EXTRA EXERCISE

Luke Decker

Outline

- Download the files
- Beginning the exercise
- Initializing Madagascar
- Generating sine function
- Header information
- 1D plotting
- Showing simultaneous plots
- Take a derivative
- Generate cosine
- Multiply data
- Integrate data
- Ending SConstruct

Download the files

If you have internet:

- Navigate to our working directory

```
cd $RSFSRC/book/rsf/school/
svn update
cd trig
```

- If you do not have internet, but have the school files:

```
cd ~/M8Rschool/trig
```

Beginning the exercise

- trig:
  - Basic Madagascar SConstruct
  - Calling functions
  - Graphical plotting

Beginning the exercise

- Make sure you are in trig
- Open SConstruct with your favorite editor

  gedit SConstruct
- Run

  scons view
- You should see a lot of plots that are constantly zero

luke@begws86:~/RSFSRC/book/rsf/school/trig

```
[luke@begws86 trig]$ gedit SConstruct &
[1] 55668
[luke@begws86 trig]$ scons view
scons: Reading SConscript files ...
scons: done reading SConscript files.
```

SConstruct (~/RSFSRC/book/rsf/school/trig) - g

File Edit View Search Tools Documents Help

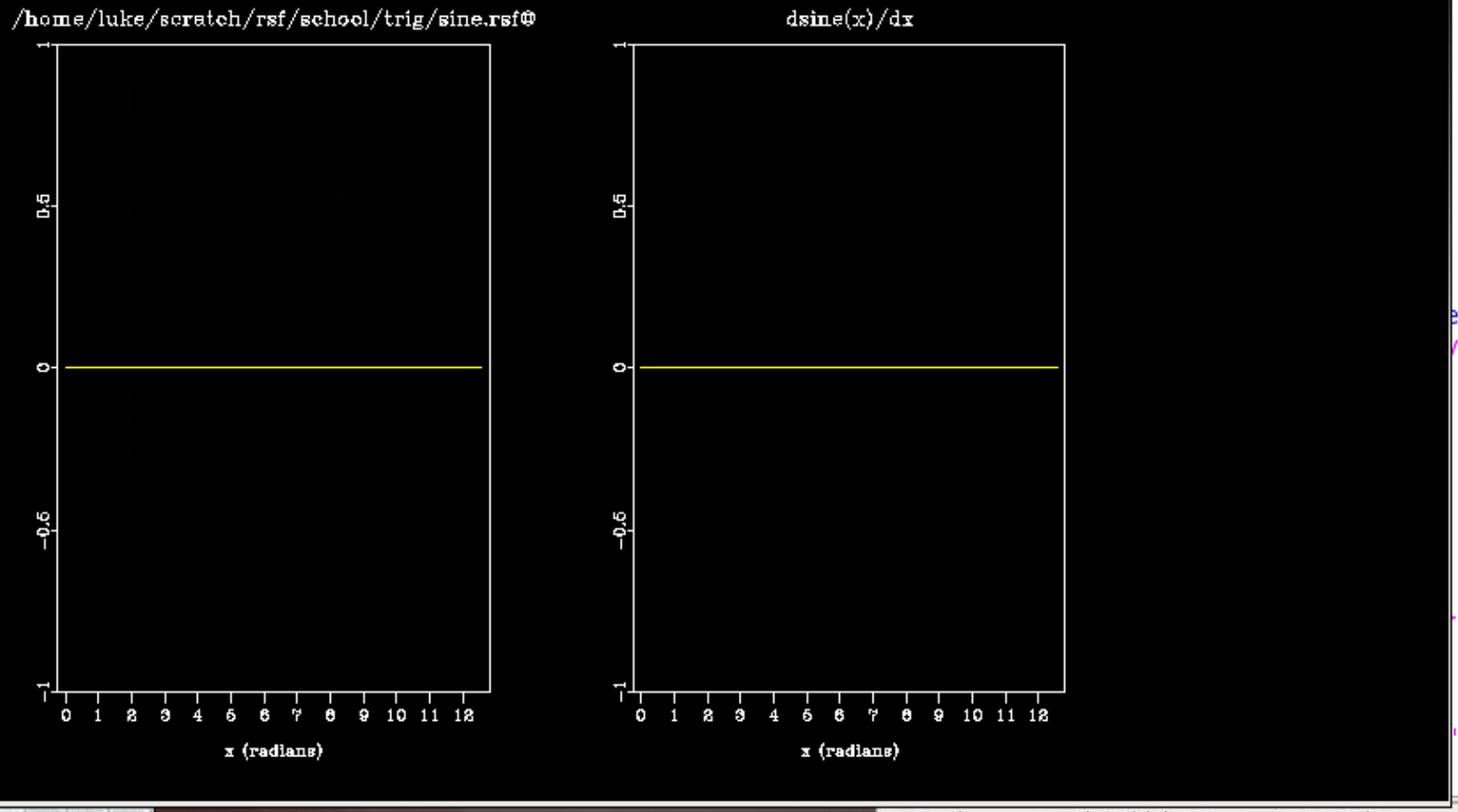
Open Save Undo

SConstruct

```
from rsf.proj import *
```

xtpen

Next Prev Quit Restart Run Stop Stretchy Forwards 0 delay 0.05



Python Tab Width: 8 Ln 1, Col 1 INS

luke@begws86:~/RSF... SConstruct (~/RSFSR... xtpen

Beginning the exercise

```
from rsf.proj import *

# generate a sine wave
  #modify below
Flow('sine',None,
     '''
     math output="0"
     d1=.01 n1=1256 o1=0
     label1=x unit1="radians"
     ''')
# plot sine
Plot('sine','graph ')
```

Initializing Madagascar

```
from rsf.proj import *

# generate a sine wave
  #modify below
Flow('sine',None,
     '''
     math output="0"
     d1=.01 n1=1256 o1=0
     label1=x unit1="radians"
     ''')
# plot sine
Plot('sine','graph ')
```

Generate sine function

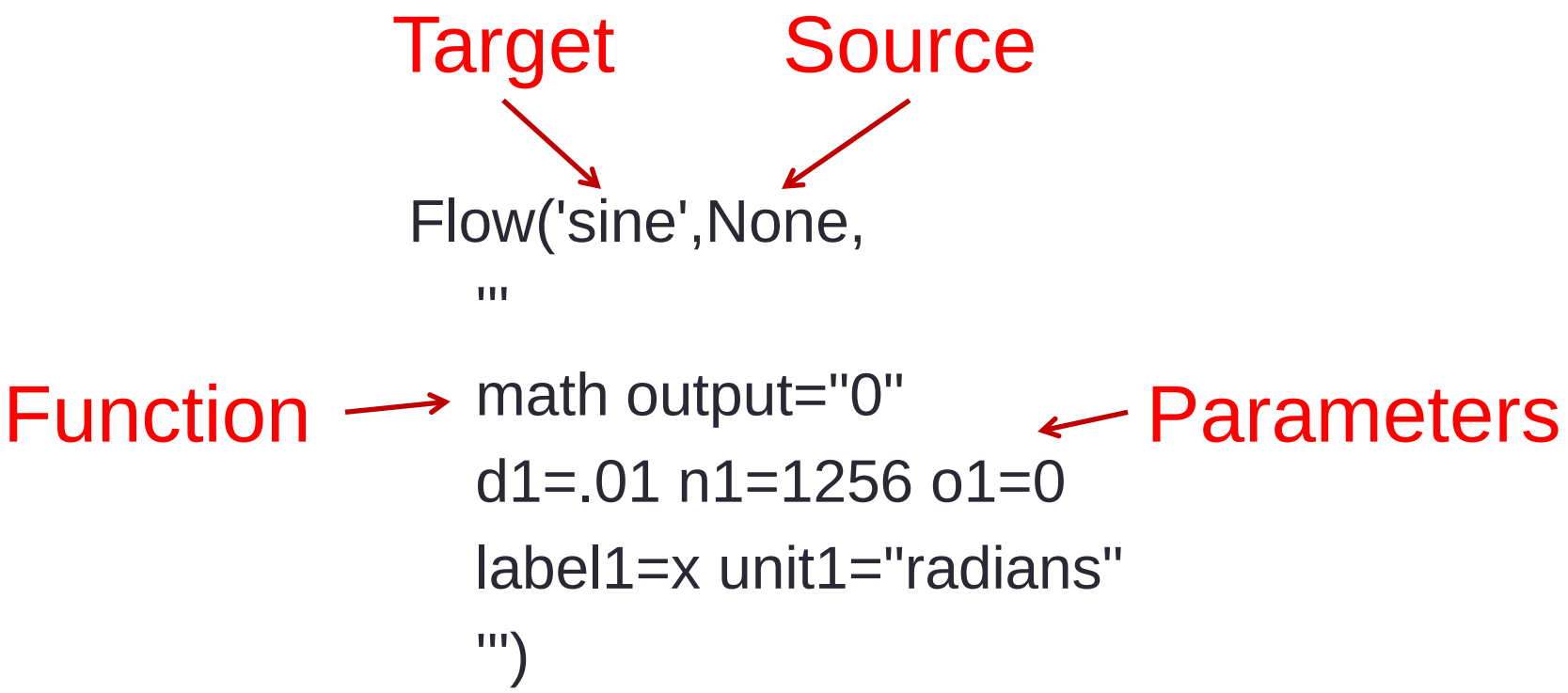
```
from rsf.proj import *

# generate a sine wave
  #modify below
Flow('sine',None,
     '''
     math output="0"
     d1=.01 n1=1256 o1=0
     label1=x unit1="radians"
     ''')
# plot sine
Plot('sine','graph ')
```

