Supporting Material for "Wide-angle X-ray Solution Scattering for Protein-Ligand Binding: Multivariate Curve Resolution with Bayesian Confidence Intervals"

David D.L. Minh
Biosciences Division,
Argonne National Laboratory, Argonne, IL [*]

Lee Makowski
Department of Chemistry and Chemical Biology
and Department of Electrical and Computer Engineering,
Northeastern University, Boston, MA

December 12, 2012

This supporting material contains WAXS data collection details, three figures, and MATLAB code to perform the described analysis.

Wide-angle X-ray solution scattering collection details

The sample cell consisted of a thin-walled quartz capillary (1.5 mm in diameter) attached to a programmable pump (Hamilton; Microlab 500 series) that was adjusted to deliver continuous flow through the capillary during data collection. The ambient temperature of the air surrounding the capillary and sample tubing was held at 4 °C by attachment of an ethylene glycol bath to the brass capillary holder and through an outer layer of tubing surrounding the inner sample tubing. X-ray scattering patterns were recorded with a MAR165 2k × 2k CCD detector. The specimen-to-detector distance was approximately 170 mm and was calibrated using powder diffraction rings from silver-behenate. The beamline is capable of delivering approximately 2×10^{13} photons/s per 100 mA of beam current. As previous experience on the BioCAT beamline has demonstrated that proteins under a variety of physical conditions are damaged after exposure times of a few tenths of a second to a few seconds at these intensity levels, in these experiments 8 to 32 20 μm aluminum foils were used (depending on the concentration of protein in the samples) as X-ray beam attenuators to control the incident beam flux. In most cases, five exposures of solution with buffer and ligand were followed by at least seven of solution containing buffer, ligand, and protein, with each exposure lasting about 2 seconds. Incident beam flux was monitored using nitrogen gas-filled ion chambers placed directly before the sample chamber and an active beam stop to monitor x-ray flux after the sample. No apparent differences in normalization were observed between the two measures of intensity, indicating that changes in ligand concentration made no observable change in absorption. We arbitrarily chose to use the ion chamber measurements for normalization, but could equally well have used data from the active beam stop which would have resulted in indistinguishable measures of scattered intensity. During data collection, the solutions were flowed so that no protein was exposed to the beam for more that 100 ms.

