(4) iFit/Plotting:

« One Class to plot them all »

God damn it ! Just plot it !

iFit/Plotting: plot

Plotting iData objects can be done with a unique method:

```
>> plot(a)
```

iFit/Plotting: plot

Plotting iData objects can be done with a unique method:

>> plot(a)

End of my talk

iFit/Plotting: plot

Plotting iData objects can be done with a unique method:

```
>> plot(a)
```

Ok, let's go on...

```
>> a = iData('filename');
>> plot(a)
```

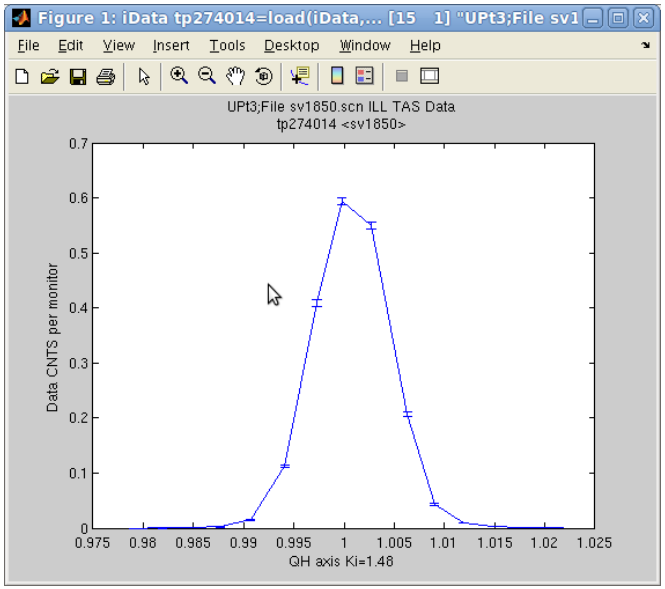
A plot WILL be rendered for 1D-3D data sets, with the guessed *Signal* and *Axes*.
You may need to tune the *Signal* definition, as well as *Aliases* and *Axes* with *setalias* and *setaxis*.

Manipulating the plots does not modify the original objects (may be possible in the future).

iFit/Plotting: 1D plots

Simple plotting of a vectorial data set

```
>> plot(a)
```



Specify the colour and line style

```
>> plot(a, 'r-')
```

b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
w	white	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

Plots can be **edited** with the  button. You can also choose the **zoom** in/out

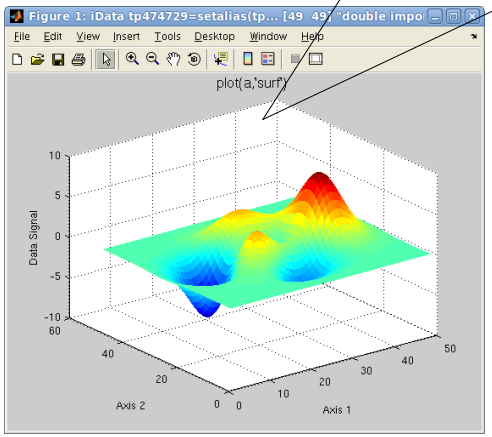
Lines have a **contextual menu** (*right mouse button*), as well as frame axes, to allow to duplicate windows, switch to log scale, toggle grid, …

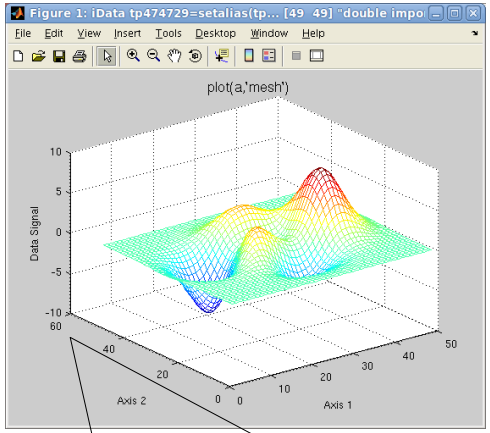
The plot method returns the graphics object handle: *h=plot(...)*

