SPRTool User Manual[1]

Software Version: 0.9
Edition 1
October 1 2003

SPRTool Development Team
Team Leader: Raimund J. Ober[*,**]
Current Members: Jerry Chao[***], Xuming Lai[*,***],
Palmer Long[***], E. Sally Ward[**,***]
Past Member: Jeffrey Caves

[*]Department of Electrical Engineering
University of Texas at Dallas
Richardson, TX 75083

[**]Cancer Immunobiology Center NB9.106
UT Southwestern Medical Center at Dallas
6000 Harry Hines Boulevard
Dallas, TX 75390-8576

[***]Center for Immunology NB9.106
UT Southwestern Medical Center at Dallas
6000 Harry Hines Boulevard
Dallas, TX 75390-8576

Email: wardlab@utsouthwestern.edu

[1]The development work for SPRTool was supported in part by the National Institutes of Health (RO1 GM58538)

Contents

Chapter 1

Overview

This chapter will give an overview of SPRTool and the components that make up the software package.

1.1 A Word of Caution

The SPRTool software has been used time and again in our lab for analyzing data from our own BIAcore experiments. While we have found the software useful and the results it produces reliable, we do not make any guarantees whatsoever on the correctness of the results that you obtain from using the SPRTool software. Before you begin using the software, you should also be aware that the SPRTool software is intended for use in research and should not be used in any way for commercial purposes. Please feel free to email us at wardlab@utsouthwestern.edu with any questions or comments concerning the software.

1.2 What is SPRTool?

The SPRTool software is designed to document and analyze Surface Plasmon Resonance experiments conducted on the BIAcore or similar machine. The difference between this software package and other similar SPR analysis software is its ability to deal with very large data sets. The software is written in the technical computing language MATLAB and also makes use of the MATLAB Optimization Toolboxfor some of its advanced features. To run SPRTool, MATLAB must be installed. To take full advantage of the capabilities of SPRTool, the MATLAB Optimization Toolbox must also be installed.

1.3 Overview of Documentation

The documentation supplied with the software are:

- SPRTool User Manual (This Document)
- SPRTool Reference Manual

1.4 What Makes Up SPRTool?

The capabilities of SPRTool are provided by four different but interrelated *classes*. You can think of a **class** simply as a set of functionalities. Here then is a list of the four sets of functionalities that make up SPRTool and a brief description of the purpose of each. For more detailed information on each class please refer to the SPRTool Reference Manual.

Note: In the descriptions below, do not worry if you're unfamiliar with the terms *script* and *object*. They will be discussed in detail in the *Tutorial* chapter of this manual.

- **Chip Class** - Stores information about the chip used in the BIAcore experiment. SPRTool provides the following scripts for working with the Chip class:
  - **Chip Script** - Stores user-supplied information about a chip in a chip object.
- **Experiment Class** - Stores the sensorgram data and plots raw data. SPRTool provides the following scripts for working with the Experiment class:
  - **Experiment Script** - Organizes and stores experimental data in an experiment object.
- **Adjust Class** - Performs routine adjustments on the sensorgram data (zero adjustment, background subtraction, etc.) for use in further analysis techniques and plots raw and adjusted data. SPRTool provides the following scripts for working with the Adjust class:

  - **Adjust Script** - Prepares experimental data for equilibrium and kinetic analysis by performing various types of adjustments on the sensorgram data. Both the raw and the adjusted data are stored in an adjust object.

- **Equilibrium Class** - Analyzes the equilibrium points of each sensorgram and has tools for determining equilibrium constants. SPRTool provides the following scripts for working with the Equilibrium class:
  - **Equilibrium Script** - Prepares sensorgram data for equilibrium analysis by estimating the equilibrium values of the sensorgrams. The calculated results are stored in an equilibrium object.
  - **Equilibrium Scatchard Script** - Performs equilibrium analysis using the traditional Scatchard technique and computes values for K_d, K_a, and R_{max}. It also generates Scatchard plots that are displayed to the user.
  - **Equilibrium Ligand Activity Loss Script** - Performs equilibrium analysis in a way that compensates for the loss of ligand activity. Using one of four available exponential decay models to describe R_{max}, the script computes values for the equilibrium constants K_d and K_a. It also generates curve-fitted plots that are displayed to the user.

In addition to the scripts listed above, SPRTool includes two kinetic scripts, four simulation scripts, and the SPRTool Viewer.

- **Kinetic Scripts**
  - **Kinetic Hankel Script** - Estimates the kinetic parameters of a sensorgram with a subspace algorithm. It estimates the association or dissociation constants for a sensorgram that resulted from a number of independent parallel interactions, including the standard 1:1 interaction.
  - **Kinetic Optimization Script** - Calculates the association rate constant, dissociation rate constant, and the R_{max} value for one sensorgram or a group of sensorgrams. The script uses either the Independent Interactions Model or the Mass Transport Model to generate the simulated data required by the computation.
- **Simulation Scripts**
  - **Equilibrium Scatchard Simulate Script** - Simulates equilibrium data for the purpose of demonstrating the analysis of equilibrium data using the traditional Scatchard technique.
  - **Equilibrium Ligand Activity Loss Simulate Script** - Simulates equilibrium data with loss of ligand activity for the four different decay models supported by the equilibrium ligand activity loss script.
  - **Kinetic Independent Interaction Simulate Script** - Simulates kinetic data with the assumption of a 1:1 interaction. The data is simulated using the independent interaction model.
  - **Kinetic Mass Transport Simulate Script** - Simulates kinetic data with the assumption of a 1:1 interaction with mass transport present. The mass transport effect is simulated using the standard compartment model.
- **SPRTool Viewer** - A Graphical User Interface (GUI) tool used to view the contents of any SPRTool object you choose. A detailed discussion of this viewer is presented in the chapter entitled **SPRTool Viewer**.

1.5 Acknowledgments

The writing of this software has been supported by the NIH Grant#RO1GM58538.

Chapter 2

Installation

This chapter will give a detailed view of the installation of SPRTool.

2.1 Platform Requirements

The SPRTool software should be portable to any platform that runs MATLAB, including several variants of the UNIX operating system. However, we must mention that the software was developed and tested exclusively on Windows 2000 and that since we do not have access to other platforms, we could not verify that SPRTool does in fact run properly on other platforms. The installation instructions that follow will strictly be given in PC format.

2.2 Software Prerequisites

1. You must have MATLAB installed and working. We recommend the use of **MATLAB 6.5** or higher since 6.5 is the version on which the SPRTool software was developed and tested. If you are running an older version of MATLAB you may encounter problems when using SPRTool.

2. You must have MATLAB's **Optimization Toolbox** installed to take advantage of some of the more advanced features of SPRTool. An example of this would be the exponential curve fitting options available for equilibrium analysis. We recommend the use of Optimization Toolbox 2.2, the version on which SPRTool was developed and tested.

- You can find information about purchasing the latest version of MATLAB at:
  http://www.mathworks.com/products/matlab/

- Or if you are a student, your school may already have a University License for you to use MATLAB. You can also look into the latest release of the Student Version of MATLAB at:
  http://www.mathworks.com/products/education/student_version/sc/

- You can find information about purchasing the latest version of the Optimization Toolbox at:
  http://www.mathworks.com/products/optimization/

2.3 Acquiring and Extracting the Program

1. With your web browser open go to the following URL:

   http://www2.utsouthwestern.edu/wardlab/sprtool/

   Follow the instructions there for acquiring the SPRTool software.

2. Using your favorite zip file extraction program, extract the SPRTool program to the **\toolbox\local** directory under your MATLAB installation directory.
   (Note: The \toolbox\local directory is part of MATLAB's standard installation and should already exist.) For example, if your MATLAB installation directory is **C:\MATLAB6p5**, then you would extract the SPRTool program to **C:\MATLAB6p5\toolbox\local**.

3. You should now see a directory called **SPRTool** in your \toolbox\local directory. Continuing with the example from the previous step, you would now see the directory **C:\MATLAB6p5\toolbox\local\SPRTool**. For the language of the rest of this manual we will refer to this **SPRTool** directory as the **SPRTool Installation Directory**.

NOTE: It is **EXTREMELY IMPORTANT** that you know the location of your SPRTool Installation Directory as you will need it time and again. If you wish, you can extract the SPRTool program to any directory you like, but you must know the location of your SPRTool Installation Directory in order to proceed with the installation of the SPRTool program and to use the program after it is installed.

2.4 Installing the SPRTool Program

Note: The instructions that follow will use **C:\MATLAB6p5\toolbox\local\SPRTool** as a sample **SPRTool Installation Directory**. If your SPRTool Installation Directory is different, then you **MUST substitute it with your own SPRTool Installation Directory**. Also, be sure you enclose the path to your SPRTool Installation Directory in **single quotes**!!

1. Start up MATLAB.
2. Go to your **SPRTool Installation Directory** by entering the following command at the **Command Window** prompt:

   ```
   >> cd('C:\MATLAB6p5\toolbox\local\SPRTool')
   ```

3. Execute the install script by entering the following command at the **Command Window** prompt:

   ```
   >> SPR_install
   ```

4. Follow the instructions in the dialog boxes that appear. (Basically your job is to locate and select your SPRTool Installation Directory and to locate and select a directory for storing your SPRTool sessions.)
5. When the install script finishes executing, you should see a dialog box telling you that everything was a success. If this line does not appear there was an error in the Install and you need to try again.
6. Go to the Tutorial Chapter to find out how to work with SPRTool.

Chapter 3

Tutorial

This chapter contains the Tutorial provided with SPRTool. This Tutorial is a guide for the user to learn and understand the workings of SPRTool. It will demonstrate the capabilities of the software and help the user get the most out of a SPRTool Session. This chapter is broken down into sections where each section describes the usage of a particular script to generate the objects you need to analyze your SPR data. Though this tutorial will only demonstrate the usage of the four *setup* scripts provided with SPRTool, going through it in its entirety should nevertheless help answer many of your questions as a lot of information about how to use the software is covered in the Tutorial.

3.1 Session Startup

An SPRTool session corresponds to a directory that contains all the scripts, data, and saved objects pertaining to a specific SPRTool analysis. In order to start the tutorial, we must first open a New Tutorial Session using the SPRTool session manager. The session manager is responsible for starting the session that you specify by taking you to the correct directory and by setting up the appropriate MATLAB environment for you to perform your analysis.

3.1.1 Starting a New SPRTool Session

Note: The instructions that follow will use **C:\MATLAB6p5\toolbox\local\SPRTool** as a sample **SPRTool Installation Directory**. If your SPRTool Installation Directory is different, then you **MUST substitute it with your own SPRTool Installation Directory**. Also, be sure you enclose the path to your SPRTool Installation Directory in **single quotes**!!

1. With MATLAB open go to your **SPRTool Installation Directory** by entering the following command at the **Command Window** prompt:

   >> **cd('C:\MATLAB6p5\toolbox\local\SPRTool')**

2. Start the session manager by entering the following command at the **Command Window** prompt:

   >> **SPR_session**

3. Click the **"NEW"** button in the window that appears to create a new session.

4. Click the **"TUTORIAL"** button in the window that appears to indicate that you want to open a tutorial session. (**NOTE: When you start doing your own SPR Analysis with SPRTool, you would click "REGULAR" at this point to create a regular session.**)

5. Enter a name for the tutorial session in the input box that appears. (If you wish, you can also accept the default name "MyTutorialSession" already in the input box.) Click "OK".

6. You have now opened a New Tutorial Session and we are ready to begin.

3.2 Chip Class

A chip object is a type of variable that holds information describing how a particular chip is coupled. Whenever a chip is coupled, creating a chip object will help keep track of the pertinent information about that chip. This information later will be utilized by the experiment object.