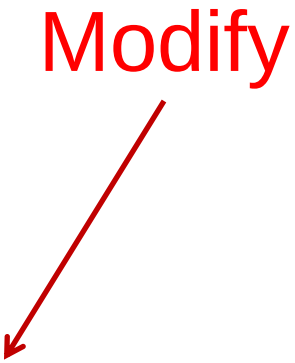
Generate sine function

Flow(‘target’,’source’,’function’)

Generate sine function

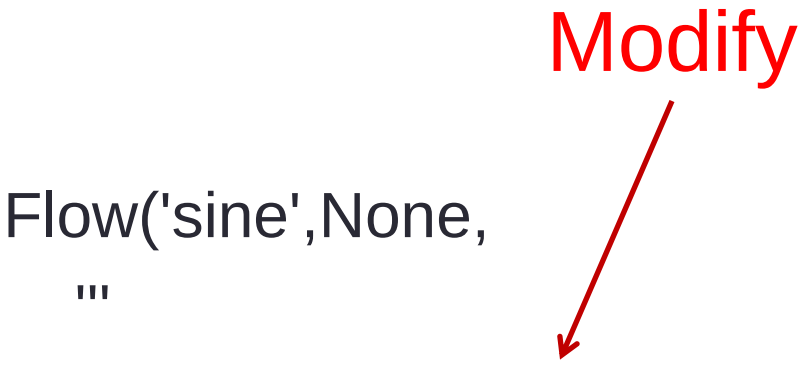


Generate sine function




```
Flow('sine',None,
    '''
    math output="0"
    d1=.01 n1=1256 o1=0
    label1=x unit1="radians"
    ''')
```

Generate sine function




```
Flow('sine',None,
    '''
    math output="sin(x1)"
    d1=.01 n1=1256 o1=0
    label1=x unit1="radians"
    '''
```

Generate sine function

- Run function

  scons sine.rsf

- Compare what appears in command line with what is in the SConstruct file

  /home/luke/rsf/bin/sfmath output="sin(x1)" d1=.01 n1=1256 o1=0 label1=x unit1="radians" > sine.rsf

- Examine function documentation

  sfmath

- Read header file

  sfin sine.rsf

Header information

- RSF or “regularly sampled file” format can be thought of as a high dimensional matrix.
- The axis corresponding to each dimension is parameterized by:
  - Axis origin → o#
  - Axis sampling → d#
  - Number of samples in axis → n#
- The rsf file in your working directory is a header
- Where is the data located?
- What else is in the header?

1D plotting

```
from rsf.proj import *

# generate a sine wave
  #modify below
Flow('sine',None,
     '''
     math output="0"
     d1=.01 n1=1256 o1=0
     label1=x unit1="radians"
     ''')
# plot sine
Plot('sine','graph ')
```

1D plotting

Plot(‘target’,’source’,’function’)

If target=source

Plot(‘target’,’function’)

