# **Simulation and Analysis of Binding Reactions.**

Let's consider a simple example of a reversible binding reaction:

*k1*

L + R ⇔ LR

*k-1*

with *k1* (or *kon*) and *k-1* (or *koff*) being the forward and reverse rate constants. The differential equations are:

d[L]/dt = -k1\*[L]\*[R] + k-1\*[LR]

d[R]/dt = -k1\*[L]\*[R] + k-1\*[LR]

d[LR]/dt = k1\*[L]\*[R] - k-1\*[LR]

At **equilibrium**, we define the dissociation constant as:

Kd = k-1/k1 = [L]\*[R]/[LR]

[LR] = [L]\*[R]/Kd (1)

Furthermore, bound ligand [LR] and free ligand [L] concentrations must add up to total ligand concentration [L0], and bound [LR] and free [R] receptor must add up to the total receptor concentration [R0]:

[R0] = [R] + [LR] = [R] + [L]\*[R]/Kd (2)

[L0] = [L] + [LR] = [L] + [L]\*[R]/Kd

These are two equations with 2 unknowns, [L] and [R], which allow the calculation of both [L] and [R] at equilibrium. Computing [R] from (1) as:

[R] = [LR]\*Kd/[L]

and inserting it into (2) leads to:

[R0] = [LR]\*Kd/[L] + [L]\*([LR]\*Kd/[L])/Kd = [LR]\*Kd/[L] + [LR]

[R0] = [LR]\*(1+Kd/[L]) = [LR]\*([L]+Kd)/[L]

[LR] = [R0]\*[L]/([L]+Kd) (3)

Equation (3) is the *hyperbolic binding function* that allows the determination of the concentration of the complex at equilibrium provided the *Kd*, the total concentration of receptor [R0], and the concentration of the free ligand [L] are known. We will use often this relationship.

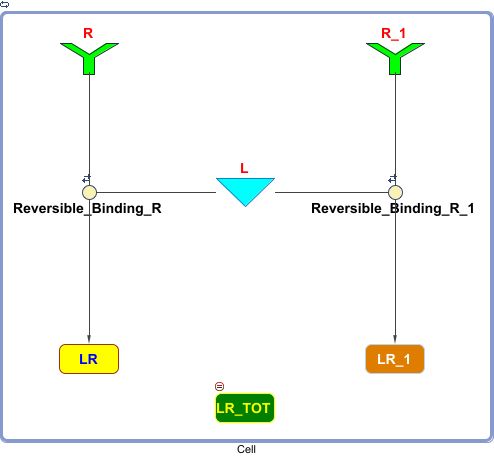
Usually, binding curves are measured with receptor concentrations around or less than the *Kd* value. The actual concentration of the receptor is not relevant for the shape of the binding curve when bound ligand is plotted against the 'free' ligand concentration. However, when bound ligand is plotted against the 'total' ligand concentration, the curve may differ, depending on the receptor concentration and affinity. For this reason, throughout our example we will plot only 'bound' versus 'free'.

In the previous chapter we have learned how to simulate 2nd order reactions like the simple binding of a ligand to a receptor.Our goal here is to use our simulation tools to analyze experimental binding data. In most cases we will have only *equilibrium* binding data, and very seldom *kinetic* binding data. However, we have learned how to obtain equilibrium values from the simulation of the entire time course of binding reactions.

In the following we will use the MATLAB Toolbox 'SymBiology', which incorporates the tools used in the previous chapter for the simulation of chemical reactions, to analyze some common cases of binding reactions. The Toolbox has a powerful GUI for the graphic generation of both chemical and enzymatic reactions, and complex metabolic pathways. Reactions are stored as *'projects'* containing all the parameters informations in either the native MATLAB format as *'.sbproj'* files or in *xml* format.

**1. One binding site in a receptor.**

The simulation and analysis of this binding reaction can be carried out by executing the cells of the MATLAB scripts: TOOLBOXES/BINDING\_KINETICS/Simple\_Binding.m



**2. Two independent binding sites in a single receptor.**

We will start by loading a project containing the model of the binding of a ligand to 2 different sites (with different *Kd*'s) in the same receptor. The best way to simulate two independent binding sites in a single receptor is to equate the system to two receptors each with one binding site. Then, the total concentration of the receptor is given by the sum of the two receptors. This is shown in the following *SimBiology* project:

addpath(genpath('../TOOLBOXES/BINDING\_KINETICS'));

sbioloadproject('../TOOLBOXES/BINDING\_KINETICS/Two\_Independent\_Binding\_Sites');

We can also load the model interactively by starting the SimBiology desktop and opening the '.sbproj' file, or by importing the data from the xml file:

m1 = ...

sbmlimport('../TOOLBOXES/BINDING\_KINETICS/Two\_Independent\_Binding\_Sites.xml');

The project contains a model, **m1**, of the binding of a ligand to two receptors RandR1with different *Kd*'s.

sbioselect(m1,'Type','compartment')

sbioselect(m1,'Type','species')

sbioselect(m1,'Type','parameter')

sbioselect(m1,'Type','reaction')

getequations(m1)

We extract the parameters from the model and save them as variables in the workspace.

C = sbioselect(m1,'Name','Cell');

L = sbioselect(m1,'Name','L');

R = sbioselect(m1,'Name','R');

R\_1 = sbioselect(m1,'Name','R\_1');

LR = sbioselect(m1,'Name','LR');

LR\_1 = sbioselect(m1,'Name','LR\_1');

LR\_TOT = sbioselect(m1,'Name','LR\_TOT');

kon = sbioselect(m1,'Name','kon');

koff = sbioselect(m1,'Name','koff');

kon\_R\_1 = sbioselect(m1,'Name','kon\_R\_1');

koff\_R\_1 = sbioselect(m1,'Name','koff\_R\_1');

Before we start the simulation we set all the necessary parameters and variable values.

C.Capacity = 1;

R.InitialAmountUnits = 'micromole/liter';

R\_1.InitialAmountUnits = 'micromole/liter';

L.InitialAmountUnits = 'micromole/liter';

LR.InitialAmountUnits = 'micromole/liter';

LR\_1.InitialAmountUnits = 'micromole/liter';

First we set a very high concentration of the ligand to observe the complete saturation of the receptor.

L.InitialAmount = 6000;

R.InitialAmount = 0.5;

R\_1.InitialAmount = 0.5;

LR.InitialAmount = 0;

LR\_1.InitialAmount = 0;

LR\_TOT.InitialAmount = 0;

The following is the default rule: both LR and LR\_1 absorb the same in the spectrophotometer.

set(m1.rule,'Rule','LR\_TOT = LR + LR\_1');

What happens if we change the rule? For example, LR\_1 could absorb less than LR.

% set(m1.rule,'Rule','LR\_TOT = 0.5\*LR + 1\*LR\_1');

Here we store the initial parameters.

C\_Init = 1;

R\_Init = 0.5;

R\_1\_Init = 0.5;

LR\_Init = 0;

LR\_1\_Init = 0;

LR\_TOT\_Init = 0;

kon.Value = 10;

koff.Value = 100; % Kd(R) = 10 µM

kon\_R\_1.Value = 10;

koff\_R\_1.Value = 3000; % Kd(R\_1) = 300 µM

We also get some information on the configuration parameters of the simulation, and we set the initial stop time. For example, we will use 2 ms as the stop time and will set the solver to ODE23t (trapezoidal solver, moderatively stiff) or ODE15s (RK solver, stiff).

configset\_m1 = getconfigset(m1);

configset\_m1

Stop = 0.0002;

set(configset\_m1, 'StopTime', Stop);

set(configset\_m1.SolverOptions, 'AbsoluteTolerance', 1.e-9);

set(configset\_m1, 'SolverType', 'ode15s');

## Now we can run a small simulation with the existing parameters. Only 2 milliseconds (msec) are necessary for the system to reach the equilibrium (steady-state). We recall here that since we are simulating the reaction from the ideal time in which there is an instantaneous mixing of ligand and receptor (time 0) until the time in which the equilibrium is reached (steady-state), the result is a reaction curve that mimics an experiment of pre-steady-state kinetics.

