The Benefit of Poor Mixing: Kinetics of Coacervation SI

Whitney C. Blocher McTigue, Elizabeth Voke, Li-Wei Chang, and Sarah L. Perry*
Department of Chemical Engineering, University of Massachusetts Amherst
*Correspondence: perrys@engin.umass.edu

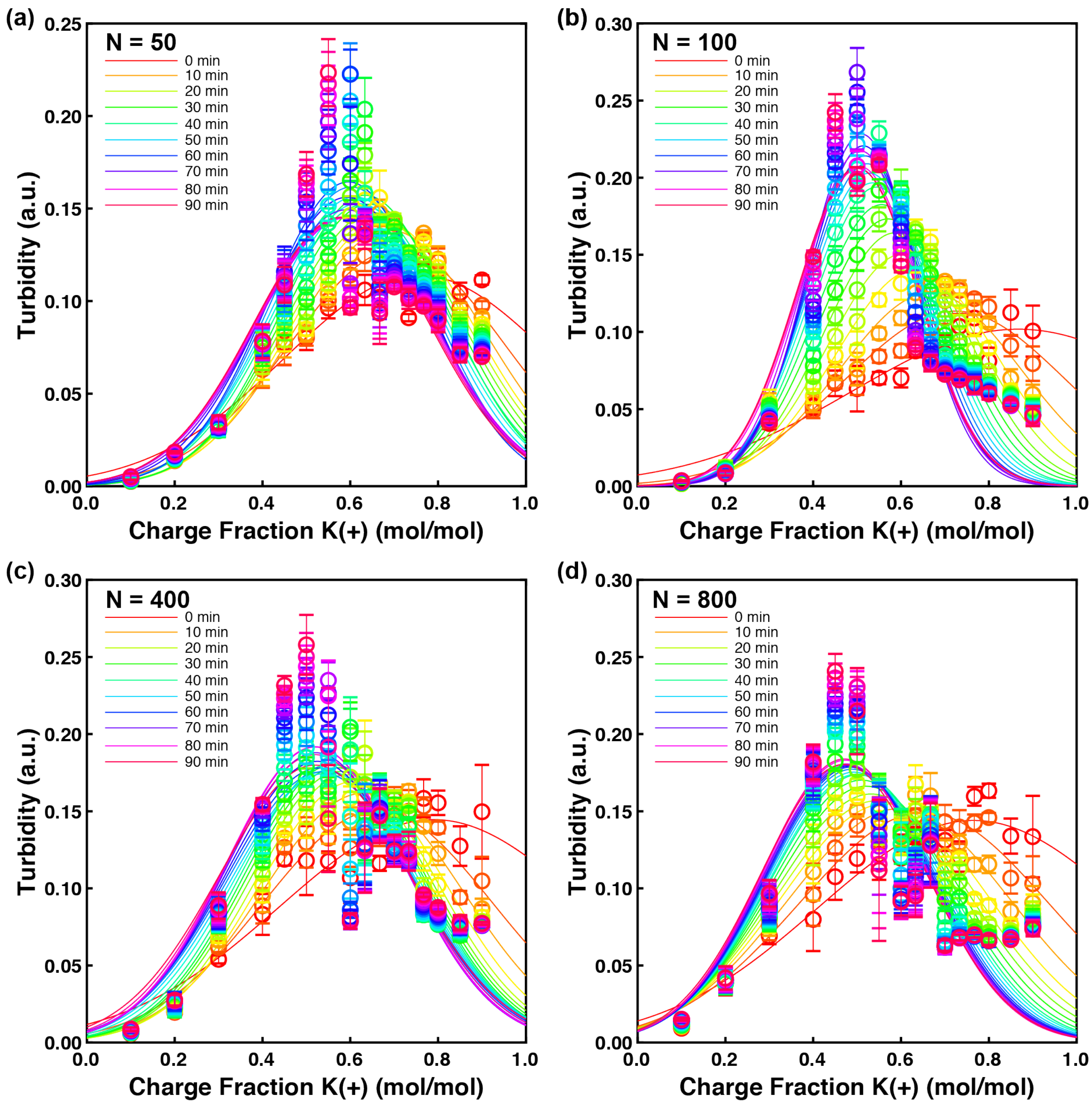


Figure S1. Plots of turbidity versus cationic charge fraction illustrating the of polymer chain length. The positive polymer was added first for E_N/K_N systems with **(a)** N = 50, **(b)** N = 100, **(c)** N = 400, and **(d)** N = 800. In all cases, the stoichiometric curves are broader than peptides whose chains are half charged. Elapsed time was **(a)** 65 min, **(b)** 80 min, **(c)** 45 min, and **(d)** 35 min. Error bars represent the standard deviation of N = 27.

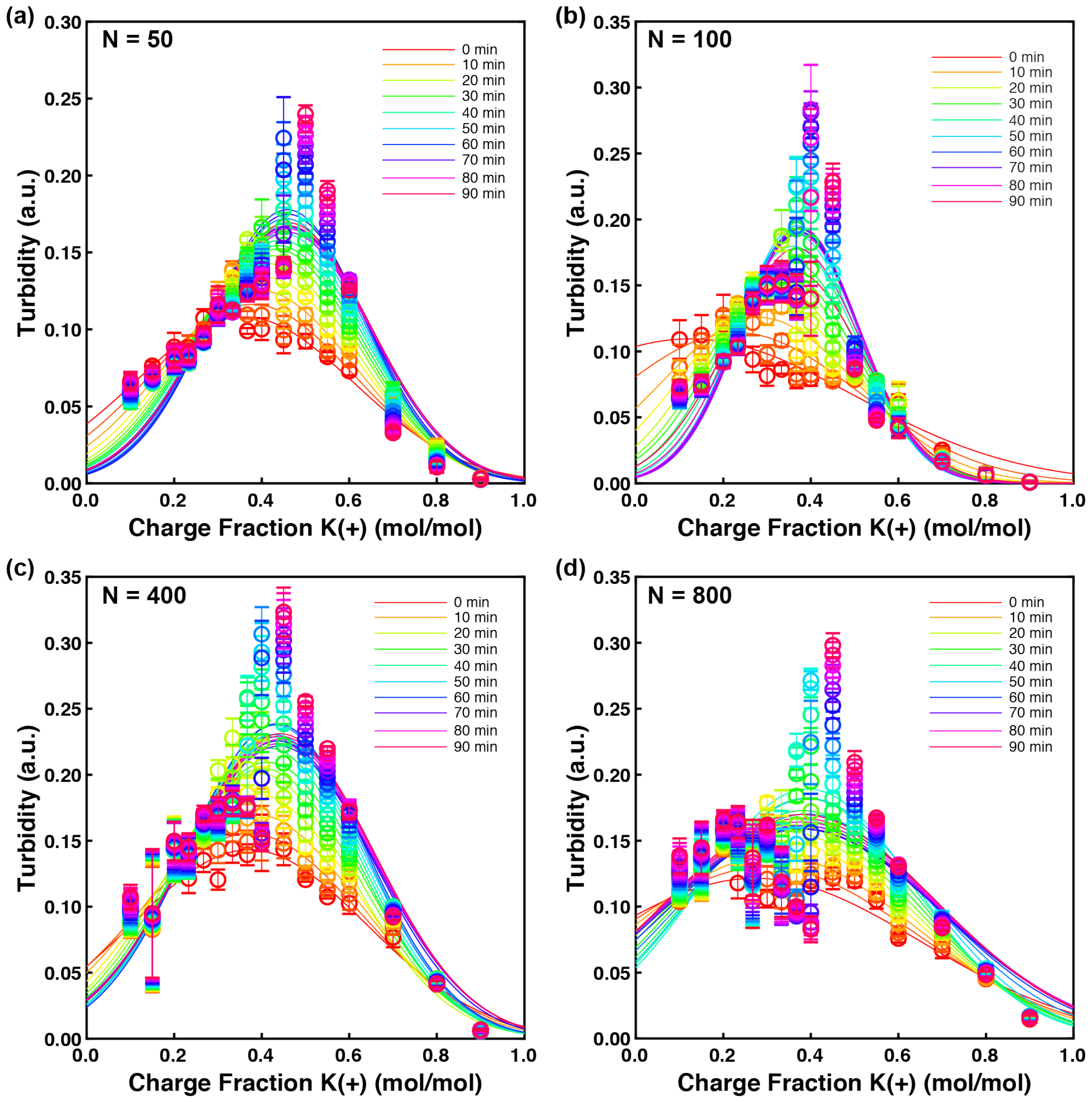


Figure S2. Plots of turbidity versus cationic charge fraction illustrating the effect of polymer chain length. The negative polymer was added first for E_N/K_N systems with **(a)** N = 50, **(b)** N = 100, **(c)** N = 400, and **(d)** N = 800. In all cases, the stoichiometric curves are broader than peptides whose chains are half charged. Elapsed time was **(a)** 65 min, **(b)** 85 min, **(c)** 65 min, and **(d)** 60 min. Error bars represent the standard deviation of N = 27.

