Electronic Supplementary Material (ESI) for ChemComm.

Fitting thermolabile ligand DSC datasets

This program has three main components.

1. Prepare data and call fitting function fminsearch.

2. Generate synthetic data using input parameters according to the model of your choice.

3. Compute the quality of the fit in the merit function.

The fitting function fminsearch increments parameters and iterates steps 2 and 3 until the optimized parameters are found.

Initialize program, call fminsearch to initiate fit

```
function [Cpcalc, params] = thermolabileligandDSC(T, Cp, params)

% Convert temperature to Kelvin
T = T + 273;

% Call fminsearch
[params, rss] = fminsearch(@merit, params, options, T, Cp);

% Generate and plot synthetic optimized data for model assuming folded, bound, and
% unfolded states
[Cpcalc, Cp_folded, PFL1, PFL2, PF, PU] = genCp(T, params);

cmap = lbmap(size(Cp,2),'RedBlue');
figure
for i = 1:size(Cp,2)
    hold on
    plot(T(1:10:end,1)-
273,Cp(1:10:end,i),'o','MarkerSize',6,'MarkerFaceColor',cmap(i,:),'MarkerEdgeColor',cmap(i,:))
    plot(T(1:end-1,1)-273,Cpcalc(:,i),'color',cmap(i,:),'LineWidth',2)
end
hold off
xlim([20 80])
set(gca,'LineWidth',2,'FontSize',12,'XColor','k','YColor','k')
set(gcf,'color','w')
xlabel('Temperature °C')
ylabel('C_{p} kJ mol^{-1} K^{-1}')
box on

figure
plot(T-273,PFL1,'--','color',[0.8500 0.3250 0.0980],'LineWidth',2)
hold on
```

```
plot(T-273,PFL2,'--','color',[0.4660 0.6740 0.1880],'LineWidth',2)
plot(T-273,PF,'--','color',[0 0.4470 0.7410],'LineWidth',2)
plot(T-273,PU,'color',[0.7 0.7 0.7],'LineWidth',2)
hold off
xlim([20 80])
ylim([-0.01 1.01])
set(gca,'LineWidth',2,'FontSize',12,'XColor','k','YColor','k')
set(gcf,'color','w')
xlabel('Temperature °C')
ylabel('Population')
box on

% Generate and plot synthetic optimized data for model assuming bound and
% unfolded states
[Cpcalc, Cp_folded, PFL1, PFL2, PU] = genCp(T, params);

cmap = lbmap(size(Cp,2),'RedBlue');
figure
for i = 1:size(Cp,2)
    hold on
    plot(T(1:10:end,1)-
273,Cp(1:10:end,i),'o','MarkerSize',6,'MarkerFaceColor',cmap(i,:),'MarkerEdgeColor',cmap(i,:))
    plot(T(1:end-1,1)-273,Cpcalc(:,i),'color',cmap(i,:),'LineWidth',2)
end
hold off
xlim([10 60])
set(gca,'LineWidth',2,'FontSize',12,'XColor','k','YColor','k')
set(gcf,'color','w')
xlabel('Temperature °C')
ylabel('C_{p} kJ mol^{-1} K^{-1}')
box on

figure
plot(T-273,PFL1,'--','color',[0.8500 0.3250 0.0980],'LineWidth',2)
hold on
plot(T-273,PFL2,'--','color',[0.4660 0.6740 0.1880],'LineWidth',2)
plot(T-273,PU,'color',[0.7 0.7 0.7],'LineWidth',2)
hold off
xlim([10 60])
ylim([-0.01 1.01])
set(gca,'LineWidth',2,'FontSize',12,'XColor','k','YColor','k')
set(gcf,'color','w')
xlabel('Temperature °C')
ylabel('Population')
box on
end
```