Binding\_Kinetics = sbiosimulate(m1);

We can get some information about the simulation. If we want to see the actual numbers, the time points are in the array Binding\_Kinetics.Time and the time-course of the simulation is in the Binding\_Kinetics.Data. The names of the species in the reaction are in Binding\_Kinetics.DataNames.

get(Binding\_Kinetics)

openvar Binding\_Kinetics.Time

openvar Binding\_Kinetics.Data

openvar Binding\_Kinetics.DataNames

Species are logged in the following order: L R LR LR\_1 LR\_TOT R\_1. We can obtain a fast plot of all the species in the simulation using the command:

sbioplot(Binding\_Kinetics)

but it makes more sense to plot only some of the species:

Two\_binding\_sites\_1 = figure;

set(Two\_binding\_sites\_1,'Units','normalized','Position',[0.4 0.0 0.6 1.0],...

'Name','Binding Curve ');

subplot1 = subplot(2,1,1,'Parent',figure(gcf));

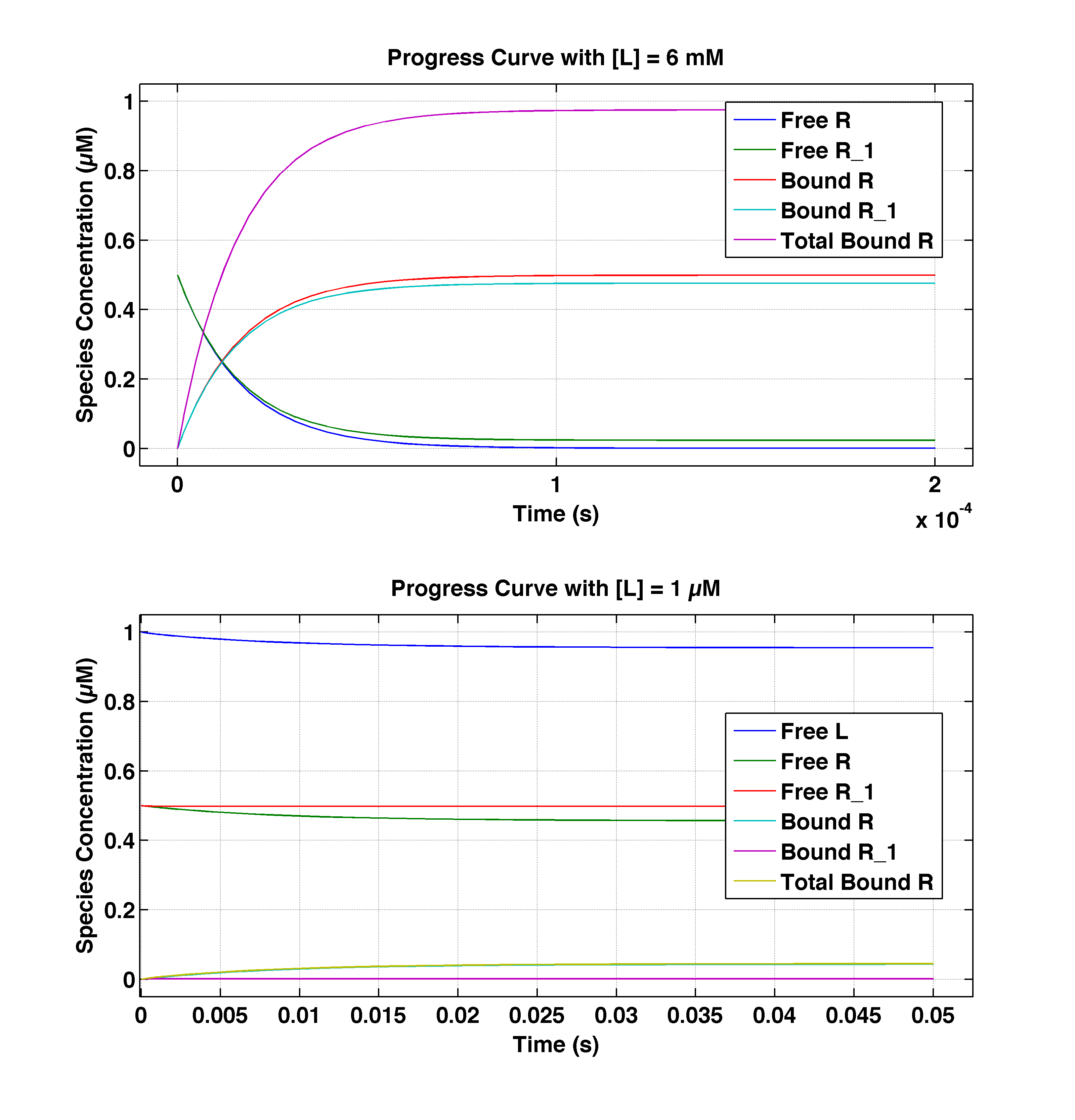
box(subplot1,'on');

grid(subplot1,'on');

hold(subplot1,'all');

plot(Binding\_Kinetics.Time,Binding\_Kinetics.Data(:,[2 6 3:5] ))

legend('Free R','Free R\\_1','Bound R','Bound R\\_1','Total Bound R','Location','Best')

set(gca,'YLim',[-0.05 1.05]);set(gca,'XLim',[-0.00001 1.05\*Stop]);

ylabel('Species Concentration (µM)');xlabel('Time (s)')

title('Progress Curve with [L] = 6 mM');

At this concentration R\_1 is only slightly less saturated than R. However, it is even more important to know what is the saturation level and how long it takes for the reaction to reach equilibrium at very low concentration of the ligand.

C.Capacity = 1;

L.InitialAmount = 1;

R.InitialAmount = 0.5;

R\_1.InitialAmount = 0.5;

LR.InitialAmount = 0;

LR\_1.InitialAmount = 0;

LR\_TOT.InitialAmount = 0;

Stop = 0.05;

set(configset\_m1, 'StopTime', Stop);

Binding\_Kinetics = sbiosimulate(m1);

subplot2 = subplot(2,1,2,'Parent',figure(gcf));

box(subplot2,'on');

grid(subplot2,'on');

hold(subplot2,'all');

plot(Binding\_Kinetics.Time,Binding\_Kinetics.Data(:,[1:2 6 3:5]))

legend('Free L','Free R','Free R\\_1','Bound R','Bound R\\_1','Total Bound R','Location','Best')

set(gca,'YLim',[-0.05 1.05]);set(gca,'XLim',[-1E-4 1.05\*Stop]);

ylabel('Species Concentration (µM)');xlabel('Time (s)')

title('Progress Curve with [L] = 1 µM');

It is quite apparent that at low concentration of ligand only the high-affinity site (receptor R) is saturated.

Now we are ready to simulate an entire equilibrium binding experiment. We recall that Kd(R) = 10 µM and Kd(R\_1) = 300 µM. In order to examine the effect of different concentrations of the ligand it is convenient to write a 'loop'. First we define a vector with all the concentrations we want to explore spaced logarithmically. It is a good idea to cover a range from 1/10 to 20-30 times the Kd value(s) in order to completely saturate the receptor(s). Thus, we will cover the range of concentrations 1-6000 µM. We also extend the simulation time to 0.5 seconds in order to be absolutely certain that under all concentrations of the ligand we have reached equilibrium.

Stop = 0.5;

set(configset\_m1, 'StopTime', Stop);

Finally, to make the simulation more realistic we can add a normally distributed random error to the volume of the binding assay and to the concentrations of receptor and ligand:

High\_Conc = 6000;

conc\_vec = logspace(log10(1),log10(High\_Conc),35);

range = conc\_vec <= (1 + High\_Conc);

Bound = zeros(length(conc\_vec),1);

Bound\_R = zeros(length(conc\_vec),1);

Bound\_R\_1 = zeros(length(conc\_vec),1);

Unbound = zeros(length(conc\_vec),1);

time = Binding\_Kinetics.Time;

for i = 1:length(conc\_vec)

error = (rand(3,1)-0.5);

error(1) = 1 + error(1)\*0.15;

error(2) = 1 + error(2)\*0.15;

error(3) = 1 + error(3)\*0.15;

set(C,'Capacity',C\_Init\*error(1));

set(R,'InitialAmount',R\_Init\*error(2)/C.Capacity);

set(R\_1,'InitialAmount',R\_1\_Init\*error(2)/C.Capacity);

set(L,'InitialAmount',conc\_vec(i)\*error(3)/C.Capacity);

set(LR,'InitialAmount',0);

set(LR\_1,'InitialAmount',0);

set(LR\_TOT,'InitialAmount',0);

Binding\_Kinetics = sbiosimulate(m1);

Bound(i) = Binding\_Kinetics.Data(end,5);

Bound\_R(i) = Binding\_Kinetics.Data(end,3);

Bound\_R\_1(i) = Binding\_Kinetics.Data(end,4);

Unbound(i) = Binding\_Kinetics.Data(end,1);

end

Assuming these synthetic data are a realistic representation of a real *equilibrium binding experiment*,we can now derive the binding parameters from them using traditional tool, and see if they correspond well to the parameters used to generate the data. First we plot the binding curve: each point is the value of the bound ligand at steady-state in a simulation in which the ligand concentration is changed to progressively higher values.

