Command line usage and plotting with Madagascar

Ben Witten

Outline

- RSF data format
- Command line usage
- Plotting

RSF data format

- RSF := regularly sampled format
  - Data exists on a N-dimensional grid

RSF data format

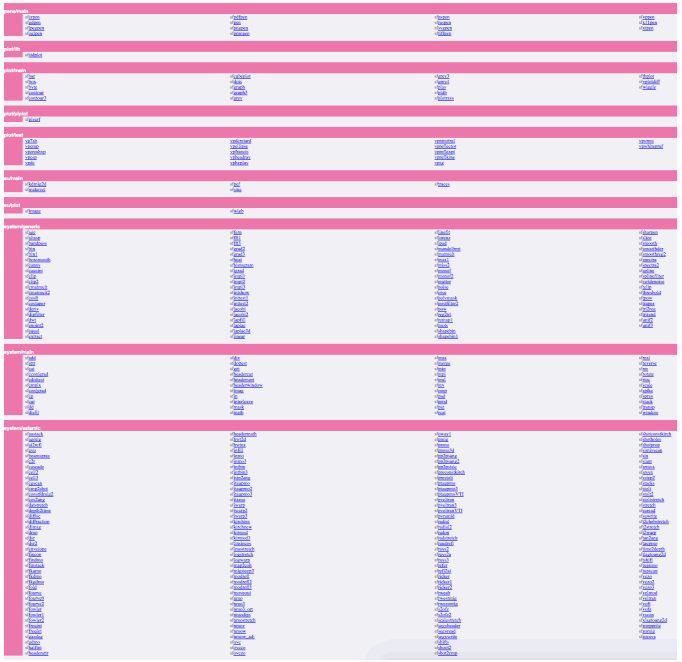
- RSF := regularly sampled format
  - Data exists on a N-dimensional grid
    - *Storing large-scale datasets in a text format may not be economical. RSF chooses the next best thing: it allows data values to be stored in a binary format but puts all data attributes in text files that can be read by humans and processed with universal text-processing utilities.*

RSF data format

*.rsf	*.rsf@
Header file	Binary file in column major order
Information about the origin, sampling, data type, data format, labels, where the binary lives, size of binary	The actual data
Human readable, contains history of what has been done	Not human readable

Madagascar programs

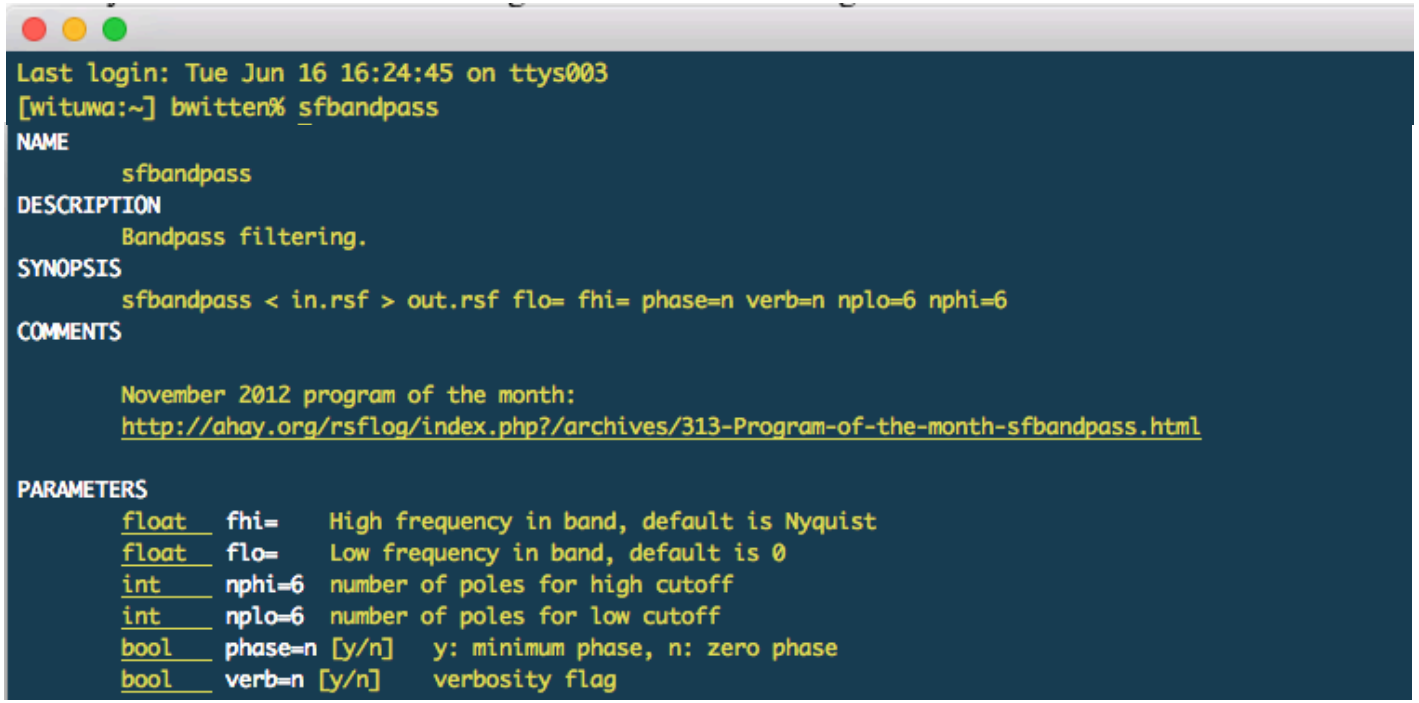
- Most programs begin with the prefix “sf”
- Can “pipe” from one program to another to avoid intermediate files
- Examples: sftransp, sffft, sfbandpass, sfsegyread



http://www.reproducibility.org/RSF/

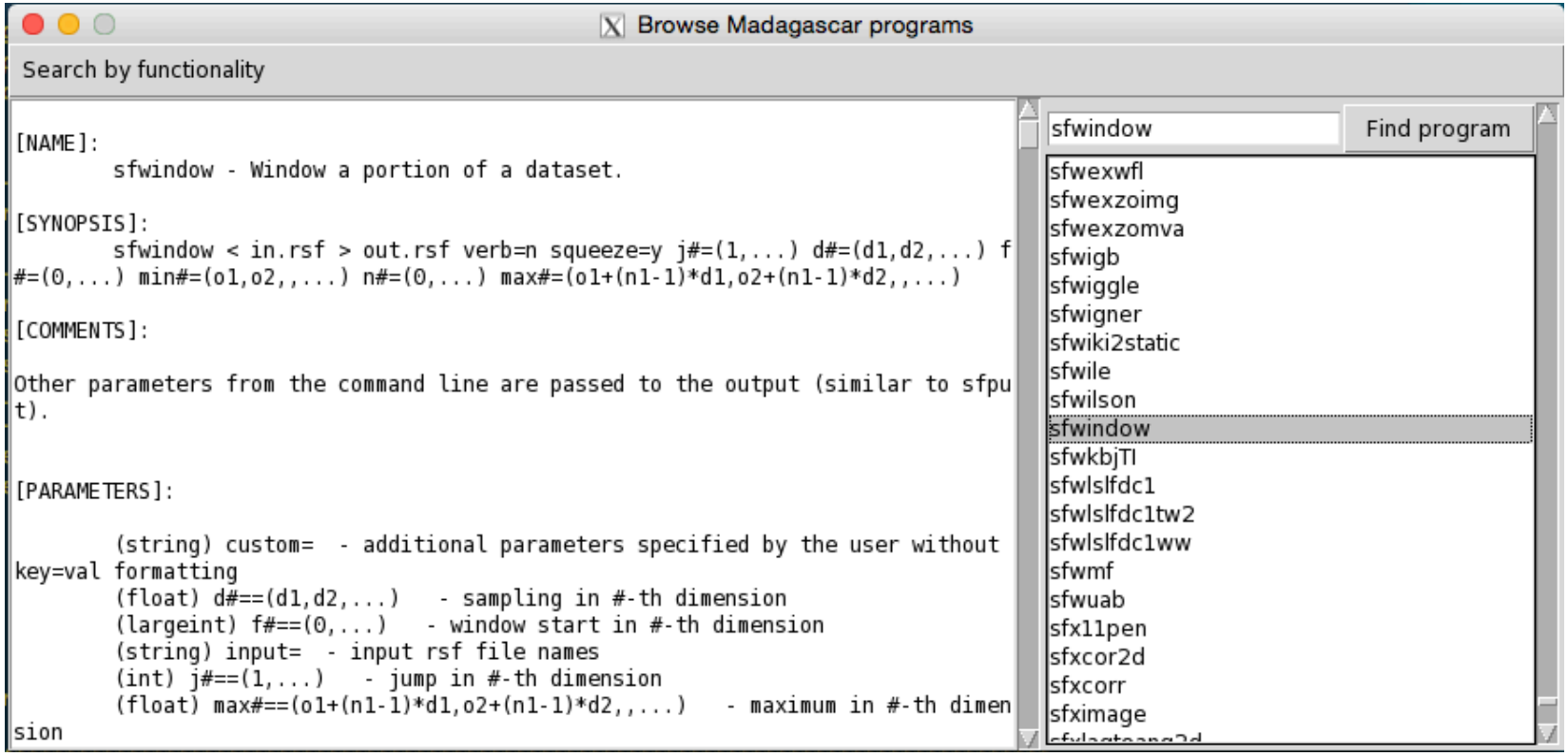
Program documentation

- Each program has self-documentation
- Type the program name without parameters to bring it up



