On the Components of Kinetic Energy During a Barostatted Molecular Dynamics Simulation of Methanol Diffusion in Water

Oliver Burch

March 12, 2026

Abstract

In this experiment we analyse the diffusion of methanol molecules in the TIP3P model of water using a barostatted molecular dynamics simulation in AMBER - a powerful molecular dynamics simulation package for biomolecular systems [1, 2, 3] -to produce sets of positional and velocity data with SHAKE. We calculated an estimate of the components of the kinetic energy of the methanol molecules and provided the evidence for and against our results. In this article we will lay out the methods used, discuss the strengths and limitations of each method, and mention future extensions to this work.

1 Introduction

This article attempts to quantify the translation, rotational and vibration components of kinetic energy of simulated methanol molecules using output position and velocity data created by AMBER. This work was done as it is the supporting framework for more advanced theories of molecular diffusion. Throughout the article we will mention the necessity of the results in future work that we could have conducted, especially in consideration to our results.

As a subset of this work we will also study the use of large data. Large data sets are a very real problem for the modern physicist. The storage [4] and initial filtering [5] of these sets provides challenges beyond the scope of this article. We can instead explore how we can work on smaller samples of the data using different techniques for the goal of reproducing this work on a large data set without the need of ever explicitly working on it.

To calculate total kinetic energy at each step:

$$K_{total}(t) = \sum_{i=1}^{N} \frac{1}{2} m v(t)_i^2 \tag{1}$$

where m is the mass of each molecule at 32 amu and v is root mean squared speed calculated by - in the positional data alone:

$$\vec{v}_i(t) = \frac{\Delta r}{\Delta t} \tag{2}$$

Here Δr is the displacement between steps and Δt is the time per step. We then use the equipartition theorem:

$$T(t) = \frac{2K_{total}(t)}{N_{dof} k_B} \tag{3}$$

where N_{dof} is the number of degrees of freedom calculated using $N_{dof} = 3N_{atoms} - N_{constraints}$ and k_B is Boltzmann's constant. We initially used N_{dof} as 14 due to there being 6 atoms, and the assumption of 4 constraints stemming from the belief that the hydrogen bonds were rigid when using SHAKE(ntc = 2). We will revisit this assumption in the limitations section of the article.

2 Methods

This study analyses multiple sets of data derived from a methanol diffusion TIP3P simulation in AMBER with SHAKE(ntc = 2). This section will analyse how each dataset was used to produce the component results we required. To get to this we will first need to discuss the positional data and can then move on to discuss the more important velocity data. The commented Mathematica code used for this processing can be found in Appendix B of this paper.

2.1 Positional data

The initial data we attempted to produce a method for are the wrapped positional data - for 209 frames - and unwrapped positional data - for 2095 frames. Each of these contained 1224 methanol molecules and had time steps a total 10 ps in length. The data also included 3 box dimensions per frame, which meant in total partitioning the data for each individual frame a total 3675 times per frame. From there we plotted positional data for each set as shown in fig. 1. The plots clearly show the Brownian motion of diffusion [6] as expected from the output of the simulation.

We now needed to calculate the average temperature of the diffusion. This was done using the dimensional components of position to directly calculate each respective velocity using eq. (2). We then used eq. (1) to calculate each individual component of kinetic energy and subbed this into eq. (3) for the final temperatures. This was done at each step as seen in fig. 2, and an average was taken using the mean function in Mathematica.

2.2 Velocity data

The other data we need a method for is the velocity data. The same partitioning for the positional data was required as the only difference was that this data had 100 frames at a much smaller time step of approximately 2 fs. As AMBER has already performed an instantaneous velocity calculation for each entry in the data, we can go straight to using eq. (1) and sub this in to eq. (3) for each step as is plotted in fig. 3. We then again use the mean function in Mathematica to calculate the average temperature and use the standard deviation function to calculate the error.

2.3 Energy calculations

This final method is for the determination of the components of energy of the methanol molecules with a dependence on temperature. Using the assumed degrees of freedom of the three components of the kinetic energy we can find:

$$K_{trans} = \frac{3}{2} N_{mol} k_B T$$
$$K_{rot} = \frac{3}{2} N_{mol} k_B T$$
$$K_{vib} = 4 N_{mol} k_B T$$

and collecting these terms results in the equation for total kinetic energy:

$$K_{tot} = 7 N_{mol} k_B T \tag{4}$$

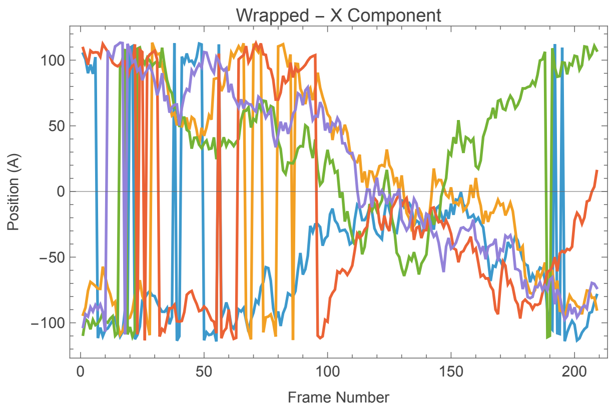
Therefore after calculating the known total kinetic energy from eq. (1) and performing a ratio calculation to find each component, we will have an estimation of the desired result.

3 Results

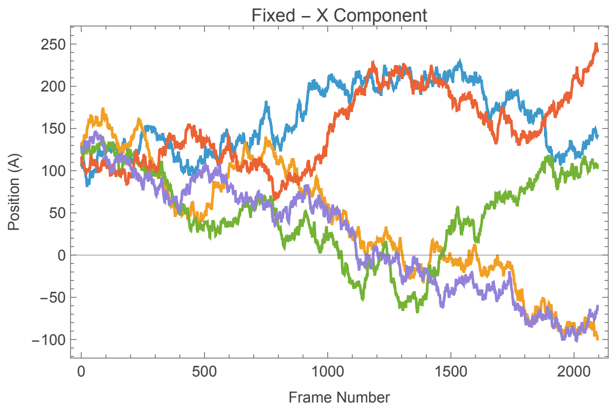
While the positional data provided little information to the main study of this paper, it is useful in regard to how the data was analysed. The results from fig. 1 clearly show a strong difference in the analysis of the data dependent on whether wrapping occurred (a) or didn't occur (b) and this is strongly reflected in fig. 2 as while both temperatures of 464.71 K and 4.02 K are incorrectly calculated due to the large time step, they allow us to view how the data can be used in future research on the diffusion constant. This method also helps us understand the stark differences that can be caused by how we approach the separation of the data, and this helps develop our method for working on a large data set.

The graph of fig. 3 shows that temperature roughly oscillates about one fixed value over time. This is the result we'd expect from Brownian motion when studying the Gaussian distribution of the molecular path. The average of this is T = 315.38 ± 11.42 K. This yields too high an estimate when considering AMBER's result T = 297.55 K. We expect our temperature to be slightly deviant from the thermostat result due to the small sample size, but as this isn't within one standard deviation we can assume something slightly different to our initial assumptions might've occurred. We will discuss this in the limitations.

We also used our findings to predict the kinetic energy to be 4.39 ± 0.16 kcal/mol from eq. (4).

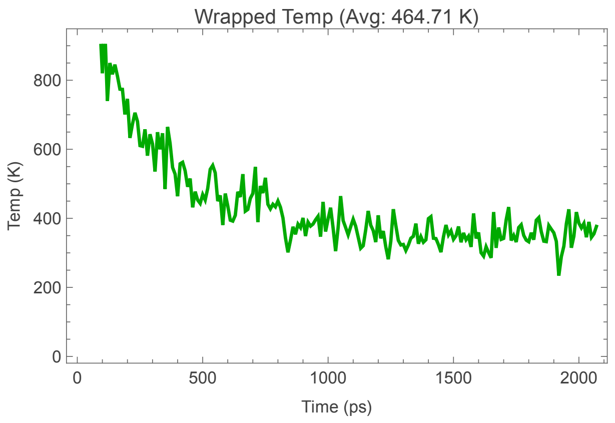


(a) Wrapped Data

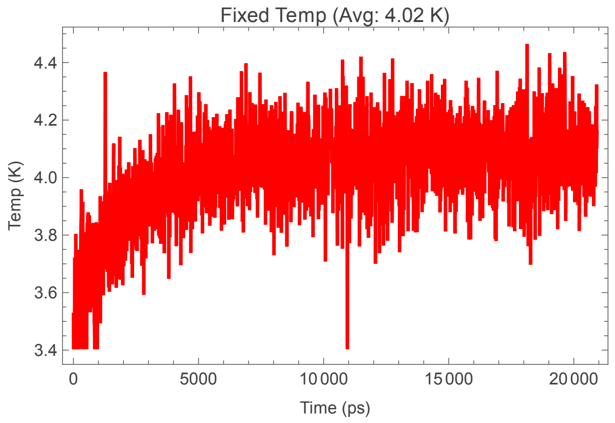


(b) Unwrapped Data

Figure 1: The graphs of the position of the X component of the first 5 molecules across all frames. Here Brownian motion is clearly observed, however the wrapped data frequently spikes due to the periodic boundary conditions of the box. Each spike is a molecule jumping from the left to the right side of the box (or vice-versa). The unwrapped data provides a clear example of the cleaned positional data which avoides the spikes.