Two\_binding\_sites\_2 = figure;

set(Two\_binding\_sites\_2,'Units','normalized','Position',[0.4 0.0 0.6 1.0],...

'Name','Binding Curve Fit');clf

subplot1 = subplot(3,1,1,'Parent',figure(gcf));

box(subplot1,'on');grid(subplot1,'on');hold(subplot1,'all');

plot(Unbound(range),Bound(range),'--or',...

'MarkerEdgeColor','k',...

'MarkerFaceColor','y',...

'MarkerSize',5);

plot(Unbound(range),Bound\_R(range),':xg',...

'MarkerEdgeColor','k',...

'MarkerFaceColor','y',...

'MarkerSize',5);

plot(Unbound(range),Bound\_R\_1(range),':+b',...

'MarkerEdgeColor','k',...

'MarkerFaceColor','y',...

'MarkerSize',5);

set(gca,'YLim',[-0.05 1.15]);set(gca,'XLim',[-100 Unbound(end)+100]);

legend('Bound Total','Bound R','Bound R\\_1','Location','Best');

xlabel('[Free Ligand]');ylabel('[Bound Ligand]');title('Binding Curve ');

We can determine the binding parameters with a *Scatchard plot* of the ratio of [bound]/[unbound (free)] *vs* the [bound] ligand concentrations. In this case we have:

Y = [Bound]/[Free]

X = [Bound]

Slope = -1/Kd.

X axis intercept = [R0] (total receptor conc.)

First, we fit the data corresponding to the high affinity receptor by linear regression (least square fit) to the equation of a straight line. We recall here that since slope = -1/Kd, if the Kd is small (high affinity) the slope is going to be negative and steeper: this is the left-hand side of the Scatchard plot. We are going to fit a straight line through the first 9 points of the Scatchard. Then, separately we fit the data corresponding to the low affinity receptor (right hand side of the plot).

Total Range of concentrations that will be used in the Scatchard plot.

s\_range = range;

s\_range\_1 = true(1,35);s\_range\_1(10:end) = false;

s\_range\_2 = true(1,35);s\_range\_2(1:22) = false;

subplot2 = subplot(3,1,2,'Parent',figure(gcf));

box(subplot2,'on');grid(subplot2,'on');hold(subplot2,'all');

X = Bound(s\_range);

Y = Bound(s\_range)./Unbound(s\_range);

plot(X,Y,'s','MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSize',5);

Ranges of concentrations that will be used in the fit of the Scatchard plot based on a visual inspection of the plot.

s\_range\_1 = true(1,35);s\_range\_1(19:end) = false;

s\_range\_2 = true(1,35);s\_range\_2(1:22) = false;

High affinity range

X1 = Bound(s\_range\_1);

Y1 = Bound(s\_range\_1)./Unbound(s\_range\_1);

f = fittype('a\*x + b');

[Scatchard\_1,GOF\_1] = fit(X1,Y1,f,'StartPoint',[-0.01 0.05]);

plot(Scatchard\_1,'-b');

ylim([0,Scatchard\_1(0)]);

xlim([0,1.1]);

Scatchard\_1

GOF\_1

Scatchard\_1\_params = coeffvalues(Scatchard\_1);

Kd\_1 = -1/Scatchard\_1\_params(1);

Receptor\_conc\_1 = -Scatchard\_1\_params(2)/Scatchard\_1\_params(1);

Low affinity range

X2 = Bound(s\_range\_2);

Y2 = Bound(s\_range\_2)./Unbound(s\_range\_2);

f = fittype('a\*x + b');

[Scatchard\_2,GOF\_2] = fit(X2,Y2,f,'StartPoint',[-0.03 0.03]);

plot(Scatchard\_2,'-r');

Scatchard\_2

GOF\_2

Scatchard\_2\_params = coeffvalues(Scatchard\_2);

Kd\_2 = -1/Scatchard\_2\_params(1);

Receptor\_conc\_2 = -Scatchard\_2\_params(2)/Scatchard\_2\_params(1);

This result is likely to be incorrect for the low affinity site because of the contribution from the high affinity site.

legend('Scatchard','Fit\\_1','Fit\\_2');

string0 = ' µM';

string1 = 'Kd\\_1 = ';

string2 = num2str(Kd\_1,'%6.2e\n');

string3 = 'Kd\\_2 = ';

string4 = num2str(Kd\_2,'%6.2e\n');

annotation(Two\_binding\_sites\_2,'textbox',...

[0.58 0.48 0.12 0.09],...

'String',{[string1 string2 string0; string3 string4 string0;]},...

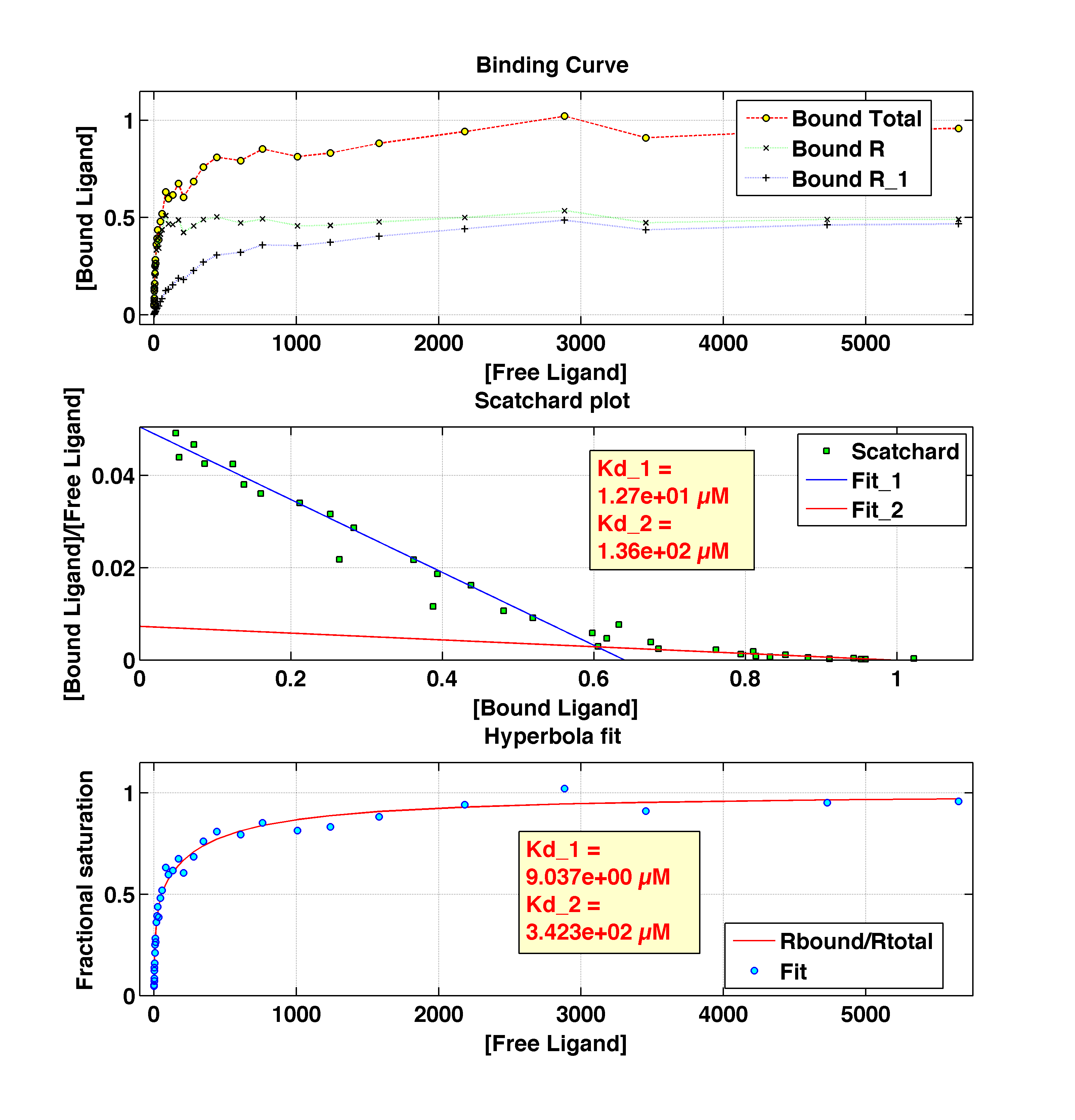
'FontWeight','bold',...

