Viscosity Calculation of Toluene System via MD Simulations:

Step-by-Step Approach

Definition of Viscosity for a Continuous System

Unlike **self-diffusivity**, which is a **single-particle property**, viscosity is a **collective property** and cannot be improved by averaging over all particles in the system.

Viscosity (η) is a measure of a fluid's **internal resistance to flow** under an applied **shear stress**. It quantifies the **extent to which a fluid resists deformation** due to relative motion between its layers.

Mathematically, viscosity is defined as:

$$\eta = \frac{\tau}{\dot{\gamma}}$$

Where:

- η = Dynamic viscosity (Pa·s or N·s/m^2)
- τ = Shear stress (force per unit area, N/m^2 or Pa)
- $\dot{\gamma}$ = Shear rate (rate of deformation, s^{-1})

Green-Kubo relation

There have been several different atomistic simulation methods developed for computing the shear viscosity of liquids. Due to its simplicity, the Green−Kubo relation based on equilibrium molecular dynamics (MD) simulations is perhaps the most widely used method.1 In the Green−Kubo approach, the shear viscosity is calculated from the integral over time of the pressure tensor autocorrelation

$$\eta = \frac{V}{k_B T} \int_0^\infty \langle P_{\alpha\beta}(t) \cdot P_{\alpha\beta}(0) \rangle \, dt$$

where V is the system volume, k_B is the Boltzmann constant, T is temperature, Pαβ denotes the element αβ of the pressure tensor, and the angle bracket indicates the ensemble average. Theoretically, the pressure tensor autocorrelation function decays to zero in the long-time limit and the integral in above equation will reach a constant value, which corresponds to the calculated shear viscosity.

So, for calculating viscosity we are requiring pressure tensor or stress tensor, here we are using Molecular Dynamics Simulation for calculating shear stress.

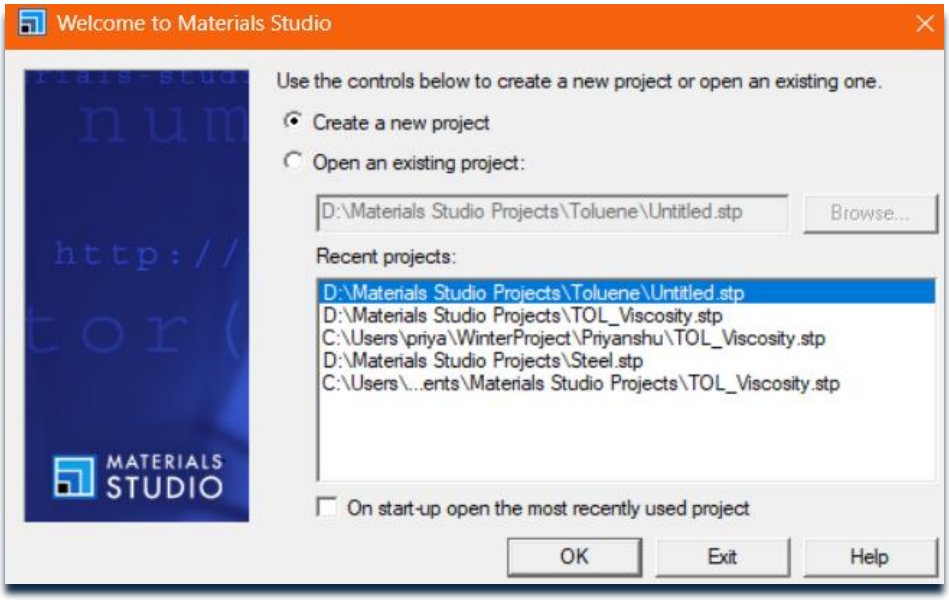
Molecular Dynamics Simulation of a Toluene System for Shear Stress Calculation in Material Studio

To perform a molecular dynamics simulation and extract stress tensor data, we first need to construct a well-defined toluene system. This involves several critical steps to ensure accuracy and reliability in our results.

Step 1: Creating a Single Toluene Molecule

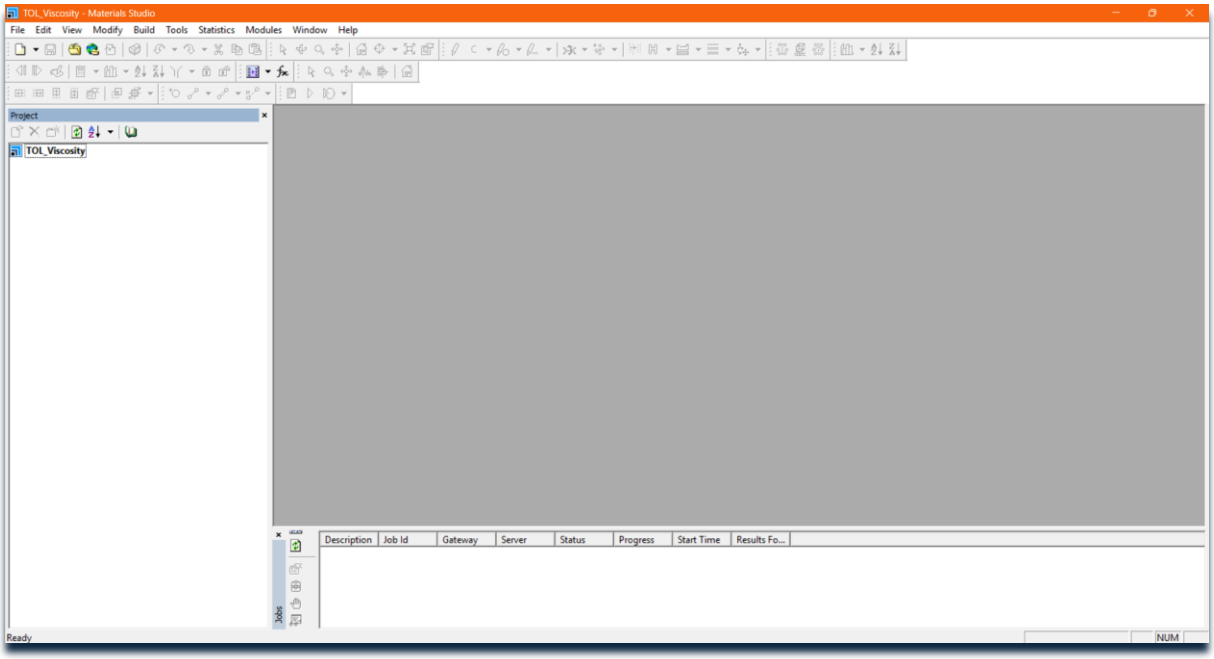
1. Open the **Materials Studio** software. **(**Ensure that Materials Studio is installed on your PC with a valid license before opening the software.)

2. The process begins with generating a single toluene molecule within Materials Studio. Open a **new project** or open an existing project if you want to run the simulation within a previously created workspace (shown below). (Instruction here is for new projects but it is similar for existing projects).

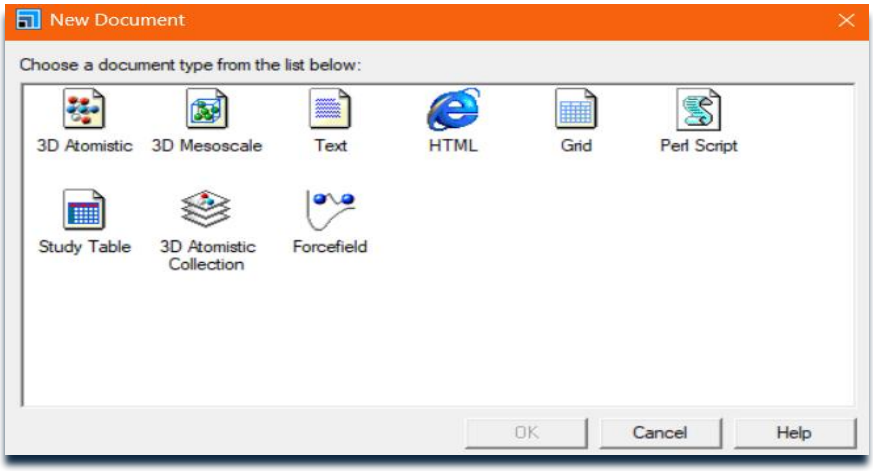


3. Click **OK** on the initial dialog box. You will be prompted with a wizard to choose the file location for your new project.
   Select a folder where you want to save your project and enter a project name, for example, **"TOL_Viscosity"**.

