Why are cooked spaghettis soft?

Frederik Heber

November 2, 2014

Abstract

In Germany there is a very famous and long-running TV show for kids called "Die Sendung mit der Maus" (the show with the mouse[1]). I must admit that I still watch it frequently because you learn a lot, e. g. how a stuff is produced in factories, how walnut oil is made, and much, much more. I have even got a "Maus"-style tear-off calendar with a question and an answer on the flipside for every day of the year!

One of these questions (from 19th of June 2014) was: Why do noodles become soft during cooking?

My first guess was that cooking breaks up the starch chains to some extent and that's why the noodles are softer afterwards. The tear-off page's flipside however taught me a different answer: They consist of milled durum and come dry out of the package. Thus, they are hard but also last for a long time. They get soaked with water during cooking and that's why they are soft.

I immediately had a ball-and-stick model in my head of starch chains with and without water and thought of how cool it would be to make a molecular dynamics simulation to check whether the answer could be validated this way. So this is me performing this virtual experiment. And the best thing of all: As there is no bond breaking involved, I can use the BOSSANOVA scheme, detailed in my thesis[Heber(2014)].

1 Introduction

So, let's start with the fun stuff: We want to perform a molecular dynamics simulation. That is we want to have atoms with electrons sitting in a certain domain (coined the "simulation box") and have a quantum chemistry code calculate the interactions between the atoms and electrons. These calculated forces are integrated over time in discrete, but very small steps. Imagine it like a simulation of planets orbiting the sun. The attractive force of the sun accelerates the planets in very small time steps and thus pulls the planets into a circular motion. However, in our case every planet is sort of a sun by itself pulling at all other suns ...

1.1 Coordinates and abundances

For this simulation to begin with, we need to know about the coordinates of every atom in this box. Sadly, we do not have some kind of "molecular scanner"

[1]. . . and not to forget: and the elephant

that would produce this kind of information. There are mass spectrometers and alikes but they only give you the abundances of the different molecules found in spaghetti, for example. For dry pasta[2], this is mostly Semolina flour. Semolina flour is actually durum, and it consists to the greatest part of carbohydrates 72.83% and second water(!) 12.67%[3].

How much water then does contain a cooked noodle, e. g. a single spaghetti? I wanted to do some measurement on my own – being tired of looking stuff up via google. So, here it comes.

1.1.1 Approximate determination of contents of dry and cooked spaghetti

First of all, let us simplify things a bit right from the start: we assume that spaghetti consist only of durum and it in turn only of starch and water. To this end, we normalize the above abundances to sum up to unity.

Then, knowing their renormalized abundances in the dry noodle – $A^d_{H_2O} = 0.15$ and $A^d_{\text{starch}} = 0.85$ – and knowing the atomic weight of amylose μ_{starch} and water μ_{H_2O} we need to determine the average weight of a dry m^d_{noodle} and a cooked m^c_{noodle} noodle and also their average volumes V^d_{noodle} and V^c_{noodle}, respectively, to determine the missing values.

So, let's measure: ten cooked spaghetti noodles replace 40 ± 5 ml of water, i. e.$V^c_{\text{noodle}} = 4$ ml. Their mass is 24 ± 2 g, i. e. $m^c_{\text{noodle}} = 2.4$ g. The first uncertainty is guessing and the second is the uncertainty of the kitchen scale.

Furthermore, 50 dry spaghetti noodles replace 30 ± 5 ml of water, i. e.$V^d_{\text{noodle}} = 0.6$ ml. Their mass is 44 ± 2 g, i. e. $m^d_{\text{noodle}} = 0.88$ g, where the uncertainties are obtained in the same manner.

Thereby, we have the densities of the dry noodle 1.47 $\frac{g}{cm^3}$ and the cooked noodle 0.6 $\frac{g}{cm^3}$, respectively. That's quite a saving in terms of packing!

1.1.2 Length and coordinates of starch chains

From a patent specification (No. DE60007798 T2) I get that Semolina consists of starch chains between 180-350 micron in length. This is far too long for a simple molecular dynamics simulation, but we get to this point later.