3.2.1 A Note About the Sections that Follow

The material presented in the section **Working with SPRTool Objects** is meant to introduce the user to the concept of an SPRTool object and to some basic operations that can be performed on an SPRTool object. While the presentation uses a chip object to demonstrate the various things that can be done with an SPRTool object, the ideas covered can be applied in a straightforward manner to other types of SPRTool objects like the experiment object and the adjust object.

The material covered in **Working with SPRTool Objects** is of particular importance to those users who feel inclined or have the need to work directly with an SPRTool object. As a general rule, however, we recommend the use of *scripts* provided by SPRTool that will allow the user to work with the various SPRTool objects in a simplified way. Details regarding the concept and use of SPRTool scripts are discussed in the section **Working with SPRTool Scripts**.

For those users who are already familiar with the concept of objects in the context of MATLAB programming, the section **Working with SPRTool Objects** may be skipped altogether.

3.2.2 Working with SPRTool Objects

SPRTool works with something called *SPRTool objects*. Quite simply, an **SPRTool object** is the program's way of storing information about a particular thing. A chip object, for instance, stores information that describes a particular chip used in a BIACore experiment. A chip object keeps track of things like the name of the person who coupled the particular chip and what ligand was coupled to each of the chip's channels. To help you become more familiar with SPRTool objects in general, let's begin the tutorial by working with a **chip object**.

1. **Creating an Empty SPRTool Object**
   Before we can work with a chip object, we need to create an empty one to begin with. That is, we need a chip object that does not yet contain information that describes a particular chip. Enter the following at the **Command Window** prompt:

   ```
   >> chip1 = Chip;
   ```

   This will create an **empty** chip object called **'chip1'** and store it in your workspace. You can verify that chip1 has been created by noticing that there is something called 'chip1' in the **Workspace** window.

2. **Finding Out What an SPRTool Object Can Store**
   Before we can store information in the empty chip object we have just created, we need to first determine what it can store. Every SPRTool object stores information in a set of *fields* that are specific to its type. Therefore, to find out what a given SPRTool object can store, we simply need to know what fields it contains. To see the fields in a chip object, enter the following at the **Command Window** prompt:

>> **help Chip**

This will generate a listing of information pertaining to chip objects in general. In particular, we are interested in the section of the listing entitled “Classfields”. (If you need to, use the Command Window scroll bar to scroll up until you see the “Classfields” section.) The “Classfields” section provides a listing of all the fields contained in a chip object. For each field, the type of information it can store is displayed along with a description of the information it is intended to store.

Let’s take, for example, the field called “WhoCoupled” which you should find towards the bottom of the field listing. As you can see, the “WhoCoupled” field can store information of type *string* and is intended to hold the name of the person who coupled the particular chip.

3. **Displaying the Contents of an SPRTool Object**
Before we store any information in our empty chip object, let’s see what it already contains. In other words, we want to see what information is currently stored in its fields. To display the contents of **any** SPRTool object, all you need to do is enter its name at the **Command Window** prompt:

   >> **chip1**
   Note: Make sure you don’t type a semicolon (;) at the end or else you won’t be able to see anything!!

   This will give a listing of the contents of the chip object. Notice that while fields like the “Object Setup Date” contain meaningful information, the majority of the fields contains the word ‘unspecified’. This is understandable since we have not stored any meaningful information in the chip object yet. Notice in particular that the “Coupled By:” field is currently unspecified.

   A note should be made here about the differences between the field names that you see in the content display and the field names that you saw with the help command in the previous step. While they are different, it should be very intuitive as to which field in the content display corresponds to which field in the help command listing. It should be fairly obvious, for instance, that the “Coupled By:” field in the content display correponds to the “WhoCoupled” field in the help command listing.

4. **Assigning Information to a Field in an SPRTool Object**
To see an example of how we store meaningful information in an object, let’s assign the name ‘John Doe’ to chip1’s “WhoCoupled” field to reflect the fact that John Doe was the person who coupled the chip represented by chip1. Let’s type the following at the **Command Window** prompt:

   >> **chip1.WhoCoupled = ’John Doe’;**
   Note: You must enclose the name ’John Doe’ in single quotes or else you will get an error!!

   A note should be made here about assigning information to *string* type fields like the chip object’s “WhoCoupled” field. Whenever a field’s information type is **string**, it is imperative that you **enclose** the information you want to assign it in **single quotes**. An error will result otherwise.

5. **Retrieving Information from a Field in an SPRTool Object**
   Now let's verify the information assignment we have just made. Type the following at the **Command Window** prompt:

   ```
   >> chip1.WhoCoupled
   ```
   Note: Make sure you don't type a semicolon (;) at the end or else you won't be able to see anything!!

   This will specifically return the information stored in the "WhoCoupled" field. You should see the name 'John Doe' displayed.

   Alternatively, you could perform the same verification by displaying the entire contents of the chip object. Issue the same command as before as follows:

   ```
   >> chip1
   ```

   Noting that the "Coupled By:" field in the content display correponds to the "WhoCoupled" field in the help command listing, you should verify that the "Coupled By:" field now contains the name 'John Doe'.

6. **Saving an SPRTool Object to Disk**
   You already know that our object **'chip1'** is saved in the workspace. However, this form of storage is not permanent. For example, it will be deleted once you exit MATLAB. To store a permanent copy of our object, let's save a copy of it to disk with the following commands at the **Command Window** prompt:

   ```
   >> chip1.ObjectFileName = 'MyFirstChip';
   >> saveobject(chip1);
   ```

   The first command specifies that our object should be saved under the file name 'MyFirstChip'. The second command actually saves our object to disk. Because the standard extension "_chipobject.mat" for a saved chip object will be appended to the file name we have specified, you should look for the file 'MyFirstChip_chipobject.mat' in the **Current Directory** window to verify that our chip object has indeed been saved to disk.

7. To demonstrate how we can load an object that has been saved to disk, first let's create another empty chip object and view its contents. Enter the following at the **Command Window** prompt:

   ```
   >> chip2 = Chip;
   >> chip2
   ```

   The first line will create an empty chip object called **'chip2'**. You should verify its existence by looking at the **Workspace** window. The second line will display its contents. In particular, notice that just like when we first created the first chip object, the "Coupled By:" field of this empty object is unspecified.

8. **Loading a Saved SPRTool Object from Disk**
   Now let's load our saved object into **'chip2'** and then view its contents again. Issue the following commands at the **Command Window** prompt:

```
>> chip2 = loadobject(Chip, 'MyFirstChip_chipobject.mat');
>> chip2
```

The first line will load our saved object into **'chip2'**. Notice that when loading a saved object, we must provide the full name (i.e. file name *and* extension) of the file that contains the saved object. The second line will display the contents of **'chip2'**. You should see that the information stored in the "Coupled By:" field is no longer unspecified. Instead you should see 'John Doe', the name that we assigned to the "WhoCoupled" field of **'chip1'** before we saved it to disk.

9. **Creating, Opening, and Printing an SPRTool Object Report**
   SPRTool provides you with a way to create a report that contains a listing of an SPRTool object's contents. The generated report will be a **.m** file that shares the same name as the file that the object would be saved to. For example, since our chip object **chip1** was saved to the file 'MyFirstChip_chipobject.mat', we can expect the generated report to have the name **'MyFirstChip_chipobject.m'**. Let's try it by issuing the following command at the **Command Window** prompt:

   ```
   >> print(chip1);
   ```

   This will **create** the report file **'MyFirstChip_chipobject.m'** and **open** it in MATLAB's editor mode. In addition to the same information that you can see by doing a content display in the Command Window, the report will contain much more detailed information on the contents of an object. Because of the simplicity of the chip object, you'll notice that the report for a chip object contains no more information than its corresponding Command Window content display. However, as you proceed with the tutorial you will see the difference between an object's report and its Command Window content display beginning with the experiment object.

   A report has two important advantages. The first is that a report will give you much more complete and detailed information on the contents of an SPRTool object. The second is that a report is a file that is permanently stored on disk and is therefore available for viewing and printing at any time you wish.

   To make a **printout** of a report, use the print option in the File menu of the MATLAB editor.

3.2.3 Working with SPRTool Scripts

SPRTool provides you with *scripts* that are intended to make it easy for you to create and manipulate various SPRTool objects. A **script** is nothing more than a file with a **.m** extension that contains a sequence of commands no different from the kind you issued at the command prompt in the previous section. The chip script, for instance, will allow you create a chip object, assign information to its fields, and save it to disk all in a quick and easy manner. We will now use the chip script to demonstrate how we work with SPRTool scripts in general.

1. To avoid any potential confusion, let's clear our workspace by entering the following command at the **Command Window** prompt:

   ```
   >> clear all;
   ```

2. **Opening a Script**
   Now let's begin by opening the chip script ('chip_script.m') to take a look at its internals. To **open** a script, you use the open command followed by the name of the script without the .m extension. Enter the following command at the **Command Window** prompt:

   ```
   >> open chip_script
   ```

   This will open the file 'chip_script.m' in editor mode. This is where you can make changes to the script itself or just review the contents. Each script is divided into sections with an explanation of what you should do in each section.

3. Review the contents of the chip script. For the chip script we have two sections. Section 1 contains the fields that keep track of the chip information like what was coupled to the chip, the coupling level, etc. Section 2 contains the name that we will give our chip object when it is saved to disk.

   You should see that **for the purposes of this tutorial**, sample information for the fields have already been supplied for you. Verify, in particular, that in *Section 1* the identifier assigned to "ChipIdentifier" is '1' and in *Section 2* the file name assigned to "SaveFileName" is 'tutorialchip'.

4. Close the editor window.

5. **Running a Script**
   Let's execute the chip script. To **execute** a script, you simply enter the name of the script without the .m extension. At the **Command Window** prompt enter:

   ```
   >> chip_script
   ```

   This will execute the chip script and store the generated chip object in the workspace under the name **'chipobject'**. You should verify the existence of 'chipobject' in the **Workspace** window. The generated chip object will also be copied to disk. Recall that the field "SaveFileName" in the script was set to 'tutorialchip'. Because the information in that field dictates the file name the object will be saved under (and recalling the file extension '_chipobject.mat' for saved chip objects), you should verify the existence of a file called **'tutorialchip_chipobject.mat'** in the **Current Directory** window.

   We now have two copies of the chip object. One is stored safely on disk while the other one is in our workspace so that we can continue to work with it.

6. **Displaying the Contents of a Chip Object**
   Let's display the contents of our object using the command:

   ```
   >> chipobject
   ```

   The data that you see displayed should match the information specified for the corresponding fields in the chip script. In particular, you should notice that "Chip ID:" has the identifier '1' as specified by the "ChipIdentifier" field in the script. You should also notice that "Object File Name" has the file name specified by the "SaveFileName" field in the script appended with the standard chip object file extension.

7. Now let's look at how we can make changes in our chip script and see those changes re-

flected in the chip object. First let's open the chip script like we did before:

>> **open chip_script**

8. **Modifying a Script**
   Go to the line in Section 1 with the text

   ```
   ChipIdentifier              = '1';
   ```

   on it and change the '1' to 'MyChip1'. (NOTE: Make sure that you leave the single quote marks otherwise you will experience errors when the script is run.) Go to Section 2 and change the file name stored in the "SaveFileName" field from 'tutorialchip' to 'mychip1'.

9. **Saving a Modified Script**
   Save the file by choosing **Save** from the **File** menu of the editor. Close the editor window.