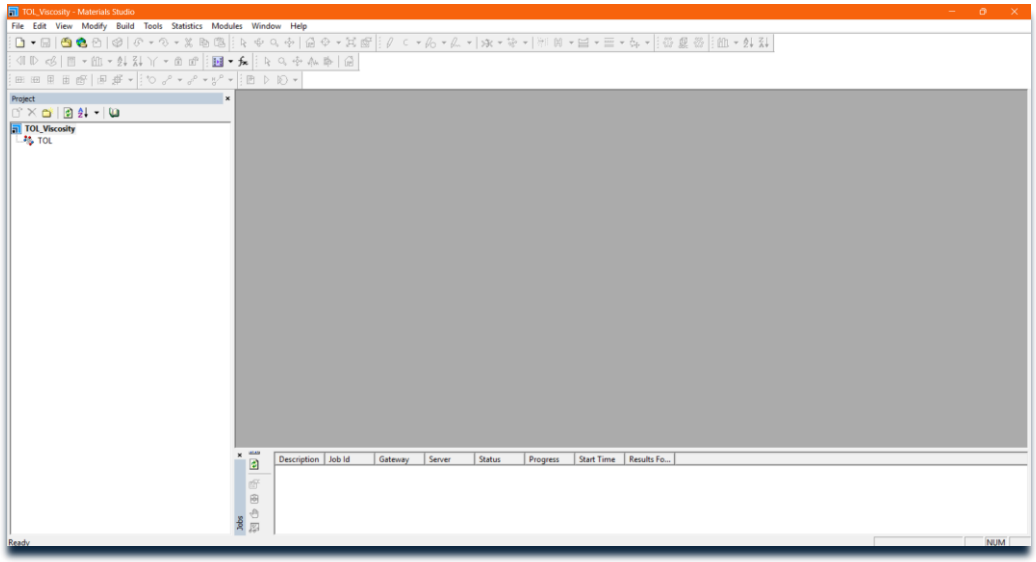
4. Once completed, you'll see the Materials Studio workspace as shown below, with your project (Toluene) listed in the **Project Explorer** pane on the left:



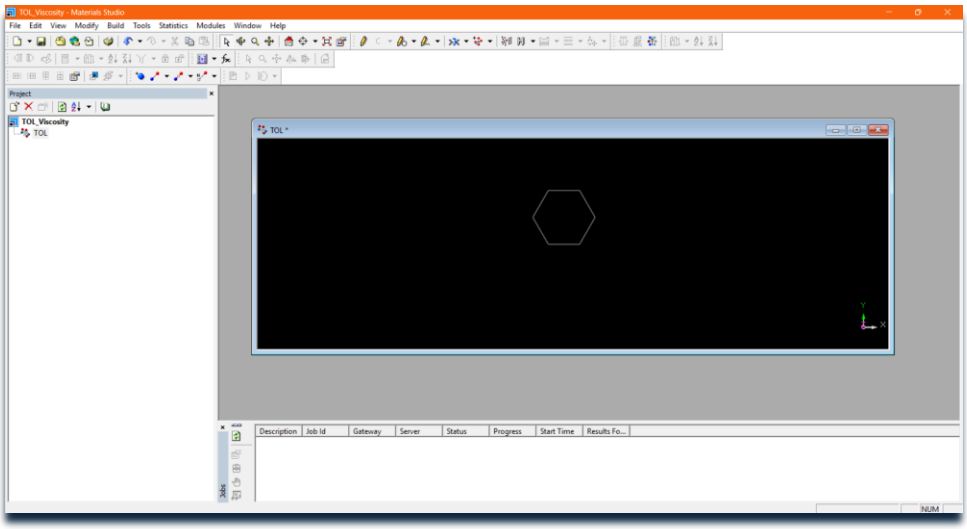
5. Now click on the **File** option (leftmost option) in the navigation bar, choose **new** from the drop menu. On completed you'll see following Materials Studio workspace. Click on **3D Atomistic** for creating molecule for the project.



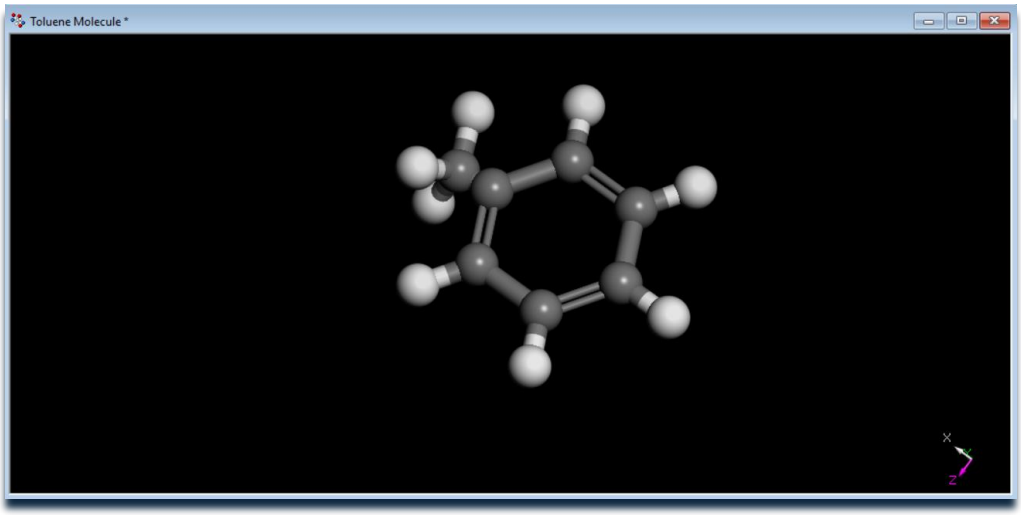
6. After following above steps you'll below workspace you can rename **3D Atomistic** (e.g. TOL).



7. Now we need to create single **Toluene** molecule. From the top menu hover on **Sketch Ring** this will show option for creating **Carbon** atom rings of 4, 5 or 6 atoms. Click on **6 member**, this will create a ring with 6 carbon atoms.



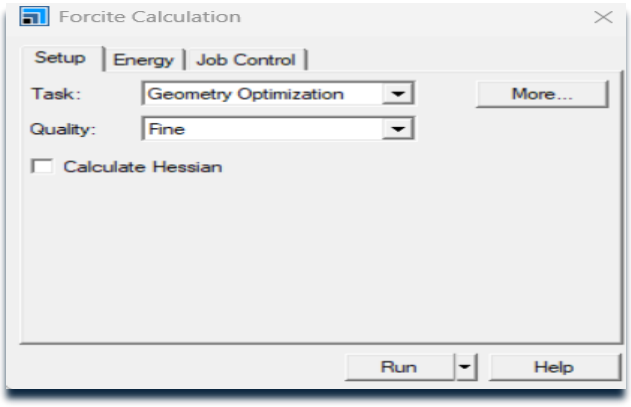
8. Now convert alternate single bond to double bond in the rings using **6th** option right to ring creation icon. Also add another carbon atom using option just left the above option and connect this to one of the six atoms of the ring. (will look like below) Now balance hydrogen by option given in the top menu. Also, the view can be changed to **ball and stick** (right click then a menu for view change will appear).



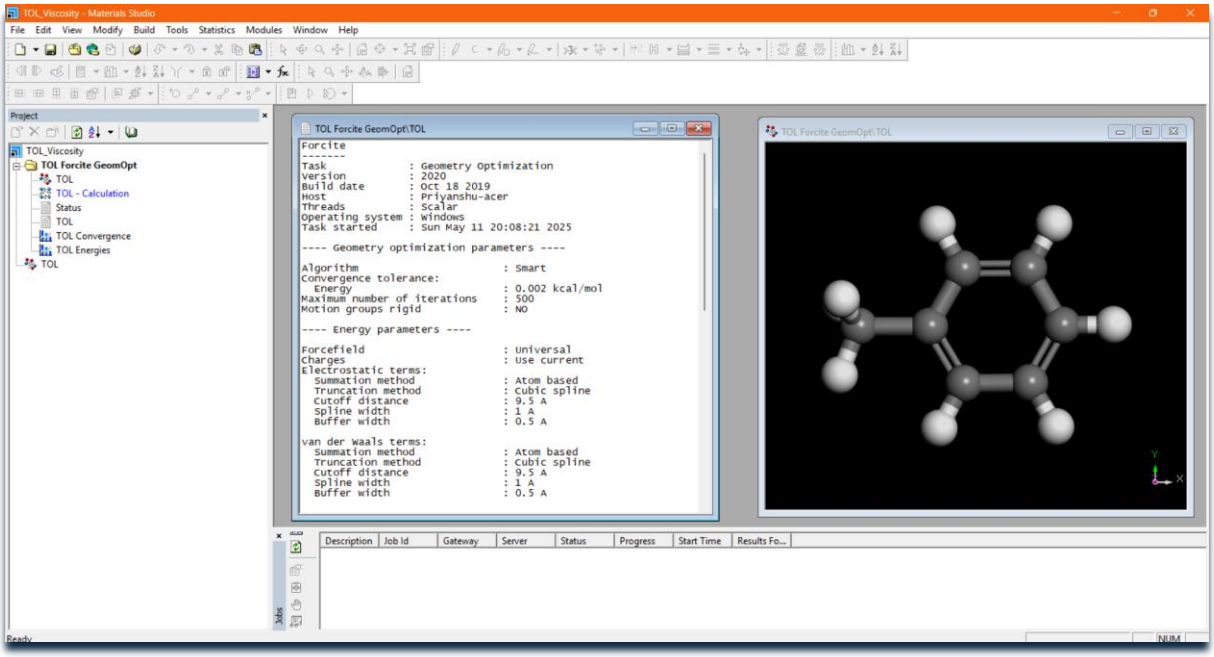
Step 2: Geometrical Optimization of the Toluene Structure