Starch chains come in branched and unbranched varieties. Unbranched the molecule is called amylose, branched it is referred to as amylopectin[4]. Thanks to some googling I was able to get a hold of files with coordinates for either type, see Figure 1 and they are much shorter. On a sidenote: I even found a reference for starch being the major constituent of durum, see [Abecassis(2000), page 41]

Note that the atomic weight of the amylose molecule $O_{56}C_{66}H_{112}$ in the file is $\mu_a = 1{,}801.57$ u, which we calculate from knowing all contained atoms, their atomic weights and summing up.

[2]See `http://en.wikipedia.org/wiki/Pasta` on Wikipedia
[3]See `http://en.wikipedia.org/wiki/Semolina` on Wikipedia
[4]So far I am only aware that potato starch and not durum consists of amylose, but let's keep it simple here

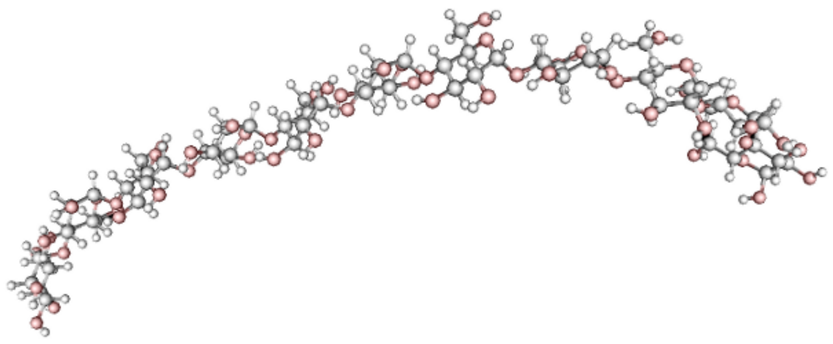


Figure 1: Ball-and-stick model of amylose: hydrogen is in white, carbon in grey, and oxygen in red.

1.1.3 Putting it all together: The question to be answered

So, now we have a lot of numbers to begin with. All that remains is to put it all together, i. e. to put amylose (and water) in the right amounts in a simulation box.

But what do we want to do afterwards? What do we want to measure? What is the number that validates or invalidates the flipside's answer? We need to know this beforehand as it has a significant impact on the setup!

For this we need a model; an abstraction of what's happening in the real world in such a way as to leave in all that's important and leave out all that's not.

This is what the computer can never do for you: To formulate the right questions. And science is all about asking the right questions eventually.

In our case our question was: *Why are cooked noodles soft while dry ones are not.*

Now this is a good question to a human being but not to a computer program, that can only measure certain quantities. We have to reformulate the question for it.

Soft refers to a reaction to an external force: it means it gives way more easily. Giving way in the case of starch chains then does not mean that the starch chains themselves give way and break eventually but that they may slide along one another. So, we are interested in the strength of intermolecular interactions, the forces between the starch chains.

Hence, the question we will ask the computer is: *How strong are the intermolecular forces between starch chains in the dry and the cooked noodle?*

2 Setting up the model

Having now our question at hand, we build the model around it. As we are looking at the interactions between two starch chains, we need at least two of them. More would be plenty, so let's just keep it down to two amylose molecules. The less atoms we have in the domain the quicker the simulation will run.

Next, what else do we need? Water is the second ingredient of Semolina flour. However, as a first step we will leave it out of the model. (Maybe to come up later with a follow-up experiment to investigate the validity of this simplification.)

So, we go about as follows:

1. Create a simulation box of a certain size.
2. Place two amylose molecules into the simulation box in a certain distance to one another.
3. Calculate a single molecular dynamics step to have the atomic forces.
4. Add up the forces for each amylose molecule and divide by the number of atoms to obtain the average molecular force.
5. Check the relation between the distance between the amylose molecules, the water contents (i. e. from cooked to dry state) against the difference in force magnitude acting between the molecules.

And hopefully we'll then have our answer!