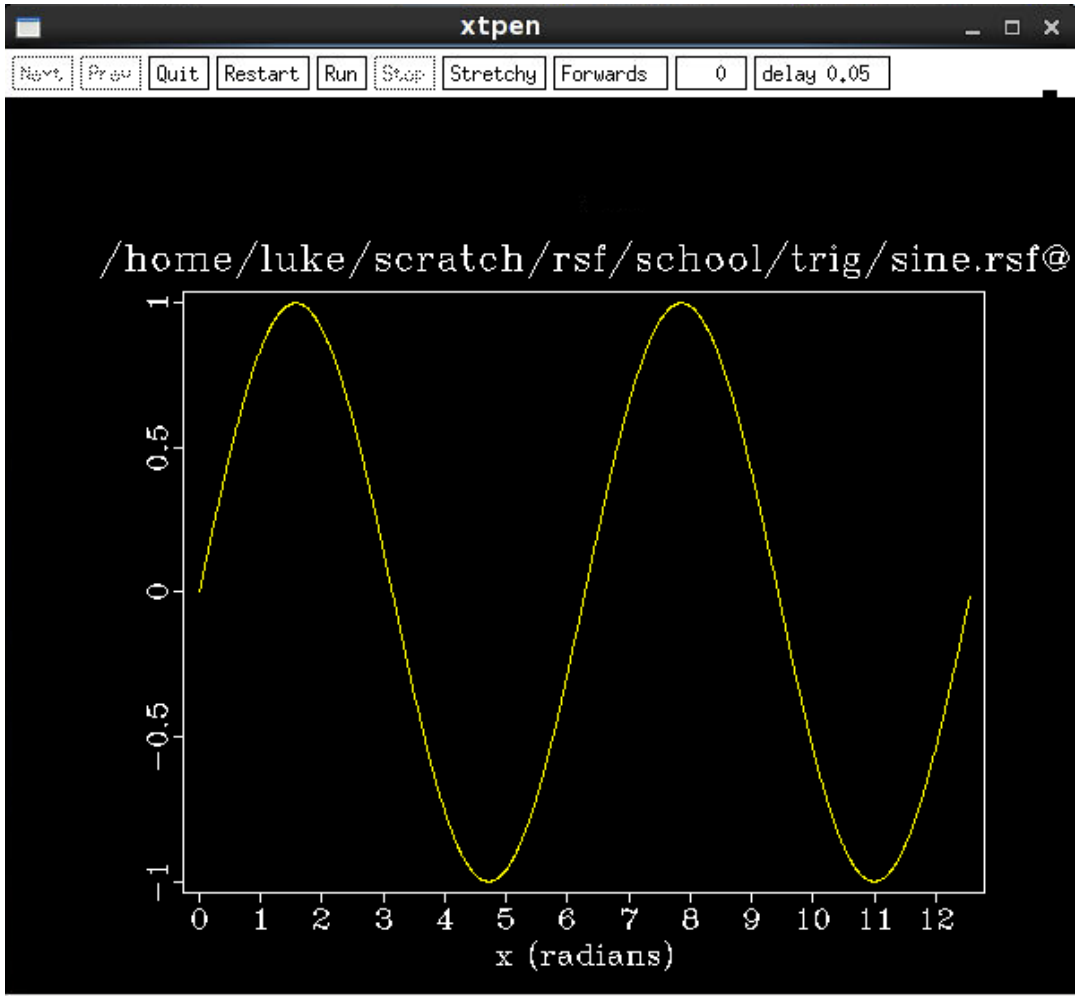
Plot may be replaced with Result

1D plotting

- To graph the sine curve

  scons sine.vpl

  sfpen sine.vpl



1D plotting

Modify

```
# plot sine
Plot('sine','graph ')
```

1D plotting

Modify

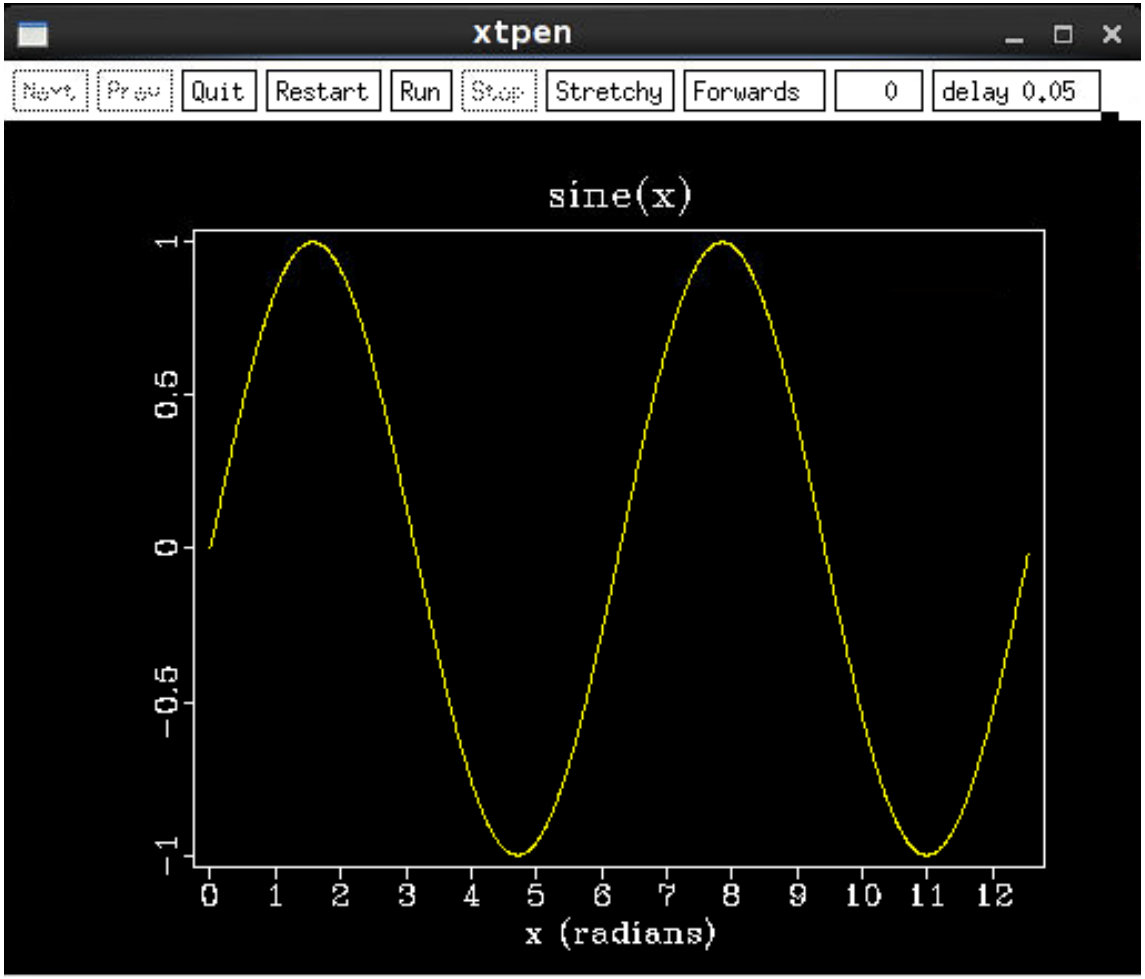
```
# plot sine
Plot('sine','graph title=sine(x)')
```

1D plotting

- To graph the sine curve

  scons sine.vpl

  sfpen sine.vpl



1D plotting

- sfpen is a tool to display *.vpl files
- What happens when you run sfin on sine.vpl?
- Where is sine.vpl located?
- We have done:

  None → sine.rsf → sine.vpl

Showing simultaneous plots

```
# take the derivative of the sine wave
  # modify below
Flow('dsine','sine','deriv ')
Plot('dsine','graph title="dsine(x)/dx"')
# plot the sine wave with its derivative
Result('sine_dsine','sine dsine','SideBySideAniso')
```

Showing simultaneous plots

- Result is like plot:
- It generates a *.vpl file
- And automically displays it on your screen, it has “built in” sfpen
- Places the *.vpl file in a different folder, Fig
- We will explore other plotting options, like calling two plots side by side

Take a derivative

```
# take the derivative of the sine wave
  # modify below
Flow('dsine','sine','deriv ')
Plot('dsine','graph title="dsine(x)/dx"')
# plot the sine wave with its derivative
Result('sine_dsine','sine dsine','SideBySideAniso')
```

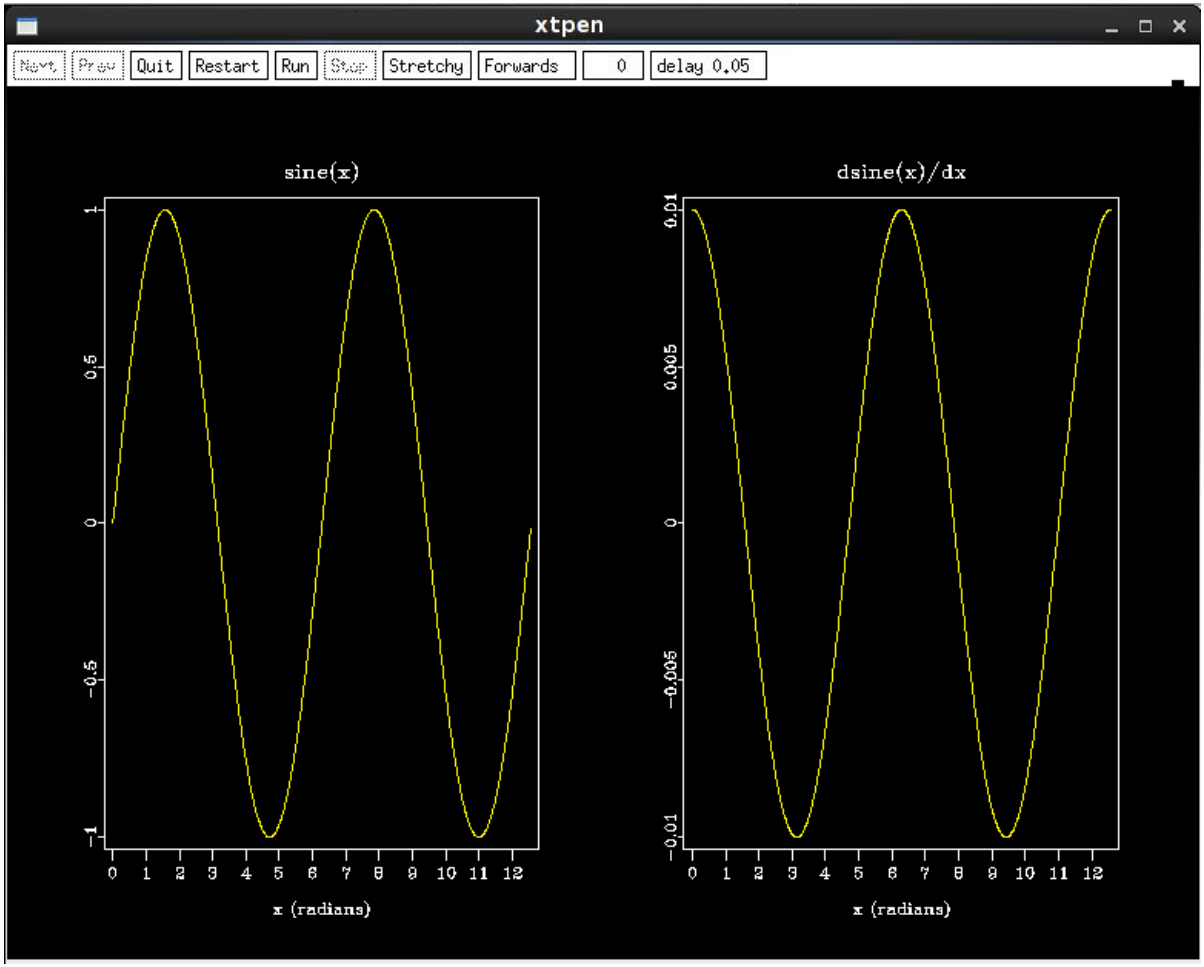
Take a derivative

Flow('dsine','sine','deriv ')

- What does deriv do?

  sfderiv
- Run

  scons sine_dsine.view
- What looks funny?
- What sfderiv parameter do we need?



