SPRTool Reference Manual\textsuperscript{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

\textsuperscript{1}The development work for SPRTool was supported in part by the National Institutes of Health (R01 GM58538)
# Contents

I Classes

1 Adjust

1.1 Adjust constructor ............................................. 7
1.2 background subtract method .................................. 8
1.3 display method .................................................. 12
1.4 document method ............................................... 13
1.5 findinds method ................................................ 14
1.6 guifinterface method ........................................... 15
1.7 guisoloadobject method ....................................... 16
1.8 guiplotdata method ............................................. 17
1.9 guisaveobject method ......................................... 18
1.10 interpolate method ............................................ 20
1.11 loadobject method ............................................. 21
1.12 print method ................................................... 22
1.13 saveobject method ............................................ 23
1.14 selector method ................................................ 24
1.15 set method ...................................................... 25
1.16 subsasgn method ............................................... 27
1.17 subsref method ................................................ 28
1.18 xalign method .................................................. 29
1.19 xalignment method ............................................ 30
1.20 xreorient method .............................................. 31
1.21 xzeroadjust method .......................................... 32
1.22 xlistboosting method ........................................ 33

2 BalanceDOS

2.1 BalanceDOS constructor ........................................ 35
2.2 balance method .................................................. 36
2.3 display method ................................................... 38
2.4 fit method ....................................................... 39
2.5 guisoloadobject method ....................................... 40
2.6 guisaveobject method ......................................... 41
2.7 hankel method .................................................. 42
2.8 loadobject method .............................................. 43
2.9 saveobject method ............................................. 44
2.10 showsingularvalues method .................................... 45
2.11 subsasgn method ............................................... 46
2.12 subsref method ................................................. 47

3 Chip

3.1 Chip constructor ................................................ 49
3.2 display method ................................................... 50
3.3 document method ............................................... 52
3.4 guisoloadobject method ....................................... 53
3.5 guisaveobject method ......................................... 54
3.6 loadobject method .............................................. 55
3.7 print method ..................................................... 56
3.8 saveobject method ............................................. 57

5
3.9 subsasgn method .................................................. 59
3.10 subsref method .................................................. 60

4 DataClass .......................................................... 61
4.1 DataClass constructor ............................................. 62
4.2 display method ................................................... 64
4.3 document method .................................................. 65
4.4 guiloaddobject method .......................................... 66
4.5 guiseavoobject method .......................................... 67
4.6 loadobject method ............................................... 68
4.7 print method ...................................................... 69
4.8 saveobject method ............................................... 70
4.9 subsasgn method .................................................. 71
4.10 subsref method .................................................. 72
4.11 view method ...................................................... 73

5 DiscreteOutputSystem ............................................. 75
5.1 DiscreteOutputSystem constructor ............................ 76
5.2 diagonalize method ............................................... 78
5.3 display method ................................................... 79
5.4 document method .................................................. 80
5.5 dos2int method .................................................... 81
5.6 guiloaddobject method .......................................... 82
5.7 guiseavoobject method .......................................... 83
5.8 isdiagonal method ............................................... 84
5.9 loadobject method ............................................... 85
5.10 print method ...................................................... 86
5.11 saveobject method ............................................... 87
5.12 showsystem method ............................................. 88
5.13 simulate method ................................................. 89
5.14 sorteigenvalues method ....................................... 90
5.15 subsasgn method .................................................. 91
5.16 subsref method .................................................. 92
5.17 view method ...................................................... 93

6 Equilibrium ......................................................... 95
6.1 Equilibrium constructor ...................................... 96
6.2 cutoutfunction method .......................................... 98
6.3 display method ................................................... 99
6.4 document method .................................................. 100
6.5 equilanalysis method ........................................... 101
6.6 equilestimate method .......................................... 103
6.7 exponential_fit method ......................................... 104
6.8 guitrlface method ............................................... 105
6.9 guiloaddobject method .......................................... 107
6.10 guiseavoobject method ........................................ 108
6.11 loadobject method ............................................... 109
6.12 maketime method ............................................... 110
6.13 plotanalysis method .......................................... 111
6.14 print method ...................................................... 113
6.15 saveobject method ............................................... 114
6.16 selector method .................................................. 115
6.17 set method ....................................................... 117
6.18 subsasgn method .................................................. 118
6.19 subsref method .................................................. 119
6.20 listboxstring method ........................................... 120
### 7 Experiment

<table>
<thead>
<tr>
<th>7.1 Experiment constructor</th>
<th>122</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2 display method</td>
<td>124</td>
</tr>
<tr>
<td>7.3 document method</td>
<td>125</td>
</tr>
<tr>
<td>7.4 equalizedatalength method</td>
<td>126</td>
</tr>
<tr>
<td>7.5 guidoobject method</td>
<td>127</td>
</tr>
<tr>
<td>7.6 guiplotdata method</td>
<td>128</td>
</tr>
<tr>
<td>7.7 guisaveobject method</td>
<td>130</td>
</tr>
<tr>
<td>7.8 loaddata method</td>
<td>131</td>
</tr>
<tr>
<td>7.9 loadobject method</td>
<td>132</td>
</tr>
<tr>
<td>7.10 plotdata method</td>
<td>133</td>
</tr>
<tr>
<td>7.11 print method</td>
<td>134</td>
</tr>
<tr>
<td>7.12 saveobject method</td>
<td>135</td>
</tr>
<tr>
<td>7.13 subsasgn method</td>
<td>136</td>
</tr>
<tr>
<td>7.14 subsref method</td>
<td>137</td>
</tr>
</tbody>
</table>

### 8 IndependentInteractionsModel

<table>
<thead>
<tr>
<th>8.1 IndependentInteractionsModel constructor</th>
<th>140</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.2 calculatekobs method</td>
<td>142</td>
</tr>
<tr>
<td>8.3 calculatekon method</td>
<td>143</td>
</tr>
<tr>
<td>8.4 calculateres method</td>
<td>144</td>
</tr>
<tr>
<td>8.5 calculateremax method</td>
<td>145</td>
</tr>
<tr>
<td>8.6 display method</td>
<td>146</td>
</tr>
<tr>
<td>8.7 document method</td>
<td>147</td>
</tr>
<tr>
<td>8.8 guidoobject method</td>
<td>148</td>
</tr>
<tr>
<td>8.9 guisaveobject method</td>
<td>149</td>
</tr>
<tr>
<td>8.10 loadobject method</td>
<td>150</td>
</tr>
<tr>
<td>8.11 print method</td>
<td>151</td>
</tr>
<tr>
<td>8.12 saveobject method</td>
<td>152</td>
</tr>
<tr>
<td>8.13 setparameters method</td>
<td>153</td>
</tr>
<tr>
<td>8.14 simulate method</td>
<td>154</td>
</tr>
<tr>
<td>8.15 subsasgn method</td>
<td>155</td>
</tr>
<tr>
<td>8.16 subsref method</td>
<td>156</td>
</tr>
<tr>
<td>8.17 view method</td>
<td>157</td>
</tr>
</tbody>
</table>

### 9 InjectTimes

<table>
<thead>
<tr>
<th>9.1 CallInjectTimeFcn method</th>
<th>160</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2 InjectTimeFcn method</td>
<td>161</td>
</tr>
<tr>
<td>9.3 InjectTimes constructor</td>
<td>162</td>
</tr>
<tr>
<td>9.4 display method</td>
<td>164</td>
</tr>
<tr>
<td>9.5 document method</td>
<td>165</td>
</tr>
<tr>
<td>9.6 print method</td>
<td>166</td>
</tr>
<tr>
<td>9.7 subsasgn method</td>
<td>167</td>
</tr>
<tr>
<td>9.8 subsref method</td>
<td>168</td>
</tr>
</tbody>
</table>

### 10 MassTransportCompModel

<table>
<thead>
<tr>
<th>10.1 MassTransportCompModel constructor</th>
<th>170</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.2 display method</td>
<td>172</td>
</tr>
<tr>
<td>10.3 document method</td>
<td>173</td>
</tr>
<tr>
<td>10.4 guidoobject method</td>
<td>174</td>
</tr>
<tr>
<td>10.5 guisaveobject method</td>
<td>175</td>
</tr>
<tr>
<td>10.6 loadobject method</td>
<td>176</td>
</tr>
<tr>
<td>10.7 print method</td>
<td>177</td>
</tr>
<tr>
<td>10.8 saveobject method</td>
<td>178</td>
</tr>
<tr>
<td>10.9 setparameters method</td>
<td>179</td>
</tr>
<tr>
<td>10.10 simulate method</td>
<td>180</td>
</tr>
<tr>
<td>10.11 subsasgn method</td>
<td>181</td>
</tr>
<tr>
<td>10.12 subsref method</td>
<td>182</td>
</tr>
<tr>
<td>10.13 view method</td>
<td>183</td>
</tr>
</tbody>
</table>
11 Model 185
11.1 Model constructor ................................................................. 186
11.2 display method ................................................................. 187
11.3 document method ................................................................. 188
11.4 guisloadobject method ........................................................ 189
11.5 guisaveobject method .......................................................... 190
11.6 loadobject method ............................................................... 191
11.7 print method .......................................................... 192
11.8 saveobject method .............................................................. 193
11.9 subsasgn method ............................................................... 194
11.10 subsref method ............................................................... 195

12 OptimizeClass 197
12.1 OptimizeClass constructor .................................................. 198
12.2 display method ................................................................. 201
12.3 document method ............................................................... 202
12.4 fit method .......................................................... 203
12.5 guisloadobject method ....................................................... 204
12.6 guisaveobject method ........................................................ 205
12.7 loadobject method ............................................................. 206
12.8 print method .......................................................... 207
12.9 saveobject method ............................................................. 208
12.10 subsasgn method ............................................................. 209
12.11 subsref method ............................................................. 210

II Functions 211

13 General Functions 213
13.1 SegmentIntersection function .............................................. 214
13.2 bdataindex function .......................................................... 215
13.3 exp viewer function ........................................................... 216
13.4 exponentialFit function ...................................................... 217
13.5 selector function .............................................................. 218
13.6 bdataindex function ........................................................... 219
13.7 masstransportmodel function .............................................. 220
13.8 optimizationerror function .................................................. 221
Part I

Classes
Chapter 1

Adjust
1.1 Adjust constructor

SPRTool\V0.9\Program\ADj

a = Adjust(varargin)

Purpose:
The class constructor for the adjustment class. Will keep track of
original data as well as adjusted data. This will be the macro object
for the SPRTool.

Syntax:
a = Adjust;
a = Adjust(num_sensorgrams);

Arguments:
a - the adjust class object
num_sensorgrams - Number of sensorgram info structures to create.

Description:
a = Adjust; - Generates the default adjust class object with one
sensorgram info structure.
a = Adjust(num_sensorgrams); - Generates an adjust object with
however many sensorgram info structures as specified
by num_sensorgrams.

Programmer Comments:
None

Algorithm:
None

Limitations:
Can only generate a default object.

Method Version:
P1.0.0

Class fields:

a.ObjectCreator = getenv('UserName'); %string object creator
a.ObjectPathName = 'NA-str'; %string path for the object
a.ObjectFileName = 'NA-str'; %string filename for the object
a.ObjectSetupDate = datestr(now); %datum time stamp
a.Description = 'Adjust Object'; %object description of the
object
a.Comments = 'NA-str'; %string general comments about
the object
a.UserData = 'NA-cell'; %cell array user data section
%for storage
a.FieldExtensions = 'NA-cell'; %cell array user defined field
extensions
a.History = 'NA-cell'; %cell array keep track of what
is done to the object
a.Error = 'NA-str'; %string to keep track of error
messages
a.NumberOfExperiments = 'NA-d'; %{double} the number of experiments
a.TotalNumberOfSensorograms = 1; %{double} the total number of
%sensorograms
a.AdjustComments = 'NA-str'; %{string} any adjustment comments
a.AdjustCreator = 'NA-str'; %{string} the adjust data creator
a.AdjustDate = 'NA-str'; %{string} the adjustment date
a.AdjustFileName = 'NA-str'; %{string} the adjust file name
a.XAlignPoint = 'NA-d'; %{double} the x alignment point
a.InjectTimesObject = 'NA-obj'; %{object} the inject times object
a.ExperimentObjects = 'NA-cell'; %{cell array} the experiment object
a.SelectorVector = 'NA-d'; %{double} the selector vector
a.ChipNumbers = 'NA-str'; %{string} the chip numbers
a.NumberOfChips = 'NA-d'; %{double} the number of chips used
a.WhatFlowedAcross = 'NA-str'; %{string} the what flowed across
%vector
a.NumberFlowedAcross = 'NA-d';

a.Sensorogram(SensorgramIndex).Selected = 'NA-str'; %{string} {'yes'|'no'}
%is sensorgram selected
a.Sensorogram(SensorgramIndex).XData = 'NA-d'; %{double} the original
% x data
a.Sensorogram(SensorgramIndex).XDataUnits = 'NA-str'; %{string} the x data
% units
a.Sensorogram(SensorgramIndex).YData = 'NA-d'; %{double} the original
% y data
a.Sensorogram(SensorgramIndex).YDataUnits = 'NA-str'; %{string} the y data
% units
a.Sensorogram(SensorgramIndex).YDataDisplayed = 'NA-str'; %{string} {'yes'|'no'}
% will the data be
% displayed for y axis
a.Sensorogram(SensorgramIndex).XDataBiacoreHeader = 'NA-str'; %{string} x axis
% biacore header
a.Sensorogram(SensorgramIndex).YDataBiacoreHeader = 'NA-str'; %{string} y axis
% biacore header
a.Sensorogram(SensorgramIndex).YDataZeroAdjust = 'NA-d'; %{double} zero adjusted
% y axis
a.Sensorogram(SensorgramIndex).YDataZeroAdjusted = 'NA-str'; %{string} {'yes'|'no'}
% was data zero adjusted
a.Sensorogram(SensorgramIndex).YDataZeroAdjustDisplayed = 'NA-str'; %{string} {'yes'|'no'}
% will data be displayed
a.Sensorogram(SensorgramIndex).InjectionStartSourcePoint = 'NA-d'; %{double} injection
% start source point
a.Sensorogram(SensorgramIndex).InjStartPtEstimated = 'NA-str'; %{string} {'yes'|'no'}
% was injection start
% point estimated
a.Sensorogram(SensorgramIndex).InjStartPtDisplayed = 'NA-str'; %{string} {'yes'|'no'}
% will injection data
% be displayed
a.Sensorogram(SensorgramIndex).InjectionStopSourcePoint = 'NA-d'; %{double} injection
% stop source point
a.Sensorogram(SensorgramIndex).InjStopPtEstimated = 'NA-str'; %{string} {'yes'|'no'}
% was injection stop
% point estimated
a.Sensorogram(SensorgramIndex).InjStopPtDisplayed = 'NA-str'; %{string} {'yes'|'no'}
% will data be displayed
a.Sensorogram(SensorgramIndex).XDataXAdjust = 'NA-d'; %{double} x axis
a.Sensorgram(SensorgramIndex).XDataAdjusted = 'NA-str'; % (string) {'yes'/'no'} % was x data adjusted
a.Sensorgram(SensorgramIndex).XDataAdjustDisplayed = 'NA-str'; % (string) {'yes'/'no'} % will data be displayed
a.Sensorgram(SensorgramIndex).XAdjustment = 'NA-d'; % (double) how much the % sensorgram was shifted
a.Sensorgram(SensorgramIndex).XDataAdjustInjStartPoint = 'NA-d'; % (double) just the x % coordinate using x % adjustment data for % start
a.Sensorgram(SensorgramIndex).XDataAdjustInjStopPoint = 'NA-d'; % (double) just the x % coordinate using x % adjustment data for % stop
a.Sensorgram(SensorgramIndex).YDataInterpolation = 'NA-d'; % (double) y data % interpolation
a.Sensorgram(SensorgramIndex).XDataInterpolation = 'NA-d'; % (double) x data % interpolation
a.Sensorgram(SensorgramIndex).DataInterpolated = 'NA-str'; % (string) {'yes'/'no'} % was data interpolated
a.Sensorgram(SensorgramIndex).DataInterpolatedDisplayed = 'NA-str'; % (string) {'yes'/'no'} % will it be displayed
a.Sensorgram(SensorgramIndex).ReferenceExperimentNumber = 'NA-d'; % (double) number of % experiment that contains % reference data
a.Sensorgram(SensorgramIndex).ReferenceRunNumber = 'NA-d'; % (double) number of run % in experiment that % contains reference data
a.Sensorgram(SensorgramIndex).ReferenceTraceNumber = 'NA-d'; % (double) number of trace % in experiment that % contains reference data
a.Sensorgram(SensorgramIndex).YDataReferenceSubtract = 'NA-d'; % (double) reference % subtraction data
a.Sensorgram(SensorgramIndex).YDataReferenceSubtracted = 'NA-str'; % (string) {'yes'/'no'} % was the Y data reference % subtracted
a.Sensorgram(SensorgramIndex).YDataReferenceSubtractDisplayed = 'NA-str'; % (string) {'yes'/'no'} % will the data be % displayed
a.Sensorgram(SensorgramIndex).ExperimentNumber = 'NA-d' % (double) experiment % number
a.Sensorgram(SensorgramIndex).NumberRun = 'NA-d' % (double) run number
a.Sensorgram(SensorgramIndex).NumberTrace = 'NA-d' % (double) trace number
a.Sensorgram(SensorgramIndex).OriginalFileName = 'NA-str' % (string) original data % file name
a.Sensorgram(SensorgramIndex).ExperimentDate = 'NA-str' % (string) experiment date
a.Sensorgram(SensorgramIndex).PurposeOfExperiment = 'NA-str' % (string) purpose of % experiment
a.Sensorgram(SensorgramIndex).ExperimentOperator = 'NA-str' % (string) experiment % operator
a.Sensorgram(SensorgramIndex).ExperimentComments = 'NA-str' % (string) any comments % about experiment
a.Sensorgram(SensorgramIndex).ChipNumber = 'NA-str' % (string) the chip
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.Sensorgram(SensorgramIndex).DateCoupled</td>
<td>= 'NA-str' (string) the date chip was coupled</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).WhoCoupled</td>
<td>= 'NA-str' (string) who coupled</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).ChipComments</td>
<td>= 'NA-str' (string) any comments about chip</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).Flowrate</td>
<td>= 'NA-d' (double) flowrate of the analyte</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).FlowrateUnits</td>
<td>= 'NA-str' (string) flowrate units</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).What Injected</td>
<td>= 'NA-str' (string) what was injected</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).Concentration</td>
<td>= 'NA-d' (double) concentration of analyte</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).ConcentrationUnits</td>
<td>= 'NA-str' (string) concentration units</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).UniformlySampled</td>
<td>= 'NA-str' (string) {‘yes’</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).SamplingRate</td>
<td>= 'NA-d' (double) sampling rate</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).SamplingRateUnits</td>
<td>= 'NA-str' (string) sampling rate units</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).SamplingInterval</td>
<td>= 'NA-d' (double) sampling interval</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).SamplingIntervalUnits</td>
<td>= 'NA-str' (string) sampling interval units</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).Temperature</td>
<td>= 'NA-d' (double) temperature</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).TemperatureUnits</td>
<td>= 'NA-str' (string) temperature units</td>
</tr>
<tr>
<td>could be channel 1,2,3,or 4.</td>
<td></td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).What Coupled</td>
<td>= 'NA-str' (string) what was coupled</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).CouplingLevel</td>
<td>= 'NA-d' (double) the coupling level</td>
</tr>
<tr>
<td>a.Sensorgram(SensorgramIndex).CouplingLevelUnits</td>
<td>= 'NA-str' (string) the coupling level units</td>
</tr>
</tbody>
</table>
1.2 background subtract method

SPRTool\V0.9\Program\Adjust

a = background subtract(a, channel, mode)

Purpose:
To do background subtraction of sensorgrams using a channel as
the reference. The channel for reference can be 1-4.

Syntax:
a = background subtract(a, channel, 'all');
a = background subtract(a, channel, 'selected');

Arguments:
a - the adjust object
channel - the channel used as reference for subtraction

Description:
a = background subtract(a, channel, 'all') - will do a background
subtraction to all the sensorgrams in the adjust object using the
channel specified.
a = background subtract(a, channel, 'selected') - will do a background
subtraction to all the sensorgrams with the Selected field set to 'yes'.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
1.3 display method

SPRTool\V0.9\Program\@Adjust

display(a, varargin)

Purpose:
    This is the display function for the adjust object. It will
display the fields in a formatted fashion.

Syntax:
    display(a);

Arguments:
    a - the adjust class input

Description:
    display(a); - Displays the formatted field data to the command window.