Now, we have been very unspecific at some points, namely those marked in above list by the appearance of the word "certain". We need to precisely specify these values and will do so in the following section.

2.1 Some prior calculations

To this end, now we need to do some calculations: The thing we will do most often is converting mass into number of molecules and vice versa. We need the Avogadro constant $N_A \approx 6.02e{+}23$ and molecular masses expressed in gram over a certain number of molecules, e. g. water's atomic mass is 18 u or *atomic mass units*. Then, 18 g of water contain $\frac{18g}{18u} = N_A$ water molecules, i. e. as many as the Avogadro constant, also referred to a 1 mol.

2.1.1 Number of molecules

Given now a single noodle, how many water molecules N_w and amylose molecules N_a does it contain?

$$N_w := \frac{A_w \cdot m_n}{\mu_w}$$
$$N_a := \frac{A_a \cdot m_n}{\mu_a},$$

respectively, where the index w and a refer to water and amylose and m_n is the mass of the noodle.

For the dry state we have the abundances and therefore we obtain the following number of water molecules $N_w^d = 7.25e{-}3$ mol. For the number of amylose molecules we get $N_a = 4.16e{-}4$ mol.

Next, we assume that the cooked noodle's abundances change only by the uptake of water. We neglect the bit of starch leaking out into the boiling water. This is what makes the noodle water milky in appeareance if you cook the noodles for too long.

Hence, the difference in mass between dry and cooked noodle is solely due to the uptake of water molecules. Therefore, we get that the relative mass abundance of water has increased to 0.69. The absolute abundance of amylose was assumed to remain the same, but the relative abundance naturally dropped to 0.31.

In the cooked state we then obtain for the number of water molecules $N_w^c = 9.17e{-}2$ mol, while the number of amylose molecules remains the same by assumption.

Thereby, we also have how many water molecules we need to add per amylose molecule: $\frac{N_w}{N_A}$, namely $\frac{N_w^d}{N_A} = 17.41$ in the dry state and $\frac{N_w^c}{N_A} = 220.37$ in the cooked state.

Already we notice the huge difference and how much drier noodles in their wrapping are.

2.2 Size of the simulation domain

The size of the simulation domain *would* actually be a trivial matter. As we use open boundary conditions (hence, two amylose molecules in vacuum) and not periodic ones (two amylose molecules in an endless sea of other ones) and as we neither add water, the simulation box just has to be large enough to contain the two completely.

However, later we want to know the typical distance between amylose molecules for the dry state and the cooked state. Therefore, we need to know about the volumes a single water molecule and a single amylose molecule occupy on average. This is just the volume of the noodle times the specific relative abundance over the total number of molecules of the specific type, i. e. amylose or water. So, how much volume do two amylose molecules and the associated number $2 \cdot \frac{N_w}{N_A}$ of water molecules require?

$$D := 2 \cdot \frac{V_n}{N_a}$$

where V_n is the volume of the noodle and D the volume of the simulation domain. Note that this automatically includes the volume of the associated water molecules.

We obtain $D^d = 4{,}789.3$ Å^3 for the dry noodle and $D^c = 31{,}929$ Å^3 for the cooked noodle.

2.3 Typical distance of amylose molecules in durum

The last missing number is the typical distance between the amylose molecules.

We consider amylose as roughly cylindrical. Therefore, we ask: By how much do we need to extend both length and radius of this cylinder, starting from the volume of a single amylose molecule, in order to match a cuboid domain of $\frac{V_n^d}{N_a} = 2{,}394.6$ Å^3 and $\frac{V_n^c}{N_a} = 15{,}964.5$ Å^3, respectively? That is the average volume of one amylose molecule including the surrounding water molecules.

Hence, we need to know the length and volume of the amylose molecule as a cylinder to obtain its radius in the same way as with the whole noodle. This can be done with *MoleCuilder*.

First, we calculate the principal axis system which will give use the length of the longest axis in the molecule.

```
$ ./molecuilder -l amylose_straightened_more_opt.pdb \
    --select-all-molecules \
    --rotate-to-principal-axis-system 1,0,0 \
    --add-empty-boundary 1e-10,1e-10,1e-10
```

The length of the box in the x direction is then the length of the cylinder, namely 50.68.

