Lab Notes
(08/15/2025)

Igor Benderskii

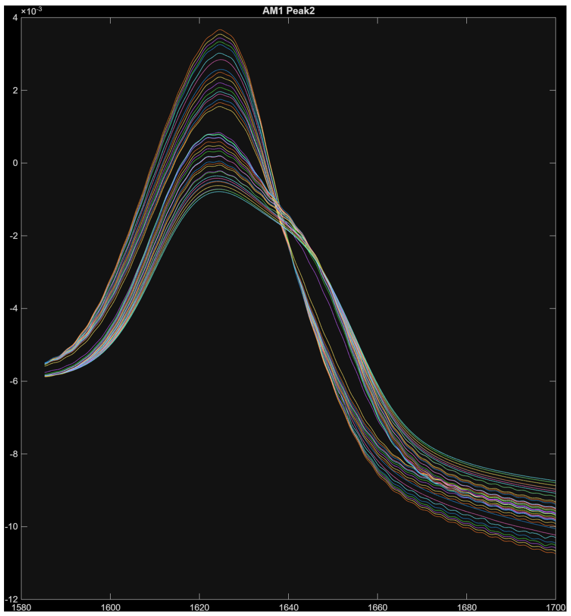
1 Wet Lab

1.1 Goals and Methods

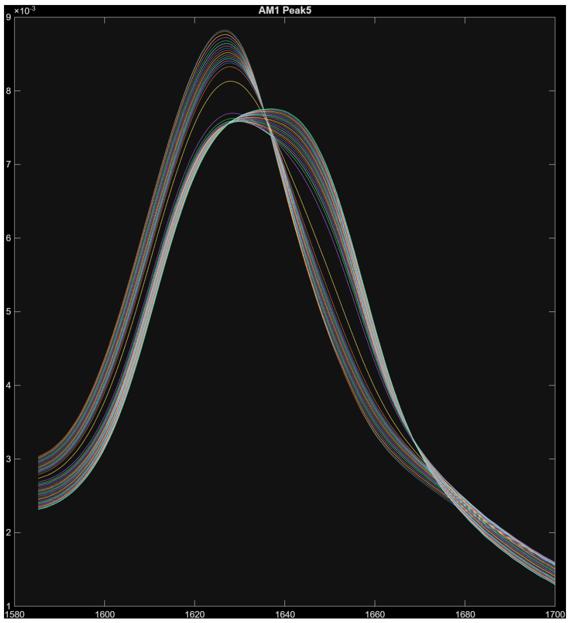
I was tasked with seeing how PNIPAM (300kDa)'s FTIR spectrum changed with temperature and with the concentration of polymer. I took T Ramp spectra from $20° - 60°C$, varying concentrations from $0.05\% \rightarrow 10\%$, with denser sampling of concentrations below 1%. I looked at the AM 1 peak specifically, and used some SVD techniques to look at how temperature affected the AM 1 peak shift, and to extract the transition temperatures. I took a T Ramp of D2O to use as a baseline subtraction, although I realized after that it really did not help.

1.2 Examples

Here are a few example normalized AM 1 peaks for different weight percents.



(a) 0.5wt



(b) 3wt

Here is an example of some different SVD components I looked at.