Generate synthetic data using input parameters, compute quality of fit

```
function [rss] = merit(params, T, Cp)

% Generate synthetic data for model assuming folded, bound, and unfolded
```

```
% states
[Cpcalc, Cp_folded, PFL1, PFL2, PF, PU] = genCp(T, params);

% Generate synthetic data for model assuming bound and unfolded states
[Cpcalc, Cp_folded, PFL1, PFL2, PU] = genCp(T, params);

% Compute rss as measure of fit quality using experimental and synthetic
% data
for i = 1:size(Cp,2)
rss(i,1) = sum((Cp(1:end-1,i) - Cpcalc(:,i)).^2);
end
rss = sum(rss);

% Model assumes folded, bound, and unfolded states
% Plot synthetic and experimental data to visually identify fit progress
pause(0.001)
subplot(1,2,1)
plot(T-273,Cp,'k')
hold on
plot(T(1:end-1,1)-273, Cpcalc,'r')
plot(T-273,Cp_folded,'b')
hold off

subplot(1,2,2)
plot(T-273,PF,'b')
hold on
plot(T-273,PFL1,'r')
plot(T-273,PFL2,'m')
plot(T-273,PU,'color',[0.7 0.7 0.7])
hold off

% Model assumes bound and unfolded states
% Plot synthetic and experimental data to visually identify fit progress
pause(0.001)
subplot(1,2,1)
plot(T-273,Cp,'k')
hold on
plot(T(1:end-1,1)-273, Cpcalc,'r')
plot(T-273,Cp_folded,'b')
hold off

subplot(1,2,2)
plot(T-273,PFL1,'r')
hold on
plot(T-273,PFL2,'m')
plot(T-273,PU,'color',[0.7 0.7 0.7])
hold off
end
```

Generate synthetic data with model assuming folded, bound, and unfolded states

```
function [Cpcalc, Cp_folded, PFL1, PFL2, PF, PU] = genCp(T, params)

% Calculate baseline, concentration, and thermodynamic parameters using
% inputs
x = params(1);
y = params(2);
z = params(3);

dHo = params(4);
dSo = params(5);

dHL1o = params(6);
dSL1o = params(7);
dHL2o = params(8);
dSL2o = params(9);
dCpL1 = params(10);
dCpL2 = params(11);

LTo = 778e-6;

LT1 = zeros(8,1);
LT1(1,1) = 778e-6;
LT1(2:7,1) = params(12:17);
LT1(8,1) = LTo - LTo;

LT2 = zeros(8,1);
LT2(1,1) = LTo - LT1(1);
for j = 2:7
LT2(j,1) = LTo - LT1(j,1);
end
LT2(8,1) = LTo;

CT = 83e-6;
R = 8.3145e-3;

To = T(1,1);
Cp_folded = x + y.*(T - To) + z.*(T - To).^2;
dCp = 0;

dH = dHo + dCp.*(T - To);
dS = dSo + dCp.*log((T./To));
dHL1 = dHL1o + dCpL1.*(T - To);
dSL1 = dSL1o + dCpL1.*log((T./To));
dHL2 = dHL2o + dCpL2.*(T - To);
dSL2 = dSL2o + dCpL2.*log((T./To));
K = exp(((-dH./(R.*T)) + (dS./R)));
KL1 = exp(((-dHL1./(R.*T)) + (dSL1./R)));
KL2 = exp(((-dHL2./(R.*T)) + (dSL2./R)));

p = zeros(size(K,1),4);
```

```
for j = 1:size(LT1,1)

    % Set up polynomial coefficients and solve concentration of free folded
    % state
    p(:,1) = K.*KL1.*KL2 + KL1.*KL2;
    p(:,2) = K.*KL1 + K.*KL2 + KL1 + KL2 + LT1(j,1).*KL1.*KL2 ...
    + LT2(j,1).*KL1.*KL2 - CT.*KL1.*KL2;
    p(:,3) = LT1(j,1).*KL1 + LT2(j,1).*KL2 + K + 1 - CT.*KL1 - CT.*KL2;
    p(:,4) = -CT;

   for i = 1:size(T,1)
       r(i,:) = roots(p(i,:));
       for k = 1:size(r,2)
           if isreal(r(i,k)) == 1 && r(i,k) > 0
               F(i,j) = r(i,k);
           end
       end
   end

    % Compute populations and calculate Cp curve
    L1 = LT1(j,1)./(1 + KL1.*F(:,j));
    L2 = LT2(j,1)./(1 + KL2.*F(:,j));

    Z = 1 + K + L1.*KL1 + L2.*KL2;
    U = K.*F(:,j);
    PU(:,j) = K./Z;
    PF(:,j) = 1./Z;
    PFL1(:,j) = (L1.*KL1)./Z;
    PFL2(:,j) = (L2.*KL2)./Z;

    dPFL1dT(:,j) = diff(PFL1(:,j))./0.1;
    dPFL2dT(:,j) = diff(PFL2(:,j))./0.1;
    dPUdT(:,j) = diff(PU(:,j))./0.1;

    Cpcalc(:,j) = Cp_folded(1:end-1,1) + dH(1:end-1,1).*dPUdT(:,j) + dHL1(1:end-
1,1).*dPFL1dT(:,j) +...
        dHL2(1:end-1,1).*dPFL2dT(:,j) + PU(1:end-1,j).*dCp +...
        PFL1(1:end-1,j).*dCpL1 + PFL2(1:end-1,j).*dCpL2;
end
end
```