(a) Wrapped Data



(b) Unwrapped Data

Figure 2: Average temperature across the simulation. It is clear from this that the region the data covered is the diffusive region; we are unable to perform temperature calculations from it. The benefit of this is the visualisation of the difference between wrapping the data and leaving it unwrapped. The result is far more useful in calculating the Diffusion Coefficient via Einstein's relation.

Assuming the number of degrees of freedom is correctly predicted, we can use the previously established component equations to work out the value of each energy. The ratios are $K_{trans} = K_{rot} = 21.5\%$ and $K_{vib} = 57\%$ and therefore we find that $K_{trans} = K_{rot} = 0.94 \pm 0.03$ kcal/mol while $K_{vib} = 2.51 \pm 0.09$ kcal/mol.

4 Discussion and Conclusion

We have deduced clear methods outlining how we could use our positional data in future works, and to predicting the components of the kinetic energy of methanol molecules during diffusion in water. From our results we have found that our estimate does not match that of AMBER's. This suggests that we have made an incorrect prediction, or that there is something more going on.

These method can be reproduced for much larger sets of data using the code found in appendix B, answering our sub-query of how we can successfully work with large data sets without ever actually opening them. An example of extending this work to a large data set could be using the methods established to unwrap the positional data to identify the diffusive region of the process. This would then mean we could accurately measure the diffusion coefficient via the Einstein relation. This is

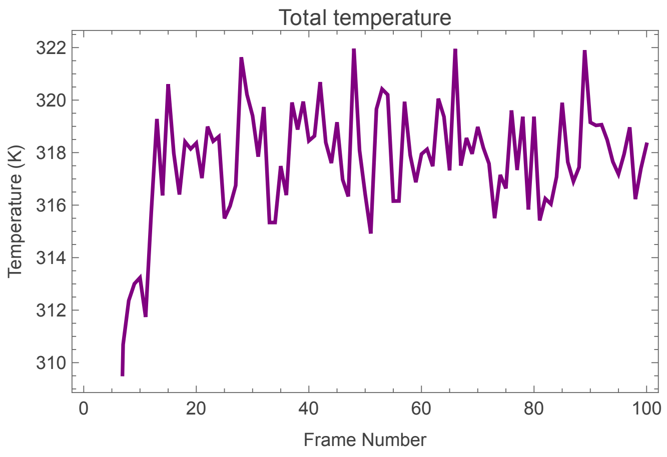


Figure 3: Temperature in each frame for the velocity data. A much stronger average of T = 315.85 ± 11.42 K is calculated when compared to the temperatures in fig. 2. This is down to the much smaller time step creating near-instantaneous velocities which hold much more information about the temperature change.

mentioned in the future work section in more detail.

4.1 Limitations

A limitation for this project is the small amount of velocity data collected. The discrepancy in the temperature that we calculated against the thermostat temperature AMBER produced could slightly be attributed to the finite data that we use instead of the instantaneous data that AMBER can generate. This is likely not the entire story and we can explore this in the next section.

4.1.1 Checking the degrees of freedom

A verification of our assumptions would be checking the degrees of freedom. Using the SHAKE(ntc=2) model that we provided AMBER with, it would be reasonable to suggest there were 14 total degrees of freedom due to the frozen hydrogen bonds, however one check we can perform is if this is consistent to the force field topology. In our initial evaluation of the project we had used N_{dof} as 18 assuming the model to be non-SHAKE. This had given us an output of T = 245.27 ± 8.88 K. Using the knowledge of our result from N_{dof} = 14, it seemed useful to plot the temperature change with a dependence on the degrees of freedom. We find from fig. 4 that the strongest fit actually seems to be using N_{dof} as 15 which produces the result T = 294.33 ± 10.66 K aligning almost perfectly with AMBER's T = 297.55 K.

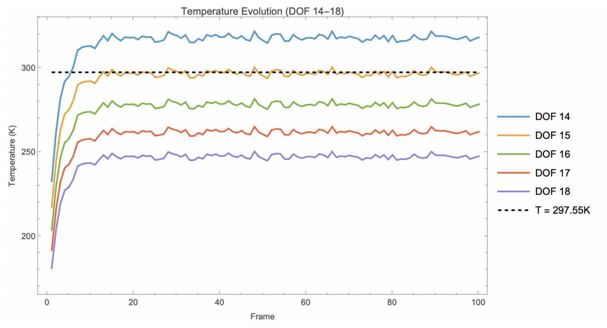


Figure 4: Temperature as a dependence on N_{dof} ranging from 14-18. The plots show that when matched to AMBER's result of T = 297.55 K, N_{dof} = 15 shows the strongest correlation. This is contradictory to the assumption of N_{dof} = 14 and is something explored in the discussion.