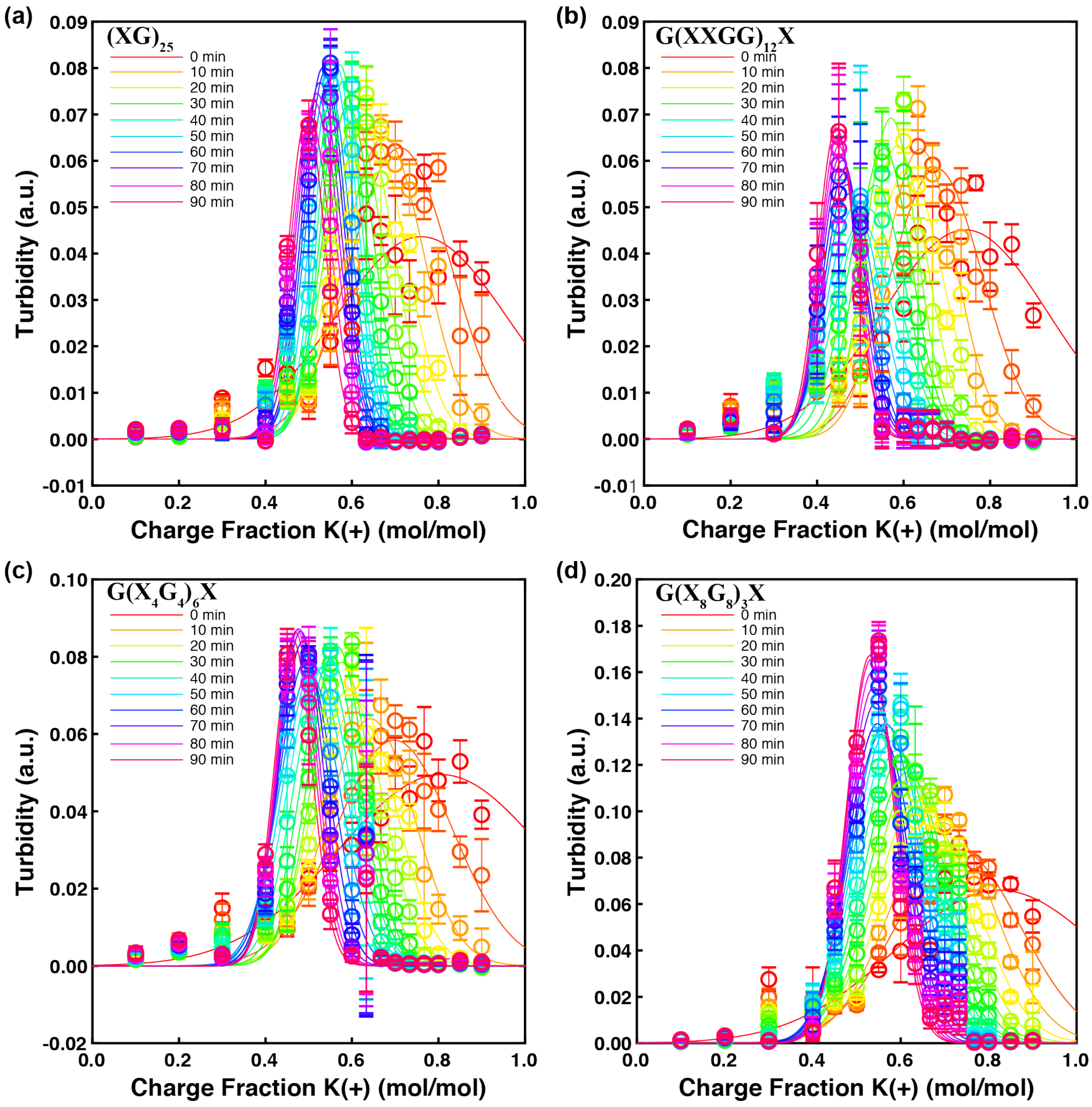


Figure S3. Plots of turbidity versus cationic charge fraction for symmetrically-patterned peptides where the the positive peptide was added first for **(a)** $(EG)_{25}/(KG)_{25}$, **(b)** $(E_2G_2)_{12}/(K_2G_2)_{12}$, **(c)** $(E_4G_4)_6/(K_4G_4)_6$, and **(d)** $(E_8G_8)_3/(K_8G_8)_3$. Elapsed time was **(a)** 80 min, **(b)** 60 min, **(c)** 80 min, and **(d)** 55 min. Error bars represent the standard deviation of N = 27.