10. Now at the command prompt enter the command:

    >> **chip_script**

    Just as before this will run our chip script and generate both the 'chipobject' workspace variable and a '_chipobject.mat' file. This time though the '_chipobject.mat' file will be called 'mychip1_chipobject.mat', matching the name we gave it in Section 2 of the chip script. You should verify its existence by looking at the **Current Directory** window.

11. Now let's view the object contents with the command:

    >> **chipobject**

    Verify that the chip ID has been changed to 'MyChip1' and that the object file name has been changed to 'mychip1_chipobject.mat'. This reflects the changes we made to the "ChipIdentifier" and the "SaveFileName" fields in the chip script.

With these basic concepts you should be able to get familiar with the scripts that you will need to analyze your data. It should be clear now how one would go about editing the scripts and verifying their results. We encourage you to experiment with changing settings in the scripts and see what they do.

3.2.4 Chip Class Commands Quick Reference

The following is a list of command prompt commands for the chip class. A short description is provided for each command. **NOTE: All commands containing the word chipobject will work ONLY after you have generated chipobject in the workspace by running the chip script.**

- **open chip_script** Will open the chip script for viewing and/or editing.
- **chip_script** Will run the chip script and generate a chip object in the workspace and on disk.
- **chipobject** Will display the contents of the chip object to the command window.
- **print(chipobject)** Will create and open a report containing a listing of the chip object's contents for viewing and/or printing.

3.3 Experiment Class

The experiment object is a type of variable that holds information describing how a particular set of injections was run. The Experiment object also holds the data collected by the SPR machine and the chip object data entered previously using the chip_script.m file. It also possesses the ability to view the sensorgram plots.

3.3.1 Before You Get Started

- Before you proceed with this portion of the tutorial, you need to have completed the Chip Class Tutorial.
- If you closed MATLAB or left the tutorial session by changing the current directory, then you must first open the tutorial session you created previously. Take the following steps to **open an Existing SPRTool Session**.

  Note: The instructions that follow will use **C:\MATLAB6p5\toolbox\local\SPRTool** as a sample **SPRTool Installation Directory**. If your SPRTool Installation Directory is different, then you **MUST substitute it with your own SPRTool Installation Directory**. Also, be sure you enclose the path to your SPRTool Installation Directory in **single quotes**!!

  1. With MATLAB open go to your **SPRTool Installation Directory** by entering the following command at the **Command Window** prompt:

     ```
     >> cd('C:\MATLAB6p5\toolbox\local\SPRTool')
     ```
  2. Start the session manager by entering the following command at the **Command Window** prompt:

     ```
     >> SPR_session
     ```
  3. Click the **"OLD"** button in the window that appears to open an existing session.
  4. A window will appear that displays a list of all existing SPRTool sessions. Highlight the tutorial session that you created previously and click "OK".
  5. You have now opened the existing tutorial session and we are ready to proceed with the experiment class tutorial.

3.3.2 Working with the Experiment Script

This section will cover the basics of working with the Experiment script. The basics of using scripts in general were discussed in the chip class. This section will only deal with the experiment class.

1. **Opening the Experiment Script**
   Open the experiment script 'experiment_script.m'. *Recall that to open a script for viewing and editing, you use the open command followed by the name of the script without the .m extension:*

   ```
   >> open experiment_script
   ```
2. Review the contents. Notice in particular the fields ChipFileName and OriginalDataFilename in Section 1. For the purpose of the tutorial, ChipFileName has been set to 'tutorialchip', the name of the file containing the saved chip object we generated in the chip

class tutorial. This specifies that chip as the one used in the experiment we're dealing with here. Also for the purpose of the tutorial, OriginalDataFilename has been set to 'TutorialData.txt', a sample BIACore data file that has been provided for use with the tutorial. You should verify its existence by looking at the **Current Directory** window.

3. Close the editor window.

4. **Running the Experiment Script**
   Now let's run the experiment script. *Recall that to execute a script, you simply need to enter its name without the .m extension:*

   >> **experiment_script**

   This will generate your experiment object (named **'expobject'**) and also save a copy of it to the file **'tutorialexp_expobject.mat'**. You can verify the former by looking at the **Workspace** window and the latter by looking at the **Current Directory** window.

5. **Displaying the Contents of an Experiment Object**
   Now let's display the contents of the experiment object. *Recall that all we need to do to display the contents of an SPRTool object is to enter the object's name at the command prompt:*

   >> **expobject**

   This will display a summarized view of the experiment object. Notice that no detailed information regarding each run of the experiment is displayed.

6. **Creating, Opening, and Printing an Experiment Object Report**
   Now let's try to create and open an object content report just like we did in the chip tutorial:

   >> **print(expobject)**

   This will generate the report in the file **'tutorialexp_expobject.m'** and open it for viewing. Recalling that a report contains a much more detailed listing of an object's contents, you should see each run in the experiment detailed in the report. *Recall that you can make a printout of the report using the print option in the editor's File menu.*

3.3.3 Experiment Class Commands Quick Reference

The following is a list of command prompt commands for the experiment class. A short description is provided for each command. **NOTE: All commands containing the word expobject will work ONLY after you have generated expobject in the workspace by running the experiment script.**

- **open experiment_script** Will open the experiment script for viewing and/or editing.
- **experiment_script** Will run the experiment script and generate an experiment object in the workspace and on disk.
- **expobject** Will display a summarized view of the contents of the experiment object to the command window.

- **print(expobject)** Will create and open a report containing a detailed listing of the experiment object's contents for viewing and/or printing.

3.3.4 Note on the Experiment Sensorgram GUI Viewer

SPRTool's Experiment class includes a GUI-based viewer that allows you to view plots of the sensorgram data stored in an experiment object. This viewer can **only** be used with an experiment object and is **completely optional** in terms of performing an SPR analyis. For those who are interested in finding out more about this viewer, please see the section entitled *Experiment Sensorgram GUI Viewer* in *Appendix A*.

3.4 Adjust Class

The adjust object is a type of variable that holds the original data and processed data for a group of experiment objects. The adjust object does some basic processing on the data like zeroing the sensorgrams, x-aligning the data points, and doing a background subtraction using one of the channels as the reference.

3.4.1 Before You Get Started

- Before you proceed with this portion of the tutorial, you need to have completed the Experiment Class Tutorial.
- If you closed MATLAB or left the tutorial session by changing the current directory, then you must first open the tutorial session you created previously. See the "Before You Get Started" section of the Experiment Class Tutorial for instructions on how to open an Existing SPRTool Session.

3.4.2 Working with the Adjust Script

This section will cover the basics of working with the Adjust script. The basics of using scripts in general were discussed in the chip class tutorial. This section will only deal with the adjust class details.

1. **Opening the Adjust Script**
   Open the adjust script 'adjust_script.m'. *Recall that to open a script for viewing and editing, you use the open command followed by the name of the script without the .m extension:*

   ```
   >> open adjust_script
   ```

2. Review the contents. Notice in particular the field ExpFileName in Section 1. For the purpose of the tutorial, ExpFileName has been set to 'tutorialexp', the name of the file containing the saved experiment object we generated in the experiment class tutorial. This specifies that experiment as the one whose sensorgram data will be operated on by the adjust object we are about to create.

3. Close the editor window.

4. **Running the Adjust Script**
   Now let's execute the adjust script. *Recall that to execute a script, you simply need to enter its name without the .m extension:*

   ```
   >> adjust_script
   ```

   This will generate our adjust object (named **'adjustobject'**) and also save a copy of it to the file **'tutorialadj_adjustobject.mat'**.

5. **Starting the SPRTool Data Viewer**
   After running the script you can view the data at any time using the SPRTool Data Viewer, a GUI-based tool that allows for easy and flexible viewing of data stored in an SPRTool object. Issue the following command to open the SPRTool Data Viewer:

   ```
   >> SPRToolViewer
   ```

This will open four windows, one of which consists of just a title bar that displays the title "SPRTool".

6. **Loading an Object into the SPRTool Data Viewer**
   Let us now load our adjust object 'tutorialadj_adjustobject.mat' into the SPRTool Data Viewer. Take the following steps:

   (a) Click the **Load New Object** button in the **Loaded Object** window to initiate the object loading process.
   (b) In the **Load Object File** window that appears, click to open the **Files of type:** popup menu and select the **Adjust Object** option. This will display in the window all the adjust object files in your session directory.
   (c) Select the file **'tutorialadj_adjustobject.mat'** and click **Open** to load our adjust object into the SPRTool Data Viewer.

   You will now see an entry for each of the 16 sensorgrams in our adjust object listed in the **Plot Selection** window.

7. **Displaying a Sensorgram Plot with the SPRTool Data Viewer**
   To view the plot of a sensorgram, you simply have to click on that sensorgram's entry in the **Plot Selection** window to display its plot in the **SPRTool Figure Window**. Try the following:

   - Pick any sensorgram you like and click on its entry in the **Plot Selection** window. You should see its plot displayed in the **SPRTool Figure Window**.

8. **Removing a Sensorgram Plot from Display**
   To remove a sensorgram plot from display, you simply have to click on that sensorgram's entry again. Try the following:

   - Click on the displayed sensorgram's entry again. You should now see its plot removed from the **SPRTool Figure Window**.

9. **Displaying Multiple Sensorgram Plots**
   With the SPRTool Data Viewer, you can display multiple sensorgram plots concurrently.

   - Try clicking on several sensorgram entries in the **Plot Selection** window. You should see multiple plots displayed to the **SPRTool Figure Window**, one for each entry you clicked.
   - Now try removing a couple of sensorgram plots from display by clicking their entries again in the **Plot Selection** window. You should see that only those sensorgrams whose entries you've clicked again are removed from display.

10. **Removing All Displayed Plots from Display**
    If you want to remove from display all the currently displayed plots, you simply have to click the **Clear All Plots** button in the lower left corner of the **Plot Selection** window.

    - Click the **Clear All Plots** button. You should now see all remaining plots removed from display.

11. **Viewing Different Types of Data**
    So far we have been viewing the original sensorgram data stored in our adjust object. You can verify this for yourself by noticing that the option *Original data* is currently selected in

the **Choose data type for selection** popup menu at the bottom of the **Plot Selection** window. To view the plots for a different type of data, simply click on the **Choose data type for selection** popup menu and select the data type you want to view.

- Click on the **Choose data type for selection** popup menu and select any data type other than *Original data* which you want to view. You should see your choice as the current selection in the popup menu.
- Click on the entries for a few sensorgrams to display their plots in the **SPRTool Figure Window**.

12. **Displaying Multiple Sensorgram Plots of Different Data Types**
With the SPRTool Data Viewer, you can display multiple sensorgram plots of different data types concurrently. You simply have to use the **Choose data type for selection** popup menu to switch to another data type and select the sensorgrams you want to view like you have been doing. You will see the plots added to the **SPRTool Figure Window**. All displayed plots of previously selected data types will remain.

    - Click on the **Choose data type for selection** popup menu and select a data type different from the currently selected one.
    - Add a few plots of the newly selected data type to the **SPRTool Figure Window** by clicking on some sensorgram entries in the **Plot Selection** window. You should see their plots displayed in addition to whatever was already displayed.

13. **Closing the SPRTool Data Viewer**
To close the SPRTool Data Viewer, you can close each of its four windows individually **or** you can just close the title bar window (i.e. the one that displays the title “SPRTool”. All windows will be closed automatically when you close the title bar window.

    - Experiment some more with the SPRTool Data Viewer if you like. After you finish, close the SPRTool Data Viewer by closing the large background window.