'FitBoxToText','off',...

'BackgroundColor',[1 1 0.8],...

'Color',[1 0 0]);

xlabel('[Bound Ligand]');ylabel('[Bound Ligand]/[Free Ligand]');title('Scatchard plot ');

Alternatively, we can fit directly the plot of the fractional saturation of the receptor (FR) *vs* the amount of free ligand (L) with the *hyperbolic function* we presented at the beginning of the example, rearranged in terms of fractional saturation:

[LR] = [R0]\*[L]/([L]+Kd)

[LR]/[R0] = [L]/([L]+Kd)

Two hyperbolas are fitted each contributing 50% to the curve. For a single hyperbola it would be: f = fittype('x/(a + x)'); For two hyperbolas we use a two-component fit with 2 unknowns. 'a' and 'b' are the two Kd's. We will use the values of Kd\_1 and Kd\_2 found by the Scatchard plot as the starting points for the fits. We can also include some lower and upper bounds for the fitting parameters.

FR = Bound/(R\_Init + R\_1\_Init);X = Unbound(s\_range);Y = FR(s\_range);

subplot3 = subplot(3,1,3,'Parent',figure(gcf));

box(subplot3,'on');grid(subplot3,'on');hold(subplot3,'all');

f = fittype('(0.5\*x/(a + x)) + (0.5\*x/(b + x))');

% [Hyperb,GOF] = fit(X,Y,f,'StartPoint',[Kd\_1 Kd\_2],...

% 'Lower',[0.3\*Kd\_1 0.3\*Kd\_2],'Upper',[1.7\*Kd\_1 1.7\*Kd\_2]);

[Hyperb,GOF] = fit(X,Y,f,'StartPoint',[Kd\_1 Kd\_2]);

plot(X,Hyperb(X),'-r');

plot(X,Y,'o','MarkerEdgeColor','b','MarkerFaceColor','c','MarkerSize',5);

ylim([0,1.15]); xlim([-100,Unbound(end)+100]);

legend('Rbound/Rtotal','Fit','Location','Best');

xlabel('[Free Ligand]');

ylabel('Fractional saturation');

Hyperb

GOF

Hyperb\_params = coeffvalues(Hyperb);

Kd\_1 = Hyperb\_params(1);

Kd\_2 = Hyperb\_params(2);

string0 = ' µM';

string1 = 'Kd\\_1 = ';

string2 = num2str(Kd\_1,'%6.3e\n');

string3 = 'Kd\\_2 = ';

string4 = num2str(Kd\_2,'%6.3e\n');

annotation(Two\_binding\_sites\_2,'textbox',...

[0.54 0.18 0.12 0.08],...

'String',{[string1 string2 string0 ; string3 string4 string0]},...

'FontWeight','bold',...

'FitBoxToText','off',...

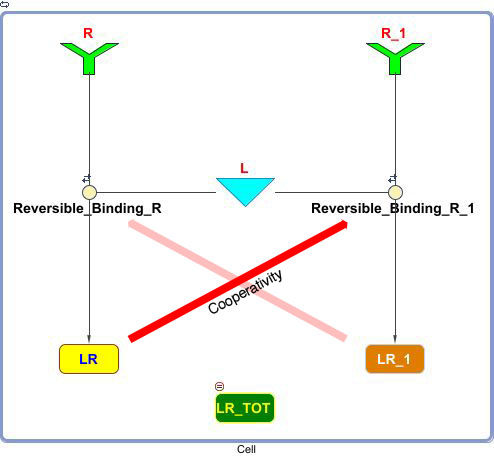
'BackgroundColor',[1 1 0.8],...

'Color',[1 0 0]);

title('Hyperbola fit ');

In conclusion: the presence of multiple binding sites can be easily inferred from the presence of a biphasic Scatchard plot. However, low affinity sites are not determined well by Scatchard. In this case, the best determination of the *Kd*'s of the individual binding sites is obtained by fitting a 2 (or more) components hyperbola to the fractional saturation plot.

**2. Two cooperative binding sites in a single receptor.**



We may wonder if we can still find the binding parameters if the equilibrium binding curve cannot be explained with a linear combination of two hyperbolas. The answer is that if we have a model for the interaction between the two sites we can fit directly the binding progress curves (instead of theoretical hyperbolic curves) to the experimental data. The following example shows how to analyze a binding experiment in which a ligand binds to two cooperative sites on a receptor. Notice that in this case there is cross-cooperativity between the two sites and the equilibrium binding curve cannot be simulated with hyperbolas.

The best way to simulate two cooperative binding sites in a single receptor is to equate the system to two receptors each with one binding site. Then, the *Kd* of one site is represented as a function of the *Kd* in the other site. The total concentration of the receptor is given by the sum of the two receptors.

close all, clear, clc

addpath(genpath('../GENERAL\_SCRIPTS\_FUNCTIONS'));

addpath(genpath('../TOOLBOXES/BINDING\_KINETICS'));

In order to do a direct fit of the progress curves we will need to bring information about the curves in and out of some internal functions. For this reason we will represent the variable involved in the process as *global variable* accessible both from the general and the functions workspace.

Declaration of global variables

global m1 conc\_vec kon\_R kon\_R\_1 koff\_R koff\_R\_1

global kon\_R\_B kon\_R\_1B koff\_R\_B koff\_R\_1B C R R\_1 LR LR\_1 LR\_TOT

global L C\_Init R\_Init R\_1\_Init

We start by loading the project:

sbioloadproject('../TOOLBOXES/ENZYME\_KINETICS/Two\_Cooperative\_Binding\_Sites');

or alternatively:

m1 =...

sbmlimport('../TOOLBOXES/BINDING\_KINETICS/Two\_Cooperative\_Binding\_Sites.xml');

Information about the model.

sbioselect(m1,'Type','compartment')

sbioselect(m1,'Type','species')

sbioselect(m1,'Type','parameter')

sbioselect(m1,'Type','reaction')

getequations(m1)

We extract the parameters from the model and save them as variables in the Workspace.

C = sbioselect(m1,'Name','Cell');

L = sbioselect(m1,'Name','L');

R = sbioselect(m1,'Name','R');

R\_1 = sbioselect(m1,'Name','R\_1');

LR = sbioselect(m1,'Name','LR');

LR\_1 = sbioselect(m1,'Name','LR\_1');

LR\_TOT = sbioselect(m1,'Name','LR\_TOT');

kon\_R = sbioselect(m1,'Name','kon\_R');

koff\_R = sbioselect(m1,'Name','koff\_R');

kon\_R\_B = sbioselect(m1,'Name','kon\_R\_B');

koff\_R\_B = sbioselect(m1,'Name','koff\_R\_B');

kon\_R\_1 = sbioselect(m1,'Name','kon\_R\_1');

koff\_R\_1 = sbioselect(m1,'Name','koff\_R\_1');

kon\_R\_1B = sbioselect(m1,'Name','kon\_R\_1B');

koff\_R\_1B = sbioselect(m1,'Name','koff\_R\_1B');

C.Capacity = 1;

R.InitialAmountUnits = 'micromole/liter';

R\_1.InitialAmountUnits = 'micromole/liter';

L.InitialAmountUnits = 'micromole/liter';

LR.InitialAmountUnits = 'micromole/liter';

LR\_1.InitialAmountUnits = 'micromole/liter';

R.InitialAmount = 0.5;

R\_1.InitialAmount = 0.5;

LR.InitialAmount = 0;

LR\_1.InitialAmount = 0;

LR\_TOT.InitialAmount = 0;

Here we store the initial parameters.