This result is highly indicative of there being vibrational freedom in the (O-H) group as it is impossible to have this for the non-polar bonds as the SHAKE algorithm works in coupling constraints. We can actually show this by plotting the length of the O-H bond over time and seeing if it vibrates (flexible) or is flat (constrained). This is done in fig. 5 which, as the mean is $\gg$ 0A, clearly shows that the bond oscillates and is therefore not rigid. As such we have fixed our major difference in temperature and our new results are such that $K_{tot} = 4.39 \pm 0.16$ kcal/mol and, as there are now 9 vibrational degrees of freedom,
$K_{trans} = K_{rot} = 0.88 \pm 0.03$ kcal/mol;
$K_{vib} = 2.63 \pm 0.10$ kcal/mol. The next section will discuss how the effects of this can be measured in future work.

4.2 Future Research

This experiment was handed to us with a limited time of six weeks to find and produce results worthy of this article. In doing so it has been made clear throughout this writing that only a small piece of the diffusion puzzle has been solved. Given more time the data could have been far more useful to completing our understanding of this process. An example of this could have been extending our analysis of the positional data to measure the diffusion coefficient via the Stokes-Einstein relation. The mainly unused method of partitioning and plotting the position data would be far more useful

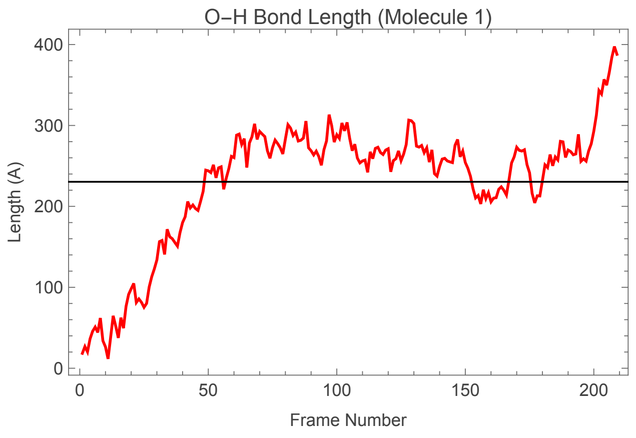


Figure 5: The oscillation of the O-H bond is clearly shown. This presents the strong evidence that the bond is flexible, not constrained as the mean line shown is $\gg 0$A.

for this than the velocity data as we could use the unwrapped data to see the preservation of the total displacement $r(t) - r(0)$ required for the MSD calculation.

In the previous section we highlighted the difference in our assumption of the SHAKE(ntc=2) result with AMBER's result. By plotting the graph in fig. 4 we can clearly see that our use of N_{dof} = 14 was incorrect due to the force field topology of the simulation. We discovered a far stronger fit of N_{dof} = 15 and now need to extend our method to discuss the changes this makes to the model.

Given more time, we would have liked to have performed a fast Fourier transform of the autocorrelation function:

$$I(\omega) = \int_{-\infty}^{\infty} \langle \mathbf{v}(0) \cdot \mathbf{v}(t) \rangle \, e^{-i\omega t} \, dt \tag{5}$$

which should show the stretch of the peak created by the O-H bond over time as it is weakened by the water molecules pulling on the hydrogen. This can be completed once on each N_{dof} = 14 and 15 and compare these to show how the constraint acts, as well as comparing our result for N_{dof} = 15 with known models of water to attempt to critique the TIP3P model used.

For the results we produced, there is still a small difference between ours and AMBER's. This is interesting for us as it likely means something in the system was directly affecting the velocity output of AMBER to be larger than our predicted result. To figure out what this could be, we would have liked to properly debugged our code line by line, then would have ran it for a two molecule simulation over an extended time frame to see if they exhibit the predicted behaviour. If this did not work we'd turn to manually calculating the kinetic energy at each step of the data and could filter any step that produces a result anomalous to the entirety of the data. We could have also re-ran the simulation for numerous sets of data to check if the set we used was an outlier or whether it was the output data.

References

[1] BioSoft. AMBER - molecular dynamics simulation package - BioSoft. URL https://www.biosoft.com/software/amber.

[2] Scott Le Grand, Andreas W. Götz, and Ross C. Walker. SPFP: Speed without compromise—a mixed precision model for GPU accelerated molecular dynamics simulations. 184(2):374–380. ISSN 00104655. doi: 10.1016/j.cpc.2012.09.022. URL https://linkinghub.elsevier.com/retrieve/pii/S0010465512003098.