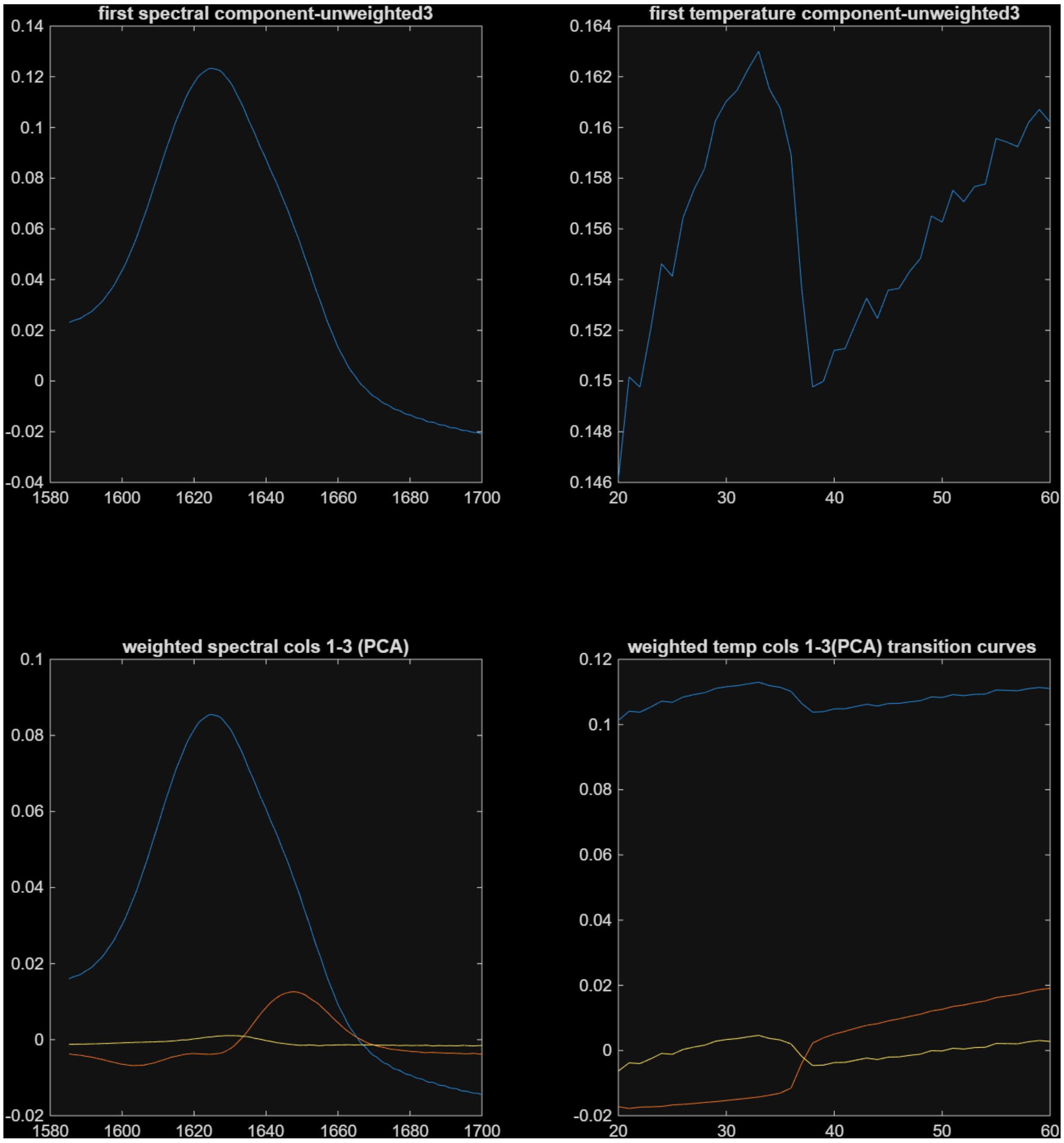
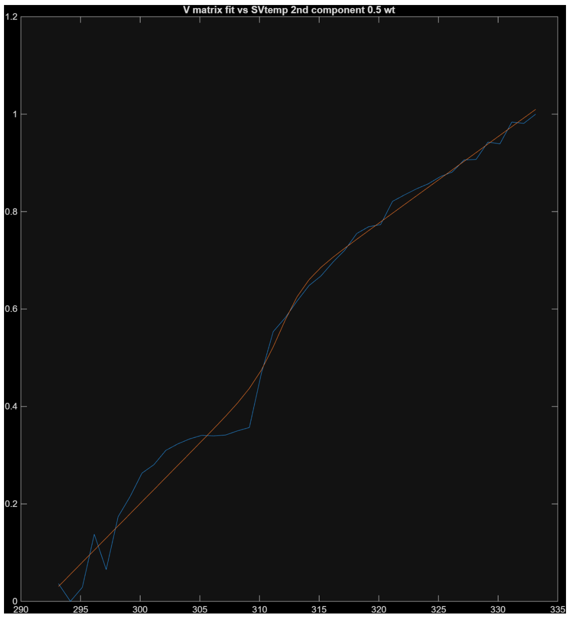
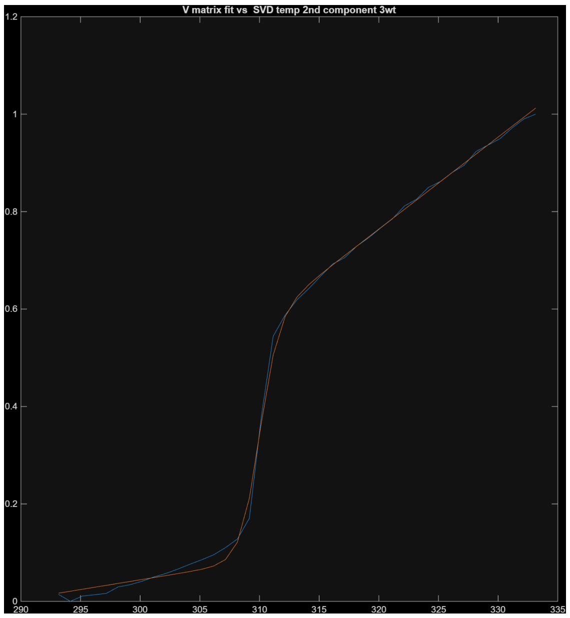


Figure 2: 3wt SVD

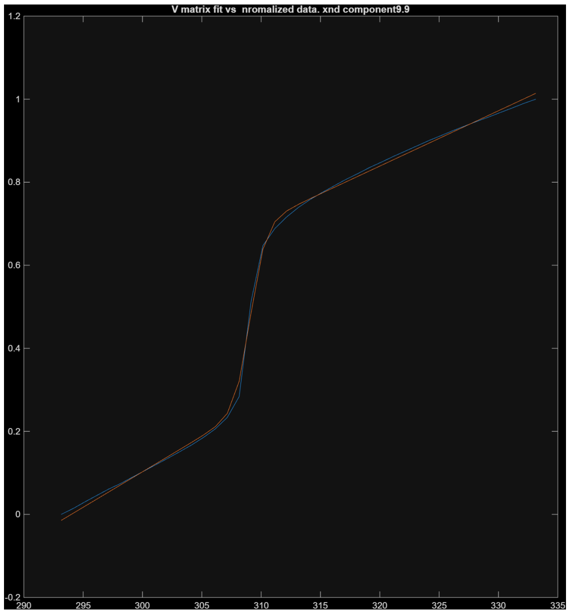
Here are some Temperature matrix SVD 2nd components fitted to a two state model for a few different concentrations to show the trend I was seeing, the lower concentrations have a smoother second component, with the biggest jump between states being around 4%:



(a) 0.5wt



(b) 3wt



(c) 9wt

I fitted the second temperature component, the one that we expect to reveal a phase transition, to a two state model (the two state AON script Sam gave me). Here are all the fitted second temp components together:

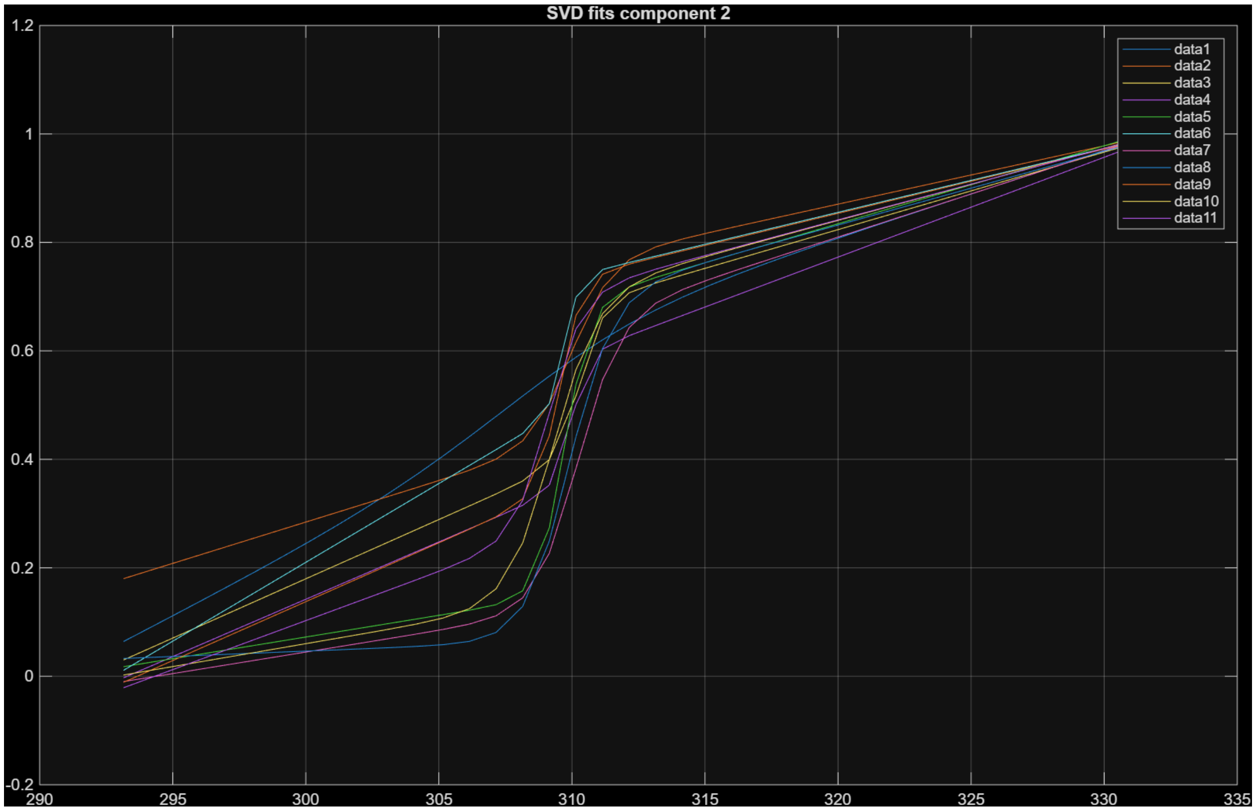


Figure 4: Second components

The biggest jumps between transitions are at around 3 - 4 percent.

Here are the transition temps plotted: At weights lower than 3, the svd started to look funny and the fit had a hard time finding where the transition occured, because the jump got smaller and looked more like a random fluctuation. From looking at the plot, i thought that all the transitions occur around 310.15c at lower concentrations. The red dots are where I think the transition is based on the graphs.

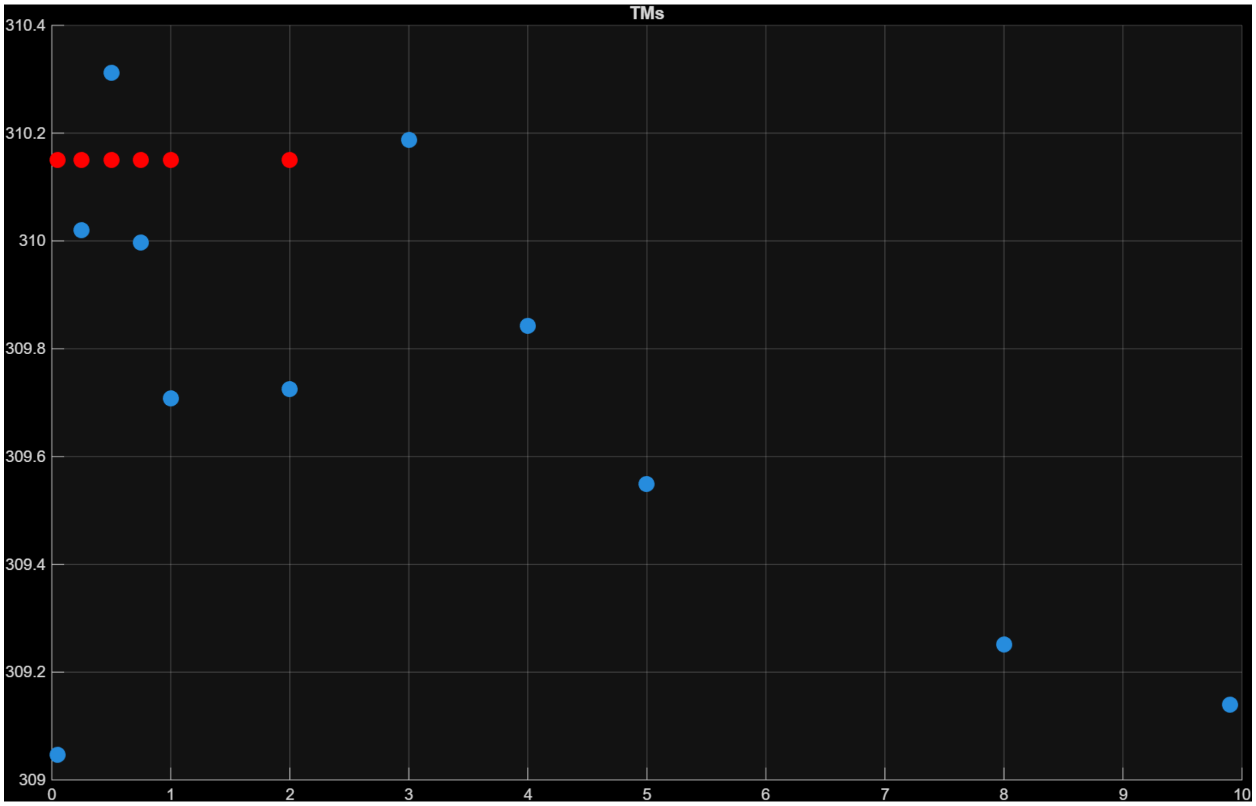


Figure 5: Transition temperatures

1.3 Results and Observations

In the SVD plots, the lower wt percents had much larger component 1s vs component 2s. The biggest jump between states is around $3-4\%$. These have the biggest dependence on the second component.

It definitely looks like the transition temp decreases with concentration, as expected. It looks like it levels out around 2% and lower, so to see this I will take more spectra to have a larger sample size.

2 Modeling

2.1 Monte Carlo lattice model of polymer collapse and extracting $D(r)$

2.1.1 Model and Hamiltonian

Take a chain of adjacent points on $\mathbb{Z}^3$.

$$\mathcal{C} = \{\mathbf{r}_0, \mathbf{r}_1, \dots, \mathbf{r}_{N-1}\}, \qquad \mathbf{r}_i \in \mathbb{Z}^3,$$

with$|\mathbf{r}_{i+1} - \mathbf{r}_i| = 1$. Self-avoidance is also enforced, $\mathbf{r}_i \neq \mathbf{r}_j$ for $i \neq j$.