Program documentation

- sfbrowser



Basic Madagascar usage

< input.rsf sfprogram par1= par2= > output.rsf

“<“ = input sign: next string is taken as input

“>” = output sign : next string is taken as output

Note: does not have to be written in this order

Basic Madagascar usage

< input.rsf sfprogram par1= par2= > output.rsf

Basic Madagascar usage

< input.rsf sfprogram par1= par2= > output.rsf

sfprogram < input.rsf par1= par2= > output.rsf

Basic Madagascar usage

< input.rsf sfprogram par1= par2= > output.rsf

sfprogram < input.rsf par1= par2= > output.rsf

sfprogram par1= par2= < input.rsf > output.rsf

Basic Madagascar usage

< input.rsf sfprogram par1= par2= > output.rsf

sfprogram < input.rsf par1= par2= > output.rsf

sfprogram par1= par2= < input.rsf > output.rsf

> output.rsf sfprogram par1= par2= < input.rsf

Let’s do some

sfspike n1=100 k1=20 > file.rsf

Let's do some

sfspike n1=100 k1=20 > file.rsf

sfin < file.rsf (or sfin file.rsf)

sfin is probably the most commonly used program. It shows information about the file

```
[wituwa:~] bwitten% sfin < file.rsf
in:
    in="/var/tmp/file.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.004        o1=0            label1="Time" unit1="s"
        100 elements 400 bytes
```

Let's do some

sfspike n1=100 k1=20 > file.rsf

sfin < file.rsf (or sfin file.rsf)

sfin is probably the most commonly used program. It shows information about the file

```
[wituwa:~] bwitten% sfin < file.rsf
in:
    in="/var/tmp/file.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.004        o1=0          label1="Time" unit1="s"
        100 elements 400 bytes
```

Location of binary

Let's do some

sfspike n1=100 k1=20 > file.rsf

sfin < file.rsf (or sfin file.rsf)

sfin is probably the most commonly used program shows information about the file

```
[wituwa:~] bwitten% sfin < file.rsf
in:
    in="/var/tmp/file.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.004        o1=0          label1="Time" unit1="s"
        100 elements 400 bytes
```

Bytes per element

Let's do some

sfspike n1=100 k1=20 > file.rsf

sfin < file.rsf (or sfin file.rsf)

sfin is probably the most commonly used program shows information about the file

```
[wituwa:~] bwitten% sfin < file.rsf
in:
    in="/var/tmp/file.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.004        o1=0          label1="Time" unit1="s"
        100 elements 400 bytes
```

Data type

Let's do some

sfspike n1=100 k1=20 > file.rsf

sfin < file.rsf (or sfin file.rsf)

sfin is probably the most commonly used program shows information about the file

```
[wituwa:~] bwitten% sfin < file.rsf
in:
    in="/var/tmp/file.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.004        o1=0          label1="Time" unit1="s"
        100 elements 400 bytes
```

Data format

Let's do some

sfspike n1=100 k1=20 > file.rsf

sfin < file.rsf (or sfin file.rsf)

sfin is probably the most commonly used program shows information about the file

```
[wituwa:~] bwitten% sfin < file.rsf
in:
    in="/var/tmp/file.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.004        o1=0          label1="Time" unit1="s"
        100 elements 400 bytes
```

Number of elements in 1st dimension

Let's do some

sfspike n1=100 k1=20 > file.rsf

sfin < file.rsf (or sfin file.rsf)

sfin is probably the most commonly used program shows information about the file

```
[wituwa:~] bwitten% sfin < file.rsf
in:
    in="/var/tmp/file.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.004        o1=0          label1="Time" unit1="s"
        100 elements 400 bytes
```

Sampling interval in 1st dimension

Let's do some

sfspike n1=100 k1=20 > file.rsf

sfin < file.rsf (or sfin file.rsf)

sfin is probably the most commonly used program shows information about the file

```
[wituwa:~] bwitten% sfin < file.rsf
in:
    in="/var/tmp/file.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.004        o1=0          label1="Time" unit1="s"
        100 elements 400 bytes
```

Origin of 1st dimension

Let's do some

sfspike n1=100 k1=20 > file.rsf

sfin < file.rsf (or sfin file.rsf)

sfin is probably the most commonly used program shows information about the file

```
[wituwa:~] bwitten% sfin < file.rsf
in:
    in="/var/tmp/file.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.004        o1=0
        100 elements 400 bytes
```

Label and unit of 1st dimension

```
label1="Time" unit1="s"
```

Let's do some

sfspike n1=100 k1=20 > file.rsf

sfin < file.rsf (or sfin file.rsf)

sfin is probably the most commonly used program shows information about the file

```
[wituwa:~] bwitten% sfin < file.rsf
in:
    in="/var/tmp/file.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.004        o1=0          label1="Time" unit1="s"
        100 elements 400 bytes
```

Total number of elements and size of binary

sfin on higher dimensionality data

```
[bwitten@geocomp202 PEIC-inv-gau]$ sfin junk.rsf
junk.rsf:
    in="/geofs/SCRATCH/bwitten/junk.rsf@"
    esize=8 type=complex form=native
    n1=200          d1=0.2          o1=0.2          label1="Frequency" unit1="Hz"
    n2=608          d2=0.01         o2=0            label2="Distance" unit2="km"
    n3=302          d3=0.01         o3=0            label3="Time" unit3="s"
    n4=1            d4=0.0005       o4=0            label4="Time" unit4="s"
    n5=1            d5=?            o5=?
        36723200 elements 293785600 bytes
[bwitten@geocomp202 PEIC-inv-gau]$
```

Let's do some

sfspike n1=100 k1=20 > file.rsf

sfin < file.rsf (or sfin file.rsf)

sfattr < file.rsf

sfattr may be the 2nd most common program

```
[wituwa:~] bwitten% sfattr < file.rsf
*****************************************
     rms =             0.1
    mean =            0.01
  2-norm =               1
variance =            0.01
 std dev =             0.1
     max =               1 at 20
     min =               0 at 1
nonzero samples = 1
  total samples = 100
*****************************************
```

sfattr provides statistics about the data set

Why is sfin important?

- Quickly provide information about your data
  - Size
  - Dimensionality
  - Sampling
  - If its all there

Why is sfin important?

- Quickly provide information about your data
  - Size
  - Dimensionality
  - Sampling
  - If its all there --- Come back to this

Piping programs

Method 1:

```
sfspike n1=100 k1=20 > file.rsf
sfbandpass < file.rsf flo=1 fhi=40 > file2.rsf
```

Piping programs

Method 1:

```
sfspike n1=100 k1=20 > file.rsf
sfbandpass < file.rsf flo=1 fhi=40 > file2.rsf
```

Let's say we don't care about file.rsf, just file2.rsf

No need to create file.rsf

Piping programs

Method 2:

```
sfspike n1=100 k1=20 |sfbandpass flo=1 fhi=40 > file3.rsf
```

No intermediate files now.

Header file

- Earlier I mentioned that the *.rsf stores the history of what has been done to the data

- Use your favorite text editor to open file2.rsf or file3.rsf

Header/History file

- File2.rsf:

```
[wituwa:~] bwitten% cat file2.rsf
1.6     sfspike Users/bwitten:  bwitten@wituwa.local    Thu Jun 18 14:21:13 2015

        o1=0
        label1="Time"
        data_format="native_float"
        esize=4
        in="/var/tmp/file.rsf@"
        unit1="s"
        d1=0.004
        n1=100
1.6     sfbandpass      Users/bwitten:  bwitten@wituwa.local    Thu Jun 18 14:27:27 2015

        data_format="native_float"
        esize=4
        in="/var/tmp/file2.rsf@"
```

Back to sfin

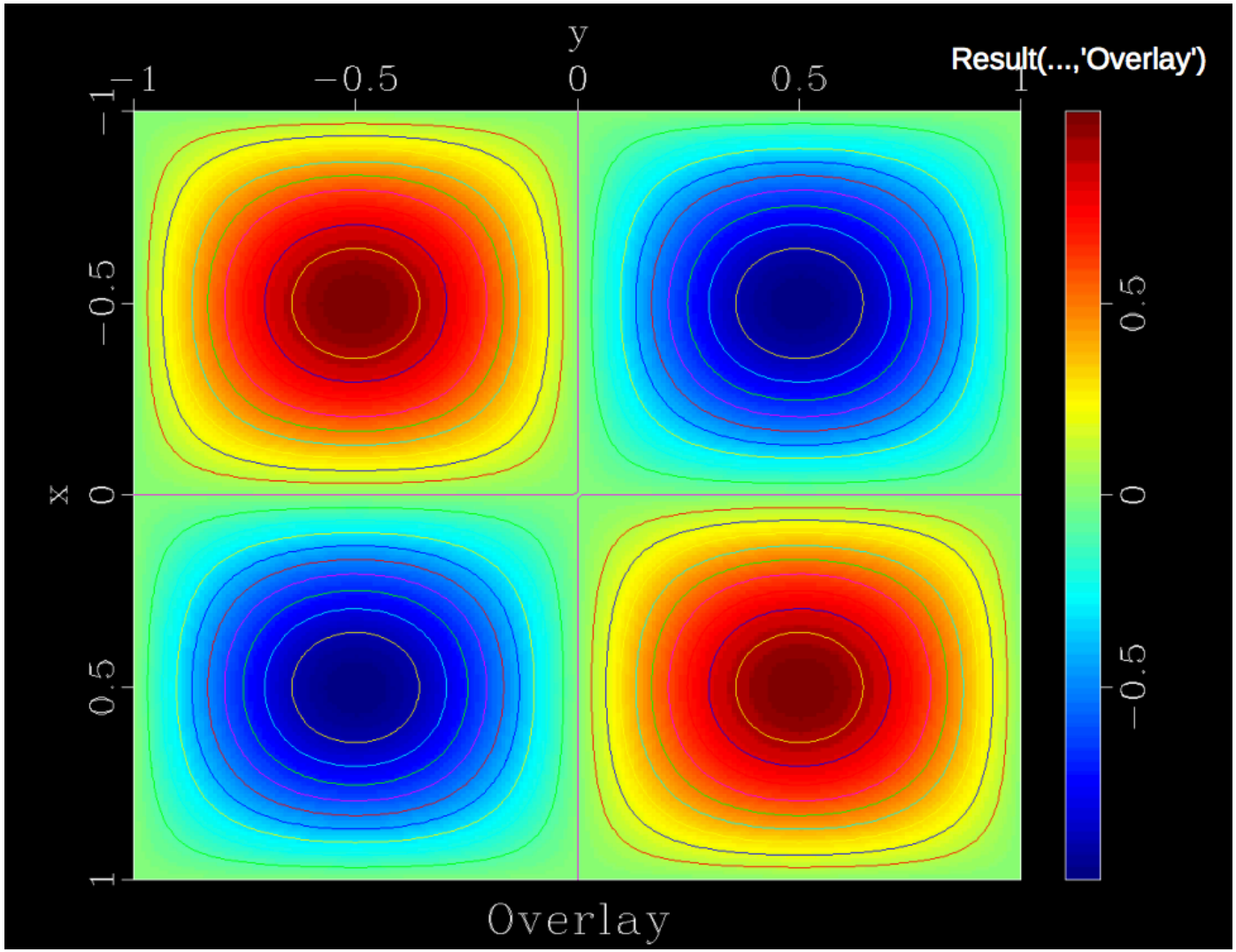
- What sfin actually does is read the *.rsf file from bottom to top looking for n[1,2,3..], d[1,2,3…], o[1,2,3…], esize, etc
- Calculate size and print to screen
- Great feature is that it can tell you if you don’t have the “correct” data size