[*]Current address: Department of Chemistry, Duke University, Durham, NC

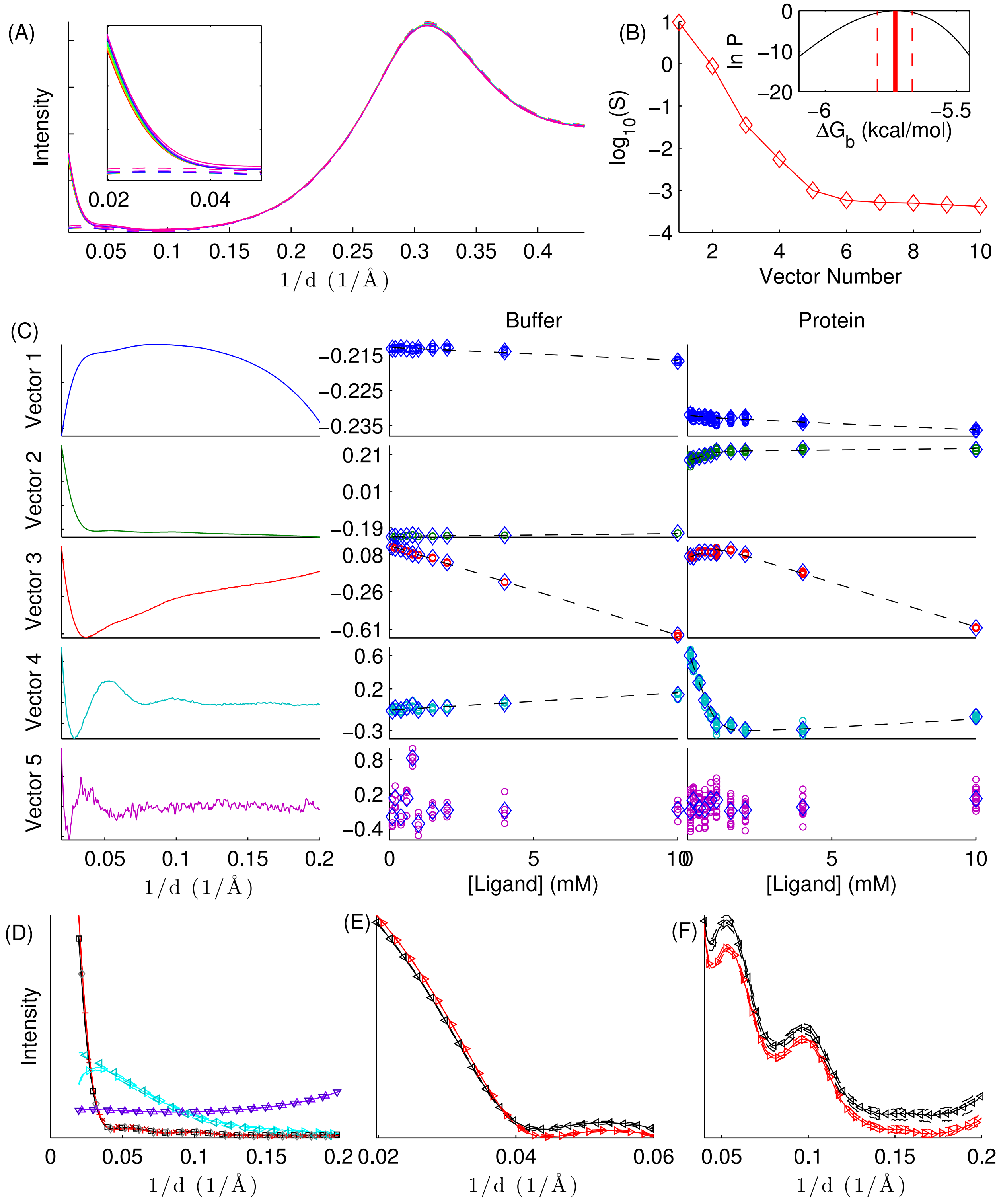

Figure S 1: Analysis of Lysozyme-$(NAG)_2$ Binding. The caption is the same as in Fig. 2.

