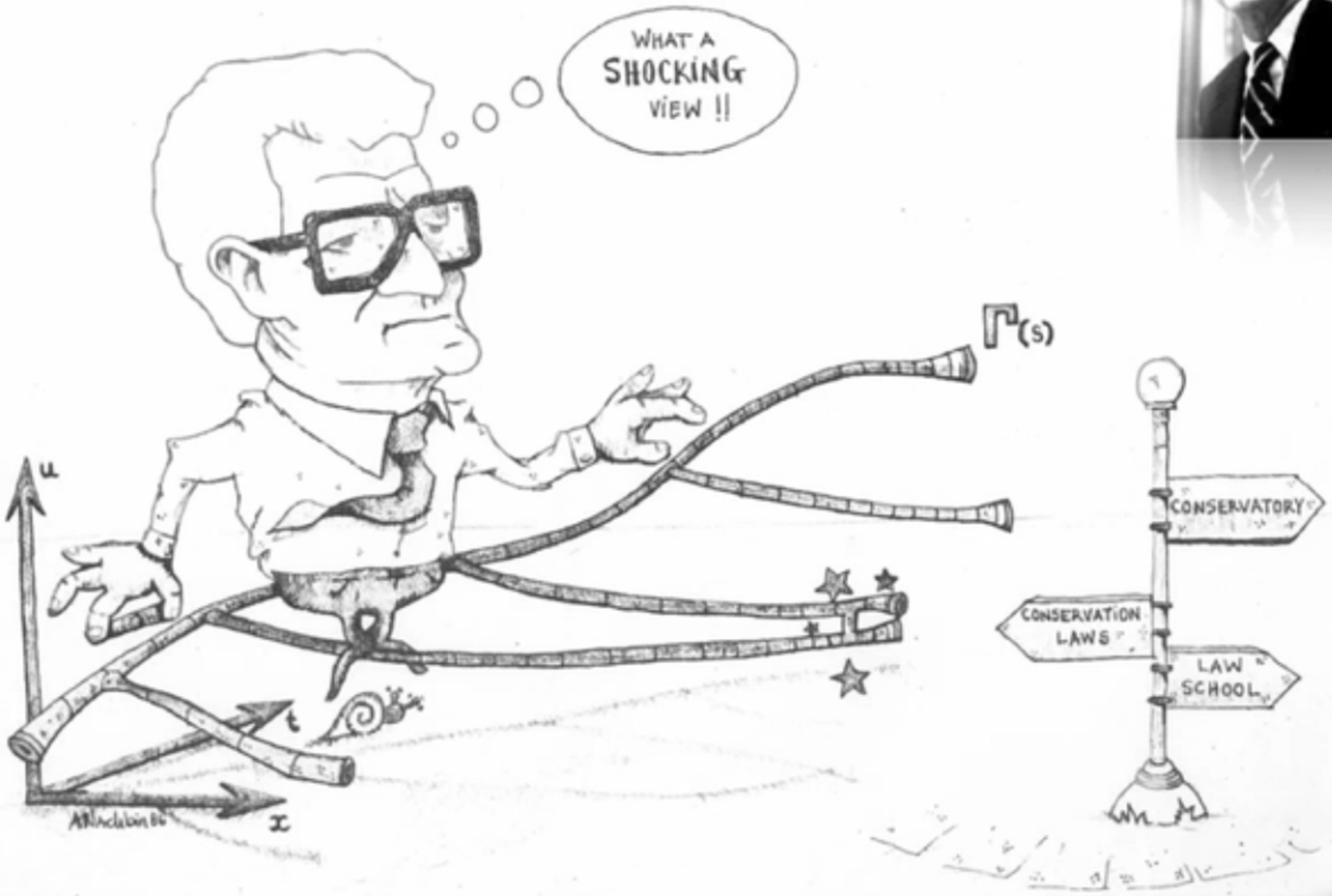
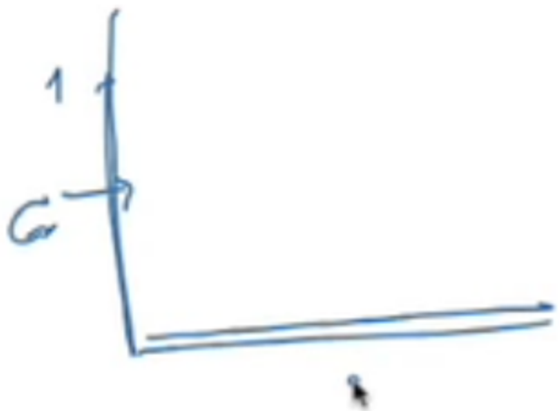
# Peter Lax

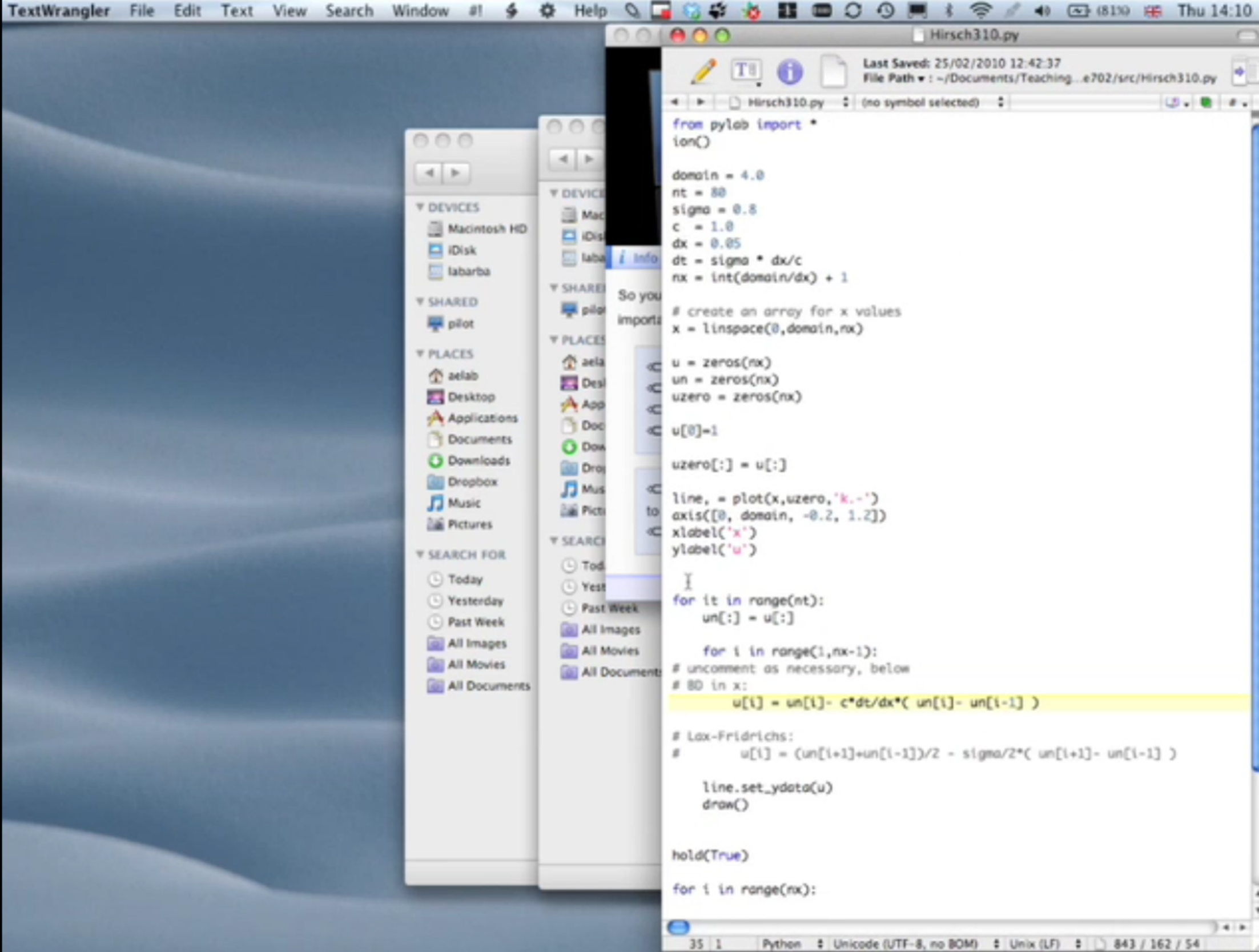
*by Andre Nachbin '86*

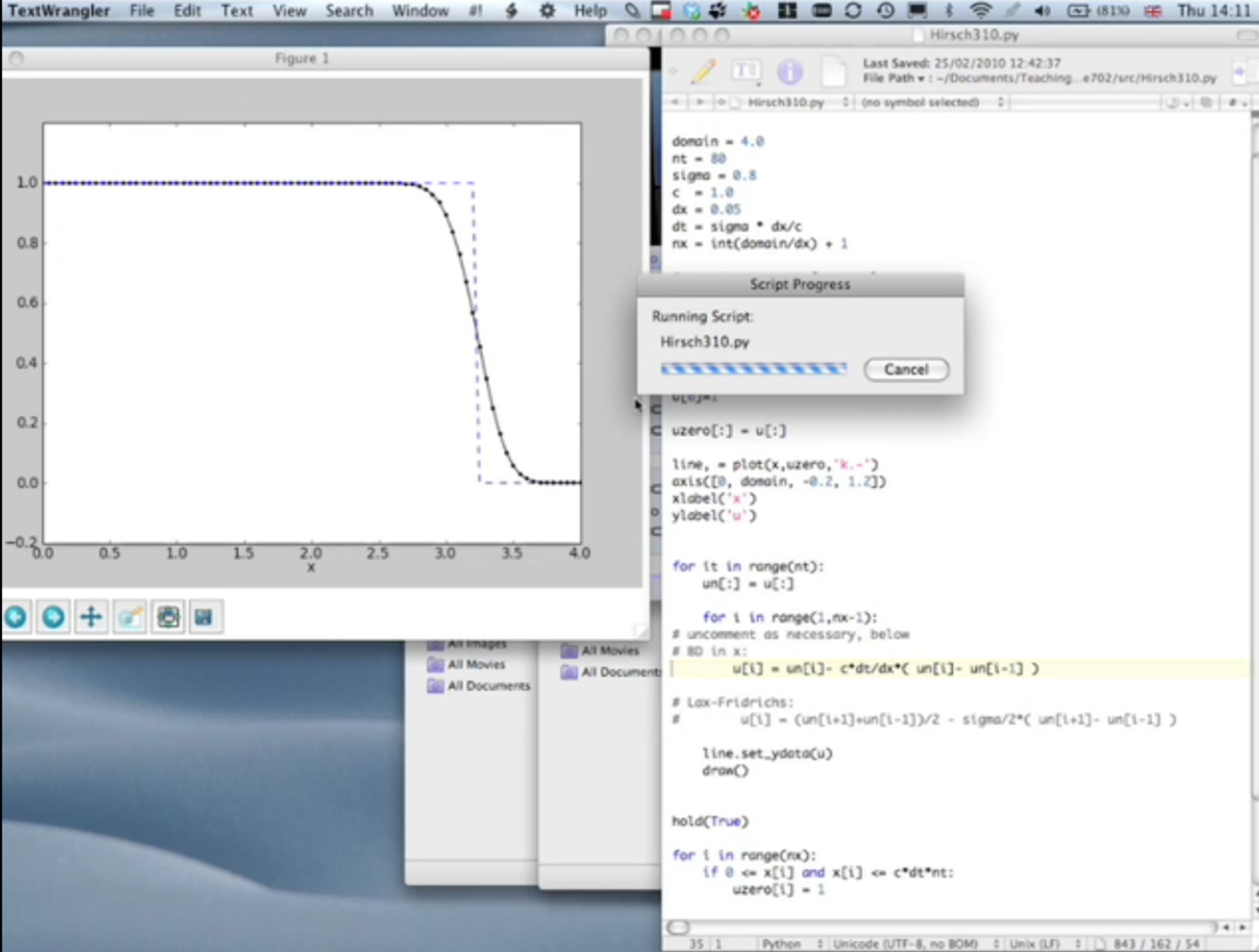


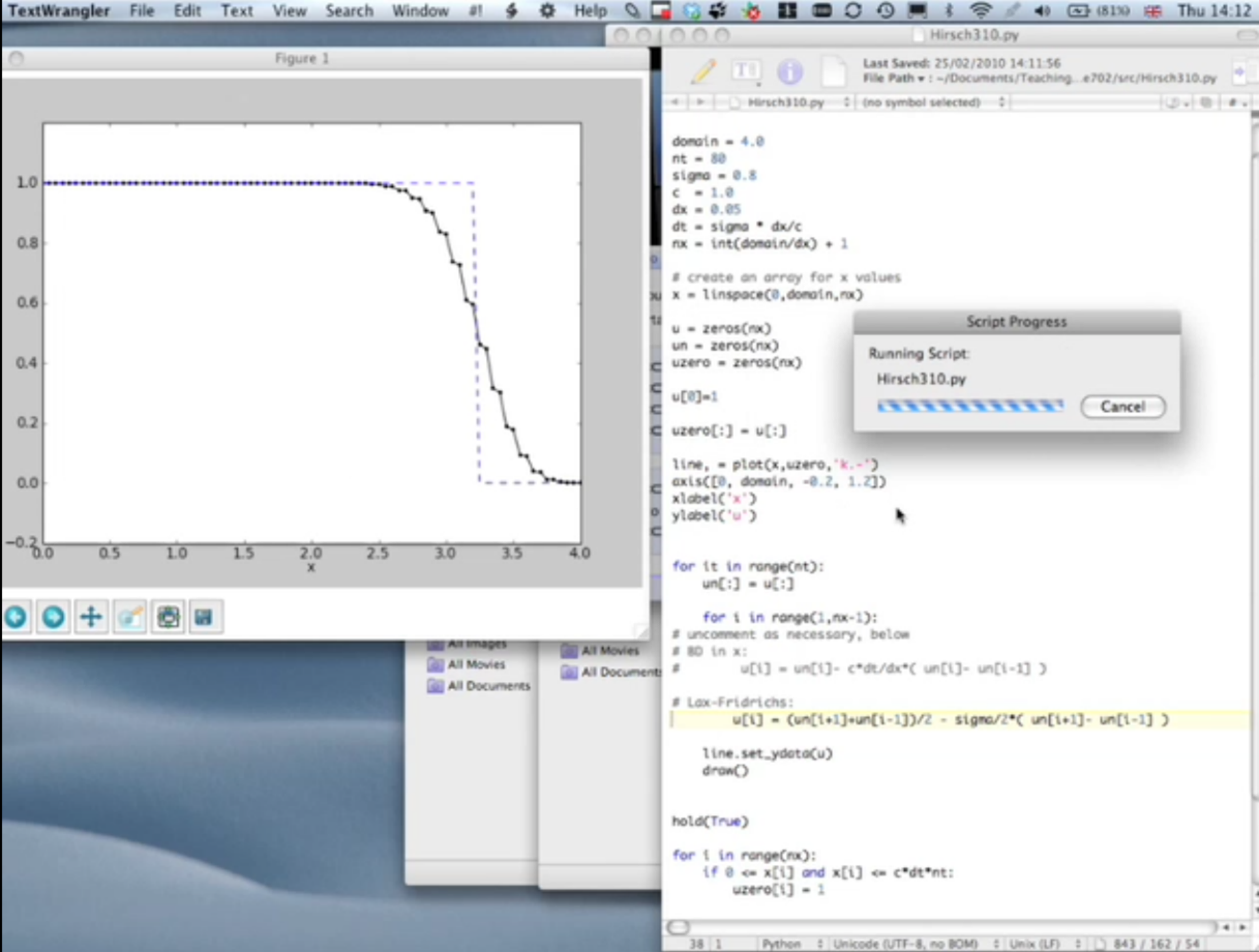
# New schemes for convection (cont.)