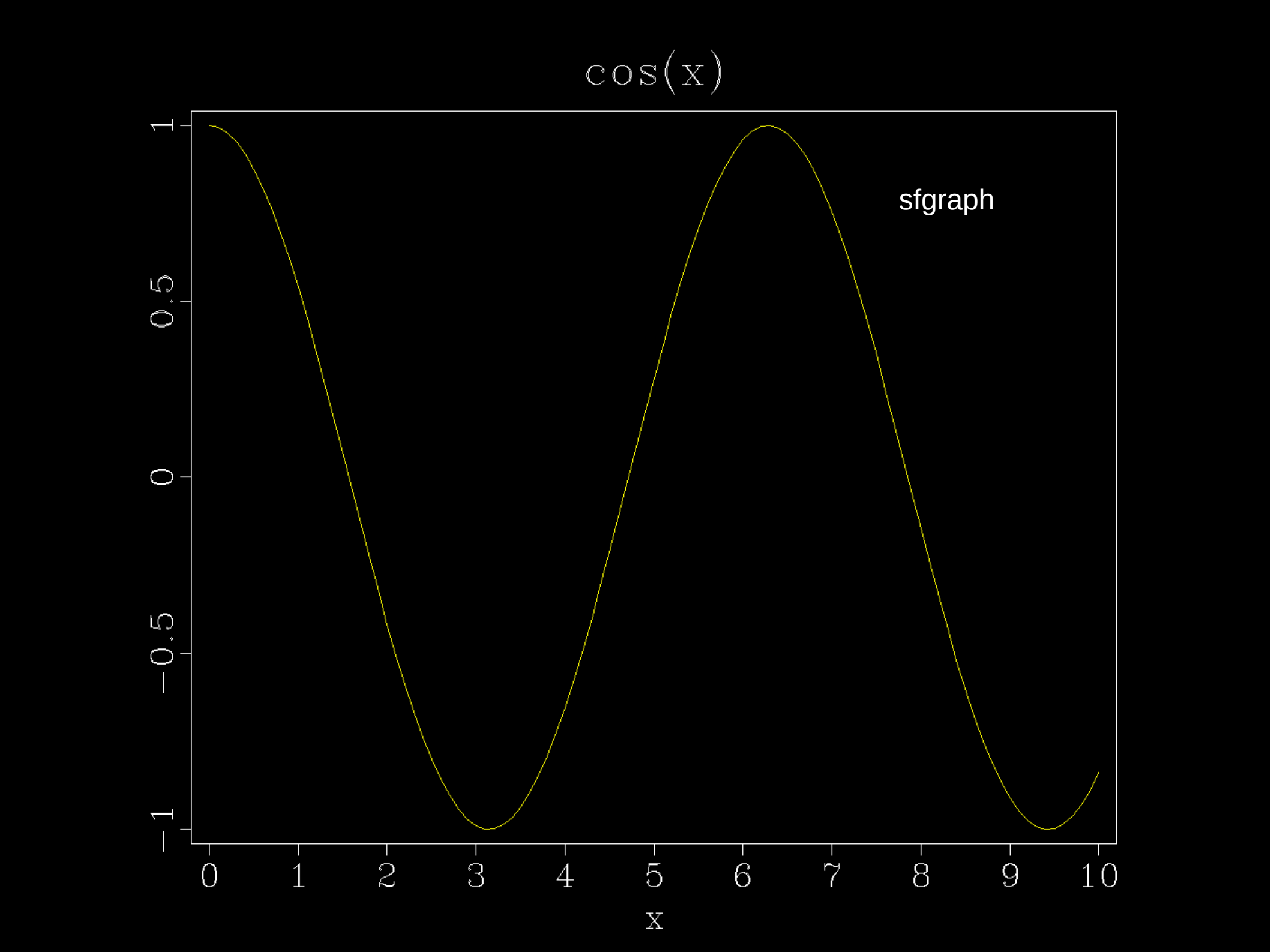
Back to sfin

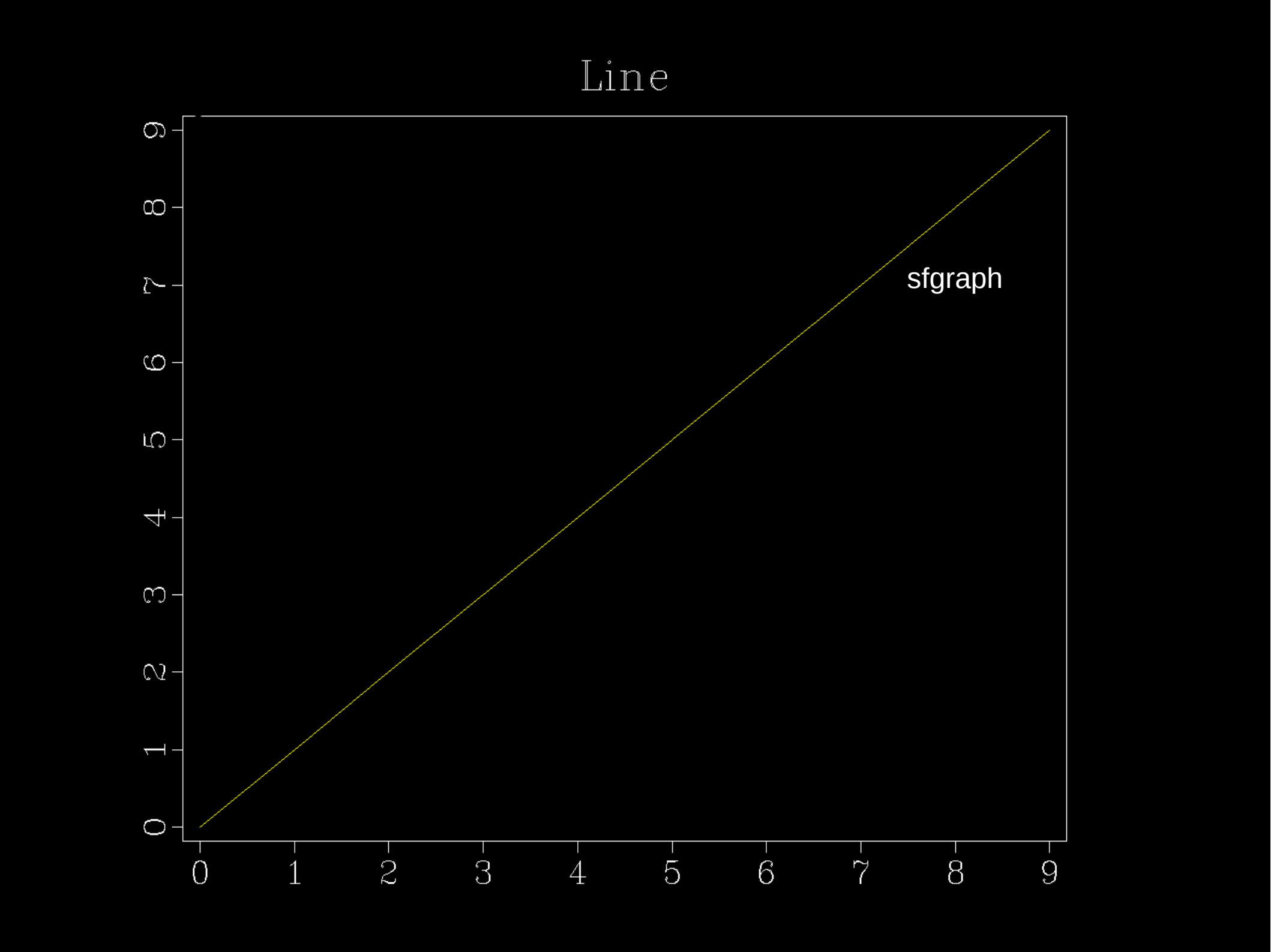
- What sfin actually does is read the *.rsf file from bottom to top looking for n[1,2,3..], d[1,2,3…], o[1,2,3…], esize, etc
- Calculate size and print to screen
- Great feature is that it can tell you if you don’t have the “correct” data size