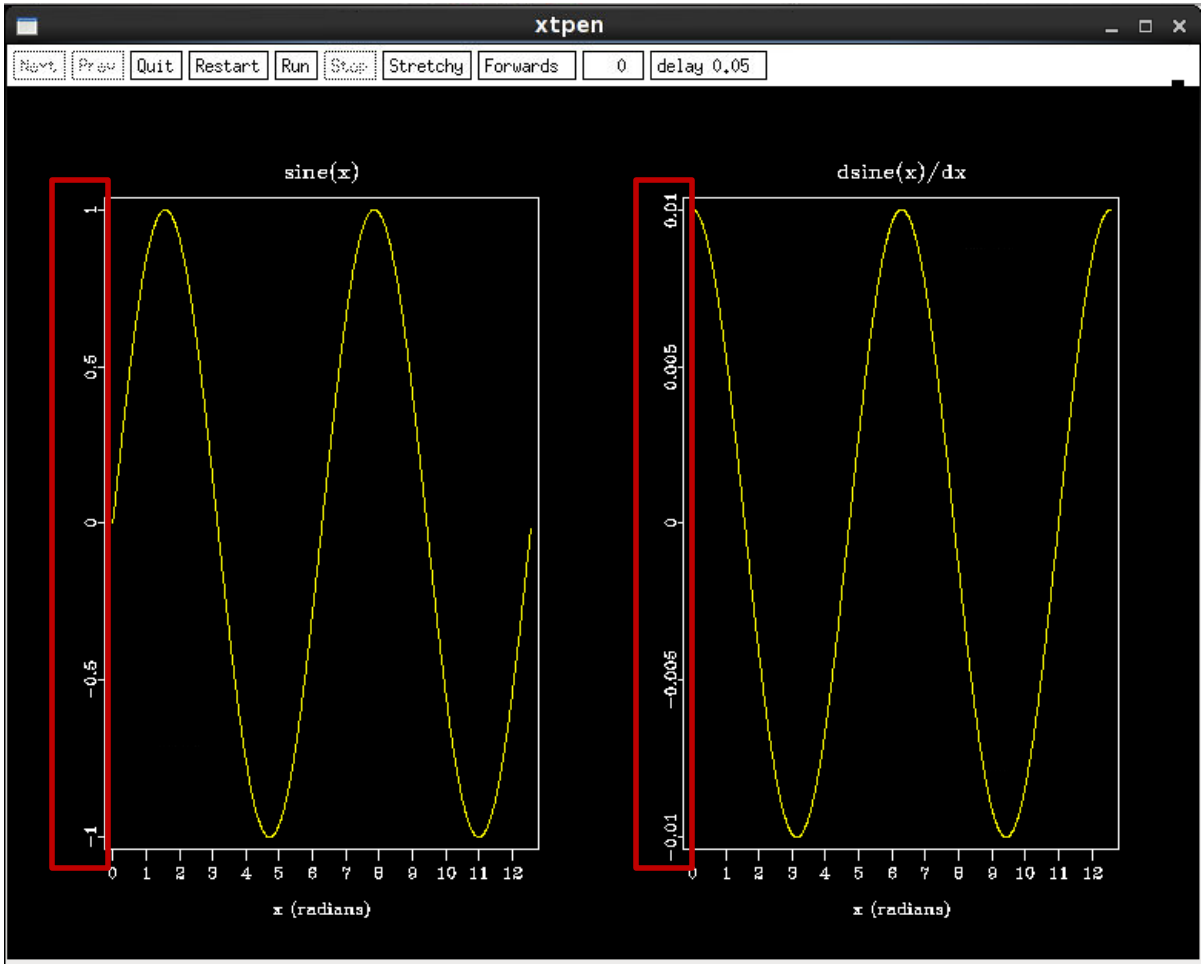
Take a derivative

Flow('dsine','sine','deriv ')

- What does deriv do?

  sfderiv
- Run

  scons sine_dsine.view
- What looks funny?
- What sfderiv parameter do we need?



Take a derivative

Modify

Flow('dsine','sine','deriv scale=y')

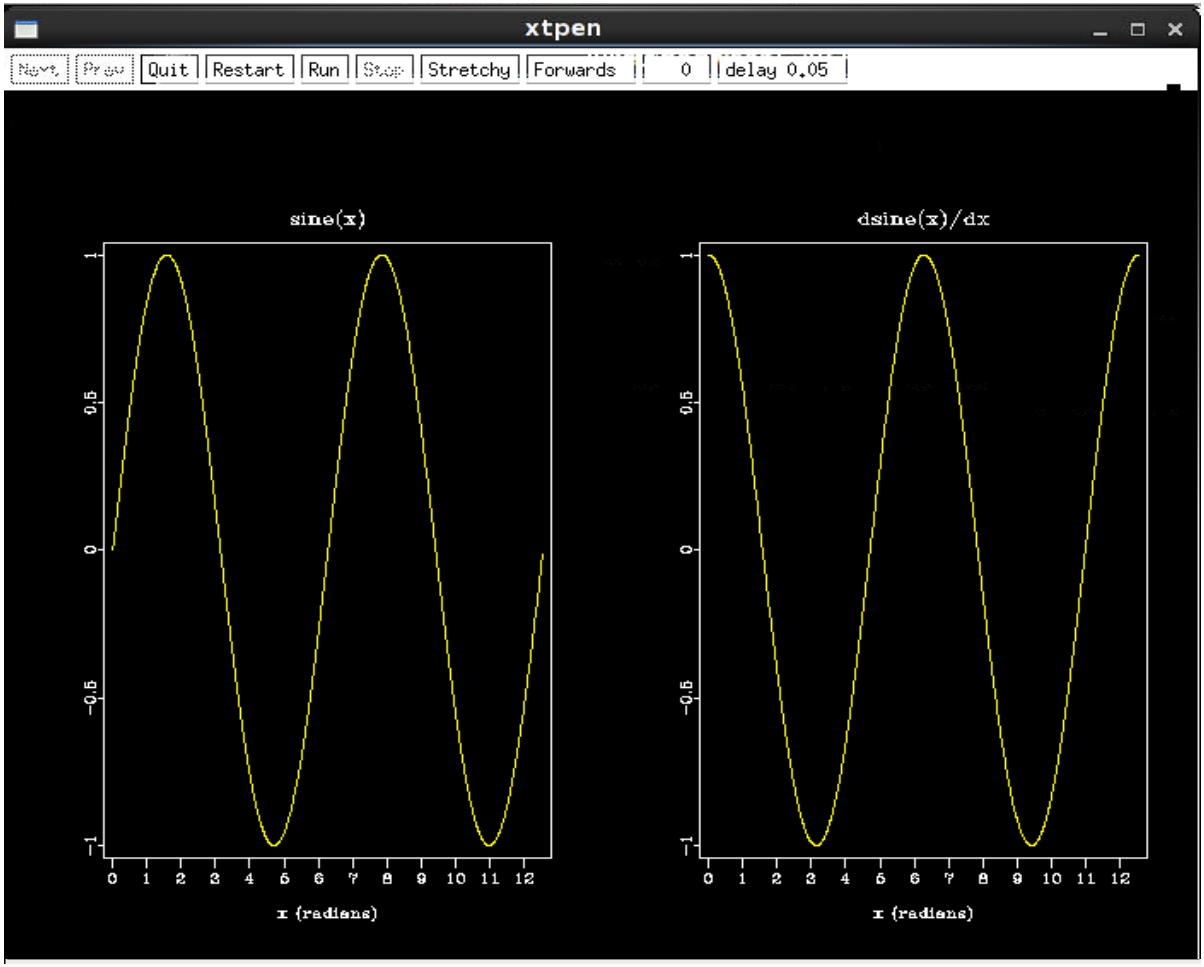
- What does deriv do?

  sfderiv

- Run

  scons sine_dsine.view

- What looks funny?
- What sfderiv parameter do we need?