[3] Romelia Salomon-Ferrer, Andreas W. Götz, Duncan Poole, Scott Le Grand, and Ross C. Walker. Routine microsecond molecular dynamics simulations with AMBER on GPUs. 2. explicit solvent particle mesh ewald. 9(9): 3878–3888. ISSN 1549-9618, 1549-9626. doi: 10.1021/ct400314y. URL https://pubs.acs.org/doi/10.1021/ct400314y.

[4] Luca Clissa, Mario Lassnig, and Lorenzo Rinaldi. How big is big data? a comprehensive survey of data production, storage, and streaming in science and industry. 6:1271639. ISSN 2624-909X. doi: 10.3389/fdata.2023.1271639. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC10620515/.

[5] Sam Jarman. CERN's trigger system - how the LHC copes with a constant flood of data. URL https://www.sciencefocus.com/science/cerns-trigger-system-how-the-lhc-copes-with-a-constant-flood-of-data.

[6] Robert Brown. *The miscellaneous botanical works of Robert Brown*, volume 1.

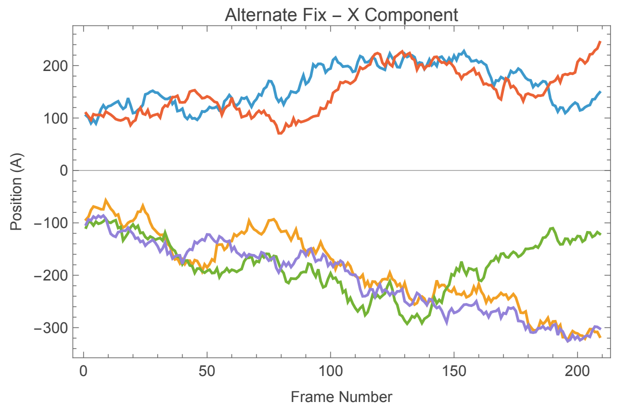
[7] Google. Gemini. URL https://gemini.google.com.

Appendix A: Alternative Positional Fix

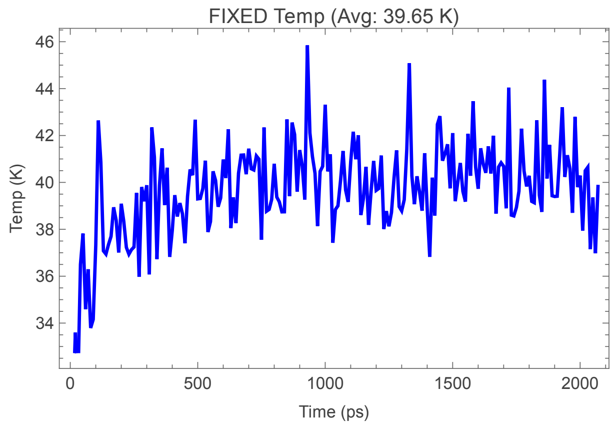
Aside from not wrapping the data, there is another fix to the positional data which presents an entirely new set of results. The fix follows the precedent that and molecule that travels further than half the box length in a single frame can be assumed to be wrapped:

$$|X_2 - X_1| > \frac{l_{box}}{2} \tag{6}$$

For these points a new coordinate system is created where the box dimensions do not exist. While this fixes the issue, it is not included in the methods as the results present further issues and therefore are not worth talking about when the better 'fixed' data is presented as well. Doing this immediately assumes that if any molecules were to cross that barrier it would be being wrapped which negates the effect of a molecule actually travelling that distance in a frame. This can be seen in fig. 6a when comparing to fig. 1b. This is made clearer in fig. 7 as the fixed data has far less noise in comparison to this alternative fix.



(a) Positions of first 5 atoms across 209 frames for the alternative approach to clean the data.



(b) Average temperature of the methanol

Figure 6: Positional data and calculated temperature shown after the alternative fix has been applied. The error in this method is clearly seen wherein no molecules position crosses the zero boundary as compared to b.