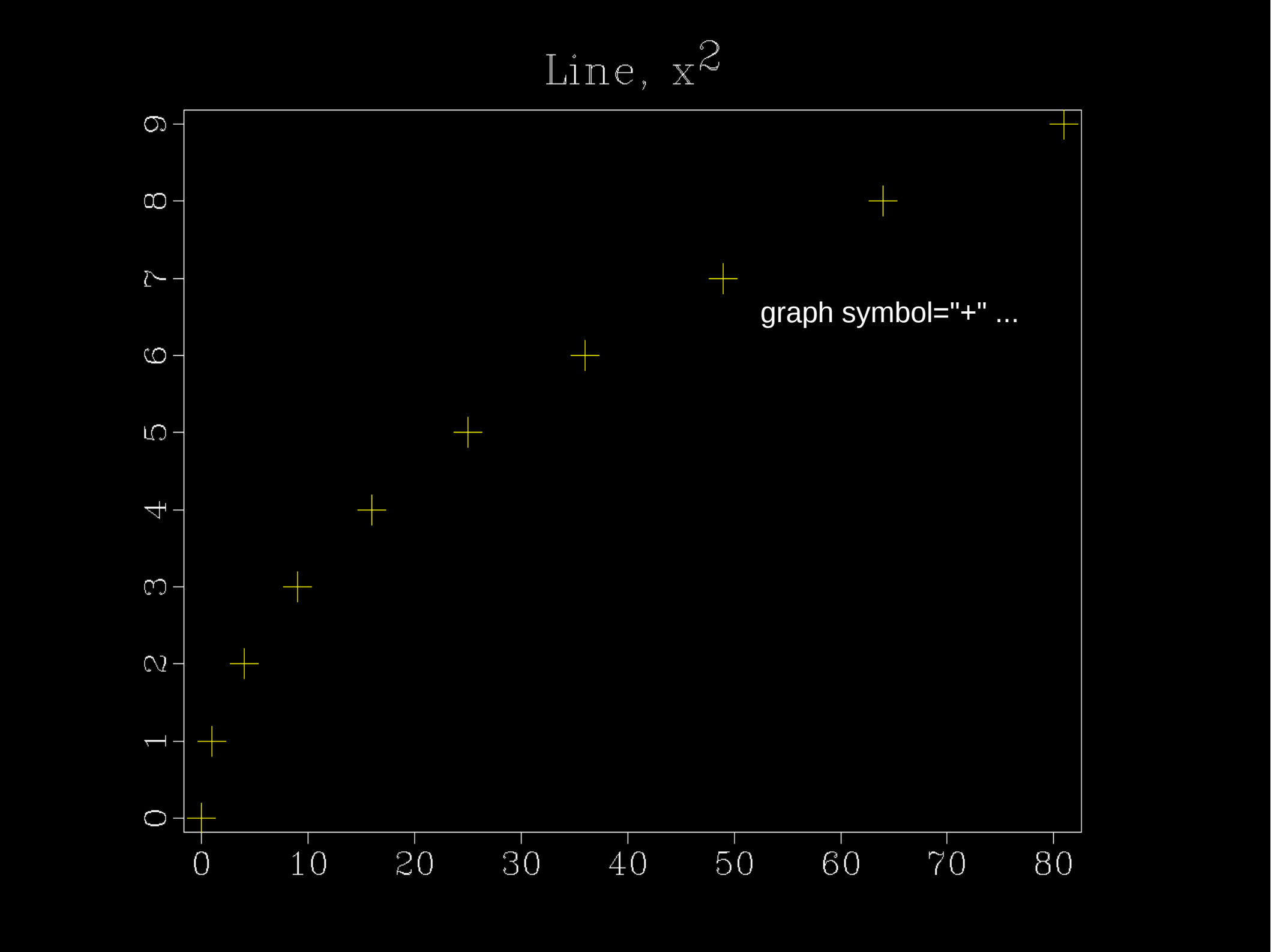
- TEST: change n1=100 to n1=90 in file2.rsf
- Run sfin file2.rsf
- Go change it back

