Competitive counterion complexation allows the true host:guest binding constants from a single titration by ionic receptors,

M. Pessêgo*, N. Basilio, M. C. Muñiz, L. Garcia-Rio*

Supporting Information

1- Single binding model

The model used is based on the assumption that a 1:1 complex is formed between the SC4 and the BTA, neglecting the effect of sodium counterions.

$$\mathbf{H} + \mathbf{G} \underset{}{\overset{K_{\mathrm{BTA}}}{\rightleftharpoons}} \mathbf{H\text{-}G}$$

Scheme S- 1

The stability of the inclusion complex can be described as an association constant, K_{BTA}

$$K_{BTA} = \frac{[H-G]}{[H][G]}$$

where [H] and [G] represent the equilibrium concentration of free species, SC4 and BTA, respectively, and [H-G] is the concentration of the 1:1 complex.
The mass balance for the total concentrations of SC4 and BTA are given by

$$[H]_0 = [H] + [H-G]$$

$$[G]_0 = [G] + [H-G]$$

The combination of these equations with the binding constant gives a second order equation for the concentration of uncomplexed G.

$$a[G]^2 + b[G] + c = 0$$

Where

$$a = K_{BTA}$$

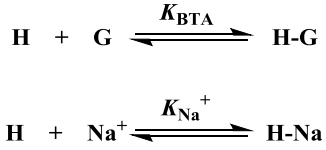
$$b = 1 + K_{BTA}([H]_0 - [G]_0)$$

$$c = -[G]_0$$

This second order equation was solved for different values of K_{BTA} in order to obtain the [G]. The value of K_{BTA} for which we obtain the best root-mean-square deviation values in the fitting of the experimental results, was taken as optimal.

2- Competitive complexation binding model

On the assumption that a competitive complexation occurs between Na^+ and BTA for the SC4 cavity, a second equilibrium should be considered.



Scheme S- 2

The complexation constants of the Na^+ and BTA are expressed as:

$$K_{Na^+} = \frac{[H-Na]}{[H][Na]}$$

$$K_{BTA} = \frac{[H-G]}{[H][G]}$$

where [H],[G] and [Na] represent the equilibrium concentration of free species, SC4, BTA and Na^+.

The mass balance for the total concentrations of SC4, BTA and Na^+ are given by

$$[H]_0 = [H] + [H-G] + [H-Na]$$

$$[G]_0 = [G] + [H-G]$$

$$[Na]_0 = [Na] + [H-Na]$$

The combination of these equations with the binding constants gives a third order equation for the concentration of free H.

$$a[H]^3 + b[H]^2 + c[H] + e = 0$$

Where

$$a = K_{BTA}K_{Na}$$

$$b = K_{BTA} + K_{Na} + K_{BTA}K_{Na}([Na]_0 + [G]_0 - [H]_0)$$

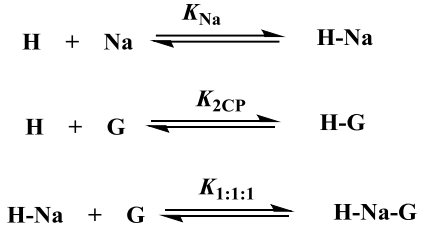
$$c = 1 + K_{Na}([Na]_0 - [H]_0) + K_{BTA}([G]_0 - [H]_0)$$

$$e = -[H]_0$$

This third order equation was solved for different values of K_{BTA}, keeping constant the K_{Na}=183 M^{-1}. The value of K_{BTA} for which we obtain the best root-mean-square deviation values in the fitting of the experimental results, was taken as optimal, in order to obtain the [H].

3- Competitive cooperative binding model

Considering the formation of a ternary complex between SC4, 2CP and Na^+, a third equilibrium should be considered.



Scheme S- 3

The complexation constants of the Na^+, 2CP and 1:1:1 are expressed as:

$$K_{Na^+} = \frac{[H-Na]}{[H][Na]}$$

$$K_{2CP} = \frac{[H-G]}{[H][G]}$$

$$K_{1:1:1} = \frac{[H-Na-G]}{[H-Na][G]}$$

where [H],[G] and [Na] represent the equilibrium concentration of free species, SC4, 2CP and Na^+.