Next, we envelope the amylose chain molecule with a tesselation surface and calculate its volume. However, the obtained volume depends on the size of the rolling sphere by which we obtain the tesselation. We obtained some values by the following command:

```
$ for radius in `seq 2 1 50`; do
$   echo -e "radius\tvolume" >volumina.dat
$   echo -n -e "$radius\t" >>volumina.dat
$   ./molecuilder -l amylose_straightened_more_opt.pdb \
      --select-all-molecules \
      --verbose 1 \
      --convex-envelope $radius \
        --convex-file convex \
        --nonconvex-file nonconvex \
        --DoOutputEveryStep 0 \
      2&>1 >convexize.log
$   grep "non-convex␣tesselated␣volume␣area" \
convexize.log | awk '{print $7}' >>volumina.dat
$ done
```

This creates a file *volumina.dat*, which is plotted in Figure 2.

So, what sphere radius is the right one? This depends on the interaction radius of the water molecule. A typical value would be 2 Å.

We then get 501.53 Å^3 as the volume of the amylose molecule and therefore $r_a = 1.77$ Å as the radius of the cylinder.

Now, we have the volume of the domain, but what we need to know is its extent in every direction. We imagine that the cylindrical amylose is sitting in the center of the domain and that the distance to the domain's boundary is equal in any direction. Hence, we are looking for a distance d to extend the minimal rectangular bounding box of the amylose's cylinder to match in volume with the known values for the dry and cooked state.

To solve then for this unknown extra distance d, we need to solve a third order polynomial of the volume with the length l and the radius d, namely

$$\begin{aligned} d^3 + (l+2r)d^2 + (2lr + r^2)d + \frac{4lr^2 - \frac{V_n}{N_a}}{4} &= 0 \\ d^3 + a_2 d^2 + a_2 d + a_0 &= 0 \end{aligned}$$

for the unknown d. This is done via the Cubic Formula (from Wolfram MathWorld), see Table 1 for the intermediate values.

We have $D < 0$ in both cases and therefore obtain three real roots d_1, d_2, d_3 for the dry and cooked state.

We obtain the following average distance between amylose molecules, $d_a^d = 1.63$ and $d_a^c = 6.58$

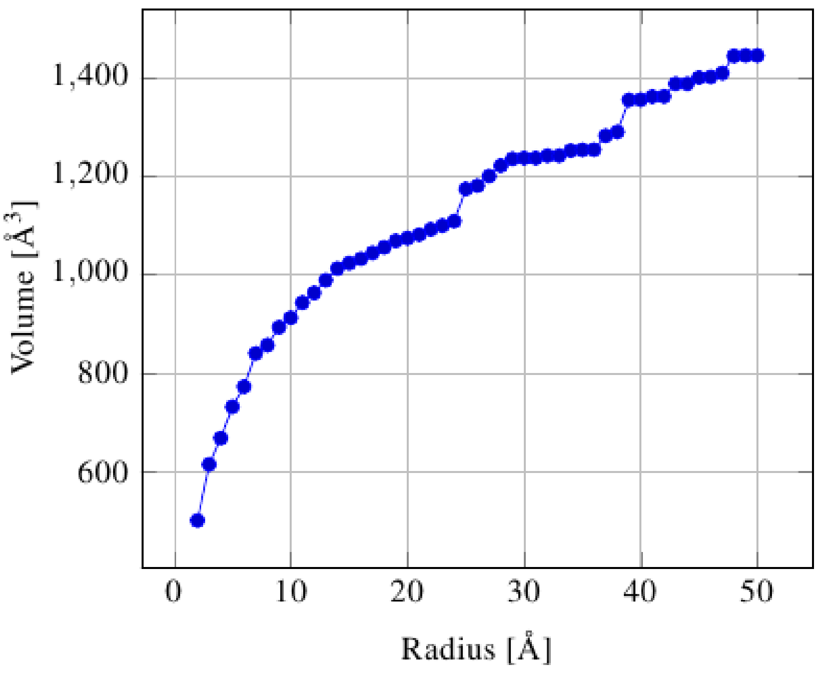


Figure 2: Dependence of the volume contained under the non-convex tesselated surface area with respect to the radius of the rolling sphere that creates it.