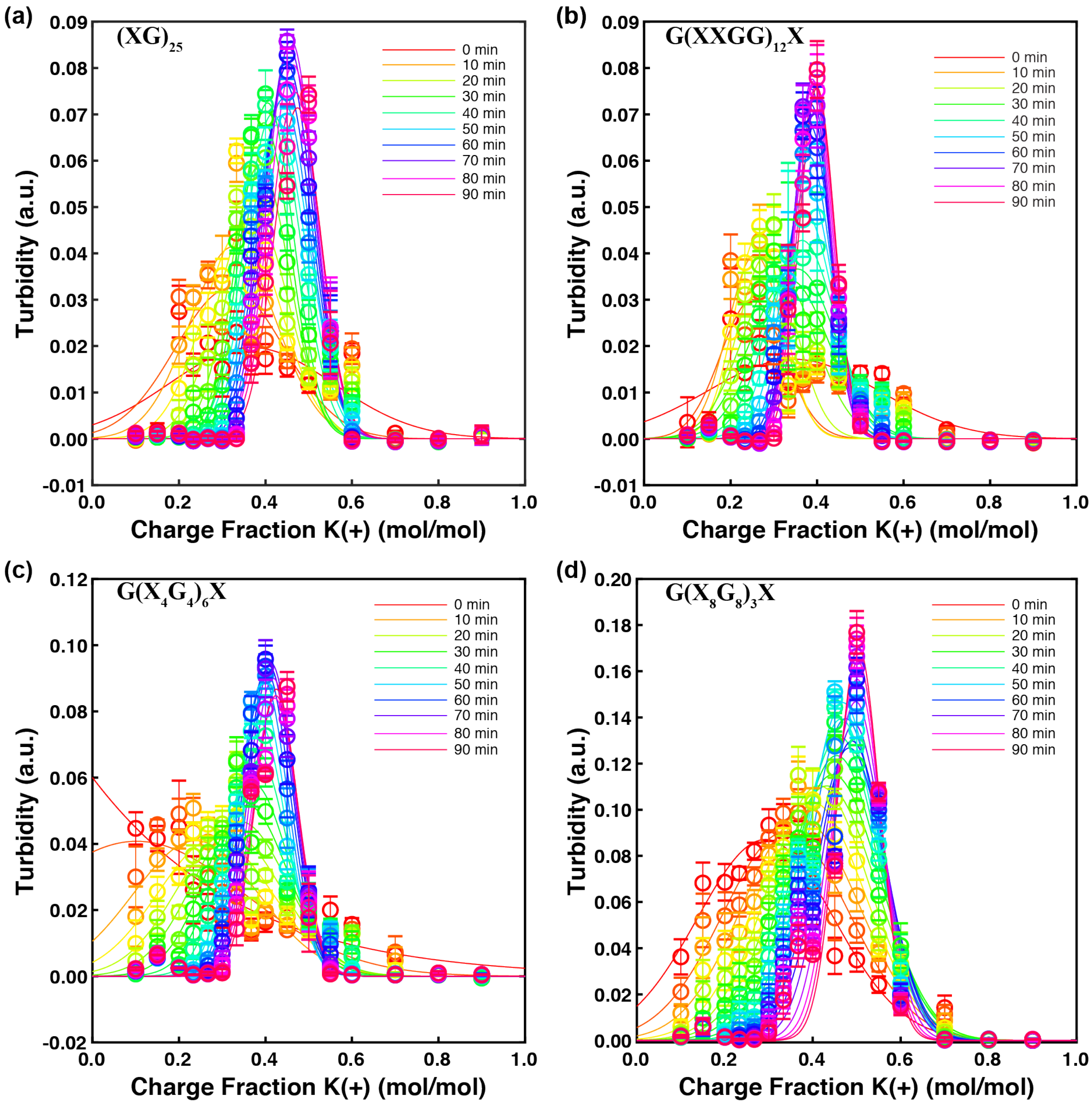


Figure S4. Plots of turbidity versus cationic charge fraction for symmetrically- patterned peptides where the negative peptide was added first for **(a)** $(EG)_{25}/(KG)_{25}$, **(b)** $(E_2G_2)_{12}/(K_2G_2)_{12}$, **(c)** $(E_4G_4)_6/(K_4G_4)_6$, and **(d)** $(E_8G_8)_3/(K_8G_8)_3$. Elapsed time was **(a)** 85 min, **(b)** 40 min, **(c)** 80 min, and **(d)** 55 min. Error bars represent the standard deviation of N = 27.

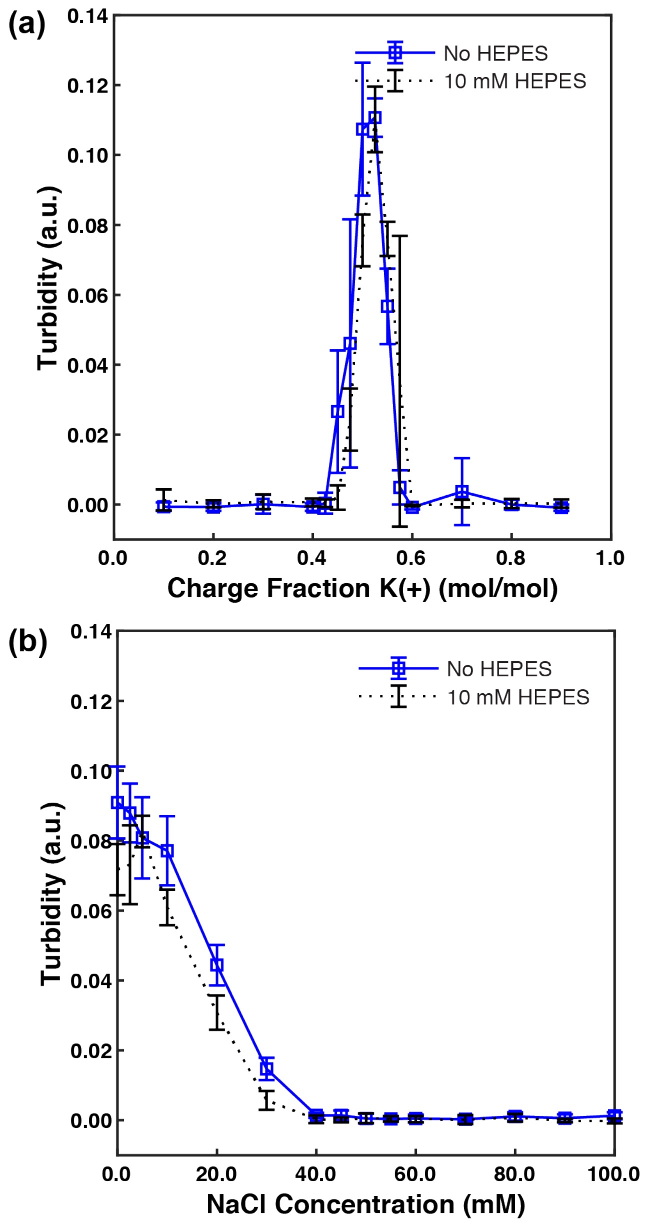
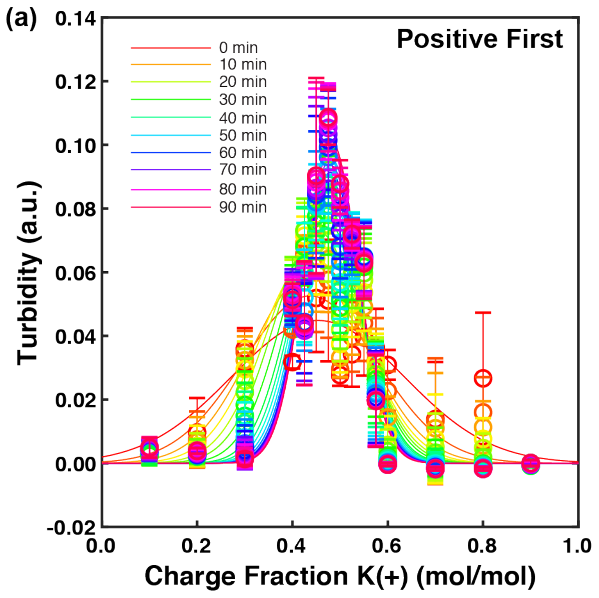


Figure S5. Turbidity comparison of complexation of $(KG)_{25}$/$(EG)_{25}$ with (black) and without (blue) 10 mM HEPES. **(a)** Complexation of $(KG)_{25}$/$(EG)_{25}$ varying the cationic mole fraction of $(KG)_{25}$ and **(b)** 50/50 mole ratio of $(KG)_{25}$/$(EG)_{25}$ as a function of increasing NaCl concentration. The salt resistance for the no buffer sample was determined to be 42 mM NaCl, while the sample with 10 mM HEPES had a salt resistance of 36 mM NaCl. Experiments were done in triplicate with three aliquots of each sample read three times each to calculate the standard deviation at each point. Error bars represent the standard deviation of N = 27.



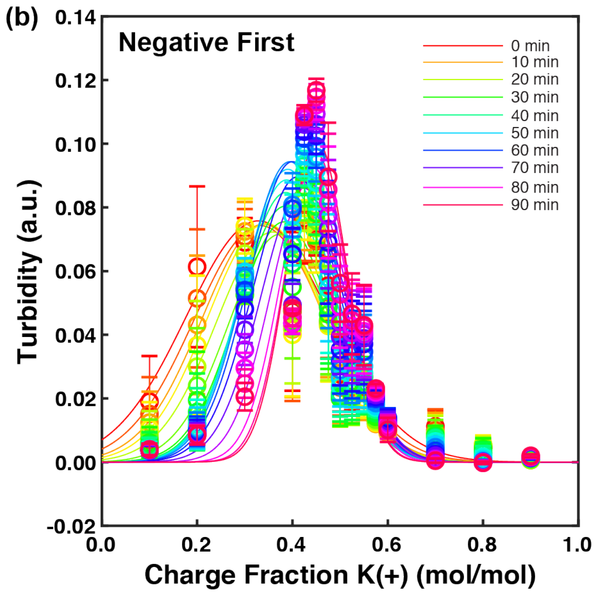


Figure S6. Plots of turbidity versus cationic charge fraction showing the complexation of K_{50} and $(EG)_{25}$. **(a)** Shows the addition of K_{50} first and **(b)** shows the addition of $(EG)_{25}$ of first in 10 mM HEPES. When the patterned polyelectrolyte is added first, there is a shift toward equilibrium, whereas there is little to no shift when the homopolymer is added first. Elapsed time for **(a)** and **(b)** was 20 min and 70 min, respectively. Error bars represent the standard deviation of N = 27.