Programmer Comments:
    None

Algorithm:
    None

Limitations:
    None

Method Version:
    P1.0.0

Classfields:
    None
1.4  document method

SPRTool\V0.9\Program\@Adjust

-----------------------------------------------

document(object, varargin)

Purpose:
    Print the adjust information to a .m file for later analysis and printing

Syntax:
    document(object);

Arguments:
    object - the adjust object

Description:
    document(object) - generates a .m file with the adjust object fields displayed

Programmer Comments:
    None

Algorithm:
    None

Limitations:
    None

Method Version:
    P1.0.0

Classfields:
    None
1.5 findinds method

InjectionInds = findinds(adjustobject, XCoordInfo)

Purpose:
Calculate the injection indicies

Syntax:
InjectionInds = findinds(adjustobject, XCoordInfo);

Arguments:
adjustobject - The adjust object
XCoordInfo - The x coordinate structure explained below.
XCoordInfo.X - a vector of length 4 that contains an x coordinate for
each flow cell. The x coordinate is the last point before association
or dissociation began for a typical sensorgram in that flow cell.
XCoordInfo.SensorgramNumber - a vector of length 4 that contains the
number of the sensorgram used to select the corresponding point in XCoordInfo.X
InjectionInds - a vector that contains the indice of the last point before
association or dissociation began for each sensorgram.

Description:
document(object) - generates a .m file with the adjust object fields displayed

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
1.6  guiinterface method

SPRTool\V0.9\Program\Adjust

---

varargout = guiinterface(adjustobject, varargin)

interfaces to gui
Possible syntax choices

adjustobject = guiinterface(adjustobject, 'OriginalData', selectorvector)
adjustobject = guiinterface(adjustobject, 'ZeroAdjust', selectorvector)
adjustobject = guiinterface(adjustobject, 'ReferenceSubtract', selectorvector)
adjustobject = guiinterface(adjustobject, 'Selection', selectorvector)
adjustobject = guiinterface(adjustobject, 'OriginalData', 'All')
adjustobject = guiinterface(adjustobject, 'ZeroAdjust', 'None')
adjustobject = guiinterface(adjustobject, 'ReferenceSubtract', 'All')
adjustobject = guiinterface(adjustobject, 'Selection', 'None')
adjustobject = guiinterface(adjustobject, 'DisplayNone')
adjustobject = guiinterface(adjustobject, 'DisplayAll')

[adjustobject, liststring] = guiinterface(adjustobject, 'OriginalData', selectorvector)
[adjustobject, liststring] = guiinterface(adjustobject, 'ZeroAdjust', selectorvector)
[adjustobject, liststring] = guiinterface(adjustobject, 'ReferenceSubtract', selectorvector)
[adjustobject, liststring] = guiinterface(adjustobject, 'Selection', selectorvector)
[adjustobject, liststring] = guiinterface(adjustobject, 'OriginalData', 'All')
[adjustobject, liststring] = guiinterface(adjustobject, 'ZeroAdjust', 'None')
[adjustobject, liststring] = guiinterface(adjustobject, 'ReferenceSubtract', 'All')
[adjustobject, liststring] = guiinterface(adjustobject, 'Selection', 'None')

% The following commands read the
liststring = guiinterface(adjustobject, 'ZeroAdjust', 'ReadListString')
liststring = guiinterface(adjustobject, 'ReferenceSubtract', 'ReadListString')
liststring = guiinterface(adjustobject, 'EquilibriumCutout', 'ReadListString')
liststring = guiinterface(adjustobject, 'Selection', 'ReadListString')
liststring = guiinterface(adjustobject, 'OriginalData', 'ReadListString')
adjustobject = guiinterface(object, ZeroAdjust, 'ListboxAdjust', values_highlighted);
adjustobject = guiinterface(object, OriginalData, 'ListboxAdjust', values_highlighted);
adjustobject = guiinterface(object, Selection, 'ListboxAdjust', values_highlighted);
adjustobject = guiinterface(object, ReferenceSubtract, 'ListboxAdjust', values_highlighted);
adjustobject = guiinterface(object, EquilibriumCutout, 'ListboxAdjust', values_highlighted);

Raimund J. Ober copied from EQUILIBRIUM\guiinterface 26 March 2001
### 1.7 guiloadobject method

```
ClassObj = guiloadobject(ClassObj)
```

**Purpose:**
To load the adjust object from the disk with the help of a gui interface.

**Syntax:**
```
ClassObj = guiloadobject(ClassObj)
```

**Arguments:**
- `ClassObj` - the adjust class object to save to disk.

**Description:**
```
ClassObj = guiloadobject(ClassObj) - load the adjust object to disk using the
gui returned values for path and filename.
```

**Programmer Comments:**
- None

**Algorithm:**
- None

**Limitations:**
- None

**Method Version:**
- P1.0.0

**Classfields:**
- None
1.8 guiplotdata method

SPRTool\V0.9\Program\Adjust

---

guiplotdata(adj)

Purpose:
To display the raw sensorgram data in a RU/seconds plot with the help
of a graphical interface to control the display of the plots and which
plots will be displayed.

Syntax:
guiplotdata(adj);

Arguments:
adj - the experiment object

Description:
guiplotdata(adj) - The explanation I will give here will be the same
as the one given in the Tutorial Manual that supplements the program.

When the gui starts you will see six grey buttons, one blue button,
two sliders, and a plot window. First lets try one of the grey buttons and
see how it works.
The Show All 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 into 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 right click onto the area of the sensorgram
plot that you wish to examine. To zoom back out again just left click the
mouse over the plot window. Now zoom into one of the sensorgram plots. When
you can see it adequately in the plot window press the magnifier button again
to deactivate the zoom feature of the figure.

If you wish to display detailed information about a sensorgram all you have
to do is click on the blue button. 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 Flovrate, What
was Injected, etc... Note: You must make sure that the magnifier button is not
depressed or else the information figure will not be propagated. You can at
anytime 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.

You will notice that in {\bf 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. Now click on the {\bf Single} button. In this display mode you
have the ability to view each sensorgram individually. Now you will see both slider
bars have reappeared to the left. One of the bars is called {\bf Run} while the
other is called {\bf Trace}. You will also notice that for each slider there are
three numbers associated with that slider. The top number is the maximum number of
runs/traces. The bottom number is the first run/trace, which is usually equal to
one. The side number is the current run/trace being display to the plot window.
When you first start Single mode the current run/trace will be set to 1.
So what is a run and a trace? A run essentially corresponds to one injection of analyte over the flow cell. The trace corresponds to the channel. For some BIA machines this might be 1, 2, or 4. This software was developed using a 4 channel BIACore machine. For the purpose of this tutorial we will have 4 runs and 4 traces. You will see that the maximum run value is 4 and the maximum trace value is 4. 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.

Another very useful button for analyzing your sensorgram data is Single Parallel}. This mode will display all runs of a specified trace with setting the start time to zero for each run. This way you can compare all runs of a trace to each other in a sort od stack format. Another tool of interest in this regard is the Single Trace. This will display the runs of a single trace over time without setting each sensorgram plot start to zero. You can change the trace you view by using the slider bar.

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 display for the superfluous traces. This is just how the program works and is a feature not an error:).

Programmer Comments:
  1)The callback function for the gui is called exp_viewer.m and is located in the 'mfiles' directory. This file will show how each button and control was implemented.

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
1.9  **guisaveobject method**

guisaveobject(ClassObj)

**Purpose:**
To save the adjust object to the disk with the help of a gui interface.

**Syntax:**
guisaveobject(ClassObj);

**Arguments:**
ClassObj - the adjust class object to save to disk.

**Description:**
ClassObj = guisaveobject(ClassObj) - save the adjust object to disk using
the value supplied by the gui interface.

**Programmer Comments:**
None

**Algorithm:**
None

**Limitations:**
None

**Method Version:**
P1.0.0

**Classfields:**
None
1.10 interpolate method

SPRTool\V0.9\Program\@Adjust

---------------------------------

a = interpolate(a, channel, InterpMethod, mode);

Purpose:
To interpolate the zero adjusted data so as to align all the association
and dissociation curves. This will help with the kinetic analysis later on.

Syntax:
a = interpolate(a, channel, InterpMethod, mode);

Arguments:
a - the adjust object
channel - the channel to align the other sensorgrams with
InterpMethod - the interpolation method used to do the alignment. Can be
one of these values. {'nearest','linear','spline','cubic'}.
mode - Selects which sensorgrams will be interpolated. Can be one of these
values. {'selected' | 'all'}.

Description:
a = interpolate(a, channel, InterpMethod, mode) - Will interpolate the sensorgram
data using one of the channels as a reference using the interpolation method
given in the variable InterpMethod. The mode setting will tell the interpolate
method which sensorgrams to do the interpolation on.

Programmer Comments:
Uses interp1.m to do the actual interpolation of the sensorgram data.

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
1.11 loadobject method

ClassObj = loadobject(ClassObj, varargin)

Purpose:
To load the adjust object from the disk.

Syntax:
ClassObj = loadobject(ClassObj);
ClassObj = loadobject(ClassObj, filename);

Arguments:
ClassObj - the adjust class object to load from disk.
filename - a string containing the filename of object to load. Please
include the file extension in the name.

Description:
ClassObj = loadobject(ClassObj) - Loads the Adjust object from disk
using the values in ObjectFileName and ObjectPathName fields.
ClassObj = loadobject(ClassObj, filename) - Loads the Adjust object
specified by filename from disk.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
1.12 print method

print(object)

Purpose:
   Displays the ASCII file created by the document method so that
   the information can be printed by using the print button.

Syntax:
   print(object)

Arguments:
   object - the adjust object

Description:
   print(object) - displays the document information for printing

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.0

Classfields:
   None
1.13  saveobject method

SPRTool\V0.9\Program\@Adjust

saveobject(ClassObj)

Purpose:
   To save the adjust object to the disk.

Syntax:
   saveobject(ClassObj);

Arguments:
   ClassObj - the adjust class object to save to disk.

Description:
   saveobject(ClassObj) - save the adjust object to disk using the values in
   the ObjectFileName and ObjectPathName fields.

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.0

Classfields:
   None
1.14 selector method

SPRTool\0.9\Program\@Adjust

---

a = selector(a, varargin)

Purpose:
To set the selection variables of the adjust object. You can use the
selector to turn on and off the Original Data, Zero Adjusted Data,
Reference Subtraction, and the Selection switches. It switches
'selected' tag on/off of sensorgram to indicate whether or not an
individual sensorgram will be used in the adjust project and later
analyzes

Syntax:

adjustobject=selector(adjustobject,'OriginalData',selectorvector)
adjustobject=selector(adjustobject,'ZeroAdjust',selectorvector)
adjustobject=selector(adjustobject,'ReferenceSubtract',selectorvector)
adjustobject=selector(adjustobject,'Selection',selectorvector)
adjustobject=selector(adjustobject,'OriginalData','All')
adjustobject=selector(adjustobject,'OriginalData','None')
adjustobject=selector(adjustobject,'ZeroAdjust','All')
adjustobject=selector(adjustobject,'Selection',indices)
adjustobject=selector(adjustobject,'ZeroAdjust',indices)
adjustobject=selector(adjustobject,'ReferenceSubtract',indices)
adjustobject=selector(adjustobject,'Selection',indices)
adjustobject=selector(adjustobject,'DisplayNone')
adjustobject=selector(adjustobject,'DisplayAll')

Arguments:

adjustobject - the adjust object
selectorvector - the sensorgrams to change
indices - a vector of sensorgram indices

Description:

adjustobject = selector(adjustobject,'OriginalData',selectorvector)
adjustobject = selector(adjustobject,'ZeroAdjust',selectorvector)
adjustobject = selector(adjustobject,'ReferenceSubtract',selectorvector)
adjustobject = selector(adjustobject,'Selection',selectorvector)

adjustobject = selector(adjustobject,'OriginalData','All')
adjustobject = selector(adjustobject,'OriginalData','None')
adjustobject = selector(adjustobject,'ZeroAdjust','All')
adjustobject = selector(adjustobject,'ZeroAdjust','None')
adjustobject = selector(adjustobject,'ReferenceSubtract','All')
adjustobject = selector(adjustobject,'ReferenceSubtract','None')
adjustobject = selector(adjustobject,'Selection','All')
adjustobject = selector(adjustobject,'Selection','None')

adjustobject = selector(adjustobject,'OriginalData','transpose',indices)
adjustobject = selector(adjustobject,'ZeroAdjust','transpose',indices)
adjustobject = selector(adjustobject,'ReferenceSubtract','transpose',indices)
adjustobject = selector(adjustobject,'Selection','transpose',indices)

25
adjustobject = selector(adjustobject,'ReferenceSubtract','transpose',indices)
adjustobject = selector(adjustobject,'Selection','transpose',indices)
adjustobject = selector(adjustobject,'DisplayNone')
adjustobject = selector(adjustobject,'DisplayAll')

Programmer Comments:
  None

Algorithm:
  None

Limitations:

Method Version:
  P1.0.0

Classfields:
  None
1.15  set method

varargout = set(a, varargin)

Purpose:
  Sets fields of the adjust class and updates the appropriate dependencies.

Syntax:
  set(a);
  a = set(a, 'ExperimentObjects', e);

Arguments:
  a - adjust object
  e - experiment object
  'ExperimentObjects' - command to add an experiment to the adjust object

Description:
  set(a) - displays the help message for adjust set method
  a = set(a, 'ExperimentObjects', e) - Adds an experiment to the adjust object and sets the appropriate fields. Setup the sensorgrams structures of the adjust object.

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  1) Will overwrite the SelectorVector field with all one at each new experiment addition.

Method Version:
  P1.0.0

Classfields:
  None
### 1.16 `subsasgn` method

```
SPRTool\0.9\Program\0Adjust

a = subsasgn(a, index, val)
```

**Purpose:**

The `subsasgn` function for the `adjust` class. Used to set fields in an `adjust` object.

**Syntax:**

```matlab
a = subsasgn(a, index, val);
classobject.fieldname = val;
```

**Arguments:**

- `a` - the object having value set
- `index` - the command structure
- `val` - value to set field
- `classobject` - the object name
- `fieldname` - the objects field name to alter

**Description:**

`a = subsasgn(a, index, val)` - Using this approach you must supply the index structure which is described in the MATLAB documentation under classes.

```
classobject.fieldname = val - This is the preferred method of setting the field values. This way the index structure is generated for you. All you need to do is issue a command like 'classobject.fieldname = val'. Let's say we have an object of class experiment called 'exp'. It has a field called 'NumberRuns' and we wish to set it to a value of 2. You would issue the command 'exp.NumberRuns = 2;'. This would set NumberRuns to a value of 2 and return the updated object.
```

**Programmer Comments:**

None

**Algorithm:**

None

**Limitations:**

None

**Method Version:**

`P1.0.0`

**Classfields:**

None
1.17 subsref method

`b=subsref(a,index)`

Purpose:
The subsref function for the `adjust` class. Used to get field values in an `adjust` object.

Syntax:
```matlab
b=subsref(a,index);
val = classobject.fieldname;
```

Arguments:
- `a` - the object to get value from
- `b` - the returned field value
- `index` - the command structure
- `val` - value to get from field
- `classobject` - the object name
- `fieldname` - the objects field name to get value from

Description:
`b=subsref(a,index)` - Using this approach you must supply the index structure which is described in the MATLAB documentation under classes.

`val = classobject.fieldname` - This is the preferred method of getting the field values. This way the index structure is generated for you. All you need to do is issue a command like `val = classobject.fieldname`. Lets say we have an object of class experiment called 'exp'. It has a field called 'NumberRuns' which we wish to get the value from. You would issue the command `val = exp.NumberRuns`. This would get NumberRuns and set the variable 'val' to the value of the field.

Programmer Comments:
- None

Algorithm:
- None

Limitations:
- Can only generate a default object

Method Version:
- F1.0.0

Classfields:
- None
1.18 xalign method

SPRTool\V0.9\Program\Adjust

adjustobject = xalign(adjustobject, XCoordInfo, target, mode, mode2);

Purpose:
   Does an x alignment of the sensorgrams using one channel as a reference.

Syntax:
   adjustobject = xalign(adjustobject, XCoordInfo, target, mode, mode2);

Arguments:
   adjustobject - the adjust object
   XCoordInfo - the x coordinate info structure explained below.
      XCoordInfo.Dissociation.SensorgramNumber - the numbers of four
         sensorgrams (of traces 1-4) whose dissociation points have been
         estimated
      XCoordInfo.Dissociation.X - A vector of four values, each one
         is an estimate of the last coordinate before dissociation for
         sensorgrams in a given trace
      XCoordInfo.Association.SensorgramNumber - the numbers of four
         sensorgrams (of traces 1-4) whose association points have been
         estimated
      XCoordInfo.Association.X - A vector of four values, each one
         is an estimate of the last coordinate before association for
         sensorgrams in a given trace
   target - the x value that the Injection Start Points from all
   sensorgrams will be aligned to.
   mode - {'displayed' | 'selected' | 'all'} describes which
   sensorgrams will be aligned.
   mode2 - {'Association' | 'Dissociation'} describes which curve to
   use to do the alignment.

Description:
   adjustobject = xalign(adjustobject, XCoordInfo, target, mode, mode2) -
   Will perform an x alignment of the sensorgrams using the values passed
   in to the XCoordInfo structure, the target point, and the mode of
   operation.

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   You must have used at least two channels in your data acquisition.

Method Version:
   P1.0.0

Classfields:
   None
1.19 xalignment method

SPRTool\0.9\Program\0Adjust

a = xalignment(a,target,StartOrStop,mode);

Purpose:
  Does the actual alignment of the sensorgrams and stores the results in
  the object.

Syntax:
  a = xalignment(a,target,StartOrStop,mode);

Arguments:
  adjustobject - the adjust object
  target - the x value that the Injection Start Points from all
  sensorgrams will be aligned to.
  StartOrStop - {‘start’ | ‘stop’} will the alignment be done on the
  injection start or stop.
  mode - {‘displayed’ | ‘selected’ | ‘all’} describes which
  sensorgrams will be aligned.

Description:
  a = xalignment(a,target,StartOrStop,mode) - Do the alignment of the
  sensorgrams using the target point, the injection start or stop, and
  the mode value.

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
1.20  zeroadjust method

SPRTool\V0.9\Program\@Adjust

---------------------------------------------------------------------------------------------------------------------

a = zeroadjust(a, Xcoordinate, mode)

Purpose:
  To zero adjust the sensorgrams that are selected.

Syntax:
  a = zeroadjust(a, Xcoordinate, 'displayed');
  a = zeroadjust(a, Xcoordinate, 'all');
  a = zeroadjust(a, Xcoordinate, 'selected');

Arguments:
  a - adjust object
  Xcoordinate - is either an x coordinate of point at which the sensorgram will zero adjusted or a vector of points whose length equals the number of Sensorgrams to be zero adjusted

Description:
  a = zeroadjust(a, Xcoordinate, 'displayed')  - Will do zero adjustment to all the sensorgrams that have the field YDataDisplayed set to 'yes'.
  a = zeroadjust(a, Xcoordinate, 'all')  - Will do a zero adjustment to all the sensorgrams.
  a = zeroadjust(a, Xcoordinate, 'selected')  - Will do a zero adjustment to only the sensorgrams that have the Selected field set to 'yes'.

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
1.21 listboxstring method

SPRTool\V0.9\Program@Adjust/private

____________________________________________________________________________________

liststring=listboxstring(adjustobject,specification)

ADJUST/private produces a string that is to be used in the selector string for
the listbox.

possible syntax

liststring=listboxstring(adjustobject,'OriginalData')
liststring=listboxstring(adjustobject,'ZeroAdjust')
liststring=listboxstring(adjustobject,'ReferenceSubtract')
liststring=listboxstring(adjustobject,'EquilibriumCutout')

Version: 16 March 2001 Raimund J. Ober
copied from EQUILIBRIUM\private\listboxstring 26 March 2001
Chapter 2

BalanceDOS
2.1 BalanceDOS constructor

kb = BalanceDOS(varargin)

Purpose: The constructor for the BalanceDOS object. A BalanceDOS object is used to calculate the state space parameter (A, B, C) of the specified data via Kung’s algorithm.