Table 1: Initial, intermediate, and final numbers in solving the third order polynomial.

quantity	dry	cooked
a_2	54.23	54.23
a_1	183.05	183.05
a_0	−439.01	−3,831.5
Q	−265.78	−265.78
R	−4,033.8	−2,337.5
D	$-2.5e{+}6$	$-1.33e{+}7$
d_1	1.63	6.58
d_2	−50.43	−48.88
d_3	−5.44	−11.93

Let us check the resulting volume 2,424.89 against the known one $\frac{V_n^d}{N_a} =$ 2,394.6 for the dry state and 15,974.42 against the known one $\frac{V_n^c}{N_a} =$ 15,964.5, which is close enough apart from some rounding issues[5]. All in all, we will have a distance between the two amylose molecules ranging from 1.63 Å to 6.58 Å. And that's what this whole calculation has been about.

3 Experimentation

Now, we can finally start the experimentation and perform it in the three general steps: We begin with the *setup*, let then the simulation *run* and finally *analyse* the values obtained to find our answer.

3.1 Setup

Let's again have a look at the amylose molecule as we obtained it from the web in Figure 1.

We notice that it features quite some bends and it is not as straight a cylinder as we imagined it to be in the calculations above. This also makes it complicated to properly assess the distance between two amylose molecules and its relation to the intermolecular forces.

Let's try to measure the volume in encompasses in a minimal rectangular bounding box:

```
$ ./molecuilder \
  --load amylose.pdb \
  --add-empty-boundary "1e-10,1e-10,1e-10"
```

The new box is then

$$\begin{pmatrix} 44.97 & 0 & 0 \\ 0 & 17.03 & 0 \\ 0 & 0 & 18.19 \end{pmatrix}$$

and has a volume of 13,930.56 Å^3. Comparing this value to $\frac{D^d}{2}$, we immediately notice that its larger.

This makes filling in the two amylose molecules and the required 34 water molecules a lot harder. We would need to place them in an interwoven fashion in some way to not require too large a box.

It is much simpler if we just straighten the molecule a bit:[6]

```
$ ./molecuilder \
  --input amylose_straightened.pdb \
  --load amylose.pdb \
  --select-atom-by-id 90 \
  --translate-to-origin \
  --create-shape "cuboid" \
    --shape-type cube
    --translation "0,-10,-15" \
```

[5]The calculation is done via pgfmath inside the LaTeX document

[6]Naturally, we picked the atom to use as point of rotation in the GUI (we just give here its id). Also, we picked the rotation angle (and axis) in a try&error fashion but this required just a few attempts and undo/redo a step is just the click on a button.

```
    --stretch "20,20,20" \
  --select-shape-by-name "cuboid" \
  --select-all-atoms-inside-shape \
  --unselect-atom-by-id 90 \
  --rotate-around-origin "50" \
    --position "0,1,-0.3" \
  --select-all-molecules \
  --rotate-to-principal-axis-system "0,0,1"
```

Note that the last action automatically aligned the longest prinicipal component of the molecule with the z-axis.

We did a two more unbending moves with the following parameters before the actual rotation to the PAS:

- Atom id #20 with axis $(0.8, 0.2, -0.1)$ and $\alpha = -40$.
- Atom id #79 axis with $(1, 0., 0.)$ and $\alpha = 20$.

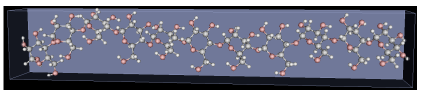


Figure 3: Ball-and-stick model of the finally straightened amylose molecule: hydrogen is in white, carbon in grey, and oxygen in red. Also, the minimal bounding box is shown.