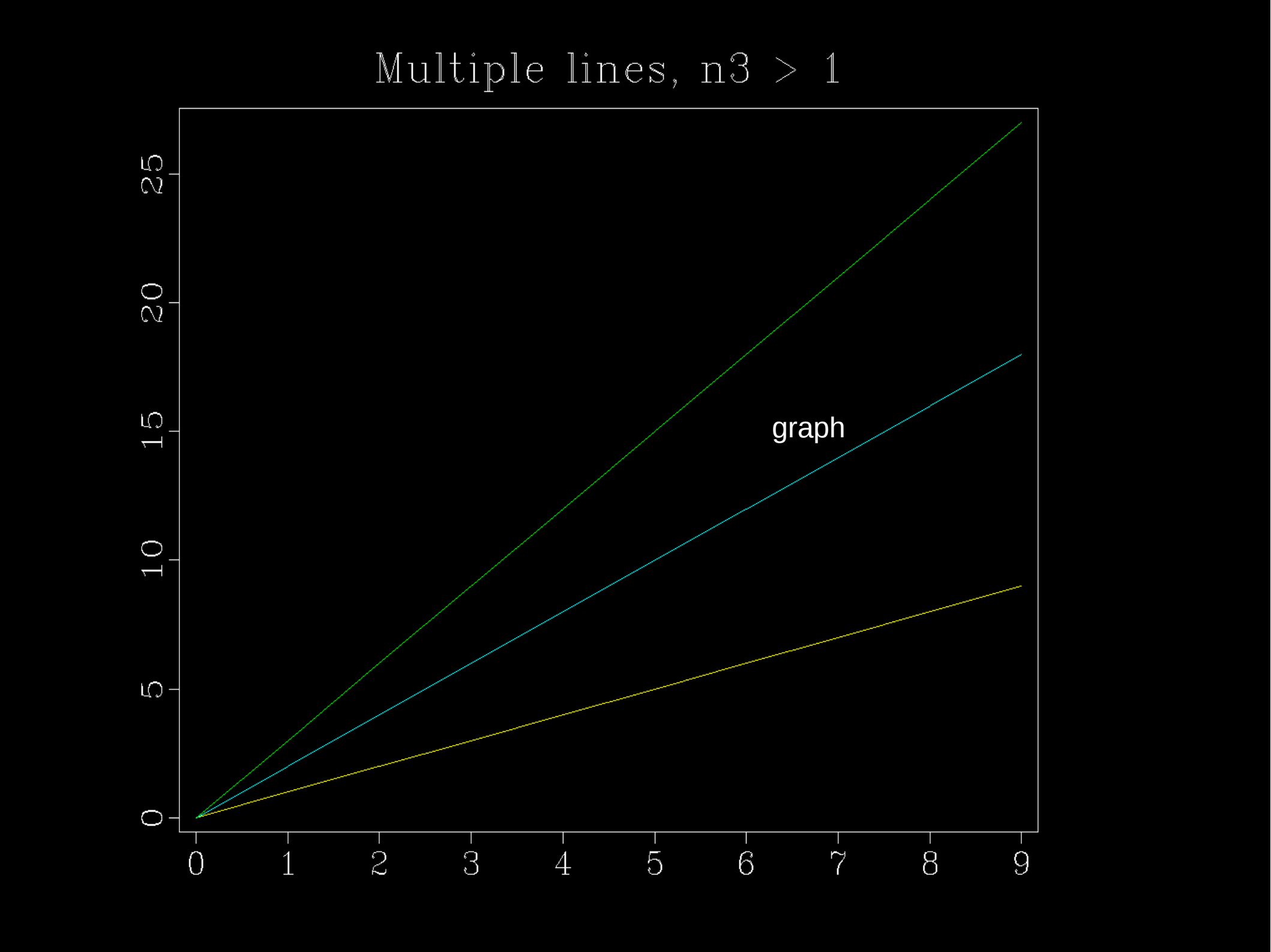
Plotting

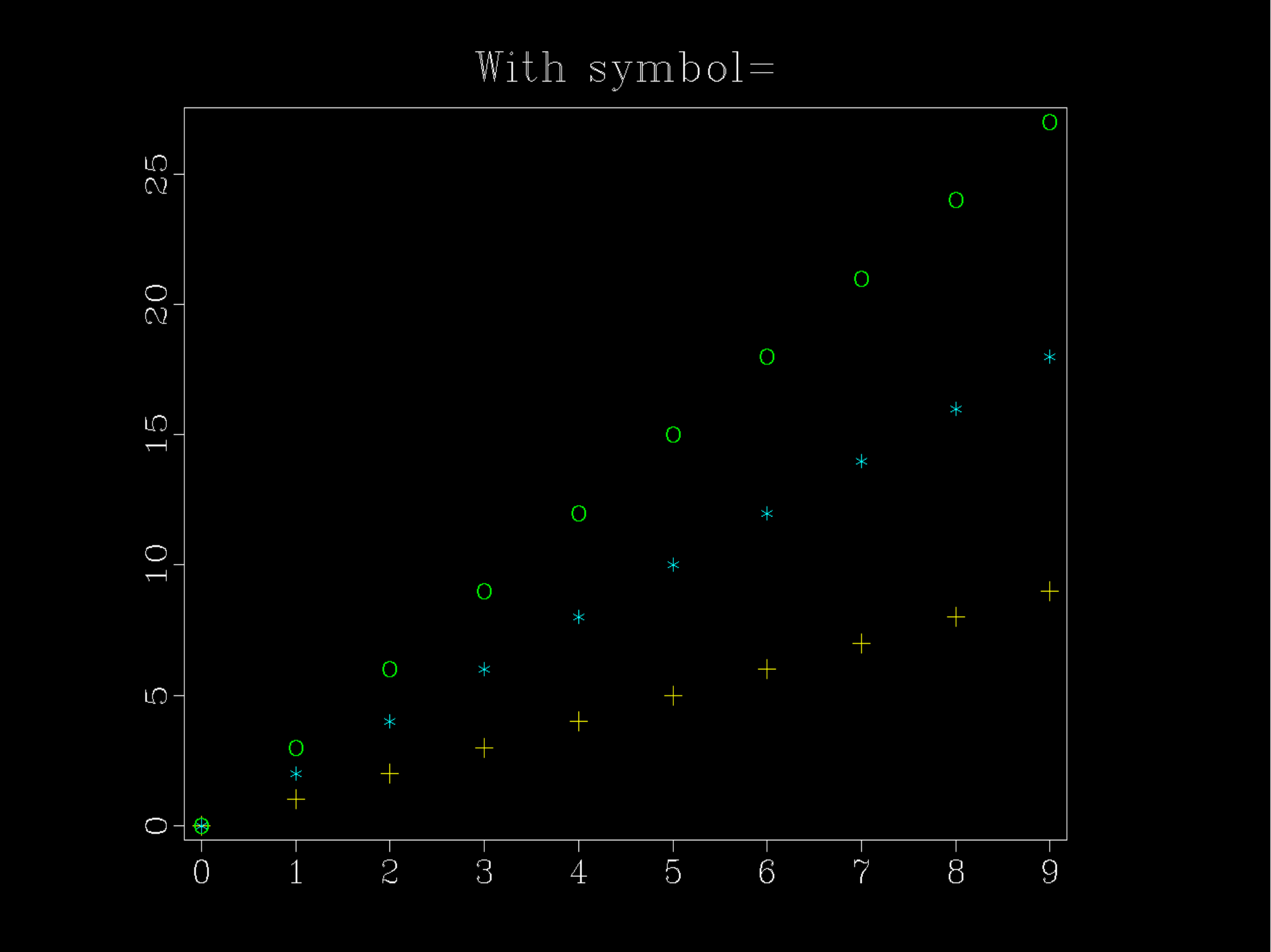



cos(x)
sfgraph
1
0.5
0
−0.5
−1
0
1
2
3
4
5
6
7
8
9
10
x

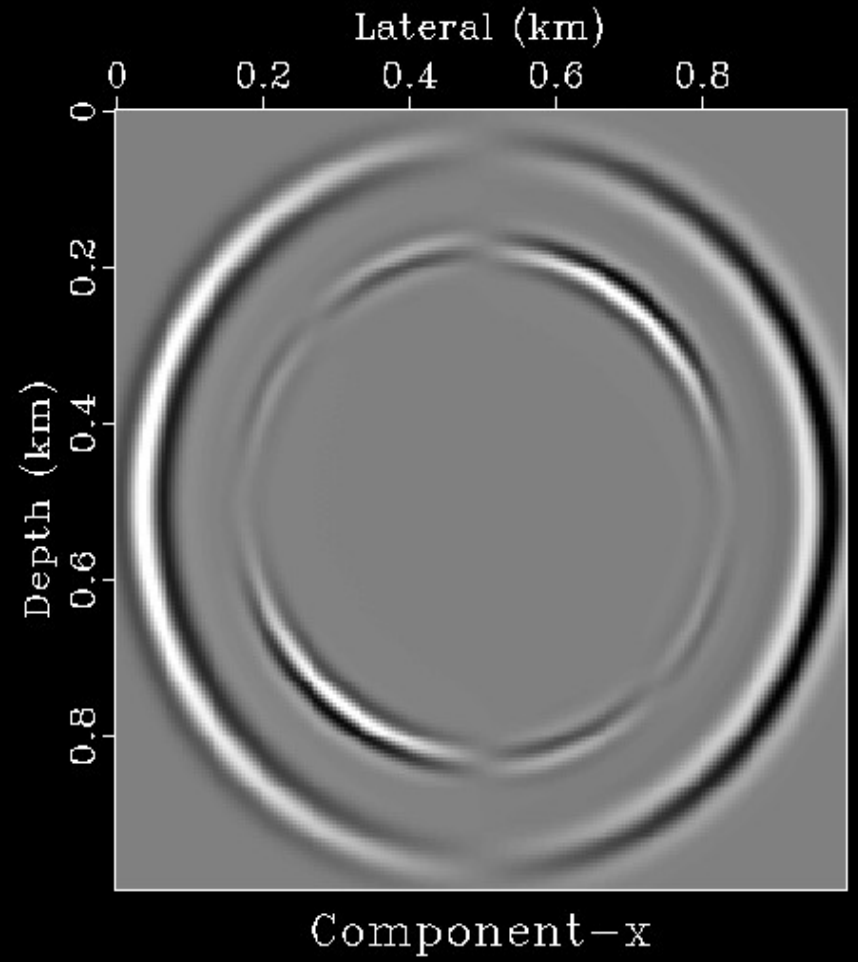

Line
sfgraph
0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9

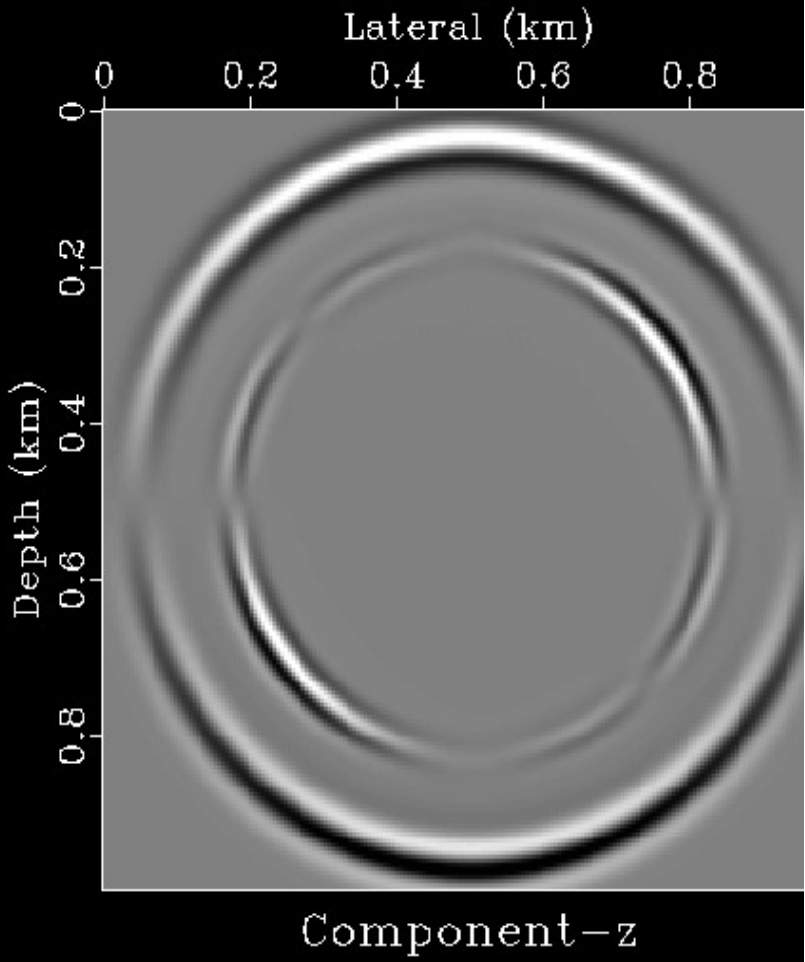

Line, x2