C\_Init = 1;

R\_Init = 0.5;

R\_1\_Init = 0.5;

LR\_Init = 0;

LR\_1\_Init = 0;

LR\_TOT\_Init = 0;

Here we establish the rate laws. Without cooperativity they would be:

ReactionFlux1 = (kon\_R\*[L]\*[R] - koff\_R\*[LR])\*Cell

ReactionFlux2 = (kon\_R\_1\*[L]\*[R\_1] - koff\_R\_1\*[LR\_1])\*Cell

% Rate law for R:

% set(m1.reactions(1), 'ReactionRate','kon\_R\*L\*R - koff\_R\*LR');

% Rate law for R\_1:

% set(m1.reactions(2), 'ReactionRate','kon\_R\_1\*L\*R\_1 - koff\_R\_1\*LR\_1');

With positive cooperativity between the two sites they become:

ReactionFlux1 = (kon\_R\*[L]\*[R] - koff\_R\*[LR])\*([R\_1]/([R\_1]+[LR\_1])) - koff\_R\_B\*[LR]\*([LR\_1]/([R\_1]+[LR\_1])))\*Cell

ReactionFlux2 = (kon\_R\_1\*[L]\*[R\_1] - koff\_R\_1\*[LR\_1])\*([R]/([R]+[LR])) - koff\_R\_1B\*[LR\_1]\*([LR]/([R]+[LR])))\*Cell

Notice that:

[R\_1]/([R\_1]+[LR\_1]) = 1-Θ [LR\_1]/([R\_1]+[LR\_1]) = Θ for R1

[R]/([R]+[LR]) = 1-Θ [LR]/([R]+[LR]) = Θ for R

where Θ is the fractional saturation of the receptor.

Rate law for R:

set(m1.reactions(1), 'ReactionRate',...

'kon\_R\*L\*R - koff\_R\*LR\*(R\_1/(R\_1+LR\_1)) - koff\_R\_B\*LR\*(LR\_1/(R\_1+LR\_1))');

Rate law for R\_1:

set(m1.reactions(2), 'ReactionRate',...

'kon\_R\_1\*L\*R\_1 - koff\_R\_1\*LR\_1\*(R/(R+LR)) - koff\_R\_1B\*LR\_1\*(LR/(R+LR))');

Modulated site R:

Without cooperativity

kon\_R.Value = 10;

koff\_R.Value = 300; % Kd(R) = 30 µM

With cooperativity

kon\_R\_B.Value = 10;

koff\_R\_B.Value = 30; % Kd(R)cooper = 3 µM

Kd = koff\_R.Value/kon\_R.Value;

display(['Kd = ' num2str(Kd) ' µM']);

Modulated site R1:

Without cooperativity

kon\_R\_1.Value = 10;

koff\_R\_1.Value = 300; % Kd(R\_1) = 30 µM

With cooperativity

kon\_R\_1B.Value = 10;

koff\_R\_1B.Value = 30; % Kd(R\_1)cooper = 3 µM

Kd\_cooper = koff\_R\_1B.Value/kon\_R\_1B.Value;

display(['Kd\_cooper = ' num2str(Kd\_cooper) ' µM']);

Configuration parameters.

configset\_m1 = getconfigset(m1)

Stop = 0.004;

set(configset\_m1, 'StopTime', Stop);

set(configset\_m1.SolverOptions, 'AbsoluteTolerance', 1.e-9);

set(configset\_m1.SolverType, 'ode15s';

Preliminary simulations can be run to determine the time required to reach equilibrium. In this case we know the binding reaction is complete in less than 0.5 s even at the lowest concentration of the ligand, and we proceed with the simulation of the binding curve.

C.Capacity = 1;

R.InitialAmount = 0.5;

R\_1.InitialAmount = 0.5;

LR.InitialAmount = 0;

LR\_1.InitialAmount = 0;

LR\_TOT.InitialAmount = 0;

High\_Conc = 600;

conc\_vec = logspace(log10(1),log10(High\_Conc),35);

range = conc\_vec <= 1 + High\_Conc;

Bound = zeros(length(conc\_vec),1);

Bound\_R = zeros(length(conc\_vec),1);

Bound\_R\_1 = zeros(length(conc\_vec),1);

Unbound = zeros(length(conc\_vec),1);

We set the simulation time to 1 second in order to be certain that under all concentrations of the ligand we have reached equilibrium. Species are logged in the following order (as derived from 'Binding\_Kinetics.DataNames'): L R LR LR\_1 LR\_TOT R\_1.

Stop = 1;

set(configset\_m1, 'StopTime', Stop);

for i = 1:length(conc\_vec)

error = (rand(3,1)-0.5);

error(1) = 1 + error(1)\*0.10;

error(2) = 1 + error(2)\*0.10;

error(3) = 1 + error(3)\*0.10;

set(C,'Capacity',C\_Init\*error(1));

set(R,'InitialAmount',R\_Init\*error(2)/C.Capacity);

set(R\_1,'InitialAmount',R\_1\_Init\*error(2)/C.Capacity);

set(L,'InitialAmount',conc\_vec(i)\*error(3)/C.Capacity);

set(LR,'InitialAmount',0);

set(LR\_1,'InitialAmount',0);

set(LR\_TOT,'InitialAmount',0);

Binding\_Kinetics = sbiosimulate(m1);

Bound(i) = Binding\_Kinetics.Data(end,5);

Bound\_R(i) = Binding\_Kinetics.Data(end,3);

Bound\_R\_1(i) = Binding\_Kinetics.Data(end,4);

Unbound(i) = Binding\_Kinetics.Data(end,1);

end

Notice how LR and LR\_1 rise together and cannot be distinguished in a pre-steady-state kinetic experiment or in an equilibrium binding experiment. This is because of the symmetric behavior of the cross-cooperativity. If only one site affected the other we would observe different binding curves for the two sites. We can try different methods to fit the synthetic data *a posteriori*. We start by simply plotting the different points of the binding curve.

Two\_cooperative\_binding\_sites\_1 = figure;

set(Two\_cooperative\_binding\_sites\_1,'Units','normalized',...

'Position',[0.1 0.0 0.6 1.0],'Name','Binding Curve Fit');clf

subplot1 = subplot(3,1,1,'Parent',figure(gcf));

box(subplot1,'on');grid(subplot1,'on');hold(subplot1,'all');

plot(Unbound(range),Bound(range),'o',...

'MarkerEdgeColor','k','MarkerFaceColor','c','MarkerSize',7.5);

plot(Unbound(range),Bound\_R(range),':o',...

'MarkerEdgeColor','k','MarkerFaceColor','y','MarkerSize',7.5);

plot(Unbound(range),Bound\_R\_1(range),':o',...

'MarkerEdgeColor','k','MarkerFaceColor','b','MarkerSize',2);

set(gca,'YLim',[-0.1 1.15]);set(gca,'XLim',[-10 10+Unbound(end)]);

xlabel('[Free Ligand]');ylabel('[Bound Ligand]');title('Binding Curve ');

Hyperbola fit. We can fit directly the plot of the fractional saturation of the receptor (FR) vs the amount of free ligand (L) with the hyperbolic function we presented at the beginning of the example, rearranged in terms of fractional saturation:

[LR] = [R0]\*[L]/([L]+Kd)

[LR]/[R0] = [L]/([L]+Kd)

Two hyperbolas are fitted each contributing 50% to the curve. For a single hyperbola it would be: f = fittype('x/(a + x)'); For two hyperbolas we use a two-component fit with 2 unknowns. 'a' and 'b' are the two Kd's. Since the sum of the two receptors (R\_Init + R\_1\_Init) is 1, the fractional saturation is equal to the 'bound' fraction.