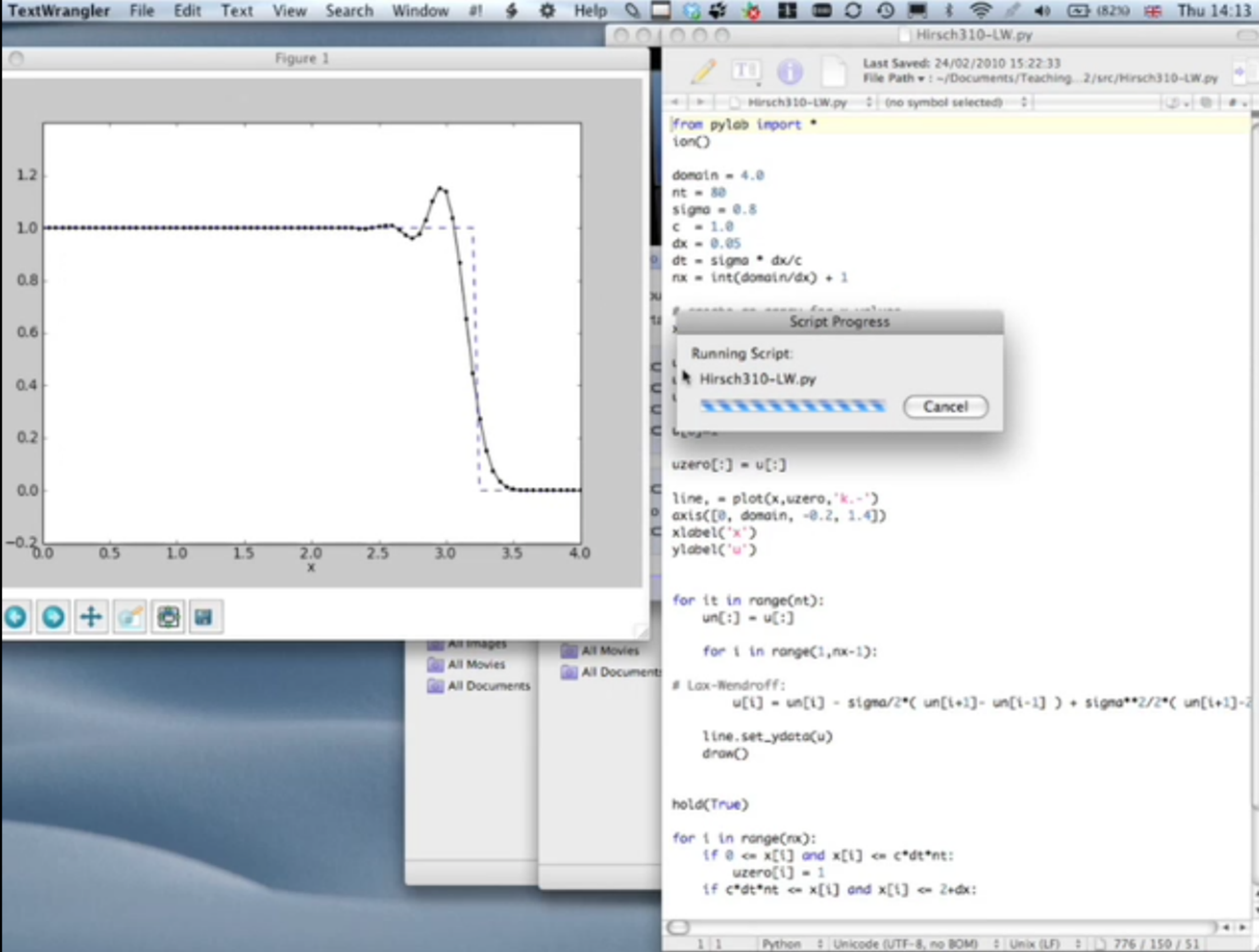
- → Lax - Friedrichs → stabilizes CD, 1st order
- → Lax - Wendroff → stable CD with an additional dissipative term, 2nd order
- → Leapfrog → 2nd order
  3 - level scheme (it is not self-starting)

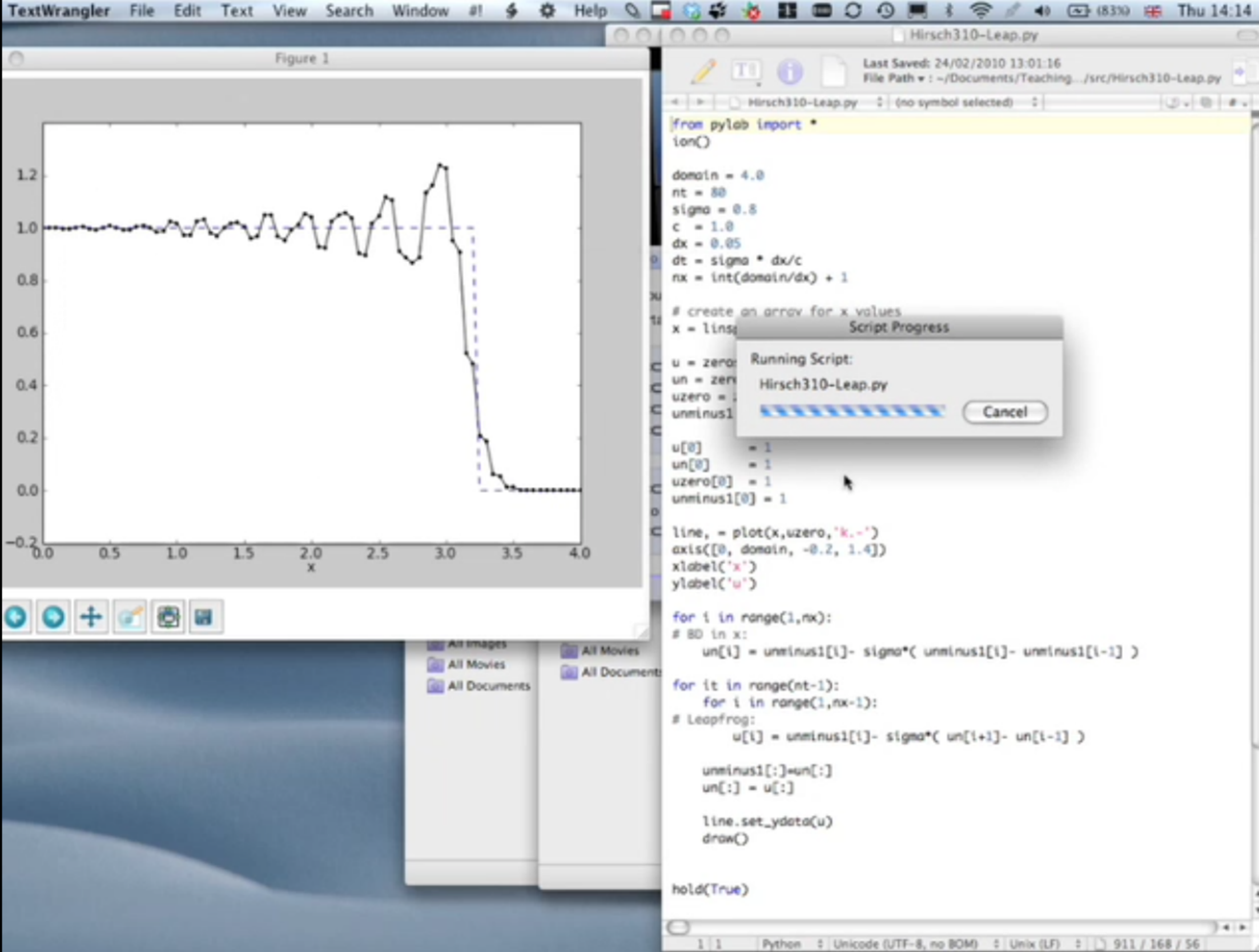



TextWrangler File Edit Text View Search Window #! Help Thu 14:10
Hirsch310.py
Last Saved: 25/02/2010 12:42:37
File Path : ~/Documents/Teaching...e702/src/Hirsch310.py
Hirsch310.py
(no symbol selected)
from pylab import *
ion()
domain = 4.0
nt = 80
sigma = 0.8
c = 1.0
dx = 0.05
dt = sigma * dx/c
nx = int(domain/dx) + 1
# create an array for x values
x = linspace(0,domain,nx)
u = zeros(nx)
un = zeros(nx)
uzero = zeros(nx)
u[0]=1
uzero[:] = u[:]
line, = plot(x,uzero,'k.-')
axis([0, domain, -0.2, 1.2])
xlabel('x')
ylabel('u')
for it in range(nt):
un[:] = u[:]
for i in range(1,nx-1):
# uncomment as necessary, below
# BD in x:
u[i] = un[i]- c*dt/dx*( un[i]- un[i-1] )
# Lax-Fridrichs:
# u[i] = (un[i+1]+un[i-1])/2 - sigma/2*( un[i+1]- un[i-1] )
line.set_ydata(u)
draw()
hold(True)
for i in range(nx):
35 1 Python Unicode (UTF-8, no BOM) Unix (LF) 843 / 162 / 54
DEVICES
Macintosh HD
iDisk
labarba
SHARED
pilot
PLACES
aelab
Desktop
Applications
Documents
Downloads
Dropbox
Music
Pictures
SEARCH FOR
Today
Yesterday
Past Week
All Images
All Movies
All Documents


TextWrangler File Edit Text View Search Window #! Help Thu 14:11
Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
x
Hirsch310.py
Last Saved: 25/02/2010 12:42:37
File Path : ~/Documents/Teaching...e702/src/Hirsch310.py
Hirsch310.py
(no symbol selected)
domain = 4.0
nt = 80
sigma = 0.8
c = 1.0
dx = 0.05
dt = sigma * dx/c
nx = int(domain/dx) + 1
Script Progress
Running Script:
Hirsch310.py
Cancel
uzero[:] = u[:]
line, = plot(x,uzero,'k.-')
axis([0, domain, -0.2, 1.2])
xlabel('x')
ylabel('u')
for it in range(nt):
un[:] = u[:]
for i in range(1,nx-1):
# uncomment as necessary, below
# BD in x:
u[i] = un[i]- c*dt/dx*( un[i]- un[i-1] )
# Lax-Fridrichs:
# u[i] = (un[i+1]+un[i-1])/2 - sigma/2*( un[i+1]- un[i-1] )
line.set_ydata(u)
draw()
hold(True)
for i in range(nx):
if 0 <= x[i] and x[i] <= c*dt*nt:
uzero[i] = 1
All Movies
All Documents
35 1 Python Unicode (UTF-8, no BOM) Unix (LF) 843 / 162 / 54


TextWrangler File Edit Text View Search Window #! Help Thu 14:12
Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
x
Hirsch310.py
Last Saved: 25/02/2010 14:11:56
File Path : ~/Documents/Teaching...e702/src/Hirsch310.py
Hirsch310.py
(no symbol selected)
domain = 4.0
nt = 80
sigma = 0.8
c = 1.0
dx = 0.05
dt = sigma * dx/c
nx = int(domain/dx) + 1
# create an array for x values
x = linspace(0,domain,nx)
u = zeros(nx)
un = zeros(nx)
uzero = zeros(nx)
u[0]=1
uzero[:] = u[:]
line, = plot(x,uzero,'k.-')
axis([0, domain, -0.2, 1.2])
xlabel('x')
ylabel('u')
for it in range(nt):
un[:] = u[:]
for i in range(1,nx-1):
# uncomment as necessary, below
# BD in x:
# u[i] = un[i]- c*dt/dx*( un[i]- un[i-1] )
# Lax-Fridrichs:
u[i] = (un[i+1]+un[i-1])/2 - sigma/2*( un[i+1]- un[i-1] )
line.set_ydata(u)
draw()
hold(True)
for i in range(nx):
if 0 <= x[i] and x[i] <= c*dt*nt:
uzero[i] = 1
Script Progress
Running Script:
Hirsch310.py
Cancel
All Images
All Movies
All Documents
All Movies
All Documents
38 1 Python Unicode (UTF-8, no BOM) Unix (LF) 843 / 162 / 54