14. **A Note on the SPRTool Data Viewer**
In this tutorial you have been shown a fairly complete demonstration of the capabilities of the SPRTool Data Viewer. However, there are a few **important capabilities** that have been left out of the tutorial. The viewer, for instance, has a zooming feature and the ability to generate an object content report. There are also ways that the user can easily select multiple consecutive sensorgram entries. For information on these capabilities and a more complete discussion of the features you have been shown, please see the chapter entitled **SPRTool Data Viewer**.

15. **Displaying the Contents of an Adjust Object**
Now let’s display the contents of our adjust object. *Recall that all we need to do to display the contents of an SPRTool object is to enter the object’s name at the command prompt:*

    ```
    >> adjustobject
    ```

    This will display a summarized view of the adjust object to the Command Window. Notice that no detailed sensorgram information is displayed.

16. **Creating, Opening, and Printing an Adjust Object Report**
To see detailed information for each sensorgram contained in our adjust object, we must create an object report. Try the following command:

```
>> print(adjustobject);
```

This will generate a report file called 'tutorialadj_adjustobject.m' and open it for viewing and printing. You should see the details for each sensorgram listed in the report. *Recall that you can make a printout of the report using the print option in the editor's File menu.*

3.4.3 Adjust Class Commands Quick Reference

The following is a list of command prompt commands for the adjust class. A short description is provided for each command. **NOTE: All commands containing the word adjustobject will work ONLY after you have generated adjustobject in the workspace by running the adjust script.**

- **open adjust_script** Will open the adjust script for viewing and/or editing.
- **adjust_script** Will run the adjust script and generate an adjust object in the workspace and on disk.
- **adjustobject** Will display a summarized view of the contents of the adjust object to the command window.
- **print(adjustobject)** Will create and open a report containing a detailed listing of the adjust object's contents for viewing and/or printing.

3.5 Equilibrium Class

The equilibrium object is a type of variable that holds the equilibrium data for a set of equilibrium experiments. Each equilibrium experiment is performed on a subset of sensorgrams of an associated adjust object that correspond to the same chip, analyte, and trace. The equilibrium object does some processing like cutting out the portion of a sensorgram needed for equilibrium analysis, estimating the equilibrium value of a sensorgram based on its cut-out portion, and computing for each equilibrium experiment estimated values like the equilibrium constants KA and KD and the maximum coupling level Rmax.

3.5.1 Before You Get Started

- Before you proceed with this portion of the tutorial, you need to have completed the Adjust Class Tutorial.
- If you closed MATLAB or left the tutorial session by changing the current directory, then you must first open the tutorial session you created before. See the "Before You Get Started" section of the Experiment Class Tutorial for instructions on how to open an Existing SPRTool Session.

3.5.2 Working with the Equilibrium Script

The equilibrium script 'equilibrium_script.m' is the script that will generate the basic equilibrium object for you. The equilibrium script performs some basic processing on the sensorgram data such as cutting out the parts of the sensorgrams needed for equilibrium analysis and estimating equilibrium values from the cut outs.

1. **Opening the Equilibrium Script**
   Open the equilibrium script 'equilibrium_script.m'. *Recall that to open a script for viewing and editing, you use the open command followed by the name of the script without the .m extension:*

   ```
   >> open equilibrium_script
   ```

2. Review the contents. Notice in particular the field AdjFileName in Section 1. For the purpose of the tutorial, AdjFileName has been set to 'tutorialadj', the name of the file containing the saved adjust object we generated in the adjust class tutorial. This specifies that adjust object as the one whose sensorgram data will be operated on by the equilibrium object we are about to create.

3. Close the editor window.

4. **Running the Equilibrium Script**
   Now let's execute the equilibrium script. *Recall that to execute a script, you simply need to enter its name without the .m extension:*

   ```
   >> equilibrium_script
   ```

   This will generate our equilibrium object (named **'equilobject'**) and also save a copy of it to the file **'tutorialequ_equobject.mat'**.

5. **Displaying the Contents of an Equilibrium Object**
   Now let's display the contents of our equilibrium object. *Recall that all we need to do to*

display the contents of an SPRTool object is to enter the object's name at the command prompt:

>> **equilobject**

This will display a summarized view of the equilibrium object to the Command Window. Notice that no detailed sensorgram and "equilibrium experiment" information is displayed.

6. **Creating and Opening an Equilibrium Object Report**
To see detailed information for each sensorgram and "equilibrium experiment" contained in our equilibrium object, we must create an object report. Try the following command:

   >> **print(equilobject);**

   This will generate a report file called 'tutorialequ_equobject.m' and open it for viewing and printing. You should see the details for each sensorgram and "equilibrium experiment" listed in the report. *Recall that you can make a printout of the report using the print option in the editor's File menu.*

3.5.3 Equilibrium Class Commands Quick Reference

The following is a list of command prompt commands for the equilibrium class. A short description is provided for each command. **NOTE: All commands containing the word equilobject will work ONLY after you have generated equilobject in the workspace by running the equilibrium script.**

- **open equilibrium_script** Will open the equilibrium script for viewing and/or editing.
- **equilibrium_script** Will run the equilibrium script and generate an equilibrium object in the workspace and on disk.
- **equilobject** Will display a summarized view of the contents of the equilibrium object to the command window.
- **print(equilobject)** Will create and open a report containing a detailed listing of the equilibrium object's contents for viewing and/or printing.

3.6 What's Next After the Setup Scripts

The tutorial in this chapter has demonstrated the functionalities of the chip, experiment, adjust, and equilibrium scripts. These four scripts are referred to as *setup* scripts because they generate the basic SPRTool objects required for further analysis of the BIAcore data.

After you have run the setup scripts in an SPRTool session, you have the option to perform an equilibrium and/or a kinetic analysis on your data with the four *analysis* scripts included in SPRTool. Two of these analysis scripts are used for equilibrium analysis, and the other two are used for kinetic analysis. The analysis scripts are designed to be modified and executed the exact same way as the setup scripts. However, please remember that you must have successfully executed all the setup scripts before you can successfully run an analysis script. (Note: There is an exception to this rule - If you only want to do a kinetic analysis, then there's no need for you to run the equilibrium script. The equilibrium script is only necessary if you want to do an equilibrium analysis.)

For detailed information concerning the analysis as well as the setup scripts, please see the chapter entitled **SPRTool Scripts**. Also, if you want to see a quick demonstration of each analysis script, please go to the subsection below.

3.6.1 Analysis Scripts Demonstration

This section will provide a demonstration of what the SPRTool analysis scripts can do. By default the analysis scripts are configured to read in and analyze sample SPRTool objects that contain simulated data. These sample SPRTool objects are included with each tutorial session and are specifically designed to illustrate the capabilities of their corresponding analysis scripts. To see what an analysis script can do, you simply have to run the script without any modifications. This section will take you through the execution of each analysis script without going over any details concerning the scripts. To learn more about the analysis scripts and the functionalities they provide, please see the chapter entitled **SPRTool Scripts**.

1. If you closed MATLAB or left the tutorial session by changing the current directory, then you must first open the tutorial session you created previously. See the "Before You Get Started" section of the Experiment Class Tutorial for instructions on how to open an Existing SPRTool Session.

2. **Running the Equilibrium Scatchard Script**
   Let's begin by executing the equilibrium scatchard script, which performs equilibrium analysis using the traditional Scatchard technique:

   >> **equilibrium_scatchard_script**

   You will see a Scatchard plot displayed with a title that shows the estimated values for K_d, K_a, and R_{max}. Close the plot figure after you're done viewing.

3. **Running the Equilibrium Ligand Activity Loss Script**
   Next let's execute the equilibrium ligand activity loss script, which performs equilibrium analysis in a way that compensates for the loss of ligand activity:

   >> **equilibrium_ligandactivityloss_script**

   You will see an exponentially fitted R_{eq} vs. $Time$ plot displayed with a title that shows the estimated values for K_d, K_a, and a few other parameters. Close the plot figure after you're done viewing. (By default this script is configured to read in and analyze the sample object 'SimulatedSingleExp_equobject.mat'. If you want, you can try modifying the script so that it analyzes the other three sample objects already listed in Section 1 of the script. To do so, you simply need to edit Section 1 and Section 4 of the script accordingly. Just make sure that the model you choose in Section 4 corresponds to the sample object you choose in Section 1.)

4. **Running the Kinetic Hankel Script**
   Now let's execute the kinetic Hankel script, which estimates the kinetic parameters of a sensorgram with a subspace algorithm:

   ```
   >> kinetic_hankel_script
   ```

   You will see two figures displayed: a plot of the estimation by the independent interactions model and a residual plot of the estimation by the independent interactions model. Close the figures after you're done viewing.

5. **Running the Kinetic Optimization Script**
   Finally, let's execute the kinetic optimization script, which calculates the association rate constant, dissociation rate constant, and the R_{max} value for one sensorgram or a group of sensorgrams:

   ```
   >> kinetic_optimization_script
   ```

   The script will take a while to run (around 10 minutes or so). During the course of its execution, you will see a figure that continuously displays the intermediate results of the curve-fitting algorithm. When the script stops running, you will see four figures displayed: a plot of the final fit produced by the analysis and a residual plot for each of three sensorgrams included in the analysis. Close the figures after you're done viewing.
   (By default this script is configured to read in and analyze the sample object 'sample_simulated_independentinteraction_adjustobject.mat'. If you want, you can try modifying the script so that it analyzes the other sample object already listed in Section 1 of the script. To do so, you simply need to edit Section 1 and Section 5 of the script accordingly. Just make sure that the model you choose in Section 5 corresponds to the sample object you choose in Section 1.)

Chapter 4

SPRTool Scripts

This chapter will provide a discussion for each of the scripts provided with SPRTool. The discussion for each script will include its description, an explanation of the fields that require user input, and for those who are interested in the inner workings of the script, the classes that are required and *directly* referenced by the script. For a detailed look at each of those classes, please refer to the *SPRTool Reference Manual.*

4.1 Two Basic Types of SPRTool Scripts

Every script provided with SPRTool (with the exception of simulation scripts that are to be discussed in the last section of this chapter) can be placed in one of two categories. An SPRTool script is either a **setup script** or an **analysis script**. The setup scripts are responsible for creating the various SPRTool objects used to keep track of the data from your BIAcore experiment and prepare the data for use by the analysis scripts. The purpose of the analysis scripts, on the other hand, is to perform analysis on your Biacore data and produce results like the equilibrium constants KA and KD.

4.2 Setup Scripts

The four subsections that follow will discuss the four setup scripts provided with SPRTool. These four scripts are the ones that were covered in the Tutorial.

4.2.1 Chip Script

Filename chip_script.m

Description Creates a chip object for keeping track of information pertaining to the chip used in a BIAcore experiment. The information includes things like the name of the person who coupled the chip and the name of the ligand coupled to a channel in the chip.

User Input Fields

- **Section 1:**

  ChipIdentifier - The chip ID given as a string.
  DateCoupled - The date the chip was coupled in (DD/MM/YYYY) format.
  WhoCoupled - The person who did the coupling.
  Comments - Any additional comments about the chip and the coupling.
  ChannelxCoupled - What a specific flowcell was coupled with. The ‘x’ can be a number from one to four.
  ChannelxCouplingLevel and ChannelxCouplingLevelUnits - The coupling level of the ligand to flowcell ‘x’ of the chip. The standard unit of measure for the coupling level is in resonance units(RU). The ‘x’ can be a number from one to four.

  Note: The SPRTool program was developed using a 4 channel BIAcore machine. If you have a smaller channel machine then just put ’NA’ in all unused channel descriptions. Example: You have two channels in your machine. In all references to channel numbers higher than two you would put the string ’NA’ beside it.