FR = Bound/(R\_Init + R\_1\_Init);X = Unbound;Y = FR;

f = fittype('(0.5\*x/(a + x)) + (0.5\*x/(b + x))');

[Hyperb,GOF] = fit(X,Y,f,'StartPoint',[3 30])

Hyperb\_params = coeffvalues(Hyperb);

Kd\_1 = Hyperb\_params(1);Kd\_2 = Hyperb\_params(2);

plot(X,Hyperb(X),'-r');ylim([-0.1,1.15]); xlim([-10,Unbound(end)+10]);

legend('LR\\_TOT','LR','LR\\_1','Hyperb. fit','Location','Best');

Scatchard Fit.

subplot2 = subplot(3,1,2,'Parent',figure(gcf));

box(subplot2,'on');grid(subplot2,'on');hold(subplot2,'all');

s\_range = range;

s\_range\_1 = true(1,35);s\_range\_1(18:end) = false;

s\_range\_2 = true(1,35);s\_range\_2(1:18) = false;

X = Bound(s\_range);

Y = Bound(s\_range)./Unbound(s\_range);

plot(X,Y,'-ys','MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSize',5);

Low affinity range

X1 = Bound(s\_range\_1);

Y1 = Bound(s\_range\_1)./Unbound(s\_range\_1);

plot(X1,Y1,'yo','MarkerEdgeColor','b','MarkerFaceColor','none','MarkerSize',8);

f = fittype('a\*x + b');

[Scatchard\_1,GOF\_1] = fit(X1,Y1,f,'Robust','Bisquare','StartPoint',[-0.004 0.035])

plot(Scatchard\_1,'-b');

Scatchard\_1\_params = coeffvalues(Scatchard\_1);

Kd\_1 = -1/Scatchard\_1\_params(1);

Receptor\_conc\_1 = -Scatchard\_1\_params(2)/Scatchard\_1\_params(1);

High affinity range

X2 = Bound(s\_range\_2);

Y2 = Bound(s\_range\_2)./Unbound(s\_range\_2);

plot(X2,Y2,'yo','MarkerEdgeColor','r','MarkerFaceColor','none','MarkerSize',8);

f = fittype('a\*x + b');

[Scatchard\_2,GOF\_2] = fit(X2,Y2,f,'Robust','Bisquare','StartPoint',[-1 1])

plot(Scatchard\_2,'-r');

Scatchard\_2\_params = coeffvalues(Scatchard\_2);

Kd\_2 = -1/Scatchard\_2\_params(1);

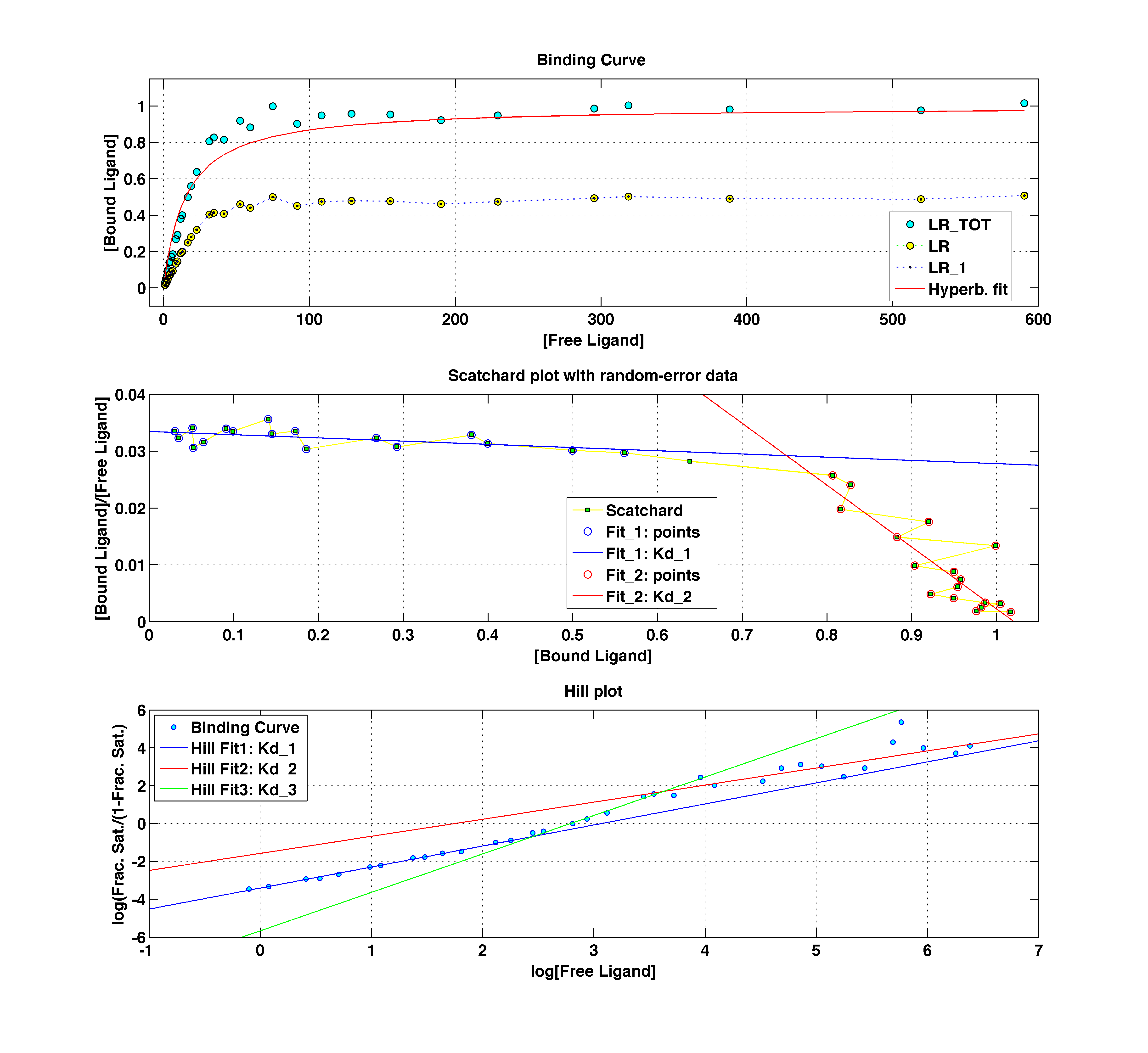
Receptor\_conc\_2 = -Scatchard\_2\_params(2)/Scatchard\_2\_params(1);

ylim([0,0.04]);xlim([0,1.05]);

xlabel('[Bound Ligand]');ylabel('[Bound Ligand]/[Free Ligand]');

title('Scatchard plot with random-error data');

legend('Scatchard','Fit\\_1: points','Fit\\_1: Kd\\_1','Fit\\_2: points','Fit\\_2: Kd\\_2','Location','Best');



The Scatchard plot provides a reasonable value for the high affinity range of the binding sites, but a poor one for the low affinity range. However, it is possible to obtain a good estimate also for the low affinity range by taking the -1/slope of the line connecting the intercept of the low affinity range on the ordinate to the intercept of the high affinity range on the abscissa (these are the two extremes of the distribution of points in the plot).

Low affinity range corrected

Kd\_1\_corr = 1/(Scatchard\_1\_params(2)/Receptor\_conc\_2)

Hill and Logistic fit. The logistic fit is based on the equation:

Response = (Max-Min)/(1+(***Kd***/x)**nH**

in which nH is the Hill coefficient, which theoretically cannot be higher than the number of binding sites that are occupied simultaneously. Three conditions are possible:

nH > 1 - Positive cooperativity: binding of the ligand to one site increases the affinity of the other site for the ligand.

nH < 1 - Negative cooperativity: binding of the ligand to one site decreases the affinity of the other site for the ligand.

nH = 1 - Non-cooperativity: the binding sites are independent.

The traditional way of determining the Hill coefficient is with a plot of the Hill equation in the form:

**log(Θ/(1-Θ)) = nH\*log[L] - log(*Kd*)**

Therefore, log[y/(1-y)] (where y is the fractional saturation **Θ** of the receptor) is plotted against log[x] (where [x] is the free (unbound) ligand concentration).

The slope is the Hill coefficient **nH**, and the intercept with the ordinate (Y axis) is equal to **-log(*Kd*)**.

IMPORTANT: it is often reported that the intercept with the X axis represents the *Kd* because at Y = 0 **Θ** = 0.5. However, this is only true if nH = 1. In the more general case, solving for the x intercept (when the y axis variable is 0) in the Hill equations gives:

0 = nH\*log[L] -log(*Kd*)

nH\*log[L] = log(*Kd*)

log[L] = (log(*Kd*))/nH

subplot3 = subplot(3,1,3,'Parent',figure(gcf));

box(subplot3,'on');grid(subplot3,'on');hold(subplot3,'all');

FR = Bound/(R\_Init + R\_1\_Init);

X = log(Unbound(s\_range));Y = log(FR(s\_range)./(1 - FR(s\_range)));

plot(X,Y,'o', 'MarkerEdgeColor','b','MarkerFaceColor','c','MarkerSize',5);

xlabel('log[Free Ligand]');ylabel('log(Frac. Sat./(1-Frac. Sat.)');title('Hill plot');

f = fittype('a\*x + b');

[Hill\_1,GOF\_1] = fit(X(1:12),Y(1:12),f,'StartPoint',[1 -4],'Robust','on')

plot(Hill\_1,'-b');

[Hill\_2,GOF\_2] = fit(X(20:35),Y(20:35),f,'StartPoint',[1 1],'Robust','on')

plot(Hill\_2,'-r');

[Hill\_3,GOF\_3] = fit(X(15:19),Y(15:19),f,'StartPoint',[2 -8],'Robust','on')

plot(Hill\_3,'-g');

ylim([-6 6]);% xlim([0 10]);

legend('Binding Curve','Hill Fit1: Kd\\_1','Hill Fit2: Kd\\_2',...