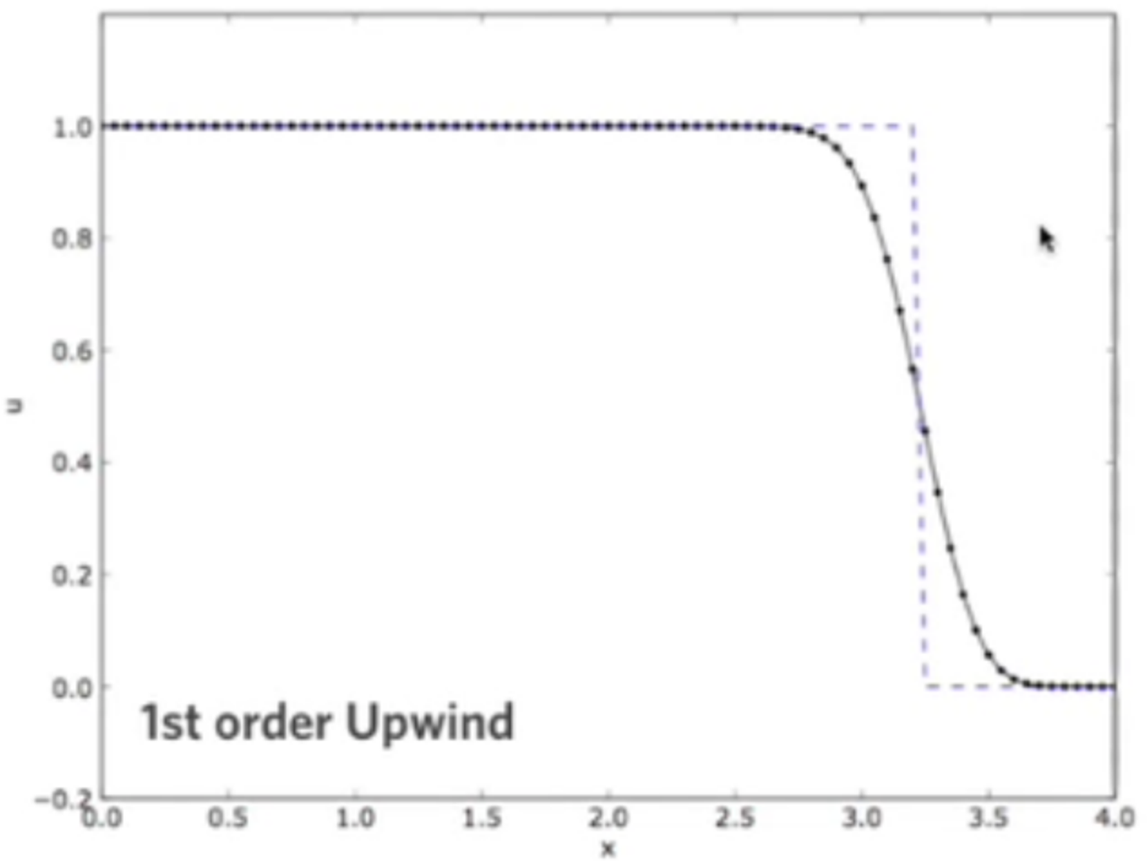

TextWrangler File Edit Text View Search Window #! Help (82%) Thu 14:13
Figure 1
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
x
Hirsch310-LW.py
Last Saved: 24/02/2010 15:22:33
File Path : ~/Documents/Teaching...2/src/Hirsch310-LW.py
Hirsch310-LW.py
(no symbol selected)
from pylab import *
ion()
domain = 4.0
nt = 80
sigma = 0.8
c = 1.0
dx = 0.05
dt = sigma * dx/c
nx = int(domain/dx) + 1
Script Progress
Running Script:
Hirsch310-LW.py
Cancel
uzero[:] = u[:]
line, = plot(x,uzero,'k.-')
axis([0, domain, -0.2, 1.4])
xlabel('x')
ylabel('u')
for it in range(nt):
un[:] = u[:]
for i in range(1,nx-1):
# Lax-Wendroff:
u[i] = un[i] - sigma/2*( un[i+1]- un[i-1] ) + sigma**2/2*( un[i+1]-2
line.set_ydata(u)
draw()
hold(True)
for i in range(nx):
if 0 <= x[i] and x[i] <= c*dt*nt:
uzero[i] = 1
if c*dt*nt <= x[i] and x[i] <= 2+dx:
All Images
All Movies
All Documents
All Movies
All Documents
1 1 Python Unicode (UTF-8, no BOM) Unix (LF) 776 / 150 / 51


TextWrangler File Edit Text View Search Window #! Help Thu 14:14
Figure 1
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
x
Hirsch310-Leap.py
Last Saved: 24/02/2010 13:01:16
File Path : ~/Documents/Teaching.../src/Hirsch310-Leap.py
Hirsch310-Leap.py
(no symbol selected)
from pylab import *
ion()
domain = 4.0
nt = 80
sigma = 0.8
c = 1.0
dx = 0.05
dt = sigma * dx/c
nx = int(domain/dx) + 1
# create an array for x values
x = linsp
u = zero
un = zer
uzero =
unminus1
u[0] = 1
un[0] = 1
uzero[0] = 1
unminus1[0] = 1
line, = plot(x,uzero,'k.-')
axis([0, domain, -0.2, 1.4])
xlabel('x')
ylabel('u')
for i in range(1,nx):
# BD in x:
un[i] = unminus1[i]- sigma*( unminus1[i]- unminus1[i-1] )
for it in range(nt-1):
for i in range(1,nx-1):
# Leapfrog:
u[i] = unminus1[i]- sigma*( un[i+1]- un[i-1] )
unminus1[:]=un[:]
un[:] = u[:]
line.set_ydata(u)
draw()
hold(True)
Script Progress
Running Script:
Hirsch310-Leap.py
Cancel
All Images
All Movies
All Documents
All Movies
All Documents
1 1 Python Unicode (UTF-8, no BOM) Unix (LF) 911 / 168 / 56

# New schemes for convection (cont.)

*Test 1* : Heaviside fnc. wave

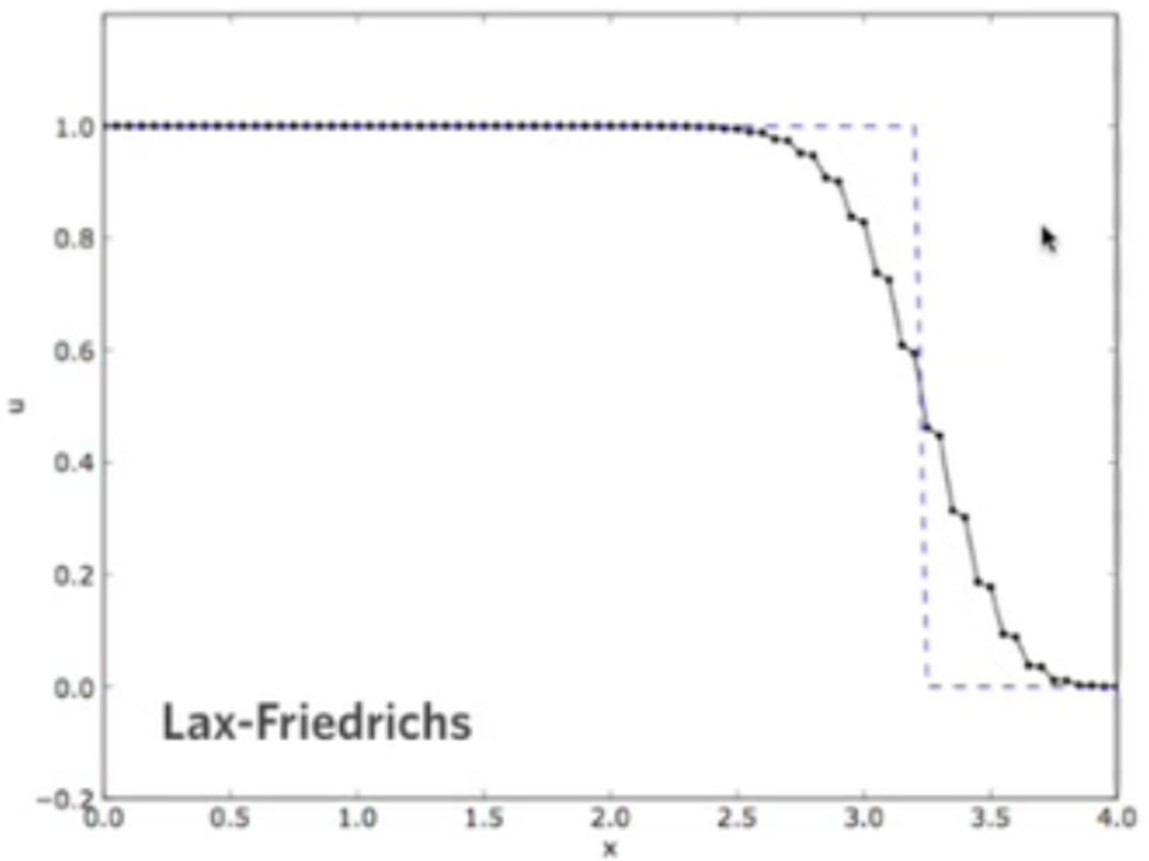
$$\begin{aligned} \sigma &= 0.8 \\ \Delta x &= 0.05 \\ n_{\text{steps}} &= 80 \end{aligned}$$



# New schemes for convection (cont.)

*Test 1* : Heaviside fnc. wave

$$\begin{aligned} \sigma &= 0.8 \\ \Delta x &= 0.05 \\ n_{\text{steps}} &= 80 \end{aligned}$$

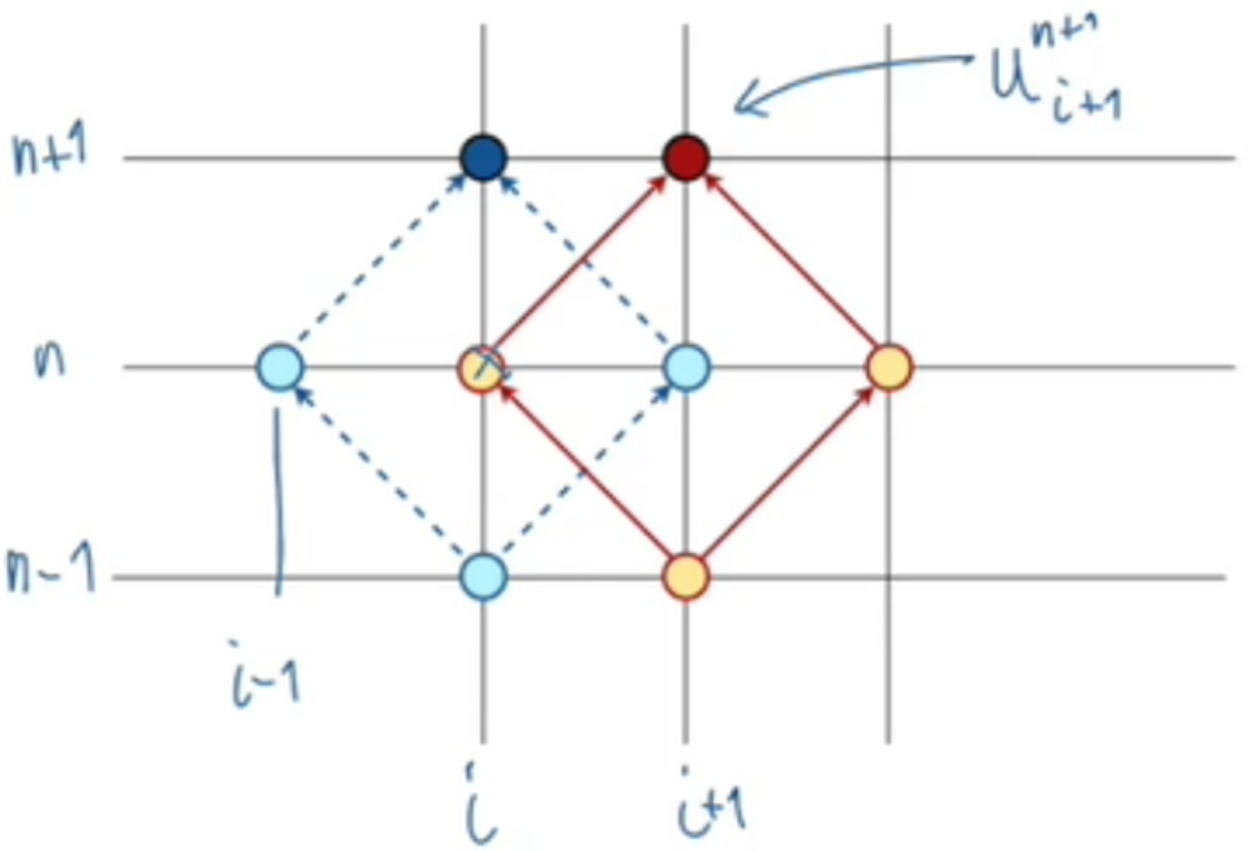


- Recall the stencil for Leapfrog:

= double solution = effect

$u_i^{n+1}$ does not depend on $u_i^n$

* Shift the stencil by $\Delta x$



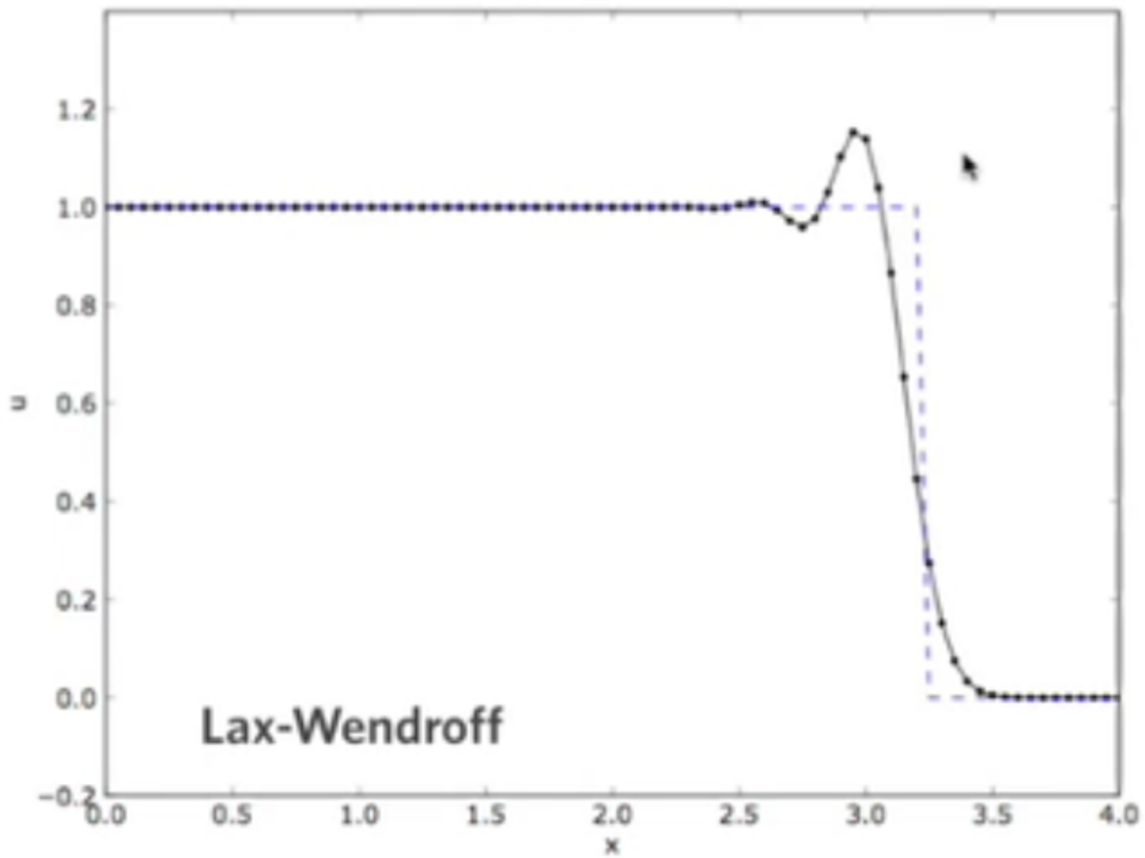

$u_i^{n+1}$ & $u_{i+1}^{n+1}$ do not share a single mesh point of their stencils

Called :