For this model, we'll use an attractive "nearest-neighbor" contact energy, which acts between **non-bonded** points in the chain, i.e. The force can only be between $\{\mathbf{r_i}, \mathbf{r_j}\}, |i - j| > 1$. For each point, or monomer indexed i and each nearest-neighbor displacement $\mathbf{v} \in \mathbb{Z}^3, ||\mathbf{v}|| = 1$ such that $\mathbf{r}_i + \mathbf{v}$ is occupied by a monomer j that is not a bonded neighbor of i, the contact contributes energy $-\varepsilon$. Counting each contact twice (once at each monomer) we define the total energy

$$E(\mathcal{C}) = -\frac{\varepsilon}{2} \sum_{i=0}^{N-1} \sum_{\mathbf{v}} \mathbf{1}_{\mathcal{C} \setminus \{\mathbf{r}_{i-1}, \mathbf{r}_{i+1}\}} \big(\mathbf{r}_i + \mathbf{v}\big), \tag{1}$$

where $\mathbf{v} \in \mathbb{Z}^3, ||\mathbf{v}|| = 1$ and $\mathbf{1}(\cdot)$ is the indicator. The 1/2 accounts for double counting of contacts.

2.1.2 Monte Carlo move set and Metropolis condition

We evolve the chain with a Metropolis-Monte Carlo condition using three generating move types chosen uniformly at each step:

1. **Pivot move.** Choose a random pivot monomer indexed $i \in \{1, \dots, N-2\}$. Rotate the tail $\{\mathbf{r}_{i+1}, \dots, \mathbf{r}_{N-1}\}$ about pivot $\mathbf{r}_i$ by a random 90° rotation (about x, y, z axes). If any new site coincides with $\{\mathbf{r}_0, \dots, \mathbf{r}_i\}$, reject the move. Otherwise compute ΔE and accept with probability determined by the metropolis condition (2)

2. **Bend-flip** Randomly choose an internal monomer i and attempt a local 180° flip of a 90° kink: if the local triplet $(\mathbf{r}_{i-1}, \mathbf{r}_i, \mathbf{r}_{i+1})$ forms a right-angle (i.e. two orthogonal unit bonds), propose moving $\mathbf{r}_i$ to the opposite corner that preserves bond lengths. Reject on self-intersection.

3. **End move** Choose one end of the chain and attempt to move the end monomer to a neighboring empty site that maintains a unit bond to the next monomer.

Each trial move that passes self-avoidance has an energy change $\Delta E = E(\mathcal{C}_{\text{new}}) - E(\mathcal{C}_{\text{old}})$. The Metropolis acceptance probability at inverse temperature $\beta = 1/T$ (I set $k_B = 1$ in the lattice code) is

$$P_{\text{acc}}(\Delta E) \;=\; \min\{1,\; e^{-\beta \Delta E}\}. \tag{2}$$

This is from my computational physics notes, and has to do with the Boltzmann distribution $\propto e^{-\beta E}$.

2.1.3 Time mapping

We map Monte Carlo frames to an effective time by defining one *MC sweep* to be N attempted moves (on average one attempt per monomer).

- A single call to the main loop attempts one move (pivot/crankshaft/end) chosen uniformly; when sampling every S steps we set $\Delta t_{\text{frame}} = S/N$ MC sweeps per stored frame. In the code this is `sample_every = 1000` with Δt_{frame} = `sample_every` (the code takes Δt_{frame} in units of attempted moves.

So τ saved frames corresponds to an "real" time window.

2.1.4 Observables: radius of gyration and snapshots

The radius of gyration is

$$R_g[\mathcal{C}] \;=\; \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}\left|\mathbf{r}_i-\bar{\mathbf{r}}\right|^2}, \quad \bar{\mathbf{r}} = \frac{1}{N}\sum_{i=0}^{N-1}\mathbf{r}_i. \tag{3}$$

Snapshots of the full chain $\mathcal{C}(t)$ are recorded every `sample_every` attempted moves; these form the trajectory array of shape $(F, N, 3)$ used for visualization and analysis.

2.1.5 Measuring $D(r)$

We get a radial dependence of an effective per-monomer diffusivity $D(r)$ from the MSD computed on the recorded trajectory. We treat each monomer as an independent particle for the purpose of this.

Displacement and MSD. For saved frames indexed by time $t = 0, \ldots, F-1$, the displacement of monomer i over τ frames is

$$\Delta\mathbf{r}_i(t;\tau) = \mathbf{r}_i(t+\tau) - \mathbf{r}_i(t).$$

The squared displacement is $\Delta r_i^2(t;\tau) = |\Delta\mathbf{r}_i(t;\tau)|^2$. For fixed lag τ we collect all such displacements for all monomers and all valid starting frames t and bin them according to the radial position at the start,

$$r_i(t) \;=\; \left|\mathbf{r}_i(t)\right|.$$

Choose the radial bin edges $\{r_k\}_{k=0}^{M}$ and the bin index k satisfy $r_k \le r_i(t) < r_{k+1}$. For a given bin k we estimate the local diffusion coefficient by:

$$\hat{D}_k \;=\; \frac{\langle \Delta r^2\rangle_k}{6\,\tau_{\text{phys}}} \;=\; \frac{\overline{\Delta r^2}_k}{6\,\tau\,\Delta t_{\text{frame}}}, \tag{4}$$

where $\overline{\Delta r^2}_k$ is the MSD over all (t,i) pairs falling into bin k, and $\tau_{\text{phys}} = \tau\,\Delta t_{\text{frame}}$ is the effective time window. In code:

$$D[k] = \frac{\text{mean}(\Delta r^2 \mid r \in \text{bin } k)}{6\,dt_{\text{frame}}\,\tau}.$$

Estimator variance. For bin k with n_k samples the standard error of the mean MSD is approximately $\sigma_{\mathrm{MSD},k} \approx \mathrm{sd}(\Delta r^2)/\sqrt{n_k}$. If you assume the sample variance is proportional to $\overline{\Delta r^2}_k^2$, you can approximate the uncertainty of D as

$$\sigma_{D,k} \approx \frac{\sigma_{\mathrm{MSD},k}}{6\,\tau_{\mathrm{phys}}} \approx \frac{\mathrm{sd}(\Delta r^2)}{6\,\tau_{\mathrm{phys}}\sqrt{n_k}}.$$

I chose $\sigma_k = 1/\sqrt{\max(n_k, 2)}$ when fitting.