Syntax:
    kb = BalanceDOS;

Arguments:
    kb - BalanceDOS objects

Description:
    kb = BalanceDOS - generates a default BalanceDOS object

Programmer Comments:
    None

Algorithm
    None

Limitations:

Method Version:
    P1.0.0

Class fields:
    kbmod.ObjectCreator = getenv('UserName'); %{string} object creator
    kbmod.ObjectPathName = 'NA-str'; %{string} path for the object
    kbmod.ObjectFileName = 'NA-str'; %{string} filename for the object
    kbmod.ObjectSetupDate = datestr(now); %{date} time stamp
    kbmod.Description = 'Fitting Method: Produce Balanced Discrete Output System'; %{string} description of the object
    kbmod.Comments = 'NA-str'; %{string} general comments about the object
    kbmod.UserData = 'NA-cell'; %{cell array} user data section for storage
    kbmod.FieldExtensions = 'NA-cell'; %{cell array} user defined field extensions
    kbmod.History = 'NA-cell'; %{cell array} keep track of what is done to %the object
    kbmod.Selected = 'NA-str'; %{'yes'|'no'} Should this data object be %used in processing
    kbmod.RealOrComplex = 'NA-str'; %{'real'|'complex'} Is the data real or %complex
    kbmod.ComplexFlag = 'NA-str'; %{'warning'|'ok'} The Complex flag; put in warning %if data has become complex during calculation
    kbmod.OutputDimension = 'NA-n'; %{double} rows of C - (p)
    kbmod.OutputLength = 'NA-n'; %{double} number of "time steps" in output
    kbmod.InputDimension = 'NA-n'; %{double} - (m)
    kbmod.StateSpaceDimension = 'NA-n'; %{double} number of rows/cols of A - (n) %inserted 'number' 10-11-2002
    kbmod.Parameters.A = 'NA-d'; %{State Space Parameter
    kbmod.Parameters.C = 'NA-d'; %{State Space Parameter

36
kbmod.Parameters.x0 = 'NA-d'; %State Space Parameter

kbmod.AEstimationTechnique = 'lsq'; %{'lsq'} Technique used to estimate A in %subspace algorithm. Right now we use %left devide, which is not the same as %least-squares solution in some cases.

kbmod.HankelCommand = 'minimum'; %{'minimum' | 'maximum' | 'minplus' | 'manual'} %type of Hankel to use

kbmod.SingularValues = 'NA-d'; %where the singular values will be stored

kbmod.ZeroSingularValue = 1e-16; %if a singular value are smaller than %BalanceDOSObj.ZeroSingularValue, it is %considered as zero
2.2 balance method

Syntax:
kbdos = balance(kbdos, dataclass);
[kbdos, H, U, S, V] = balance(kbdos, dataclass);
[kbdos, H, U, S, V] = balance(kbdos, dataclass, SelectedIndex);
[kbdos, H, U, S, V] = balance(kbdos, dataclass, SelectedIndex, svd_choice);

Arguments:
kbdos - the BalanceDOS object
H - Hankel Matrix
U, V - Unitary matrices
S - diagonal matrix S, of the same dimension as X and with nonnegative diagonal elements in decreasing order.
SelectedIndex - the number of the data set (Censorgram) to be analyzed.

Purpose: Calculate the state space realization of the data

Description:
kbdos = balance(kbdos, dataclass); Takes a Data object and the BalanceDOS object as inputs and returns the estimated balanced system parameters (obtained via Kung/subspace algorithm)

Programmer Comments
1. The parameters in BalancedDOSObj are consistent with those in DataObj because the parameters are updated in function hankel() based on the parameters in the DataObj.

Algorithm:

Limitations:
1. Only the least-squares algorithm are available to calculate the matrix A

Method Version:
P1.0.0

Class fields:
None
2.3 display method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\methods\@BalanceDOS

display(m)

Purpose:
This is the display function for the BalanceDOS object. It will display the fields in a formatted fashion.

Syntax:
display(m);

Arguments:
m - the BalanceDOS class input

Description:
display(c); - Displays the formated field data to the command window.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
### 2.4 fit method

[DOSobject, BalanceDOSObj, res] = fit(BalanceDOSObj, data, varargin)

**Purpose:** To fit the model to the data and return the residuals of the fit.

**Syntax:**

```matlab
[DOSobject, BalanceDOSObj, res] = fit(BalanceDOSObj, data)
[DOSobject, BalanceDOSObj, res] = fit(BalanceDOSObj, data, SelectedIndex)
```

**Arguments:**

- `data`: Data object
- `DOSobject`: The Discrete Output System object
- `res`: The residuals of the fit
- `SelectedIndex`: The number of the data set (Censorgram) to be processed.

**Description:**

[DOSobject, BalanceDOSObj, res] = fit(BalanceDOSObj, data) - calculate the estimate of the data from the state space realization.

**Programmer Comments:**

None

**Algorithm:**

None

**Limitations:**

None

**Method Version:**

P1.0.0

**Classfields:**

None
2.5 guiloadobject method

ClassObj = guiloadobject(ClassObj)

Purpose:
To load the BalanceDOS object from the disk with the help of a gui interface.

Syntax:
ClassObj = guiloadobject(ClassObj)

Arguments:
ClassObj - the BalanceDOS class object to save to disk.

Description:
ClassObj = guiloadobject(ClassObj) - load the BalanceDOS object to disk using the
gui returned values for path and filename.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
2.6  guisaveobject method

ClassObj = guisaveobject(ClassObj)

Purpose:
   save the object

Syntax:
    ClassObj = guisaveobject(ClassObj)

Arguments:
    ClassObj - a BalancedDOS object

Description:
    ClassObj = guisaveobject(ClassObj) - save the object via GUI

Programmer Comments:
    None

Algorithm:
    None

Limitations:
    None

Method Version:
    P1.0.0

Classfields:
    None
2.7  hankel method

[ H, varargout ] = hankel(BalanceDOSObj, DataClassObj, varargin)

Purpose:
To construct an Hankel matrix.

Syntax:
H = hankel(BalanceDOSObj, DataClassObj, SelectedIndex)
[H, BalanceDOSObj] = hankel(BalanceDOSObj, DataClassObj)
[H, BalanceDOSObj] = hankel(BalanceDOSObj, DataClassObj, SelectedIndex)

Arguments:
BalanceDOSObj - a BalanceDOSObj object.
DataClassObj - a DataClass object.
SelectedIndex - the number of the data set (Censorgram) to be analyzed.
p = output dimension - rows
m = output dimension - cols

Description:
H = hankel(BalanceDOSObj, DataClassObj, SelectedIndex);
It will construct the Hankel matrix based on the selected data set.

Programmer Comments:
1. All parameters related to the data are read from the DataClass object

Limitation:
1. YData has to be in a matrix/vector valued column vector.

Method Version:
P1.0.0

Classfields:
None
2.8 loadobject method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\methods@BalanceDOS

----------------------------------------------------------------------------------

ClassObj = loadobject(ClassObj, varargin)

Purpose:
To load the BalanceDOS object from the disk.

Syntax:
ClassObj = loadobject(ClassObj);
ClassObj = loadobject(ClassObj, filename);

Arguments:
ClassObj - the BalanceDOS class object to save to disk.
filename - a string containing the filename of object to load. Please include the file extension in the name.

Description:
ClassObj = loadobject(ClassObj) - load the BalanceDOS object to disk using the values in ObjectFileName and ObjectPathName fields or filename variable.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
2.9 saveobject method

saveobject(ClassObj)

Purpose:
To save the BalanceDOS object to the disk.

Syntax:
saveobject(ClassObj);

Arguments:
ClassObj - the BalanceDOS class object to save to disk.

Description:
saveobject(ClassObj) - save the BalanceDOS object to disk using the fields in
the ObjectFileName and ObjectPathName fields.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
2.10 showsingularvalues method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\methods\@BalanceDOS

varargout=showsingularvalues(BalanceDOSObj,varargin)

Purpose:
To display the singular values of the Hankel matrix

syntax:
  showsingularvalues(BalanceDOSObject)
  showsingularvalues(BalanceDOSObject, first_few)

  singularvalues=showsingularvalues(BalanceDOSObject)
  singularvalues=showsingularvalues(BalanceDOSObject)

Arguments:
  BalancedDOSObj: BalancedDOS object
  first_few: the number of the singular value to be displayed

Description:
  showsingularvalues(BalanceDOSObject)
    - plots singular values
  showsingularvalues(BalanceDOSObject, first_few)
    - plots the number of the singular values specified by first_few
  singularvalues=showsingularvalues(BalanceDOSObject)
    - plots singular
  singularvalues=showsingularvalues(BalanceDOSObject)
    - plots the number of the singular values specified by first_few

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
2.11 subsasgn method

a=subsref(a,index,val)

Purpose:
This is the subsasgn function for the BalanceDOS class. It will evaluate field
value changes by the user and set the fields accordingly.

Syntax:
a=subsref(a,index,val)

Arguments:
a - the BalanceDOS class
index - structure array with the fields: .subs and .type. See matlab
documents for complete explanation.
val - the value to change the fields current content too.

Description:
a=subsref(a,index,val); - sets the field values of the class

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
2.12 subsref method

b=subsref(a, index)

Purpose:
To display the fields of the class

Syntax:
b=subsref(a, index);

Arguments:
b, a - the BalanceDOS class
index - structure array with the fields: .subs and .type. See matlab documents for complete explanation.

Description:
b=subsref(a, index); - sets the field values of the class

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
Chapter 3

Chip
3.1 Chip constructor

SPRTool\V0.9\Program\Chip

---

c=Chip

Purpose:
This class holds the information for the chip that was used in the biacore experiment. It records what was coupled, what level, and by whom.

Syntax:
c=chip;

Arguments:
c - the outputed chip class

Description:
c=chip; - Generates a default chip class with all fields set to defualts

Programmer Comments:
None

Algorithm:
None

Limitations:
Can only generate a default object

Method Version:
P1.0.0

Classfields:
c.ObjectCreator = getenv('UserName'); %{string} object creator
c.ObjectPathName = 'NA-str'; %{string} path for the object
c.ObjectFileName = 'NA-str'; %{string} filename for the object
c.ObjectSetupDate = datestr(now); %{datum} time stamp
c.Description = 'Chip Object'; %{data object} description of the object
c.Comments = 'NA-str'; %{string} general comments about the object
c.UserData = 'NA-cell'; %{cell array} user data section for storage

c.FieldExtensions = 'NA-cell'; %{cell array} user defined field extensions

c.History = 'NA-cell'; %{cell array} keep track of what is done to the object

c.Error = 'NA-str'; %{string} to keep track of error messages

c.ChipNumber = 'NA-str'; %{string} number or other description for identification of chip

c.DateCoupled = 'NA-str'; %{string} date with format DD/MM/YYYY

c.Channel1Coupled = 'NA-str'; %{string} what was coupled to channel 1

c.Channel2Coupled = 'NA-str'; %{string} what was coupled to channel 2

c.Channel3Coupled = 'NA-str'; %{string} what was coupled to channel 3

c.Channel4Coupled = 'NA-str'; %{string} what was coupled to channel 4

c.Channel1CouplingLevel = 'NA-d'; %{double} coupling level for channel 1

c.Channel1CouplingLevelUnits = 'NA-str'; %{string} coupling level units

c.Channel2CouplingLevel = 'NA-d'; %{double} coupling level for channel 2

c.Channel2CouplingLevelUnits = 'NA-str'; %{string} coupling level units

c.Channel3CouplingLevel = 'NA-d'; %{double} coupling level for channel 3

c.Channel3CouplingLevelUnits = 'NA-str'; %{string} coupling level units

c.Channel4CouplingLevel = 'NA-d'; %{double} coupling level for channel 4
c.Channel4CouplingLevelUnits = 'NA-str'; %{string} coupling level units
c.WhoCoupled = 'NA-str'; %{string} who did the coupling
c.Comments = 'NA-str'; %{string} additional comments
c.ChipFileName = 'NA-str'; %{string} the chip file name
3.2 display method

SPRTool\V0.9\Program\Chip

---------------------------------------------
display(c)

Purpose:
This is the display function for the chip object. It will display the fields in a formatted fashion.

Syntax:
display(c);

Arguments:
c - the chip class input

Description:
display(c); - Displays the formatted field data to the command window.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
3.3 document method

SPRTool\V0.9\Program\Chip

-----------------------------------------------
document(chipobject)

Purpose:
    Print the chip information to a .m file for later analysis

Syntax:
    document(chipobject)

Arguments:
    chipobject - the chip object

Description:
    document(chipobject) - generates a .m file with the chip object fields displayed

Programmer Comments:
    None

Algorithm:
    None

Limitations:
    None

Method Version:
    P1.0.0

Classfields:
    None
3.4 guiloaddobject method

SPRTool\V0.9\Program\0Chip

ClassObj = guiloaddobject(ClassObj)

Purpose:
To load the chip object from the disk with the help of a gui interface.

Syntax:
ClassObj = guiloaddobject(ClassObj)

Arguments:
ClassObj - the chip class object to save to disk.

Description:
ClassObj = guiloaddobject(ClassObj) - load the chip object to disk using the
gui returned values for path and filename.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
3.5 guisaveobject method

SPRTool\V0.9\Program\@Chip

---------------------------------------------------------------

guisaveobject(ClassObj)

Purpose:
To save the chip object from the disk with the help of a gui interface.

Syntax:
guisaveobject(ClassObj);

Arguments:
ClassObj - the chip class object to save to disk.

Description:
ClassObj = guisaveobject(ClassObj) - save the chip object to disk using
the value supplied by the gui interface.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
3.6 loadobject method

SPRTool\V0.9\Program\0Chip

ClassObj = loadobject(ClassObj, varargin)

Purpose:
To load the experiment object from the disk.

Syntax:
ClassObj = loadobject(ClassObj);
ClassObj = loadobject(ClassObj, filename);

Arguments:
ClassObj - the experiment class object to save to disk.
filename - a string containing the filename of object to load. Please
include the file extension in the name.

Description:
ClassObj = loadobject(ClassObj) - load the experiment object to disk using the
values in ObjectFileName and ObjectPathName fields or filename variable.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
3.7 print method

SPRTool\V0.9\Program\0Chip

print(object)

Purpose:
  None

Syntax:
  None

Arguments:
  None

Description:
  None

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
3.8 saveobject method

saveobject(ClassObj)

Purpose:
   To save the experiment object to the disk.

Syntax:
   saveobject(ClassObj);

Arguments:
   ClassObj - the experiment class object to save to disk.

Description:
   saveobject(ClassObj) - save the experiment object to disk using the fields in the ObjectFileName and ObjectPathName fields.

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.0

Classfields:
   None
3.9 subsasgn method

c=subsasgn(c,index,val)

Purpose:
This is the subsasgn function for the chip class. It will evaluate field value changes by the user and set the fields accordingly.

Syntax:
c=subsasgn(c,index,val);

Arguments:
c - the chip class
   index - structure array with the fields: .subs and .type. See matlab documents for complete explanation.
   val - the value to change the fields current content too.

Description:
c=subsasgn(c,index,val); - sets the field values of the class

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
3.10 subsref method

SPRTool\V0.9\Program\0Chip

---------------------------------------------------------------

d=subsref(c,index)

Purpose:
  To display the fields of the class

Syntax:
  c=subsref(c,index);

Arguments:
  c - the chip class
  d - the output field value
  index - structure array with the fields: .subs and .type. See matlab
documents for complete explanation.

Description:
  d=subsref(c,index); - sets the field values of the class

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
Chapter 4

DataClass
4.1 DataClass constructor

```
spRTool\V0.9\Program\biodatanalysis\V0.9\classes\data\DataClass

d = DataClass(varargin)

Purpose:
To create a default object for the class DataClass.

Syntax:
d = DataClass;
d = DataClass(num_sets);

Arguments:
d - DataClass object
num_sets - Number of Set structures to initialize

Description:
d = DataClass; - Construct a default DataClass object with one Set
structure initialized.
d = DataClass(num_sets); - Construct a DataClass object with as
many Set structures initialized as
specified by num_sets.

Programmer Comments:
None

Algorithm:
None

Limitations:
Can only generate a default object

Method Version:
P1.0.0

Classfields:

d.ObjectCreator = getenv('UserName');  %{string} object creator
d.ObjectPathName = 'NA-str';           %{string} object path
d.ObjectFileName = 'NA-str';           %{string} filename for the object
d.ObjectSetupDate = datestr(now);      %{datum} time stamp
d.Description = 'DataClass Object';    %{string} description of the object

d.Comments = 'NA-str';                  %{string} general comments about the
%object

d.UserData = 'NA-cell';                 %{cell array} user data section for
%storage

d.FieldExtensions = 'NA-cell';          %{cell array} user defined field
%extensions

d.History = 'NA-cell';                  %{cell array} keep track of what is
%done to the object

d.ErrorMessage = 'NA-str';              %{string} to keep track of error
%messages

d.NumberOfSets = 'NA-d';                %{integer} Number of Data Sets for this data
%object

d.Selected = 'NA-str';                   %{'yes'|'no'} Should this data object be used in
```
processing

%{'real'|'complex'} is the data real or complex

natural number} Index specifies data set that %should be used if a function/method can only use
%one data set (extend to multiple data sets)

{'yes'|'no'} Is this data set visible

{'yes'|'no'} Is the data uniform

{double} What is the sample interval

{string} sample interval units

{vector of natural numbers} indices of
%the outliners in the data set

{double array} the x axis data
%{string} X axis units

{double} Total no of points in x data
%{vector of natural numbers} see above
%{double array} the y axis data, the data
%is assumed to be a matrix of size
%dimOutput*dimInput*length(X)

{string} Y axis units

{double} Total no of points in x data
%{dimension (natural number)} see above
%{dimension (natural number)} see above

{string} comments about the data set
%{string} Description of the data set

{cell array} User data section for the data set
{double} if its elements are 1, then the correspond-
ng data point are used

{'yes'|'no'} whether DataPointsSelector should used
4.2 display method

display(d, varargin)

Purpose:
   To display information about the DataClass object

Syntax:
   display(d)
   display(d, 'verbose');

Arguments:
   d - The DataClass object
   'verbose' - the verbose command will print out details of all data sets

Description:
   display(d) - Gets the DataClass object and displays the information contained
               in it. Also Matlab calls this display method whenever an object is
               the result of a statement that is not terminated by a semicolon.
   display(d, 'verbose'); - Used to display all the data set info.

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.0

Classfields:
   None
4.3 document method

SPRTool\V0.9\Program\biodatanaalysis\V0.9\classes\data\@DataClass

document(object, varargin)

Purpose:
  Print the DataClass information to a .m file for later analysis and printing

Syntax:
  document(object);

Arguments:
  object - the DataClass object

Description:
  document(object) - generates a .m file with the DataClass object fields displayed

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
4.4 guiloadobject method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\data\@DataClass

---------------------------------------------------------------------------------

ClassObj = guiloadobject(ClassObj)

Purpose:
To load the DataClass object from the disk with the help of a gui interface.

Syntax:
ClassObj = guiloadobject(ClassObj)

Arguments:
ClassObj - the DataClass object to save to disk.