'Hill Fit3: Kd\\_3','Location','NorthWest');

Retrieve the fit parameters.

Hill\_1\_params = coeffvalues(Hill\_1);

Hill\_2\_params = coeffvalues(Hill\_2);

Hill\_3\_params = coeffvalues(Hill\_3);

nH\_1 = Hill\_1\_params(1);

nH\_2 = Hill\_2\_params(1);

nH\_3 = Hill\_3\_params(1);

Kd\_1 = exp(-Hill\_1\_params(2));

Kd\_2 = exp(-Hill\_2\_params(2));

Kd\_3 = exp(-Hill\_3\_params(2));

xlabel('log[Free Ligand]');ylabel('log(Frac. Sat./(1-Frac. Sat.)');title('Hill plot');

Thus, either a Scatchard plot with corrected low affinity range (Kd\_1\_corr) or a Hill plot can provide a good initial estimate of the boundary values for the *Kd*'sof the two binding sites.

Further improvement of this initial estimate can be obtained with a direct fit of the progress curves to the binding curve. This fit is just another application of non-linear least squares; the difference with respect to the previous examples in which a hyperbola or other type of theoretical curve is fit to the experimental points is that in this case we start with a guess of the rate constants (typically just the *koff* values) and we calculate the entire binding curve based on that guess. Then, we refine the rate constants by minimizing the residual between the calculated binding curve and the experimental (or in our case the synthetic) one.

The initial guess for the *koff* values is obtained by multiplying the *Kd* values from the Hill plot by a reasonable value of *kon* (typically 10 µM-1 s-1). These values are stored in a vector '*pin*' (*'parameters in*') that is used as input for the function that refines the rate constants.

p1 = Kd\_1\*10;

p2 = Kd\_2\*10;

pin=[p1;p2];

Here we set the vector, 'Product\_vector', of the experimental binding curve against which we refine the binding curve 'Product\_sim\_vector' calculated with the updated values of the vector *pin*:

Product\_vector = Bound;

Conc\_vector = conc\_vec;

Next, we set up the function that calculates the binding curve vector at each iteration and passes it to the minimizer. Notice how the *cross-cooperativity* is assumed to be exactly symmetric, so only two values of *koff* (modulated and unmodulated) need to be determined for both R and R\_1. The relevant variables necessary for the calculation are passed to the function workspace as global variable:

%-------------------------------------------------------------------------------------------------------------------------

function Product\_sim\_vector = two\_cooperative\_binding\_sites\_fit(pin,Conc\_vector)

global m1 conc\_vec kon\_R kon\_R\_1 koff\_R koff\_R\_1

global kon\_R\_B kon\_R\_1B koff\_R\_B koff\_R\_1B C R R\_1 LR LR\_1 LR\_TOT

global L C\_Init R\_Init R\_1\_Init

%

set(koff\_R,'Value',pin(1));

set(koff\_R\_1,'Value',pin(1));

set(koff\_R\_B,'Value',pin(2));

set(koff\_R\_1B,'Value',pin(2));

C.Capacity = 1;

R.InitialAmount = 0.5;

R\_1.InitialAmount = 0.5;

LR.InitialAmount = 0;

LR\_1.InitialAmount = 0;

LR\_TOT.InitialAmount = 0;

Bound = zeros(length(conc\_vec),1);

Bound\_R = zeros(length(conc\_vec),1);

Bound\_R\_1 = zeros(length(conc\_vec),1);

Unbound = zeros(length(conc\_vec),1);

for i = 1:length(conc\_vec)

set(C,'Capacity',C\_Init);

set(R,'InitialAmount',R\_Init);

set(R\_1,'InitialAmount',R\_1\_Init);

set(L,'InitialAmount',conc\_vec(i));

set(LR,'InitialAmount',0);

set(LR\_1,'InitialAmount',0);

set(LR\_TOT,'InitialAmount',0);

Binding\_Kinetics = sbiosimulate(m1);

Bound(i) = Binding\_Kinetics.Data(end,5);

Bound\_R(i) = Binding\_Kinetics.Data(end,3);

Bound\_R\_1(i) = Binding\_Kinetics.Data(end,4);

Unbound(i) = Binding\_Kinetics.Data(end,1);

end

Product\_sim\_vector = Bound;

end

%-------------------------------------------------------------------------------------------------------------------------

We can carry out the refinement using minimizers from different toolboxes as we did for the non-linear least squares example analyzed in previous chapters.

**Refinement with *lsqcurvefit*.**

This function is slower than *nlinfit* (see below), but usually has a wider radius of convergence, and thus can be used as a first attempt if we suspect the *Kd* value derived from the Scatchard or the Hill plot are not very accurate.

options = ...

optimoptions('lsqcurvefit','Display','iter','FinDiffType','central','TolFun',1e-8,'TolX',1e-8);

[u,sos,res,~,~,~,J] = ...

lsqcurvefit(@two\_cooperative\_binding\_sites\_fit,pin,Conc\_vector,Product\_vector,...

[],[],options);

**Refinement with *nlinfit*.**

Here we use the nlinfit function from the Statistics Toolbox. This is the best choice if the toolbox is available. It can be used after lsqcurvefit to first narrow the radius of convergence.

if exist('u','var')

pin = u;

end

options=statset('TolX',1e-9,'TolFun',1e-9,'Display','iter',...

'MaxIter',200);

[u,R1,J1,Cov1,MSE1] = ...

nlinfit(Conc\_vector,Product\_vector,'two\_cooperative\_binding\_sites\_fit',pin,options);

[Corr1,sigma1] = corrcov(Cov1);

sos1 = R1'\*R1

[fcurve1,delta1] = nlpredci('two\_cooperative\_binding\_sites\_fit',Conc\_vector,u,R1,'Covar',Cov1);

u\_ci = nlparci(u,R1,'covar',Cov1);

**Refinement with *fminsearch*.**

The following is an alternative solution with the generic MATLAB function *fminsearch* (uses Simplex search rather than derivatives: it minimizes the scalar output of a function. For this reason the model function includes the calculation of the sum of squared residuals, and therefore there is no Jacobian). This is the slowest method, but in some cases it may be able to find the minimum better than other methods.

modelfun = @(pin) sum((two\_cooperative\_binding\_sites\_fit(pin,Conc\_vector)-Product\_vector).^2);

options = optimset('Display','iter', 'TolFun',1e-6, 'TolX',1e-6);

[u,fval,exitflag,output] = fminsearch(modelfun,pin,options);

Here we regenerate the binding curves:

C.Capacity = 1;

R.InitialAmount = 0.5;

R\_1.InitialAmount = 0.5;

LR.InitialAmount = 0;

LR\_1.InitialAmount = 0;

LR\_TOT.InitialAmount = 0;

Bound = zeros(length(conc\_vec),1);

Bound\_R = zeros(length(conc\_vec),1);

Bound\_R\_1 = zeros(length(conc\_vec),1);

Unbound = zeros(length(conc\_vec),1);

for i = 1:length(conc\_vec)

set(C,'Capacity',C\_Init);

set(R,'InitialAmount',R\_Init);

set(R\_1,'InitialAmount',R\_1\_Init);

set(L,'InitialAmount',conc\_vec(i));

set(LR,'InitialAmount',0);

set(LR\_1,'InitialAmount',0);

set(LR\_TOT,'InitialAmount',0);

Binding\_Kinetics = sbiosimulate(m1);

Bound(i) = Binding\_Kinetics.Data(end,5);

Bound\_R(i) = Binding\_Kinetics.Data(end,3);

Bound\_R\_1(i) = Binding\_Kinetics.Data(end,4);