2.1.6 Smoothing and fitting $D(r)$

To obtain a smooth functional form, we fit the binned $D(r)$ to a hyperbolic tangent profile of the form

$$D_{\mathrm{fit}}(r; D_{\mathrm{core}}, D_{\mathrm{shell}}, r_c, w) \;=\; D_{\mathrm{shell}} \;-\; \frac{D_{\mathrm{shell}} - D_{\mathrm{core}}}{2}\left(1 - \tanh\!\left(\tfrac{r-r_c}{w}\right)\right).$$

This has limits $D(r \ll r_c) \approx D_{\mathrm{core}}$ and $D(r \gg r_c) \approx D_{\mathrm{shell}}$, center r_c and transition width w.

2.1.7 Practical choices in the implementation

- **Burn-in.** The code discards an initial fraction (30%) of frames as burn-in to ensure the chain is sampled at equilibrium
- **Center of Mass** For the diffusion analysis the chain snapshots are recentered by subtracting each frame's center of mass:

  $$\tilde{\mathbf{r}}_i(t) = \mathbf{r}_i(t) - \overline{\mathbf{r}}(t).$$

  This subtracts global translations.

2.1.8 Interpretation

The extracted $D(r)$ is an *effective* radial diffusivity of monomers based on their radial distance from the chain center-of-mass. To use $D(r)$ in a continuous model for the radial distribution $c(r,t)$,you have to worry about:

- **Coordinates:** Going from a lattice to continuum in different spaces is a heuristic jump to make.
- **Units:** The SAW code uses dimensionless temperature T with $k_B = 1$. The mapping of temperatures and energies just preserves their ratio, but is also just a choice.

2.1.9 Application

For a 44-mer at $t = 297K$ with a contact energy $\epsilon = 198K \cdot k_B$ I found

- $w = 2.961$ units

- $r_c = 4.281$ units
- $D_{core} = 0.0001$
- $D_{shell} = 0.002$

.

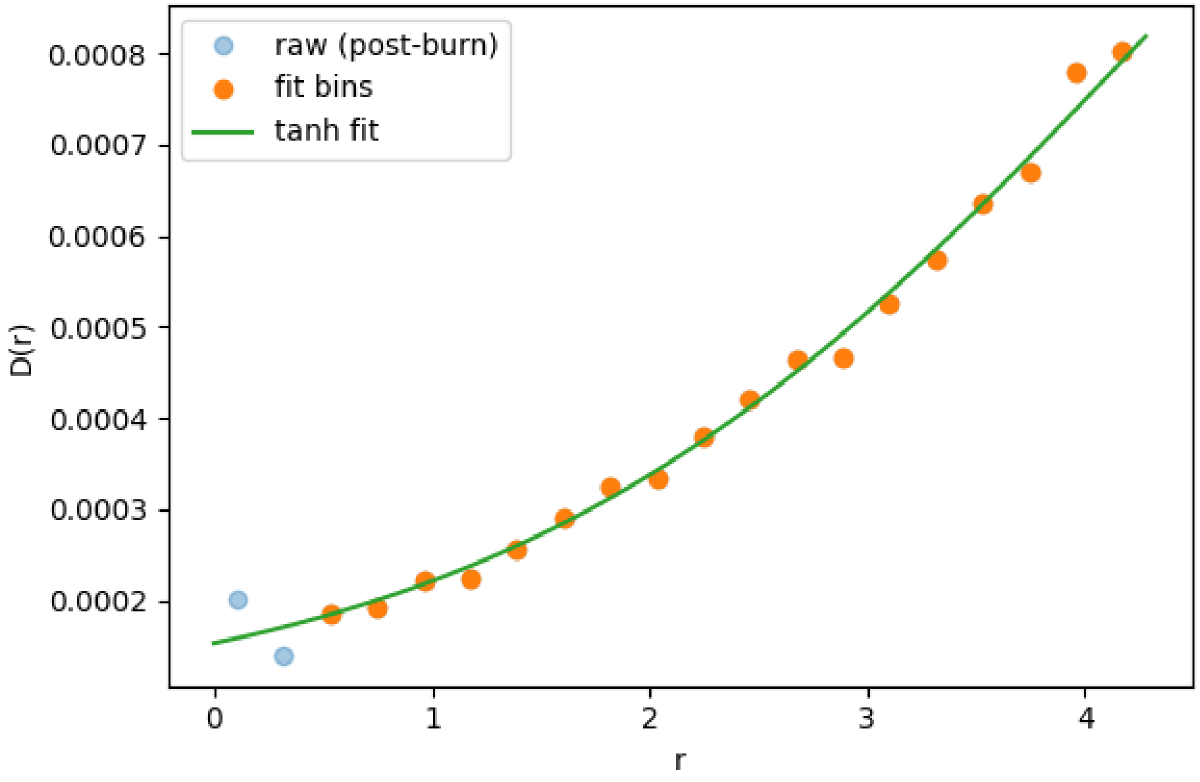


Figure 6: fitted D(r)

I then took this, scaled it to my diffusion solver, and used Ian's data to visualize this. There was some confusion, I now plot both volume and radial prb densities.

Note: Ian's density distributions are of $P(R_G)$ I think. So, taking them as shell density distributions is very heuristic. However, I still think it is cool to look at. The distribution does not make sense to me. I'd expect it to be centered around 0, where 0 is the center of mass, that is how D(r) is defined in my lattice solver.

2.2 Smoluchowski Solver for a collapsing polymer in spherical symmetry

2.2.1 Model

Let $c(r,t)$ denote the 3D probability density (per unit volume) for the "polymer size" at radius $r \in [0,R]$ and time $t \geq 0$. The Smoluchowski equation we want to solve is

$$\partial_t c = \frac{1}{r^2}\,\partial_r\left[r^2 D(r)\left(\partial_r c + \beta\, c\, U'(r)\right)\right], \qquad \beta = \frac{1}{k_B T}, \tag{5}$$

where $U'(r) = \frac{dU}{dr}$ and $k_B T$ has its usual meaning. Writing the radial probability flux

$$J_r(r,t) = -\,D(r)\Big(\partial_r c + \beta c\, U'(r)\Big), \tag{6}$$

the conservation law is

$$\partial_t c = -\frac{1}{r^2}\partial_r\left(r^2 J_r\right). \tag{7}$$

Also introduce the *radial probability density*

$$P(r,t) \;=\; 4\pi r^2\, c(r,t), \tag{8}$$

which integrates to one over $[0,R]$. Then $c = P/(4\pi r^2)$ and the evolution of P is

$$\partial_t P \;=\; -\,\partial_r J_P, \qquad J_P \;=\; -\,D(r)\Big(\partial_r P + \left[\beta U'(r) - \tfrac{2}{r}\right]P\Big). \tag{9}$$

We discretize (5) for $c(r,t)$ directly.