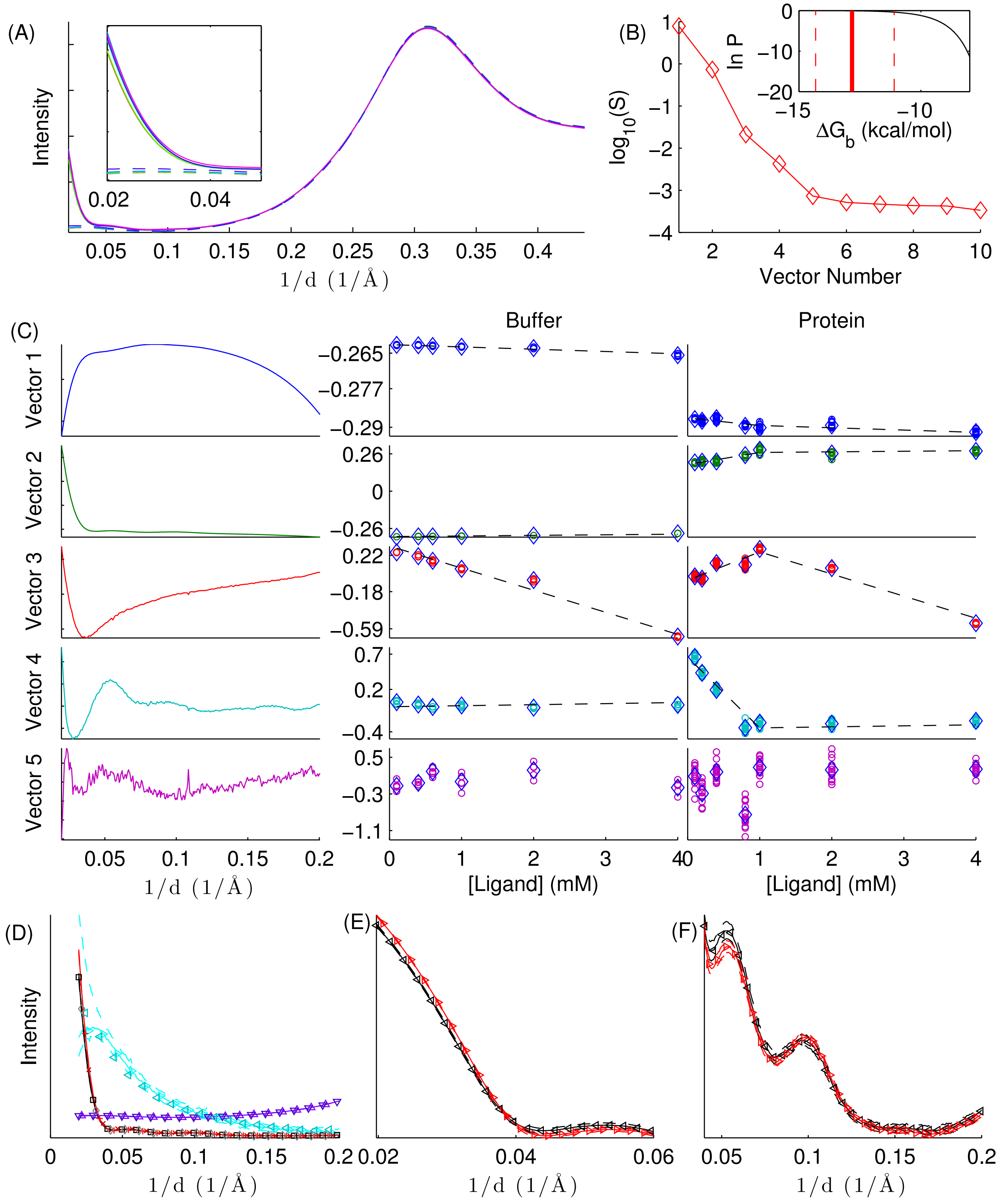

Figure S 2: Analysis of Lysozyme-$(NAG)_3$ Binding. The caption is the same as in Fig. 2.