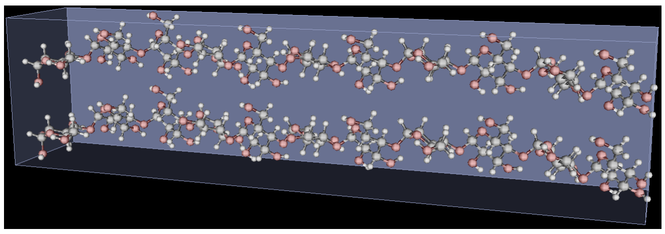
The molecule looks finally like displayed in Figure 3, you notice that a few bends are gone, and also the volume of the bounding box is now just 2,987.89 Å^3.

Note that it's still larger than half of D^d = 4,789.3 Å^3, but we'll leave it at that. The dry state is only achievable if we compress the box and perform some heavier optimization. Eventually, we are just interested in the general tendency from dry to cooked state of the intermolecular force. However, instead of worrying about it, we rather calculate to which distance this value corresponds. It's again a cubic polynomial to solve but instead of $\frac{V_n^d}{N_A}$ we have the above volume.

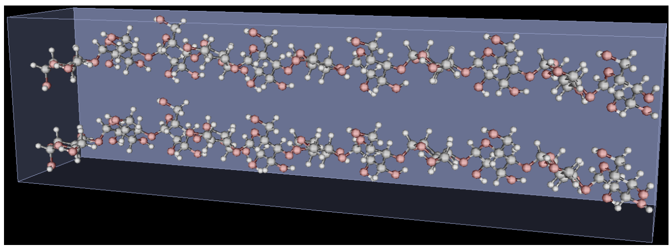
We obtain d_{initial} = 2.01 Å. This is not too far away from 1.63 Å.

We now have a straight molecule but also a strained structure. We need to relax it, especially around the points of rotation. To this end, we perform at least a brief structure optimization right now.

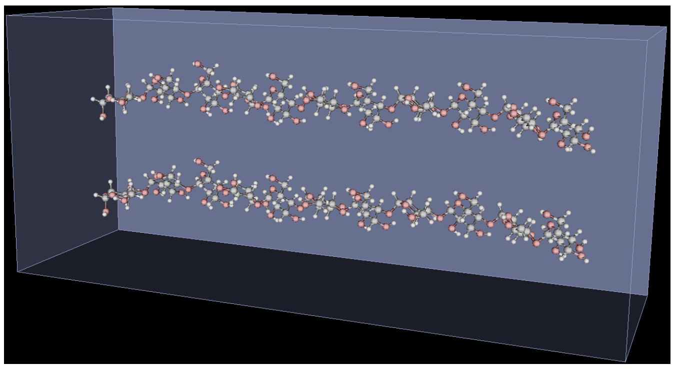
```
$ ./molecuilder \
  --input \
amylose_straightened_more_opt.pdb \
  --load amylose_straightened_more.pdb \
  --add-empty-boundary 1,1,1 \
  --select-all-atoms \
  --optimize-structure \
        --order 4 \
        --distance 3. \
        --fragment-executable mpqc \
        --keep-bondgraph 1 \
        --steps 30
```



(a) $4(r + d) = 12.744$



(b) $4(r + d) = 14.304$



(c) $4(r + d) = 20.704$

Figure 4: Ball-and-stick model of two straightened, optimized amylose molecules: hydrogen is in white, carbon in grey, and oxygen in red. The bounding box is shown. The given value $4(r + d)$ is basically the extension in the y direction.

The last step is then to replicate the box on the z-axis.

And then we are done, the final setup is depicted in Figure 4.

3.1.1 Scripting

Naturally, we need to do this with more than just one distance. We need several ones in the interval $I = [2.01, 6.58]$ Å with an increasing number of water molecules, too.

What we need is a script that does it all for us for a given set of chosen points in the above interval I. For example, let's pick $N = 11$ equidistant points. This is not quite as easy, as we need to adjust the fill-in mesh in integral values. Hence, we obtain a few more values, where we go from 2×2 to 3×3 and so on.