iFit/Plotting: 2D plots

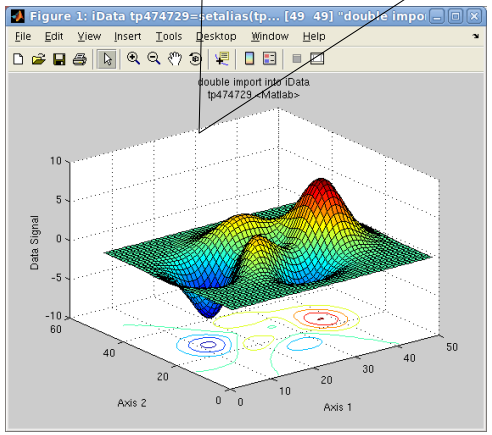
Simple plotting of a matrix data set

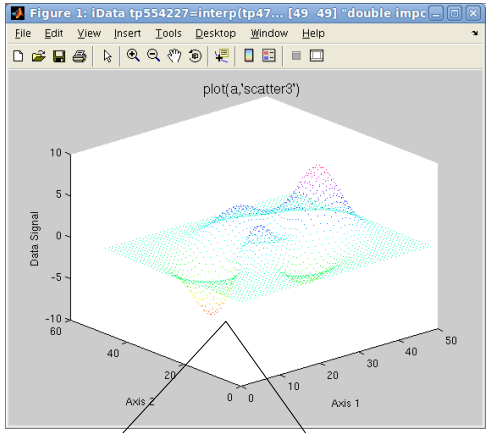
```
>> plot(a);
>> surf(a) % surface
```






```
>> plot(a,'surfc');
>> surfc(a) % surface+contour
```






```
>> plot(a,'mesh');
>> mesh(a) % wire-mesh
```

```
>> plot(a,'scatter3');
>> scatter3(a) % coloured points
```

```
>> image(r,g,b);   % up to 3 colour channels superposed
```

And more methods: *contour, image, pcolor, lighted surface, waterfall, plot3, surface mapping (13 methods)*

Plot options:

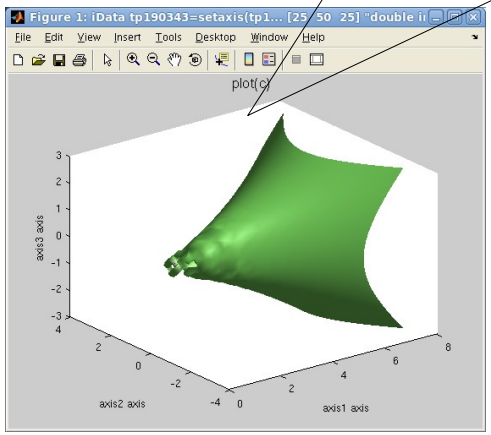
Transparency, light, tight axis, top/side view, hide axes, ...

```
>> plot(a,'surfl transparent');
```

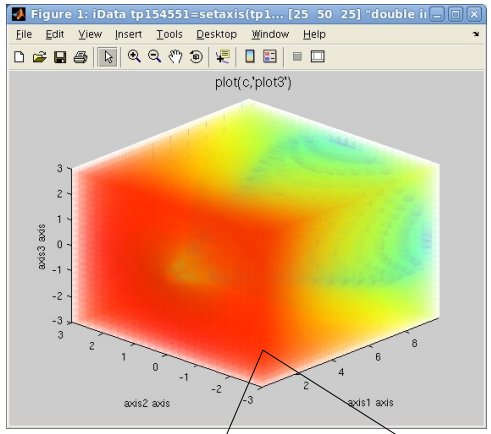
iFit/Plotting: 3D plots

Simple plotting of a volume data set

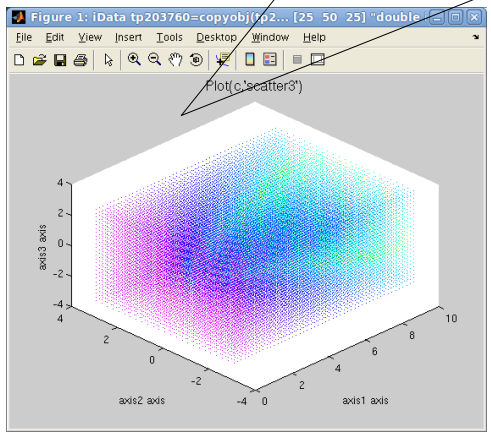
```
>> plot(a);
>> surf(a) % surface
```