Take a derivative

```
# take the derivative of the sine wave
  # modify below
Flow('dsine','sine','deriv scale=y')
Plot('dsine','graph title="dsine(x)/dx"')
# plot the sine wave with its derivative
Result('sine_dsine','sine dsine','SideBySideAniso')
```

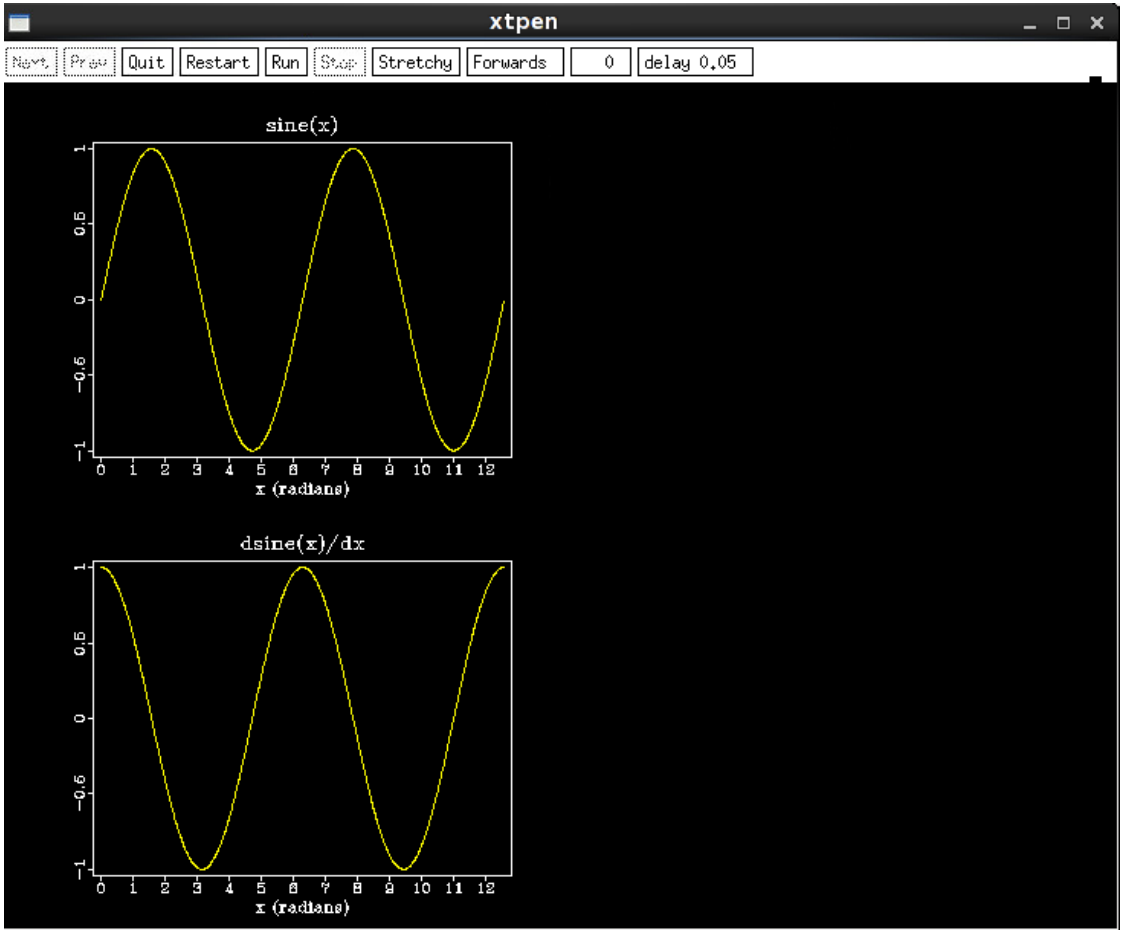
Take a derivative

Result('sine_dsine','sine dsine','SideBySideAniso')

Result('sine_dsine','sine dsine',‘OverUnderIso')

Take a derivative

- Run
  - scons sine_dsine.view
- Notice the plots are not streched to fit window shape: they are isotropic and maintain their aspect ratios



Generate cosine

```
# generate a cosine curve
Flow('cosine','sine',
     '''
     math output="cos(x1)"
     ''')
# get cosine squared by multiplying the derivative of sine with the
cosine function
  # modify below
Flow('cos2','dsine cosine',
     '''
     math A=${SOURCES[0]} B=${SOURCES[1]}
     output=".0"
     ''')
# plot the cosine squared function
  # modify below
Result('cos2','graph label1="cos\^2\_(x)" ')
```

Generate cosine

```
# generate a cosine curve
Flow('cosine','sine',
    '''
    math output="cos(x1)"
    ''')
```

We are now using sine.rsf as a source file

We don’t operate sine.rsf data, we just use its header to determine the dimensions of cosine.rsf

Multiply data

```
# generate a cosine curve
Flow('cosine','sine',
     '''
     math output="cos(x1)"
     ''')
# get cosine squared by multiplying the derivative of sine with the
cosine function
  # modify below
Flow('cos2','dsine cosine',
     '''
     math A=${SOURCES[0]} B=${SOURCES[1]}
     output=".0"
     ''')
# plot the cosine squared function
  # modify below
Result('cos2','graph label1="cos\^2\_(x)" ')
```

Multiply data

- We already have a cosine function from the derivative of sine(x)
- To create $\cos^2 x$ we multiply that derivative by the cosine function

```
# get cosine squared by multiplying the derivative of sine with the cosine function
  # modify below
Flow('cos2','dsine cosine',
    '''
    math A=${SOURCES[0]} B=${SOURCES[1]}
    output=".0"
    ''')
```

Multiply data

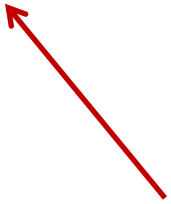
- We can call different source files in the function by treating the sources as a Python list.
- ${SOURCES[0]} is dsine.rsf
- ${SOURCES[1]} is cosine.rsf
- Why doesn't this work?

```
# get cosine squared by multiplying the derivative
of sine with the cosine function
  # modify below
Flow('cos2','dsine cosine',
    '''
    math A=${SOURCES[0]} B=${SOURCES[1]}
    output=".0"
    ''')
```

Multiply data

- We can call different source files in the function by treating the sources as a Python list.
- ${SOURCES[0]} is dsine.rsf
- ${SOURCES[1]} is cosine.rsf
- Why doesn't this work?

```
# get cosine squared by multiplying the derivative
of sine with the cosine function
  # modify below
Flow('cos2','dsine cosine',
    '''
    math A=${SOURCES[0]} B=${SOURCES[1]}
    output=".0"
    '''')
```

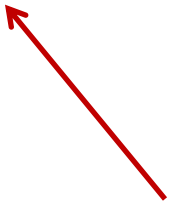


Modify

Multiply data

- We can call different source files in the function by treating the sources as a Python list.
- ${SOURCES[0]} is dsine.rsf
- ${SOURCES[1]} is cosine.rsf
- Why doesn't this work?

```
# get cosine squared by multiplying the derivative
of sine with the cosine function
  # modify below
Flow('cos2','dsine cosine',
    '''
    math A=${SOURCES[0]} B=${SOURCES[1]}
    output="A*B"
    '''')
```



Modify

Multiply data

```
# generate a cosine curve
Flow('cosine','sine',
     '''
     math output="cos(x1)"
     ''')
# get cosine squared by multiplying the derivative of sine with the
cosine function
  # modify below
Flow('cos2','dsine cosine',
     '''
     math A=${SOURCES[0]} B=${SOURCES[1]}
     output="A*B"
     ''')
# plot the cosine squared function
  # modify below
Result('cos2','graph label1="cos\^2\_(x)" ')
```