Description:
ClassObj = guiloadobject(ClassObj) - Gets the user input class object saved with extension _dataclass.mat and loads it into the workspace with the output argument name 'ClassObj'

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
4.5 guisaveobject method

```
SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\data\DataClass
```

```
guisaveobject(ClassObj)

Purpose:
To save the DataClass object from the disk with the help of a gui interface.

Syntax:
guisaveobject(ClassObj);

Arguments:
ClassObj - the DataClass object to save to disk.

Description:
ClassObj = guisaveobject(ClassObj) - Gets the user input for filename
and path for the DataClass object to be saved and saves it into the specified path

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
4.6 loadobject method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\data\@DataClass

ClassObj = loadobject(ClassObj, varargin)

Purpose:
To load the DataClass object from the disk.

Syntax:
ClassObj = loadobject(ClassObj);
ClassObj = loadobject(ClassObj, filename);

Arguments:
ClassObj - the DataClass object to load from disk.
filename - a string containing the filename of object to load. Please
include the file extension in the name.

Description:
ClassObj = loadobject(ClassObj) - Obtains the object filename from the
input DataClass object and loads it into the workspace
with the output argument name 'ClassObj'
ClassObj = loadobject(ClassObj, filename) - Loads the DataClass object
specified by filename into the workspace with the output
argument name 'ClassObj'

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
4.7 print method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\data\DataClass

----------------------------------

print(object)

Purpose:
Displays the ASCII file created by the document method so that
the information can be printed by using the print button.

Syntax:
print(object)

Arguments:
object - the DataClass object

Description:
print(object) - displays the document information for printing

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

ClassFields:
None
4.8 saveobject method

saveobject(ClassObj)

Purpose:
To save the DataClass object to the disk.

Syntax:
saveobject(ClassObj);

Arguments:
ClassObj - the DataClass object to save to disk.

Description:
saveobject(ClassObj) - saves the DataClass object to disk using the values in
the ObjectFileName and ObjectPathName fields.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
4.9 subsasgn method

a = subsasgn(a,index,val)

Purpose:
The subsasgn function for the DataClass class. Used to set fields in a DataClass object.

Syntax:
a = subsasgn(a,index,val);
classobject.fieldname = val;

Arguments:
a - the object having value set
index - the command structure
val - value to set field
classobject - the object name
fieldname - the object's field name to alter

Description:
a = subsasgn(a,index,val) - Using this approach you must supply the index structure which is described in the MATLAB documentation under classes.

classobject.fieldname = val - This is the preferred method of setting the field values. This way the index structure is generated for you. All you need to do is issue a command like 'classobject.fieldname = val'. Let's say we have an object of class DataClass called 'data1'. It has a field called 'Selected' and we wish to set it to a value of 'yes'. You would issue the command "data1.Selected = 'yes';". This would set Selected to a value of 'yes' and return the updated object.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
4.10 subsref method

b=subsref(a,index)

Purpose:
The subsref function for the DataClass class. Used to get field values in a DataClass object.

Syntax:
b=subsref(a,index);
val = classobject.fieldname;

Arguments:
a - the object to get value from
b - the returned field value
index - the command structure
val - value to get from field
classobject - the object name
fieldname - the object’s field name to get value from

Description:
b=subsref(a,index) - Using this approach you must supply the index structure which is described in the MATLAB documentation under classes.
val = classobject.fieldname - This is the preferred method of getting the field values. This way the index structure is generated for you. All you need to do is issue a command like 'val = classobject.fieldname. Lets say we have an object of class DataClass called 'data1'. It has a field called 'Selected' which we wish to get the value from. You would issue the command 'val = data1.Selected'. This would get Selected and set the variable 'val' to the value of the field.

Programmer Comments:
None

Algorithm:
None

Limitations:

Method Version:
P1.0.0

Classfields:
None
4.11 view method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\data\@DataClass

view(DataClassObject, varargin)

Purpose:
The purpose of this method is to plot the input datasets loaded into the
DataClass object.

Syntax:
view(DataClassObject)
view(DataClassObject, 'initial', datasetno)

Arguments:
DataClassObject - The DataClass object.
'initial' - The 'initial' command will initialize the GUI for viewing the plots.
datasetno - The dataset number to view initially.

Description:
view(DataClassObject) - Takes the DataClass object and plots the first data set.
Click on previous or next dataset button to view the respective datasets.
view(DataClassObject, 'initial', datasetno) - Takes the DataClass object,
initializing string (must be 'initial') and the dataset number to plot
the specified dataset. Click on previous or next dataset button to view
the respective datasets.

Programmer Comments:
None

Algorithm:
None

Limitations:
The input string argument in syntax 2 must be 'initial'.

Method Version:
P1.0.0

Classfields:
None
Chapter 5

DiscreteOutputSystem
5.1 DiscreteOutputSystem constructor

```matlab
% ds = DiscreteOutputSystem(varargin)

Purpose: To create a default discrete output system object, which carries information of the state space realizations of the system

Syntax:
   ds = DiscreteOutputSystem;

Arguments:
   ds - the DiscreteOutputSystem class object

Description:
   ds = DiscreteOutputSystem : Construct a default DiscreteOutputSystem class object

Programmer Comments:
   None

Algorithm:
   Calculate the output of the system vis its state space realization (A, x0, c) as

   \[ x(k+1) = Ax(k), \]
   \[ y(k) = Cx(k); \]
   or
   \[ y(0) = Cx0 \]
   \[ y(1) = CAx0 \]
   \[ y(2) = CA^2 x0 \]
   .
   .
   .
   \[ y(k) = CA^{k-1} x0 \]

In matlab you must use the equation \( y(k) = CA^{k-1} x0, k=1,2,3,\ldots \)

Limitations:

Method Version:
   P1.0.0

Classfields:
   ds.Description = 'Fitting Method: %Produce Balanced Discrete Output System'; %{string} description of the object
ds.OutputDimension = 'NA-n'; %{double} rows of C - (p)
ds.OutputUnits = 'RU'; %{'RU'} Default unit for output
ds.InputDimension = 'NA-n'; %{double} - (m)
ds.StateSpaceDimension = 'NA-n'; %{double} rows/cols of A - (n)
ds.ComplexFlag = 'NA-str'; %{'warning'|'ok'} The Complex flag; put in warning if data has become complex during calculation
ds.IsDiagonalFlag = 'NA-str'; %{string} (Yes/No) flag to check matrix A is diagonal or not
ds.DiagonalCheckConstant = 1e-16; %{double} Constant value to check matrix A is diagonal or not. Default value is 1e-16
```
ds.Parameters.A = 'NA-n';  %{double} A matrix values
ds.Parameters.C = 'NA-n';  %{double} C matrix values
ds.Parameters.x0 = 'NA-n';  %{double} Initial conditions
5.2 diagonalize method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@DiscreteOutputSystem

varargout = diagonalize(ds)

Purpose: To convert matrix A to a diagonal matrix.

Syntax:
    dds = diagonalize(ds);
    [dds,w] = diagonalize(ds);

Arguments:
    ds - Discrete output system object
    dds - Discrete output system object with diagonal matrix dds.parameters.A
    w - full matrix whose columns are corresponding eigen vectors

Description:
    dds = diagonalize(ds): Changes a discrete output system object into a
diagonalized discrete output system object and passes back only the
object.
    [dds,w] = diagonalize(ds) : Changes a discrete output system object
into a diagonalized discrete output system object and passes back the
object and the full matrix w.

Programmer Comments:
    None

Algorithm:
    None

Limitations:
    None

Method Version:
    P1.0.0

Classfields:
    None
5.3 display method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@DiscreteOutputSystem

display(object)

Purpose:
This is the display function for the DiscreteOutputSystem object. It will display the fields in a formatted fashion.

Syntax:
    display(object);

Arguments:
    object - the DiscreteOutputSystem class object

Description:
    display(object); - Gets the DiscreteOutputSystem class object and displays the information contained in it.

Programmer Comments:
    None

Algorithm:
    None

Limitations:
    None

Method Version:
    P1.0.0

Classfields:
    None
5.4 document method

document(object, varargin)

Purpose:
Print the Discrete Output System information to a .m file for later
analysis and printing

Syntax:
document(object);

Arguments:
object - the Discrete Output System object

Description:
document(object) - generates a .m file with the DOS object
fields displayed

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
5.5 dos2iint method

```
SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\DiscreteOutputSystem

iim = dos2iint(ds,iim)
```

Purpose: To find the association and dissociation parameters of Independent Interaction Model object from Discrete Output System object

Syntax:
```
iim = dos2iint(ds,iim)
```

Arguments:
- `ds` - the DiscreteOutputSystem class object
- `iim` - IndependentInteractionsModel class object

Description:
```
iim = dos2iint(ds,iim) - Obtains the A,C and x0 value from the DOS object and calculates the association and dissociation parameters of the IINT object
```

Programmer Comments:
```
None
```

Algorithm:
```
None
```

Limitations:
```
None
```

Method Version:
```
P1.0.0
```

Classfields:
```
None
```
5.6  guiloadobject method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@DiscreteOutputSystem

---------------------------------------------

ClassObj = guiloadobject(ClassObj)

Purpose:
To load the DiscreteOutputSystem object from the disk with the help of a gui interface.

Syntax:
ClassObj = guiloadobject(ClassObj)

Arguments:
ClassObj - the DiscreteOutputSystem class object to save to disk.

Description:
ClassObj = guiloadobject(ClassObj) - load the DiscreteOutputSystem object to disk using the
gui returned values for path and filename.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
5.7 guisaveobject method

ClassObj = guisaveobject(ClassObj)

Purpose:
save the DiscreteOutputSystem object with the help of GUI

Syntax:
    ClassObj = guisaveobject(ClassObj)

Arguments:
    ClassObj - a DiscreteOutputSystem object

Description:
    ClassObj = guisaveobject(ClassObj) - save the object via GUI

Programmer Comments:
    None

Algorithm:
    None

Limitations:
    None

Method Version:
    P1.0.0

Classfields:
    None
5.8  isdiagonal method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@DiscreteOutputSystem

---

ds = isdiagonal(ds)

Purpose: To check whether the state space matrix A is diagonal or not

Syntax:
   ds = isdiagonal(ds);

Arguments:
   ds - the DiscreteOutputSystem class object

Description:
   ds = isdiagonal(ds) : Obtains the diagonalized parameters from the object ds
   and manipulates to determine if matrix A is diagonal or not

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.0

Classfields:
   None
5.9 loadobject method

ClassObj = loadobject(ClassObj, varargin)

Purpose:
To load the DiscreteOutputSystem object from the disk.

Syntax:
ClassObj = loadobject(ClassObj);
ClassObj = loadobject(ClassObj, filename);

Arguments:
ClassObj - the DiscreteOutputSystem class object to save to disk.
filename - a string containing the filename of object to load. Please include the file extension in the name.

Description:
ClassObj = loadobject(ClassObj) - load the DiscreteOutputSystem object to disk using the values in model.ObjectFileName and model.ObjectPathName fields or filename variable.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
5.10 print method

print(object)

Purpose:
Displays the ASCII file created by the document method so that
the information can be printed by using the print button.

Syntax:
print(object)

Arguments:
object - the Model object

Description:
print(object) - displays the document information for printing

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
5.11 saveobject method

saveobject(ClassObj)

Purpose:
To save the DiscreteOutputSystem object to the disk.

Syntax:
saveobject(ClassObj);

Arguments:
ClassObj - the DiscreteOutputSystem class object to save to disk.

Description:
saveobject(ClassObj) - save the DiscreteOutputSystem object to disk using the fields in the model.ObjectFileName and model.ObjectPathName fields.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
5.12 showsystem method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@DiscreteOutputSystem

[A,C,x0] = showsystem(ds)

Purpose: To output the system from the discrete output system model

Syntax:
[A,C,x0] = showsystem(ds);

Arguments:
ds - Type(DiscreteOutputSystem), the DiscreteOutputSystem class object
A - (nxn) state space matrix
C - (pxn) output matrix
x0 - (nx1) initial conditions

Description:
[A,C,x0] = showsystem(ds) : reads out system parameters from the DiscreteOutputSystem object and return the values to the output arguments

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
5.13 simulate method

```
SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\0DiscreteOutputSystem
```

---

d = simulate(d)

**Purpose:** To simulate the system output with the model parameters stored in the **DiscreteOutputSystem** object

**Syntax:**
```
d = simulate(d);
```

**Arguments:**
- `d` - the **DiscreteOutputSystem** class object

**Description:**
```
d = simulate(d) : Creates a vector, yim, of simulated data based on the parameters in the **DiscreteOutputSystem** model object, ds, and updates the object with the simulated data into the SimulationDataObject of the model object. This method simulates the v output for the particular dataset no passed in as argument. It obtains the time from NumberOfPointsXData from the simulation info data object of model object.
```

**Programmer Comments:**
```
The code is written in such a way that it could be extend to handle multiple data sets with minimum change.
```

**Algorithm:**
- None

**Limitations:**
- It is assumed that there is only one set of the data

**Method Version:**
- P1.0.0

**Classfields:**
- None
5.14 **sorteigenvalues method**