2.2.2 Spatial mesh and interface quantities

Let nodes be $r_j = j\,\Delta r$, $j = 0,\ldots,N_r-1$, with $\Delta r = R/(N_r-1)$. Interfaces are $r_{j+\frac{1}{2}} = \frac{1}{2}(r_j + r_{j+1})$. Assemble

$$D_j = D(r_j), \qquad D_{j+\frac{1}{2}} = \frac{2D_j D_{j+1}}{D_j + D_{j+1}} \quad \text{(harmonic mean)}, \tag{10}$$

$$A_{j+\frac{1}{2}} = r^2_{j+\frac{1}{2}}\, D_{j+\frac{1}{2}}, \qquad U'_{j+\frac{1}{2}} \approx \frac{U_{j+1} - U_j}{\Delta r}. \tag{11}$$

2.2.3 Crank–Nicolson discretization

The main goal of this scheme is to more accurately solve differential equations. It is half implicit, half explicit. The scheme is shown here:

THIS SECTION IS NOT DONE- THERE IS A LOT OF EXPLAINING TO DO HERE

Integrate (5) over the spherical shell around r_j and apply the divergence theorem:

$$\frac{c_j^{n+1} - c_j^n}{\Delta t} \;=\; -\frac{1}{r_j^2\,\Delta r}\left[\, r^2_{j+\frac{1}{2}} J^{n+\frac{1}{2}}_{j+\frac{1}{2}} - r^2_{j-\frac{1}{2}} J^{n+\frac{1}{2}}_{j-\frac{1}{2}}\,\right], \tag{12}$$

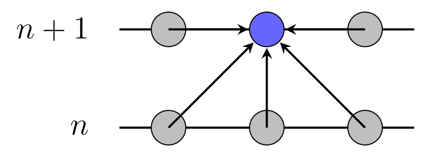


Figure 7: Crank–Nicolson stencil: central point at $n+1$ coupled to neighbors at n and $n+1$.

where this method uses the in between flux

$$J^{n+\frac{1}{2}}_{j+\frac{1}{2}} \;=\; -\,D_{j+\frac{1}{2}}\left(\frac{c^{n+\frac{1}{2}}_{j+1}-c^{n+\frac{1}{2}}_{j}}{\Delta r}+\beta\,U'_{j+\frac{1}{2}}\,\frac{c^{n+\frac{1}{2}}_{j+1}+c^{n+\frac{1}{2}}_{j}}{2}\right), \quad c^{n+\frac{1}{2}}=\tfrac{1}{2}(c^{n+1}+c^{n}). \tag{13}$$

Collecting unknowns at level $n+1$ on the left and known terms at level n on the right gives a tridiagonal matrix equation

$$a_j\,c^{n+1}_{j-1}+b_j\,c^{n+1}_{j}+c_j\,c^{n+1}_{j+1} \;=\; \mathrm{RHS}_j, \tag{14}$$

where each coefficient is the sum of a diffusion part and a drift part. Define

$$A^{\star}_{j-\frac{1}{2}} \equiv \frac{\Delta t}{2\,r_j^2\,\Delta r^2}\,A_{j-\frac{1}{2}}, \qquad A^{\star}_{j+\frac{1}{2}} \equiv \frac{\Delta t}{2\,r_j^2\,\Delta r^2}\,A_{j+\frac{1}{2}}, \tag{15}$$

and

$$G_{j-\frac{1}{2}} \equiv \frac{\Delta t}{2\,r_j^2\,\Delta r}\,A_{j-\frac{1}{2}}\,\beta U'_{j-\frac{1}{2}}, \qquad G_{j+\frac{1}{2}} \equiv \frac{\Delta t}{2\,r_j^2\,\Delta r}\,A_{j+\frac{1}{2}}\,\beta U'_{j+\frac{1}{2}}. \tag{16}$$

Then the *implicit* contributions (left-hand matrix) are

$$a_j^{(\mathrm{imp})} = -A^{\star}_{j-\frac{1}{2}} \;-\; \tfrac{1}{2}\,G_{j-\frac{1}{2}}, \tag{17}$$

$$b_j^{(\mathrm{imp})} = 1 + A^{\star}_{j-\frac{1}{2}} + A^{\star}_{j+\frac{1}{2}} \;+\; \tfrac{1}{2}\,(G_{j+\frac{1}{2}} - G_{j-\frac{1}{2}}), \tag{18}$$

$$c_j^{(\mathrm{imp})} = -A^{\star}_{j+\frac{1}{2}} \;+\; \tfrac{1}{2}\,G_{j+\frac{1}{2}}, \tag{19}$$

and the *explicit* contributions (right-hand side) carry opposite signs:

$$a_j^{(\mathrm{exp})} = +A^{\star}_{j-\frac{1}{2}} \;+\; \tfrac{1}{2}\,G_{j-\frac{1}{2}}, \tag{20}$$

$$b_j^{(\mathrm{exp})} = 1 - (A^{\star}_{j-\frac{1}{2}} + A^{\star}_{j+\frac{1}{2}}) \;-\; \tfrac{1}{2}\,(G_{j+\frac{1}{2}} - G_{j-\frac{1}{2}}), \tag{21}$$

$$c_j^{(\mathrm{exp})} = +A^{\star}_{j+\frac{1}{2}} \;-\; \tfrac{1}{2}\,G_{j+\frac{1}{2}}. \tag{22}$$

The resulting linear system is solved each time step with a Thomas (tridiagonal) solver.