"odd-even decoupling"

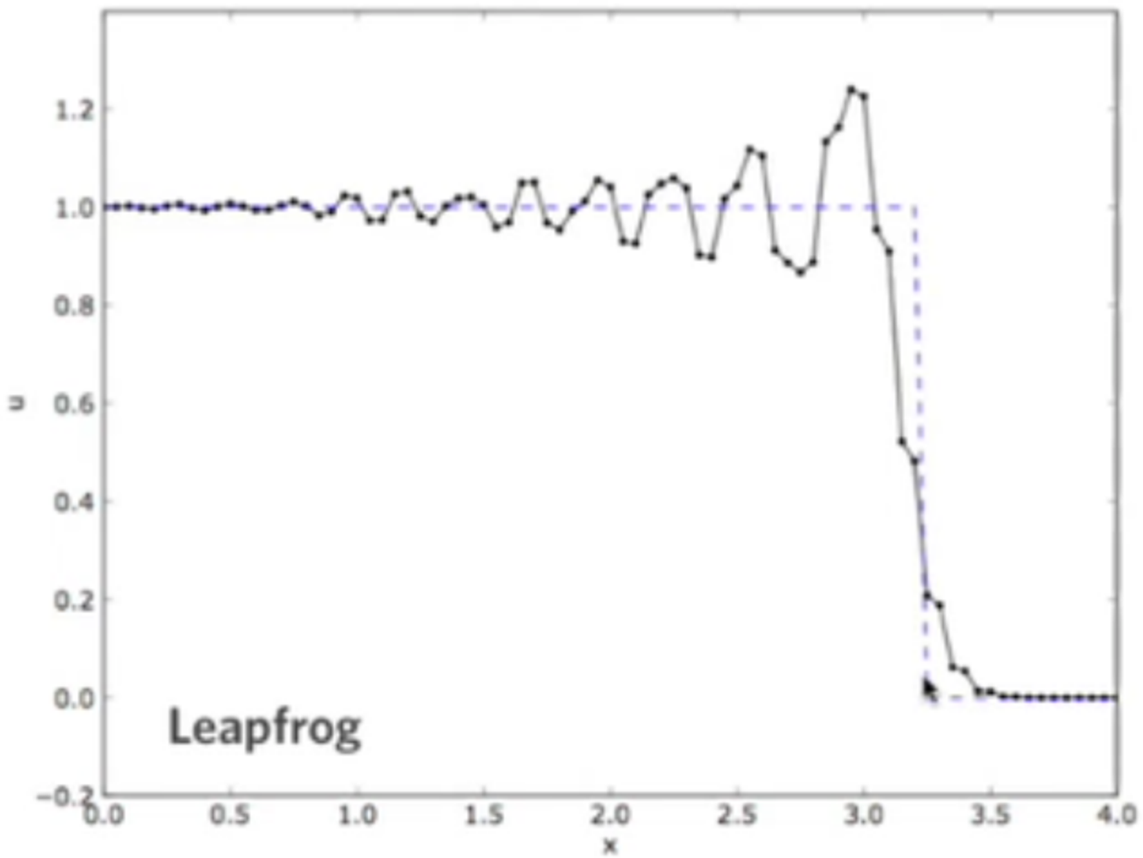
# *Test 1* : Heaviside fnc. wave

$$\begin{aligned} \sigma &= 0.8 \\ \Delta x &= 0.05 \\ n_{\text{steps}} &= 80 \end{aligned}$$



# *Test 1* : Heaviside fnc. wave

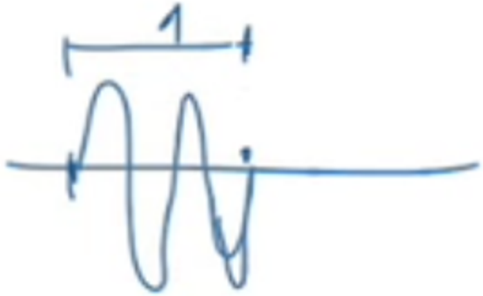
$$\begin{aligned} \sigma &= 0.8 \\ \Delta x &= 0.05 \\ n_{\text{steps}} &= 80 \end{aligned}$$