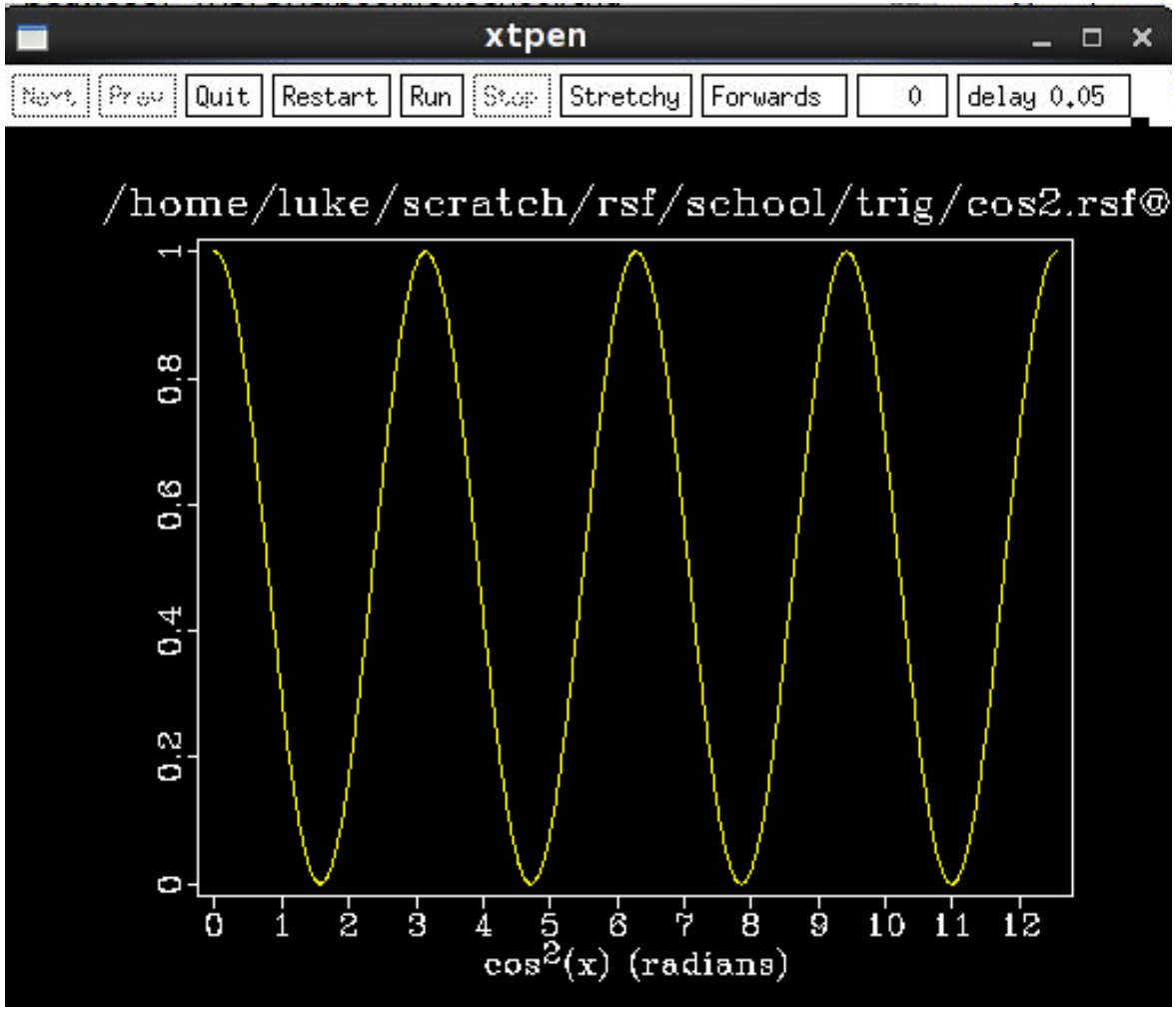
Multiply data

- Run

  scons cos2.view

- What looks funny?

```
# plot the cosine squared function
  # modify below
Result('cos2','graph label1="cos\^2\_(x)" ')
```



Multiply data

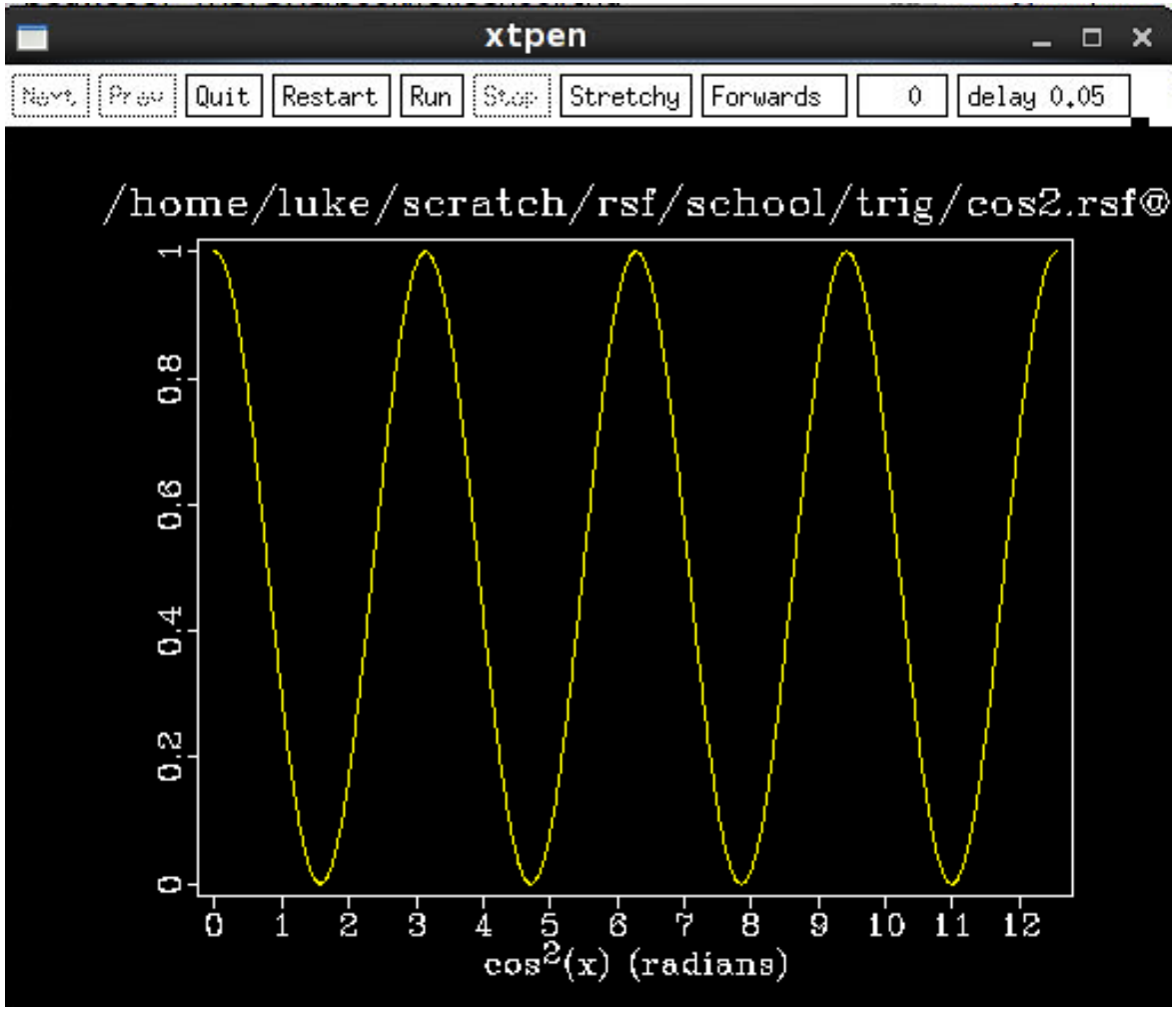
- Run

  scons cos2.view

- What looks funny?

```
# plot the cosine squared function
  # modify below
Result('cos2','graph label1="cos\^2\_(x)" ')
```