This script is given in 1. What I have done so far, was all in the GUI of *MoleCuilder*. I then obtained this script by simply saving the session and some further, small modifications, e. g. adding loops.

3.2 Running

Now, we need to run the molecular dynamics simulation, or rather just the force calculation.

This is done by the script 2 as follows:

```
$ for i in ‘seq 0 10‘; do
$   ./calculateit.sh amylose_straightened_fillin—${i}.data
$ done
```

Note that we do not use the right bounding box to prevent (in a very simple fashion) the fragmentation "interfragmenter" from combining fragments over the boundary walls (i. e. no periodic boundary conditions apply).

Furthermore, we use "3-21G" as basis set and "MBPT2" as level of theory as we must include non-covalent bonding.

3.3 Analysis

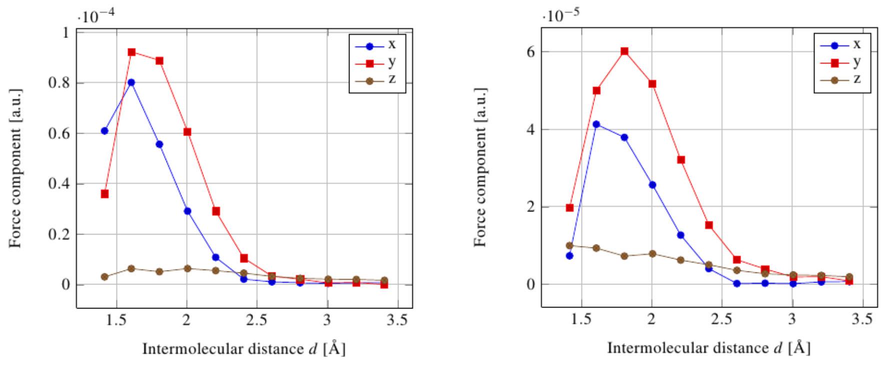
And finally, we may analyse the forces per (amylose) molecule per time step, done by the script 3.

In Figure 5 we give the resulting force with respect to the distance vector between the two amylose molecules for two different basis sets, *3-21G* and *4-31G*.

We see what we expected: The forces are strongest close to the dry state and they get weaker the further apart the amylose molecules are and also the closer we get to the cooked state. However, we also notice that we are still quite a bit away from the true cooked intermolecular distance of $d^c = 6.58$ Å. However, the force has already reached a plateau at $d \approx 3$ Å, half the distance of the cooked state.

Hence, we conclude that noodles indeed become softer during cooking because they take up water, decrease thereby their density and especially increase the distance between the amylose molecules. Thus, the noodle looses its brittle nature.

What we did not expect is that the maximum force between the amylose molecules is acting at a distance of between $d = 1.6$ Å and $d = 1.8$ Å depending



(a) 3-21G (b) 4-31G

Figure 5: Average force acting between the two amylose molecules for each component over the distance between the two molecules considered as cylinders as described in the text.

on the basis set. This is really close to $d^d = 1.63$ Å. This comes at a nice surprise that we basically recover here the intermolecular distance d^d for the dry state from the force-distance profile.

Also, we notice that the larger basis set *4-31G* sees significant interactions beyond $d = 2.5$ Å, while they end before that with the other basis set *3-21G*.

4 Conclusions

In this experiment we looked at starch chains, the main ingredient of most noodles such as spaghetti, and asked why they become soft during cooking. We translated this question into something the computer can give an answer to, essentially a number; the intermolecular force acting between two (short) amylose molecules, representing the starch chains. We calculated the profile of each intermolecular force component with respect to the distance between the two molecules. The profile showed that the amylose molecules in spaghetti are at a distance to one another where the force acting between them is strongest. Furthermore, the force becomes weaker with greater distance and reaches a plateau some good way before reaching the typical intermolecular distance for the cooked state. Here, water inclusion would probably shift this trend a bit.

Eventually, we realize that sometimes (actually even often) a lot of juggling of numbers is required prior to performing the simulation. Not so much for the simulation itself but rather to allow interpretation of its results.