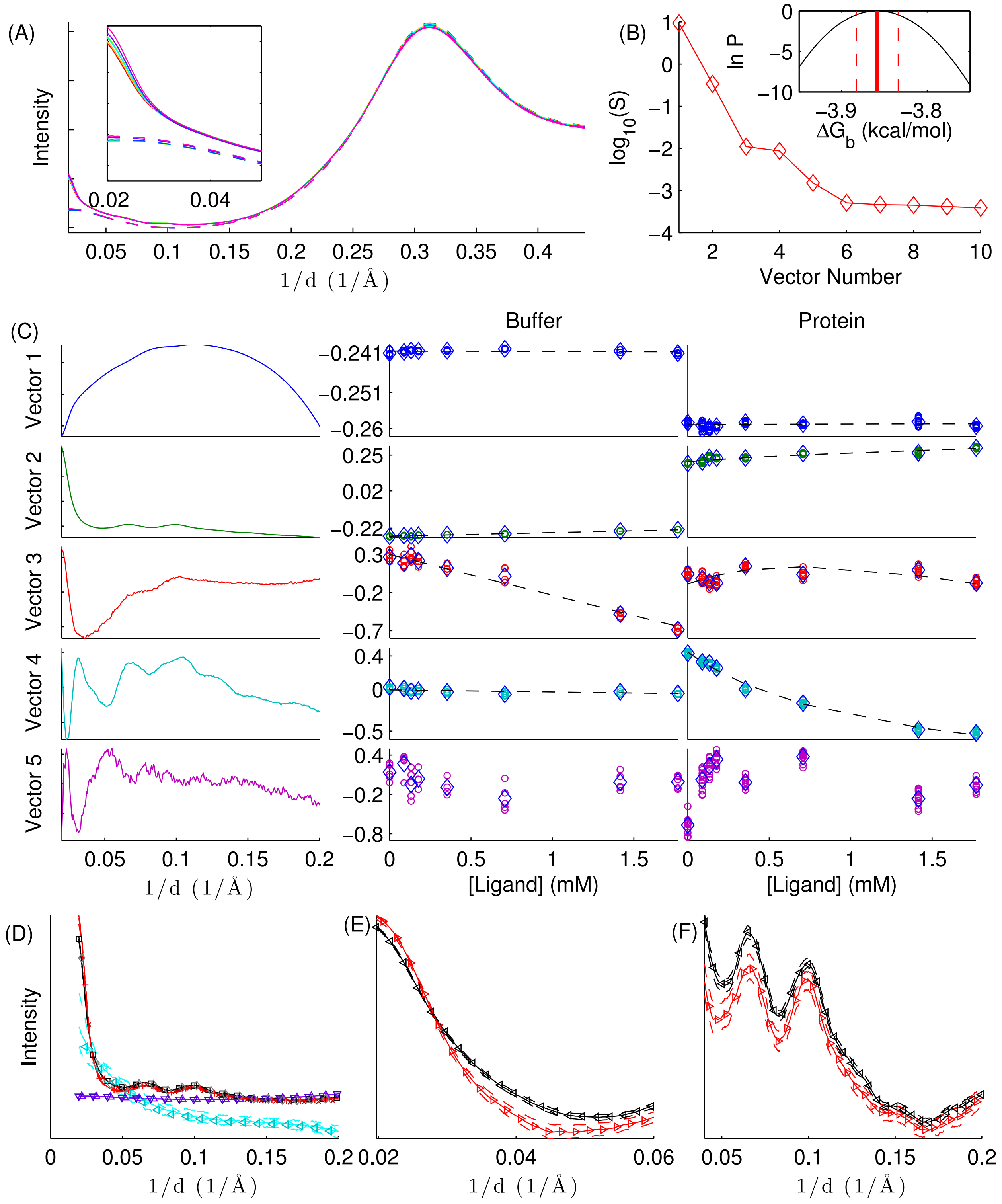

Figure S 3: Analysis of Alcohol Dehydrogenase-NAD$^+$ Binding. The caption is the same as in Fig. 2.