After creating a single toluene molecule in **Material Studio**, the next crucial step is **geometrical optimization**. This ensures that the molecular structure is stable and represents a realistic configuration before proceeding to system construction and molecular dynamics simulation.

1. Click on **Modules -> Forcite -> Calculation.** This will open a menu for selection of **Task** and **Quality** of the task, choose “Geometry optimization” in the task and choose quality as per need. Then click on run to start the task.



2. Upon completion, Material studio will generate a directory having the results of the geometry optimization.

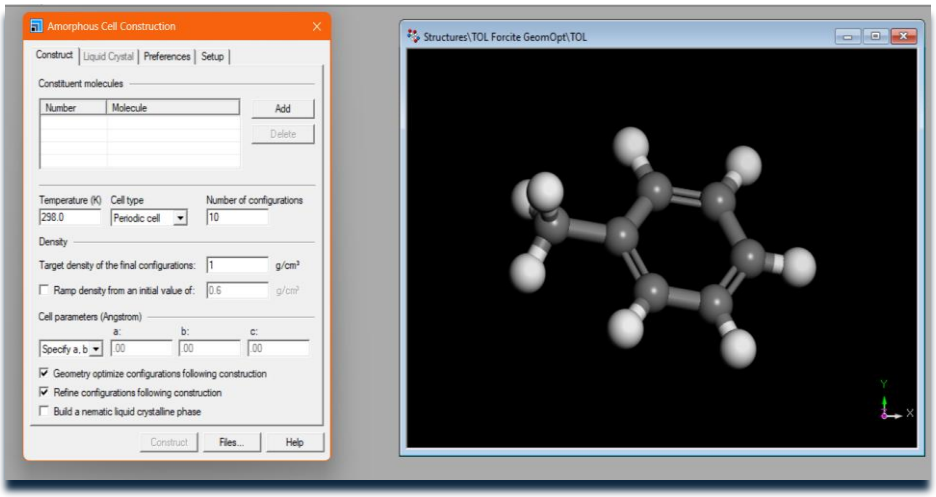


The toluene molecule consists of a benzene ring with a methyl (-CH_3) group attached. Ensuring correct bond lengths, angles, and torsions is essential for a realistic representation. Before proceeding further, the molecule should be **energy minimized** using a suitable force field (e.g., COMPASS or Universal). This eliminates steric clashes and ensures a stable conformation for subsequent simulations. Once the single molecule is optimized, it serves as the fundamental unit for constructing a larger system, which will be used to analyse shear stress components under simulated conditions.

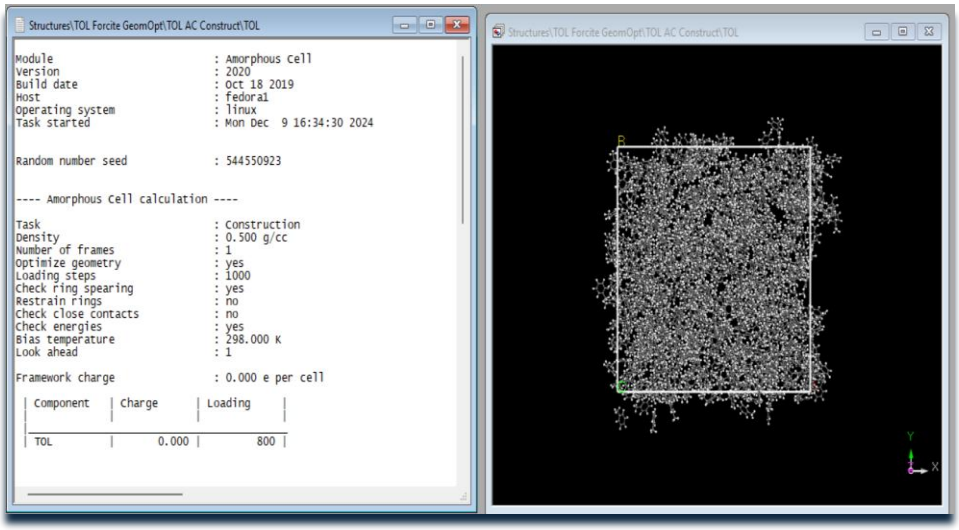
Step 3: Constructing the Toluene System – Amorphous Cell Creation

With the geometrically optimized **single toluene molecule** ready, the next step is to build a **bulk molecular system** that serves as the foundation for molecular dynamics simulations. This is done by generating an **Amorphous Cell** containing enough toluene molecules.

1. Open **Geometrically Optimized** molecule then go to **Modules-> Amorphous Cell-> Construction (Legacy)** this option will open following menu:



2. Select number of molecules (800 taken here) carefully for reliable result and within computational limitations. Select Temperature, Cell type, Number of configurations as require also give Ramp density. Also, in the Setup here “compass” force field is chosen. Preferences can be kept default. Keep target density low initially, after this Click on Construct. This will generate a cell with 800 molecules with target density, detail can be viewed in the report generated.



Step 4: NPT Equilibration (2 ns in NPT ensemble)

Purpose:

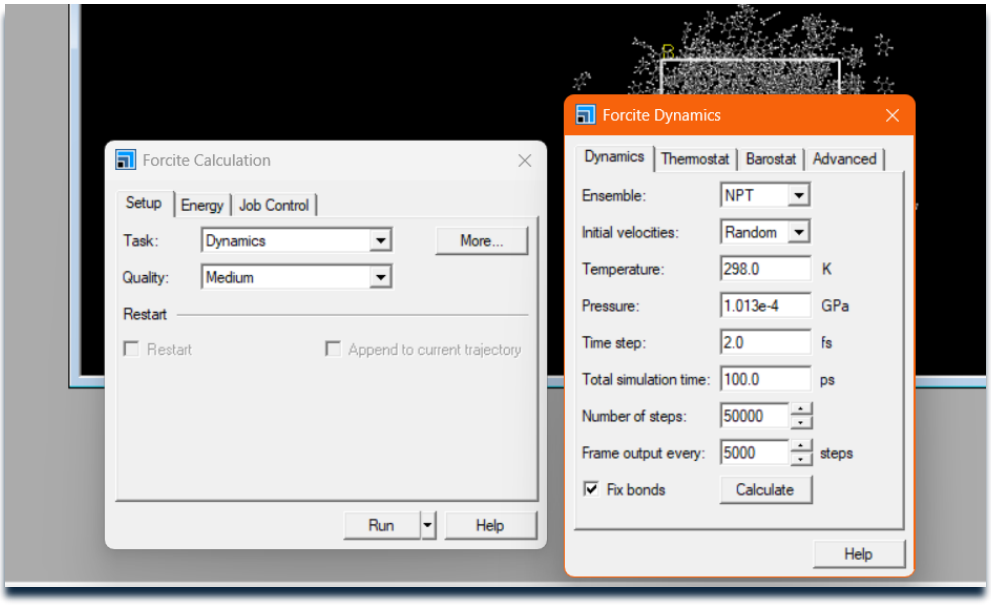
- To **relax the system under realistic conditions** of **constant pressure and temperature** so that:
- **Volume (and hence density)** of the cell adjusts naturally to reach equilibrium.
- Internal stresses, bad contacts, and high-energy conformations are minimized.