2.2.4 Boundary conditions

At the origin $r = 0$. Spherical symmetry imposes $J_{1/2} = 0$ and c finite. In practice, my code uses a "mirror" condition $c_{-1} = c_{+1}$, which leads to the special first-row coefficients

$$a_0 = 0, \quad c_0 = -\alpha, \quad b_0 = 1 + \alpha$$

with α chosen to make the $r \to 0$ limit work, ($\alpha = 1.5\,\Delta t\,D_{1/2}/\Delta r^2$).

At the outer wall $r = R$**.** **Absorbing**: $c(R,t) = 0$ Numerically: last row is identity and you set $c_{N_r-1} = 0$. This matches your current code and requires renormalization if you want to keep total probability at 1.

2.2.5 Initial condition and normalization

Here I use a distribution I got from Ian-

2.2.6 Two-state potential

To model coil $\leftrightarrow$ globule with a barrier, I use a sum of Gaussians to create a double well This is variable, and what I chose here is not perfect, but an example.

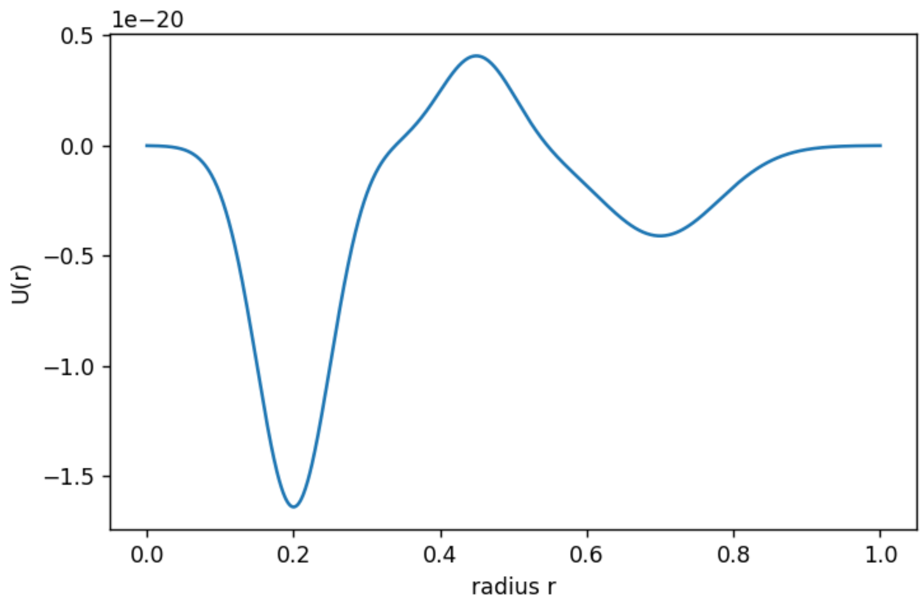


Figure 8: Chosen Potential

2.2.7 Plotting and Usage

Again, for a 44-mer at $t = 297K$ with a contact energy $\epsilon = 198K \cdot k_B$ I have

- $w = 2.961$ units
- $r_c = 4.281$ units
- $D_{core} = 0.0001$
- $D_{shell} = 0.002$

.

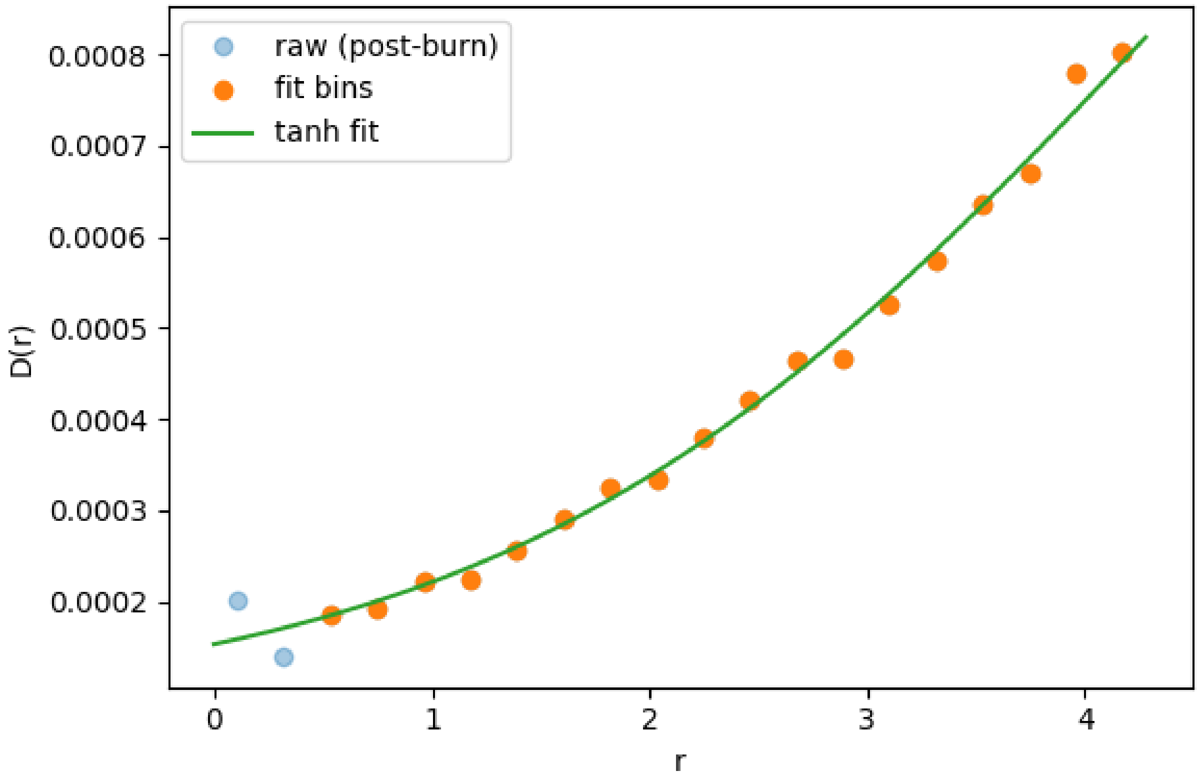


Figure 9: fitted D(r)

I then took this, scaled it to my diffusion solver, and used Ian's data to visualize this. There was some confusion, I now plot both volume and radial prob densities.

Note: Ian's density distributions are of $P(R_G)$. So, taking them as shell density distributions is very heuristic. However, I still think it is cool to look at.