- **Section 2:**

  SaveFileName - The file name for your chip object. The extension ‘_chipobject.mat’ will be appended to the name you supply for you.

Required Classes Chip

4.2.2 Experiment Script

Filename experiment_script.m

Description Creates an experiment object for storing the sensorgram data generated in a BIAcore experiment. In addition, the generated experiment object keeps track of important information like the name of the analyte and the flowrate used in each run of the experiment.

User Input Fields

- **Section 1:**

  ChipFileName - The name of the saved chip object file. It should always be the **same** as the name specified in Section 2 of the chip script.
  OriginalDataFilename - The name of the file with the SPR data.

- **Section 2:**

  MachineType - Type of machine used in the experiment.
  DataFileType - Type of the data file you specified in Section 1. It must be either 'ascii' or 'excel'.
  ControlSoftwareVersion - Version number of the control software you used.

- **Section 3:**

  ExperimentDate - The date of the experiment.
  ExperimentPurpose - The purpose of the experiment.
  ExperimentOperator - The name of the person who conducted the experiment.
  ExperimentComments - Things to note concerning the experiment.

- **Section 4:**

  DisplayData - Whether or not you wish to see a GUI display of the data after the script has finished executing. If you specify **'no'** then the data display will be skipped. If you specify **'yes'** then the Experiment Sensorgram GUI Viewer will be opened so that you may view the SPR plots. A detailed discussion on the Experiment Sensorgram GUI Viewer can be found in Appendix A.

- **Section 5:**

  The idea here is that you need to supply, for each run, an information structure so that later on in the use of this program your data will make sense. By supplying the correct information for each run you can make use of certain features of the software that would be useless without it. One example of this would be the **Trace Details** button on the Experiment Sensorgram GUI Viewer. Without the proper information filled out for each run this feature becomes pretty worthless and possibly a problem down the line.

  For **each** run of your experiment, you need to specify the correct information for the following fields (i.e. **If you have n runs in your experiment, you should fill out the following fields for Run(1) through Run(n).**):

  Flowrate and FlowrateUnits - The rate of analyte flow through the channel. The standard unit of measure is in microliters per minute(uL/min). This is the same

parameter that you would've set in your BIA program.
WhatInjected - The name of the analyte used.
Concentration and ConcentrationUnits - The concentration of the analyte used. The standard unit of measure is in nanomolars(nM).
UniformlySampled - You have a choice between 'yes' or 'no'. This specifies if the data collection was done uniformly or not.
SamplingRate and SamplingRateUnits - The rate the data was acquired from the machine. The standard values are 0.1, 1, 2, 10 and the standard unit of measure is in Hertz(Hz). The sampling rate is nothing more than the inverse of the sampling interval (sampling rate = 1/sampling interval).
SamplingInterval and SamplingIntervalUnits - The time between samples taken from the flowcell. The standard unit of measure is in seconds (s). This is nothing more that the inverse of the sampling rate (sampling interval = 1/sampling rate).
Temperature and TemperatureUnits - The temperature of the flow. The standard unit of measure is in Kelvin (KV).

- **Section 6:**

  SaveFileName - File name for Experiment object. The extension '_expobject.mat' will be appended to the name you supply for you.

Required Classes Chip, Experiment

4.2.3 Adjust Script

Filename adjust_script.m

Description Creates an adjust object that performs routine adjustments on the sensorgram data belonging to one or more experiment objects. The routine adjustments include things like zero adjustment and background subtraction. The resulting adjusted data are stored in the generated adjust object.

User Input Fields

- **Section 1:**

  ExpFileName - The name(s) of the saved experiment object file(s). The name(s) should always be the **same** as the name(s) specified in Section 6 of the experiment script.

  You can specify as many experiment objects as you want to include for a single adjust object. ExpFileName1 specifies the first experiment object to include in the adjust object. **If you want to include n experiments for the adjust object, you must supply file names for ExpFileName{1} through ExpFileName{n}.** The script shows a demonstration of how one could include a second experiment object in the adjust object.

- **Section 2:**

  AdjustCreator - The name of the creator of the adjust object.
  AdjustComments - Things to note concerning the adjust object.

- **Section 3:**

  FirstTimeRunThisScript - Whether or not the script is to be run for the first time. Make sure you set it to **'yes'** the first time you run the script and to **'no'** for all subsequent executions.

- **Section 4:**

  SelectionMethod - Method of choosing which sensorgrams to include in the data processing steps. You have two choices:

  - **'SelectAll'** - This option will include all sensorgrams in the data processing steps.
  - **'SelectPartialList'** - This option gives you the flexibility to include any sensorgrams in the data processing steps by specifying the *selectorvector* below.

  selectorvector - This field allows you to select exactly which sensorgrams to **exclude** from the data processing steps. You must fill out this field only if you chose the **'SelectPartialList'** option for the selection method. We recommend that you use the SPRTool Viewer to determine which sensorgrams to exclude from further processing.

  You select which sensorgrams you do **NOT** want to include by setting the **selectorvector** with the proper index equal to one. For example, if you wanted to **NOT** include sensogram 10 then you would just add the line 'selectorvector(10)=1;' to Section 4.

As you can see in the script the sensorgrams for run number one have been excluded. This is just to demonstrate how it can be done. Sensorgram fifteen will not be included because it had an error in the acquisition.

- **Section 5:**

  Xcoordinate - This field lets the user specify which point on the x axis will be used to zero out the sensorgram plots. The idea is that you pick a point on the x axis that has the lowest y value before the association curve. The program will then use that x axis point to set the corresponding y data point to zero. You can specify the zeroing x axis point in one of two ways:

  - You can specify **ONE** x axis point. In this case **all** sensorgrams will be zero adjusted using that same x axis point.
  - **OR** You can specify a zeroing x axis point for **each** sensorgram by providing a **vector** of values. For example, to specify 16 x axis points for 16 sensorgrams, you would provide a comma-separated list of 16 values enclosed in square brackets (i.e. []). The nth value in the list would correspond to the x axis point you want to use to zero adjust sensorgram n.

- **Section 6:**

  backgroundsubtractchannel - The number of the channel/trace that will be used as the background for background subtraction.

- **Section 7:**

  This section describes the software's ability to do an x alignment of the sensorgrams. The problem with the standard SPR measurement is that the flow of analyte across each channel does not happen in parallel. The analyte must flow from channel 1, to 2, to 3, etc. This will cause the association and dissociation curves to be slightly off from one another with respect to time. This x alignment option will try to adjust for this time lag between channels.

  Your job is to select the best points on the sensorgrams so that the program can try to do its adjustments. Just select the point right before the association curve for each trace/channel. It is recommended that you use the SPRTool Viewer to **zoom in very close** to the sensorgram to enter the coordinate as precisely as possible. You are given the choice of doing the alignment on the association or dissociation curves. If trying one does not seem to give optimal results then try the other. The interpolation method used for now is linear.

  The fields in this section are as follows:

  PerformXAlign - Whether or not you wish to do an x alignment of the sensorgrams. The options are **'yes'** or **'no'**.
  FlowCell_n_Association_X_value - the x-coordinate of the last point before association for one sensorgram from flow cell n. The format to use is [**X_value , Sensorgram_Number**] where X_value is the x-coordinate of the last point before association and Sensorgram_Number is the index of the flow cell n sensorgram from which you obtained the x-coordinate value.
  FlowCell_n_Dissociation_X_value - the x-coordinate of the last point before dis-

sociation for one sensorgram from flow cell n. The format is the same as for FlowCell_n_Association_X_value except that X_value is the x-coordinate of the last point before **dissociation**.
InterpMethod - The type of interpolation done on the data. Your only choice here is **'linear'**.
AlignToPhase - The slope of the graph to use for x alignment. The available options are **'Association'** and **'Dissociation'**.

- **Section 8:**

  SaveFileName - File name for Adjust object. The extension '_adjustobject.mat' will be appended to the name you supply for you.

Required Classes Experiment, Adjust

4.2.4 Equilibrium Script

Filename equilibrium_script.m

Description Creates an equilibrium object that prepares the sensorgram data in an associated adjust object for equilibrium analysis. The preliminary processing of the data consists of identifying the equilibrium segment of each sensorgram and computing an estimated equilibrium value from the equilibrium segment. The equilibrium segments and estimated equilibrium values are stored in the generated equilibrium object.

User Input Fields

- **Section 1:**

  AdjFileName - The name of the saved adjust object file. The name should always be the **same** as the name specified in Section 8 of the adjust script.

- **Section 2:**

  EquilCreator - The name of the creator of the equilibrium object.
  EquilComments - Things to note concerning the equilibrium object.

- **Section 3:**

  SelectionMethod - Method of choosing which sensorgrams to include in the data processing steps. You have three choices:

  - **'SelectAll'** - This option will include all sensorgrams in the data processing steps.
  - **'SelectPartialList'** - This option gives you the flexibility to include any sensorgrams in the data processing steps by specifying the *selectorvector* below.
  - **'PreviousSelectList'** - This option will include all sensorgrams that were previously selected in the adjust object specified in Section 1.

  selectorvector - This field allows you to select exactly which sensorgrams to **exclude** from the data processing steps. You must fill out this field only if you chose the **'SelectPartialList'** option for the selection method.

  You select which sensorgrams you do **NOT** want to include by setting the **selectorvector** with the proper index equal to one. For example, if you wanted to **NOT** include sensogram 10 then you would just add the line 'selectorvector(10)=1;' to Section 3.

  As you can see in the script the sensorgrams for run number one have been excluded. This is just to demonstrate how it can be done. Sensorgram fifteen will not be included because it had an error in the acquisition.

- **Section 4:**

  Xcoordinates - This field lets the user specify the x-coordinates that define the interval to be cut out from the sensorgrams for equilibrium analysis. The interval provided by the user should specify the data segment where the sensorgram is in equilibrium. You can specify the cut out interval in one of two ways:

  - You can specify **ONE** interval. In this case the specified interval will apply to **all** selected sensorgrams. The format to use is **[x1,x2]** where x1 is the x-coordinate of the start point of the interval and x2 is the x-coordinate of the end point of the interval. You should use the SPRTool Viewer to view and zoom in on the sensorgrams of the adjust object very closely in order to select the x-coordinates with precision. The interval you choose in this case should ideally encompass the equilibrium segments of all selected sensorgrams.

  - **OR** You can specify an interval for **each** sensorgram by providing a **vector** of intervals. For example, to specify 16 intervals for 16 sensorgrams, you would provide a semicolon-separated list of 16 x-coordinate pairs enclosed in square brackets (i.e. []). The nth pair in the list would correspond to the cut out interval for sensorgram n. An example is given in the script to demonstrate the format for specifying the list. Again you should use the SPRTool Viewer to view and zoom in on the sensorgrams of the adjust object in order to select the x-coordinates with precision.

- **Section 5:**

  SaveFileName - File name for Equilibrium object. The extension '_equobject.mat' will be appended to the name you supply for you.

Required Classes Adjust, Equilibrium

4.3 Analysis Scripts

The four subsections that follow will discuss the four analyis scripts provided with SPRTool.

4.3.1 Equilibrium Scatchard Script

Filename equilibrium_scatchard_script.m

Description Performs equilibrium analysis on each "equilibrium experiment" contained in a given equilibrium object using the traditional Scatchard technique. The Scatchard technique uses the linear graph of $\frac{R_{eq}}{C}$ versus R_{eq}, where R_{eq} is the equilibrium estimate and C is the concentration, to estimate the equilibrium constants. Each "equilibrium experiment" corresponds to a subset of sensorgrams in the equilibrium object that are associated with the same chip, analyte, and trace. Estimates for the equilibrium constants K_a and K_d and the maximum coupling level R_{max} are computed for each such "equilibrium experiment". In addition to storing the results of the analysis in the given equilibrium object, the script displays the Scatchard plot for each "equilibrium experiment" in a separate figure with the computed results shown in the title. The equilibrium Scatchard script does *not* generate any objects. It only analyzes and stores its computed results in an existing equilibrium object.