Why important:

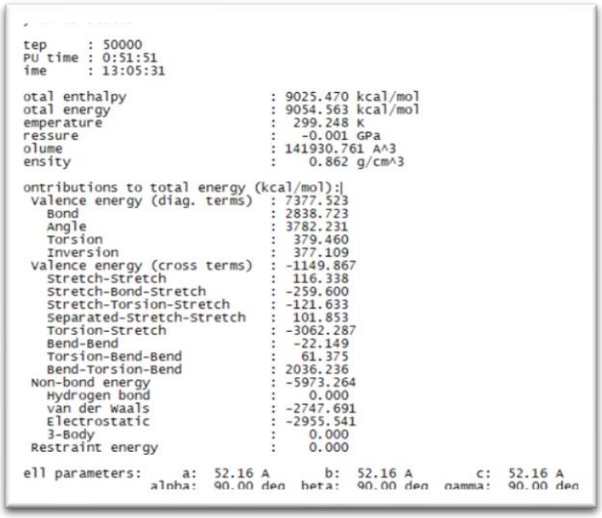
- If you started with an arbitrary cell size or density (as in amorphous cell construction), it may not be physical.
- NPT ensemble allows the **simulation box size to change**, reaching the correct equilibrium volume at the desired pressure (usually 1 atm) and temperature (e.g., 298 K).

Above cell was created with arbitrary density taken (0.5 g/cc in the above example) but we need to optimize density of this structure for further analysis.

1. Open Toluene system created in previous step. Click on **Modules-> Forcite-> Calculation** this will open a menu choose “Dynamics” in the task and at least **medium** quality, in Energy/Forcefield choose according to the project.



2. In the Dynamic pane give Initial velocities->Random, Temperature->298 K, Pressure->1 atm, Time step->2 fs, Total simulation time->100 ps.
3. In "Thermostat" choose "Berendsen" (for Toluene system, varies with molecular systems), in "Barostat" choose "Berendsen" (for Toluene system, varies with molecular systems).
4. Close the "Dynamics Tab" and click run. This will create more optimized cell (shown below). The initially assigned density was an approximation, so adjustments are required to bring it closer to the **experimental density of toluene (~0.866 g/cm³ at 25°C)**.

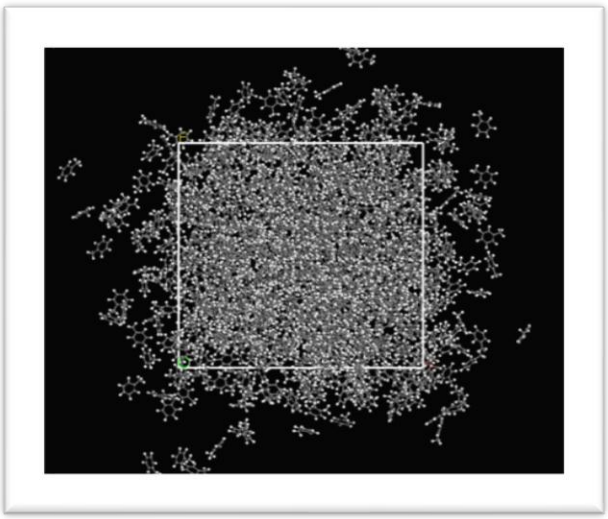


```
tep     : 50000
PU time : 0:51:51
ime     : 13:05:31

otal enthalpy                  : 9025.470 kcal/mol
otal energy                    : 9054.563 kcal/mol
emperature                     :  299.248 K
ressure                        :   -0.001 GPa
olume                          : 141930.761 A^3
ensity                         :    0.862 g/cm^3

ontributions to total energy (kcal/mol):
 Valence energy (diag. terms)  : 7377.523
   Bond                        : 2838.723
   Angle                       : 3782.231
   Torsion                     :  379.460
   Inversion                   :  377.109
 Valence energy (cross terms)  : -1149.867
   Stretch-Stretch             :  116.338
   Stretch-Bond-Stretch        : -259.600
   Stretch-Torsion-Stretch     : -121.633
   Separated-Stretch-Stretch   :  101.853
   Torsion-Stretch             : -3062.287
   Bend-Bend                   :  -22.149
   Torsion-Bend-Bend           :   61.375
   Bend-Torsion-Bend           : 2036.236
 Non-bond energy               : -5973.264
   Hydrogen bond               :    0.000
   van der Waals               : -2747.691
   Electrostatic               : -2955.541
   3-Body                      :    0.000
 Restraint energy              :    0.000

ell parameters:     a:  52.16 A       b:  52.16 A       c:  52.16 A
                 alpha:  90.00 deg  beta:  90.00 deg  gamma:  90.00 deg
```



Step 5: NVT Production Run (Canonical Ensemble)

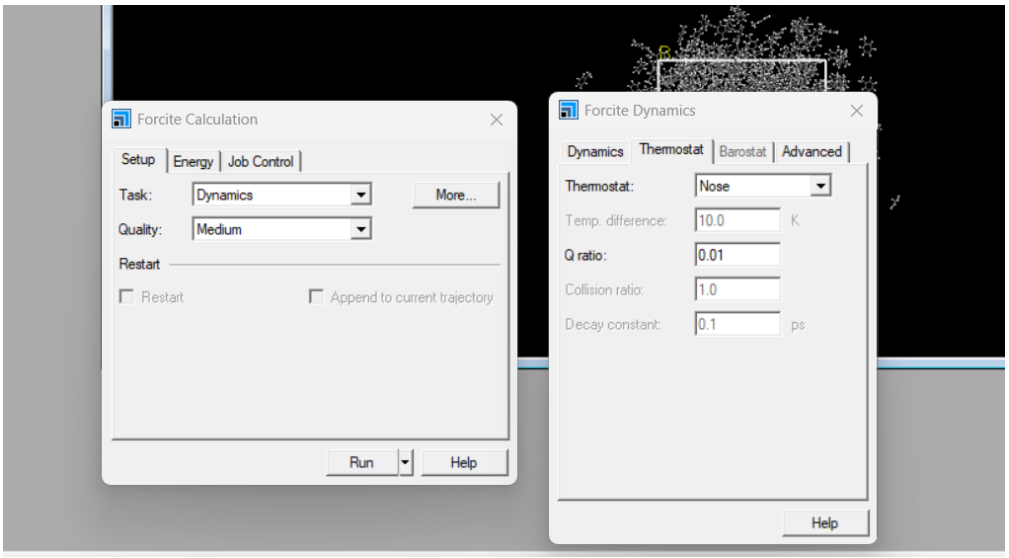
Purpose:
To **collect meaningful data** (e.g., diffusion, viscosity, RDF, etc.) under **constant volume and temperature** conditions after the system has been equilibrated.

Why important:

- In NVT, the volume is fixed — this helps ensure stable conditions for property calculations.
- The system is already relaxed and has correct density from NPT, so no need to adjust volume anymore.
- Properties like energy, molecular mobility, radial distribution functions, etc., are more stable and reliable in NVT.

Steps for NVT Run:

1. Open NPT equilibrated system. Click on **Modules-> Forcite-> Calculation** this will open a menu choose “Dynamics” in the task and at least **medium** quality, in Energy/Forcefield choose according to the project.
2. In the Dynamic pane give Initial velocities->Random, Temperature->298 K, Pressure->1 atm, Time step->2 fs, Total simulation time->100 ps.
3. In “Thermostat” choose “Nose” (for Toluene system, varies with molecular systems), keep advanced default.
4. Close Dynamics tab and click on run. This will generate the more stable and reliable system for further analysis.



Properties like shear stress, energy, molecular mobility, radial distribution functions, etc., are more stable and reliable now after NVT.

Step 6: Simulated Annealing for System Stabilization

After optimizing the **density of the toluene system**, the next step is **simulated annealing** to ensure the system reaches its most stable energy state. This process helps in refining molecular arrangements, reducing residual stress, and improving the accuracy of shear stress calculations, here why it is important:

1. Avoids Getting Trapped in Local Minima:

- Molecular systems have **complex energy landscapes** with many local minima.

- A standard MD simulation at low temperature might **settle quickly into a local minimum**, missing the true lowest-energy (most stable) configuration.
- Simulated annealing involves **heating the system up**, allowing atoms to overcome energy barriers, and then **gradually cooling it**, enabling relaxation into a better (often global) minimum.

---

2. Better Sampling of Configuration Space:

- At high temperatures, atoms explore a **wider range of configurations**.
- During cooling, the system can sample **diverse conformations** and **settle into low-energy states** that wouldn't be found through regular MD.