The mass balance for the total concentrations of SC4, 2CP and Na^+ are given by

$$[H]_0 = [H] + [H-G] + [H-Na] + [H-Na-G]$$

$$[G]_0 = [G] + [H-G] + [H-Na-G]$$

$$[Na]_0 = [Na] + [H-Na] + [H-Na-G]$$

These equations where solved using the Newton Raphson method.

4- Titration of TMA by SC4

Sulfonocalixarene titration of TMA was carried out keeping constant the [TMA]=0.985 mM. Figure S- 1 show the complexation induced chemical shift for the TMA hydrogen atoms as well as their fit to the competitive complexation binding model, allowing us to obtain the binding constant, K_{TMA}=5.50x10^5 M^{-1}.

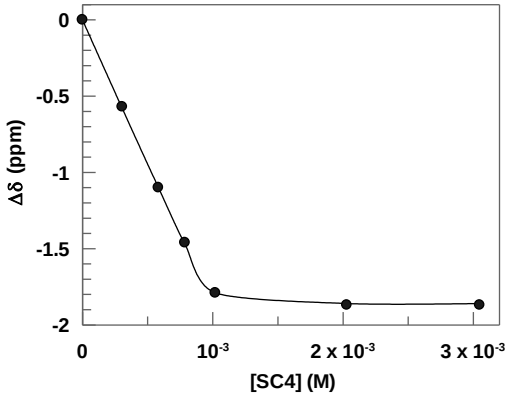
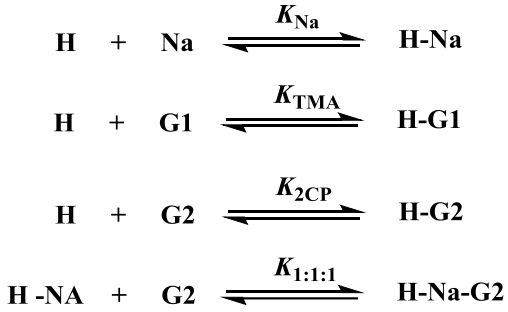


Figure S- 1: Plots of $\Delta\boldsymbol{\delta_{obs}}$ (ppm) for the TMA resonance signal *vs* SC4 concentrations in D_2O at 25ºC. Solid line represents the fitting process to the competitive complexation binding model.

5- NMR displacement assays

Taking into account that adding TMA to the mixture of 2CP and SC4, should be considered the competitive binding process of TMA, Na^+ and 2CP to the SC4, and the formation of a ternary complex (SC4-NA-2CP), allow us to propose this complexation scheme:



Scheme S- 4

The complexation constants of the Na^+, 2CP, 1:1:1 and TMA are expressed as:

$$K_{Na^+} = \frac{[H-Na]}{[H][Na]}$$

$$K_{2CP} = \frac{[H-G2]}{[H][G2]}$$

$$K_{1:1:1} = \frac{[H-Na-G2]}{[H-Na][G2]}$$

$$K_{TMA} = \frac{[H-G1]}{[H][G1]}$$

where [H], [G1], [G2] and [Na] represent the equilibrium concentration of free species, SC4, TMA, 2CP and Na^+, respectively.

The mass balance for the total concentrations of SC4, 2CP, TMA and Na^+ are given by:

$$[H]_0 = [H] + [H-G2] + [H-Na] + [H-Na-G2] + [H-G1]$$

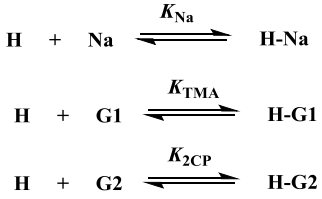
$$[G2]_0 = [G2] + [H-G2] + [H-Na-G2]$$

$$[Na]_0 = [Na] + [H-Na] + [H-Na-G2]$$

$$[G1]_0 = [G1] + [H - G1]$$

These equations where solved using the Newton Raphson method.