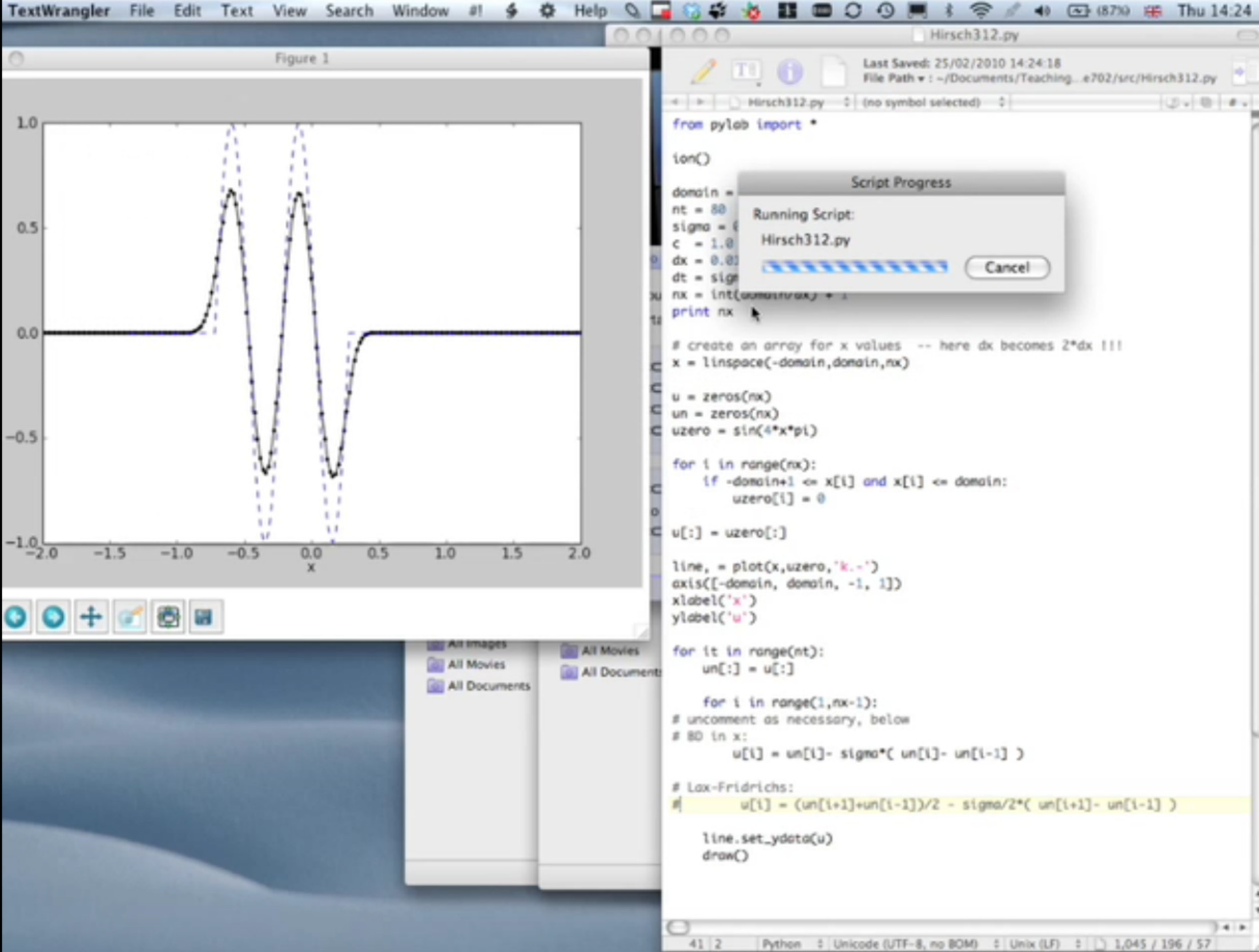


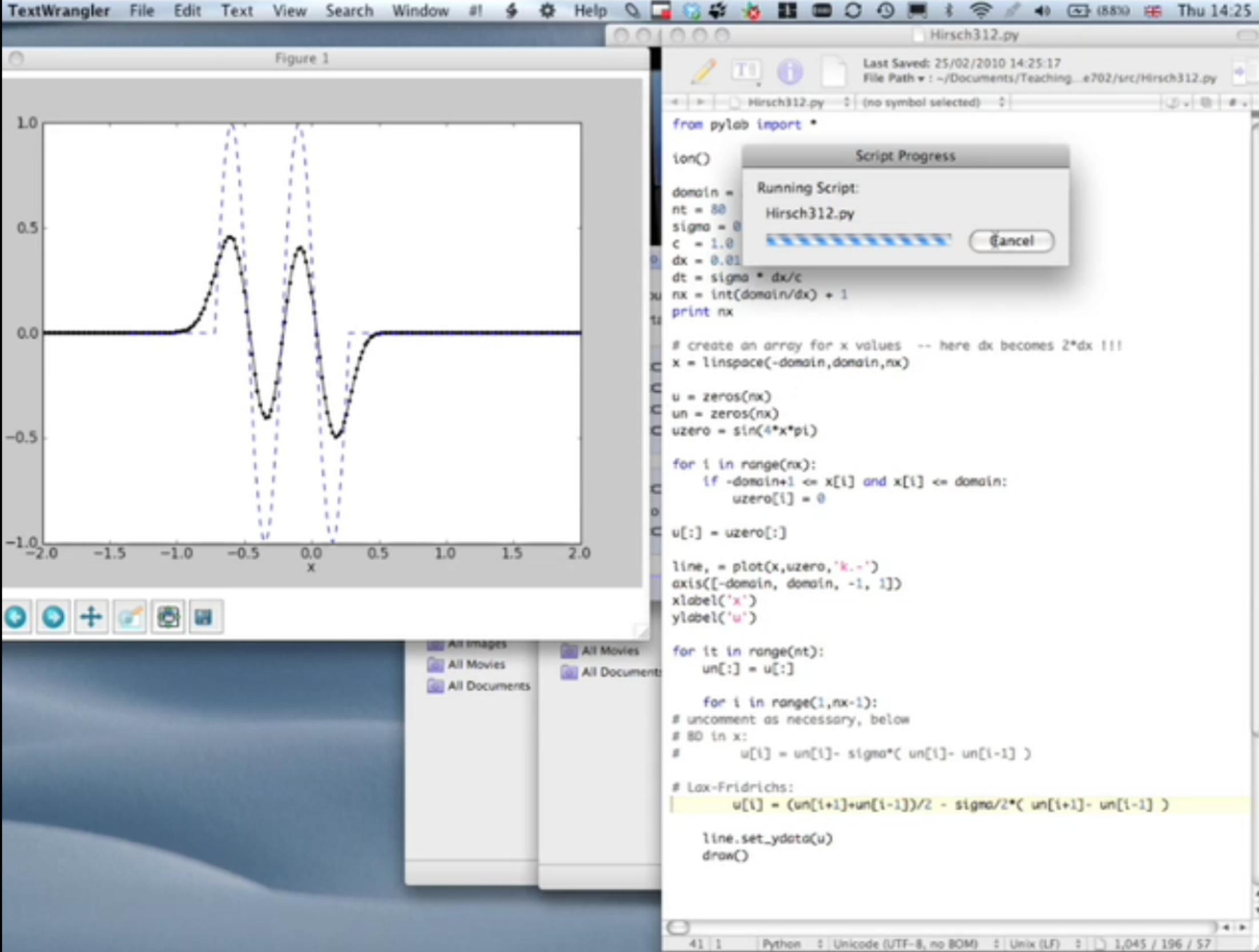
# *Test 2* : sinusoidal wave packet

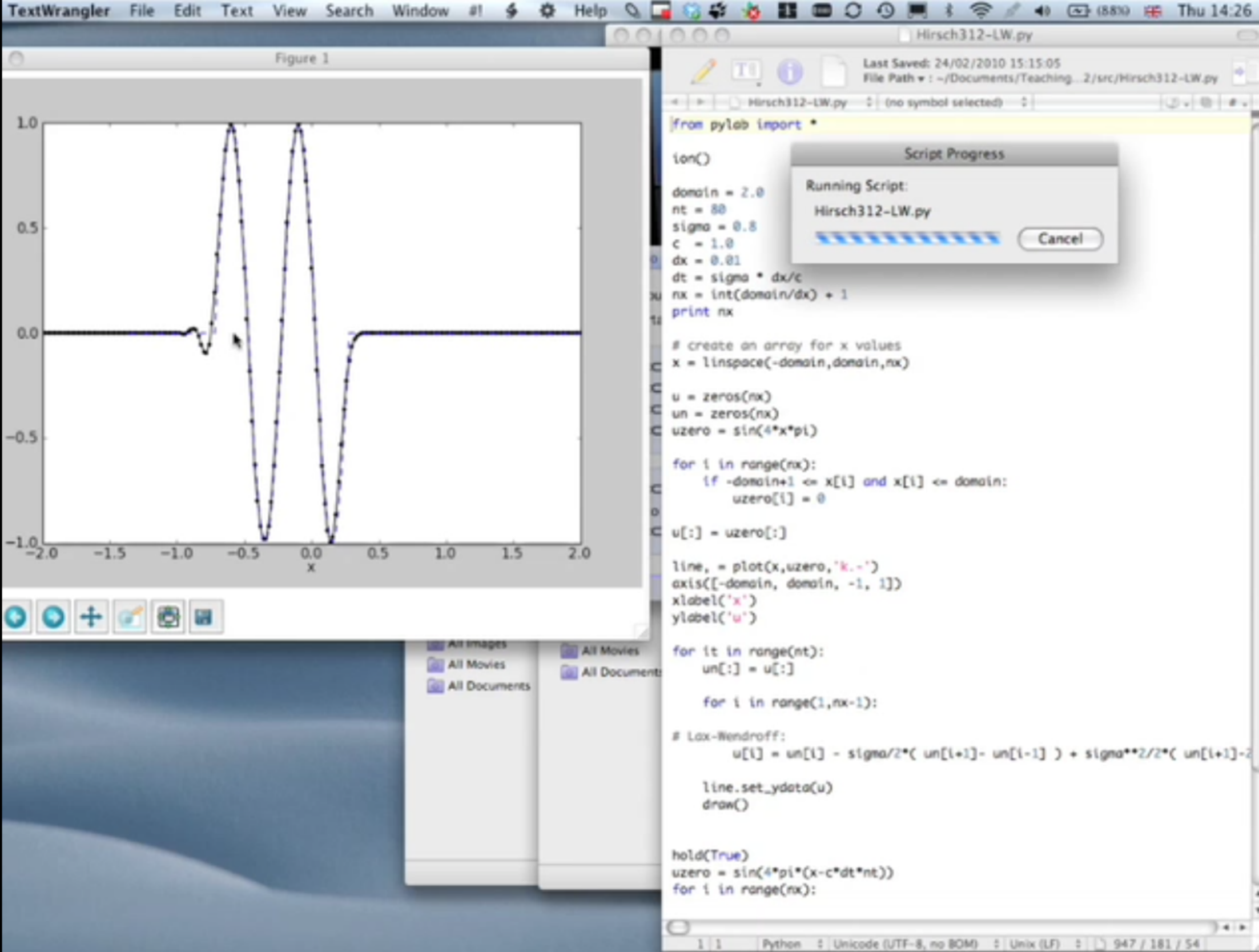
2 periods in a distance = 1

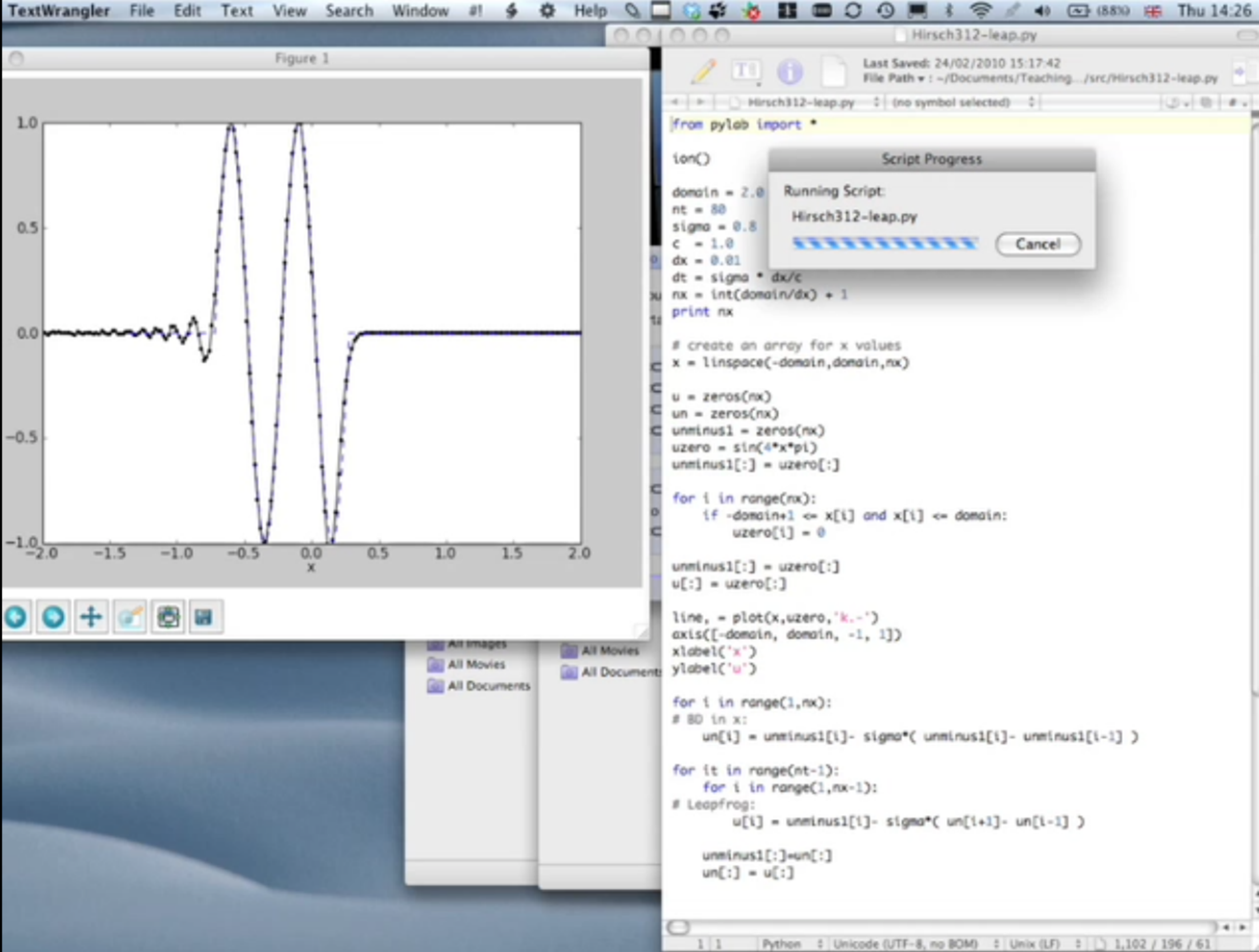
corresponding wave number $k = \frac{2\pi}{\lambda} = 4\pi$




TextWrangler File Edit Text View Search Window #! Help (87%) Thu 14:24
Figure 1
1.0
0.5
0.0
−0.5
−1.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x
Hirsch312.py
Last Saved: 25/02/2010 14:24:18
File Path : ~/Documents/Teaching...e702/src/Hirsch312.py
Hirsch312.py
(no symbol selected)
from pylab import *
ion()
domain =
nt = 80
sigma =
c = 1.0
dx = 0.0
dt = sig
nx = int(
print nx
Script Progress
Running Script:
Hirsch312.py
Cancel
# create an array for x values -- here dx becomes 2*dx !!!
x = linspace(-domain,domain,nx)
u = zeros(nx)
un = zeros(nx)
uzero = sin(4*x*pi)
for i in range(nx):
if -domain+1 <= x[i] and x[i] <= domain:
uzero[i] = 0
u[:] = uzero[:]
line, = plot(x,uzero,'k.-')
axis([-domain, domain, -1, 1])
xlabel('x')
ylabel('u')
for it in range(nt):
un[:] = u[:]
for i in range(1,nx-1):
# uncomment as necessary, below
# BD in x:
u[i] = un[i]- sigma*( un[i]- un[i-1] )
# Lax-Fridrichs:
# u[i] = (un[i+1]+un[i-1])/2 - sigma/2*( un[i+1]- un[i-1] )
line.set_ydata(u)
draw()
All Images
All Movies
All Documents
All Movies
All Documents
41 2 Python Unicode (UTF-8, no BOM) Unix (LF) 1,045 / 196 / 57

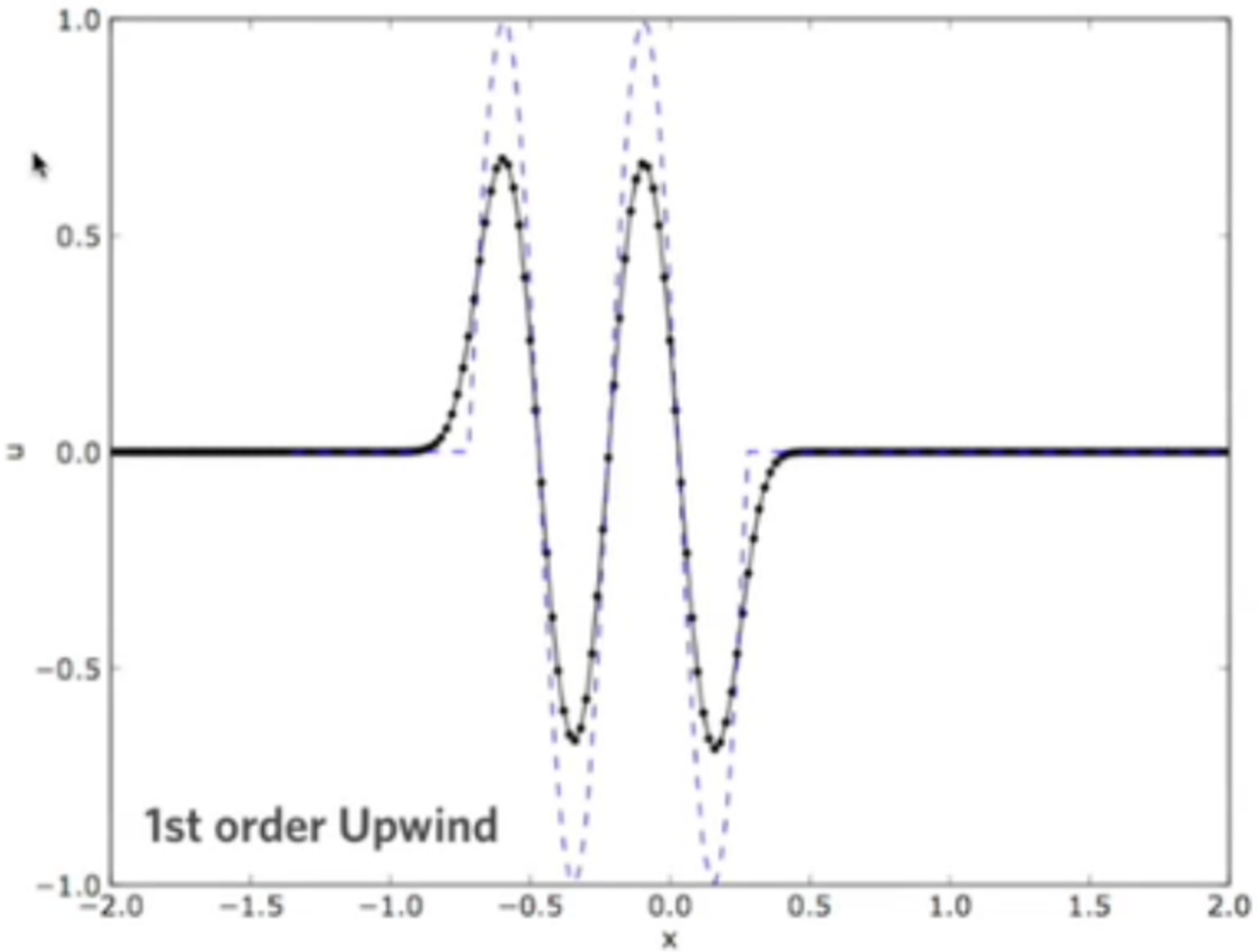

TextWrangler File Edit Text View Search Window #! Help (88%) Thu 14:25
Figure 1
1.0
0.5
0.0
−0.5
−1.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x
Hirsch312.py
Last Saved: 25/02/2010 14:25:17
File Path : ~/Documents/Teaching...e702/src/Hirsch312.py
Hirsch312.py
(no symbol selected)
from pylab import *
ion()
Script Progress
Running Script:
Hirsch312.py
Cancel
domain =
nt = 80
sigma = 0
c = 1.0
dx = 0.01
dt = sigma * dx/c
nx = int(domain/dx) + 1
print nx
# create an array for x values -- here dx becomes 2*dx !!!
x = linspace(-domain,domain,nx)
u = zeros(nx)
un = zeros(nx)
uzero = sin(4*x*pi)
for i in range(nx):
if -domain+1 <= x[i] and x[i] <= domain:
uzero[i] = 0
u[:] = uzero[:]
line, = plot(x,uzero,'k.-')
axis([-domain, domain, -1, 1])
xlabel('x')
ylabel('u')
for it in range(nt):
un[:] = u[:]
for i in range(1,nx-1):
# uncomment as necessary, below
# BD in x:
# u[i] = un[i]- sigma*( un[i]- un[i-1] )
# Lax-Fridrichs:
u[i] = (un[i+1]+un[i-1])/2 - sigma/2*( un[i+1]- un[i-1] )
line.set_ydata(u)
draw()
All Movies
All Documents
41 1 Python Unicode (UTF-8, no BOM) Unix (LF) 1,045 / 196 / 57


TextWrangler File Edit Text View Search Window #! Help (88%) Thu 14:26
Figure 1
1.0
0.5
0.0
−0.5
−1.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x
Hirsch312-LW.py
Last Saved: 24/02/2010 15:15:05
File Path : ~/Documents/Teaching...2/src/Hirsch312-LW.py
Hirsch312-LW.py
(no symbol selected)
from pylab import *
ion()
domain = 2.0
nt = 80
sigma = 0.8
c = 1.0
dx = 0.01
dt = sigma * dx/c
nx = int(domain/dx) + 1
print nx
# create an array for x values
x = linspace(-domain,domain,nx)
u = zeros(nx)
un = zeros(nx)
uzero = sin(4*x*pi)
for i in range(nx):
if -domain+1 <= x[i] and x[i] <= domain:
uzero[i] = 0
u[:] = uzero[:]
line, = plot(x,uzero,'k.-')
axis([-domain, domain, -1, 1])
xlabel('x')
ylabel('u')
for it in range(nt):
un[:] = u[:]
for i in range(1,nx-1):
# Lax-Wendroff:
u[i] = un[i] - sigma/2*( un[i+1]- un[i-1] ) + sigma**2/2*( un[i+1]-2
line.set_ydata(u)
draw()
hold(True)
uzero = sin(4*pi*(x-c*dt*nt))
for i in range(nx):
Script Progress
Running Script:
Hirsch312-LW.py
Cancel
All Images
All Movies
All Documents
All Movies
All Documents
1 1 Python Unicode (UTF-8, no BOM) Unix (LF) 947 / 181 / 54