---

3. Useful for Structure Prediction & Optimization:

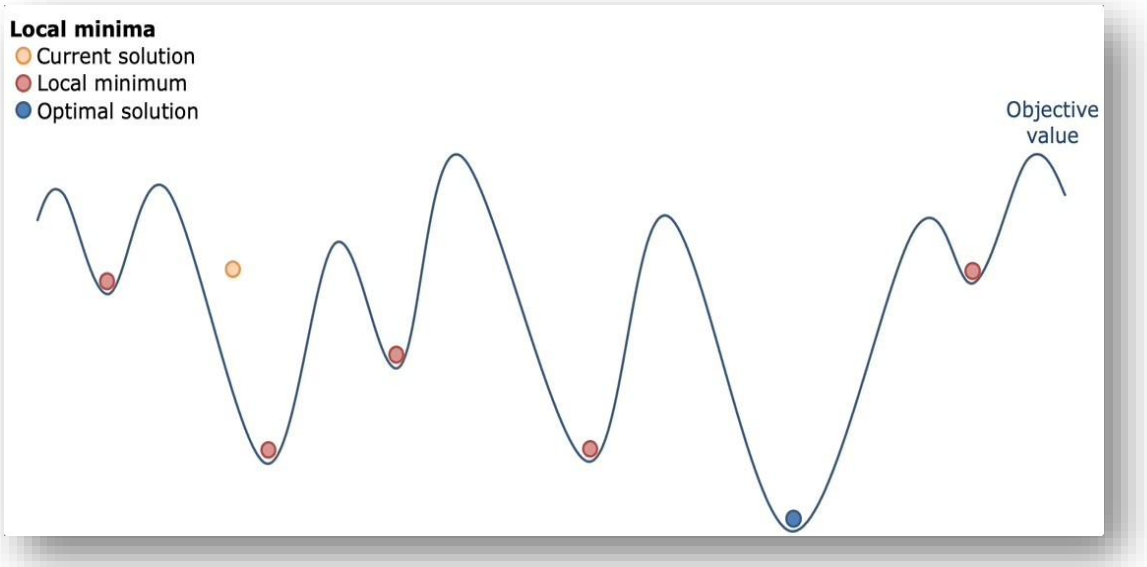
- It helps in **predicting stable molecular structures**, such as in protein folding or crystal structure prediction.

- Also used in **energy minimization of initial configurations** to improve starting points for production MD runs.

---

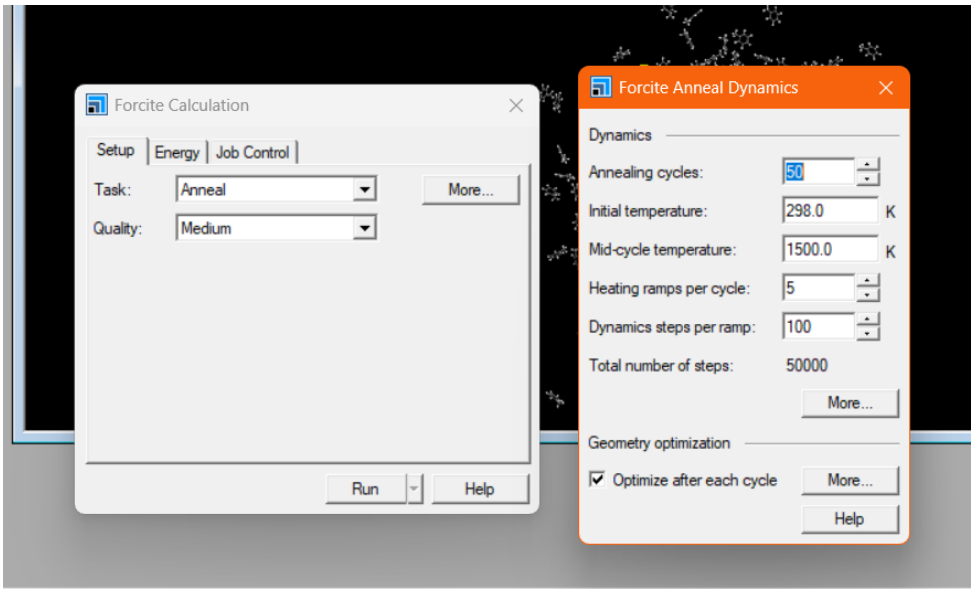
4. Improves Convergence:

- When applied carefully, simulated annealing **improves convergence** towards a thermodynamically favorable structure or phase.



Procedure for Simulated Annealing:

1. Open the NVT optimized toluene cell. Click on modules->Forcite->Calculation. Choose task as “Anneal”, quality medium (here).
2. In “Energy” tab in this example “Forcefield” is chosen “Compass”. Other options are kept default. Click on “More” (in the Setup tab).
3. Choose number of “Annealing cycles”, “Initial temperature”, “Mid-cycle temperature”, “Heating ramps per cycle”, “Dynamics steps per ramp”. Close the “More” tab and click on Run.



After simulated annealing in Materials Studio, you obtain a more stable, lower-energy configuration of your system, such as an amorphous toluene cell. The process helps overcome local energy minima by gradually heating and cooling the system, allowing atoms to rearrange into more favorable positions. The output includes the annealed structure, energy evolution data showing how potential energy decreases during temperature ramps, and optional trajectory files if enabled. This final structure is more physically realistic and can be used as input for further steps like NPT/NVT dynamics or property calculations such as viscosity, density, diffusivity, or radial distribution functions.

Step 7: Generating Independent Trajectories for Enhanced Accuracy

With the **annealed and stabilized toluene system** ready, the next crucial step is to **generate multiple independent trajectories**. Here is why large Independent Trajectories are required:

1. Ensures Statistical Reliability (Better Sampling)

- MD simulations rely on **statistical mechanics** to compute averages of physical quantities (e.g., energy, pressure, diffusion coefficients).
- Running multiple **independent trajectories** helps gather **uncorrelated data**, which improves the **accuracy and reliability** of ensemble averages.

- A single trajectory might not explore the full phase space due to being trapped in one region, especially for complex systems.

---

2. Overcomes Initial Condition Bias

- The outcome of an MD simulation can **depend strongly on initial conditions** like atomic positions and velocities.
- Generating multiple trajectories with different randomized starting conditions (e.g., different velocity seeds) helps eliminate **bias from any specific initial setup**.

---

3. Captures Rare Events

- In many systems, **rare but important events** (e.g., conformational changes, folding, defect formation) may not occur within a single trajectory.
- Independent trajectories increase the **chance of observing rare events**, improving insight into the system's behavior.

---

4. Reduces Autocorrelation Errors

- Data points in a single MD run are often **autocorrelated** (i.e., each state depends on the previous one).
- Independent runs provide **uncorrelated snapshots**, improving the **statistical independence** of data for meaningful error estimation.

---

5. Better Ensemble Representations

- For properties that depend on ensemble sampling (canonical, microcanonical, etc.), using multiple trajectories allows building a **representative ensemble**.
- This is especially crucial when studying **thermodynamic or kinetic properties**.

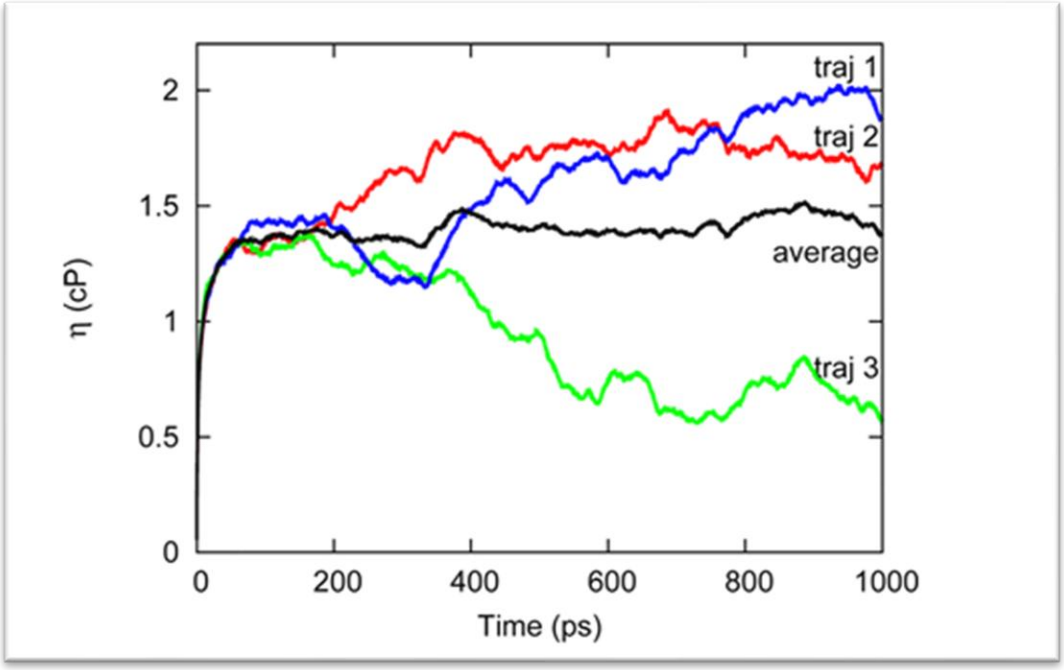


Figure: Various Independent Trajectories

Procedure for generating Trajectories:

1. Open annealed Toluene system. Now we only need to do NVT runs multiple time upon this annealed system. Follow step 5 for NVT settings.
2. Only difference here is we need to save trajectories at a particular time interval for each step below is the Perl script for one trajectory. Repeat this step to get more independent trajectories.

```
#!perl

use strict;
use Getopt::Long;
use MaterialsScript qw(:all);

my $doc = $Documents{"NVT1.xsd"};
my $results = Modules->Forcite->Dynamics->Run($doc, Settings(
    CurrentForcefield => 'COMPASSII',
    Ensemble0D => 'NVT',
    Ensemble3D => 'NVT',
    Pressure => 0.000101325,
    TimeStep => 1,
    NumberOfSteps => 2000000,
    TrajectoryFrequency => 500000,
    ReducedTrajectoryFrequency => 2,
    ReducedTrajectorySet => "",
    InitialVelocities => 'Current',
    StressXX => -0.000101325,
    StressYY => -0.000101325,
```

```
•       StressZZ => -0.000101325,
•       UseMultipleTimeSteps => 'No'));
•   my $outTrajectory = $results->Trajectory;
•
```

Until now we have seen how to generate independent trajectories of toluene system (with geometrically optimized toluene molecules, optimized density and annealed system) and why large number of trajectories is necessary. Next step is to calculate viscosity of these annealed systems. We are using Green-Kubo method to calculate viscosity of the system.

In the Green−Kubo approach, the shear viscosity is calculated from the integral over time of the pressure tensor autocorrelation function. As discussed, pressure tensor of the system or shear stress tensor of the system is required to calculate viscosity using Green-Kubo method both will work same. So, now we need to extract stress tensor from these generated trajectories.

Calculation of Viscosity from the generated trajectories:

First, we need to extract stress tensor values from the trajectory files generated by material studio:

For each trajectory we need to run the following Perl script to extract Stress Tensor,

```
# Perl script to extract and save the stress tensor for the first 5 frames of TOL.xtd to a CSV file
use strict;
use Getopt::Long;
use MaterialsScript qw(:all);
use warnings;

# Open the trajectory document Path to Trajectory Document
my $trajectoryDoc = $Documents{"NVT1 NoMatter.xtd"};
my $trajectory = $trajectoryDoc->Trajectory;
my $symmetrysystem = $trajectoryDoc->SymmetrySystem;

# Define the output file path
my $outputFilePath = "C:/<path to store csv file of stress tensor>/stress_tensor.csv";

# Open a CSV file for writing
open(my $csv_fh, '>', $outputFilePath) or die "Cannot open $outputFilePath: $!";

# Write CSV header
print $csv_fh "Frame,StressXX,StressYY,StressZZ,StressXY,StressXZ,StressYZ\n";

# Loop through all frames
for (my $frame = 1; $frame <=  $trajectory->NumFrames; ++$frame) {
   $trajectory->CurrentFrame = $frame;
   my $stressTensor = $symmetrysystem->Stress;
```

```
    # Extract stress tensor components
    my $stressXX = $stressTensor->Eij(1, 1);
    my $stressYY = $stressTensor->Eij(2, 2);
    my $stressZZ = $stressTensor->Eij(3, 3);
    my $stressXY = $stressTensor->Eij(1, 2);
    my $stressXZ = $stressTensor->Eij(1, 3);
    my $stressYZ = $stressTensor->Eij(2, 3);

    # Write stress tensor to CSV file
    print $csv_fh "$frame,$stressXX,$stressYY,$stressZZ,$stressXY,$stressXZ,$stressYZ\n";
}

close($csv_fh);
print "Stress tensor extraction complete. Results saved to $outputFilePath.\n";
```

Above Script will generate CSV file storing Stress tensor data following below formatting:

Frame	StressXX	StressYY	StressZZ	StressXY	StressXZ	StressYZ
1	-0.06665	-0.13673	-0.10209	0.014428	-0.01319	0.046994
2	-0.03544	-0.07522	-0.05382	0.018979	-0.02007	0.036212
3	0.011986	0.035665	0.012144	0.020924	-0.01581	0.023884
4	0.0184	0.108809	0.039716	0.018956	-0.00278	0.013983
5	-0.03312	0.098713	0.013485	0.011868	0.01538	0.009206
6	-0.09582	0.041265	-0.02546	-0.00103	0.033512	0.010921
7	-0.0922	0.021224	-0.00892	-0.01889	0.046392	0.017838

. . . . . .

After extraction of stress tensor from all trajectories and saving it to csv file we can now proceed to calculate viscosity of the systems.

The **shear viscosity η** is given by the **Green–Kubo formula**:

$$\eta = \frac{V}{k_B T} \int_0^\infty \langle P_{\alpha\beta}(0) \cdot P_{\alpha\beta}(t) \rangle \, dt$$

Where:

η = shear viscosity (e.g., in Pa·s)

V = volume of the system

k_B = Boltzmann constant

T = absolute temperature (in Kelvin)

$P_{\alpha\beta}(t)$ = off-diagonal component of stress tensor (typically P_{xy}, P_{yx}, P_{xz}, P_{zx}, P_{zy} or P_{yz})

$\langle \cdot \rangle$ = time autocorrelation function (ensemble average)

Practical Steps to Compute Viscosity

1. **Choose shear components**
   - Select off-diagonal stress tensor components: P_{xy}, P_{yx}, P_{xz}, P_{zx}, P_{zy}, P_{yz}
   - Extract these components over time (i.e., each frame) from your trajectory data or CSV file.
2. **Compute the autocorrelation function (ACF)**
   - For each time lag **τ**, compute:

     $$C(\tau)=\langle P_{\alpha\beta}(t) \cdot P_{\alpha\beta}(t+\tau) \rangle_t$$

   - Average over all time origins t.
3. **Numerically integrate the ACF over time**
   - Use numerical methods such as the trapezoidal or Simpson's rule to evaluate:

     $$\int_0^{\infty} C(\tau) d\tau$$

4. **Apply the Green–Kubo relation**
   - Finally, calculate viscosity using:

$$\eta = \frac{V}{k_B T} \int_0^{\infty} C(\tau)\, d\tau$$

Below is the Python program of above equations for calculating viscosity:

```
# Define Auto-Correlation Function (ACF) using FFT for efficiency
def acf(data):
    steps = data.shape[0]
    lag = steps//2

    # Nearest size with power of 2 (for efficiency) to zero-pad the input data
    size = 2 ** int(np.ceil(np.log2(2*steps - 1)))

    # Compute the FFT
    FFT = np.fft.fft(data, size)

    # Get the power spectrum
    PWR = FFT.conjugate() * FFT

    # Calculate the auto-correlation from inverse FFT of the power spectrum
```

```
    COR = np.fft.ifft(PWR)[:steps].real

    autocorrelation = COR / np.arange(steps, 0, -1)

    return autocorrelation[:lag]



# Viscosity from Green-Kubo relation

def green_kubo():
    # Calculate the ACFs
    Pxy_acf = acf(Pxy)
    Pxz_acf = acf(Pxz)
    Pyz_acf = acf(Pyz)

    # Calculate the shear components of the pressure tensor and their ACF
    if args.diag:
        Pxxyy = (Pxx - Pyy) / 2
        Pyyzz = (Pyy - Pzz) / 2
        Pxxzz = (Pxx - Pzz) / 2

        Pxxyy_acf = acf(Pxxyy)
        Pyyzz_acf = acf(Pyyzz)
        Pxxzz_acf = acf(Pxxzz)

    # Compute the average ACF
    if args.diag:
        avg_acf = (Pxy_acf + Pxz_acf + Pyz_acf + Pxxyy_acf + Pyyzz_acf +
Pxxzz_acf) / 6
    else:
        avg_acf = (Pxy_acf + Pxz_acf + Pyz_acf) / 3

    # Integrate the average ACF to get the viscosity
    timestep_sec = args.timestep * 1e-12  # Convert ps to seconds
    integral = integrate.cumulative_trapezoid(y=avg_acf, dx=timestep_sec,
initial=0)
    viscosity_gk = integral * (args.volume * 1e-30) / kBT
    return avg_acf, viscosity_gk
```

For complete Program for above calculation see following GitHub Repository:

Molecular-Dynamic-Simulation/Toluene Viscosity at main · priyanshu17291/Molecular-Dynamic-Simulation

The above program generates the **viscosity vs. time data** for the **toluene system** using the **Green–Kubo method**. Similarly, we will calculate the viscosity-time profile for **all independent trajectories**, ensuring statistical robustness of our results.