If we ignore the formation of a ternary complex (SC4-NA-2CP), the complexation procedure can be expressed as:



Scheme S- 5

The complexation constants of the Na^+, 2CP and TMA are expressed as:

$$K_{Na^+} = \frac{[H - Na]}{[H][Na]}$$

$$K_{2CP} = \frac{[H - G2]}{[H][G2]}$$

$$K_{TMA} = \frac{[H - G1]}{[H][G1]}$$

where [H], [G1], [G2] and [Na] represent equilibrium concentration of free species, SC4, TMA, 2CP and Na^+, respectively.

The mass balance for the total concentrations of SC4, 2CP, TMA and Na^+ are given by:

$$[H]_0 = [H] + [H - G2] + [H - Na] + [H - G1]$$

$$[G2]_0 = [G2] + [H - G2]$$

$$[Na]_0 = [Na] + [H - Na]$$

$$[G1]_0 = [G1] + [H - G1]$$

These equations where solved using the Newton Raphson method.

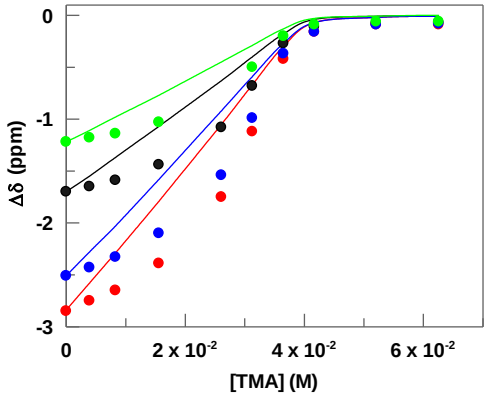


Figure S- 2: Plots of $\Delta\delta_{obs}$ (ppm) for the different resonance signals of 2CP, [2CP]=4mM, vs. TMA concentration in the presence of [SC4]=40mM in D_2O at 25ºC. Solid lines correspond to the binding model showed in scheme S-5.