```plaintext
sortds = sorteigenvalues(ds)
```

**Purpose:**
To sort the eigenvalues of diagonalized matrix A in descending order

**Syntax:**
```plaintext
sortds = sorteigenvalues(ds);
```

**Arguments:**
- `ds` - the `DiscreteOutputSystem` class object
- `sortds` - the `DiscreteOutputSystem` class object

**Description:**
sortds = sorteigenvalues(ds) : Obtains the diagonalized parameters from the object ds and sorts the eigen values of matrix A in descending order and outputs the result in the object sortds

**Programmer Comments:**
None

**Algorithm:**
None

**Limitations:**
None

**Method Version:**
P1.0.0

**Classfields:**
None
5.15  subsasgn method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@DiscreteOutputSystem

---------------------------------------------

d = subsasgn(d,index,val)

Purpose: Evaluates the value read in and sets the field correctly, it will give
an error if the field is set to an invalid value

Syntax:
    d = subsasgn(d,index,val)

Arguments:
    d - Type(DiscreteOutputSystem), the DiscreteOutputSystem class object
    index - Type(structure), with two fields index.type and index.subs
        index.type : string containing '(','{}', or '.' specifying subscript type
        index.subs : call array or string containing the actual subscripts
    val - new value

Description:
    d = subsasgn(d,index,val) - Obtains the argument values and assigns 'val'
to the field of DiscreteOutputSystem object d

Programmer Comments:
    None

Algorithm:
    None

Limitations:
    None

Method Version:
    P1.0.0

Classfields:
    None
5.16  subsref method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@DiscreteOutputSystem

b=subsref(d,index)

Purpose:
To display the fields of the class

Syntax:
b=subsref(d,index);

Arguments:
b, d - the DiscreteOutputSystem class
index - structure array with the fields: .subs and .type. See matlab
documents for complete explanation.

Description:
b=subsref(d,index); - sets the field values of the class

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
5.17 view method

view(ds,varargin)

Purpose: The purpose of this method is to plot the datasets of the SimulationDataObject in the Model class object

Syntax:
   view(ds);
   view(ds,datasetno);

Arguments:
   ds - the DiscreteOutputSystem object
   datasetno - the dataset number to be simulated

Description:
   view(ds) - Obtains the DataClass object, SimulationDataObject, from the modelobject and passes to the view function of the DataClass to plot the first dataset. Click on previous or next dataset button to view the respective datasets.
   view(ds,datasetno) - Obtains the DataClass object, SimulationDataObject, from the modelobject and passes to the view function of the DataClass to plot the particular data set passed in as argument. Click on previous or next dataset button to view the respective datasets.

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.0

Classfields:
   None
Chapter 6

Equilibrium
6.1 Equilibrium constructor

SPRTool\V0.9\Program\Equilibrium

---
equil = Equilibrium(varargin)

Purpose:
The class constructor for the Equilibrium class. Will generate an
equilibrium object for use in equilibrium analysis.

Syntax:
equil = Equilibrium;
equil = Equilibrium(num_sensorgrams);
equil = Equilibrium(num_sensorgrams, num_equilexps);

Arguments:
equil - the equilibrium class object
num_sensorgrams - Number of sensorgram info structures to create.
num_equilexps - Number of equilibrium experiment info structures
to create.

Description:
equil = Equilibrium; - Generates the default equilibrium class
object with one sensorgram info structure and
one equilibrium info structure.
equil = Equilibrium(num_sensorgrams); - Generates an equilibrium
object with however many sensorgram info
structures as specified by num_sensorgrams and
one equilibrium experiment info structure.
equil = Equilibrium(num_sensorgrams, num_equilexps); -
Generates an equilibrium object with however
many sensorgram and equilibrium experiment
info structures as specified by num_sensorgrams
and num_equilexps, respectively. (If you only
want to specify the number of equilibrium
experiments, pass in the value 1 for
num_sensorgrams followed by the number of
equilibrium experiments you want.)

Programmer Comments:
None

Algorithm:
None

Limitations:
Can only generate a default object.

Method Version:
P1.0.0

Classfields:
equil.ObjectCreator  = getenv('UserName');  %{string} object
equil.ObjectPathName = 'NA-str';    %{string} path for the

equil.ObjectFileName = 'NA-str';  % {string} filename for the object
equil.ObjectSetupDate = datestr(now);  % {datum} time stamp
equil.Description = 'Equilibrium Object';  % {object} description of the object
equil.Comments = 'NA-str';  % {string} general comments about the object
equil.UserData = 'NA-cell';  % {cell array} user data section for storage
equil.FieldExtensions = 'NA-cell';  % {cell array} user defined field extensions
equil.History = 'NA-cell';  % {cell array} keep track of what is done to the object
equil.Error = 'NA-str';  % {string} to keep track of error messages

equil.EquilComments = 'NA-str';  % {string} comments
equil.EquilCreator = 'NA-str';  % {string} creator
equil.EquilFileName = 'NA-str';  % {string} filename
equil.EquilDate = datestr(now);  % {string} date

equil.Sensorgram(k).Cutout = 'NA-str';  % {string} was data cutout
equil.Sensorgram(k).XDataCutout = 'NA-d';  % {double} x data cutout
equil.Sensorgram(k).YDataCutout = 'NA-d';  % {double} y data cutout
equil.Sensorgram(k).CutoutDisplayed = 'NA-str';  % {string} cutout displayed
equil.Sensorgram(k).EquilEstimate = 'NA-d';  % {double} equilibrium estimate

equil.NumberOfEquilibriumExperiments = 'NA-d';  % {double} number of experiments

equil.EquilExps(k).ChipNumber = 'NA-str';  % {string} chip number
equil.EquilExps(k).WhatFlowedAcross = 'NA-str';  % {string} analyte
equil.EquilExps(k).NumberTrace = 'NA-d';  % {double} trace number
equil.EquilExps(k).Sensorgrams = 'NA-d';  % {double} sensorgrams
equil.EquilExps(k).Reqs = 'NA-d';  % {double} req values
equil.EquilExps(k).ReqsHat = 'NA-d';  % {double} req hat values
equil.EquilExps(k).Concentrations = 'NA-cell';  % {cell} concentrations

equil.EquilExps(k).ReqsByConcentrations = 'NA-d';  % {double} reqs by concentrations
equil.EquilExps(k).KAEstimate = 'NA-d';  % {double} KA estimate
equil.EquilExps(k).KDEstimate = 'NA-d';  % {double} KD estimate
equil.EquilExps(k).KmaxEstimate = 'NA-d';  % {double} Kmax estimate
equil.EquilExps(k).EstimateTechnique = 'NA-str';  % {string} estimate technique
equil.EquilExps(k).CurvfitParameters = 'NA-d';  % {double} curve fit parameters
6.2 cutoutfunction method

SPRTool\V0.9\Program\@Equilibrium

------------------------------------------

a = cutoutfunction(a,Xcoordinates,mode)

Purpose:
Cuts out a segment of a sensorgram.

Syntax:
  a = cutoutfunction(a,Xcoordinates,mode);

Arguments:
a - the equilibrium object
Xcoordinates - Is either a 1x2 vector of x coordinates that determine the
interval that will be cut out or a nx2 vector (n= the number of sensorgrams)
of x coordinates that determine the intervals that will be cut out for each
sensorgram
mode - {'displayed','all','selected'} the sensorgrams to use for the cut out

Description:
a = cutoutfunction(a,Xcoordinates,mode) - you pass the x coordinates to do the
cutting and the mode of sensorgram selection. The method then returns the
updated equilibrium object with the cutout section stored.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
6.3 display method

SPRTool\V0.9\Program\@Equilibrium

display(a, varargin)

Purpose:
Command window display of an equilibrium object

Syntax:
display(a);

Arguments:
a - the Equilibrium class input

Description:
display(a); - Displays the formatted field data to the command window.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
6.4 document method

document(equilObject)

Purpose:
   EQUILIBRIUM/document writes essential information of an equilibrium object
to .m file (text file)
The file that is it being saved to has the same name as
given in equilobject.ObjectFileName but with .m extension.

Syntax:
   document(equilObject);

Arguments:
   equilObject - the equilibrium object

Description:
   document(equilObject) - generates a .m file with the equilibrium object fields displayed

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.0

Classfields:
   None
6.5  equilanalysis method

SPRTool\V0.9\Program\@Equilibrium

---------------------------------------------

a=equilanalysis(a, model, IConditions, ETime, varargin)

Purpose:
Analyzes equilibrium data and provides estimates for equilibrium constants (KA, KD) and the maximum coupling level Rmax. The analysis techniques supported are Scatchard, Single Exponential, Double Exponential, Single Exponential Without Constant, and Constant Fit. The estimated values will be assigned to the respective entries of the equilibrium object. To view the results use the plotanalysis method.

Syntax:
a=equilanalysis(a, model, IConditions, ETime);
a=equilanalysis(a, model, IConditions, ETime, Options);

Arguments:
a - the equilibrium object
model - {'all' | 'selected' | 'equilsegmentdisplayed'} - the type of sensorgrams to include in the equilibrium experiments.
IConditions - Vector of initial conditions for the exponential and constant fit techniques. See the equilibrium exponential script for the specific format to use for each technique. Supply the value {} to use the default initial conditions. For techniques that do not require initial conditions (Scatchard, for example), supply the value {} as a place filler for this parameter as it is a mandatory parameter.
ETime - Structure array of experiment time. See the equilibrium exponential script for the exact format to use. Supply the value {} to use the default experiment time generated by the program as it is a mandatory parameter.
Options - Optimization options for the lsqnonlin function. This is an optional parameter.

Description:
a=equilanalysis(a, model, IConditions, ETime); - Takes the equilibrium object a and performs equilibrium analysis on its data using the technique specified in the equilibrium object a. If IConditions and ETime contain valid data, then their values are used as the initial condition parameters and experiment time by the method.

a=equilanalysis(a, model, IConditions, ETime, Options); - Does the same things as the other syntax, but allows for the specification of options for the lsqnonlin function.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
6.6  equilestimate method

SPRTool\V0.9\Program\@Equilibrium

a=equilestimate(a)

Purpose:
Calculates the mean of the sensorgram cut outs.

Syntax:
  a = equilestimate(a);

Arguments:
  a - the equilibrium object

Description:
  TimeTrace = maketime(a,trace) - Takes the equilibrium object and the trace that you
  want to generate the time vector for.

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
6.7 exponential_fit method

SPRTool\V0.9\Program\Equilibrium

>Error] = exponential_fit(X, Optim)

Purpose:
This is the function to be minimized in the estimation of Kd.

Syntax:
This function is passed in as the first parameter to MATLAB’s lsqnonlin
function as follows:
lsqnonlin('exponential_fit', X, LB, UB, OPTIONS, Optim);

Arguments:
X - Initial conditions for the minimization; different parameters are
expected for each supported curve fitting technique:
For 'SingleExp', X = [Beta Alpha Gamma Kd]
For 'DoubleExp', X = [Beta1 Alpha1 Beta2 Alpha2 Kd]
For 'Constant', X = [Delta, Kd]
Optim - Structure containing data like concentrations and equilibrium
estimates (Req) needed by the minimization.

Description:
lsqnonlin('exponential_fit', X, LB, UB, OPTIONS, Optim);

Programmer Comments:
None

Algorithm:
None

Limitations:
Only supports the single exponential, the double exponential, the
single exponential without constant, and the constant curve fitting
techniques.

Method Version:
P1.0.1

Classfields:
None
6.8 guiinterface method

SPRTool\0.9\Program\0Equilibrium

varargout=guiinterface(equilobject, varargin)

interfaces to gui
Possible syntax choices

equilobject=guiinterface(equilobject,'OriginalData',selectorvector)
equilobject=guiinterface(equilobject,'ZeroAdjust',selectorvector)
equilobject=guiinterface(equilobject,'ReferenceSubtract',selectorvector)
equilobject=guiinterface(equilobject,'EquilibriumCutout',selectorvector)
equilobject=guiinterface(equilobject,'Selection',selectorvector)
equilobject=guiinterface(equilobject,'OriginalData','All')
equilobject=guiinterface(equilobject,'OriginalData','None')
equilobject=guiinterface(equilobject,'ZeroAdjust','All')
equilobject=guiinterface(equilobject,'ZeroAdjust','None')
equilobject=guiinterface(equilobject,'ReferenceSubtract','All')
equilobject=guiinterface(equilobject,'ReferenceSubtract','None')
equilobject=guiinterface(equilobject,'EquilibriumCutout','All')
equilobject=guiinterface(equilobject,'EquilibriumCutout','None')
equilobject=guiinterface(equilobject,'Selection','All')
equilobject=guiinterface(equilobject,'Selection','None')

equilobject=guiinterface(equilobject,'DisplayNone')
equilobject=guiinterface(equilobject,'DisplayAll')

[equilobject , liststring]=guiinterface(equilobject,'OriginalData',selectorvector)
[equilobject , liststring]=guiinterface(equilobject,'ZeroAdjust',selectorvector)
[equilobject , liststring]=guiinterface(equilobject,'ReferenceSubtract',selectorvector)
[equilobject , liststring]=guiinterface(equilobject,'EquilibriumCutout',selectorvector)
[equilobject , liststring]=guiinterface(equilobject,'Selection',selectorvector)
[equilobject , liststring]=guiinterface(equilobject,'OriginalData','All')
[equilobject , liststring]=guiinterface(equilobject,'OriginalData','None')
[equilobject , liststring]=guiinterface(equilobject,'ZeroAdjust','All')
[equilobject , liststring]=guiinterface(equilobject,'ZeroAdjust','None')
[equilobject , liststring]=guiinterface(equilobject,'ReferenceSubtract','All')
[equilobject , liststring]=guiinterface(equilobject,'ReferenceSubtract','None')
[equilobject , liststring]=guiinterface(equilobject,'EquilibriumCutout','All')
[equilobject , liststring]=guiinterface(equilobject,'EquilibriumCutout','None')
[equilobject , liststring]=guiinterface(equilobject,'Selection','All')
[equilobject , liststring]=guiinterface(equilobject,'Selection','None')

%The following commands read the
liststring=guiinterface(equilobject,'ZeroAdjust','ReadListString')
liststring=guiinterface(equilobject,'ReferenceSubtract','ReadListString')
liststring=guiinterface(equilobject,'EquilibriumCutout','ReadListString')
liststring=guiinterface(equilobject,'Selection','ReadListString')
equilobject=guiinterface(equilobject,OriginalData,'ReadListString')
equilobject=guiinterface(equilobject,ZeroAdjust,'ListboxAdjust',values_highlighted);
equilobject=guiinterface(equilobject,OriginalData,'ListboxAdjust',values_highlighted);
equilobject=guiinterface(equilobject,ReferenceSubtract,'ListboxAdjust',values_highlighted);
equilobject=guiinterface(equilobject,EquilibriumCutout,'ListboxAdjust',values_highlighted);
6.9 guiloadobject method

\[ SPRTool\V0.9\Program\@Equilibrium \]

ClassObj = guiloadobject(ClassObj)

Purpose:
To load the equilibrium object from the disk with the help of a gui interface.

Syntax:
ClassObj = guiloadobject(ClassObj)

Arguments:
ClassObj - the equilibrium class object to save to disk.

Description:
ClassObj = guiloadobject(ClassObj) - load the equilibrium object to disk using the
gui returned values for path and filename.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

None

Classfields:
6.10 guisaveobject method

SPRTool\V0.9\Program\@Equilibrium

guisaveobject(ClassObj)

Purpose:
To save the equilibrium object to the disk with the help of a gui interface.

Syntax:
guisaveobject(ClassObj);

Arguments:
ClassObj - the equilibrium class object to save to disk.

Description:
ClassObj = guisaveobject(ClassObj) - save the equilibrium object to disk using
the value supplied by the gui interface.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
6.11 loadobject method

ClassObj = loadobject(ClassObj, varargin)

Purpose:
To load the equilibrium object from the disk.

Syntax:
ClassObj = loadobject(ClassObj);
ClassObj = loadobject(ClassObj, filename);

Arguments:
ClassObj - the equilibrium class object to save to disk.
filename - a string containing the filename of object to load. Please
include the file extension in the name.

Description:
ClassObj = loadobject(ClassObj) - load the Equilibrium object from disk
using the values in ObjectFileName and ObjectPathName fields.
ClassObj = loadobject(ClassObj, filename) - Loads the Equilibrium object
specified by filename from disk.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
6.12 maketime method

TimeTrace = maketime(a,trace)

Purpose:
Makes a vector of time that records how long into the experiment each sensogram
was executed.

Syntax:
TimeTrace = maketime(a,trace);

Arguments:
a - the equilibrium object
trace - the trace you wish to create a time index for
TimeIndex - for each sensogram k Timeindex will have an entry that is the sum of the
duration of that and all previous sensograms.

Description:
TimeTrace = maketime(a,trace) - Takes the equilibrium object and the trace that you
want to generate the time vector for.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
6.13 plotanalysis method

SPRTool\V0.9\Program\@Equilibrium

a = plotanalysis(a, mode, ETime, varargin)

Purpose:
Plots results of equilibrium data analysis.

Syntax:
  a = plotanalysis(a, 'showall', ETime);
  a = plotanalysis(a, 'printall', ETime);
  a = plotanalysis(a, 'showsingle', ETime, ExpIndex);

Arguments:
a - the equilibrium object
'printall' - Specifies that all equilibrium experiments in
equilibrium object a should be plotted and printed.
'showall' - Specifies that all equilibrium experiments in
equilibrium object a should be plotted.
'showsingle' - Specifies plotting of only one equilibrium
  experiment in equilibrium object a.
ETime - Structure array of experiment time. See the equilibrium
  exponential script for the exact format to use. Supply the
  value [] to use the default experiment time generated by the
  program as it is a mandatory parameter.
ExpIndex - Index of the equilibrium experiment to be plotted in
  'showsingle' mode. It is required only by the 'showsingle'
  mode.

Description:
a = plotanalysis(a, 'showall', ETime); - Plots the results of
  all equilibrium experiments in equilibrium object a.
  If ETime contains valid data, then its values are used
  as the experiment time by the method.
a = plotanalysis(a, 'printall', ETime); - Plots and prints the
  results of all equilibrium experiments in equilibrium
  object a. If ETime contains valid data, then its values
  are used as the experiment time by the method.
a = plotanalysis(a, 'showsingle', ETime, ExpIndex); - Plots just
  one equilibrium experiment in the equilibrium object a
  whose index is specified by ExpIndex.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0
Classfields:
  None
6.14 print method

print(object)

Purpose:
Displays the ASCII file created by the document method so that
the information can be printed by using the print button.

Syntax:
print(object)

Arguments:
oobject - the equilibrium object

Description:
print(object) - displays the document information for printing

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
6.15 saveobject method

SPRTool\V0.9\Program\@Equilibrium

saveobject(ClassObj)

Purpose:
To save the equilibrium object to the disk.

Syntax:
saveobject(ClassObj);

Arguments:
ClassObj - the equilibrium class object to save to disk.

Description:
saveobject(ClassObj) - save the equilibrium object to disk using the fields in
the ObjectFile\Name and ObjectPath\Name fields.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
6.16 selector method

SPRTool\V0.9\Program\Equilibrium

a = selector(a, varargin)

Purpose:
To set the selection variables of the equilibrium object. You can use the selector to turn on and off the Original Data, Zero Adjusted Data, Reference Subtraction, and the Selection switches. It switches 'selected' tag on/off of sensorgram to indicate whether or not an individual sensorgram will be used in the adjust project and later analyzes

Syntax:

equilobject=selector(equilobject, 'OriginalData', selectorvector)
equilobject=selector(equilobject, 'ZeroAdjust', selectorvector)
equilobject=selector(equilobject, 'ReferenceSubtract', selectorvector)
equilobject=selector(equilobject, 'EquilibriumCutout', selectorvector)
equilobject=selector(equilobject, 'Selection', selectorvector)
equilobject=selector(equilobject, 'OriginalData', 'All')
equilobject=selector(equilobject, 'OriginalData', 'None')
equilobject=selector(equilobject, 'ZeroAdjust', 'All')
equilobject=selector(equilobject, 'ZeroAdjust', 'None')
equilobject=selector(equilobject, 'ReferenceSubtract', 'All')
equilobject=selector(equilobject, 'ReferenceSubtract', 'None')
equilobject=selector(equilobject, 'EquilibriumCutout', 'All')
equilobject=selector(equilobject, 'EquilibriumCutout', 'None')
equilobject=selector(equilobject, 'Selection', 'All')
equilobject=selector(equilobject, 'Selection', 'None')

indices is a vector of sensorgram indices. If a sensorgram index is in this vector the display properties of the corresponding sensorgram will be transposed, e.g. YDataDisplayed='no' will transposed to YDataDisplayed='yes', YDataDisplayed='yes' will be turned to YDataDisplayed='no'

   equilobject=selector(equilobject, 'OriginalData', 'transpose', indices)
equilobject=selector(equilobject, 'ZeroAdjust', 'transpose', indices)
equilobject=selector(equilobject, 'ReferenceSubtract', 'transpose', indices)
equilobject=selector(equilobject, 'EquilibriumCutout', 'transpose', indices)
equilobject=selector(equilobject, 'Selection', 'transpose', indices)
equilobject=selector(equilobject, 'DisplayNone')
equilobject=selector(equilobject, 'DisplayAll')

Arguments:
a - the equilibrium object
selectorvector - the sensorgrams to change
indices - a vector of sensorgram indices

Description:
equilobject=selector(equilobject, 'OriginalData', selectorvector)
equilobject=selector(equilobject, 'ZeroAdjust', selectorvector)
equilobject=selector(equilobject, 'ReferenceSubtract', selectorvector)
equilobject=selector(equilobject, 'EquilibriumCutout', selectorvector)
equilobject=selector(equilobject, 'Selection', selectorvector)
equilobject=selector(equilobject, 'OriginalData', 'All')
Programmer Comments:
   selectorvector is row vector whose length equals the number of
   sensorgrams in the adjust object. The kth element is 1 if the
   corresponding sensorgram is to be selected and 0 otherwise.

Algorithm:
   None

Limitations:

Method Version:
   P1.0.0

Classfields:
   None
6.17 set method

SPRTool\V0.9\Program\Equilibrium

--------------------------

varargout = set(a, varargin)

Purpose:
Sets the parent adjust object of the equilibrium class.

Syntax:
set(equ);
equ = set(equ, 'adjust', adj);

Arguments:
equ - equilibrium object
adj - parent adjust object
'adjust' - command to set the parent of the equilibrium object

Description:
set(equ) - displays the help message for equilibrium set method.
equ = set(equ, 'adjust', adj) - Sets the parent adjust object of the
equilibrium object to adj and allots an empty sensorgram structure
and an empty equilibrium experiment structure for every sensorgram in
the parent adjust object.

Programmer Comments:
None

Algorithm:
None

Limitations:
Assumes this method will only be called once with an adjust object. To
allow for multiple invocations, the code needs to make sure the number of
empty sensorgram and equilibrium experiment structures allotted be
equal to the number of sensorgrams in the parent adjust object.

Method Version:
P1.0.0

Classfields:
None
6.18 subsasgn method

SPRTool\V0.9\Program\\Equilibrium

a = subsasgn(a, index, val)

Purpose:
The subsasgn function for the Equilibrium class. Used to set fields in
an Equilibrium object.

Syntax:
a = subsasgn(a, index, val);
classobject.fieldname = val;

Arguments:
a - the object having value set
index - the command structure
val - value to set field
classobject - the object name
fieldname - the objects field name to alter

Description:
a = subsasgn(a, index, val) - Using this approach you must supply the
index structure which is described in the MATLAB documentation under
classes.

classobject.fieldname = val - This is the preferred method of setting
the field values. This way the index structure is generated for you.
All you need to do is issue a command like 'classobject.fieldname =
val'. Let's say we have a object of class Equilibrium called 'equ'. It
has a field called 'EquilComments' and we wish to set it to a value of
'none'. You would issue the command "equ.EquilComments = 'none';". This
would set EquilComments to the value 'none' and return the updated object.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
6.19 subsref method

b = subsref(a, index)

Purpose:
The subsref function for the Equilibrium class. Used to get field values in an Equilibrium object.

Syntax:
b = subsref(a, index);
val = classobject.fieldname;

Arguments:
a - the object to get value from
b - the returned field value
index - the command structure
val - value to get from field
classobject - the object name
fieldname - the objects field name to get value from

Description:
b = subsref(a, index) - Using this approach you must supply the index structure which is described in the MATLAB documentation under classes.

val = classobject.fieldname - This is the preferred method of getting the field values. This way the index structure is generated for you.
All you need to do is issue a command like 'val = classobject.fieldname.'

Let's say we have a object of class equilibrium called 'equ.' It has a field called 'EquilFileName' and we wish to get the value from.
You would issue the command 'val = equ.EquilFileName'. This would get EquilFileName and set the variable 'val' to the value of the field.

Programmer Comments:
None

Algorithm:
None

Limitations:
Can only generate a default object

Method Version:
P1.0.0

Classfields:
None
6.20  listboxstring method

SPRTool\V0.9\Program\@Equilibrium\private

liststring=listboxstring(equilobject, specification)

Purpose: EQUILIBRIUM/private produces a string that is to be used in the selector string for
the listbox.

Syntax:
liststring=listboxstring(equilobject,’OriginalData’)
liststring=listboxstring(equilobject,’ZeroAdjust’)
liststring=listboxstring(equilobject,’ReferenceSubtract’)
liststring=listboxstring(equilobject,’EquilibriumCutout’)

Arguments:
equilobject - the Equilibrium object

Description:
liststring=listboxstring(equilobject,’OriginalData’) - produces a proper
string for original data
liststring=listboxstring(equilobject,’ZeroAdjust’) - produces a proper
string for zeroadjust data
liststring=listboxstring(equilobject,’ReferenceSubtract’) - produces a proper
string for referencesubtract data
liststring=listboxstring(equilobject,’EquilibriumCutout’) - produces a proper
string for equilibriumcutout data

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
Chapter 7

Experiment
7.1 Experiment constructor

SPRTool\V0.9\Program\@Experiment

-------------------------------------------------------------------------------

e = Experiment(varargin)

Purpose:
The purpose of the experiment object is to keep track of the data acquired from the Biacore machine. It will hold all the data for a single experiment.

Syntax:
e = Experiment;
e = Experiment(num_runs);

Arguments:
e - Experiment object
num_runs - Number of run info structures to create

Description:
e = Experiment; - Generates a default experiment object that has one run info structure
e = Experiment(num_runs); - Generates an experiment object with however many run info structures as specified by num_runs.

Programmer Comments:
None

Algorithm:
None

Limitations:
Can only generate a default object

Method Version:
P1.0.0

Classfields:

exp._ObjectCreator = getenv('UserName'); %{string} object creator
exp._ObjectPathName = 'NA-str'; %{string} path for the object
exp._ObjectFileName = 'NA-str'; %{string} filename for the object
exp._ObjectSetupDate = datestr(now); %{datum} time stamp
exp._Description = 'Experiment Object'; %{data object} description of the object
exp. Comments = 'NA-str'; %{string} general comments about the object
exp. UserData = 'NA-cell'; %{cell array} user data section for storage
exp. FieldExtensions = 'NA-cell'; %{cell array} user defined field extensions
exp. History = 'NA-cell'; %{cell array} keep track of what is done to the object
exp. Error = 'NA-str'; %{string} to keep track of error messages
exp. MaxDataPoints = 'NA-n'; %{integer} max number of data points
exp. NumberSensorgrams = 'NA-n'; %{integer} number of sensorgrams
exp. NumberRuns = 'NA-n'; %{integer} number of runs
exp. OriginalFileName = 'NA-str'; %{string} original DATA file name
exp.ExperimentDate = 'NA-str';  %{string} date of experiment, %format: DD/MM/YYYY
exp.PurposeOfExperiment = 'NA-str';  %{string} purpose of the experiment
exp.ExperimentOperator = 'NA-str';  %{string} the experiment operator
exp.ExperimentComments = 'NA-str';  %{string} any experiment comments
exp.ExpFileName = 'NA-str';  %{string} experiment file name
exp.Run(k).Flowrate = 'NA-d';  %{double} flow rate of the analyte
exp.Run(k).FlowrateUnits = 'NA-str';  %{string} units of the flow rate
exp.Run(k).WhatInjected = 'NA-str';  %{string} what was injected into the flow
exp.Run(k).Concentration = 'NA-d';  %{double} Concentration level of the %analyte
exp.Run(k).ConcentrationUnits = 'NA-str';  %{string} The units for the concentration
exp.Run(k).UniformlySampled = 'NA-str';  %{string} ('yes'/'no') was the data %uniformly sampled
exp.Run(k).SamplingRate = 'NA-d';  %{double} the rate of sampling for the data
exp.Run(k).SamplingRateUnits = 'NA-str';  %{string} the units for sampling
exp.Run(k).SamplingInterval = 'NA-d';  %{double} the sampling interval
exp.Run(k).SamplingIntervalUnits = 'NA-str';  %{string} the units for the sampling %interval
exp.Run(k).Temperature = 'NA-d';  %{double} the temperature
exp.Run(k).TemperatureUnits = 'NA-str';  %{string} the temperature units
exp.Run(k).Trace(n).XData = 'NA-d';  %{double} the x axis position of data %point(s)
exp.Run(k).Trace(n).XDataUnits = 'NA-str';  %{string} the units of the x axis
exp.Run(k).Trace(n).XDataBiacoreHeader = 'NA-str';  %{string} the x axis data biacore header
exp.Run(k).Trace(n).YData = 'NA-d';  %{double} the y axis position of data %point(s)
exp.Run(k).Trace(n).YDataUnits = 'NA-str';  %{string} the units of the y axis
exp.Run(k).Trace(n).YDataBiacoreHeader = 'NA-str';  %{string} the y axis data biacore header
exp.chip = 'NA-obj';  %{object} the default parent chip object
7.2 display method

SPRTool\V0.9\Program\@Experiment

----------------------------------------

display(e, varargin)

Purpose:
This is the display function for the experiment object. It will display the fields in a formatted fashion.

Syntax:
    display(e);

Arguments:
e - the experiment class input

Description:
display(e); - Displays the formatted field data to the command window.

Programmer Comments:
    None

Algorithm:
    None

Limitations:
    None

Method Version:
    P1.0.0

Classfields:
    None
7.3 document method

SPRTool\V0.9\Program\@Experiment

--------------------------------------------------------

document(object, varargin)

Purpose:
  Print the experiment information to a .m file for later analysis

Syntax:
  document(object);

Arguments:
  object - the experiment object

Description:
  document(object) - generates a .m file with the experiment object fields displayed

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
7.4 equalizedatalength method

expobject = equalizedatalength(expobject)

Purpose:
Takes an experiment object already loaded with data and makes sure that
data from all traces of the same run are of equal length by padding
shorter data vectors with zeros.

Syntax:
expobject = equalizedatalength(expobject);

Arguments:
expobject - the experiment object

Description:
expobject = equalizedatalength(expobject); - Takes expobject and makes
sure that data from all traces of the same run are of
equal length by padding shorter data vectors with zeros.

Programmer Comments:
This method assumes that the experiment object passed in has been loaded
with data.

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
7.5 guiloadobject method

SPRTool\V0.9\Program\@Experiment

ClassObj = guiloadobject(ClassObj)

Purpose:
To load the experiment object from the disk with the help of a gui interface.

Syntax:
ClassObj = guiloadobject(ClassObj)

Arguments:
ClassObj - the experiment class object to save to disk.

Description:
ClassObj = guiloadobject(ClassObj) - load the experiment object to disk using the
gui returned values for path and filename.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
7.6 guiplotdata method

SPRTool\V0.9\Program\Experiment

---

guiplotdata(exp)

Purpose:
To display the raw sensorgram data in a RU/seconds plot with the help of a graphical interface to control the display of the plots and which plots will be displayed.

Syntax:
guiplotdata(exp);

Arguments:
exp - the experiment object

Description:
guiplotdata(exp) - The explanation I will give here will be the same as the one given in the Tutorial Manual that supplements the program.

When the gui starts you will see six grey buttons, one blue button, two sliders, and a plot window. First lets try one of the grey buttons and see how it works.
The Show All 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 into 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 right click onto the area of the sensorgram plot that you wish to examine. To zoom back out again just left click the mouse over the plot window. Now zoom into one of the sensorgram plots. When you can see it adequately in the plot window press the magnifier button again to deactivate the zoom feature of the figure.

If you wish to display detailed information about a sensorgram all you have to do is click on the blue button. 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... Note: You must make sure that the magnifier button is not depressed or else the information figure will not be propagated. You can at anytime 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.

You will notice that in {\bf 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. Now click on the {\bf Single} button. In this display mode you have the ability to view each sensorgram individually. Now you will see both slider bars have reappeared to the left. One of the bars is called {\bf Run} while the other is called {\bf Trace}. You will also notice that for each slider there are three numbers associated with that slider. The top number is the maximum number of runs/traces. The bottom number is the first run/trace, which is usually equal to one. The side number is the current run/trace being display to the plot window. When you first start Single mode the current run/trace will be set to 1.
So what is a run and a trace? A run essentially corresponds to one injection of analyte over the flow cell. The trace corresponds to the channel. For some BIA machines this might be 1, 2, or 4. This software was developed using a 4 channel BIACore machine. For the purpose of this tutorial we will have 4 runs and 4 traces. You will see that the maximum run value is 4 and the maximum trace value is 4. 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.

Another very useful button for analyzing your sensorgram data is Single Parallel}. This mode will display all runs of a specified trace with setting the start time to zero for each run. This way you can compare all runs of a trace to each other in a sort od Stack format. Another tool of interest in this regard is the Single Trace. This will display the runs of a single trace over time without setting each sensorgram plot start to zero. You can change the trace you view by using the slider bar.

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 display for the superfluous traces. This is just how the program works and is a feature not an error:).

Programmer Comments:
1) The callback function for the gui is called exp_viewer.m and is located in the 'mfiles' directory. This file will show how each button and control was implemented.

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

ClassFields:
None
7.7 guisaveobject method

SPRTool\V0.9\Program\0Experiment

---------------------------------------------------------------
guisaveobject(ClassObj)

Purpose:
To save the experiment object from the disk with the help of a gui interface.

Syntax:
guisaveobject(ClassObj);

Arguments:
ClassObj - the experiment class object to save to disk.

Description:
ClassObj = guisaveobject(ClassObj) - save the experiment object to disk using
the value supplied by the gui interface.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
7.8 loaddata method

SPRTool\V0.9\Program\@Experiment

exp = loaddata(exp, varargin)

Purpose:
Load the data from text file or excel worksheet. updates fields accordingly

Syntax:
exp = loaddata(exp);
exp = loaddata(exp, path, file);

Arguments:
exp - the experiment object
path - the path name
file - the file name

Description:
exp = loaddata(exp) - load the file specified in the object or if no file has ever been loaded then a gui is provided to select the data file to use. The file path and name are then stored in the object for later retrieval.

exp = loaddata(exp, path, file) -

Programmer Comments:
None

Algorithm:
None

Limitations:
Can only generate a default object

Method Version:
P1.0.1

Classfields:
None
7.9  loadobject method

SPRTool\V0.9\Program\@Experiment

ClassObj = loadobject(ClassObj, varargin)

Purpose:
  To load the chip object from the disk.

Syntax:
  ClassObj = loadobject(ClassObj);
  ClassObj = loadobject(ClassObj, filename);

Arguments:
  ClassObj - the chip class object to save to disk.
  filename - a string containing the filename of object to load. Please
  include the file extension in the name.

Description:
  ClassObj = loadobject(ClassObj) - load the chip object to disk using the
  values in ObjectFileName and ObjectPathName fields or filename variable.

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
7.10 plotdata method

Purpose:
To display the raw sensorgram data in a RU/seconds plot. There are many choices of data display

Syntax:
plotdata(a,'showall');
plotdata(a,'showsingleplot', run, trace);
plotdata(a,'showsingletrace', trace);
plotdata(a,'showsinglerun', run);
plotdata(a,'showtraceparallel', trace);
plotdata(a,'showsingletraceparallel', trace);

Arguments:
a - experiment object
run - a specific run number
trace - a specific trace number

Description:
plotdata(a,'showall') - will plot all runs/traces to a single display window.
plotdata(a,'showsingleplot', run, trace) - will display a specific sensorgram to the display window.
plotdata(a,'showsingletrace', trace) - shows a single trace over time
plotdata(a,'showsinglerun', run) - shows a single run
plotdata(a,'showtraceparallel', trace) - shows all traces in parallel
plotdata(a,'showsingletraceparallel', trace) - shows a specific trace in parallel.

Programmer Comments:
1) If you have a display window with the 'Tag' field called 'Tag_plotwindow' then the plot will be pasted into that window instead of in a new figure.

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
7.11 print method

SPRTool\V0.9\Program\@Experiment

print (object)

Purpose:
None

Syntax:
None

Arguments:
None

Description:
None

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
7.12 saveobject method

saveobject(ClassObj)

Purpose:  
   To save the chip object to the disk.

Syntax:  
   saveobject(ClassObj);

Arguments:  
   ClassObj - the chip class object to save to disk.

Description:  
   saveobject(ClassObj) - save the chip object to disk using the fields in  
   the ObjectFileName and ObjectPathName fields.

Programmer Comments:  
   None

Algorithm:  
   None

Limitations:  
   None

Method Version:  
   P1.0.0

Classfields:  
   None
7.13 subsasgn method

SPRTool\V0.9\Program\0Experiment

---

a = subsasgn(a,index,val)

Purpose:
The subsasgn function for the experiment class. Used to set fields in
the an experiment object.

Syntax:
a = subsasgn(a,index,val);
classobject.fieldname = val;

Arguments:
a - the object having value set
index - the command structure
val - value to set field
classobject - the object name
fieldname - the objects field name to alter

Description:
a = subsasgn(a,index,val) - Using this approach you must supply the
index structure which is described in the MATLAB documentation under
classes.

classobject.fieldname = val - This it the perfered method of setting
the field values. This way the index structure is generated for you.
All you need to do is issue a command like 'classobject.fieldname =
val'. Lets say we have a object of class experiment called 'exp'. It
has a field called 'NumberRuns' and we wish to set it to a value of 2.
You would issue the command 'exp.NumberRuns = 2;'. This would set
NumberRuns to a value of 2 and return the updated object.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
7.14 subsref method

b=subsref(a,index)

Purpose:
The subsref function for the experiment class. Used to get field values in an experiment object.

Syntax:
b=subsref(a,index);
val = classobject.fieldname;

Arguments:
a - the object to get value from
b - the returned field value
index - the command structure
val - value to get from field
classobject - the object name
fieldname - the objects field name to get value from

Description:
b=subsref(a,index) - Using this approach you must supply the index structure which is described in the MATLAB documentation under classes.

val = classobject.fieldname - This is the preferred method of getting the field values. This way the index structure is generated for you. All you need to do is issue a command like 'val = classobject.fieldname'. Let's say we have an object of class experiment called 'exp'. It has a field called 'NumberRuns' and we wish to get the value from. You would issue the command 'val = exp.NumberRuns'. This would get NumberRuns and set the variable 'val' to the value of the field.

Programmer Comments:
None

Algorithm:
None

Limitations:
Can only generate a default object

Method Version:
F1.0.0

Classfields:
None
Chapter 8

IndependentInteractionsModel
8.1 IndependentInteractionsModel constructor

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\IndependentInteractionsModel

---

iim = IndependentInteractionsModel(varargin)

Purpose: To create a default independent interactions model

Model Equations:
  Association - The equation used to evaluate kth association interaction is
  \[ Y_k = \text{Req}_k \cdot (1 - \exp(-kobs_k(t-t0+t_shift))) + \text{offset}_k; \]
  Dissociation - The equation used to evaluate kth dissociation interaction is
  \[ Y_k = R0 \cdot \exp(-koff_k(t-t0)) + \text{offset}_k; \]

Syntax:
  iim = IndependentInteractionsModel;

Arguments:
  iim - the IndependentInteractionsModel class object
  kon = Association rate constant
  koff = Dissociation rate constant
  C = Concentration of analyte that flows over the chip
  Rmax = (koff+kon*C)/(kon*C)*Req, Maximum analyte binding capacity
  kobs = (kon * C) + koff
  Req = Equivalent analyte binding
  R0 = Starting signal of the chosen dissociation data set.

Description:
  iim = IndependentInteractionsModel - Construct a default
  IndependentInteractionsModel class object

Programmer Comments:
  If the class is not a default class no more change on the
  NumberOfInteractions

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  iim.ModelType = 'NA-str';  %{string} Association/Dissociation/fullcurve
  iim.FixedC = 'NA-d';      %{double} Fixed C value
  iim.FixedCUnits = 'NA-str';  %{'nM'} units for Fixed C
  iim.AssociationStartTime = 'NA-d';  %{double} Association curve starting time.
  %It is assigned internally
  iim.AssociationStartTimeUnits = 'NA-str';  %{'s'} Unit of association curve starting
  %time. It is assigned internally
  iim.AssociationEndTime = 'NA-d';  %{double} Association curve ending time.
  %It is assigned internally
  iim.AssociationEndTimeUnits = 'NA-str';  %{'s'} Unit of association curve ending
  %time. It is assigned internally
  iim.DissociationStartTime = 'NA-d';  %{double} Dissociation curve starting time.
  %It is assigned internally
  iim.DissociationStartTimeUnits = 'NA-str';  %{'s'} Unit of dissociation curve starting
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iim.DissociationEndTime</td>
<td>'NA-d'; time. It is assigned internally</td>
</tr>
<tr>
<td>iim.DissociationEndTimeUnits</td>
<td>'NA-str'; '%{double} Dissociation curve ending time.</td>
</tr>
<tr>
<td>iim.NumberOfInteractions</td>
<td>= 1; '%{double} total no of protein interactions</td>
</tr>
<tr>
<td>iim.DefaultClass</td>
<td>= 'yes' '%{yes'</td>
</tr>
<tr>
<td>iim.AssociationParameters(k).kobs</td>
<td>= 'NA-d'; '%{double} kobs values</td>
</tr>
<tr>
<td>iim.AssociationParameters(k).kobsUnits</td>
<td>= 'NA-str'; '%{1/s} units for kobs</td>
</tr>
<tr>
<td>iim.AssociationParameters(k).kon</td>
<td>= 'NA-d'; '%{double} kon values</td>
</tr>
<tr>
<td>iim.AssociationParameters(k).konUnits</td>
<td>= 'NA-str'; '%{1/(nM)s} units for kon</td>
</tr>
<tr>
<td>iim.AssociationParameters(k).Req</td>
<td>= 'NA-d'; '%{double} Req values</td>
</tr>
<tr>
<td>iim.AssociationParameters(k).ReqUnits</td>
<td>= 'NA-str'; '%{RU} units for Req</td>
</tr>
<tr>
<td>iim.DissociationParameters(k).koff</td>
<td>= 'NA-d'; '%{double} koff values</td>
</tr>
<tr>
<td>iim.DissociationParameters(k).koffUnits</td>
<td>= 'NA-str'; '%{1/s} units for koff</td>
</tr>
<tr>
<td>iim.DissociationParameters(k).RO</td>
<td>= 'NA-d'; '%{double} RO values</td>
</tr>
<tr>
<td>iim.DissociationParameters(k).ROUnits</td>
<td>= 'NA-str'; '%{RU} units for RO</td>
</tr>
<tr>
<td>iim.AssociationParametersSmallestEigenvalue</td>
<td>= 'NA-d'; % Smallest eigenvalue of DOS A matrix; filled in by % dos2iint</td>
</tr>
<tr>
<td>iim.AssociationParametersReqTotal</td>
<td>= 'NA-d'; % Req value corresponding to the smallest eigenvalue of DOS A % matrix; filled in by dos2iint</td>
</tr>
<tr>
<td>iim.AssociationParametersReqTotalUnits</td>
<td>= 'NA-str'; '%{s} units for ReqTotal; filled in % by dos2iint</td>
</tr>
<tr>
<td>iim.ComplexFlag</td>
<td>= 'NA-str'; '%{ok'</td>
</tr>
<tr>
<td>iim.TimeShift</td>
<td>= 0 % time shift calculated internally</td>
</tr>
<tr>
<td>iim.TimeShiftUnits</td>
<td>= 's' '%{s} Unit of time shift calculated internally based on the estimation</td>
</tr>
<tr>
<td>iim.MaximumBindingCapacity</td>
<td>= 'NA-d' % maximum analyte binding capacity</td>
</tr>
<tr>
<td>iim.MaximumBindingCapacityUnits</td>
<td>= 'NA-str' '%{RU} Unit of maximum analyte binding capacity</td>
</tr>
</tbody>
</table>
8.2 calculatekobs method