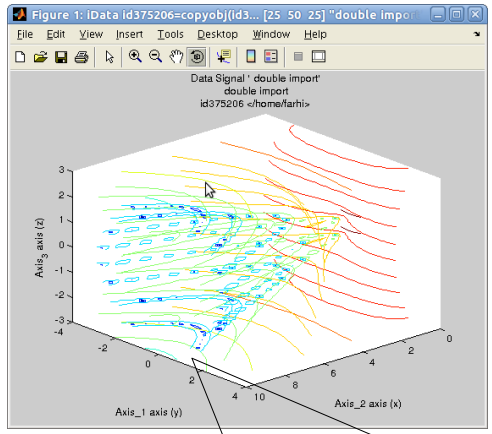

```
>> plot(a,'plot3');
>> plot3(a) % transparent volume
```




```
>> plot(a,'scatter3');
>> scatter3(a) % coloured points
```

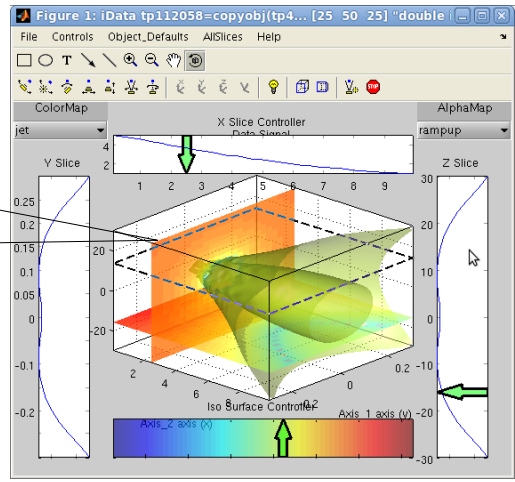



```
>> plot(a,'waterfall');
>> watterfall(a) % lines
```



As for 1D and 2D, plots can be edited, zoomed, have contextual menu.

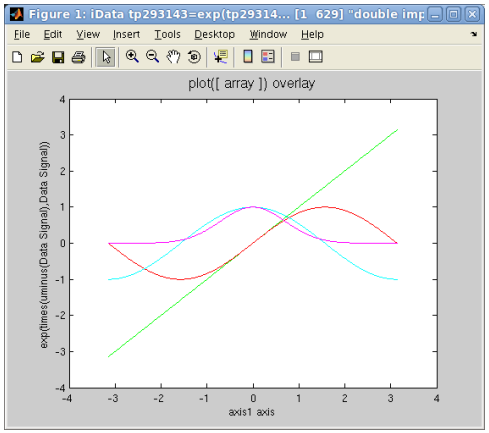
```
>> slice(a);
% volume explorer
```



iFit/Plotting: object arrays: overlay

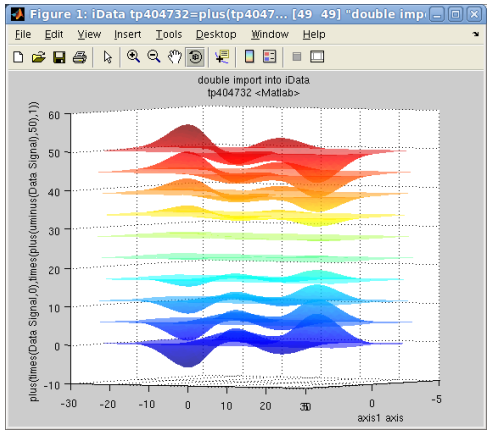
When manipulating arrays of objects, it is possible to plot them in one shot, in *the same* axis frame (**overlay**)

```
>> plot([ array ])
```



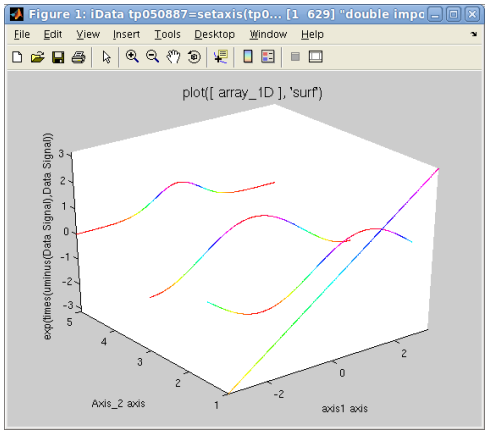
This is the same as an iterative

```
>> hold on
```



