Supplementary Material - Singular Value Decomposition (SVD) and Ligand Binding Analysis

André Luiz Galo and Márcio Francisco Colombo

MatLab® scripts

a) Input Data format

$$\text{data.dat} = \begin{pmatrix} ND & [DNA]_1 & [DNA]_2 & \cdots & [DNA]_n \\ ND & [ligand\,]_1 & [ligand\,]_2 & \cdots & [ligand\,]_n \\ \lambda_1 & A_{11} & A_{12} & \cdots & A_{1n} \\ \lambda_2 & A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \lambda_m & A_{m1} & A_{m1} & \cdots & A_{mn} \end{pmatrix}$$

were *ND* values are not defined (blank), $[DNA]_n$ e $[ligand]_n$ are DNA and ligand concentration, respectively, for n-th titration.

b) Main script

```
load data.dat
[m,n] = size(data);
DNA = data(1,2:n);
Lt = data(2,2:n);
lb = Data(3:m,1);
abs = Data(3:m,2:n);
plot(lb,abs)                                    % absorbance plot view
pause
x = abs;
[m,n] = size(x);
r = min(m-1,n);
avg = mean(x);
avgx=avg(ones(m,1),:);
absnorm = (x - avgx); clear x
 [U,S,V] = svd(absnorm,0);                % SVD compute
variances=diag(S).^2;
percent_explained = 100*variances/sum(variances);
peso=percent_explained(1:4)               % weight of first four components
```

```
%%   Plots for define the matrix rank   %%
plot(lb,U(:,1:3))                                   % plot the first three U components
plot(log10(concActd),V(:,1:3),'.')                  % plot the first three V components
plot(lb,absnorm*V(:,1:3))
%%   Non-linear fitting  - Input initial parameters   %%
 nc = 2                                        % nc = number of significative principal components
K1 = 5.1 ;  ns1 = 8 ;
K2 = 6.8 ;  ns2 = 60;                          % 10^K1 and 10^K2 are binding constants;
                                               % ns1 e ns2 are number of occupied sites
nsk1=-log10(ns1)+K1;                           % input parameters are   ns1*K1  and ns2*K2
nsk2=-log10(ns2)+K2;                           % This give more stability
par=[K1 nsk1 K2 nsk2];
ajusteScatchard                                % call ajusteScatchard script and return plots
%%   View parameters fitting and Scatchard plot   %%
par_K=[10^par(1) 10^par(3)]                              % display k1 and k2
par_n=[10^(par(1)-par(2)) 10^(par(3)-par(4))]            % display n1 and n2
plot(L_bound./DNA,L_bound./DNA./L_free,'o-')             % display Scatchard plot
```