User Input Fields

- **Section 1:**

  EquFileName - The name of the saved equilibrium object file. The name should always be the **same** as the name specified in Section 5 of the equilibrium script.

- **Section 2:**

  SelectionMethod - Method of choosing which sensorgrams to include in the equilibrium analysis. You have three choices:

  – **'SelectAll'** - This option will include all sensorgrams in the equilibrium analysis steps.
  – **'SelectPartialList'** - This option gives you the flexibility to include any sensorgrams in the equilibrium analysis steps by specifying the *selectorvector* below.
  – **'PreviousSelectList'** - This option will include all sensorgrams that were previously selected in the equilibrium object specified in Section 1.

  selectorvector - This field allows you to select exactly which sensorgrams to **exclude** from the equilibrium analysis steps. You must fill out this field only if you chose the **'SelectPartialList'** option for the selection method.

  You select which sensorgrams you do **NOT** want to include by setting the **selectorvector** with the proper index equal to one. For example, if you wanted to **NOT** include sensogram 10 then you would just add the line 'selectorvector(10)=1;' to Section 2.

  As you can see in the script the sensorgrams for run number one have been excluded. This is just to demonstrate how it can be done. Sensorgram fifteen will not be included because it had an error in the acquisition.

- **Section 3:**

  printoption - Whether or not to print the results (Scatchard plots) of the equilibrium analysis. The options are **'Yes'** or **'No'**.

Required Classes Equilibrium

4.3.2 Equilibrium Ligand Activity Loss Script

Filename equilibrium_ligandactivityloss_script.m

Description Performs equilibrium analysis on each "equilibrium experiment" contained in a given equilibrium object in a way that compensates for the loss of ligand activity. The compensation is accomplished by modeling the maximum analyte binding capacity of the chip (R_{max}) to be a decaying function of time. The script makes available four different decay models, the details of which can be found in the following paper:

Raimund J. Ober and E. Sally Ward. Compensation for loss of ligand activity in surface plasmon resonance experiments. Analytical Biochemistry, 306:228-236, 2002.

Each "equilibrium experiment" corresponds to a subset of sensorgrams in the equilibrium object that are associated with the same chip, analyte, and trace. Estimates for the equilibrium constants K_a and K_d are computed for each such "equilibrium experiment" using the decay model chosen by the user. In addition to storing the results of the analysis in the given equilibrium object, the script displays the curve-fitted $R_{eq} vs. Time$ plot for each "equilibrium experiment" in a separate figure with the computed results shown in the title. The equilibrium ligand activity loss script does *not* generate any objects. It only analyzes and stores its computed results in an existing equilibrium object.

User Input Fields

- **Section 1:**

  EquFileName - The name of the saved equilibrium object file. The name should always be the **same** as the name specified in Section 5 of the equilibrium script.

- **Section 2:**

  SelectionMethod - Method of choosing which sensorgrams to include in the equilibrium analysis. You have three choices:

  - **'SelectAll'** - This option will include all sensorgrams in the equilibrium analysis steps.
  - **'SelectPartialList'** - This option gives you the flexibility to include any sensorgrams in the equilibrium analysis steps by specifying the *selectorvector* below.
  - **'PreviousSelectList'** - This option will include all sensorgrams that were previously selected in the equilibrium object specified in Section 1.

  selectorvector - This field allows you to select exactly which sensorgrams to **exclude** from the equilibrium analysis steps. You must fill out this field only if you chose the **'SelectPartialList'** option for the selection method.

  You select which sensorgrams you do **NOT** want to include by setting the **selectorvector** with the proper index equal to one. For example, if you wanted to **NOT** include sensogram 10 then you would just add the line 'selectorvector(10)=1;' to Section 2.

  As you can see in the script the sensorgrams for run number one have been excluded. This is just to demonstrate how it can be done. Sensorgram fifteen will not be included because it had an error in the acquisition.

- **Section 3:**

  printoption - Whether or not to print the results (curve-fitted $R_{eq} vs.Time$ plots) of the equilibrium analysis. The options are **'Yes'** or **'No'**.

Section 4:

EstimateTechnique - The decay model to use to describe the loss of ligand activity in the estimation of the equilibrium constants K_a and K_d. The available options are **'SingleExponent'**, **'DoubleExponent'**, and **'SingleExponentNoConstant'**, and **'Constant'**.
InitialConditions - The initial conditions to use for the various parameters that are to be estimated. The values expected for this field are model dependent. You have two ways to specify the initial conditions:

- You can choose to use the **default** initial conditions. To do so you simply have to set **InitialConditions** = []. You can use this option with any of the four available decay models.
- **OR** you can supply your own **customized** initial conditions. In this case, if you chose the
  - **Single Exponential** decay model, then you have to specify the initial conditions in the format [**Beta Alpha Gamma Kd**] where Beta, Alpha, Gamma, and Kd are the parameters required by the model. Make sure you provide the values in their appropriate units as specified in the script.
  - **Double Exponential** decay model, then you have to specify the initial conditions in the format [**Beta1 Alpha1 Beta2 Alpha2 Kd**] where Beta1, Alpha1, Beta2, Alpha2, and Kd are the parameters required by the model. Make sure you provide the values in their appropriate units as specified in the script.
  - **Single Exponential Without Constant** decay model, then you have to specify the initial conditions in the format [**Beta Alpha Kd**] where Beta, Alpha, and Kd are the parameters required by the model. Make sure you provide the values in their appropriate units as specified in the script.
  - **'Constant Exponential'** decay model, then you have to specify the initial conditions in the format [**Delta Kd**] where Delta and Kd are the parameters required by the model. Make sure you provide the values in their appropriate units as specified in the script.
- **Section 5:**

  ExperimentTime(n).TimeIndex - The experiment time for the equilibrium estimate value of each sensorgram. This is an optional field that expects values to be given in a very specific format. Please see the script for a detailed description of this field and instructions on how to specify the values it expects.

Required Classes Equilibrium

4.3.3 Kinetic Hankel Script

Filename kinetic_hankel_script.m

Description Estimates the kinetic parameters of a BIAcore sensorgram with a subspace algorithm, the details of which are discussed in the following paper:

R. J. Ober, J. Caves and E. S. Ward. Analysis of exponential data using a non-iterative technique: application to surface plasmon experiments. Analytical Biochemistry, 312:57-65,2003.

In this script, kinetic parameters such as k_{on}, R_{eq}, k_{off}, and R_0 are calculated for a single curve and exponential functions are then used to fit the curve. The state space realization of the curve is first estimated via the subspace algorithm and the kinetic parameters are computed from the resulting state space realizations. The curve is then modeled as the sum of multiple exponential functions, each of which represents an independent interaction. The estimate of the curve and the residuals are displayed in two plots with titles showing the estimated values for the kinetic parameters.

User Input Fields

- **Section 1:**

  AdjFileName - The name of the saved adjust object file. The name should always be the **same** as the name specified in Section 8 of the adjust script.

- **Section 2:**

  KineticCreator - Name of person performing the kinetic analysis.
  KineticComments - Any comments regarding the analysis.
  KineticDataDescription - Description of kinetic data.

- **Section 3:**

  selectorvector - The sensorgram(s) that you wish to perform the kinetic analysis on. To select sensorgram n, use the statement "selectorvector(n)=1;".

- **Section 4:**

  AssociationStartTime - Start time of the Association Phase of the sensorgrams to be analyzed. The value you specify will be uniformly applied to all the sensorgrams you have selected.
  AssociationEndTime - End time of the Association Phase of the sensorgrams to be analyzed. The value you specify will be uniformly applied to all the sensorgrams you have selected.
  DissociationEndTime - End time of the Dissociation Phase of the sensorgrams you have selected. The value you specify will be uniformly applied to all the sensorgrams you have selected.
  AssociationSegmentTimeDomain - Segment of the Association Phase you want to use for the analysis. The format to use is [time1, time2] where time1 and time2 are the start and end times of the segment, respectively.
  DissociationSegmentTimeDomain - Segment of the Dissociation Phase you want to use for the analysis. The format to use is [time1, time2] where time1 and time2 are the start and end times of the segment, respectively.

- **Section 5:**

  AnalysisType - Type of analysis you wish to perform. The available options are **'Association'** and **'Dissociation'**.
- **Section 6:**

  NumberOfInteractions - Assumed number of interactions, i.e. the number of interactions for which kinetic constants will be determined.
- **Section 7:**

  DisplayDataObject - Whether to display the data object that contains the data for the selected sensorgrams to screen. Your options are **'yes'** or **'no'**.
  PlotSingularValues - Whether to plot the singular values. The options are **'yes'** or **'no'**.
  plotRealizationEstimate - Whether to plot the realization estimate. The options are **'yes'** or **'no'**.

Required Classes Adjust, DataClass, BalanceDOS, DiscreteOutputSystem, IndependentInteractionsModel

4.3.4 Kinetic Optimization Script

Filename kinetic_optimization_script.m

Description Calculates the association rate constant, the dissociation rate constant, and the R_{max} value for one sensorgram or a group of sensorgrams. The two models used to simulate the data are the Independent Interactions Model and the Mass Transport Model.
For the Independent Interactions Model the script gives the choice of analyzing one sensorgram to calculate the association or the dissociation constant separately. The script also provides the choice to calculate them both at the same time using Global Analysis techniques. With the global case more than one sensorgram can be included. All included sensorgrams must be from the same channel and have the same proteins but the concentrations can vary.
For the Mass Transport Model, Global Analysis is the only available option. The same rules apply for the Mass Transport Model analysis as they did for the Independent Interactions Model when doing Global Analysis. That is, more than one sensorgram can be included but they must be from the same channel of the machine and have the same proteins. The concentration can and should vary between the different sensorgrams. In addition to the rate constants, the value for parameter 'ktr' is returned by the analysis. The value of ktr helps to determine if the data you are analyzing has any mass transport effect associated with it. The closer this value approaches zero the more mass transport you have.

User Input Fields

- **Section 1:**

  AdjFileName - The name of the saved adjust object file. The name should always be the **same** as the name specified in Section 8 of the adjust script.

- **Section 2:**

  KineticCreator - Name of person performing the kinetic analysis.
  KineticComments - Any comments regarding the analysis.
  KineticDataDescription - Description of kinetic data.

- **Section 3:**

  selectorvector - The sensorgram(s) that you wish to perform the kinetic analysis on. To select sensorgram n, use the statement "selectorvector(n)=1;".

- **Section 4:**

  AssociationStartTime - Start time of the Association Phase of the sensorgrams to be analyzed. The value you specify will be uniformly applied to all the sensorgrams you have selected.
  AssociationEndTime - End time of the Association Phase of the sensorgrams to be analyzed. The value you specify will be uniformly applied to all the sensorgrams you have selected.
  DissociationEndTime - End time of the Dissociation Phase of the sensorgrams you have selected. The value you specify will be uniformly applied to all the sensorgrams you have selected.
  AssociationSegmentTimeDomain - Segment of the Association Phase you want to use for the analysis. The format to use is [time1, time2] where time1 and time2

are the start and end times of the segment, respectively.
DissociationSegmentTimeDomain - Segment of the Dissociation Phase you want to use for the analysis. The format to use is [time1, time2] where time1 and time2 are the start and end times of the segment, respectively.