graph symbol="+" ...
0
1
2
3
4
5
6
7
8
9
0
10
20
30
40
50
60
70
80

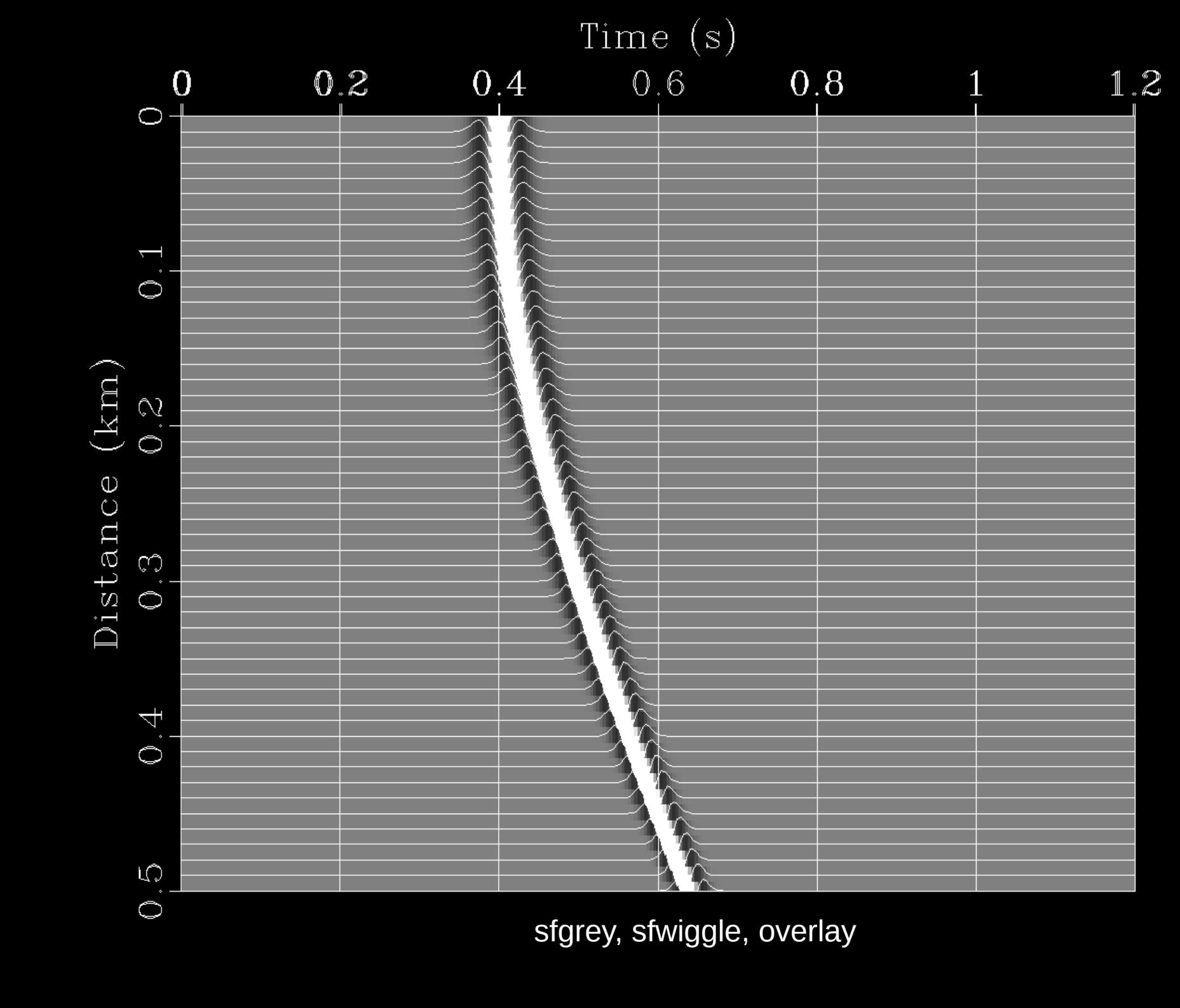

Multiple lines, n3 > 1
0
5
10
15
20
25
0
1
2
3
4
5
6
7
8
9
graph


With symbol=
0
5
10
15
20
25
0
1
2
3
4
5
6
7
8
9

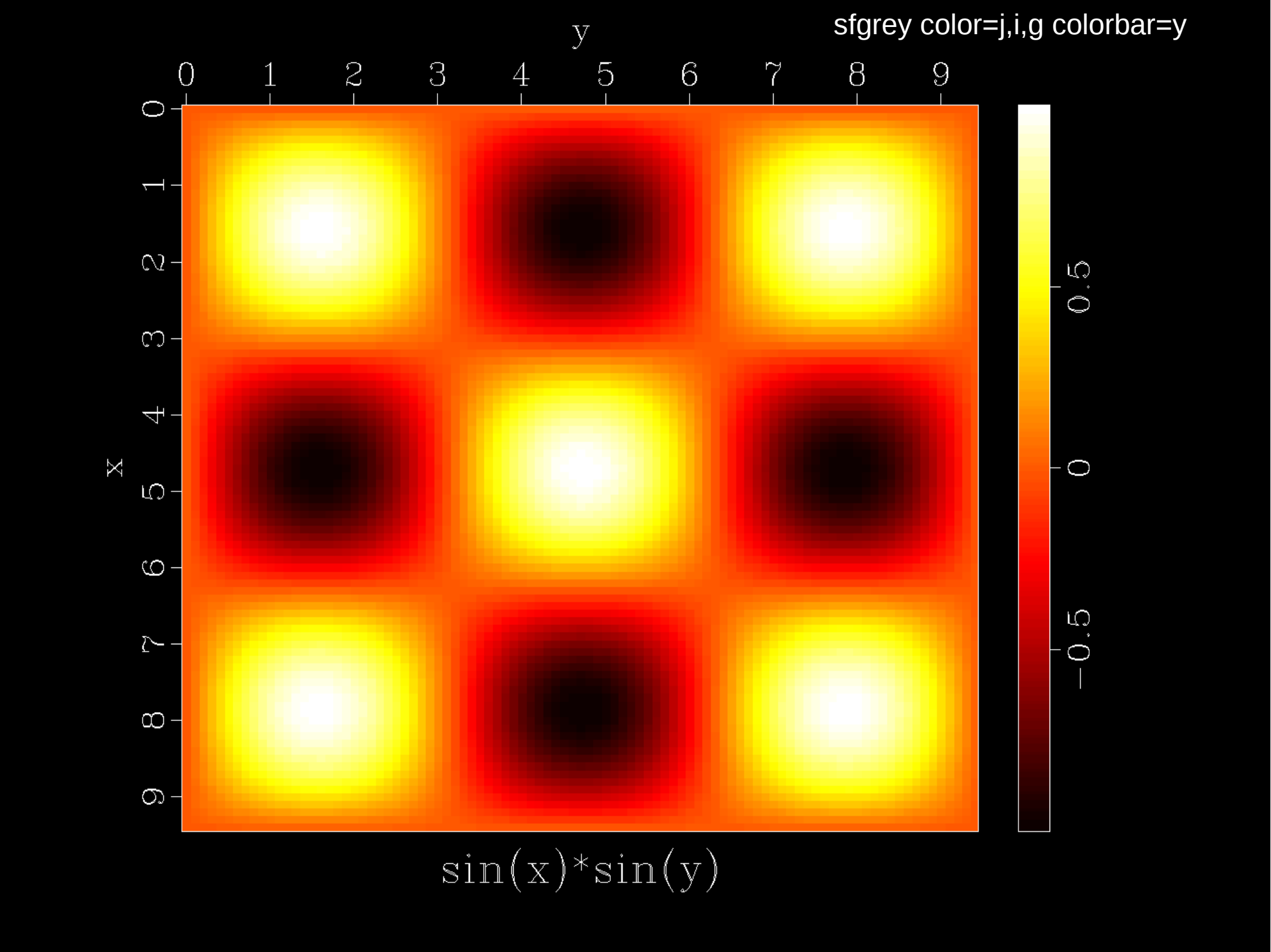
Plot(...,'sfgrey color=i ...');
Result(...,'SideBySideIso')