Unbound(i) = Binding\_Kinetics.Data(end,1);

end

figure;plot(conc\_vec,Bound,'-b',conc\_vec,Product\_vector,'or');hold on

We can compare the residual from this fit with that we would obtain taking the Hill plot solution:

set(koff\_R,'Value',Kd\_1\*10);

set(koff\_R\_1,'Value',Kd\_1\*10);

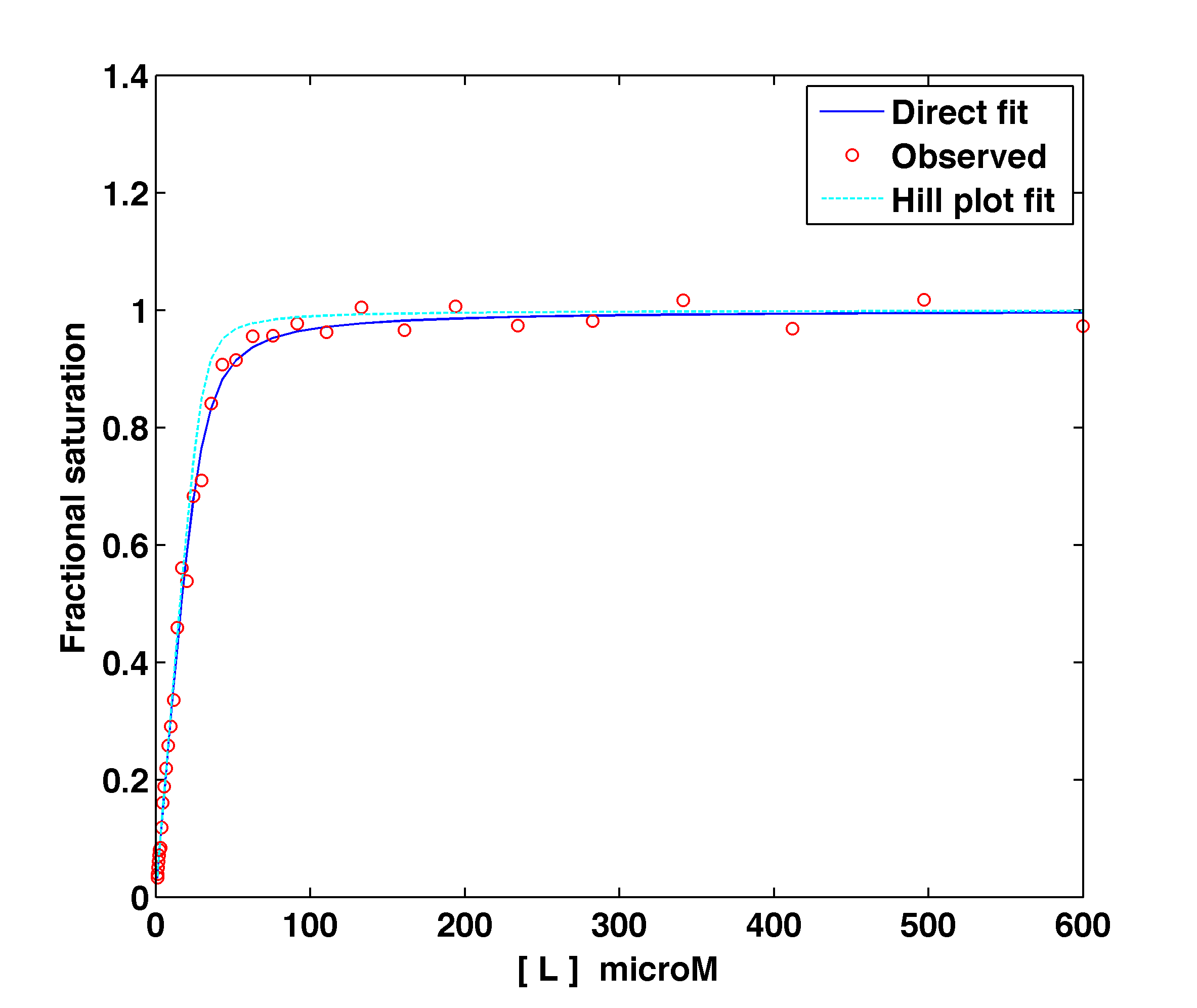
set(koff\_R\_B,'Value',Kd\_2\*10);

set(koff\_R\_1B,'Value',Kd\_2\*10);

C.Capacity = 1;

R.InitialAmount = 0.5;

R\_1.InitialAmount = 0.5;

LR.InitialAmount = 0;

LR\_1.InitialAmount = 0;

LR\_TOT.InitialAmount = 0;

Bound = zeros(length(conc\_vec),1);

Bound\_R = zeros(length(conc\_vec),1);

Bound\_R\_1 = zeros(length(conc\_vec),1);

Unbound = zeros(length(conc\_vec),1);

for i = 1:length(conc\_vec)

set(C,'Capacity',C\_Init);

set(R,'InitialAmount',R\_Init);

set(R\_1,'InitialAmount',R\_1\_Init);

set(L,'InitialAmount',conc\_vec(i));

set(LR,'InitialAmount',0);

set(LR\_1,'InitialAmount',0);

set(LR\_TOT,'InitialAmount',0);

Binding\_Kinetics = sbiosimulate(m1);

Bound(i) = Binding\_Kinetics.Data(end,5);

Bound\_R(i) = Binding\_Kinetics.Data(end,3);

Bound\_R\_1(i) = Binding\_Kinetics.Data(end,4);

Unbound(i) = Binding\_Kinetics.Data(end,1);

end

R\_hill = (Bound-Product\_vector);

sos\_hill = R\_hill'\*R\_hill

sos1

plot(conc\_vec,Bound,'--c');hold off

legend('Direct Fit','Observed','Hill plot fit')

xlabel('[L] microM ')

ylabel('Fractional Saturation ')

%%

close all

save(savefile);

It is very easy to check what the binding curve would look if there was negative instead of positive cooperativity: in this case we will increase the value of koff\_R\_B and koff\_R\_1B rather than decreasing it.

Modulated site R:

Without cooperativity

kon\_R.Value = 10;

koff\_R.Value = 300; % Kd(R) = 30 µM

With cooperativity

kon\_R\_B.Value = 10;

koff\_R\_B.Value = 3000;

Kd = koff\_R.Value/kon\_R.Value;

display(['Kd = ' num2str(Kd) ' µM']);

Modulated site R1:

Without cooperativity

kon\_R\_1.Value = 10;

koff\_R\_1.Value = 300; % Kd(R\_1) = 30 µM

With cooperativity

kon\_R\_1B.Value = 10;

koff\_R\_1B.Value = 3000;

Kd\_cooper = koff\_R\_1B.Value/kon\_R\_1B.Value;

display(['Kd\_cooper = ' num2str(Kd\_cooper) ' µM']);

**Conclusion**

The presence of positive cooperativity between binding sites can be inferred from the presence of a 'convex' Scatchard plot. In this case, an accurate determination of the Kd's of the individual binding sites may be difficult with standard methods like Schatchard or hyperbola fitting. It is often useful to carry out a logistic fit of the binding data and to compare its determination of the Hill coefficient with that derived from the Hill plot.

Negative cooperativity gives origin to a 'concave' Scatchard, somewhat similar to that obtained with two independent binding sites, but more markedly represented by a single curve rather than by two distinct phases. None of the different analysis methods give very good values for the Kd's of the sites. Hill plot and logistic fit both show good performance in detecting the presence of negative cooperativity.

**PRACTICE**

**1.** The X-ray structure ofreceptor R for ligand L has been determined and it shows the receptor to be a homodimer with one binding site for L in each monomer. We are interested in determining whether the two binding sites observed in the X-ray structure are functionally identical or not. For this purpose, we carry out an equilibrium binding experiment in which increasing concentration of L (up to 6 mM) are added to a solution containing 1 μM R. Binding is detected by following the fluorescence changes at 380 nm after excitation in the ultraviolet. The concentrations of ligand used in the binding experiments and the observed fluorescence changes are reported in the 1st and 2nd column (respectively) of the matrix:

../DATABASE/Problem\_Binding\_Curve.txt

Retrieve these data with the '*dlmread*' function, and using one or more of the tools described in this chapter:

a. Determine if the two binding sites are identical (behaving as a single site), independent, or cooperative.

b. Determine the *Kd* of L at these binding sites using different analysis tools and select the best result based on the residual of your fits with respect to the observed fluorescence changes.