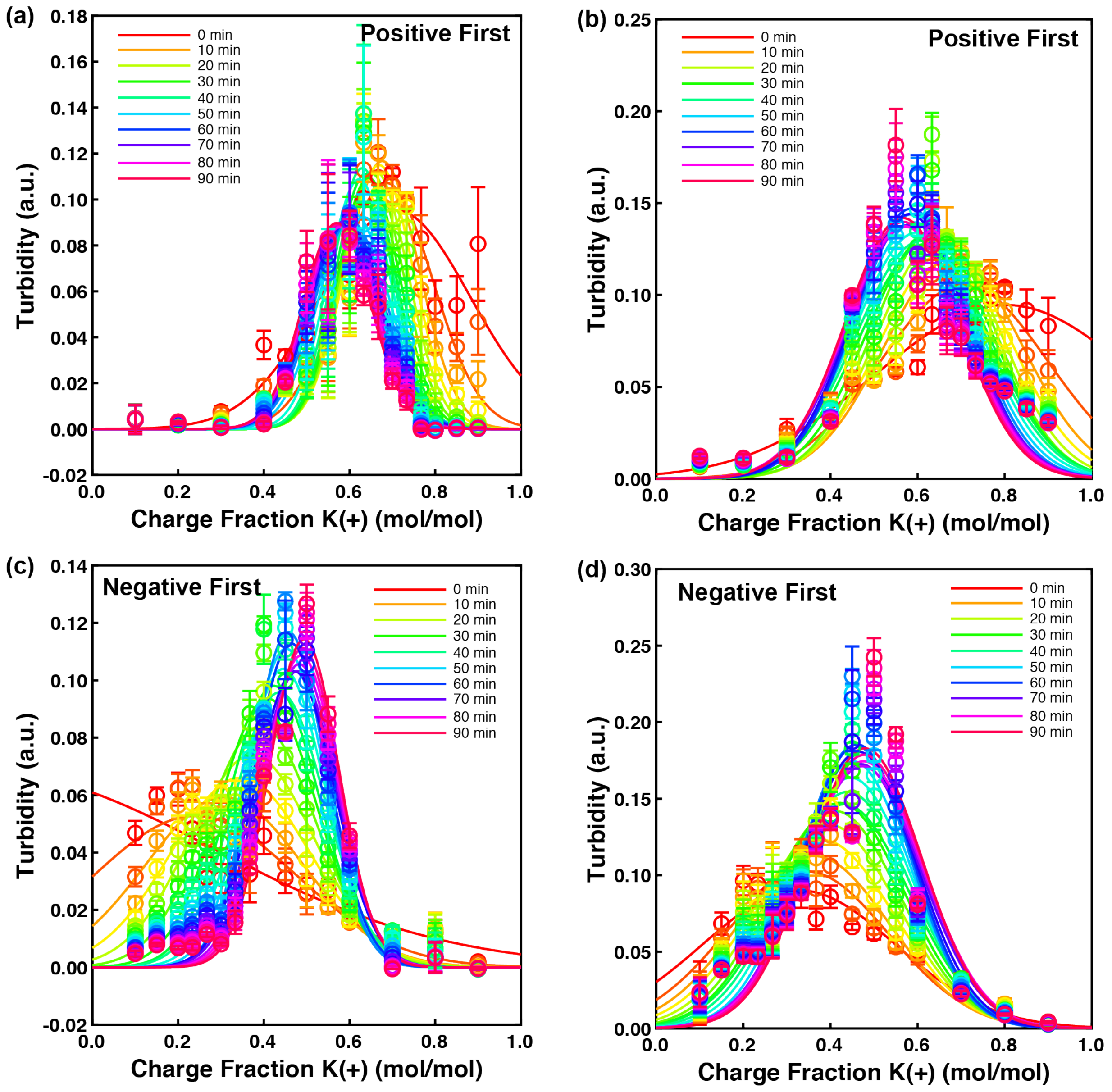


Figure S7. Plots of turbidity versus cationic charge fraction showing the complexation of K_{50} with **(a, c)** $(EG)_{25}$ and **(b, d)** E_{100}. Data where the positive polymer was added first are shown in **(a, b)**, while those corresponding to experiments where the negative polymer was added first are in **(c, d)**. Elapsed time was **(a)** 80 min, **(b)** 70 min, **(c)** 65 min, and **(d)** 65 min. Error bars represent the standard deviation of N = 27.

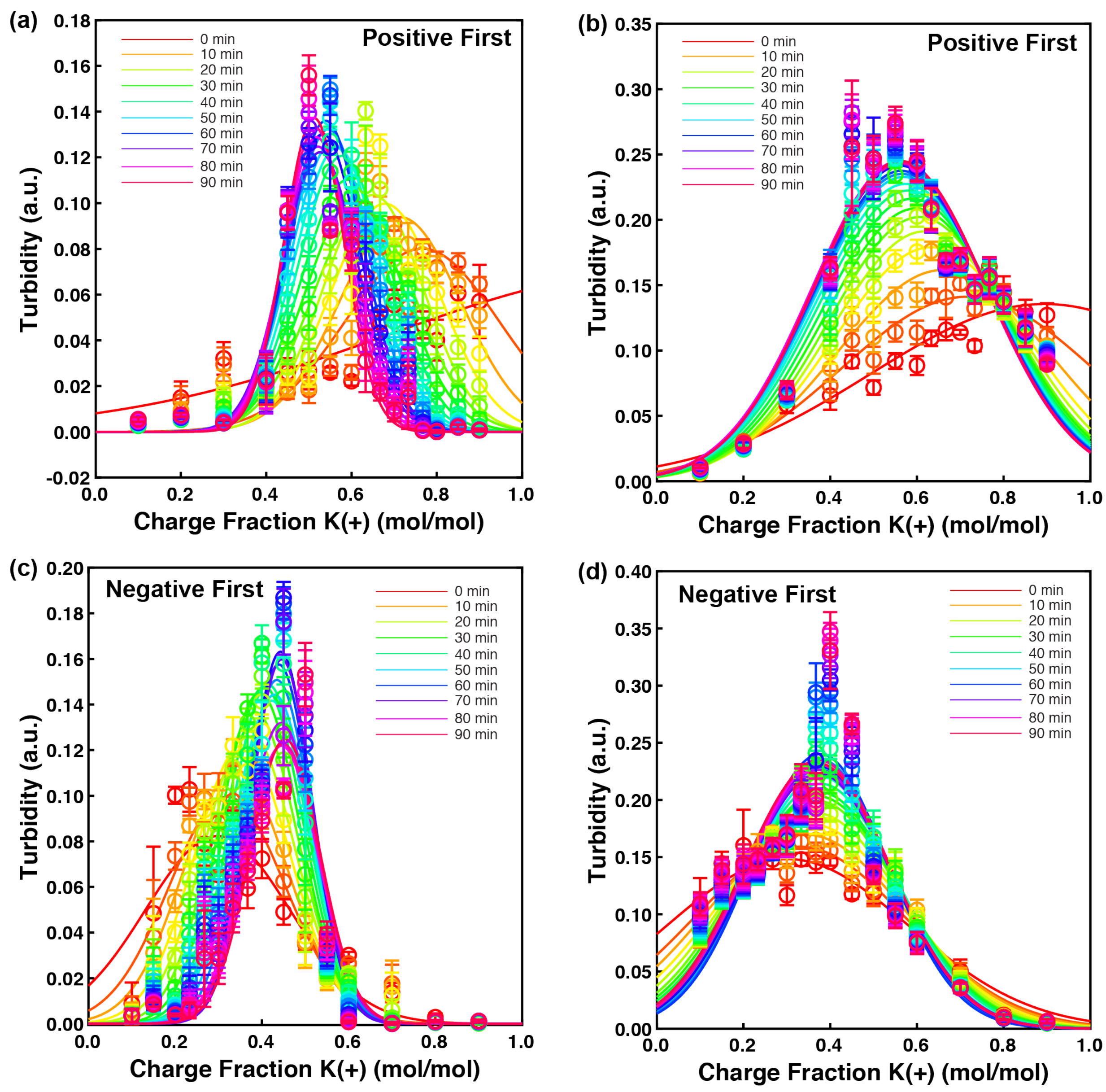


Figure S8. Plots of turbidity versus cationic charge fraction showing the complexation of E_{50} with **(a, c)** $(KG)_{25}$ and **(b, d)** K_{100}. Data where the positive polymer was added first are shown in **(a, b)**, while those corresponding to experiments where the negative polymer added first are in **(c, d)**. Elapsed time was **(a)** 65 min, **(b)** 60 min, **(c)** 75 min, and **(d)** 60 min. Error bars represent the standard deviation of N = 27.