```
SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\IndependentInteractionsModel

iim = calculatekobs(iim)

Purpose: To calculate iim.Parameters(k).kobs from
   iim.Parameters(k).kon, iim.Parameters(k).koff and iim.FixedC.

Syntax:
iim = calculatekobs(iim)

Arguments:
iim - the IndependentInteractionsModel class object
kon = Association rate constant
kobs = (kon * C) + koff

Description:
iim = calculatekobs(iim) - To calculate iim.Parameters(k).kobs from iim.Parameters(k).kon,
iim.Parameters(k).koff and iim.FixedC.

Programmer Comments:
None

Algorithm:
kobs = (kon * C) + koff

Limitations:
1. Currently the algorithm works for one independent interaction only

Method Version:
P1.0.0

Classfields:
None
```
8.3 calculatekon method

SPRTool\V0.9\Program\bodatanalysis\V0.9\classes\models\0IndependentInteractionsModel

iim = calculatekon(iim)

Purpose: To calculate associate rate constant iim.Parameters(k).kon from iim.Parameters(k).kobs, iim.Parameters(k).koff and iim.FixedC.

Syntax:
   iim = calculatekon(iim)

Arguments:
   iim - the IndependentInteractionsModel class object
   kon = Association rate constant
   kobs = (kon * C) + koff

Description:
   iim = calculatekon(iim) - calculate associate rate constant iim.Parameters(k).kon from iim.Parameters(k).kobs iim.Parameters(k).koff and iim.FixedC.

Programmer Comments:
   None

Algorithm:
   kon = (kobs-koff)/C

Limitations:
   1. Currently the algorithm works for one independent interaction only

Method Version:
   P1.0.0

Classfields:
   None
8.4 calculateReq method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\IndependentInteractionsModel

iim = calculateReq(iim)

Purpose: To calculate the Req

Syntax:
    iim = calculateReq(iim)

Arguments:
    iim - the IndependentInteractionsModel class object

Description:
    iim = calculateReq(iim) - To calculate Req

Programmer Comments:
    None

Algorithm:
    Req = kon*C*Rmax/(ka*C+koff)

Limitations:
    1. Currently the algorithm works for one independent interaction only

Method Version:
    P1.0.0

Classfields:
    None
8.5 calculatormax method

SPRTool\V0.9\Program\biodataanalysis\V0.9\classes\models\IndependentInteractionsModel

iim = calculatormax(iim)

Purpose: To calculate the maximum binding capacity

Syntax:

    iim = calculatormax(iim)

Arguments:

    iim - the IndependentInteractionsModel class object

Description:

    iim = calculatormax(iim) - To calculate the maximum binding capacity of the surface ligand for the analyte in the test.

Programmer Comments:

    None

Algorithm:

    Rmax = (koff+kon*C)/(kon*C)*Req

Limitations:

    1. Currently the algorithm works for one independent interaction only

Method Version:

    P1.0.0

Classfields:

    None
8.6 display method

SPRTool\V0.9\Program\biodatabas\V0.9\classes\models\IndependentInteractionsModel

-----------------------------------------------

display(object, varargin)

Purpose: To display information about the IndependentInteractionsModel class object

Syntax:
   function display(object)

Arguments:
   iim - IndependentInteractionsModel class object

Description:
   display(object) - Gets the IndependentInteractionsModel class object and displays the
   information contained in it. Also Matlab calls this display method whenever
   an object is the result of a statement that is not terminated by a semicolon.

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.0

Classfields:
   None
8.7 document method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\0IndependentInteractionsModel

document(object, varargin)

Purpose:
Print the Independent Interaction information to a .m file for
later analysis and printing

Syntax:
document(object);

Arguments:
object - the Mass Transport object

Description:
document(object) - generates a .m file with the MassTransport
object fields displayed

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
8.8 guiloaderobject method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\IndependentInteractionsModel

-------------------------------------------------

ClassObj = guiloaderobject(ClassObj)

Purpose:
To load the IndependentInteractionsModel object from the disk with the help of a gui interface.

Syntax:
ClassObj = guiloaderobject(ClassObj)

Arguments:
ClassObj - the IndependentInteractionModel class object to save to disk.

Description:
ClassObj = guiloaderobject(ClassObj) - load the IndependentInteractionModel object to
disk using the gui returned values for path and filename.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
8.9 guisaveobject method

```
ClassObj = guisaveobject(ClassObj)
```

**Purpose:**
- save the object

**Syntax:**
```
ClassObj = guisaveobject(ClassObj)
```

**Arguments:**
- `ClassObj` - a `IndependentInteractionsModel` object

**Description:**
```
ClassObj = guisaveobject(ClassObj) - save the object via GUI
```

**Programmer Comments:**
- None

**Algorithm:**
- None

**Limitations:**
- None

**Method Version:**
- P1.0.0

**Classfields:**
- None
8.10  **loadobject method**

ClassObj = loadobject(ClassObj, varargin)

**Purpose:**
To load the IndependentInteractionsModel object from the disk.

**Syntax:**
```
ClassObj = loadobject(ClassObj);
ClassObj = loadobject(ClassObj, filename);
```

**Arguments:**
- `ClassObj` - the IndependentInteractionsModel class object to save to disk.
- `filename` - a string containing the filename of object to load. Please include the file extension in the name.

**Description:**
ClassObj = loadobject(ClassObj) - load the IndependentInteractionsModel object to disk using the values in `model.ObjectFileName` and `model.ObjectPathName` fields or `filename` variable.

**Programmer Comments:**
None

**Algorithm:**
None

**Limitations:**
None

**Method Version:**
P1.0.0

**Classfields:**
None
8.11 print method

print(object)

Purpose:
Displays the ASCII file created by the document method so that the information can be printed by using the print button.

Syntax:
print(object)

Arguments:
object - the Model object

Description:
print(object) - displays the document information for printing

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
8.12 `saveobject` method

saveobject(ClassObj)

Purpose:
To save the `IndependentInteractionModel` object to the disk.

Syntax:
saveobject(ClassObj);

Arguments:
ClassObj - the `IndependentInteractionModel` class object to save to disk.

Description:
saveobject(ClassObj) - save the `IndependentInteractionModel` object to disk using the fields in the model.ObjectFileName and model.ObjectPathName fields.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
8.13 setparameters method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\IndependentInteractionsModel

iim = setparameters(iim, val, varargin)

Purpose: To set the parameters based on various cases.

Syntax:
    iim = setparameters(iim, val, varargin)

Arguments:
    iim - the IndependentInteractionsModel class object
    val - (cell array) val{k} are parameter vectors for kth interaction.
        Association case: [kobe(1/s), Req(RU)]
        Dissociation case: [koff(1/s), R0(RU)]
        FullCurve: [kon(1/(mMs)) koff(1/s) Rmax(RU)]

TypicalValue - (cell array), val{k}.*TypicalValue(k) are the right parameters value
        TypicalValue are mainly for optimization algorithm, i.e. lsqnonlin.

DataSetNumber: additional information may be available in
  iim.model.SimulationInfoDataobject.Set(DataSetNumber)..
  For example, the concentration information saved in
  dataobject.Set(DataSetNumber).UserData

Description:
    iim = setparameters(iim, val, varargin) - To set the parameters based on various cases.

Programmer Comments:
    Making val as a cell array so that setparameters may have the same
    format for other models too

Algorithm:

Limitations:
  1. It works for 1 interaction only now

Method Version:
    P1.0.0

Classfields:
    None
8.14 simulate method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\0IndependentInteractionsModel

-----------------------------------------------------------------------------------------------

iim = simulate(iim, varargin)

Purpose: To simulate the system output with the model parameters stored in
the IndependentInteractionsModel object

Syntax:
  iim = simulate(iim);
  iim = simulate(iim, datasetno);

Arguments:
  iim - IndependentInteractionsModel class object
datasetno - the dataset number to be simulated

Description:
  iim = simulate(iim) : Creates a vector, yim, of simulated data based on the
parameters in the IndependentInteractionsModel object, iim, and updates
the object with the simulated data into the SimulationDataObject of the
modelobject. This method obtains the time from NumberOfPointsXData from
the SimulationInfoData object of modelobject.
  iim = simulate(iim, datasetno) : Creates a vector, yim, of simulated data based on
the parameters in the IndependentInteractionsModel object, iim, and updates
the object with the simulated data into the SimulationDataObject of the
modelobject. This method simulates the y output for the particular
datasetno passed in as argument. It obtains the time from NumberOfPointsXData
from the SimulationInfoData object of modelobject.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
8.15 **subsasgn method**