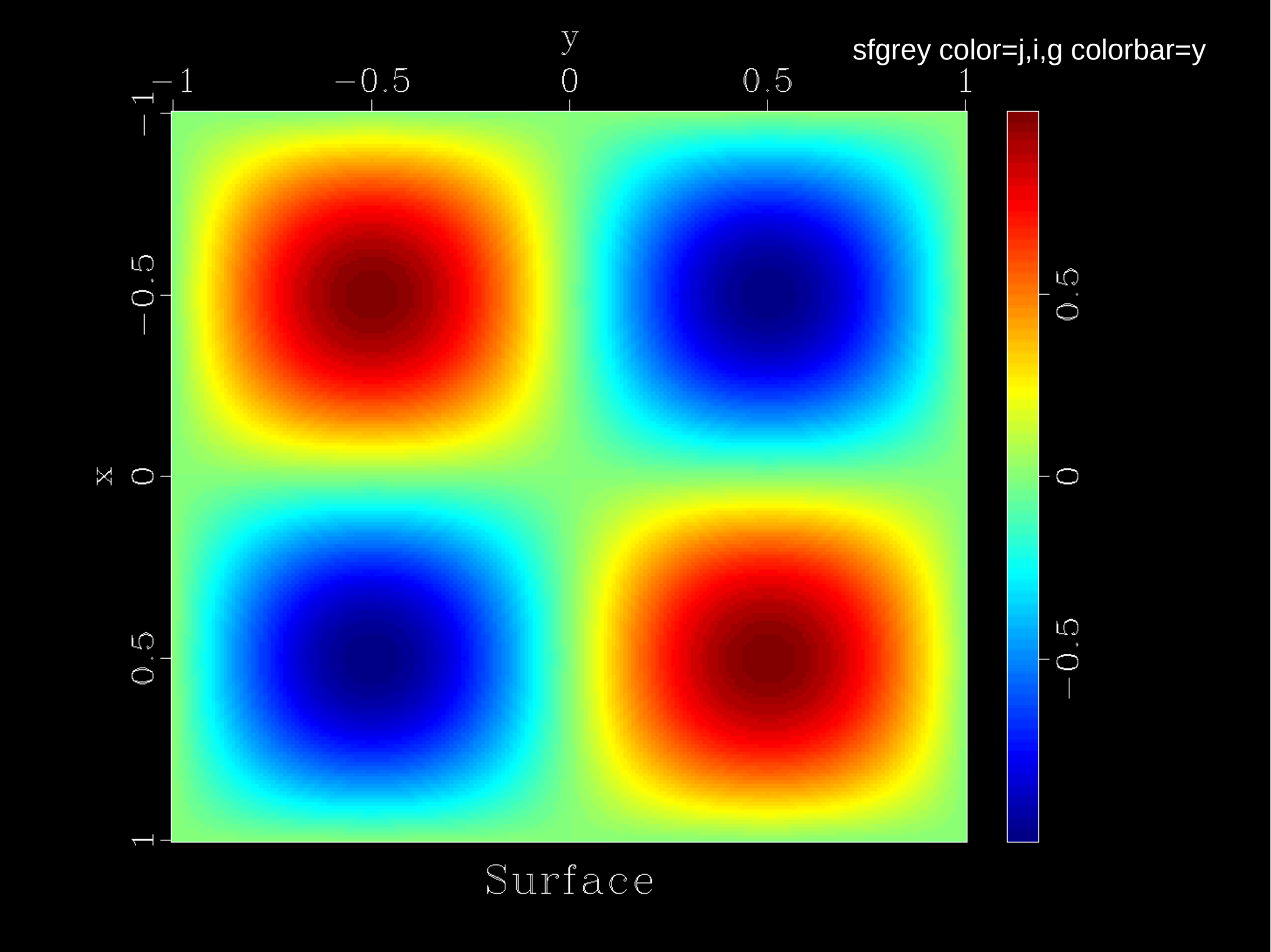


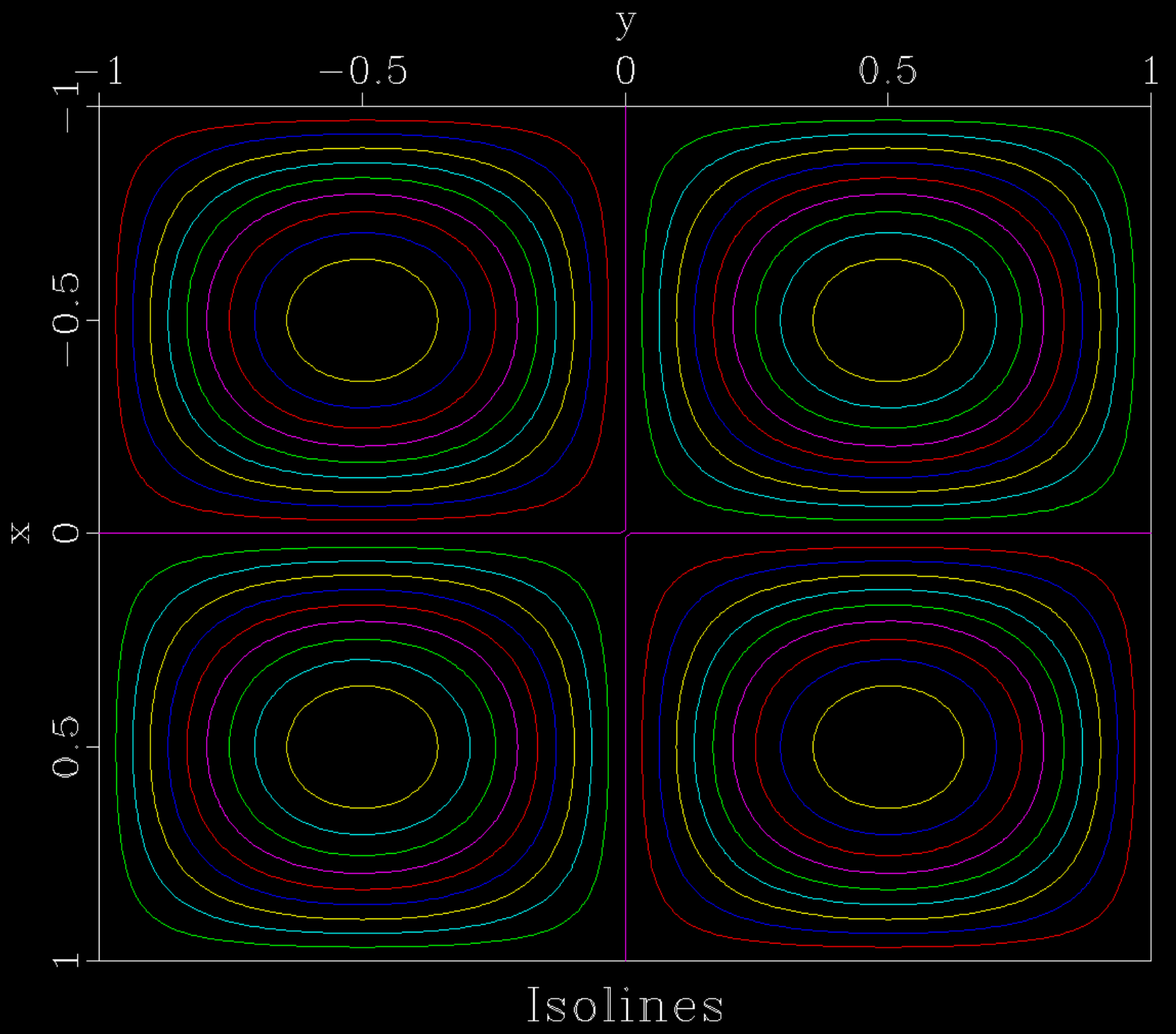




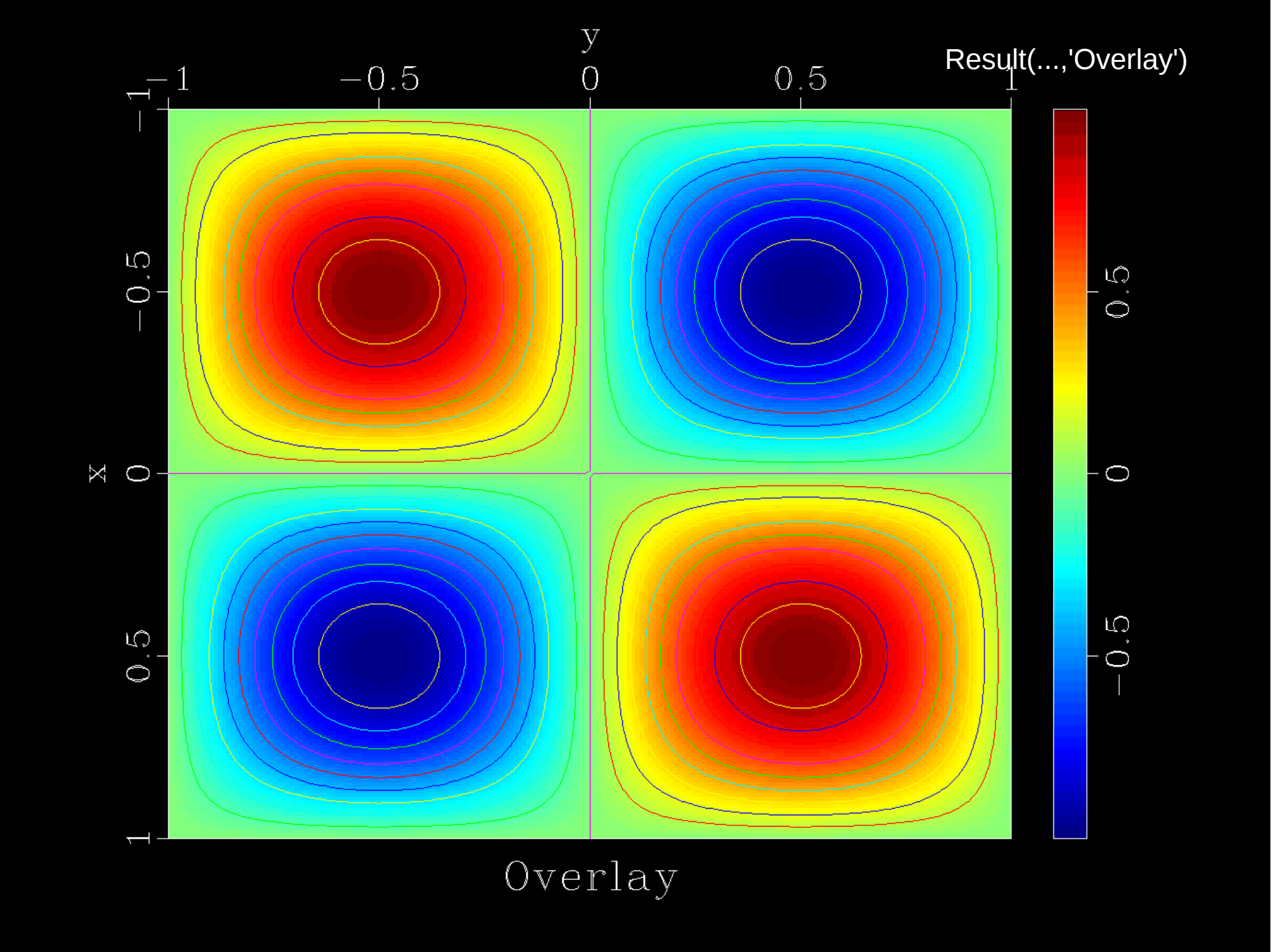
sfgrey, sfwiggle, overlay

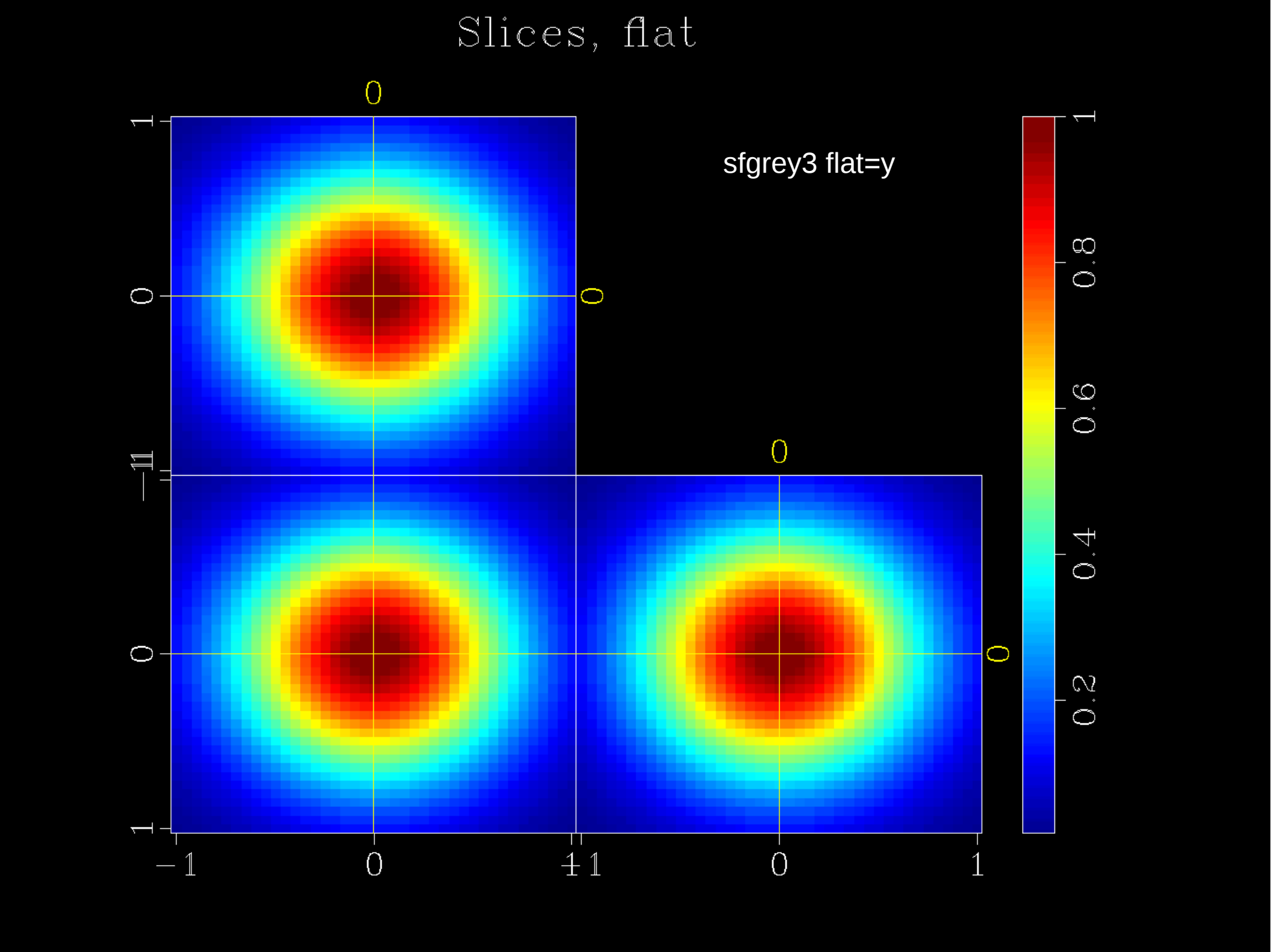

sfgrey color=j,i,g colorbar=y
y
0 1 2 3 4 5 6 7 8 9
x
0 1 2 3 4 5 6 7 8 9
0.5
0
−0.5
sin(x)*sin(y)