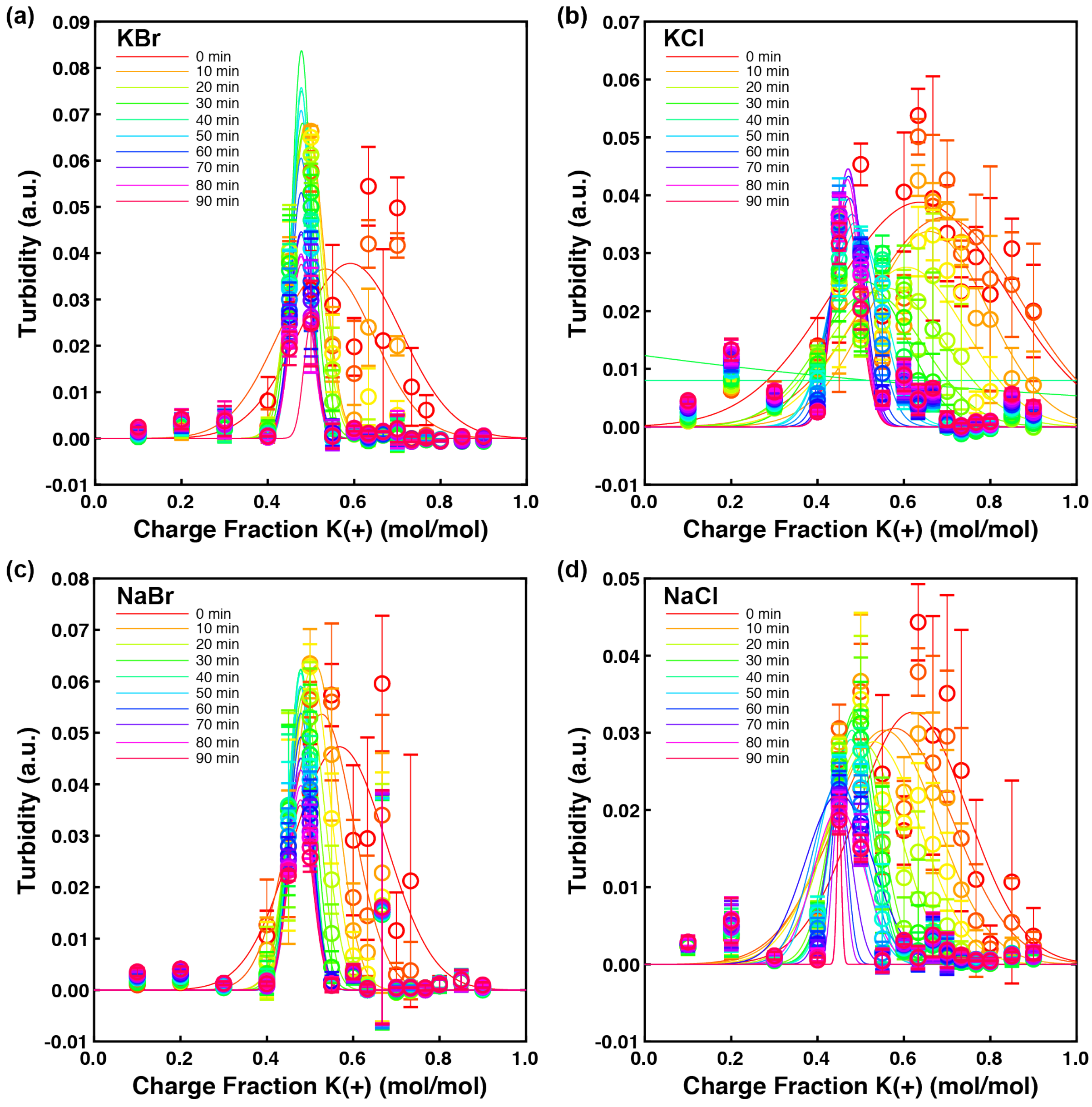


Figure S9. Plots of turbidity versus cationic charge fraction showing the effect of 10 mM **(a)** KBr, **(b)** KCl, **(c)** NaBr, and **(d)** NaCl on the $(EG)_{25}/(KG)_{25}$ system, for samples where the positive peptide was added first. Elapsed time was **(a)** 5 min, **(b)** 25 min, **(c)** 5 min, and **(d)** 50 min. Error bars represent the standard deviation of N = 27.