Now that we have **viscosity vs. time data for each independent trajectory**, we can proceed with further analysis:

- **Plotting the viscosity vs. time data** for all trajectories allows us to **visually assess convergence behavior** and **identify the time window** over which the system approaches a steady state (plateau region).
- A consistent plateau in the viscosity values across trajectories indicates that the simulation has run for a **sufficiently long time** and that **statistical averaging** is meaningful.
- From the plateau region, we can extract the **final viscosity value** for each trajectory.
- By calculating the **mean viscosity** and its **standard deviation** across all trajectories, we obtain the **ensemble-averaged viscosity** along with an estimate of the **statistical uncertainty**.

Python Script for plotting data:

```
import os

import pandas as pd
import matplotlib.pyplot as plt

# input method GK or Einstein
method = input("Enter the method (GK or Einstein): ")

# Define the path to the output CSV file
input_csv = f"Trajectory_Analysis_CSV_Files/avg_min_max_visc_{method}.csv"

# Define the output directory for saving the plot
output_plot_dir = "Plots"
os.makedirs(output_plot_dir, exist_ok=True)

# Read the CSV file
data = pd.read_csv(input_csv)

# Extract the "time(ps)" column
time = data["time(ps)"]

# Convert Average, Minimum, and Maximum viscosity columns from Pa·s to mPa·s
data["Average Viscosity (mPa.s)"] = data["Average Viscosity (Pa.s)"] * 1000
data["Minimum Viscosity (mPa.s)"] = data["Minimum Viscosity (Pa.s)"] * 1000
data["Maximum Viscosity (mPa.s)"] = data["Maximum Viscosity (Pa.s)"] * 1000

# Plot Average, Minimum, and Maximum viscosity columns
plt.figure(figsize=(12, 8))

# Fill the area between Minimum and Maximum viscosities with light yellow
plt.fill_between(
    time,
```

```
    data["Minimum Viscosity (mPa.s)"],
    data["Maximum Viscosity (mPa.s)"],
    color="lightblue",
    label="Range (Min-Max)"
)

# Plot the Average Viscosity line in royal blue
plt.plot(time, data["Average Viscosity (mPa.s)"], label="Average Viscosity",
color="royalblue", linewidth=2)

# Customize the plot
plt.xlabel("Time (ps)", fontsize=14)
plt.ylabel("Viscosity (mPa.s)", fontsize=14)
plt.title("Average, Min, and Max Viscosity vs Time (in mPa.s)", fontsize=16)
plt.legend(loc="best", fontsize=12)
plt.grid(True)

# Save the plot
output_plot_path = os.path.join(output_plot_dir, f"avg_min_max_visc_{method}.png")
plt.savefig(output_plot_path)
plt.close()

print(f"Plot saved: {output_plot_path}")
```

Below is the plot of viscosity vs. time for 40 independent trajectories:

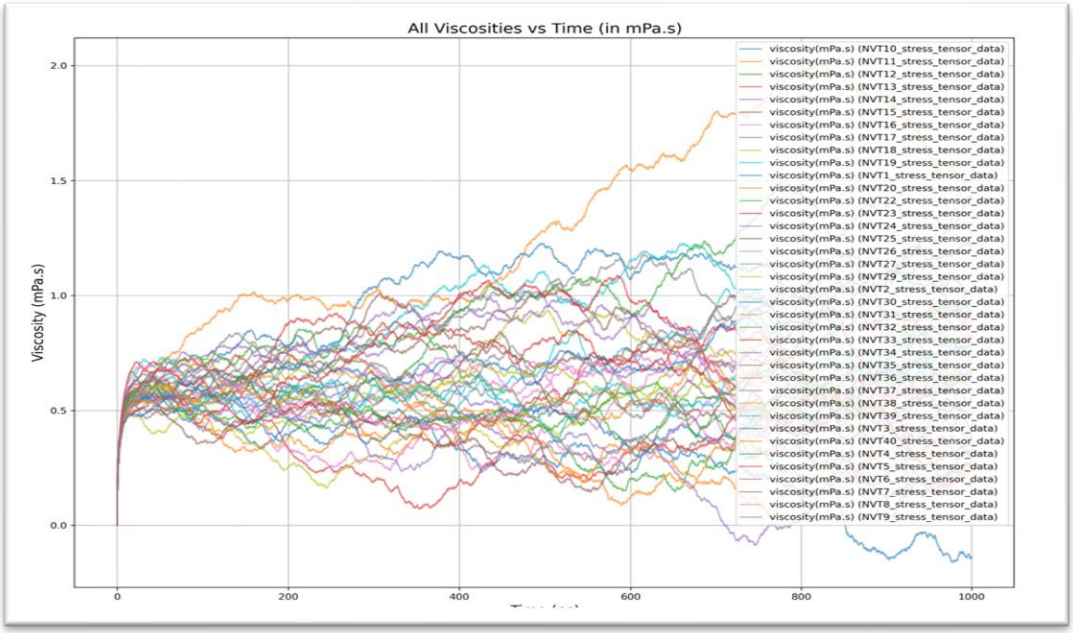


Fig: Viscosity vs. Time of 40 NVT Trajectories

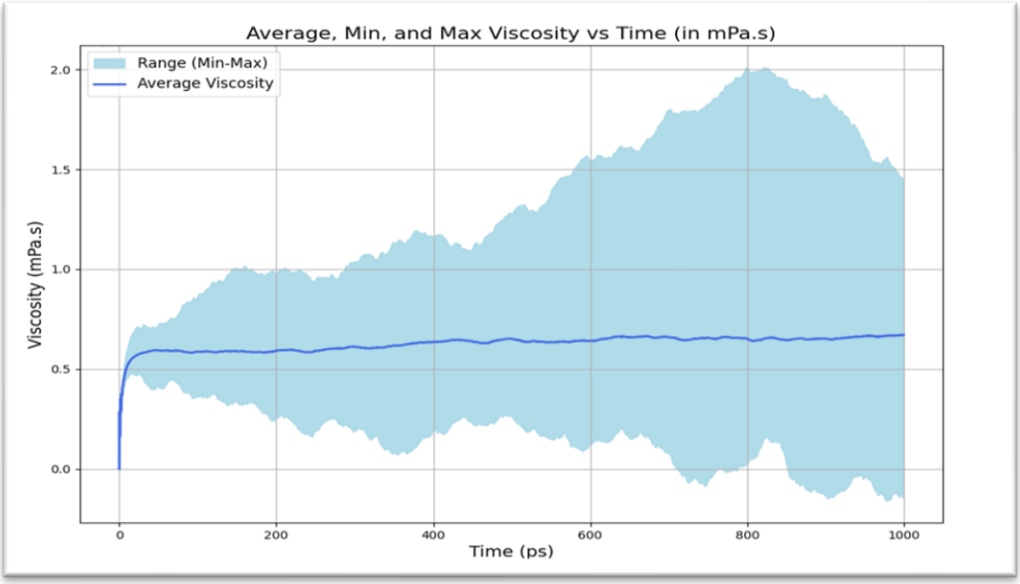


Fig: avg, max and min Viscosity vs time of 40 trajectories

Standard deviation of the Viscosity vs. Time data:

- Calculate the average of the running integrals over N trajectories ⟨η(t)⟩ and the standard deviation, which is a function of time:

$$\sigma(t) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\eta(t)_i - \langle\eta(t)\rangle)^2}$$