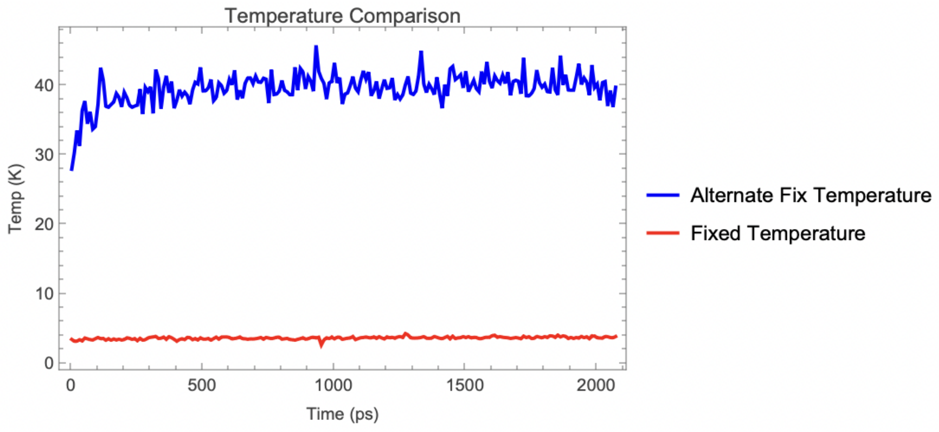


Figure 7: A comparison of temperature from AMBER's fix and the alternative fix. It's clearly shown that AMBER provides much cleaner temperature data.

Appendix B: Mathematica Processing Code

The following code is the finalized script used for processing. The groundwork for this was performed by the google assistant Gemini [7] then was strongly developed to include the niches we needed. It includes helper functions for Periodic Boundary Condition (PBC) correction, 2D plotting, and specific handling for AMBER velocity dumps including a SHAKE degree-of-freedom correction.

B.1 Helper Functions

```
(* --- 1. HELPER FUNCTIONS --- *)

(* A. Unwrapping Function (Fixes PBC Jumps) *)
UnwrapDimension[coords_, boxDim_] := Module[{frames, atoms, unwrapped, delta, L,
    trueDelta},
    {frames, atoms} = Dimensions[coords];
    unwrapped = coords;
    Do[
        delta = coords[[f]] - coords[[f - 1]];
        L = boxDim[[f]];
        (* Adjust if displacement > half box length *)
        trueDelta = delta - L * Round[delta / L];
        unwrapped[[f]] = unwrapped[[f - 1]] + trueDelta;,
        {f, 2, frames}
    ];
    unwrapped
];

(* B. Temperature Calculation with SHAKE Correction *)
CalculateTemperature[posData_, timeStepPs_] := Module[
    {displacements, velocities, speedsSq, atomMasses, totalKE,
     kB, amuToKg, velConv, numFrames, numAtoms, massMethanol, correctionFactor},

    If[Length[posData] < 2, Return[{}]];
    {numFrames, numAtoms, dim} = Dimensions[posData];

    (* Constants *)
    kB = 1.380649*10^-23;
    amuToKg = 1.66054*10^-27;
    velConv = 100.0;

    (* Mass for Methanol (CH3OH) approx 32.04 amu *)
    massMethanol = 32.04;
    atomMasses = ConstantArray[massMethanol, numAtoms];

    (* DOF Correction for SHAKE: 18 coords -> 14 DOFs. Factor = 18/14 *)
    correctionFactor = 18.0 / 14.0;

    displacements = Differences[posData];
    velocities = displacements / timeStepPs;

    Table[
        frameV = velocities[[t]];
        speedsSq = Total[frameV^2, {2}] * (velConv^2);
        totalKE = Total[0.5 * (atomMasses * amuToKg) * speedsSq];
        (* T = 2K / (N_dof * kB) *)
        ((2 * totalKE) / (3 * numAtoms * kB)) * correctionFactor,
        {t, 1, Length[velocities]}
    ]
];

```

```
(* C. 2D Plotting Helper *)
Plot2DComponents[x_, y_, z_, title_, frames_] := Module[
    {dataX, dataY, dataZ, opts},
    dataX = Transpose[x[[All, 1 ;; 5]]];
    dataY = Transpose[y[[All, 1 ;; 5]]];
    dataZ = Transpose[z[[All, 1 ;; 5]]];

    opts = {Frame -> True, ImageSize -> 400, DataRange -> {1, frames},
            FrameLabel -> {"Frame Number", "Position (A)"},
            PlotStyle -> Thickness[0.005]};

    Column[{
        ListLinePlot[dataX, PlotLabel -> title <> " X Component", Sequence @@ opts],
        ListLinePlot[dataY, PlotLabel -> title <> " Y Component", Sequence @@ opts],
        ListLinePlot[dataZ, PlotLabel -> title <> " Z Component", Sequence @@ opts]
    }]
];
```