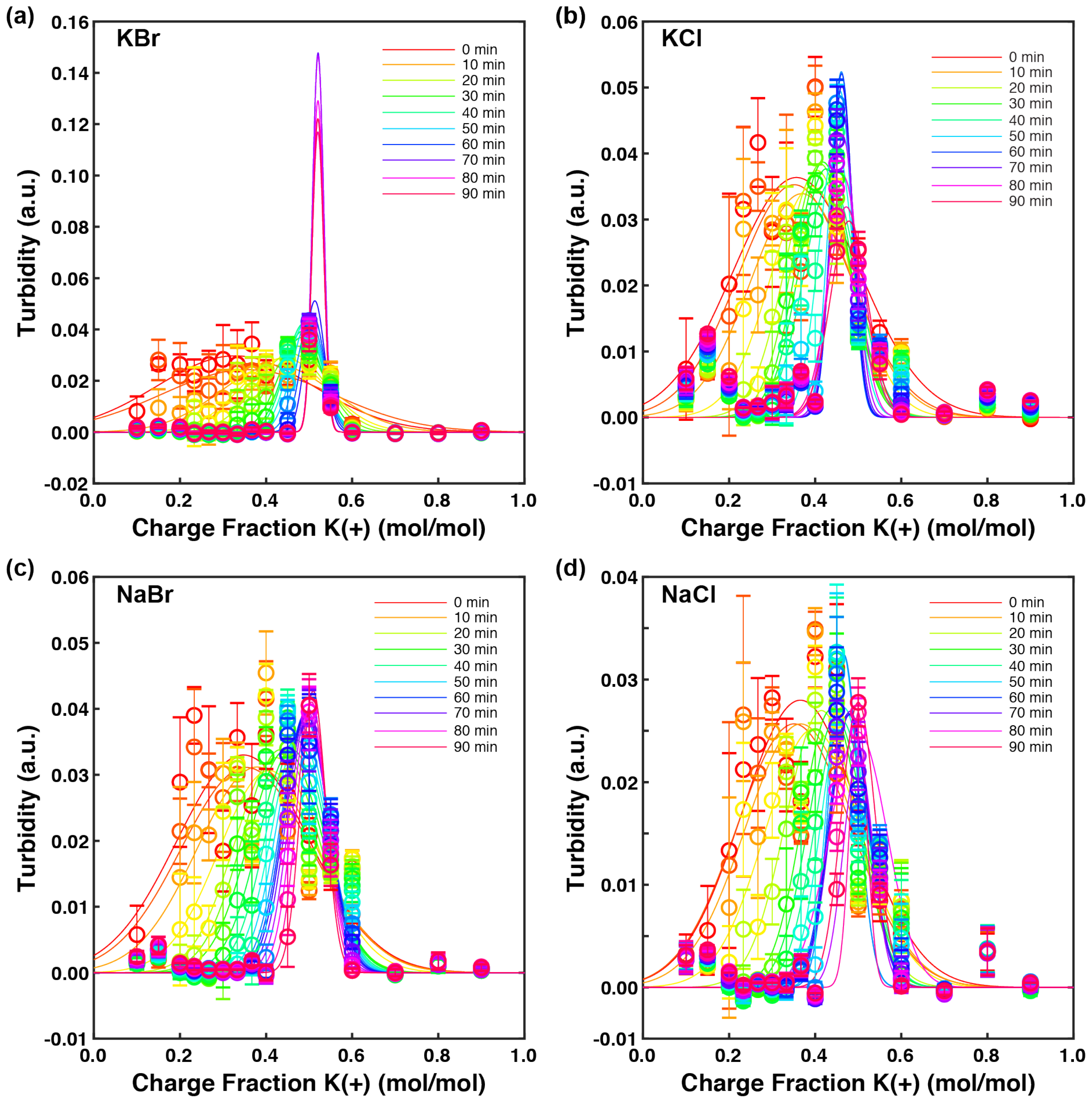


Figure S10. Plots of turbidity versus cationic charge fraction showing the effect of 10 mM **(a)** KBr, **(b)** KCl, **(c)** NaBr, and **(d)** NaCl on the $(EG)_{25}/(KG)_{25}$ system, for samples where the negative peptide was added first. Elapsed time was **(a)** 5 min, **(b)** 90 min, **(c)** 65 min, and **(d)** 75 min. Error bars represent the standard deviation of N = 27.

Table S1: Polycation, polyanion, order of addition, type of salt, salt concentration, half time, its standard deviation and elapsed time with its standard deviation for systems with no added buffer.

Polycation	**Polyanion**	**Order**	**Salt Type**	**[Salt] (mM)**	**$t_{1/2}$ (min)**	**SD (min)**	**t_e (min)**	**SD (min)**
K_4G_4	E_4G_4	P/N	-	-	14.34	1.6	80.0	5.0
K_8G_8	E_8G_8	P/N	-	-	16.3	0.9	55.0	5.0
K_{100}	E_{50}	P/N	-	-	23.7	0.7	60.0	5.0
K_{50}	E_{100}	P/N	-	-	8.8	0.7	70.0	5.0
K_{400}	E_{400}	P/N	-	-	14.9	3.0	45.0	5.0
K_{50}	E_{50}	P/N	-	-	16.3	1.4	65.0	5.0
K_4G_4	E_4G_4	N/P	-	-	14.6	2.6	80.0	5.0
K_8G_8	E_8G_8	N/P	-	-	12.5	1.7	55.0	5.0
K_{100}	E_{50}	N/P	-	-	7.5	5.8	60.0	5.0
K_{50}	E_{100}	N/P	-	-	31.6	4.6	65.0	5.0
K_{400}	E_{400}	N/P	-	-	4.7	3.6	65.0	5.0
K_{50}	E_{50}	N/P	-	-	31.6	1.5	65.0	5.0
K_{800}	E_{800}	N/P	-	-	21.3	1.5	60.0	5.0
K_{800}	E_{800}	P/N	-	-	15.3	1.0	35.0	5.0
K_2G_2	E_2G_2	N/P	-	-	35.0	1.1	40.0	5.0
$(KG)_{25}$	$(EG)_{25}$	N/P	KBr	10	10.4	0.2	5.0	5.0
$(KG)_{25}$	$(EG)_{25}$	N/P	KCl	10	57.5	2.1	90.0	5.0
$(KG)_{25}$	$(EG)_{25}$	N/P	NaBr	10	4.0	0.8	65.0	5.0
$(KG)_{25}$	$(EG)_{25}$	N/P	NaCl	10	47.5	1.3	75.0	5.0
$(KG)_{25}$	$(EG)_{25}$	N/P	KBr	25	2.5	0.2	5.0	5.0
$(KG)_{25}$	$(EG)_{25}$	N/P	KCl	25	2.5	0.2	5.0	5.0
$(KG)_{25}$	$(EG)_{25}$	N/P	NaBr	25	17.5	0.3	20.0	5.0
$(KG)_{25}$	$(EG)_{25}$	N/P	NaCl	25	57.5	4.1	60.0	5.0
$(KG)_{25}$	$(EG)_{25}$	N/P	-	-	21.3	1.0	85.0	5.0
K_{50}	$(EG)_{25}$	N/P	-	-	8.5	1.2	65.0	5.0
K_2G_2	E_2G_2	N/P	-	-	13.7	0.4	60.0	5.0
$(KG)_{25}$	$(EG)_{25}$	P/N	KBr	10	2.5	0.3	5.0	5.0
$(KG)_{25}$	$(EG)_{25}$	P/N	KCl	10	22.5	3.1	25.0	5.0
$(KG)_{25}$	$(EG)_{25}$	P/N	NaBr	10	2.5	0.3	5.0	5.0
$(KG)_{25}$	$(EG)_{25}$	P/N	NaCl	10	8.4	1.2	50.0	5.0
$(KG)_{25}$	$(EG)_{25}$	P/N	KBr	25	0.0	-	0.0	-
$(KG)_{25}$	$(EG)_{25}$	P/N	KCl	25	3.4	1.0	30.0	5.0
$(KG)_{25}$	$(EG)_{25}$	P/N	NaBr	25	17.5	0.2	20.0	5.0
$(KG)_{25}$	$(EG)_{25}$	P/N	NaCl	25	63.3	0.4	85.0	5.0
$(KG)_{25}$	$(EG)_{25}$	P/N	-	-	9.9	1.2	80.0	5.0
$(KG)_{25}$	E_{50}	P/N	-	-	4.0	1.9	65.0	5.0
K_{100}	E_{100}	P/N	-	-	17.5	2.4	80.0	5.0
K_{50}	$(EG)_{25}$	P/N	-	-	46.2	2.8	80.0	5.0
K_{100}	E_{100}	N/P	-	-	14.4	4.9	85.0	5.0
$(KG)_{25}$	E_{50}	N/P	-	-	8.7	1.9	75.0	5.0

Table S2. Table of counter-ions, molecular weights with (M_{nc}) and without (M_n) the counter-ions, and polydispersity index (PDI) of polypeptides. Polypeptides of chain length 50 were made in house, while N = 100, 400, and 800 were purchased from Alamanda Polymers.

		Counter Ion[a]	M_{nc} (g/mol)	M_n (g/mol)	PDI[e]	$\langle R_g^2\rangle$[f] (Å)
50	K_{50}	TFA^-	12,000	6,400	-	204
	E_{50}[b]	Na^+	7,600	6,500	-	204
	KG[c]	TFA^-	7,500	4,600	-	204
	EG[c]	Na^+	5,200	4,700	-	204
100	K_{100}	Br^-	21,000	13,000	1.06	825
	E_{100}[d]	Na^+	15,000	13,000	1.02	825
400	K_{400}	TFA^-	97,000	51,000	1.08	1630
	E_{400}[d]	Na^+	60,000	52,000	1.01	1630
800	K_{800}	TFA^-	194,000	103,000	1.06	3270
	E_{800}[d]	Na^+	120,000	103,000	1.06	3270

[a] TFA is defined as trifluoroacetate.
[b] E_{50} was made in house and is composed of alternating D and L monomers.
[c] Patterned peptides have the same number of glycine and lysine/glutamate residues and have the same molecular weight.
[d] E_n was purchased from Alamanda Polymers and is racemic, but without sequence control.
[e] PDI was reported by the manufacturer based on characterization using gel permeation chromatography (GPC).
[f] Mean squared radius of gyration was estimated assuming ideal flexible chain behavior $\langle R_g^2\rangle = \frac{b^2N}{6}$ where b is the Kuhn length (7 Å) and N is the number of Kuhn monomers per chain approximated as two chemical monomers.[1-3] We therefore estimate the persistence length for each of our polymers to be ~3.5 Å, corresponding to a single animo acid residue.