TextWrangler File Edit Text View Search Window #! Help (88%) Thu 14:26
Figure 1
1.0
0.5
0.0
−0.5
−1.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x
Hirsch312-leap.py
Last Saved: 24/02/2010 15:17:42
File Path : ~/Documents/Teaching.../src/Hirsch312-leap.py
Hirsch312-leap.py
(no symbol selected)
from pylab import *
ion()
Script Progress
Running Script:
Hirsch312-leap.py
Cancel
domain = 2.0
nt = 80
sigma = 0.8
c = 1.0
dx = 0.01
dt = sigma * dx/c
nx = int(domain/dx) + 1
print nx
# create an array for x values
x = linspace(-domain,domain,nx)
u = zeros(nx)
un = zeros(nx)
unminus1 = zeros(nx)
uzero = sin(4*x*pi)
unminus1[:] = uzero[:]
for i in range(nx):
if -domain+1 <= x[i] and x[i] <= domain:
uzero[i] = 0
unminus1[:] = uzero[:]
u[:] = uzero[:]
line, = plot(x,uzero,'k.-')
axis([-domain, domain, -1, 1])
xlabel('x')
ylabel('u')
for i in range(1,nx):
# BD in x:
un[i] = unminus1[i]- sigma*( unminus1[i]- unminus1[i-1] )
for it in range(nt-1):
for i in range(1,nx-1):
# Leapfrog:
u[i] = unminus1[i]- sigma*( un[i+1]- un[i-1] )
unminus1[:]=un[:]
un[:] = u[:]
All Images
All Movies
All Documents
All Movies
All Documents
1 1 Python Unicode (UTF-8, no BOM) Unix (LF) 1,102 / 196 / 61