Generate synthetic data with model assuming bound and unfolded states

```
function [Cpcalc, Cp_folded, PFL1, PFL2, PU] = genCp(T, params)

% Calculate baseline, concentration, and thermodynamic parameters using
% inputs
x = params(1);
y = params(2);
z = params(3);

dHL1o = params(4);
```

```
dSL1o = params(5);
dHL2o = params(6);
dSL2o = params(7);
dCpL1 = params(8);
dCpL2 = params(9);

LTo = 778e-6;

LT1 = zeros(8,1);
LT1(1,1) = 778e-6;
LT1(2:7,1) = params(10:15);
LT1(8,1) = LTo - LTo;

LT2 = zeros(8,1);
LT2(1,1) = LTo - LT1(1);
for j = 2:7
LT2(j,1) = LTo - LT1(j,1);
end
LT2(8,1) = LTo;

CT = 83e-6;
R = 8.3145e-3;

To = T(1,1);
Cp_folded = x + y.*(T - To) + z.*(T - To).^2;
dCp = 0;

dHL1 = dHL1o + dCpL1.*(T - To);
dSL1 = dSL1o + dCpL1.*log((T./To));
dHL2 = dHL2o + dCpL2.*(T - To);
dSL2 = dSL2o + dCpL2.*log((T./To));
KL1 = exp(((-dHL1./(R.*T)) + (dSL1./R)));
KL2 = exp(((-dHL2./(R.*T)) + (dSL2./R)));

p = zeros(size(KL1,1),4);

for j = 1:size(LT1,1)

    % Set up polynomial coefficients and solve for concentration of
    % unfolded state
    p(:,1) = KL1.*KL2;
    p(:,2) = -KL1.*KL2.*CT + KL1 + KL2 + KL1.*KL2.*LT2(j,1) + KL1.*KL2.*LT1(j,1);
    p(:,3) = KL1.*LT1(j,1) + KL2.*LT2(j,1) + 1 - CT.*KL2 - CT.*KL1;
    p(:,4) = -CT;

    for i = 1:size(T,1)
        r(i,:) = roots(p(i,:));
        for k = 1:size(r,2)
            if isreal(r(i,k)) == 1 && r(i,k) > 0
                U(i,j) = r(i,k);
            end
        end
    end
```

```
    % Compute populations and calculate Cp curve
    FL1 = (KL1.*LT1(j,1).*U(:,j))./(1 + KL1.*U(:,j));
    FL2 = (KL2.*LT2(j,1).*U(:,j))./(1 + KL2.*U(:,j));

    PU(:,j) = U(:,j)./CT;
    PFL1(:,j) = FL1./CT;
    PFL2(:,j) = FL2./CT;

    dPFL1dT(:,j) = diff(PFL1(:,j))./0.1;
    dPFL2dT(:,j) = diff(PFL2(:,j))./0.1;

    Cpcalc(:,j) = Cp_folded(1:end-1,1) + dHL1(1:end-1,1).*dPFL1dT(:,j) + ...
        dHL2(1:end-1,1).*dPFL2dT(:,j) + PFL1(1:end-1,j).*dCpL1 + PFL2(1:end-1,j).*dCpL2;
end
end
```

Published with MATLAB® R2016a