Gaussian Fits

Gaussian fits were calculated using a MATLAB script. Based on preliminary plots, a unimodal of Gaussian curves were used. The mathematics behind the code will be discussed in the next sections.

To use the code, the data was imported as a matrix with the first column comprised of the various time points. The second column lists the different charge fractions, defined as the ratio of positive charges over the total number of positive and negative charges. The following columns list the turbidity readings of each charge fraction as a function of time with its corresponding standard deviation. Three samples at each charge fraction for each order of addition were made and each sample was read three times for nine total turbidity readings per stoichiometric point for both negative first and positive first runs.

The data was separated into its constituents and then the maximum turbidity for each time point series was determined. This maximum value was used as a guess for the peak of the Gaussian curve for its respective time. Next, the mole fraction corresponding to this peak was found and used as a guess for the point on the x-axis for the peak location. The standard deviation of the turbidity readings for that time point is then calculated and used as the standard deviation for the Gaussian curve.

A Gaussian curve follows the equation:

$$G = Ae^{-\frac{(x-b)^2}{2c^2}} \tag{S2}$$

where A is the height of the peak, b is the position of the center of the peak, c is the standard deviation of the curve, and x is the independent variable, which is the charge fraction. Using the guesses from above, a second function was written to minimize the sum of the squared error between the raw data and the curve from the Gaussian fit. This function changed the values of A, b, and c to find the set that would lead to the best fit. Once found, the raw data and Gaussian fit could be plotted together.

Plotting and Determining Kinetics

In addition to Gaussian fits, plots of peak location versus time were made. This was done using the max values were found using MATLAB and the max() function plotted versus time. We defined the time to reach equilibrium as the time point at which the peak location first reached its maximum or minimum, depending on the order of addition. In addition to this elapsed time calculation, we also evaluated our data by determining the “half-time,” or the time required to reach half of the maximum or minimum value (Figure 2b,d). This halftime was found via linear interpolation between points for the time corresponding to peak$_{1/2}$.

$$peak_{1/2} = \frac{peak_{max}+peak_{min}}{2} \tag{S3}$$

The upper and lower bounds of this half time were calculated using the same method along the upper and lower bound curves. A list of all the halftimes and their corresponding standard deviations are listed in Table S1.

Reynolds Number and Characteristic Time Calculations

Calculating the Reynolds Number

To calculate the Reynolds number (Re), the following equation was used:

$$Re = \frac{\rho v l}{\mu} \tag{S4}$$

where ρ is the density, v is the velocity, l is the height of the liquid level in the well, and μ is the dynamic viscosity. The system is assumed to be dilute with respect to the coacervate (~2 mM polyelectrolyte concentration on a monomer basis) such that the density and viscosity of water was used: ρ is 0.997 g/mL or 0.000997 kg/mL and μ is 0.890 cP or 0.00089 kg/(cm*s) at 25°C. A 384-well plate has the dimensions of a truncated pyramid with the side of the bottom rounded square equal to 3.130 mm and the side of the top rounded square 3.144 mm. For simplicity, and using 35 µL as the volume, 3.130 mm was used for l. The speed of the orbital shaker was 950 rpm and was converted to cm/s by:

$$v = rpm * C \tag{S5}$$

where C is the circumference of the well, calculated by assuming the side of the well (3.130 mm) is the diameter of a circle and using $C = \pi d$. The velocity was calculated to be approximately 15.5 cm/s. Using the equation for the Reynolds number, Re ~ 546 for the 384-well plate categorizing the flow as laminar.

The same approach was used for the round 96-well plate with the bottom diameter equal to 6.35 mm and the top equal to 6.85 mm, thus using 6.35 mm as l. The velocity was calculated as described above resulting in an approximate velocity of 31.6 cm/s. Using these values and the physical properties of water, the Reynolds number was calculated to be Re ~ 2247, in the regime of transitional flow.

Characteristic Time for Diffusive Mixing

To further expand the discussion of diffusive versus convective transport, we calculated a characteristic time by:

$$\tau = \frac{L^2}{2D} \tag{S6}$$

where L is the distance over which diffusion occurs and D is the diffusivity. We assumed this time would be for the polymer to diffuse half way across the well, resulting in a time of ~3.4 hr and ~14.0 hr for the 384- and 96-well plates, respectively. The relative magnitude of this timeframe in the 384-well plate is of the same order of magnitude as the 90 min length of our experiments.