- Fit the standard deviation to a power law function $\sigma(t) = A.t^B$

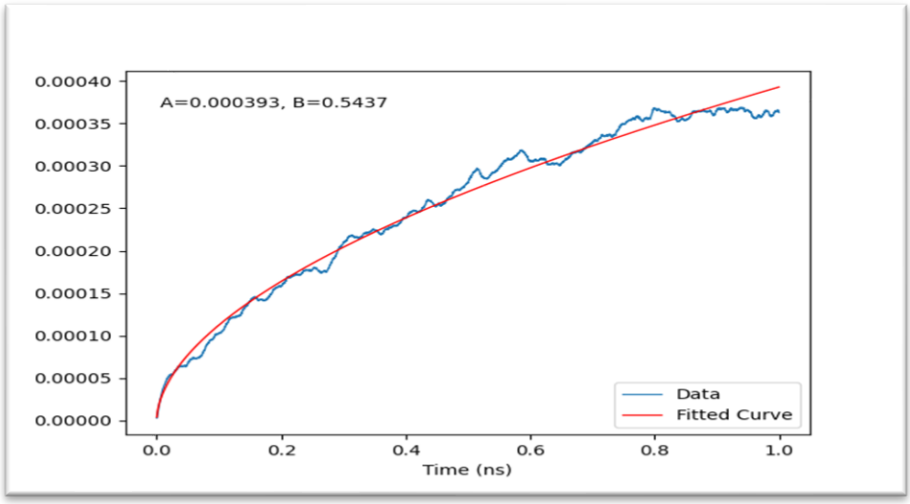


Fig.: Standard Deviation vs Time

Why analysis of Average, Maximum (at each frame) and Minimum (at each frame) viscosity, standard deviation calculation is required:

1. Assessing Convergence of the Viscosity

- **Mean viscosity vs. time** allows you to observe whether the simulation has **reached a steady value (plateau)**, indicating convergence.
- If the mean continues to fluctuate heavily, it suggests the simulation time is insufficient.

---

2. Estimating Statistical Uncertainty

- **Standard deviation at each time frame** quantifies the **spread of viscosity values** across independent trajectories.
- A **narrow standard deviation band** indicates high confidence in your average viscosity estimate.
- A **wide band** suggests either more sampling is needed or that the system exhibits significant time-dependent fluctuations.

---

3. Identifying Outliers or Anomalous Trajectories

- Plotting **min and max** values alongside the mean helps detect if **any trajectory deviates strongly** from the rest.
- Persistent deviation in min/max could signal:
    - Poor equilibration
    - Numerical artifacts
    - Rare events affecting transport properties

---

4. Time Window Selection for Final Viscosity

- These plots help you **choose the plateau region** more objectively — where mean stabilizes and std. deviation becomes relatively flat.
- You can then average only over that time window to get the **final reported viscosity**.

---

5. Verifying Ensemble Sampling Quality

- If independent trajectories give consistent results (mean ± std. dev. overlap), it validates your **ensemble sampling approach**.
- Inconsistent behavior might mean you need more independent runs or better initial conditions.

---

6. Improving Reproducibility and Reliability

- This statistical treatment **enhances the credibility** of your simulation data.
- It shows due diligence in verifying that results are **not artifacts of noise** or specific initial conditions.

Now the question is the viscosity of the system. Studies have indicated that the time when σ(t) is about 40% of ⟨η(t)⟩ is a good choice. Take the long time limit of the fitted double-exponential function as the calculated viscosity. So, we need to fit the the double exponential function and then need to calculate the viscosity at time $t = t_{cut,}$ where t_{cut} is the time at which the standard deviation σ(t) is about 40% of the average viscosity.

$$\eta(t) = A\alpha\tau_1(1 - e^{-t/\tau_1}) + A\,(1 - \alpha)\,\tau_2(1 - e^{-t/\tau_2})$$

where A, α, τ_1, and τ_2 are fitting parameters.

Below is the python script for double exponential fitting:

```
      import numpy as np

import pandas as pd
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

def double_exponential(t, A, a, T1, T2):
    return A*a*T1*(1 - np.exp(-t / T1)) + A*(1 - a)*T2*(1 - np.exp(-t / T2))

inputFile = input("Enter method (Green-Kubo or Einstein): ")
t_cut = int(input("Enter t_cut row number: "))
data = pd.read_csv(f"Trajectory_Analysis_CSV_Files/std_{inputFile}.csv")

# Select only the first 1661 rows for fitting
data_subset = data.iloc[:t_cut]
time = data_subset["time(ns)"].values
visc = data_subset["mean_visc"].values

initial_guess = [0.0003, 0.5, 0.001, 0.001]
params, _ = curve_fit(double_exponential, time, visc, p0=initial_guess)

print("Fitted parameters:")
print(f"A = {params[0]}")
print(f"a = {params[1]}")
print(f"T1 = {params[2]}")
print(f"T2 = {params[3]}")

plt.plot(time, visc*1000, label="Data", linewidth=1)
plt.plot(time, double_exponential(time, *params) * 1000, color="red",
label="Fitted Curve", linewidth=1)
```

```
plt.text(0.05, 0.9, f"A={params[0]:.4g}, a={params[1]:.4g},
T1={params[2]:.4g}, T2={params[3]:.4g}", transform=plt.gca().transAxes,
fontsize=10)
plt.xlabel("Time (ns)")
plt.ylabel("Mean Viscosity (mPa.s)")
plt.legend()
plt.savefig(f"plots/fitted_avgvisc_{inputFile}.png")
plt.close()
```

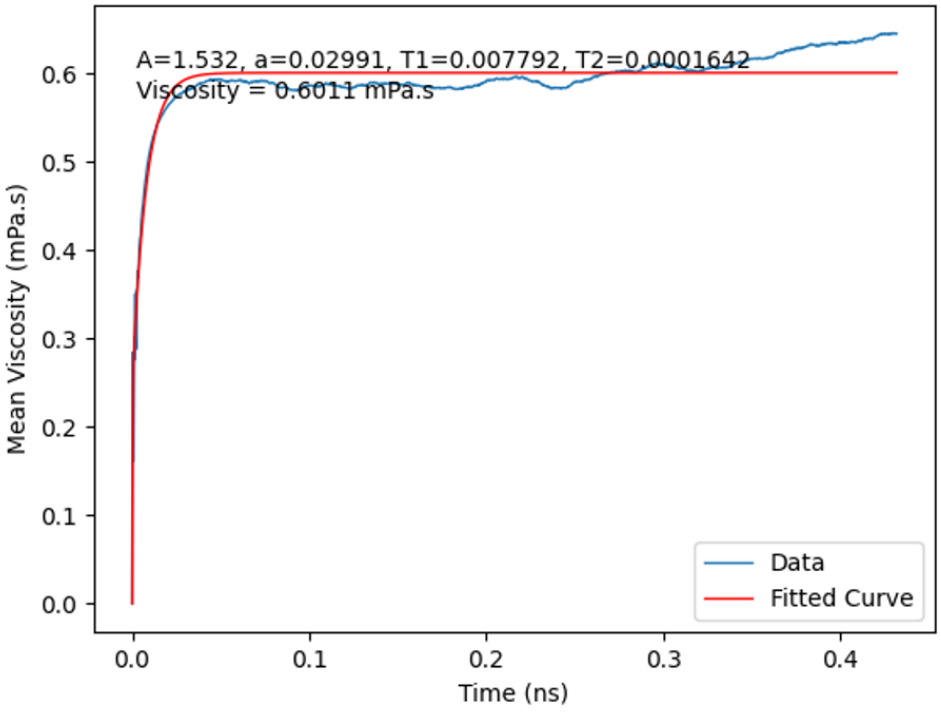


Fig: Double Exponential fitted Viscosity vs Time

Results:

The shear viscosity obtained after fitting the running integral is **0.6011 mPa·s**, which is in close agreement with the experimentally measured value of **0.554 mPa·s**. This small deviation can be attributed to factors such as finite simulation time, system size effects, and statistical uncertainties inherent in molecular dynamics simulations.

To improve the accuracy of the computed viscosity, a larger dataset with an extended trajectory can be utilized. Increasing the number of sampled data points will help reduce fluctuations in the pressure tensor autocorrelation function, leading to a more reliable estimation of viscosity. Additionally, averaging over multiple independent simulations can further minimize statistical errors and improve convergence toward the experimental value.

References:

- Reliable Viscosity Calculation from Equilibrium Molecular Dynamics Simulations: A Time Decomposition Method Yong Zhang, Akihito Otani, and Edward J. Maginn
- Best Practices for Computing Transport Properties. Self-Diffusivity and Viscosity from Equilibrium Molecular Dynamics Edward J. Maginn, Richard A. Messerly, Daniel J. Carlson, Daniel R. Roe, J. Richard Elliott
- Link to GitHub Repository: Molecular-Dynamic-Simulation/Toluene Viscosity at main · priyanshu17291/Molecular-Dynamic-Simulation