6-Matlab code for cooperative competitive complexation method

```
clear all
kk=importdata('constantes.txt');
[m,n] = size(kk);
% open the results file
fileID = fopen('resultados.txt','w');
% Newton's iteration starting value
x0 =[2e-4;1e-3;5e-5;1e-3;1e-4;1e-4];
fprintf(fileID,'    y1    ||    y2    ||   y3    ||    x    ||    w    ||    z      \n');
 for s = 1:m
   k=kk(s,:);
   k=k';
   %  algorithm constants
   tol=1.e-10; maxiter =100;
   %  Newton's iteration method
  [x,F,iter] = newtonsys(@Ffun ,@Jfun ,x0,k ,tol ,maxiter);
  fprintf(fileID,'%e||%e||%e||%e||%e||%e \n',x);
  fprintf('The constants are %e\n',k);
  fprintf('The solution is \n');
  fprintf(' y1=%e\n',x(1));
  fprintf(' y2=%e\n',x(2));
  fprintf(' y3=%e\n',x(3));
  fprintf('  x=%e\n',x(4));
  fprintf('  w=%e\n',x(5));
  fprintf('  z=%e\n',x(6));
  fprintf('The residual:\n');
  fprintf(' %e %e %e %e %e %e\n',feval(@Ffun,x,k));
  fprintf('_____ \n');
end
% close the results file
fclose(fileID);
%
%  Newton's  function
%
function [x,F,iter ] = newtonsys(Ffun,Jfun,x0,k,tol,nmax,varargin)
%NEWTONSYS tries to get a zero of a nonlinear system.
% [ZERO ,F,ITER ]= NEWTONSYS(FFUN ,JFUN ,X0 ,TOL ,NMAX)
% The nonlinear system  is defined
% in function FFUN with Jacobian matrix defined in JFUN.
% X0 is  the Newton's iteration starting value
iter = 0; err = tol + 1; x = x0;
while err > tol & iter <= nmax
J = feval(Jfun ,x,k,varargin {:});
F = feval(Ffun ,x,k,varargin {:});
delta = - J\F;
x = x + delta;
err = norm(delta );
iter = iter + 1;
end
F = norm(feval(Ffun ,x,k,varargin {:}));
if iter >= nmax
fprintf(' The algorithm exceeds the maximum number of iterations \n ');
fprintf(' The residual is %e\n',F);
else
fprintf(' The method converges at iteration %i with a residual %e\n',iter ,F);
end
return
%
```

```
% Function Ffun which defines the nonlinear system
%
function F=Ffun(x,k)
F(1 ,1) = k(1)*x(5)*x(4)-x(2);
F(2 ,1) = k(2)*x(5)*x(6)-x(1);
F(3 ,1) = k(1)*k(3)*x(5)*x(4)*x(6)-x(3);
F(4 ,1) = x(4)+x(2)+x(3)-k(6);
F(5 ,1) = x(6)+x(1)+x(3)-k(4);
F(6 ,1) = x(5)+x(1)+x(3)+x(2)-k(5);
return
%
% Function Jfun which defines the Jacobian matrix
%
function J=Jfun(x,k)
J(1 ,1) = 0; J(1 ,2) = -1; J(1 ,3) = 0; J(1 ,4) = k(1)*x(5);...
J(1 ,5) = k(1)*x(4); J(1 ,6) = 0;...
J(2 ,1) = -1; J(2 ,2) = 0; J(2 ,3) = 0; J(2 ,4) = 0;...
J(2 ,5) = k(2)*x(6);  J(2 ,6) = k(2)*x(5);...
J(3 ,1) = 0; J(3 ,2) = 0; J(3 ,3) = -1; J(3 ,4) = k(1)*k(3)*x(5)*x(6);...
J(3 ,5) = k(1)*k(3)*x(4)*x(6);  J(3 ,6) = k(1)*k(3)*x(5)*x(4);...
J(4 ,1) = 0; J(4 ,2) = 1; J(4 ,3) = 1; J(4 ,4) = 1;...
J(4 ,5) = 0;  J(4 ,6) = 0;...
J(5 ,1) = 1; J(5 ,2) = 0; J(5 ,3) = 1; J(5 ,4) =0;...
J(5 ,5) = 0;  J(5 ,6) = 1;...
J(6 ,1) = 1; J(6 ,2) = 1; J(6 ,3) = 1; J(6 ,4) =0;...
J(6 ,5) = 1;  J(6 ,6) = 0;...
 return
```