MATLAB code for analysis with SVD

```
function [dGb,dGb_lower,dGb_upper,R,R_lower,R_upper,dGb_space,lnP_dGb_space,CB] = ...
    analyze1site(D_all,Lo,Mo,Ns,RT,hasM,mapping)
% The function "analyze1site" estimates the binding free energy and
% reference patterns, returning,
%
% dGb - the point estimate of the binding free energy
% dGb_lower - the lower bound of the 68% confidence interval for dGb
% dGb_upper - the upper bound of the 68% confidence interval for dGb
% R - the 4 x A matrix of reference patterns
% R_lower - the lower bound of the 68% confidence interval for R
% R_upper - the upper bound of the 68% confidence interval for R
% CB - a M x K matrix with the best fit the model to the data
%
% The required input is,
%
% D_all - the M_all x A data matrix of all M_all measurements
% Lo - a vector with M ligand concentrations (molar)
% Mo - a vector with M macromolecule concentrations (molar)
% Ns - the number of measurements for each of the M conditions
% RT - the gas constant times the temperature
% hasM - a vector of M logicals, true if the condition contains the
%   macromolecule.
% mapping - a vector that contains a unique number for each condition

K = 4; % Number of modeled factors
D = mapped_moments_var(D_all,mapping);

% Do SVD and global fitting
[Uo,So,Vo] = svd(D',0);
Vo_all = (diag(1./diag(So))*Uo'*D_all')';

% Trim matricies
U = Uo(:,1:K);
S = So(1:K,1:K);
V = Vo(:,1:K);

V_all = Vo_all(:,1:K);

% Estimate variances
varV_shrink = var_shrink_fun(V_all,mapping,hasM,Ns);

% In-line function that returns the posterior distribution
lnP_fun = @(dGb) -1/2*log(2*pi*1000)-dGb.^2/2000 ... % Diffuse prior for binding free energy
    + lnL_WLS_fun(C_fun(Mo,Lo,exp(dGb/RT)),V,1./varV_shrink);  % Likelihood

%% Estimate confidence intervals for dGb
```

```
% Calculate posterior over broad region from -15 to 0
dGb_space_full = linspace(-15,0,301);
lnP_dGb_space_full = arrayfun(lnP_fun,dGb_space_full);
inds = find((lnP_dGb_space_full-max(lnP_dGb_space_full))>log(1E-5));
% Over a smaller region with considerable density
dGb_space = dGb_space_full(inds(1)):0.001:dGb_space_full(inds(end));
lnP_dGb_space = arrayfun(lnP_fun,dGb_space);
Post_dGb_space = exp(lnP_dGb_space-max(lnP_dGb_space));
% Estimate cumulative distribution function
cdf = cumsum(Post_dGb_space)/sum(Post_dGb_space);

[~,ind]=min(abs(ones(3,1)*cdf - ...
    [0.16 0.5 0.84]'*ones(1,length(dGb_space))),[],2);
dGb_lower = dGb_space(ind(1));
dGb = dGb_space(ind(2));
dGb_upper = dGb_space(ind(3));
clear ind

%% Estimate confidence intervals for reference patterns
C = C_fun(Mo,Lo,exp(dGb/RT)); % Predicted concentrations
B = ((C'*C)\C')*V;
CB = C*B;

R = (U*S*B')'; % Point estimate of reference patterns

% Estimate confidence intervals for reference patterns
N = 1000;
Rs = zeros(numel(R),N);
dGbs = AR(@(x) exp(lnP_fun(x)-max(lnP_dGb_space)),...
    [dGb_space(1) dGb_space(end)],1,N);
for n = 1:N
    Cn = C_fun(Mo,Lo,exp(dGbs(n)/RT));
    Vn = V + randn(size(varV_shrink)).*sqrt(varV_shrink);
    [~,Bn] = lnL_WLS_fun(Cn,Vn,1./varV_shrink);
    [~,stdR] = lscov(Cn,D); % Estimate error in reference patterns with direct OLS error
    Rn = (U*S*Bn')' + randn(size(stdR)).*stdR;
    Rs(:,n) = reshape(Rn,1,[]);
end
clear n Cn Pn varVoM

% Matrix pts
R_lower = zeros(size(R));
R_upper = zeros(size(R));
ind_lower = 0.16*N;
ind_higher = 0.84*N;
for p = 1:numel(R)
    sorted = sort(Rs(p,:));
    R_lower(p) = sorted(ind_lower);
    R_upper(p) = sorted(ind_higher);
end
clear Rn Rs N

return

% Returns the mean and average over distinct conditions
function [meanX,varX] = mapped_moments_var(X,mapping)
    nconds = max(mapping); % Number of distinct conditions

    % Dimension to average over.
    % If the dimensions are same, the averaging
    % will be over the first.
    dim = find(size(X)==length(mapping),1);

    if dim == 1
        % Average over the first dimension
        ndims = size(X,2);
        meanX = zeros(nconds,ndims);
        varX = zeros(nconds,ndims);
        for n = 1:nconds
            curX = X(mapping==n,:);
            meanX(n,:) = mean(curX);
            varX(n,:) = var(curX);
        end
    elseif dim == 2
        % Average over the second dimension
        ndims = size(X,1);
        meanX = zeros(ndims,nconds);
        varX = zeros(ndims,nconds);
        for n = 1:nconds
            curX = X(:,mapping==n);
            meanX(:,n) = mean(curX,2);
            varX(:,n) = var(curX,0,2);
        end
    end
return

%% Simple 1:1 Binding Model
function C = C_fun(Mo, Lo, Kd)
    ML = ((Lo + Mo + Kd) - sqrt((Lo + Mo + Kd).^2 - 4*Lo.*Mo))/2;
    C = [ones(size(Lo))*1E-3; ...   % Background
          Lo-ML; ...                 % Free ligand
          Mo-ML; ...                 % Free protein
          ML]'; ...                  % Protein-ligand
return

%% Estimate variance of A using a shrinkage approximation
```

```
function varA = var_shrink_fun(A_all,mapping,hasM,Ns)
    M = size(Ns,1);
    K = size(A_all,2);
    dev  = zeros(M,K);
    for n = 1:K
        dev(:,n) = accumarray(mapping',A_all(:,n));
    end
    NsM = Ns*ones(1,K);
    dev2_all = (A_all - dev(mapping,:)./NsM(mapping,:)).^2;
    dev2_hasM = dev2_all(hasM(mapping),:);
    dev2_noM = dev2_all(~hasM(mapping),:);
    varAa_hasM = sum(dev2_hasM)/(size(dev2_hasM,1)-1);
    varAa_noM = sum(dev2_noM)/(size(dev2_noM,1)-1);
    varA = zeros(size(dev));
    varA(hasM,:) = ones(sum(hasM),1)*varAa_hasM;
    varA(~hasM,:) = ones(sum(~hasM),1)*varAa_noM;
    varA = varA./NsM;
return

%% Log likeihood based on weighted least squares regression
function [l,B_WLS,stdB_WLS] = lnL_WLS_fun(C,V,w)
    K = size(V,2);
    B_WLS = zeros(4,K);
    stdB_WLS = zeros(4,K);
    for n = 1:K
        [B_WLS(:,n),stdB_WLS(:,n)] = lscov(C,V(:,n),w(:,n));
    end
    l = -1/2*sum(sum((V-C*B_WLS).^2.*w));
return

%% Acceptance-Rejection
% P is inline function for probability
% interval is maximum and minimum
% max is maximum value of probability
function X = AR(P,interval,max,N)
    X = zeros(1,N);
    n = 1;
    diff = interval(2) - interval(1);
    while n<=N
        xtrial = rand*diff + interval(1);
        if rand<(P(xtrial)/max)
            X(n) = xtrial;
            n = n + 1;
        end
    end
return
```

