnergy2 = intraEnergy(temp)
280     interEnergy1 = interEnergy(temp)
281     interEnergy2 = interEnergy(temp)
282 
283     totalEnergy1 = intraEnergy1 + intraEnergy2 + interEnergy1 +
        interEnergy2
284 
285     #VERTEX 2
286     intraEnergy3 = intraEnergy(temp)
287     intraEnergy4 = intraEnergy(temp)
288     interEnergy3 = interEnergy(temp)
289     interEnergy4 = interEnergy(temp)
290 
291     totalEnergy2 = intraEnergy3 + intraEnergy4 + interEnergy3 +
        interEnergy4
292 
293     deltaEnergy = abs(totalEnergy1 - totalEnergy2)
294     pressure = deltaEnergy / (xang*xang*yang)
295     return pressure
296 
297 def pressureY(temp):
298     #VERTEX 1
299     intraEnergy1 = intraEnergy(temp)
300     intraEnergy2 = intraEnergy(temp)
301     interEnergy1 = interEnergy(temp)
302     interEnergy2 = interEnergy(temp)
303 
304     totalEnergy1 = intraEnergy1 + intraEnergy2 + interEnergy1 +
        interEnergy2
305 
306     #VERTEX 2
307     intraEnergy3 = intraEnergy(temp)
308     intraEnergy4 = intraEnergy(temp)
309     interEnergy3 = interEnergy(temp)
310     interEnergy4 = interEnergy(temp)
311 
312     totalEnergy2 = intraEnergy3 + intraEnergy4 + interEnergy3 +
        interEnergy4
313 
314     deltaEnergy = abs(totalEnergy1 - totalEnergy2)
315     pressure = deltaEnergy / (xang*yang*yang)
316     return pressure
317 
318 def avgPressureX(temp, times):
319     totalP = 0
320     for i in range(times):
321         totalP += pressureX(temp)
322     avgP = totalP/times
323     return avgP
324 
325 def avgPressureY(temp, times):
326     totalP = 0
327     for i in range(times):
328         totalP += pressureY(temp)
329     avgP = totalP/times
```

```
330     return avgP
331
332
333 #-------------------------
334 #----- SPECIFIC HEAT -----
335 #-------------------------
336
337
338 def specificHeat(tempInitial, tempFinal):
339
340     q = 0
341     for i in range(tempFinal, tempInitial):
342         q += avgPhysicalEntropy(i, 30) * i
343     specificHeat = q/(mass*(tempFinal - tempInitial))
344     return specificHeat
345
346 #-------------------------
347 #----- CRYSTALLINITY -----
348 #-------------------------
349
350 def crystallinity(temp, intraNum, interNum):
351     intraProbs = intraProbabilities(temp)
352     interProbs = interProbabilities(temp)
353     intraProbMultiple = intraProbs[0]**intraNum
354     interProbMultiple = interProbs[0]**interNum
355     crystal = 1 - (intraProbMultiple * interProbMultiple)
356
357     return crystal
```

Listing 1: CODE FOR MODEL