References

[Abecassis(2000)] Joël Abecassis, Jean-Claude Autran – Durum Wheat, Semolina and Pasta Quality: Recent Achievements and New Trends, Editions Quae, 201 pages, 2001.

[Heber(2014)] Frederik Heber – Ein systematischer, linear skalierender Fragmentansatz für das Elektronenstrukturproblem, Dissertation Rheinische Friedrich-Wilhelms Universität Bonn, 2014.

Listing 1: Script for creating initial setup with just two amylose molecules in a specific distance: *filling.py*

```
import pyMoleCuilder

def fillin(boundary, f):
        "This function encapsulates the filling of the water molecules"
        # clear the World
        pyMoleCuilder.reinit()
        pyMoleCuilder.CommandVerbose("1")
        # load amylose, prepare boundary, load water, and fill domain
        pyMoleCuilder.MoleculeLoad("amylose_opt.pdb")
        pyMoleCuilder.WorldAddEmptyBoundary("%f,%f,%f"
                % (boundary,boundary,boundary))
        # repeat box
        pyMoleCuilder.WorldRepeatBox("1 2 1")
        domain = pyMoleCuilder.getBoundingBox()
        domx=domain[1]
        domy=domain[3]
        domz=domain[5]
        pyMoleCuilder.ParserSetTremoloAtomdata(
                "F      neighbors      u      x", "0")
        pyMoleCuilder.WorldOutputAs("amylose_straightened_fillin-%d.data"
                % (nr))
        pyMoleCuilder.WorldOutputAs("amylose_straightened_fillin-%d.pdb"
                % (nr))
        f.write("%d\t%f\t%f\t%f\t%f\n"
                % (nr, boundary, domx, domy, domz))

f = open("filling.dat", "w")
f.write("nr\tboundary\tdomainx\tdomainy\tdomainz\n")
nr=0
for i in range(0,11):
        boundary=2.*float(i)/10.
        if boundary == 0:
                boundary=0.01
        fillin(boundary, f)
        nr=nr+1
```

Listing 2: Script to calulate the forces on each initial configuration: *calculateit.sh*

```
#!/bin/bash

MOLECUILDER=molecuilder

test ! -z $1 || { echo "Usage: $0 <.data file>"; exit 255; }
```

```
file=$1
domx=`grep Box ${file/pdb/data} | awk '{print $3}'`
domy=`grep Box ${file/pdb/data} | awk '{print $7}'`
domz=`grep Box ${file/pdb/data} | awk '{print $11}'`
nr=`echo $file | sed -e "s#\..*##g" | awk -F"-" '{print $NF}'`

$MOLECUILDER \
        --input ${file/data/pdb} \
        --set-output tremolo \
        --set-tremolo-atomdata "F        neighbors        u        x" \
                --reset 0 \
        --add-empty-boundary 5,5,5 \
        --update-molecules \
        --select-all-atoms \
        --set-parser-parameters mpqc \
                --parser-parameters "theory=MBPT2;basis=4-31G" \
        --fragment-molecule BondFragment \
                --order 3 \
                --inter-order 3 \
                --distance `echo "scale=9; 4+${nr}/2.5" | bc` \
        --fragment-automation \
                --fragment-executable mpqc \
                --server-address 127.0.0.1 \
                --server-port 1026 \
        --analyse-fragment-results \
                --fragment-prefix BondFragment \
        --select-all-molecules \
        --average-molecule-force \
        &>calculation-${nr}.log

        #--change-box "$domx,0,$domy,0,0,$domz"
```

Listing 3: Script to analyse the intermolecular forces of each calculated configuration: *analyse.sh*

```
#!/bin/bash

echo -e "nr\tforce1x\tforce1y\tforce1z\tforce2x\tforce2y\tforce2z" \
        >forces.dat
for i in `seq 0 10`; do
        forces=(`grep "average force" calculation-${i}.log \
                | awk '{print $9}'`)
        echo -e "$i\t${forces[0]}\t${forces[1]}" | tr -d \(\) \
                | sed -e "s#,#\t#g" >>forces.dat
done
```