- **Section 5:**

  ModelType - The type of Model you wish to use for the analysis. Your options are **'Independent Interaction'** and **'Mass Transport'**.

- **Section 6:**

  AnalysisType - The type of analysis you wish to perform. Your options are **'Association'**, **'Dissociation'**, and **'FullCurve'**. If you have selected the Mass Transport model in Section 5, the only valid option here is 'FullCurve'.

- **Section 7:**

  op_.InitialConditions - Initial condition parameters to be used in the analysis. Please see the script for the exact format to use to specify the parameter values for each valid model and analysis type combination.

- **Section 8:**

  TypicalValue - Common values for the various parameters. This field allows you to rescale all the different parameters to help the optimization process achieve more accurate estimates. This field expects the exact same format as the InitialConditions field. Please see the script for the exact format to use to specify the parameter values for each valid model and analysis type combination.
  op.Options.MaxIter - The maximum number of iterations the optimization algorthim executes.

- **Section 9:**

  DisplayDataObject - Whether to display the data object that contains the data for the selected sensorgrams to screen. Your options are **'yes'** or **'no'**.
  PlotIntermediateResults - Whether you want to see plots of intermediate optimization results displayed. Your options are **'yes'** or **'no'**.
  SaveSomeMemory - Whether you want to save some MATLAB workspace memory. Your options are **'yes'** or **'no'**. (Unless you are working with a very large data set, MATLAB normally has sufficient memory to work with. Therefore, this field is set to 'no' by default.)

Required Classes Adjust, DataClass, OptimizeClass, Model, MassTransportCompModel, IndependentInteractionsModel

4.4 Simulation Scripts

In addition to the setup and analysis scripts, SPRTool provides the user with four **simulation scripts**. The simulation scripts are used to generate SPRTool objects containing simulated data that can be analyzed with the analysis scripts. The simulation scripts are provided with SPRTool for two main purposes. First, the objects they generate can be specified as input to the analysis scripts to demonstrate the functionalities and capabilities of the analysis scripts and second, they can be modified like any other script to generate your own simulated data for whatever purpose. Like the setup and analysis scripts, the simulation scripts are included in every SPRTool session you create, tutorial or regular. Each simulation script is already filled in with valid values such that sample objects containing simulated data can be generated readily by running the script without modification. With a tutorial session, these sample objects are already included such that you can run the analysis scripts directly to analyze their data and to see a demonstration of the capabilities of the analysis scripts.

The following is a list of the simulation scripts included with SPRTool. The discussion for each script will include its description and its associated analysis script (i.e. the analysis script(s) used to analyze the simulated data it generates).

- **Equilibrium Scatchard Simulate Script**

  Filename equilibrium_scatchard_simulate_script.m

  Description Simulates equilibrium data for the purpose of demonstrating the analysis of equilibrium data using the traditional Scatchard technique. An equilibrium object will be created to store the simulated data.

  Associated Analysis Scripts equilibrium_scatchard_script.m

- **Equilibrium Ligand Activity Loss Simulate Script**

  FileName equlibrium_ligandactivityloss_simulate_script.m

  Description Simulates equilibrium data with loss of ligand activity for four different decay models: Single Exponential, Double Exponential, Single Exponential Without Constant, and Constant (i.e. no loss). An equilibrium object will be created to store the simulated data.

  Associated Analysis Scripts equilibrium_ligandactivityloss_script.m

- **Kinetic Independent Interaction Simulate Script**

  FileName kinetic_independentinteraction_simulate_script.m

  Description Simulates kinetic data with the assumption of a 1:1 interaction. The data is simulated using the independent interaction model. An adjust object will be generated to store the simulated data.

  Associated Analysis Scripts kinetic_hankel_script.m, kinetic_optimization_script.m

- **Kinetic Mass Transport Simulate Script**

  FileName kinetic_masstransport_simulate_script.m

  Description Simulates kinetic data with the assumption of a 1:1 interaction with mass transport present. The mass transport effect is simulated using the standard compartment model. An adjust object will be generated to store the simulated data.

  Associated Analysis Scripts kinetic_optimization_script.m

Chapter 5

Working with Your Own Data

This chapter will discuss how to perform your own SPR analysis with your own data using SPRTool. You should go through the sections in the order they are presented whenever you wish to start a new session of SPR analysis. This chapter assumes that you have gone through the Tutorial Chapter in full. If you have not please do so before continuing.

5.1 How To Prepare Data For Processing

This section contains directions on how to prepare the data acquired from the BIAcore using the BIAEvaluation software version 3.1. If your data is acquired from a machine with **LESS THAN FOUR CHANNELS**, be sure to follow the instructions in the subsection *Preparing Data from Machines with Less Than Four Channels.*

5.1.1 A Note on the Data File Expected By SPRTool

We have time and again used SPRTool to analyze data acquired from a four-channel BIAcore machine and the software has worked well with both tab-delimited text data files and Microsoft Excel data files. If your data is acquired from a different type of machine or any machine with less than four channels, then you need to first modify your data file to conform to the four-channel BIAcore data format that is expected by SPRTool. Please refer to the subsection *Preparing Data from Machines with Less Than Four Channels* for an example of what the data file should look like as well as instructions for modifying your data file to conform to the expected format. You can also open and view the tutorial data files TutorialData.txt and TutorialData.xls provided with every tutorial session to see an example of the format expected by SPRTool.

If you have any questions concerning the data format expected by SPRTool or if SPRTool does not work with your data file properly, please do not hesitate to contact us at *wardlab@utsouthwestern.edu*.

5.1.2 Exporting BIAcore Data to Text File

This is the prefered method for converting your data. If this format does not work you can try the Excel format.

1. Open the BIAcore Evaluation program.
2. Open the desired '.blr' data file.
3. Save all the data as a '.ble' BIAEvaluation data file.
4. Click on **File** and select **Export**.
5. Export as a '.txt'(Text) file.

5.1.3 Importing BIAEvaluation Text Data into Excel Format

This section describes how to convert data in text file format to Excel format. It is only necessary if you have problems using the text file format or if you prefer the Excel format.

1. Open Microsoft Excel.
2. If necessary, click the **New** button to create a new, blank workbook.
3. On the **Data** menu, select **Get External Data** and then select **Import Text File**.
4. In the **Import Text File** window that pops up, locate the text file you are importing and then click **Import**.
5. The **Text Import Wizard** will open.
6. Make sure **Delimited** is selected and advance to step 3 of 3 in the **Text Import Wizard** by clicking **Next**. At step 3 of 3 click **Finish**.

7. In the dialog that pops up, make sure **Existing Worksheet** is selected and the box contains '=$A$1', then click OK.

8. There are two possibilities at this point:

   - If the data set is small enough to fit on one worksheet, then you are done. Just make sure that you save the Excel file.
   - Otherwise, on large sets of data a box will appear telling you that there is more data than will fit on one worksheet. In this case, take the following steps:
     (a) Click OK and it will import the first 256 columns of the data onto **Sheet 1**.
     (b) Next click on the tab for **Sheet 2** and repeat steps 3-5.
     (c) Continue through the **Text Import Wizard** to step 3 of 3 and in the **Data Preview** area, select the first 256 columns.
     (d) Under **Column Data Format**, click **Do Not Import Column (skip)** and then **Finish**.
     (e) In the dialog that pops up, make sure **Existing Worksheet** is selected and the box contains '=$A$1', then click OK.
     (f) If the data doesn't fit on two sheets, repeat steps (a) through (e) but click on Sheet 3 instead in step (b) and exclude the first 512 columns instead in steps (c) and (d). Continue in this fashion for as many worksheets as needed.
     (g) After you're done importing all the data, save the Excel file.

5.1.4 Preparing Data from Machines with Less Than Four Channels

If you acquire data from a machine that has less than four channels, you must take the following steps in order for SPRTool to work with your data:

1. Export your BIAcore data file to a text file by following the instructions in the above subsection *Exporting BIAcore Data to Text File.*

2. Convert the text file to Excel format by following the instructions in the above subsection *Importing BIAEvaluation Text Data into Excel Format.*

3. Modify the Excel file by adding columns of zeros such that the data ends up resembling data from a four-channel machine with zeros for the columns that correspond to the "made up" channels. Make sure that the size of each column of zeros you add is the **same** as that of the real data columns **from the same run**. Also, it is imperative that **every run** includes data from the **reference channel** AND that the reference channel number is the **same** for all runs. In other words, given the reference channel number N where N can be 1, 2, 3, or 4, columns (2N-1)+8k-8 and 2N+8k-8 for each run k in your data file must contain respectively the X and Y values of reference channel data for run k.

   Example: Suppose you have a two-channel machine and your original Excel file data looks as follows: (Note that given the conditions specified, either channel 1 or channel 2 must be the reference channel in this example.)

Ch1_X	Ch1_Y	Ch2_X	Ch2_Y
0.5	20.9	0.6	92.3
1.5	20.8	1.6	92.5
2.5	20.9	2.6	92.4
3.5	20.8	3.6	92.5

You must then modify the Excel file by adding four columns of zeros: the X and Y data for the "made up" channel 3 and the X and Y data for the "made up" channel 4. Note that the columns of zeros are the same size as the columns of real data. The result should be as follows:

Ch1_X	Ch1_Y	Ch2_X	Ch2_Y	Ch3_X	Ch3_Y	Ch4_X	Ch4_Y
0.5	20.9	0.6	92.3	0	0	0	0
1.5	20.8	1.6	92.5	0	0	0	0
2.5	20.9	2.6	92.4	0	0	0	0
3.5	20.8	3.6	92.5	0	0	0	0

4. Save the modified Excel file and use it as the data file for your SPR analysis (i.e. the data file to be read in by the experiment script).

5.2 Creating A New SPRTool Session

This section contains directions on how to start a new SPRTool session for working with your own data.

Note: The instructions that follow will use **C:\MATLAB6p5\toolbox\local\SPRTool** as a sample **SPRTool Installation Directory**. If your SPRTool Installation Directory is different, then you **MUST substitute it with your own SPRTool Installation Directory**. Also, be sure you enclose the path to your SPRTool Installation Directory in **single quotes**!!

1. With MATLAB open go to your **SPRTool Installation Directory** by entering the following command at the **Command Window** prompt:

   ```
   >> cd('C:\MATLAB6p5\toolbox\local\SPRTool')
   ```

2. Start the session manager by entering the following command at the **Command Window** prompt:

   ```
   >> SPR_session
   ```

3. Click the **"NEW"** button in the window that appears to create a new session.

4. Click the **"REGULAR"** button in the window that appears to indicate that you want to open a regular session.

5. Enter a name for the session in the input box that appears. Another window also appears that contains a list of the names of sessions you have created previously. Unless you wish to completely overwrite one of these existing SPRTool sessions, you should provide a unique name for your new session. Click "OK".

6. When asked whether you want to copy your data file into your session directory, select **Yes** and using the file browser that appears, navigate to the directory where you exported your '.txt' data file. Select that file and click "Open". (Note: The data file can also be a Microsoft Excel file.)

7. You have now created a new regular SPRTool session. Your current directory will be changed to the new session directory that you've just created. If you look at the **Current Directory** window, you should see a copy of each script we discussed in the tutorial as well as the data file you specified in the previous step.

5.3 Filling in and Running Your Scripts

Now you are ready to fill in and run your template scripts. The following is an outline of the steps you need to take for filling in and running a script.

Procedure for Filling in and Running a Script

1. Open the script.
2. Fill in the appropriate fields. Instructions on how to fill in the fields are given in the scripts themselves. **Please follow those instructions carefully.** As a supplement, you may also refer to the appropriate section in the *SPRTool Scripts* chapter for a description of a script's fields.
3. Save the changes you've made by going to the **File** menu and selecting **Save**.
4. Run the script.