In order to assemble 1D objects side-by-side, we mimic a surface with:

```
>> plot([ array ], 'surf')
>> surf([array])
```

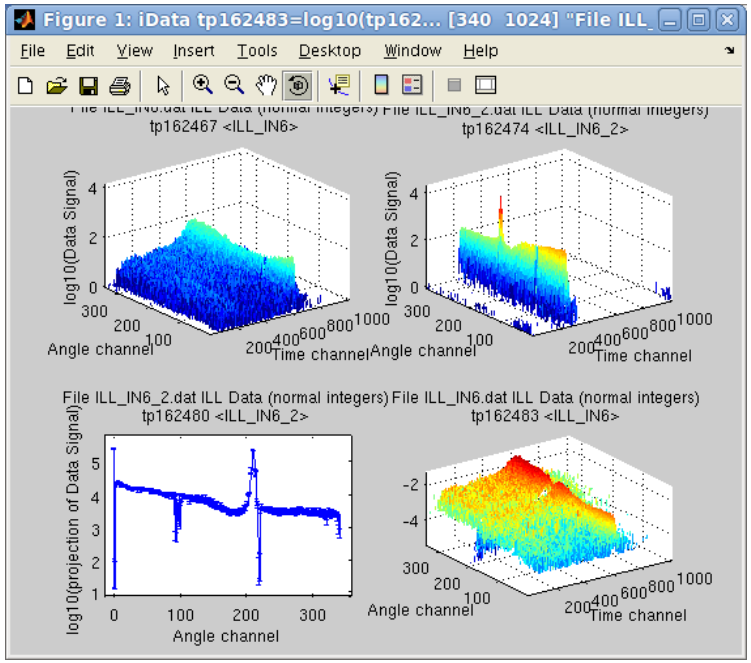


If a 2nd rank axis exists, it is used, or defaulted to the index in the array.

iFit/Plotting: object arrays: multiple plots

When manipulating arrays of objects, it is possible to plot them in one shot, in *separated* axis frames (**multiple-plots** *aka* **subplot** *aka* **overview**)

```
>> subplot([ array ])
>> subplot([ array ], 'surf')
>> subplot([ array ], [m n], 'options')      m*n tiles
```



iFit/Plotting: labels, axis limits

As seen in the iData object description, **labels** can be attached to Aliases.
Axes are linked to Aliases, and thus use their labels.

```
>> a.Title = 'blah'      % sets the object Title (does not depend on the Signal)
>> title(a, 'bloh')      % sets the Signal label
>> label(a, 0, 'bluh')
>> label(a, 'Signal', 'blih'')
>> label(a, 1, 'blang') % sets the X axis (rank 2, columns) label
>> xlabel(a, 'blong'); ylabel(a, 'bling'); ...
```

These affect the object, and should be executed ***before*** the object is displayed.

The usual

```
>> title('bang'); xlabel('bong')
```

affect the current plot, and not the original object.

Similarly, the

```
>> xlim(a, [min max]); ylim(a, [min max]);
```

can be used on the original object to **crop** it, prior to plotting.
This is different from xlim([min max]) which restrict the plot rendering.

iFit/Plotting: saving

It is possible to save the plot directly from the *File/Save as* menu. But the initial object is not retained.

To retain most of the initial object, use the *saveas* method:

```
>> plot(a);
>> saveas(a, '', 'mat');
>> saveas(a, '', 'hdf');
```

iFit/Plotting: Exercise

Retrieve the data you worked with in *(3) Exercise: load/save*.
Plot it and experiment the different plot rendering types.

To experiment 1D plots, you may use:

```
>> a=load(iData, [ ifitpath 'Data/sv1850.scn' ]); plot(a)
```

To experiment 2D plots, you may use:

```
>> a=load(iData, [ ifitpath 'Data/Ag_3_a.edf' ]); plot(a)
```

Try surfc, surfl, scatter3, mesh, contour, contourf, waterfall

To experiment 3D plots, you may use:

```
>> load wind
>> a=iData(x,y,z,u)
```

Try plot3, scatter3, waterfall, surf

iFit/Plotting: Exercise

Edit the plot labels, title, axes, …

Try the **contextual** menus both on the axes and the objects.

Change the **labels** and axes **limits** in the plot, and in the object.

Save the figure/window (.fig, image, PDF, …)
Compare the file size with that of the original object (.mat, .hdf).