Modify



Multiply data

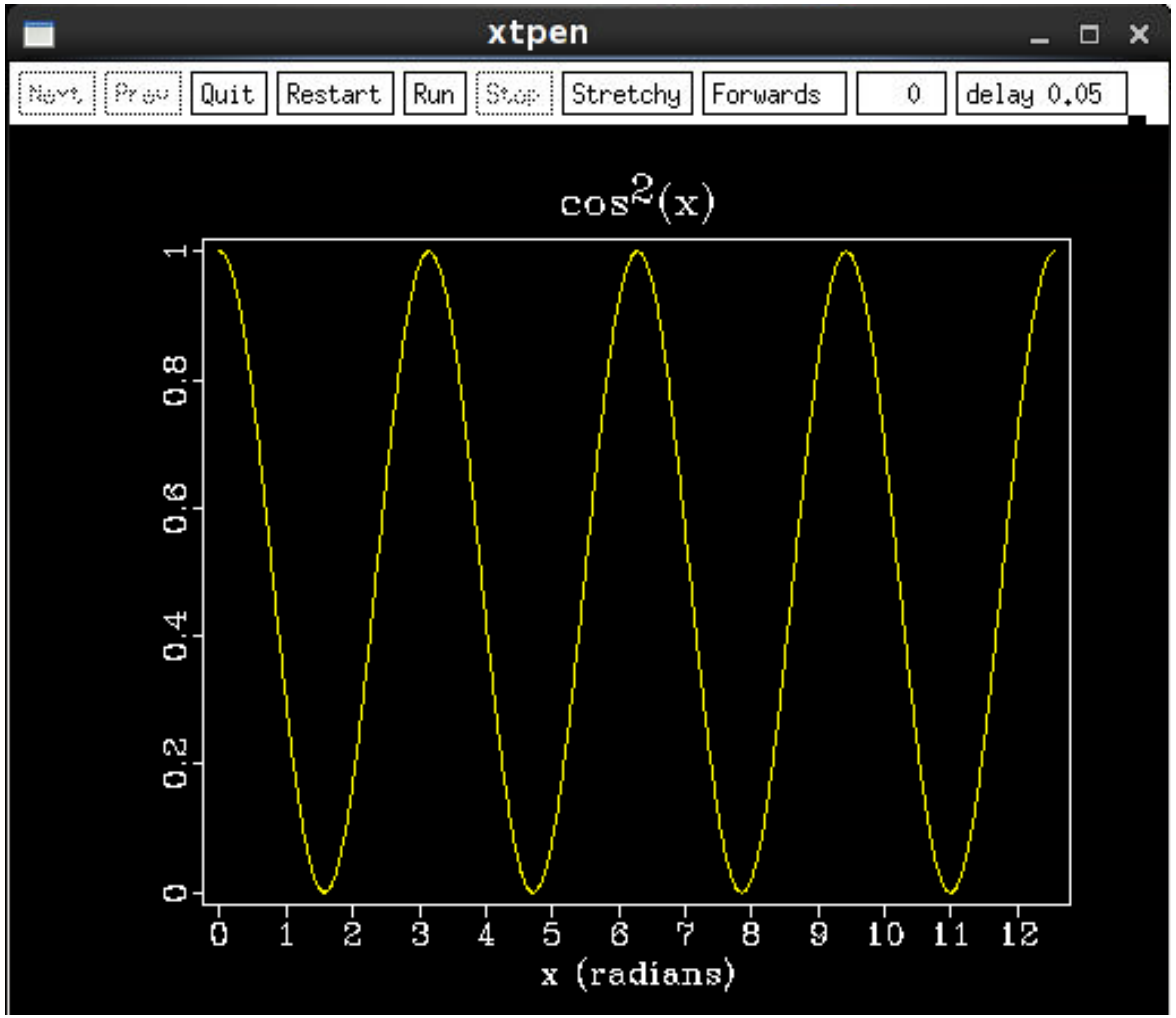
- Run

  scons cos2.view

- What looks funny?

Modify

```
# plot the cosine squared function
  # modify below
Result('cos2','graph title="cos\^2\_(x)" ')
```



Integrate data

```
# integrate cosine squared, scale by dx
  # modify below
Flow('int-cos2','cos2','cp | scale dscale=1')
# plot the integral
Result('int-cos2','graph title="x/2+sin(2x)/4"')
End()
```

Integrate data

- We want to integrate the cosine squared function
- What does sfcp do?
- To find an integrating program we will use sfdoc

  sfdoc –k integration
- What do you see?

```
# integrate cosine squared, scale by dx
  # modify below
Flow('int-cos2','cos2',‘cp | scale dscale=1')
```

Integrate data

- We will use sfcausint
- Type sfcausint for a description of the program

```
# integrate cosine squared, scale by dx
  # modify below
Flow('int-cos2','cos2',‘cp | scale dscale=1')
```

Modify

Integrate data

- We will use sfcausint
- Type sfcausint for a description of the program

```
# integrate cosine squared, scale by dx
  # modify below
Flow('int-cos2','cos2',‘causint | scale dscale=1')
```

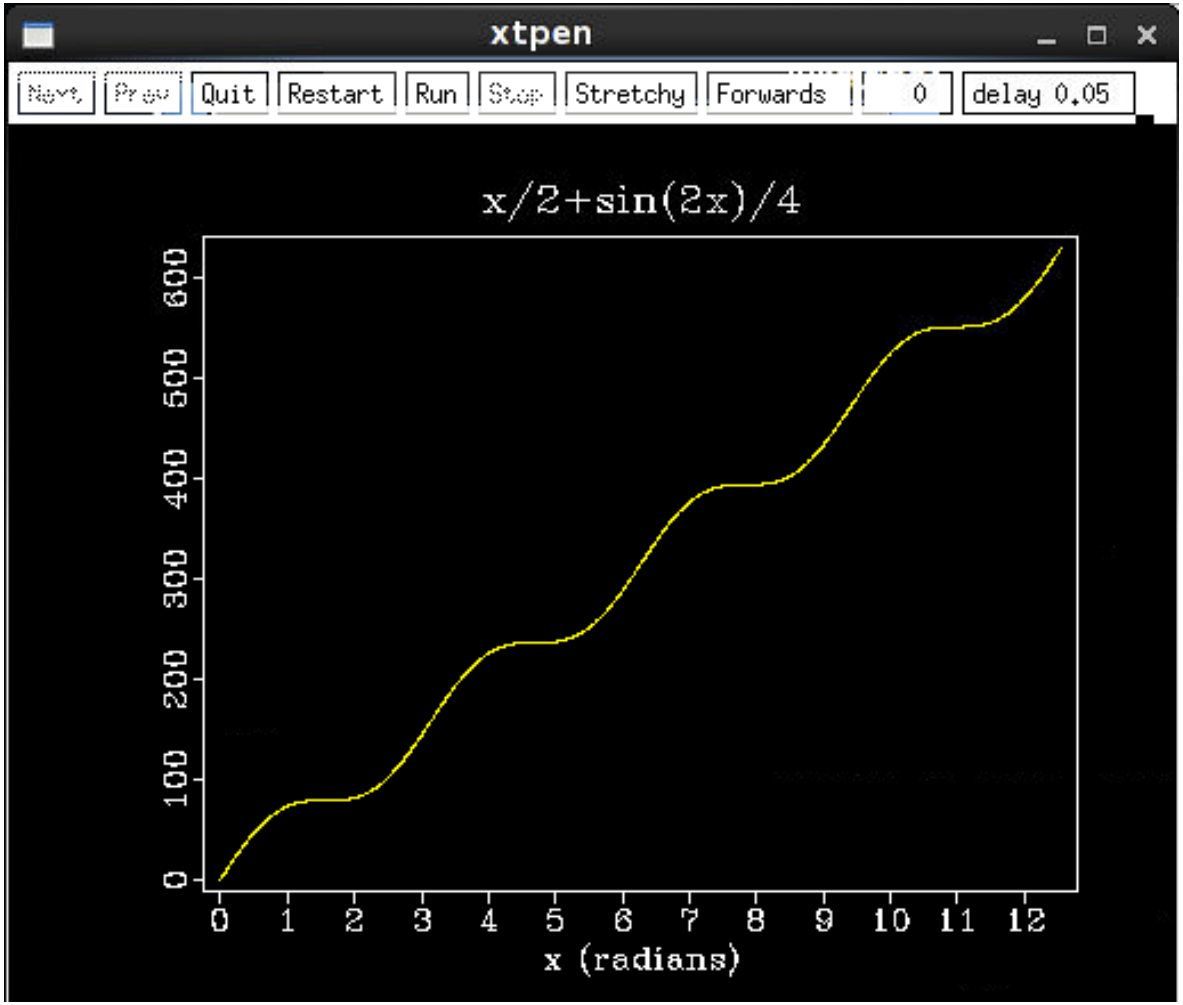
Modify

Integrate data

- See the results of causal integration

  scons int-cos2.view

- What looks odd?