Since the objects generated by the various scripts build on one another, **you will need to run the scripts in the following order:**

Running the Scripts in Order

1. chip_script
2. experiment_script
3. adjust_script
   Provided that you've run the adjust script, you can run the following scripts in any order:
   - kinetic_hankel_script - *(optional)*
   - kinetic_optimization_script - *(optional)*
4. equilibrium_script - *(optional)*
   Provided that you've run the equilibrium script, you can run the following scripts in any order:
   - equilibrium_scatchard_script - *(optional)*
   - equilibrium_ligandactivityloss_script - *(optional)*

It is important that you do not try to run these out of order because this could cause trouble at best. If you do not feel comfortable yet with filling in and running the scripts, please refer back to the tutorial chapter and do all of your experimenting with the tutorial scripts before doing any analysis with your own data.

5.4 Further Assistance

If you can not figure out something, find a bug, or have any questions about the software please feel free to contact us at:
wardlab@utsouthwestern.edu
We would be happy to answer any questions you have.

Chapter 6

SPRTool Viewer

This chapter will discuss the SPRTool Viewer that is intended to provide the user with a very flexible way of viewing data stored in the various SPRTool objects that he or she creates.

6.1 What is the SPRTool Viewer?

The SPRTool Viewer is an easy-to-use, GUI-based tool that allows you to load and view the contents of any SPRTool object that you have created with the various SPRTool scripts. For each SPRTool object that you load, you can view its content report and plots of its sensorgram data (if applicable) all with the use of your mouse.

6.2 Starting the SPRTool Viewer

After you have opened an SPRTool session, enter the following command at the **Command Window** prompt to start the SPRTool Viewer:

```
>> SPRToolViewer
```

You will see four windows opened.

6.3 Description and Usage of the SPRTool Viewer

The SPRTool Viewer is composed of four windows. The next four subsections will provide a detailed description of each window.

6.3.1 SPRTool Window

The SPRTool Window consists of just a title bar that displays the title "SPRTool". **When you close this window, the other three windows will be closed as well.**

6.3.2 SPRTool Figure Window

The SPRTool Figure Window is where the currently selected sensorgrams will be displayed. This window has three major components which you should pay attention to:

- The **body** of the window consists of a set of **axis** on which the currently selected sensorgrams will be plotted. By **clicking on a specific plot**, the sensorgram number, trace number, and data type of the corresponding sensorgram will be displayed in the **title** of the graph.
- There is a **toolbar** just above the body of the window that contains several buttons. The **only** button you need to familiarize yourself with is the **zoom in** button. The zoom in button is the one with the magnifying glass that has a plus sign in the middle. To **zoom in** on whatever is displayed in the body of the window, you would first click this button and then left click onto your area of interest. To **zoom back out** you would right click over the display area instead. After you're done, make sure you **deactivate** the zoom feature by clicking the zoom in button again.
- The **title bar** of the window will display the **coordinates** of wherever your mouse pointer is currently pointing. The coordinates will be updated as you move your mouse over the body of the window. It is important to remember that this feature will work **only** when the zoom feature is deactivated.

6.3.3 Loaded Object Window

The Loaded Object Window will allow you to load an SPRTool object for viewing. This window contains three major components:

- The **Load New Object** button will allow you to select an SPRTool object for viewing. When you click this button, a typical file browsing tool will appear. To load an SPRTool object, first select the type of object from the **Files of type:** popup menu at the bottom of the file browsing tool and then navigate to the location of the desired object file. Once there, select the desired object file with your mouse and click **Open**.

- The **status bar** at the top of the window will show the type of the currently loaded object. It is useful in case you forget the type of the currently loaded object.

- The **Display Object Information** button will allow you to view the content report of the currently loaded object. When you click this button, the content report will open in MATLAB's editor mode, exactly like what you would see if you generated and opened the report from the command line. Remember that you can **print** a copy of the report using the *print* option from the editor's *File* menu.

6.3.4 Plot Selection Window

The Plot Selection Window will allow you to select exactly which sensorgrams you want to see plotted in the SPRTool Figure Window. This window contains three major components:

- The **list box** will contain a listing of all the sensorgrams in the currently loaded SPRTool object **provided that plotting is available for the currently loaded object.** If the currently loaded object is a **chip or experiment** object, you will see the message *No Sensorgrams read in as yet* in the list box, indicating that no plotting will be available for chip and experiment objects. In fact, if the currently loaded object is a **chip or experiment** object, **the list box is the ONLY component you will see in this window.**

  If the currently loaded object is an **adjust or equilibrium** object, you will see a listing of all the sensorgrams it contains. Each entry in the list box will correspond to one sensorgram and will begin by showing the sensorgram's index, run, and trace numbers in the format **S: x; R: y; T: z;**, where x is the **S**ensorgram index, y is the sensorgram's **R**un number, and z is the sensorgram's **T**race number. This will be followed by additional, self-explanatory information about the sensorgram. At a minimum, there will be a string that indicates **whether the sensorgram is currently displayed** in the SPRTool Figure Window.

  Displaying a Sensorgram: By selecting an entry in the list box with your mouse, you will either display the corresponding sensorgram or remove the corresponding sensorgram from display, depending on what the current display status is. If the corresponding sensorgram is **not currently displayed**, then selecting its entry with your mouse **will display it in the SPRTool Figure Window**. On the other hand, if the corresponding sensorgram is **currently displayed**, then selecting its entry with your mouse **will remove it from display**.

  NOTE: You can **select more than one sensorgram entry at a time** by dragging your mouse and highlighting multiple consecutive sensorgram entries. Another way to accomplish the same thing is to first highlight the first of the multiple entries you want to select, and then highlight the last of the entries while holding down the Shift key on your keyboard.

- The **Choose data type for selection** popup menu will allow you to select the type of sensorgram data you want to plot, **provided that plotting is available for the currently loaded object.** If the currently loaded object is a **chip or experiment** object, then you will **NOT** see this popup menu at all as no plotting is available for chip and experiment objects.

  NOTE: You can display plots of **different data types concurrently in the SPRTool Figure Window**.

- The **Clear All Plots** button will remove all plots from the SPRTool Figure Window.

Chapter 7

Appendix A

7.1 Experiment Sensorgram GUI Viewer

The Experiment Sensorgram GUI Viewer is a graphical tool that allows you to view an experiment object's sensorgram data in a variety of ways. This section will provide details on how to open and use this viewer.

7.1.1 Starting the Viewer

Provided that you have generated the experiment object *expobject* in your current MATLAB workspace by running the experiment script, type the following at the **Command Window** prompt to start the Experiment Sensorgram GUI Viewer:

>> **guiplotdata(expobject);**

Note: The Experiment Sensorgram GUI Viewer can also be started by running the experiment script with the field *DisplayData* in Section 3 of the script set to 'yes'.

7.1.2 Description and Usage

When the Experiment Sensorgram GUI Viewer starts you will see **six** grey buttons, **one** blue button, **two** sliders, and a plot window. Let's take a look at each control in turn:

The Two Sliders

- **Run** - This slider will allow you to select the run to view. A run essentially corresponds to one injection of analyte over the flow cell. There are three numbers associated with it. The **top** number is the maximum number of runs (i.e. the highest-numbered run). The bottom number is the first run, which is usually equal to one. The **side** number is the current run being displayed to the plot window. When you first start Single mode the current run will be set to 1.
- **Trace** - This slider will allow you to select the trace to view. The trace corresponds to the channel. For some BIA machines this might be 1, 2, or 4. The **top** number is the maximum number of traces (i.e. the highest-numbered trace). The bottom number is the first trace, which is usually equal to one. The **side** number is the current trace being displayed to the plot window. When you first start Single mode the current trace will be set to 1.

The Six Grey Buttons

- **Show All** - This button will display all of the sensorgram plots in one graph over time. The plots will probably be hard to see and will look more like lines than sensorgram plots.

  You can **zoom in** on a specific plot by using the MATLAB figure tool. To access the tool check the menu item **View->Figure Toolbar**. A new toolbar will appear on which you should see a magnifying glass with a plus sign in the middle. Click on the icon, which will depress the magnifier button. Now you can left click onto the area of the sensorgram plot that you wish to examine. To **zoom out** again just right click over the plot window. After zooming in to one of the sensorgram plots such that you can see it adequately in the plot window, you can press the magnifier button again to **deactivate** the zoom feature of the figure.

Notice that in **Show All** mode the two slider bars have **disappeared**. The reason for this is that the sliders are not necessary for viewing all the plots at one time.

- **Single** - This button will display each sensorgram individually. You will see **both** slider bars appear to the right. You can use the sliders to move through the SPR plots. With this view you do not have the clutter of any other plots on the screen than the one you are interested in.
- **Single Trace** - This button will display all runs of a single trace over time. Notice that **only** the **Trace** slider is available for trace selection.
- **Single Run** - This button will display all traces of a specified run. Notice that **only** the **Run** slider is available for run selection.
- **Single Parallel** - This button will display all runs of a specified trace **with the start time set to zero for each run**. This way you can compare all runs of a trace with each other in a parallel format. Notice that **only** the **Trace** slider is available for trace selection.
- **All Parallel** - This button will display all runs of a trace in parallel (**i.e. with the start time set to zero for each run**) for all traces. The plots for each trace will probably look more like a single line than sensorgram plots. To see the sensorgrams of a trace close up, use the zooming tool to zoom in on that trace's "line". Notice that in **All Parallel** mode the two slider bars have **disappeared**. The reason for this is that the sliders are not necessary for viewing all the plots at one time.

The Blue Button

- **Trace Details** - This button will display detailed information about a sensorgram. When clicked a figure will be displayed to the lower left. Then click on the sensorgram plot that you want information on. The info figure will then be propagated with the sensorgram information such as Flowrate, What was Injected, etc...

  You can at **any time** through the course of using the GUI click on a sensorgram and call up the information of that plot. To **remove** the detailed view of a sensorgram just click on the blue button again. This will remove the verbose output figure from the screen.

  Note: When using the information figure, you must make sure that the magnifier button is **NOT** depressed or else the information figure will not be propagated.

NOTE: In this release of the program the number of traces is hard coded to four. Do not be alarmed if you only have two channels of data but the maximum trace number shows four. If you do not have four traces of data then there is nothing displayed for the superfluous traces. This is just how the program works and is a feature not an error:).

Chapter 8

Appendix B

8.1 Current Directory Navigation

1. With MATLAB open go to the Current Directory window. (If the Current Directory window is not visible, click the **VIEW->DESKTOP LAYOUT->FIVE PANEL** drop-down from the VIEW MENU of the main MATLAB window. This should make all window types viewable. The Current Directory window will be to the center bottom.)

2. Click on the '...' button on the Current Directory window. Use the Directory Browser that appears to navigate to your target directory. Highlight the target directory and click **OK**. The contents of the target directory should now be displayed in the Current Directory Window.

8.2 Opening a .m File in Editor Mode

1. With MATLAB open use the Current Directory window to navigate to the directory containing the .m file to be opened. (See the above section if you don't know how to navigate.) The .m file to be opened should now be listed in the Current Directory window.

2. Open the .m file in editor mode using any **ONE** of the following three ways:

   - At the Command Window prompt, type:

     >> **open** *file_name* [**enter**]

     where *file_name* is the name of the .m file **without** the .m extension.
   - **Right-click** the .m file in the Current Directory window and select **Open** on the menu that appears.
   - **Double-click** the .m file in the Current Directory window.