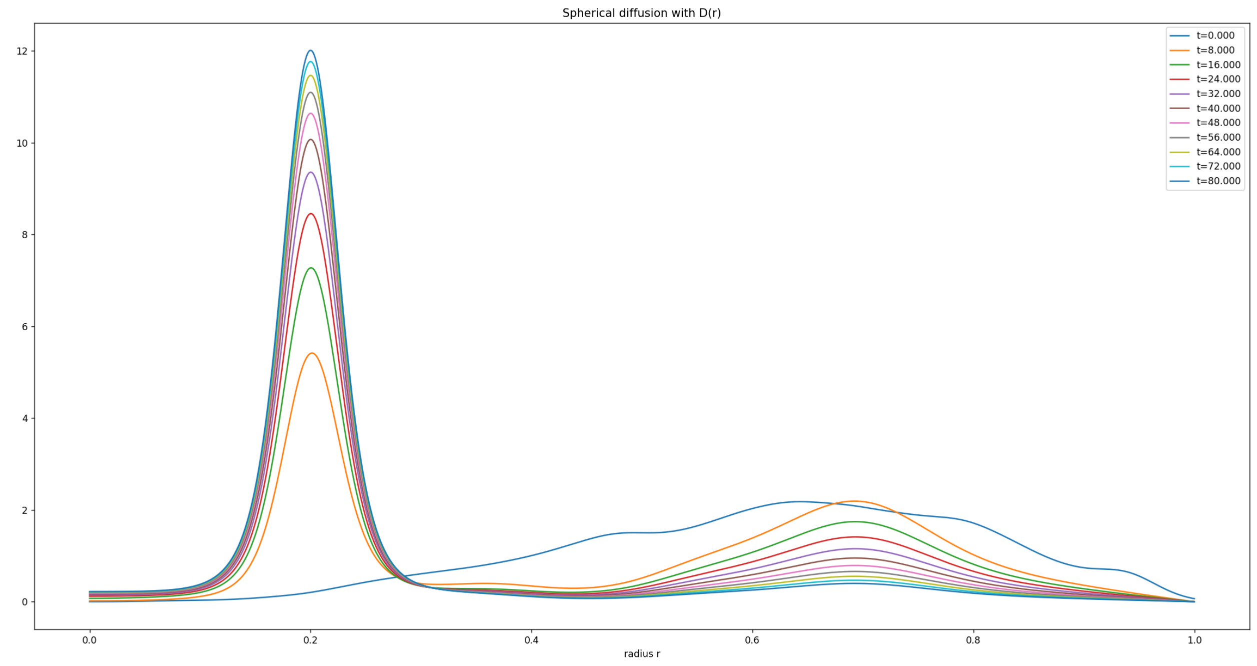


Figure 10: Spherical Probability Density

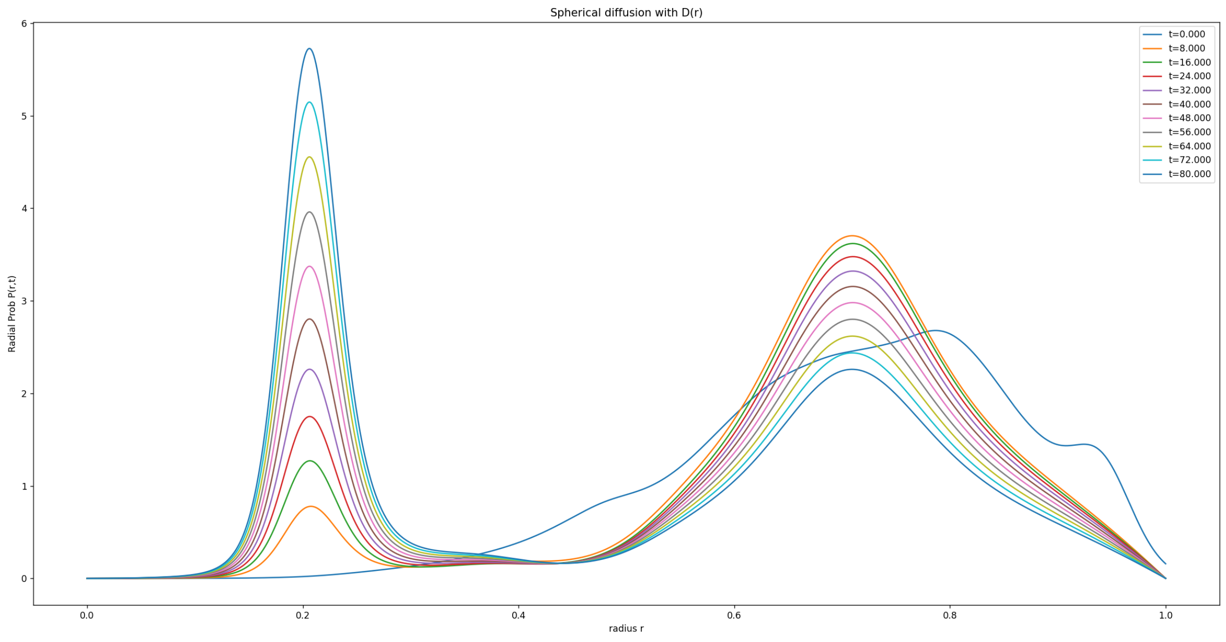


Figure 11: Radial Prob Density

2.3 Next steps

I don't feel confident in how the diffusion solver handles boundary conditions $r = 0, r = R$. The origin problem is inherent to the algorithm I chose, which I initially picked because of accuracy concerns. I now think that I can choose a simpler scheme and avoid a division by 0 issue at the origin. Also, the issue of the density distributions not matching up should be accounted for. Also, polymers don't just interact with themselves. I should be able to model more than 1 polymer. Maybe down the road I can look at N-polymer systems? This would be helpful to see how different concentrations behave!

I'm using a reduced temperature and reduced contact energy in the monte carlo code because it is only their ratio that matters and metropolis algorithm needs unitless parameters. This means that the temperature assignment is kind of arbitrary, as long as the ratios of ϵ/t are consistent. To get a better assignment for t, **I need a value for contact energy** ϵ. What I could do change this and then sweep different temperatures and see where the chain collapses the fastest to find a transition temperature. However, this would probably take forever, definitely something i could use the supercomputers for.

Also, initial distributions are still an issue. So is how the thing diffuses. Both things to consider.