7-Matlab code for displacement assay of competitive complexation method with cooperative competitive complexation model.

```
clear all
 kk=importdata('constantes.txt');
 [m,n] = size(kk);
 %  open the results file
 fileID = fopen('resultados.txt','w');
 % Newton's iteration starting value
 x0 =[3e-4;3e-2;5e-5;1e-4;2e-1;2e-5;3e-3;4e-3];
 fprintf(fileID,'    y1    ||    y2    ||  y3    ||    y4    ||    a    ||    b    ||    c    ||    d       \n');
 for s = 1:m
    k=kk(s,:);
    k=k';
   %  algorithm constants
    tol=1.e-10; maxiter =100;
    %  Newton's iteration method
    [x,F,iter] = newtonsys(@Ffun ,@Jfun ,x0,k ,tol ,maxiter);
    fprintf(fileID,'%e||%e||%e||%e||%e||%e||%e||%e \n',x);
    fprintf('Las constantes son %e\n',k);
   fprintf('La solucion es \n');
   fprintf(' y1=%e\n',x(1));
   fprintf(' y2=%e\n',x(2));
   fprintf(' y3=%e\n',x(3));
   fprintf(' y4=%e\n',x(4));
   fprintf('  a=%e\n',x(5));
   fprintf('  b=%e\n',x(6));
   fprintf('  c=%e\n',x(7));
   fprintf('  d=%e\n',x(8));
   fprintf('The residual:\n');
   fprintf(' %e %e %e %e %e %e %e %e\n',feval(@Ffun,x,k));
   fprintf('_____ \n');

end

% close the results file
fclose(fileID);
%
%  Newton's  function
%
function [x,F,iter ] = newtonsys(Ffun,Jfun,x0,k,tol,nmax,varargin)
%NEWTONSYS tries to get a zero of a nonlinear system.
% [ZERO ,F,ITER ]= NEWTONSYS(FFUN ,JFUN ,X0 ,TOL ,NMAX)
% The nonlinear system  is defined
% in function FFUN with Jacobian matrix defined in JFUN.
% X0 is  the Newton's iteration starting value
iter = 0; err = tol + 1; x = x0;
while err > tol & iter <= nmax
J = feval(Jfun ,x,k,varargin {:});
F = feval(Ffun ,x,k,varargin {:});
delta = - J\F;
x = x + delta;
err = norm(delta );
iter = iter + 1;
end
F = norm(feval(Ffun ,x,k,varargin {:}));
if iter >= nmax
fprintf(' The algorithm exceeds the maximum number of iterations \n ');
```

```
fprintf(' The residual is %e\n',F);
else
fprintf(' The method converges at iteration %i with a residual  %e\n',iter ,F);
end
return

%
% Function Ffun which defines the nonlinear system
%
function F=Ffun(x,k)
F(1 ,1) = k(1)*x(6)*x(5)-x(1);
F(2 ,1) = k(2)*x(6)*x(7)-x(2);
F(3 ,1) = k(3)*x(6)*x(8)-x(3);
F(4 ,1) = k(1)*k(4)*x(6)*x(8)*x(5)-x(4);
F(5 ,1) = x(5)+x(1)+x(4)-k(5);
F(6 ,1) = x(6)+x(2)+x(3)+x(4)+x(1)-k(6);
F(7 ,1) = x(7)+x(2)-k(7);
F(8 ,1) = x(8)+x(3)+x(4)-k(8);
return


%
% Function Jfun which defines the Jacobian matrix
%
function J=Jfun(x,k)
J(1 ,1) = -1; J(1 ,2) = 0; J(1 ,3) = 0; J(1 ,4) = 0; J(1 ,5) = k(1)*x(6);...
J(1 ,6) = k(1)*x(5); J(1 ,7) = 0; J(1 ,8) = 0;...
J(2 ,1) = 0; J(2 ,2) = -1; J(2 ,3) = 0; J(2 ,4) = 0; J(2 ,5) = 0;...
J(2 ,6) = k(2)*x(7);  J(2 ,7) = k(2)*x(6); J(2 ,8) = 0;...
J(3 ,1) = 0; J(3 ,2) = 0; J(3 ,3) = -1; J(3 ,4) = 0; J(3 ,5) = 0;...
J(3 ,6) = k(3)*x(8); J(3 ,7) = 0; J(3 ,8) = k(3)*x(6);...
J(4 ,1) = 0; J(4 ,2) = 0; J(4 ,3) = 0; J(4 ,4) = -1; J(4 ,5) = k(1)*k(4)*x(6)*x(8);...
J(4 ,6) = k(1)*k(4)*x(8)*x(5);  J(4 ,7) = 0; J(4 ,8) = k(1)*k(4)*x(6)*x(5);...
J(5 ,1) = 1; J(5 ,2) = 0; J(5 ,3) = 0; J(5 ,4) =0;...
J(5 ,5) = 1;  J(5 ,6) = 0; J(5 ,7) = 0; J(5 ,8) = 0;...
J(6 ,1) = 1; J(6 ,2) = 1; J(6 ,3) = 1; J(6 ,4) =1;...
J(6 ,5) = 0;  J(6 ,6) = 1; J(6 ,7) = 0; J(6 ,8) = 0;...
J(7 ,1) = 0; J(7 ,2) = 1; J(7 ,3) = 0; J(7 ,4) =0;...
J(7 ,5) = 0;  J(7 ,6) = 0; J(7 ,7) = 1; J(7 ,8) = 0;...
J(8 ,1) = 0; J(8 ,2) = 0; J(8 ,3) = 1; J(8 ,4) = 1;...
J(8 ,5) = 0;  J(8 ,6) = 1; J(8 ,7) = 0; J(8 ,8) = 1;...
return
```