Listing 1: Core logic for unwrapping and temperature calculation

B.2 Processing Set 1 & 2 (Positional)

```
(* --- 2. PROCESS SET 1: Wrapped & FIXED --- *)
If[ValueQ[Cdata] && Length[Cdata] > 0,
    numAtoms1 = 1224;
    valsPerFrame1 = (numAtoms1 * 3) + 3; (* Coords + BoxDims *)

    If[Divisible[Length[Cdata], valsPerFrame1],
        frameData1 = Partition[Cdata, valsPerFrame1];
        numFrames1 = Length[frameData1];
        coords1 = frameData1[[All, 1 ;; 3 * numAtoms1]];
        boxDims1 = frameData1[[All, -3 ;; -1]];

        (* Reshape to (Frames, Atoms, 3) *)
        xyzRaw1 = Map[Partition[#, 3] &, coords1];
        xRaw1 = xyzRaw1[[All, All, 1]];
        yRaw1 = xyzRaw1[[All, All, 2]];
        zRaw1 = xyzRaw1[[All, All, 3]];

        (* Wrapped Plots *)
        Print[Plot2DComponents[xRaw1, yRaw1, zRaw1, "Wrapped", numFrames1]];

        (* Apply alternative unwrapping Fix *)
        boxT = Transpose[boxDims1];
        xFix1 = UnwrapDimension[xRaw1, boxT[[1]]];
        yFix1 = UnwrapDimension[yRaw1, boxT[[2]]];
        zFix1 = UnwrapDimension[zRaw1, boxT[[3]]];

        xyzFix1 = Table[Transpose[{xFix1[[f]], yFix1[[f]], zFix1[[f]]}], {f, 1,
    numFrames1}];

        (* Fixed Plots *)
        Print[Plot2DComponents[xFix1, yFix1, zFix1, "Alternate Fix", numFrames1]];

        (* Temperature Calculation *)
        tempFix1 = CalculateTemperature[xyzFix1, 10.0];
        times1 = Range[0, Length[tempFix1] - 1] * 10.0;
        avgT1 = Mean[tempFix1];

        Print[ListLinePlot[Transpose[{times1, tempFix1}],
            PlotLabel -> "FIXED Temp (Avg: " <> ToString[NumberForm[avgT1, {5, 2}]] <> "
    K)",
            Frame -> True, PlotStyle -> Blue, FrameLabel -> {"Time (ps)", "Temp (K)"}]];
    ];
];

(* --- 3. PROCESS SET 2: Unwrapped DATA --- *)
If[ValueQ[Cdatafixed] && Length[Cdatafixed] > 0,
    numAtoms2 = 1224;
    valsPerFrame2 = (numAtoms2 * 3) + 3;

    frameData2 = Partition[Cdatafixed, valsPerFrame2];
    coords2 = frameData2[[All, 1 ;; 3 * numAtoms2]];
    xyz2 = Map[Partition[#, 3] &, coords2];

    temp2 = CalculateTemperature[xyz2, 10.0];
    times2 = Range[0, Length[temp2] - 1] * 10.0;

    (* Comparison Plot *)
    minLen = Min[Length[times1], Length[times2]];
    data1Trim = Transpose[{times1[[1 ;; minLen]], tempFix1[[1 ;; minLen]]}];
    data2Trim = Transpose[{times2[[1 ;; minLen]], temp2[[1 ;; minLen]]}];

    Print[ListLinePlot[{data1Trim, data2Trim},
        PlotLegends -> {"Set 1 (My Fix)", "Set 2 (Reference)"},
        PlotStyle -> {Blue, Red}, Frame -> True,
        PlotLabel -> "Temperature Comparison"
    ]];
];
```