```plaintext
iim = subsasgn(iim,index,val)
```

**Purpose:** Evaluates the value read in and sets the field correctly, it will give an error if the field is set to an invalid value.

**Syntax:**
```
iim = subsasgn(iim,index,val)
```

**Arguments:**
- `iim` - `IndependentInteractionsModel` class object
- `index` - with two fields `index.type` and `index.subs`
  - `index.type` : string containing `'('`, `'{}`, or `.'` specifying subscript type
  - `index.subs` : call array or string containing the actual subscripts
- `val` - new value

**Description:**
```
iim = subsasgn(iim,index,val) - Obtains the argument values and and assigns 'val'
    to the field of IndependentInteractionsModel object iim
```

**Programmer Comments:**
```
None
```

**Algorithm:**
```
None
```

**Limitations:**
```
None
```

**Method Version:**
```
P1.0.0
```

**ClassFields:**
```
None
```
8.16 subsref method

b = subsref(iim,index)

Purpose: Evaluates the value read in and returns the field value

Syntax:
  b = subsref(iim,index)

Arguments:
  iim - IndependentInteractionsModel class object
  index - with two fields index.type and index.subs
    index.type : string containing ’()’,’{ }’, or ’.’ specifying subscript type
    index.subs : call array or string containing the actual subscripts

Description:
  b = subsref(iim,index) - Obtains the IndependentInteractionsModel object and the
    index, and evaluates the value read in

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
8.17 view method

view(iim, varargin)

Purpose: The purpose of this method is to plot the datasets of the SimulationDataObject in the Model class object

Syntax:
view(iim);
view(iim, datasetno);

Arguments:
- iim - Type(IndependentInteractionsModel), the IndependentInteractionsModel object
- datasetno - Type(double), the dataset number to be simulated

Description:
- view(iim) - Obtains the DataClass object, SimulationDataObject, from the modelobject and passes to the view function of the DataClass to plot the first dataset. Click on previous or next dataset button to view the respective datasets.
- view(iim, datasetno) - Obtains the DataClass object, SimulationDataObject, from the modelobject and passes to the view function of the DataClass to plot the particular data set passed in as argument. Click on previous or next dataset button to view the respective datasets.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
Chapter 9

InjectTimes
9.1 CallInjectTimeFcn method

SPRTool\V0.9\Program\InjectTimes

------------------------------------------

function adj = CallInjectTimeFcn(inj, adj)

Purpose:
   Will call the inject times function specified in the InjectTimeFcn
   field of the injecttimes class. Will perform the inject time calculations
   and return the adjjust object.

Syntax:
   adj = CallInjectTimeFcn(inj, adj);

Arguments:
   adj - the adjust object
   inj - the injecttimes object

Description:
   adj = CallInjectTimeFcn(inj, adj) - Evaluates the method handle stored
   in the InjectTimeFcn field. Does calculation of injection times and
   returns the adjust object with injection parameters entered.

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.0

Classfields:
   None
9.2 InjectTimeFcn method

SPRTool\V0.9\Program\0InjectTimes

---

function adjustobject = InjectTimeFcn(inj, adjustobject)

Purpose:
Identifies approximate start and endpoint of injection in sensorgram data.
Returns these points to fields in the adjustobject

Syntax:
a = InjectTimeFcn(i, a);

Arguments:
a - adjust object
i - injecttimes object

Description:
a = InjectTimeFcn(i, a) - Calculates the injection start and stop times for the adjust data.

Programmer Comments:
You have to run the zero adjust on the adjust object first.

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
9.3 InjectTimes constructor

SPRTool\V0.9\Program\@InjectTimes

function i = InjectTimes

Purpose:
The constructor for the InjectTimes class. Used to calculate and store
the analyte injection time points. Will store the start and stop points
for each sensorgram.

Syntax:
i = InjectTimes;

Arguments:
i - the InjectTimes object

Description:
i = InjectTimes - will generate the default InjectTimes object

Programmer Comments:
None

Algorithm:
None

Limitations:
Can only generate a default object

Method Version:
P1.0.0

Classfields:
i.ObjectCreator = getenv('UserName'); %#string object creator
i.ObjectPathName = 'NA-str'; %#string path for the object
i.ObjectFileName = 'NA-str'; %#string filename for the object
i.ObjectSetupDate = datestr(now); %#datum time stamp
i.Description = 'Adjust Object'; %#object description of the object
i.Comments = 'NA-str'; %#string general comments about the object
i.UserData = 'NA-cell'; %#cell array user data section for structure
age
i.FieldExtensions = 'NA-cell'; %#cell array user defined field extensions
to the object
i.Error = 'NA-str'; %#string to keep track of error messages

i.HowToGetTimes = 'NA-str'; %#string {UserAssignment | GUISelection}
i.InjectTimeFcn = 'NA-fcn'; %#function handle @InjectTimeFcn function
on handle
i.InjectFcn_preview = 'NA-str'; %#string 'yes' {'yes' | 'no'} do you want to preview
at to preview
i.InjectFcn_PreInjectionInds = 'NA-d'; %#double [] pre-injection indicies
i.InjectFcn.InjectionInds = 'NA-d'; %{double} [] injection indices
i.InjectFcn.PostInjectionInds = 'NA-d'; %{double} [] post injection indices
i.InjectFcn.InterpMethod = 'NA-str'; %{string} 'linear' {'nearest','linear',
,'spline','cubic'} type of extrapolation
i.InjectFcn.ExtrapolationResolution = 'NA-d'; %{double} [.01] extrapolation resolution
i.InjectFcn.Tolerance = 'NA-d'; %{double} [.05] tolerance

i.TotalNumberOfSensorgrams = 'NA-d'; %{double} 1 Stores number of sensorgrams
i.NumberOfFlowCells = 'NA-d'; %{double} 1 Stores number of flow cells
i.StartPoints.FlowCell = 'NA-d'; %{double} [] stores start points of flow cells
i.StopPoints.FlowCell = 'NA-d'; %{double} [] stores stop points of flow cells
i.StartPoints.Sensorgram = 'NA-d'; %{double} [] stores start points of sensorgrams
9.4 display method

function display(c)

Purpose:
This is the display function for the InjectTimes object.
It will display the fields in a formatted fashion.

Syntax:
display(c);

Arguments:
c - the InjectTimes class input

Description:
display(c); - Displays the formatted field data to the command window.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
9.5  document method

function document(inj, varargin)

Purpose:
Print InjectTimes information to a .m file for later analysis

Syntax:
document(inj)

Arguments:
inj - the InjectTimes object

Description:
document(inj) - generates a .m file with the InjectTimes object fields displayed

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
9.6 print method

SPRTool\V0.9\Program\@InjectTimes

-----------------------------------------------

function print(object)

Purpose:
    None

Syntax:
    None

Arguments:
    None

Description:
    None

Programmer Comments:
    None

Algorithm:
    None

Limitations:
    None

Method Version:
    P1.0.0

Classfields:
    None
function a = subsasgn(a,index,val)

Purpose:
The subsasgn function for the injecttimes class. Used to set fields in
the injecttimes object.

Syntax:
a = subsasgn(a,index,val);
classobject.fieldname = val;

Arguments:
a - the object having value set
index - the command structure
val - value to set field
classobject - the object name
fieldname - the objects field name to alter

Description:
a = subsasgn(a,index,val) - Using this approach you must supply the
index structure which is described in the MATLAB documentation under
classes.

classobject.fieldname = val - This it the prefered method of setting
the field values. This way the index structure is generated for you.
All you need to do is issue a command like 'classobject.fieldname =
val'. Lets say we have a object of class experiment called 'exp'. It
has a field called 'NumberRuns' and we wish to set it to a value of 2.
You would issue the command 'exp.NumberRuns = 2;'. This would set
NumberRuns to a value of 2 and return the updated object.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
9.8 subsref method

function b=subsref(a,index)

Purpose:
The subsref function for the InjectTimes class. Used to get field values
in an InjectTimes object.

Syntax:
b=subsref(a,index);
val = classobject.fieldname;

Arguments:
a - the object to get value from
b - the returned field value
index - the command structure
val - value to get from field
classobject - the object name
fieldname - the objects field name to get value from

Description:
b=subsref(a,index) - Using this approach you must supply the
index structure which is described in the MATLAB documentation under
classes.

val = classobject.fieldname - This it the preferred method of getting
the field values. This way the index structure is generated for you.
All you need to do is issue a command like 'val = classobject.fieldname.
Lets say we have a object of class experiment called 'exp'. It
has a field called 'NumberRuns' and we wish to get the value from.
You would issue the command 'val = exp.NumberRuns'. This would get
NumberRuns and set the variable 'val' to the value of the field.

Programmer Comments:
None

Algorithm:
None

Limitations:

Method Version:
F1.0.0

Classfields:
None
Chapter 10

Mass Transport Comp Model
10.1 MassTransportCompModel constructor

SPRTool\v0.9\Program\biodataanalysis\v0.9\classes\models\@MassTransportCompModel

m = MassTransportCompModel(varargin)

Purpose: The constructor for the mass transport compartmental model.

Syntax:
   m = MassTransportCompModel;

Arguments:
   m - MassTransportCompModel objects

Description:
   m = MassTransportCompModel - generates a default MassTransportComp model

Limitations:
   None

Method Version:
   P1.0.0

Class fields:
   mtc.ObjectCreator = getenV('UserName'); % (string) object creator
   mtc.ObjectPathName = 'NA-str'; % (string) path for the object
   mtc.ObjectFileName = 'NA-str'; % (string) filename for the object
   mtc.ObjectSetupDate = datenstr(now); % (datum) time stamp
   mtc.Description = 'Mass Transport Compartmental Model Object'; % (object) description of % the object
   mtc.Comments = 'NA-str'; % (string) general comments about the object
   mtc.UserData = 'NA-cell'; % (cell array) user data section for storage
   mtc.FieldExtensions = 'NA-cell'; % (cell array) user defined field extensions
   mtc.History = 'NA-cell'; % (cell array) keep track of what is done to the % object
   mtc.Error = 'NA-str'; % (string) to keep track of error messages
   mtc.ModelType = 'NA-str'; % (string) {'fullcurve'} model type to use
   mtc.Parameters.ka = 'NA-d'; % (double) the association constant
   mtc.Parameters.kaUnits = 'NA-str'; % (string) the association constant units
   mtc.Parameters.kd = 'NA-d'; % (double) the dissociation constant
   mtc.Parameters.kdUnits = 'NA-str'; % (string) the dissociation constant units
   mtc.Parameters.ktr = 'NA-d'; % (double) the mass transport constant
   mtc.Parameters.ktrUnits = 'NA-str'; % (string) the mass transport constant units
   mtc.Parameters.RT = 'NA-d'; % (double) the transport R max
   mtc.Parameters.RTUnits = 'NA-str'; % (string) the transport R max Units
   mtc.FixedC = 'NA-d'; % (double) the fixed concentration (It overlaps with % InputLevel, try to avoid it.)
   mtc.FixedCUUnits = 'NA-d'; % (string) the fixed concentration units
   mtc.VectorT = 'NA-d'; % (double) the time vector
   mtc.UniformlySampled = 'NA-str'; % (string) {'yes'!'no'} was data uniformly sampled
   mtc.SamplingInterval = 'NA-d'; % (double) what was the sampling interval
   mtc.SamplingUnit = 'NA-str'; % (string) {'um'} what are the sampling units
   mtc.DdeInitials = 'NA-d'; % (double) the initial conditions
   mtc.InputIntervals = 'NA-d'; % (double) the input intervals
   mtc.InputLevel = 'NA-d'; % (double) the input level
   mtc.InputLevelUnits = 'NA-d'; % (double) {'nm'} Units of the input level or analyse concentrations

170
mtc.ODESolverHandle = 'NA-d';  %{double} function handle for the ODE
mtc.ComplexFlag = 'NA-str';  %{string} flag to warn if calculation is complex
10.2 display method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@MassTransportCompModel

 display(object)

Purpose:
 To display information about the Mass Transport Compartment Model class object

Syntax:
 display(object)

Arguments:
 object - Mass Transport Compartmental Model class object

Description:
 display(object) - Gets the Mass Transport class object and displays the information contained in it. Also Matlab calls this display method whenever an object is the result of a statement that is not terminated by a semicolon.

Programmer Comments:
 None

Algorithm:
 None

Limitations:
 None

Method Version:
 P1.0.0

Classfields:
 None
10.3  document method

Purpose:
   Print the Mass Transport information to a .m file for later analysis and printing

Syntax:
   document(object);

Arguments:
   object - the Mass Transport object

Description:
   document(object) - generates a .m file with the MassTransport object fields displayed

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.0

Classfields:
   None
10.4 guiloadoobject method

ClassObj = guiloadoobject(ClassObj)

Purpose:
To load the MassTransportCompModel object from the disk with the help of a gui interface.

Syntax:
ClassObj = guiloadoobject(ClassObj)

Arguments:
ClassObj - the MassTransportCompModel class object to save to disk.

Description:
ClassObj = guiloadoobject(ClassObj) - load the MassTransportCompModel object to disk using the gui returned values for path and filename.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
10.5 guisaveobject method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@MassTransportCompModel

ClassObj = guisaveobject(ClassObj)

Purpose:
  save the object

Syntax:
  ClassObj = guisaveobject(ClassObj)

Arguments:
  ClassObj - a MassTransportCompModel object

Description:
  ClassObj = guisaveobject(ClassObj) - save the object via GUI

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
10.6 loadobject method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@MassTransportCompModel

------------------------------------------------------------------------------------------

ClassObj = loadobject(ClassObj, varargin)

Purpose:
To load the MassTransportCompModel object from the disk.

Syntax:
ClassObj = loadobject(ClassObj);
ClassObj = loadobject(ClassObj, filename);

Arguments:
ClassObj - the MassTransportCompModel class object to save to disk.
filename - a string containing the filename of object to load. Please
include the file extension in the name.

Description:
ClassObj = loadobject(ClassObj) - load the MassTransportCompModel object to disk using
the values in model.ObjectFileName and model.ObjectPathName fields or filename variable.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
10.7 print method

print(object)

Purpose:
Displays the ASCII file created by the document method so that
the information can be printed by using the print button.

Syntax:
print(object)

Arguments:
object - the Model object

Description:
print(object) - displays the document information for printing

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
10.8 saveobject method

saveobject(ClassObj)

Purpose:
To save the MassTransportCompModel object to the disk.

Syntax:
saveobject(ClassObj);

Arguments:
ClassObj - the MassTransportCompModel class object to save to disk.

Description:
saveobject(ClassObj) - save the MassTransportCompModel object to disk using the
fields in the model.ObjectFileName and model.ObjectPathName fields.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
10.9 setparameters method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\MassTransportCompModel

-----------------------------------------------------------------------------------------------------------------------

mtc = setparameters(mtc, val, varargin)

Purpose: To set the parameters based on various cases.

Syntax:
mtc = setparameters(mtc, val, varargin)

Arguments:
mtc - the IndependentInteractionsModel class object
val - (cell array) val{k} are parameter vectors for kth interaction.
    Association case: [kobe(1/s), Req(RU)]
    Dissociation case: [koff(1/s), RO(RU)]
    FullCurve: [kon(1/(mMs)) koff(1/s) Rmax(RU)]

TypicalValue - (cell array), val{k}.TypicalValue(k) are the right parameters value
    TypicalValue are mainly for optimization algorithm, i.e. lsqnonlin.

DataSetNumber: additional information may be available in
mtc.model.SimulationInfoDataobject.Set(DataSetNumber).
    For example, the concentration information saved in
dataobject.Set(DataSetNumber).UserData

Description:
mtc = setparameters(mtc, val, varargin) - To set the parameters based on various cases.

Programmer Comments:
Making val as a cell array so that setparameters may have the same
    format for other models too

Algorithm:

Limitations:
1. It works for 1 interaction only now

Method Version:
P1.0.0

Classfields:
None
10.10 simulate method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@MassTransportCompModel

@MassTransportCompModel\simulate.m

Purpose: Generates Simulated Data Based on model object & parameters

Syntax:
[ simdata ] = simulate(ModelObject)
[ simdata, NewModel ] = simulate(ModelObject)
[ simdata ] = simulate(ModelObject, SetNumber)
[ simdata, NewModel ] = simulate(ModelObject, SetNumber)
[ simdata ] = simulate(ModelObject, DataClassObject, SetNumber)

Arguments:
mtc: MassTransportComp object

Description:
[ simdata ] = simulate(ModelObject) - to give results for a given ModelObject
[ simdata, NewModel ] = simulate(ModelObject) - to give results for a given ModelObject
[ simdata ] = simulate(ModelObject, SetNumber) - for Global Analysis
[ simdata, NewModel ] = simulate(ModelObject, SetNumber) - for Global Analysis
[ simdata ] = simulate(ModelObject, DataClassObject, SetNumber) - to optimize

Programmer Comments:
Steps General to all Simulate functions:
1) get parameters from model object
2a) approximate simdata based on parameters, x (or t) data in DataClassObject, and data set number
or:
2b) approximate simdata based on parameters and x (or t) data in Model Object
sim data = ode23s solution to problem:

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
10.11 subsasgn method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@MassTransportCompModel

-----------------------------------------------------------------------------------------------

mt = subsasgn(mt,index,val)

Purpose: Evaluates the value read in and sets the field correctly, it will give
an error if the field is set to an invalid value

Syntax:
    mt = subsasgn(mt,index,val)

Arguments:
    mt - MassTransportCompModel class object
    index - with two fields index.type and index.subs
        index.type : string containing '()', '{}', or '.' specifying subscript type
        index.subs : call array or string containing the actual subscripts
    val - new value

Description:
    mt = subsasgn(mt,index,val) - Obtains the argument values and and assigns 'val'
to the field of MassTransportCompModel object mt

Programmer Comments:
    None

Algorithm:
    None

Limitations:
    None

Method Version:
    P1.0.0

Classfields:
    None

181
10.12 subsref method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\MassTransportCompModel

---------------------------------------------------------------

b = subsref(mt,index)

Purpose: Evaluates the value read in and returns the field value

Syntax:
   b = subsref(mt,index)

Arguments:
   mt - MassTransportCompModel class object
   index - with two fields index.type and index.subs
       index.type : string containing '()','{}', or '.' specifying subscript type
       index.subs : call array or string containing the actual subscripts

Description:
   b = subsref(mt,index) - Obtains the MassTransportCompModel object and the
       index, and evaluates the value read in

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.0

Classfields:
   None
10.13 view method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@MassTransportCompModel

view(varargin)

Purpose: The purpose of this method is to plot the datasets of the SimulationDataObject in the Model class object

Syntax:
view(mt);

Arguments:
mt - Type(MassTransportCompModel), the MassTransportCompModel object

Description:
view(mt) - Plot the simulated data

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
Chapter 11

Model
# Model constructor