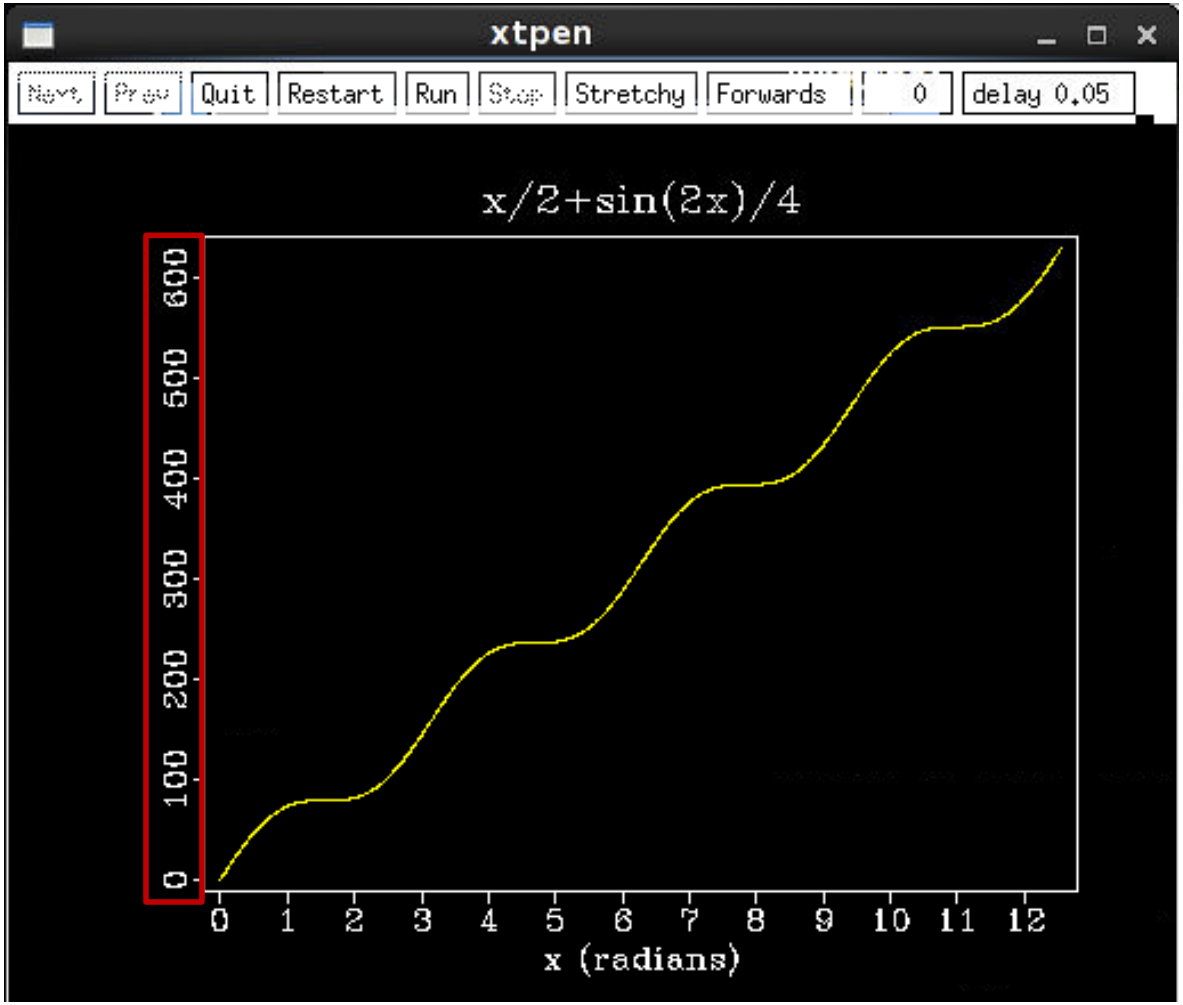


Integrate data

- See the results of causal integration

  scons int-cos2.view

- What looks odd?



Integrate data

- We need to scale the data by dx
- Type

  sfscale
- What does dscale do?

```
# integrate cosine squared, scale by dx
  # modify below
Flow('int-cos2','cos2',‘causint | scale dscale=1')
```

Integrate data

- We need to scale the data by dx
- Type

  sfscale
- What does dscale do?

```
# integrate cosine squared, scale by dx
  # modify below
Flow('int-cos2','cos2',‘causint | scale dscale=1')
```

Modify

Integrate data

- We need to scale the data by dx
- Type

  sfscale
- What does dscale do?

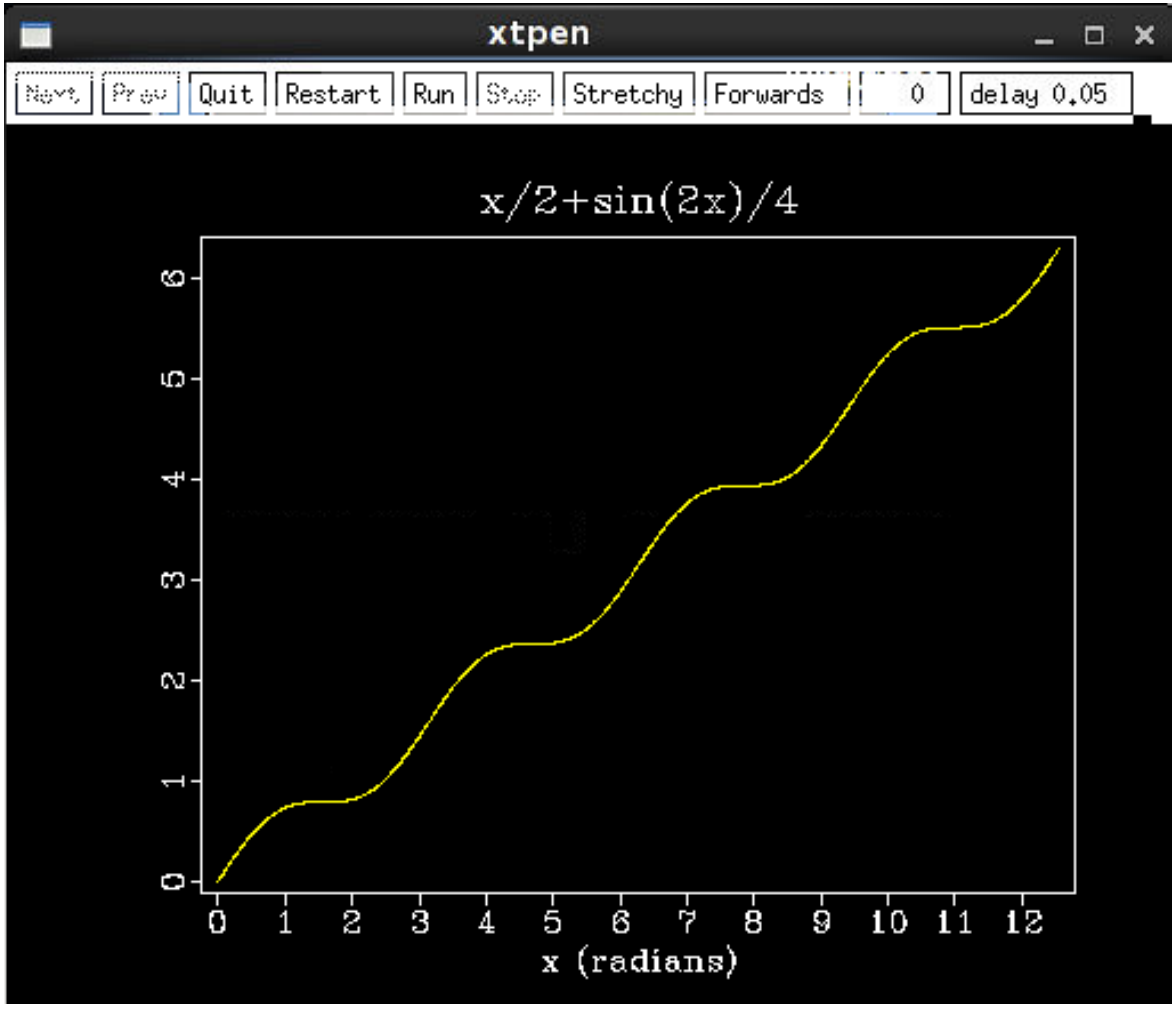
```
# integrate cosine squared, scale by dx
  # modify below
Flow('int-cos2','cos2',‘causint | scale dscale=.01')
```

Modify

Integrate data

- See the results of causal integration

  scons int-cos2.view



Ending SConstruct

```
# integrate cosine squared, scale by dx
  # modify below
Flow('int-cos2','cos2','cp | scale dscale=1')
# plot the integral
Result('int-cos2','graph title="x/2+sin(2x)/4"')
End()
```

CONGRATULATIONS!