Listing 2: Processing Wrapped/Alternative fix and Unwrapped

B.3 Velocity Dump Processing

```
(* --- 4. PROCESS SET 3: AMBER VELOCITIES --- *)
If[FileExistsQ[txtFileName],
    fileContent = Import[txtFileName, "Text"];
    startPos = StringPosition[fileContent, "velocities ="];

    If[Length[startPos] > 0,
        dataStart = startPos[[1, 2]] + 1;
        semiColonPos = StringPosition[fileContent, ";", dataStart];
        dataEnd = semiColonPos[[1, 1]] - 1;
        velString = StringTake[fileContent, {dataStart, dataEnd}];

        cleanString = StringReplace[velString, {"\n" -> "", " " -> ","}];
        rawFlatVels = Flatten[ImportString[cleanString, "CSV"]];

        (* Topology Auto-Solver *)
        nFrames3 = 100;
        nDims3 = 3;
        actualCount = Length[rawFlatVels];
        nAtomsTotal = actualCount / (nFrames3 * nDims3);

        (* Composition: 1224 Methanol (6 sites), Rest is Water (3 sites) *)
        nSoluteMols = 1224;
        nSitesSolute = 6;
        atomsFromSolute = nSoluteMols * nSitesSolute;
        atomsRemaining = nAtomsTotal - atomsFromSolute;
        nWaterMols = atomsRemaining / 3;

        If[IntegerQ[nWaterMols] && nWaterMols > 0,
            (* Constants *)
            massC = 12.01; massH = 1.008; massO = 16.00;
            amuToKg = 1.66054*10^-27;
            kB = 1.380649*10^-23;
            velConv = 100.0;

            (* Construct Mass Array *)
            mMethanol = Flatten[ConstantArray[{massC, massH, massH, massH, massO, massH},
     nSoluteMols]];
            mWater = Flatten[ConstantArray[{massO, massH, massH}, nWaterMols]];
            massArray3 = Join[mMethanol, mWater];

            (* Process Velocities *)
            vels3Unscaled = Partition[Partition[rawFlatVels, nDims3], nAtomsTotal];
            scaleFactor = 20.455; (* Conversion factor *)
            vels3 = vels3Unscaled * scaleFactor;

            (* Calculate Temperature with SHAKE Correction Factor (18/14) *)
            temp3 = Table[
                frameV = vels3[[t]];
                speedSq = Total[frameV^2, {2}] * (velConv^2);
                ke = 0.5 * (massArray3 * amuToKg) * speedSq;
                ((2 * Total[ke]) / (3 * nAtomsTotal * kB)) * (18.0/14.0),
                {t, 1, nFrames3}
            ];

            Print[ListLinePlot[Transpose[{Range[nFrames3], temp3}],
                PlotLabel -> "Set 3: True Temperature (Velocities)", Frame -> True,
                FrameLabel -> {"Frame", "Temp (K)"}, PlotStyle -> Purple]];
        ];
    ];
];
```

Listing 3: Parsing velocity text dumps, solving topology, and calculating temperature

B.4 Sensitivity & Constraint Analysis

```
(* --- 5. VARIABLE DOF ANALYSIS (14-18) --- *)

Print[Style["\n> Variable DOF Analysis (14-18)", "Section"]];

(* 1. Calculate Temperature curves for DOF=14 to 18 *)
(* Note: Uses variables from previous section (vels3, massArray3, etc.) *)
dofCurves = Table[
    Table[
        frameV = vels3[[t]];
        speedSq = Total[frameV^2, {2}] * (velConv^2);
        ke = 0.5 * (massArray3 * amuToKg) * speedSq;

        (* Test Formula: Scale T by (18/d) to test DOF assumption *)
        ((2 * Total[ke]) / (3 * nAtomsTotal * kB)) * (18.0 / d),
        {t, 1, nFrames3}
    ],
    {d, 14, 18}
];

(* 2. Reference Line at T = 297.55 K *)
refLine = ConstantArray[297.55, nFrames3];

(* 3. Plot Comparison *)
```

```
Print[ListLinePlot[Append[dofCurves, refLine],
    PlotRange -> All, Frame -> True,
    FrameLabel -> {"Frame", "Temperature (K)"},
    PlotLabel -> "Temperature Evolution (DOF 14-18)",
    PlotLegends -> Append[Table["DOF " <> ToString[d], {d, 14, 18}], "T = 297.55K"],
    PlotStyle -> Append[Table[Automatic, {5}], Directive[Black, Dashed, Thick]],
    ImageSize -> Large
]];

(* --- 6. HYPOTHESIS TEST: O-H BOND LENGTH --- *)

(* 1. Select a molecule (e.g., Index 1) *)
molIndex = 1;

(* 2. Define Atom Indices (C=1, H=2,3,4, O=5, H=6) *)
atomO = 5;
atomH = 6;

(* 3. Extract Trajectories from Fixed Data *)
(* xyzFix1 shape: {Frames, Atoms, 3} *)
trajO = xyzFix1[[All, (molIndex - 1) * 6 + atomO]];
trajH = xyzFix1[[All, (molIndex - 1) * 6 + atomH]];

(* 4. Calculate Euclidean Distance per frame *)
bondLengths = MapThread[EuclideanDistance, {trajO, trajH}];

(* 5. Statistics *)
meanBond = Mean[bondLengths];
stdDev = StandardDeviation[bondLengths];

(* 6. Visualization *)
Print[ListLinePlot[bondLengths,
    PlotLabel -> "O-H Bond Length (Molecule " <> ToString[molIndex] <> ")",
    Frame -> True, FrameLabel -> {"Frame Number", "Length (A)"},
    PlotStyle -> {Red, Thickness[0.005]},
    Epilog -> {Black, InfiniteLine[{{0, meanBond}, {1, meanBond}}]}
]];
```

Listing 4: Variable Degree of Freedom Analysis and Bond Length Verification