MATLAB code for analysis without SVD

```
function [dGb,dGb_lower,dGb_upper,R,R_lower,R_upper,dGb_space,lnP_dGb_space] = ...
    analyze1site_noSVD(D_all,Lo,Mo,Ns,RT,hasM,mapping)
% The function "analyze1site" estimates the binding free energy and
% reference patterns, returning,
%
% dGb - the point estimate of the binding free energy
% dGb_lower - the lower bound of the 68% confidence interval for dGb
% dGb_upper - the upper bound of the 68% confidence interval for dGb
% R - the 4 x A matrix of reference patterns
% R_lower - the lower bound of the 68% confidence interval for R
% R_upper - the upper bound of the 68% confidence interval for R
%
% The required input is,
%
% D_all - the M_all x A data matrix of all M_all measurements
% Lo - a vector with M ligand concentrations (molar)
% Mo - a vector with M macromolecule concentrations (molar)
% Ns - the number of measurements for each of the M conditions
% RT - the gas constant times the temperature
% hasM - a vector of M logicals, true if the condition contains the
%   macromolecule.
% mapping - a vector that contains a unique number for each condition

D = mapped_moments_var(D_all,mapping);
varD_shrink = var_shrink_fun(D_all,mapping,hasM,Ns);

% In-line function that returns the posterior distribution
lnP_fun = @(dGb) -1/2*log(2*pi*1000)-dGb.^2/2000 ... % Diffuse prior for binding free energy
    + lnL_WLS_fun(C_fun(Mo,Lo,exp(dGb/RT)),D,1./varD_shrink);  % Likelihood

%% Estimate confidence intervals for dGb
fprintf('Estimating confidence interval for binding free energies\n')
dGb_space_full = linspace(-15,0,301);
lnP_dGb_space_full = arrayfun(lnP_fun,dGb_space_full);
inds = find((lnP_dGb_space_full-max(lnP_dGb_space_full))>log(1E-5));
dGb_space = dGb_space_full(inds(1)):0.001:dGb_space_full(inds(end));
lnP_dGb_space = arrayfun(lnP_fun,dGb_space);
Post_dGb_space = exp(lnP_dGb_space-max(lnP_dGb_space));

cdf = cumsum(Post_dGb_space)/sum(Post_dGb_space);
[~,ind]=min(abs(ones(3,1)*cdf - ...
    [0.16 0.5 0.84]'*ones(1,length(dGb_space))),[],2);
dGb_lower = dGb_space(ind(1));
dGb = dGb_space(ind(2));
dGb_upper = dGb_space(ind(3));
clear ind
```

```
%% Estimate confidence intervals for reference patterns
fprintf('Estimating confidence interval for reference patterns\n')
C = C_fun(Mo,Lo,exp(dGb/RT)); % Predicted concentrations
[~,R,~]=lnL_WLS_fun(C,D,1./varD_shrink); % Point estimate of reference pattern

% Estimate confidence intervals for reference patterns
N = 1000;
Rs = zeros(numel(R),N);
dGbs = AR(@(x) exp(lnP_fun(x)-max(lnP_dGb_space)),...
    [dGb_space(1) dGb_space(end)],1,N);
for n = 1:N
    Cn = C_fun(Mo,Lo,exp(dGbs(n)/RT));
    [~,Rn,stdRn]=lnL_WLS_fun(Cn,D,1./varD_shrink);
    Rn = Rn + randn(size(stdRn)).*stdRn;
%     rand_C(:,n) = reshape(Cn(variableC),1,[]);
    Rs(:,n) = reshape(Rn,1,[]);
end
clear n

% Matrix pts
R_lower = zeros(size(R));
R_upper = zeros(size(R));
ind_lower = 0.16*N;
ind_higher = 0.84*N;
for p = 1:numel(R)
    sorted = sort(Rs(p,:));
    R_lower(p) = sorted(ind_lower);
    R_upper(p) = sorted(ind_higher);
end
clear Rn Rs N

return % End main function
```

The auxiliary functions called from analyze1site_noSVD are the same as from analyze1site.