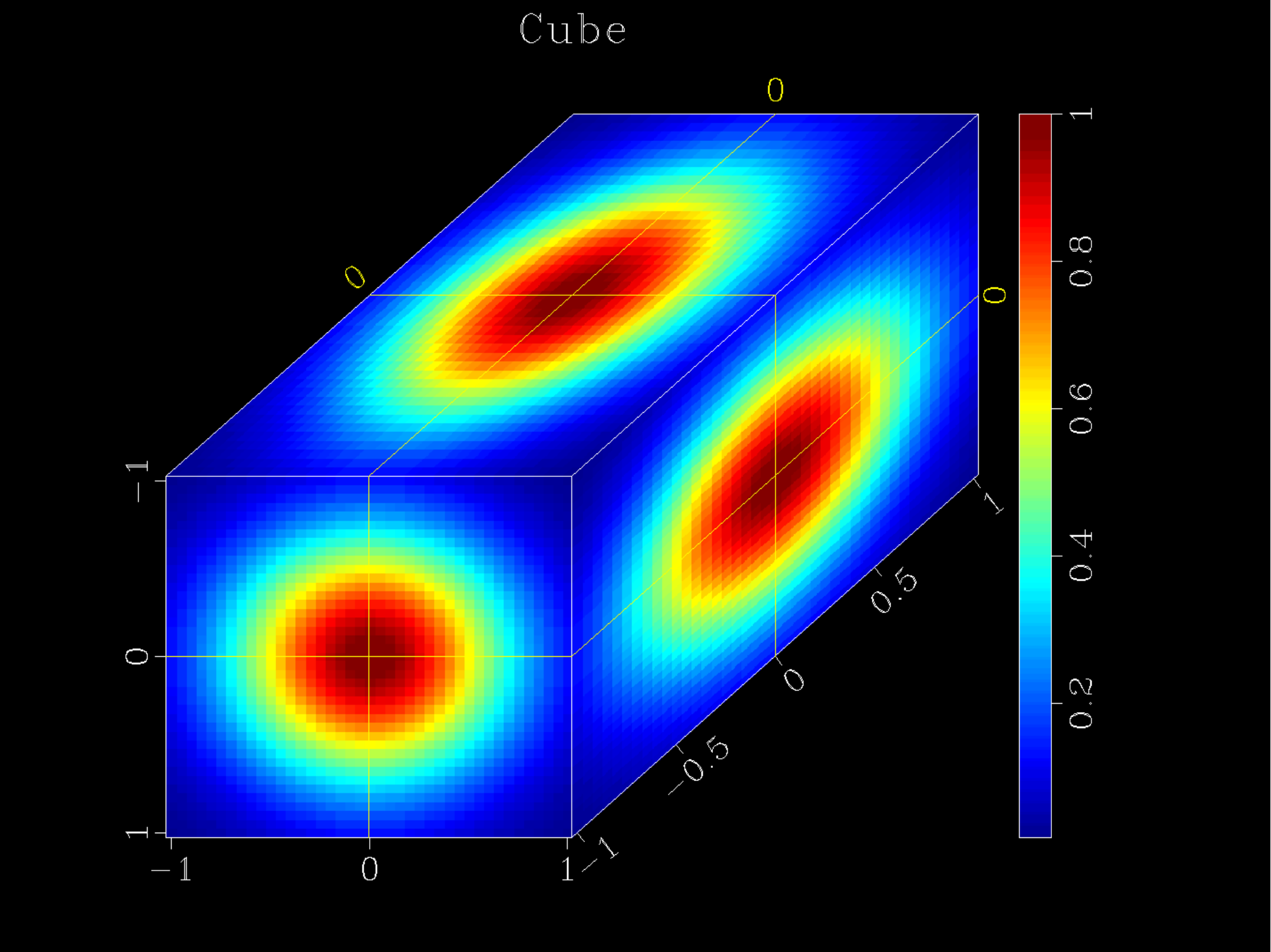

sfgrey color=j,i,g colorbar=y
y
−1
−0.5
0
0.5
1
x
−1
−0.5
0
0.5
1
0.5
0
−0.5
Surface



sfcontour


Result(...,'Overlay')
y
−1
−0.5
0
0.5
1
x
−1
−0.5
0
0.5
1
0.5
0
−0.5
Overlay


Slices, flat
sfgrey3 flat=y
0
0
0
0
1
0
−1
1
0
−1
1
0
−1
0
1
0.2
0.4
0.6
0.8
1


Cube
0
0
0
−1
0
1
−1
0
1
−1
−0.5
0
0.5
1
0.2
0.4
0.6
0.8
1

Plotting

- Numerous ways to make plots in Madagascar:

List of available plotting programs in Madagascar.	
sfbox	make box-line plots
sfcontour	make contour plots
sfcontour3	make contour plots of 3D surfaces
sfdots	plot signal with lollipops
sfgraph	create line plots, or scatter plots
sfgraph3	generate 3-D cube plots for surfaces.
sfgrey	create raster plots or 2D image plots
sfgrey3	create 3D image plots of panels (or slices) of a 3D cube
sfgrey4	generate movies of 3-D cube plots
sfplotrays	make plots of rays
sfthplot	make hidden-line surface plots
sfwiggle	plot data with wiggly traces

Plotting

- Numerous ways to make plots in Madagascar:

List of available plotting programs in Madagascar.	
sfbox	make box-line plots
sfcontour	make contour plots
sfcontour3	make contour plots of 3D surfaces
sfdots	plot signal with lollipops
sfgraph	create line plots, or scatter plots
sfgraph3	generate 3-D cube plots for surfaces.
sfgrey	create raster plots or 2D image plots
sfgrey3	create 3D image plots of panels (or slices) of a 3D cube
sfgrey4	generate movies of 3-D cube plots
sfplotrays	make plots of rays
sfthplot	make hidden-line surface plots
sfwiggle	plot data with wiggly traces

My most commonly used plotting functions

Let’s generate a plot of our file

- sfgraph

Let’s generate a plot of our file

- sfgraph
- sfgraph < file3.rsf > file3.vpl

Vplot

- VPLOT provides a method for making plots that are small in size, aesthetically pleasing, and easily compatible with Latex for rapid creation of production-quality images in Madagascar.

- The VPLOT file format (.vpl suffix) is a self-contained binary data format that describes in vector format how to draw a plot on the screen or a page using an interpreter. Since VPLOT is not a standard imaging format, VPLOT files must be viewed with interpreter programs which, for historical reasons, are called pens. Each pen interfaces VPLOT with a third-party graphing library such as X11, plplot, opengl, and others.

- This flexibility makes VPLOT files almost as portable as standard image formats such as: EPS, GIF, JPEG, PDF, PNG, SVG, and TIFF. Unlike rasterized formats, VPLOT files can be scaled to any size without losing image quality.

How to view vpl files?

- Method 1:
  - Use vpconvert to convert from vplot to standard format (e.g. avi, gif, jpg, pdf, etc)
- Method 2:
  - Use sfpen to plot to screen

Sfpen

- sfpen renders your plot to the screen

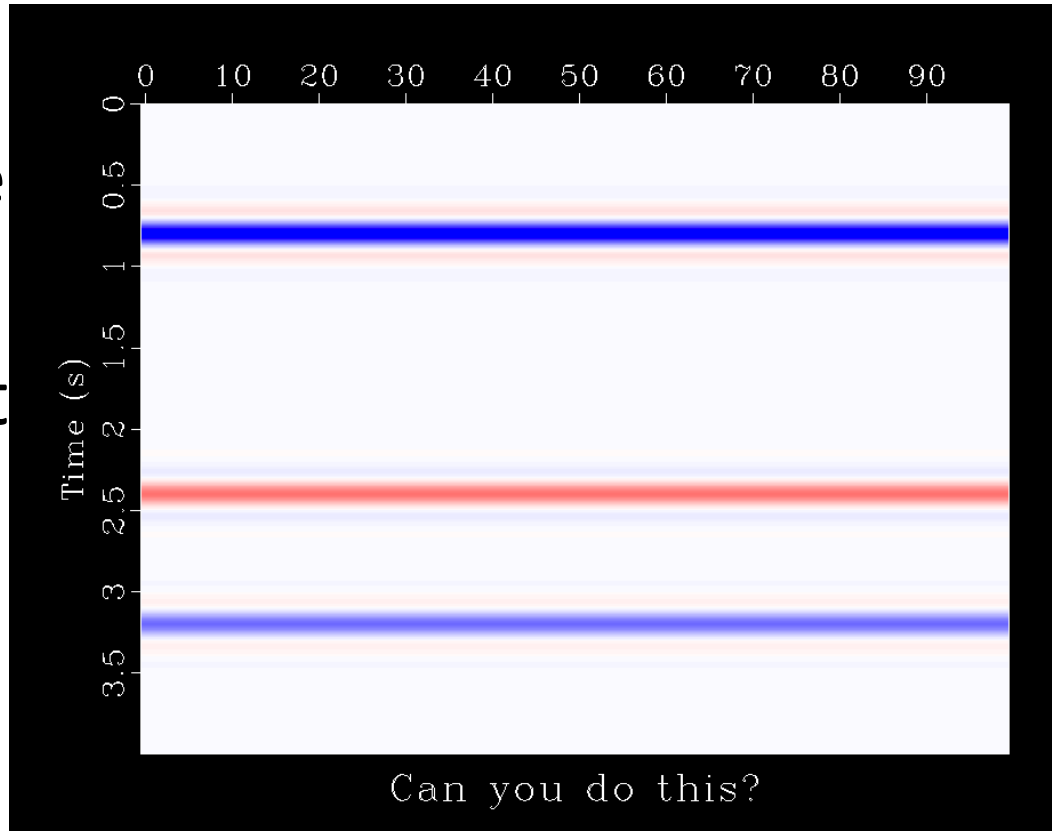
- < file3.vpl sfpen

Everything that we did could be done in 1 line

- sfspike n1=100 k1=20 |sfbandpass flo=1 fhi=20 |sfgraph |sfpen

Pop Quiz

- Use:
  - sfspike, sfpen, sfspray, sfbandpass, sfgrey
  - To make something like this:
  - Bonus: can you use sfwindow to extract and plot a single trace?



Solution:

- sfspike nsp=3 n1=1000 k1=200,600,800 mag=1,-0.5,.5 | sfbandpass fhi=5 | sfspray axis=2 n=100 | sfgrey color=e title="Can you do this?"| sfpen &