c) ajustScatchard script

```
par = lsqnonlin(@errVscatchard,par,[],[],[],Lt, DNA, nc, U, S, V, absnorm)
    if max(size(par))==2
         ns1 = 10^par(1)/10^par(2)
    else
         ns1 = 10^par(1)/10^par(2)
         ns2 = 10^par(3)/10^par(4)
    end
pause                                  % pause – return computed value
                                       % after lsqnonlin converging – return F matrix
%%   compute F matrix   %%
m = max(size(Lt)); par
L_free = zeros(size(Lt));
L_f=0.25*Lt(1);                        % Initial condition (titration) – most of binding are bound
                                       % then,  we start with [ligand free] = 0,25*[ligand total]
    for i=1:m
         L_f=fzero(@LfScatchard,L_f,optimset('fzero'),par, DNA(i),Lt(i));
         L_free(i)=L_f;
     end                               % recursively calls LfScatchard
```

```
                                            % and compute the concentrations by Scatchard model
L_bound = calculo_bound(par, DNA, L_free);
    if nc==1
        F=[L_free]; ni=L_bound./DNA; niL=ni./L_free;
    end
    if nc==2
       if max(size(par))==4
           L_bound=L_bound(:,1)+L_bound(:,2);
       end
    F=[L_free L_bound]; ni=L_bound./DNA; niL=ni./L_free;
    end
    if nc==3
        F=[L_free L_bound];
        ni=(Lt-L_free)./DNA;    niL=ni./L_free;
ni1=F(:,2)./DNA;       ni2=F(:,2)./DNA;
    end
%%  Vfit (concentrations), Dfit e absfit  compute   %%
H = V(:,1:nc)'*pinv(F');  err=V(:,1:nc)'-H*F';  fitV=H*F';
plot(log10(Lt),fitV,log10(Lt),V(:,1:nc),'.')                % comparison plot:  V e Vfit versus Lt
xlabel('[Lt]')
ylabel('V')
pause                                                         % pause, press any key to continue
plot(log10(DNA),fitV,log10(DNA),V(:,1:nc),'.')          % comparison plot: V e Vfit versus DNA
xlabel('[DNA]')
ylabel('V')
pause
plot(log10(L_free), L_bound,'-o')                           % ligand free versus ligand bound
ylabel('L_bound')
xlabel('Lt')
pause
Dfit=(avgx+U(:,1:nc)*S(1:nc,1:nc)*V(:,1:nc)')*pinv(F')/10;      % recalculated spectra
plot(lb,Dfit)                                                             % warning: 10 is the optical pathway
xlabel('nm')
ylabel('Abs/mol/cm')
pause
absfit = U(:,1:nc)*S(1:nc,1:nc)*H*F';
plot(lb,absnorm-absfit)                                                  % residual plot
xlabel('nm')
```

```
ylabel('Abs_experimental - Abs_fit')
pause
erro = [norm(absnorm-absfit) norm(err)]                    % errors
```

d) errVscatchard function

```
%%  Errors of  Scatchard parameters   %%
function err = errVscatchard(par, Lt, DNA, nc, U, S, V, absnorm);
m = max(size(Lt)); par;
L_free = zeros(size(Lt));
L_f=Lt(1);
    for i=1:m
        L_f=fzero(@LfScatchard,L_f,optimset('fzero'),par, DNA(i),Lt(i));
        L_free(i)=L_f;
    end
%%   Bound ligand concentration   %%
L_bound = calculo_bound(par, DNA, L_free);            % call calculo_bound
                                                      % compute F = [L_free L_bound] matrix
    if nc==1
        F=[L_free];
    end
    if nc==2
         if max(size(par))==4
               L_bound=L_bound(:,1)+L_bound(:,2);
         end
    F=[L_free L_bound];
    end
    if nc==3
         F=[L_free L_bound(:,1) L_bound(:,2)];
    End

H = V(:,1:nc)'*pinv(F');
err=S(1:nc,1:nc)*(V(:,1:nc)'-H*F');                   % value used by lsqnonlin
```

e) LfScatchard function

```
function y = LfScatchard(x, par, DNA, L)
    if max(size(par))==2                              % one site model
         K1=10^par(1);
```

```
        nsk1=10^par(2);
        y = -K1 * x^2 + (L*K1-1-DNA*nsk1) * x + L;
    else                                                    % two site model
        K1=10^par(1);
        nsk1=10^par(2);
        K2=10^par(3);
        nsk2=10^par(4);
        y = -K1*K2 * x^3 + ((L*K1-1)*K2-K1-DNA*(nsk1*K2+nsk2*K1)) * x^2 + ...
        (L*K2+L*K1-1-DNA*(nsk1+nsk2)) * x + L;
    end
```

f) calculo_bound function

```
function L_bound = calculo_bound(par, DNA, L)
UM=ones(size(L));
    if max(size(par))==2                                    % one site
        K1=10^par(1);
        nsk1=10^par(2);
        L_bound = nsk1*L.*DNA./(UM+K1*L);
    else                                                    % two site
        K1=10^par(1);
        nsk1=10^par(2);
        K2=10^par(3);
        nsk2=10^par(4);
        L_bound = [nsk1*L.*DNA./(UM+K1*L) nsk2*L.*DNA./(UM+K2*L)];
    end
```