```
mod = Model(varargin)
```

**Purpose:**
To create a default object for the class Model.

**Syntax:**
```
mod = Model
```

**Arguments:**
- `mod` - Model class object

**Description:**
```
mod = Model : construct a default Model class object
```

**Programmer Comments:**
None

**Algorithm:**
None

**Limitations:**
Can only generate a default object

**Method Version:**
```
P1.0.0
```

**Classfields:**
- `m.ObjectCreator` = getenv('UserName'); % (string) object creator
- `m.ObjectPathName` = 'NA-str'; % (string) path for the object
- `m.ObjectFileName` = 'NA-str'; % (string) filename for the object
- `m.ObjectSetupDate` = datestr(now); % (datum) time stamp
- `m.Description` = 'Model Object'; % (string) description of the object
- `m.Comments` = 'NA-str'; % (string) general comments about the object
- `m.UserData` = 'NA-cell'; % (cell array) user data section for storage
- `m.FieldExtensions` = 'NA-cell'; % (cell array) user defined field extensions
- `m.History` = 'NA-cell'; % (cell array) keep track of what is done to the object
- `m.Error` = 'NA-str'; % (string) to keep track of error messages
- `m.Selected` = 'NA-str'; % ('yes'|'no') Should this data object be used in processing
- `m.RealOrComplex` = 'NA-str'; % ('real'|'complex') Is the data real or complex
- `m.SimulationInfoDataObject` = 'NA-Object'; % data class object that contains information important to produce a simulation, e.g. x axis data, units, sampling regime, etc.
- `m.SimulationDataObject` = 'NA-Object'; % data class object that contains simulated data
11.2 display method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@Model

display(object)

Purpose:
   To display information about the Model class object

Syntax:
   display(m)

Arguments:
   m - Model class object

Description:
   display(object) - Gets the Model class object and displays the information
   contained in it. Also Matlab calls this display method
   whenever an object is the result of a statement that is
   not terminated by a semicolon.

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.0

Classfields:
   None
11.3 document method

Purpose:
Print the Model information to a .m file for later analysis and printing

Syntax:
document(object);

Arguments:
object - the Model object

Description:
document(object) - generates a .m file with the Model object fields displayed

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
11.4 guiloadobject method

ClassObj = guiloadobject(ClassObj)

Purpose:
To load the Model object from the disk with the help of a gui interface.

Syntax:
ClassObj = guiloadobject(ClassObj)

Arguments:
ClassObj - the Model class object to save to disk.

Description:
ClassObj = guiloadobject(ClassObj) - load the Model object to disk using the
gui returned values for path and filename.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
11.5 guisaveobject method

SPRTool\V0.9\Program\biodatanalysis\V0.9\classes\models\@Model

------------------------------------------------------------------

guisaveobject(ClassObj)

Purpose:
To save the Model object to the disk with the help of a gui interface.

Syntax:
guisaveobject(ClassObj);

Arguments:
ClassObj - the Model class object to save to disk.

Description:
ClassObj = guisaveobject(ClassObj) - save the Model object to disk using
the value supplied by the gui interface.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
11.6  loadobject method

Purpose:
To load the Model object from the disk.

Syntax:
ClassObj = loadobject(ClassObj);
ClassObj = loadobject(ClassObj, filename);

Arguments:
ClassObj - the Model class object to load from disk.
filename - a string containing the filename of object to load. Please
include the file extension in the name.

Description:
ClassObj = loadobject(ClassObj) - Loads the Model object from disk using the
values in ObjectFileName and ObjectPathName fields.
ClassObj = loadobject(ClassObj, filename) - Loads the Model object
specified by filename from disk.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
11.7 print method

print(object)

Purpose:
  Displays the ASCII file created by the document method so that
  the information can be printed by using the print button.

Syntax:
  print(object)

Arguments:
  object - the Model object

Description:
  print(object) - displays the document information for printing

Programmer Comments:
  None

Algorithm:
  None

Limitations:
  None

Method Version:
  P1.0.0

Classfields:
  None
11.8 saveobject method

saveobject(ClassObj)

Purpose:
To save the Model object to the disk.

Syntax:
    saveobject(ClassObj);

Arguments:
    ClassObj - the Model class object to save to disk.

Description:
    saveobject(ClassObj) - save the Model object to disk using the values in
    the ObjectFileName and ObjectPathName fields.

Programmer Comments:
    None

Algorithm:
    None

Limitations:
    None

Method Version:
    P1.0.0

Classfields:
    None
11.9 subsasgn method

m = subsasgn(m,index,val)

Purpose:
The subsasgn function for the Model class. Used to set fields in a Model object.

Syntax:
m = subsasgn(m,index,val)
classobject.fieldname = val;

Arguments:
m - the object having value set
index - the command structure
val - value to set field
classobject - the object name
fieldname - the object’s field name to alter

Description:
a = subsasgn(a,index,val) - Using this approach you must supply the index structure which is described in the MATLAB documentation under classes.

classobject.fieldname = val - This is the preferred method of setting the field values. This way the index structure is generated for you. All you need to do is issue a command like ‘classobject.fieldname = val’. Let’s say we have an object of class Model called ‘model1’. It has a field called ‘Selected’ and we wish to set it to a value of ‘yes’. You would issue the command "model1.Selected = 'yes'". This would set Selected to a value of ‘yes’ and return the updated object.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
11.10  **subsref method**

```
b = subsref(m,index)
```

**Purpose:**
The `subsref` function for the Model class. Used to get field values in a Model object.

**Syntax:**
```
b = subsref(m,index)
val = classobject.fieldname;
```

**Arguments:**
- `m` - the object to get value from
- `b` - the returned field value
- `index` - the command structure
- `val` - value to get from field
- `classobject` - the object name
- `fieldname` - the object’s field name to get value from

**Description:**
```
b = subsref(m,index) - Using this approach you must supply the index structure which is described in the MATLAB documentation under classes.
```
```
val = classobject.fieldname - This is the preferred method of getting the field values. This way the index structure is generated for you. All you need to do is issue a command like 'val = classobject.fieldname'. Lets say we have an object of class Model called 'modell'. It has a field called 'Selected' which we wish to get the value from. You would issue the command 'val = modell.Selected'. This would get Selected and set the variable 'val' to the value of the field.
```

**Programmer Comments:**
None

**Algorithm:**
None

**Limitations:**

**Method Version:**
```
F1.0.0
```

**Classfields:**
None
Chapter 12

OptimizeClass
12.1 OptimizeClass constructor

optim = OptimizeClass(varargin)

Purpose:
The class constructor for the OptimizeClass class. This will construct the OptimizationClass object for use in performing the lsqnonlin.m function to optimize the fitting of the data to the model.

Syntax:
optim = OptimizeClass;
optim2 = OptimizeClass(optim1);
optim = OptimizeClass('translate', optim_struct);

Arguments:
optim, optim1, optim2 - OptimizeClass object
optim_struct - The OptimizeClass object initialization structure (see below)
'translate' - command to use the structure to construct the object.

Description:
optim = OptimizeClass; - Generates the default OptimizeClass class object
optim2 = OptimizeClass(optim1); - Returns an exact copy of the optim1 object in the optim2 object
optim = OptimizeClass('translate', optim_struct);
- Generates an OptimizeClass object using field values specified in the initialization structure optim_struct

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Class fields:

% Structure for OptimizeClass

% Note: You can copy and paste into workspace. Then alter to your liking. Then pass to the constructor to create your object.

op.ObjectCreator = getenv('UserName'); % {string} object creator
op.ObjectPathName = 'NA-str'; % {string} path for the object
op.ObjectFileName = 'NA-str'; % {string} filename for the object
op.ObjectSetupDate = datestr(now); % {datum} time stamp
op.Description = 'OptimizeClass Object'; % {object} description of the object
op.Comments = 'NA-str';  % {string} general comments about
% the object
op.UserData = 'NA-cell';  % {cell array} user data section
% for storage
op.FieldExtensions = 'NA-cell';  % {cell array} user defined field
% extensions
op.History = 'NA-cell';  % {cell array} keep track of what
% is done to the object
op.Error = 'NA-str';  % {string} to keep track of error
% messages
op.Selected = 'yes';  % {'yes'|'no'} Should this data object
% be used in processing
op.RealOrComplex = 'real';  % {'real'|'complex'} Is the data real
% or complex
op.Optimized = 'no';  % {'yes'|'no'} has optimization been
% performed
op.IdentificationTechnique = 'matlab_lsqnonlin';  % {string} optimization technique used,
% example: lsqnonlin.m
op.Results.parameters = [];  % {structure} the results from the optimization
op.Results.ResidualNorm = [];  % {double array} residual of the sum of squares
op.Results.Residual = [];  % residuals
op.Results.ExitFlag = [];  % from matlab_lsqnonlin output - indicates success
% of optimization
op.Results.OutPut = '';  % from matlab_lsqnonlin output; output is structure
op.Results.Lambda = [];  % from matlab_lsqnonlin output
op.Results.Jacobian = '';  % from matlab_lsqnonlin output
op.Options.LargeScale = 'off';  % {'off'|'on'} Use large scale or medium scale
% approximation using finite differences
op.Options.Diagnostics = 'on';  % {'off'|'on'} Display diagnostic information to the
% screen
op.Options.Display = 'iter';  % {'none'|'iter'|'final'} Display the iterations of
% the optimization
op.Options.Jacobian = 'off';  % {'off'|'on'} use user-supplied Jacobian fun or
% approximate using finite differences
op.Options.MaxFunEvals = 10000;  % {double} max number of function evaluations
op.Options.MaxIter = 5000;  % {double} max number of iteration allowed
op.Options.TolFun = 1e-7;  % {double} the termination tolerance for function
% value(s)
op.Options.TolX = 1e-7;  % {double} the termination tolerance for the x
% value(s)
op.Options.JacobMult = [];  % {func handle} function handle for the Jacobian
% Multiply
% function
op.Options.JacobPattern = [];  % Sparsity pattern of the Jacobian for
% finite-differencing
op.Options.MaxPCGIter = '';  % Maximum number of PCG (preconditioned conjugate
% gradient) iterations
op.Options.PrecondBandWidth = [];  % Upper bandwidth of preconitioner for PCG.
op.Options.TolPCG = '';  % Termination tolerance on the PCG iteration.
op.Options.TypicalX = [];  % Typical x values.
op.Options.DerivativeCheck = 'off';  % {'off'|'on'} Compare user-supplied derivatives
% (Jacobian) to finite-differencing derivatives.
op.Options.DiffMaxChange = '';  % Maximum change in variables for finite-differencing.
op.Options.DiffMinChange = '';  % Minimum change in variables for finite-differencing.
op.Options.LevenbergMarquardt = 'on';  % {'off'|'on'} Choose Levenberg-Marquardt over
% Gauss-Newton algorithm.
op.Options.LineSearchType = 'cubicpoly';  % {'cubicpoly'|'quadcubic'} Line search algorithm
%choice.

%{Iteration}'Final'}!!!not sure whether we need it

%{Iteration}'Final'}!!!not sure whether we need it

%{'All' | [..vector of which errors to display..]}

%{'All' | [..vector of which states to display..]}

%{'yes'|'no'}

%{'yes'|'no'}

%The current iteration

%Set these before calling optimization routine

%{'yes'|'no'}

%{@ode23s | @ode15s | @ode45s etc...}

%Fixed parameters indices

200
### 12.2 display method

```plaintext
display(m)
```

**Purpose:**
This is the display function for the OptimizeClass object. It will display the fields in a formatted fashion.

**Syntax:**
```plaintext
display(m);
```

**Arguments:**
m - the OptimizeClass class input

**Description:**
display(m); - Displays the formatted field data to the command window.

**Programmer Comments:**
None

**Algorithm:**
None

**Limitations:**
None

**Method Version:**
P1.0.0

**Classfields:**
None
12.3 document method

SPRTool\V0.9\Program\biodeatanalysis\V0.9\classes\methods\@OptimizeClass

---

document(object, varargin)

Purpose:
Print the OptimizeClass information to a .m file for later analysis and printing

Syntax:
document(object);

Arguments:
object - the OptimizeClass object

Description:
document(object) - generates a .m file with the OptimizeClass object fields displayed

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
12.4 fit method

[model, op, Parameters] = fit(op, data, model, ParametersStructure)

Purpose:
Performs the optimization using the model and data objects. The
data object supplies the time vector and the model object supplies the
curve.

Syntax:
[model, op, Parameters] = fit(op, data, model, ParametersStructure)

Arguments:
op - the OptimizeClass class object
data - the DataClass object
model - the Model object
Parameters - Optimized parameters
ParametersStructure: a structure containing the following field.
ParametersStructure.DataPointsWeights;
ParametersStructure.TypicalValue;
ParametersStructure.NumberOfIteration;
ParametersStructure.PlotIntermediateResults;
ParametersStructure.H_axis;

Description:
[model, op, Parameters] = fit(op, data, model, ParametersStructure)
- Performs the optimization using the model and data objects.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

ClassFields:
None
12.5  guiloadobject method

ClassObj = guiloadobject(ClassObj)

Purpose:
To load the OptimizeClass object from the disk with the help of a gui interface.

Syntax:
ClassObj = guiloadobject(ClassObj)

Arguments:
ClassObj - the OptimizeClass class object to save to disk.

Description:
ClassObj = guiloadobject(ClassObj) - load the OptimizeClass object to disk using the
gui returned values for path and filename.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
12.6  guisaveobject method

Purpose:
To save the OptimizeClass object to the disk with the help of a gui interface.

Syntax:
guisaveobject(ClassObj);

Arguments:
ClassObj - the OptimizeClass class object to save to disk.

Description:
ClassObj = guisaveobject(ClassObj) - save the OptimizeClass object to disk using
the value supplied by the gui interface.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
12.7 loadobject method

ClassObj = loadobject(ClassObj, varargin)

Purpose:
To load the OptimizeClass object from the disk.

Syntax:
ClassObj = loadobject(ClassObj);
ClassObj = loadobject(ClassObj, filename);

Arguments:
ClassObj - the OptimizeClass class object to load from disk.
filename - a string containing the filename of object to load. Please
include the file extension in the name.

Description:
ClassObj = loadobject(ClassObj) - Loads the OptimizeClass object from
disk using the values in ObjectFileName and ObjectPathName fields.
ClassObj = loadobject(ClassObj, filename) - Loads the OptimizeClass object
specified by filename from disk.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
12.8 print method

print(object)

Purpose:
Displays the ASCII file created by the document method so that
the information can be printed by using the print button.

Syntax:
print(object)

Arguments:
oobject - the OptimizeClass object

Description:
print(object) - displays the document information for printing

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
12.9  saveobject method

saveobject(ClassObj)

Purpose:
To save the OptimizeClass object to the disk.

Syntax:
saveobject(ClassObj);

Arguments:
ClassObj - the OptimizeClass class object to save to disk.

Description:
saveobject(ClassObj) - save the OptimizeClass object to disk
using the values in the ObjectFileName and ObjectPathName fields.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
12.10 subsasgn method

a = subsasgn(a, index, val)

Purpose:
The subsasgn function for the OptimizeClass class. Used to set fields in an OptimizeClass object.

Syntax:
a = subsasgn(a, index, val);
classobject.fieldname = val;

Arguments:
a - the object having value set
index - the command structure
val - value to set field
classobject - the object name
fieldname - the objects field name to alter

Description:
a = subsasgn(a, index, val) - Using this approach you must supply the index structure which is described in the MATLAB documentation under classes.

classobject.fieldname = val - This is the preferred method of setting the field values. This way the index structure is generated for you. All you need to do is issue a command like 'classobject.fieldname = val'. Let's say we have an object of class OptimizeClass called 'op'. It has a field called 'DisplayErrorsRate' and we wish to set it to a value of 'Final'. You would issue the command "op.DisplayErrorsRate = 'Final';". This would set DisplayErrorsRate to a value of 'Final' and return the updated object.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.0

Classfields:
None
12.11 subsref method

b=subsref(a,index)

Purpose:
The subsref function for the OptimizeClass class. Used to get field values in an OptimizeClass object.

Syntax:
b=subsref(a,index);
val = classobject.fieldname;

Arguments:
a - the object to get value from
b - the returned field value
index - the command structure
val - value to get from field
classobject - the object name
fieldname - the objects field name to get value from

Description:
b=subsref(a,index) - Using this approach you must supply the index structure which is described in the MATLAB documentation under classes.

val = classobject.fieldname - This is the preferred method of getting the field values. This way the index structure is generated for you. All you need to do is issue a command like 'val = classobject.fieldname. Lets say we have an object of class OptimizeClass called 'op'. It has a field called 'DisplayErrorsRate' which we wish to get the value from. You would issue the command 'val = op.DisplayErrorsRate'. This would get DisplayErrorsRate and set the variable 'val' to the value of the field.

Programmer Comments:
None

Algorithm:
None

Limitations:
Can only generate a default object

Method Version:
F1.0.0

Classfields:
None
Part II

Functions
Chapter 13

General Functions
13.1 **SegmentIntersection function**

SPRTool\V0.9\Program\mfiles

[Xval,Yval] = SegmentIntersection(x,y1,y2,CloseEnough)

**Purpose:**
Finds the intersection of lines or curves that are described
by the values in the inputs.

**Syntax:**
[Xval,Yval] = SegmentIntersection(x,y1,y2,CloseEnough)

**Arguments:**
x - vector x
y1 - vector y1
y2 - vector y2
CloseEnough - tolerance value
Xval - x value to use
Yval - y value to use

**Description:**
[Xval,Yval] = SegmentIntersection(x,y1,y2,CloseEnough) - pass the values
for the segments defined by (x,y1) and (x,y2) and the close enough
tolerance value. Returns the intersection point Xval and Yval.

**Programmer Comments:**
None

**Algorithm:**
None

**Limitations:**
None

**Method Version:**
P1.0.1

**Classfields:**
None
13.2  bda_findext function

SPRTool\V0.9\Program\mfiles

[filenamewithoutextension,extension] = bda_findext(filename)

Purpose:
Takes a filename and extracts the extension from the name.

Syntax:
[filenamewithoutextension,extension] = bda_findext(filename)

Arguments:
filename - the file name to analyze.
filenamewithoutextension - file name with no extension
extension - the extension

Description:
[filenamewithoutextension,extension] = bda_findext(filename) - recovers
the filename extension of a file.

Programmer Comments:
None

Algorithm:
None

Limitations:
None

Method Version:
P1.0.1

Classfields:
None
13.3 exp_viewer function

SPRTool\V0.9\Program\mfiles

varargout = exp_viewer(varargin)

Purpose:
The experiment gui plot data viewer callback function.

Syntax:
   expobj = exp_viewer(expobj);

Arguments:
   expobj - experiment object

Description:
   expobj = exp_viewer(expobj) - loads the experiment data and
displays the gui to plot the data.

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.1

Classfields:
   None
13.4 exponential_fit function

SPRTool\V0.9\Program\mfiles

[Error] = exponential_fit(X, Optim)

Purpose:
This is the function to be minimized in the estimation of Kd.

Syntax:
This function is passed in as the first parameter to MATLAB's lsqnonlin
function as follows:
lsqnonlin('exponential_fit', X, LB, UB, OPTIONS, Optim);

Arguments:
X - Initial conditions for the minimization; different parameters are
expected for each supported curve fitting technique:
For 'SingleExp', X = [Beta Alpha Gamma Kd]
For 'DoubleExp', X = [Beta1 Alpha1 Beta2 Alpha2 Kd]
For 'Constant', X = [Delta, Kd]
Optim - Structure containing data like concentrations and equilibrium
estimates (Req) needed by the minimization.

Description:
lsqnonlin('exponential_fit', X, LB, UB, OPTIONS, Optim);

Programmer Comments:
None

Algorithm:
None

Limitations:
Only supports the single exponential, the double exponential, the
single exponential without constant, and the constant curve fitting
techniques.

Method Version:
P1.0.1

ClassFields:
None
13.5 selector function

SPRTool\V0.9\Program\mfiles

a=selector(a,varargin)

Purpose:
To select the proper sensorgrams for the gui

Syntax:
   a=selector(a,varargin)

Arguments:
   a - adjust object

Description:
   a=selector(a,varargin) - does the selection for the gui.

Programmer Comments:
   None

Algorithm:
   None

Limitations:
   None

Method Version:
   P1.0.1

Classfields:
   None
13.6 bda_findext function

SPRTool\V0.9\Program\biodatanalysis\V0.9\mfiles

---------------------------------------------

BDA_FINDEXT Summary of this function goes here
Detailed explanation goes here
file1_object1.mat

find all occurrences of '.' in filename
13.7 masstransportmodel function

SPRTool\V0.9\Program\biodatanalysis\V0.9\mfiles

function dx=masstransportmodel(t,x,flag,kinparameters,input)

Purpose: To calculate the dx for the feval function

Syntax:
   dx = masstransportmodel(t,x,flag,kinparameters,input);

Arguments:
   t -
   x -
   flags -
   kinparameters - kinetic parameters
   input -

Description:

Created By: Raimund
Date: 3-28-2000
13.8 optimizationerror function

SPRTool\V0.9\Program\biodataanalysis\V0.9\mfiles\optimizemfiles

Purpose: The function used by lsqnonlin to compute the error between the model simulation output and the actual data.

Syntax:
   Error = optimizationerror(x0, model, data, ParametersStructure)

Arguments:
   xo - Starting point(s)
   model - the model object used to simulate the data
   data - the dataclass object to use
   DataPointsWeights - cell array contain weight vector for each data set
   Error - the error between the simulated data and the actual
   ParametersStructure: a structure containing the following field.
       ParametersStructure.DataPointsWeights;
       ParametersStructure.TypicalValue;
       ParametersStructure.NumberOfIteration;
       ParametersStructure.PlotIntermediateResults;
       ParametersStructure.H_axis;

Description:
   Error = optimizationerror(x0, model, data, ParametersStructure) -
   Put parameters (x0 and fixed parameters into model object)
   this may need to be restructured to take some code out of the error function