MATLAB Script

```
function Linear_Interp_Stats_Perry

clear,clc % clears command window and workspace

files = dir('*.xlsx')'; % required for multiple file runs

for runs = 1:length(files) % required for multpile file runs
namefile = files(runs).name % required for multiple file runs

data = xlsread(namefile);

% Note that the sheet you inport can only have a header line and then the
% data that you are inporting. Make sure that you are in the current folder
% or be sure to write the whole extension starting with the hard drive and
% navigating to the correct file. Use doc xlsread if you are unsure how to
% import data using xlsread.

% For your data, the first column is your times. The second is your x.
% The third, and every 2 after, is your y data. The column that is skipped
% is the standard deviation for that y column:
% times x y std y std y std y std...

time = data(:,1); % makes time array
for j = 1:length(time)
    times = time(~isnan(time)); % gets rid of any NaN
end

molfrac = data(:,2);
for k = 1:length(molfrac)
    mol_frac = molfrac(~isnan(molfrac)); % gets rid of any NaN
end
    % Uses the second column as the mole fractions
[~, numcol] = size(data);
readings = data(:,3:numcol);
readings(~any(~isnan(readings), 2),:)=[]; % gets rid of any NaN
[~,readcol] = size(readings);
    % The rest of the columns are the reads for the data set

colorVec = hsv(readcol); % creates a vector of rainbow colors
legend_names = cell(1,readcol/2); % generates a vector for the legend
peaks = zeros(readcol/2,1); % creates a preallocated vector
peak_st = zeros(readcol/2,1);
c1 = zeros(readcol/2,3); % preallocated vector for Gaussian coef

file_names{runs} = namefile;

figure(2*runs-1)
for i = 1:readcol
    if mod(i,2) == 1
        sigma = std(readings(:,i)); % standard deviation of data set
        guess(1) = max(readings(:,i)); % guess for amplitude / highest y value
        [r,s] = find(readings(:,i)==guess(1));
        guess(2) = mol_frac(r(1),s(1)); % guess for x where y is highest
            % mole fraction of max, specifying first as some
            % runs try to generate multiple peaks
        guess(3) = sigma; % guess for standard deviation
        gaus = @(guess,mol_frac)(guess(1).*exp((-(mol_frac-
guess(2)).^2)./(2*guess(3)^2)));
        y = readings(:,i); % corresponding y data for fitting
        [beta, ~, ~,cov] = nlinfit(mol_frac,y,gaus,guess);
        coef_sd = sqrt(diag(cov)); % obtains std dev of each coef
        c1((i+1)/2,:) = beta;

        % Creating and plotting the Gaussian
        x = 0:0.001:1; % mole fraction values for fit
        gaussian = beta(1).*exp((-(x-beta(2)).^2)./(2*beta(3)^2)); % fit
        peaks((i+1)/2) = beta(2); % mole fraction peak for set of data
        peak_st((i+1)/2) = coef_sd(2);
        plot(x,gaussian,'Color',colorVec(i,:))
        hold on
        fwhm((i+1)/2,:) = 2.*sqrt(2.*log(2)).*beta(3);
    end
end

FWHM0(runs) = fwhm(1);
FWHM90(runs) = fwhm(19);

for i = 1:readcol
    if mod(i,2) == 1
        errorbar(mol_frac,readings(:,i),readings(:,i+1),'o','Color',...
            colorVec(i,:)) % plotting data with error bars
         legend_names{((i+1)/2)} = [num2str(times((i+1)/2)) ' min']; % generating
legend names
        hold on
    end

end

lgd = legend(legend_names);

legend('boxoff')
hold off
xtickformat('%.1f')
ytickformat('%.2f')
set(findall(gcf,'-property','FontSize'),'FontSize',8)
xlabel('Charge Fraction K(+)
(mol/mol)','fontsize',10,'fontweight','bold','FontName','Arial')
ylabel('Turbidity (a.u.)','fontsize',10,'fontweight','bold','FontName','Arial')

lgd.FontSize = 6;
% Note: Use sprintf() to include subtitles by using \n

set(figure(2*runs-1), 'PaperUnits', 'centimeters','PaperPosition',[0 0 8.25 8.25]);
print(figure(2*runs-1), '-depsc', num2str([namefile '_turb.eps']),'-r600');

figure(2*runs)
for i = 1:readcol
    if mod(i,2) == 1
       m((i+1)/2) = max(readings(:,i));
       [r,s] = find(readings(:,i)==m((i+1)/2));
       n((i+1)/2) = mol_frac(r(1),s(1));
       sdr = readings(r(1),s(1)+1);
       ll((i+1)/2) = n((i+1)/2) - 2.306.*sdr./sqrt(9);
       ul((i+1)/2) = n((i+1)/2) + 2.306.*sdr./sqrt(9);

       errorbar(times((i+1)/2),n((i+1)/2),sdr,'o','Color',...
            colorVec(i,:)) % plotting data, add in error bars
       legend_names{(i+1)/2} = [num2str(times((i+1)/2)) ' min']; % generating legend
names
```

```
        hold on
    end
end

nhalf = (max(n)+min(n))./2; % Calculating the stoichiometry for t1/2
 [pall] = find(n==max(n)); % determine the increments corresponding to the maximum
value of n
[uall] = find(n==min(n)); % determine the increments corresponding to the minimum
value of n

if n(1) < nhalf % used for experiments which start at low stoichiometry and increase

    [p]=[pall(1)]; % determine the index of the lowest index corresponding to the
maximum
    if size(uall,2) == 1
        [u]=[uall(1)]; % determine the index of the highest index corresponding to the
minimum
    else
        [u]=size(uall,2); % determine the index of the highest index corresponding to
the minimum
    end

else % used for experiments which start at high stoichiometry and decrease

    [u]=[uall(1)]; % determine the index of the lowest index corresponding to the
minimum
    if size(pall,2) == 1
        [p]=[pall(1)]; % determine the index of the highest index corresponding to the
minimum
    else
        [p]=size(pall,2); % determine the index of the highest index corresponding to
the minimum
    end
end

minll = ll(u); % determine the value of the lowest increment for the bottom asymptote
minul = ul(u); % determine the value of the lowest increment for the upper asymptote
maxll = ll(p); % determine the value of the highest increment for the lower asymptote
maxul = ul(p); % determine the value of the highest increment for the upper asymptote
mll = (minll - maxll)./(times(u)-times(p)); % calculate the slope of the line for the
lower error bound
mul = (minul - maxul)./(times(u)-times(p)); % calculate the slope of the line for the
upper error bound
bll = minll - mll.*times(u); % calculate the y-intercept of the line for the lower
error boundary
bul = minul - mul.*times(u); % calculate the y-intercept of the line for the upper
error boundary

if n(1) < nhalf % used for experiments which start at low stoichiometry and increase
    mline = (max(n)-min(n))./(times(u)-times(p)); % calculate the slope of the line
for the actual data
    bline = max(n) - mline.*times(u); % calculate the y-intercept of the line for the
actual data
    telapsed(runs) = times(p);
else
    mline = (max(n)-min(n))./(times(p)-times(u)); % calculate the slope of the line
for the actual data
    bline = max(n) - mline.*times(p); % calculate the y-intercept of the line for the
actual data
    telapsed(runs) = times(u);
end
```

```
lt = [times(u) times(p)]; % times for the bounds
llm = [minll maxll]; % Calculating the stoichiometry values for the upper error bound
ulm = [minul maxul]; % Calculating the stoichiometry values for the upper error bound
tl(runs) = (nhalf - bll)./mll; % Calculating the time for the upper error bound
tu(runs) = (nhalf - bul)./mul; % Calculating the time for the lower error bound
th(runs) = (nhalf - bline)./mline; % Calculating the time for n1/2

CI_ul(runs) = abs(tu(runs) - th(runs)); % Calculating the upper boundary of the
confidence interval
CI_ll(runs) = abs(tl(runs) - th(runs)); % Calculating the lower boundary of the
confidence interval

tvl = [tl(runs) tl(runs)]; % indices for plotting the lines
tvu = [tu(runs) tu(runs)];
mvl = [0 nhalf];
mvu = [0 nhalf];
tvh = [th(runs) th(runs)];
mvh = [0 nhalf];

tspline = 0:0.001:90;
nspline = interp1(times,n,tspline);
nspline_ll = interp1(times,ll,tspline);
nspline_ul = interp1(times,ul,tspline); %spline

if n(1) < nhalf
    n_ub = find(nspline<nhalf, 1, 'last'); % When the charge first becomes positive
    n_lb = find(nspline>nhalf, 1, 'first'); % When the charge is last to be negative
else
    n_ub = find(nspline<nhalf, 1, 'first'); % When the charge first becomes positive
    n_lb = find(nspline>nhalf, 1, 'last'); % When the charge is last to be negative
end

n_half(runs) = nhalf;
thalf(runs) = (tspline(n_ub)+tspline(n_lb))./2;
% tll(runs) = (tspline(ll_ub)+tspline(ll_lb))./2;
% tul(runs) = (tspline(ul_ub)+tspline(ul_lb))./2;

tul(runs) = thalf(runs) + CI_ul(runs);
tll(runs) = thalf(runs) - CI_ll(runs);

tulb = [tul(runs) tul(runs)];
tllb = [tll(runs) tll(runs)];
thb = [thalf(runs) thalf(runs)];

plot(tspline,nspline_ll,'b',tspline,nspline_ul) %changed from ll3 and ul3
plot(lt, llm, 'k-', lt, ulm, 'k-')
plot(tvl, mvl, 'k-', tvu, mvu, 'k-', tvh, mvh, 'b*-')
plot(tllb, mvl, 'k-', tulb, mvu, 'k-', thb, mvh, 'b*-')
plot(times,ll,'k:',times,ul,'k:',thalf(runs),n_half(runs),'kd')
hold off

set(findall(gcf,'-property','FontSize'),'FontSize',8)
xlabel('Time (min)','fontsize',10,'fontweight','bold','FontName','Arial')
ylabel('Peak Location (mol/mol)','fontsize',10,'fontweight','bold','FontName','Arial')

set(figure(2*runs), 'PaperUnits', 'centimeters','PaperPosition',[0 0 8.25 8.25]);
print(figure(2*runs), '-depsc', num2str([namefile '_peaks.eps']),'-r600');

end
```

```
T =
table(thalf',tll',tul',telapsed',FWHM0',FWHM90','VariableNames',{'t_half','tll','tul',
'telapsed','FWHM0','FWHM90'},'RowNames',file_names)
writetable(T,'Kinetics_Symmetry_Data.xlsx','WriteRowNames',true)

end
```

References

(1) Marciel, A. B.; Srivastava, S.; Tirrell, M. V. Structure and rheology of polyelectrolyte complex coacervates. *Soft Matter* **2018**, *14* (13), 2454–2464 DOI: 10.1039/C7SM02041D.

(2) Rubinstein, M.; Colby, R. H. *Polymer Physics*; Oxford University Press, 2003.

(3) Hanke, F.; Serr, A.; Kreuzer, H. J.; Netz, R. R. Stretching single polypeptides: The effect of rotational constraints in the backbone. *EPL* **2010**, *92* (5), 53001–53007 DOI: 10.1209/0295-5075/92/53001.