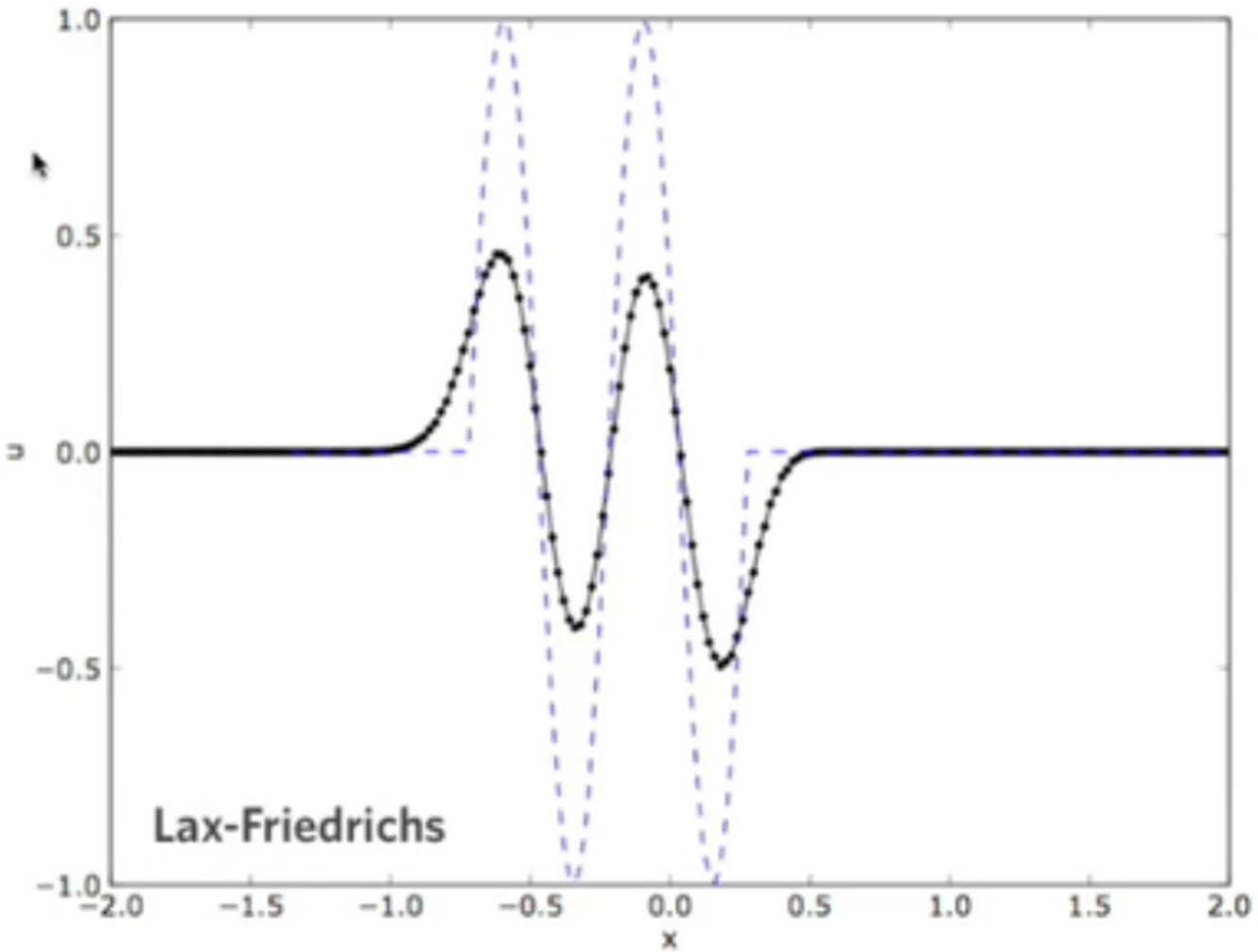
# *Test 2* : sinusoidal wave packet

$$\sigma = 0.8$$
$$\Delta x = 0.02$$
$$n_{\text{steps}} = 80$$



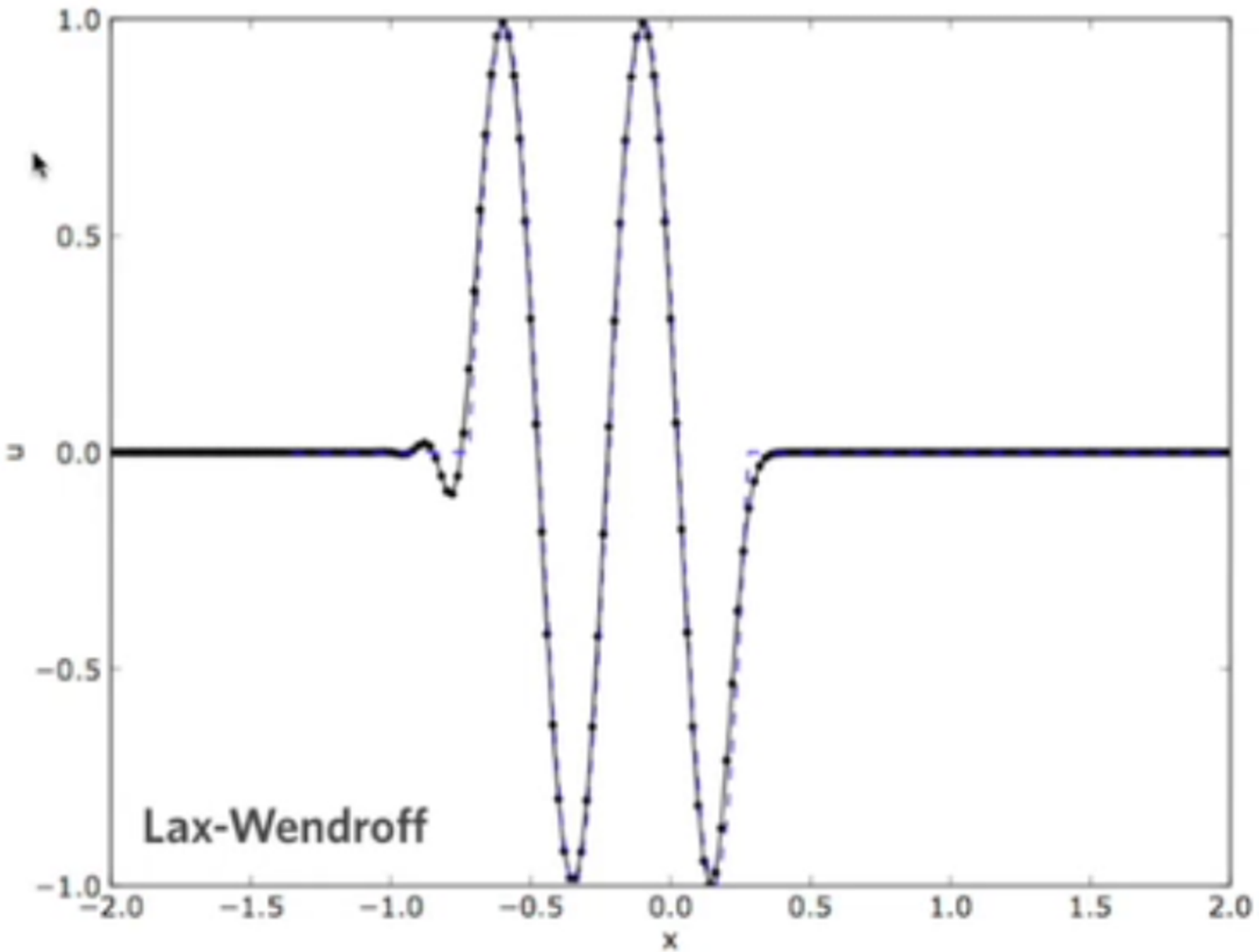
# *Test 2* : sinusoidal wave packet

$$\sigma = 0.8$$
$$\Delta x = 0.02$$
$$n_{\text{steps}} = 80$$



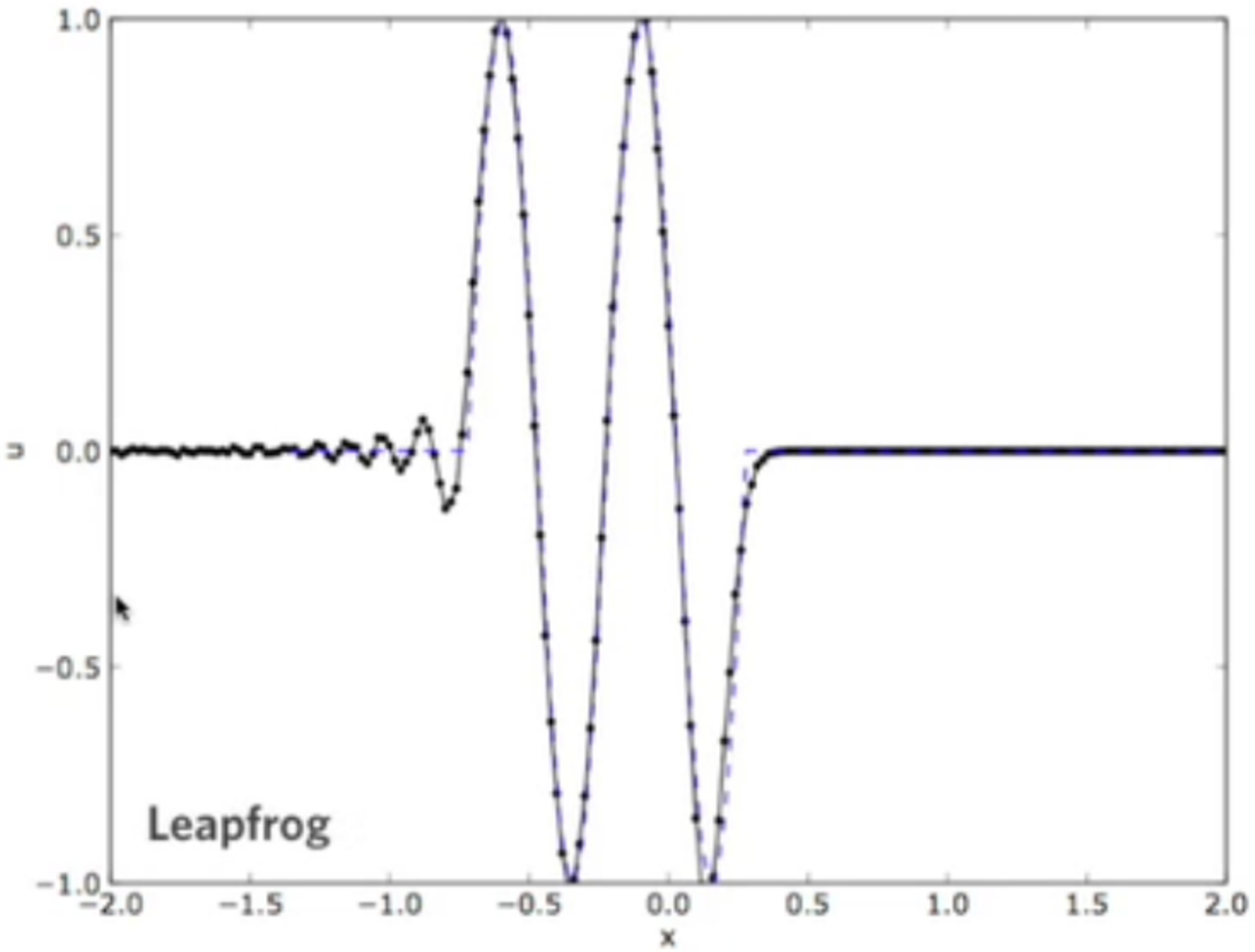
# *Test 2* : sinusoidal wave packet

$$\sigma = 0.8$$
$$\Delta x = 0.02$$
$$n_{\text{steps}} = 80$$



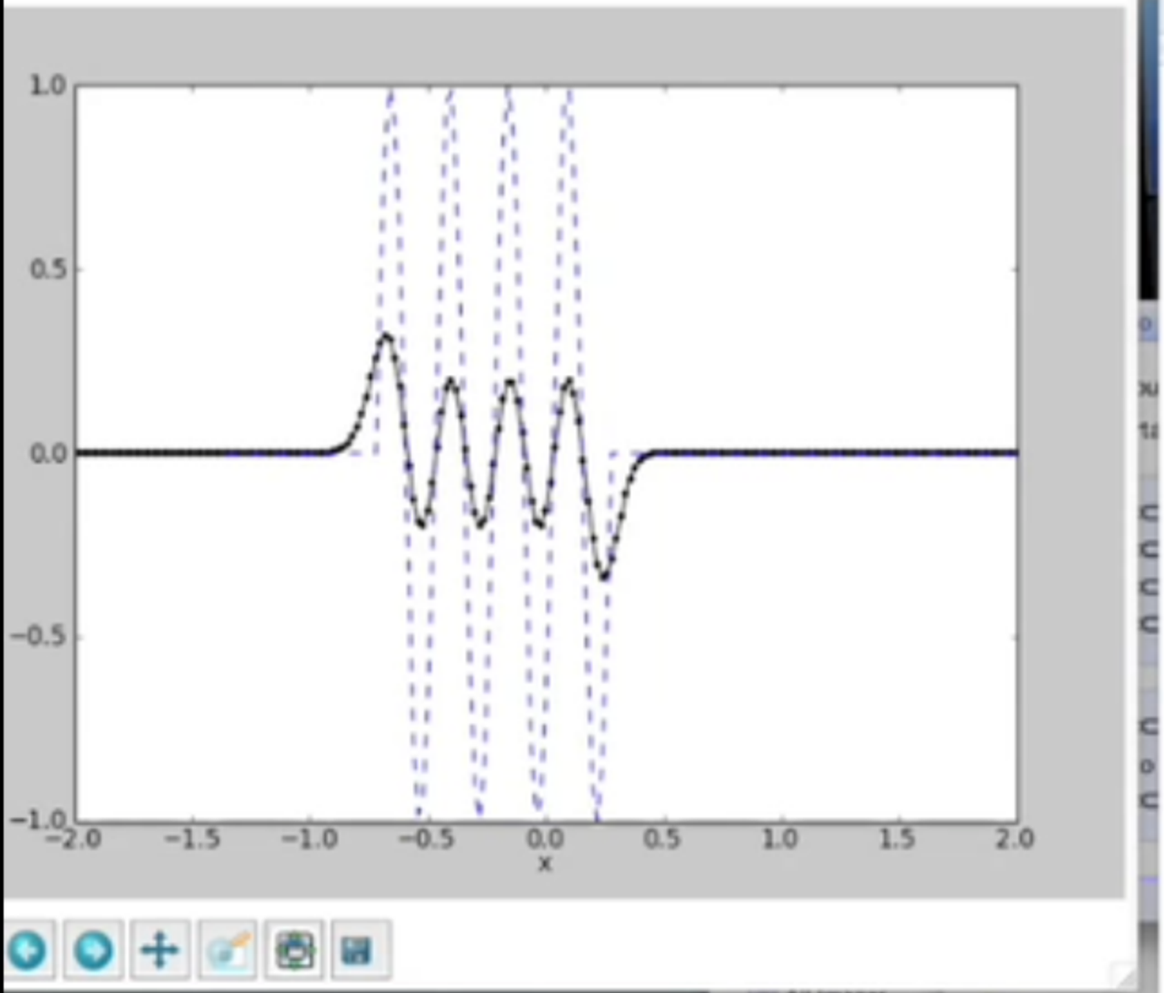
# *Test 2* : sinusoidal wave packet

$$\sigma = 0.8$$
$$\Delta x = 0.02$$
$$n_{\text{steps}} = 80$$



TextWrangler File Edit Text View Search Window #! Help (89%) Thu 14:28

Figure 1




Hirsch313.py

Last Saved: 24/02/2010 15:55:56
File Path ▾ : ~/Documents/Teaching...e702/src/Hirsch313.py

Hirsch313.py (no symbol selected)

```
nt = 80
sigma = 0.8
c  = 1.0
dx = 0.01
dt = sigma * dx/c
nx = int(domain/dx) + 1
print nx

# create an array for x values  -- here dx becomes 2*dx !!!
x = linspace(-domain,domain,nx)

u = zeros(nx)
un = zeros(nx)
uzero = sin(8*x*pi)

for i in range(nx):
    if -domain+1 <= x[i] and x[i] <= domain:
        uzero[i] = 0

u[:] = uzero[:]

line, = plot(x,uzero,'k.-')
axis([-domain, domain, -1, 1])
xlabel('x')
ylabel('u')

for it in range(nt):
    un[:] = u[:]

    for i in range(1,nx-1):
# uncomment as necessary, below
# BD in x:
        u[i] = un[i]- sigma*( un[i]- un[i-1] )

# Lax-Fridrichs:
#        u[i] = (un[i+1]+un[i-1])/2 - sigma/2*( un[i+1]- un[i-1] )

    line.set_ydata(u)
    draw()


hold(True)
uzero = sin(8*pi*(x-c*dt*nt))
for i in range(nx):
    if -domain <= x[i] and x[i] <= -domain+c*dt*nt:
        uzero[i] = 0
```

Script Progress

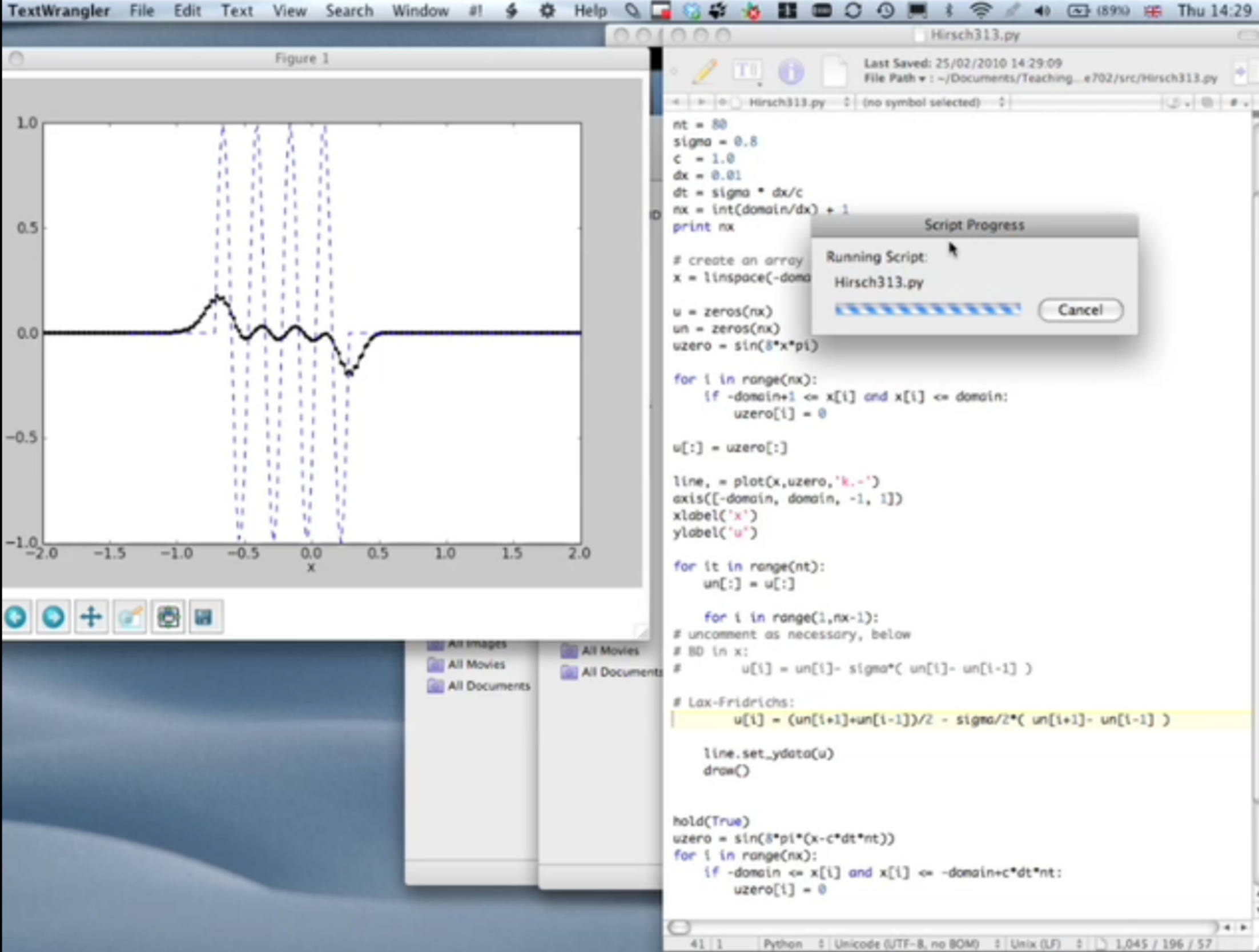
Running Script:

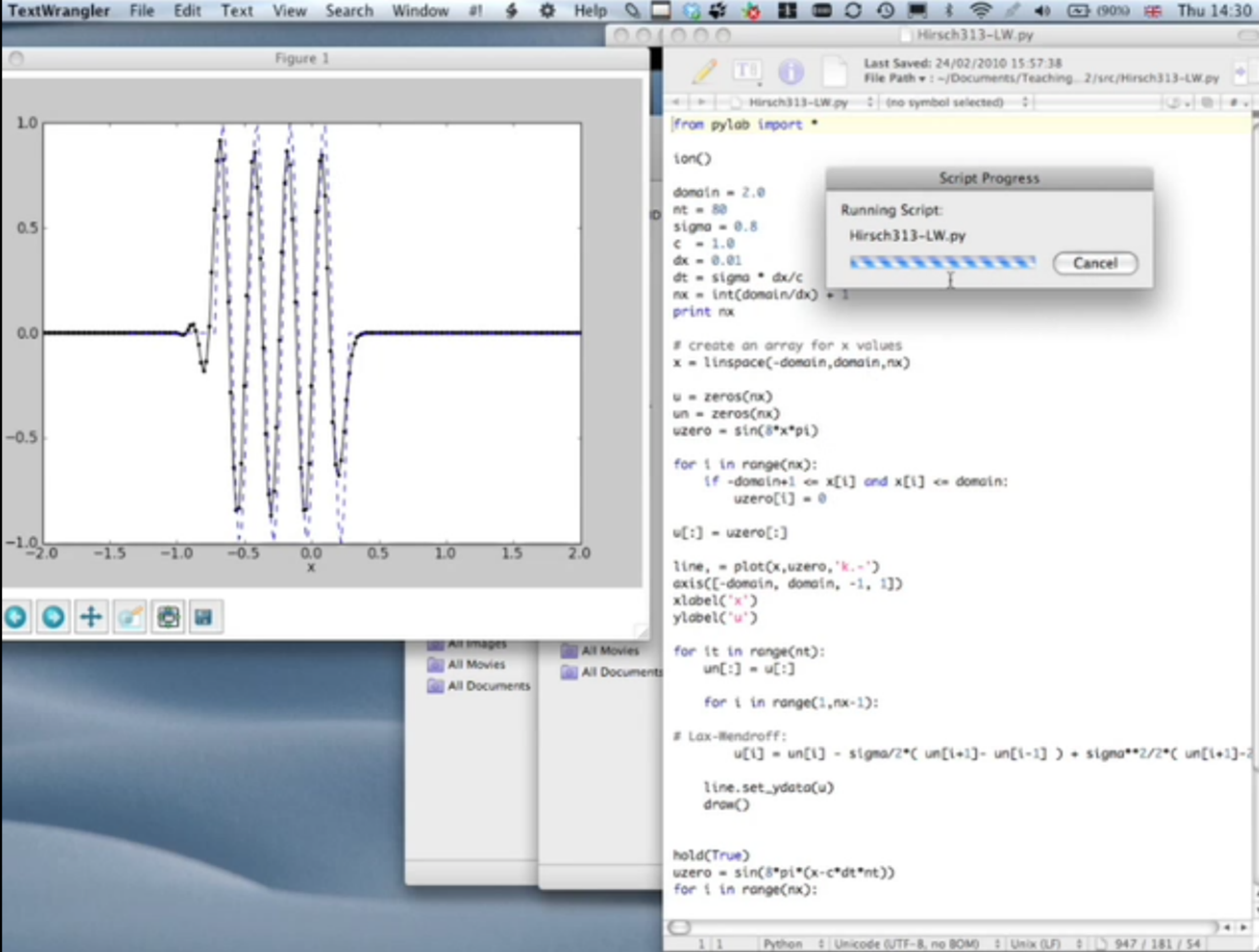
Hirsch313.py

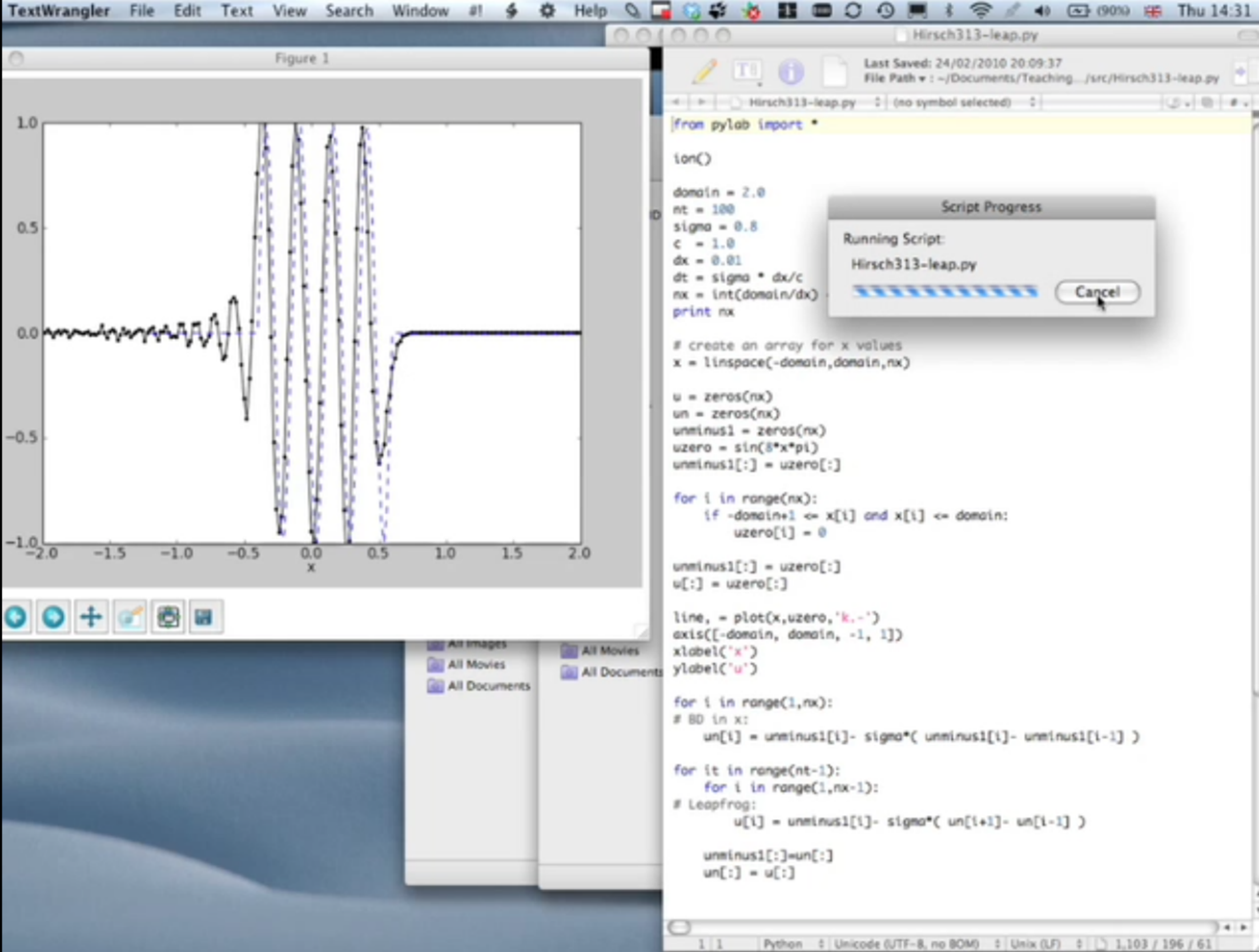
Cancel

All Images
All Movies
All Documents
All Movies
All Documents

1 | 1 Python Unicode (UTF-8, no BOM) Unix (LF) 1,045 / 196 / 57

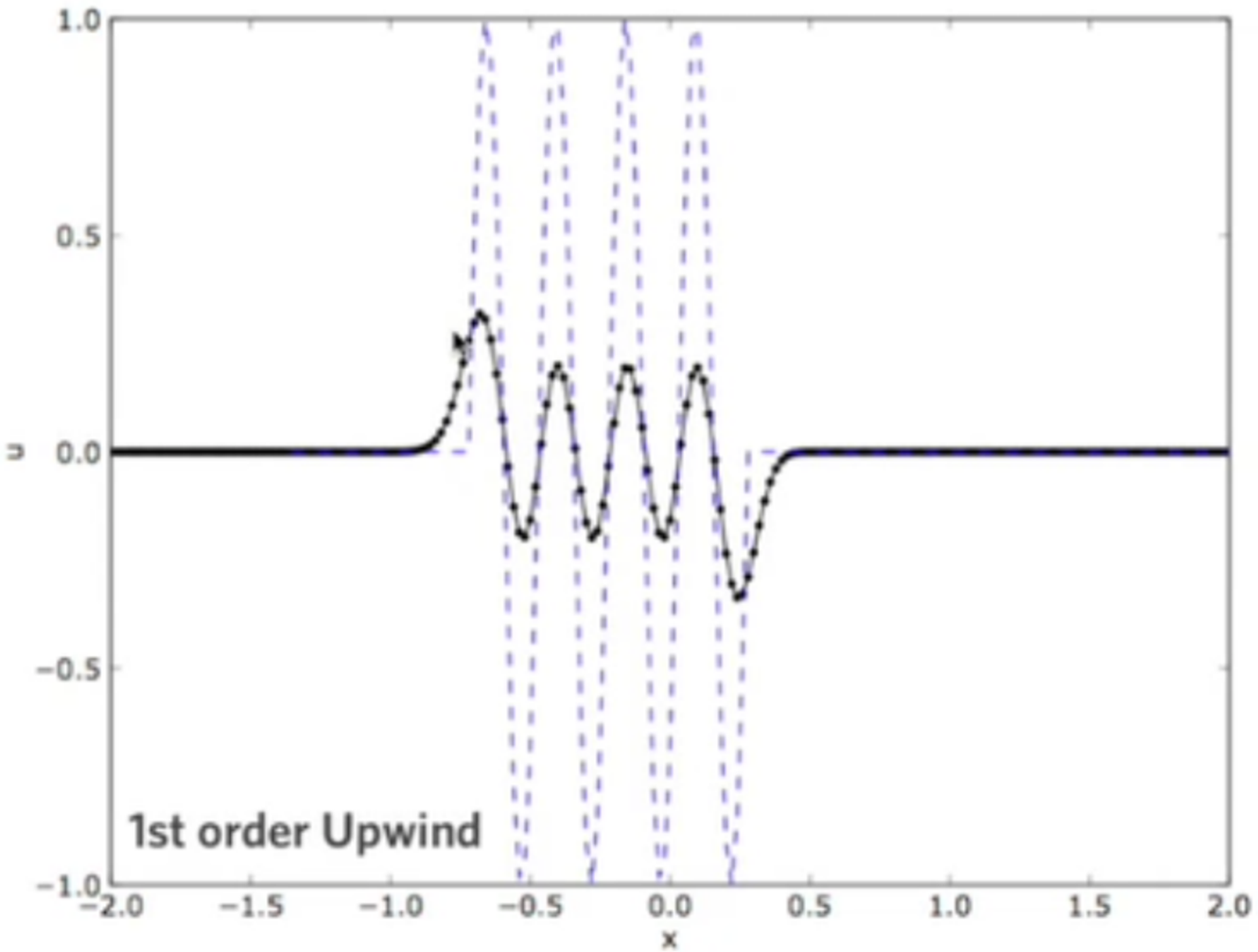

TextWrangler File Edit Text View Search Window #! Help Thu 14:29
Figure 1
1.0
0.5
0.0
−0.5
−1.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x
Hirsch313.py
Last Saved: 25/02/2010 14:29:09
File Path : ~/Documents/Teaching...e702/src/Hirsch313.py
Hirsch313.py
(no symbol selected)
nt = 80
sigma = 0.8
c = 1.0
dx = 0.01
dt = sigma * dx/c
nx = int(domain/dx) + 1
print nx
# create an array
x = linspace(-doma
u = zeros(nx)
un = zeros(nx)
uzero = sin(8*x*pi)
for i in range(nx):
if -domain+1 <= x[i] and x[i] <= domain:
uzero[i] = 0
u[:] = uzero[:]
line, = plot(x,uzero,'k.-')
axis([-domain, domain, -1, 1])
xlabel('x')
ylabel('u')
for it in range(nt):
un[:] = u[:]
for i in range(1,nx-1):
# uncomment as necessary, below
# BD in x:
# u[i] = un[i]- sigma*( un[i]- un[i-1] )
# Lax-Fridrichs:
u[i] = (un[i+1]+un[i-1])/2 - sigma/2*( un[i+1]- un[i-1] )
line.set_ydata(u)
draw()
hold(True)
uzero = sin(8*pi*(x-c*dt*nt))
for i in range(nx):
if -domain <= x[i] and x[i] <= -domain+c*dt*nt:
uzero[i] = 0
Script Progress
Running Script:
Hirsch313.py
Cancel
All Movies
All Documents
All Movies
All Documents
41 1 Python Unicode (UTF-8, no BOM) Unix (LF) 1,045 / 196 / 57


TextWrangler File Edit Text View Search Window #! Help (90%) Thu 14:30
Figure 1
1.0
0.5
0.0
−0.5
−1.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x
Hirsch313-LW.py
Last Saved: 24/02/2010 15:57:38
File Path : ~/Documents/Teaching...2/src/Hirsch313-LW.py
Hirsch313-LW.py
(no symbol selected)
from pylab import *
ion()
domain = 2.0
nt = 80
sigma = 0.8
c = 1.0
dx = 0.01
dt = sigma * dx/c
nx = int(domain/dx) + 1
print nx
# create an array for x values
x = linspace(-domain,domain,nx)
u = zeros(nx)
un = zeros(nx)
uzero = sin(8*x*pi)
for i in range(nx):
if -domain+1 <= x[i] and x[i] <= domain:
uzero[i] = 0
u[:] = uzero[:]
line, = plot(x,uzero,'k.-')
axis([-domain, domain, -1, 1])
xlabel('x')
ylabel('u')
for it in range(nt):
un[:] = u[:]
for i in range(1,nx-1):
# Lax-Wendroff:
u[i] = un[i] - sigma/2*( un[i+1]- un[i-1] ) + sigma**2/2*( un[i+1]-2
line.set_ydata(u)
draw()
hold(True)
uzero = sin(8*pi*(x-c*dt*nt))
for i in range(nx):
Script Progress
Running Script:
Hirsch313-LW.py
Cancel
All Images
All Movies
All Documents
All Movies
All Documents
1 | 1 Python Unicode (UTF-8, no BOM) Unix (LF) 947 / 181 / 54


TextWrangler File Edit Text View Search Window #! Help (90%) Thu 14:31
Figure 1
1.0
0.5
0.0
−0.5
−1.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x
Hirsch313-leap.py
Last Saved: 24/02/2010 20:09:37
File Path : ~/Documents/Teaching.../src/Hirsch313-leap.py
Hirsch313-leap.py
(no symbol selected)
from pylab import *
ion()
domain = 2.0
nt = 100
sigma = 0.8
c = 1.0
dx = 0.01
dt = sigma * dx/c
nx = int(domain/dx)
print nx
# create an array for x values
x = linspace(-domain,domain,nx)
u = zeros(nx)
un = zeros(nx)
unminus1 = zeros(nx)
uzero = sin(8*x*pi)
unminus1[:] = uzero[:]
for i in range(nx):
if -domain+1 <= x[i] and x[i] <= domain:
uzero[i] = 0
unminus1[:] = uzero[:]
u[:] = uzero[:]
line, = plot(x,uzero,'k.-')
axis([-domain, domain, -1, 1])
xlabel('x')
ylabel('u')
for i in range(1,nx):
# BD in x:
un[i] = unminus1[i]- sigma*( unminus1[i]- unminus1[i-1] )
for it in range(nt-1):
for i in range(1,nx-1):
# Leapfrog:
u[i] = unminus1[i]- sigma*( un[i+1]- un[i-1] )
unminus1[:]=un[:]
un[:] = u[:]
Script Progress
Running Script:
Hirsch313-leap.py
Cancel
All Images
All Movies
All Documents
All Movies
All Documents
1 1 Python Unicode (UTF-8, no BOM) Unix (LF) 1,103 / 196 / 61

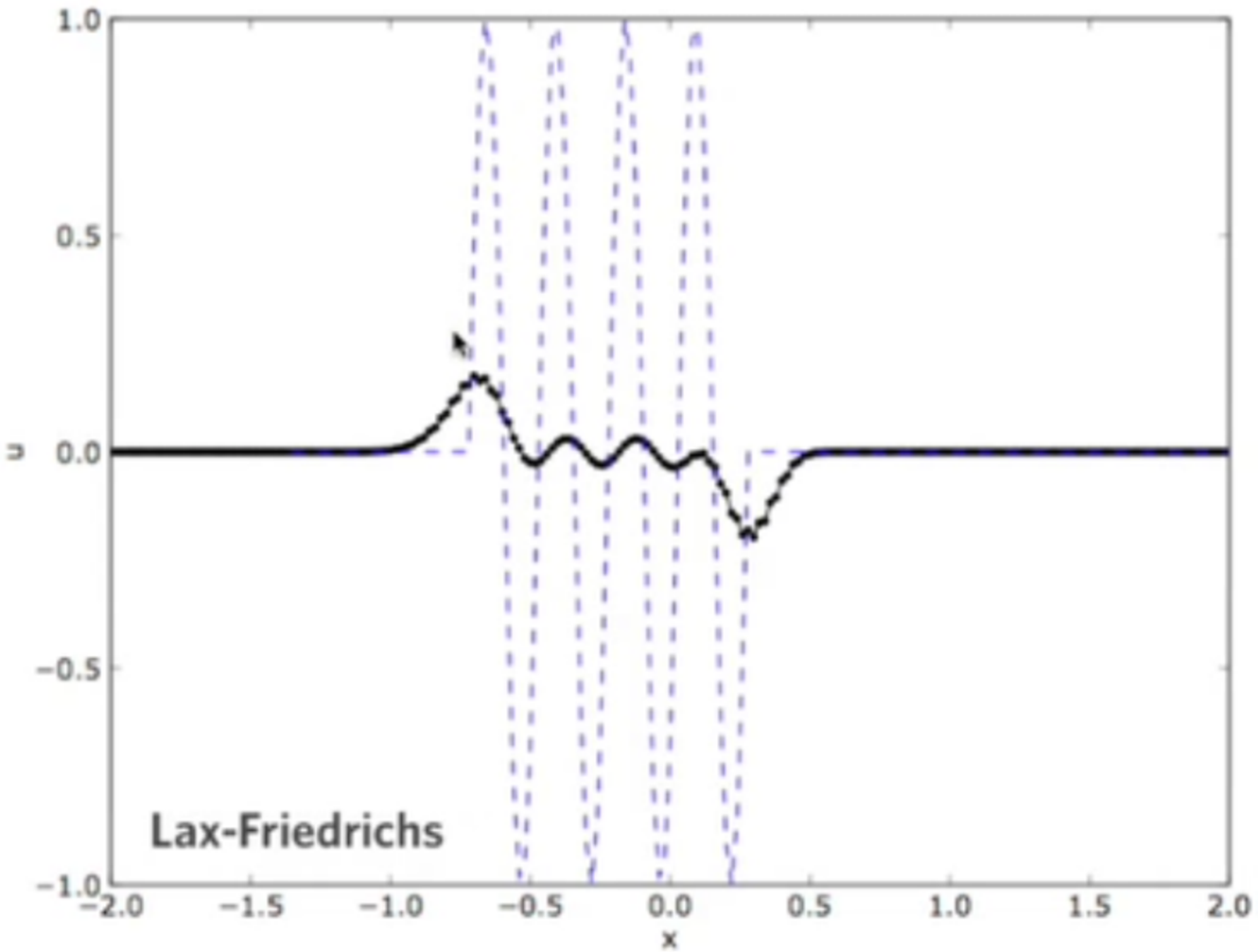
# *Test 3* : sinusoidal wave packet, double frequency

$$\sigma = 0.8$$
$$\Delta x = 0.02$$
$$n_{\text{steps}} = 80$$



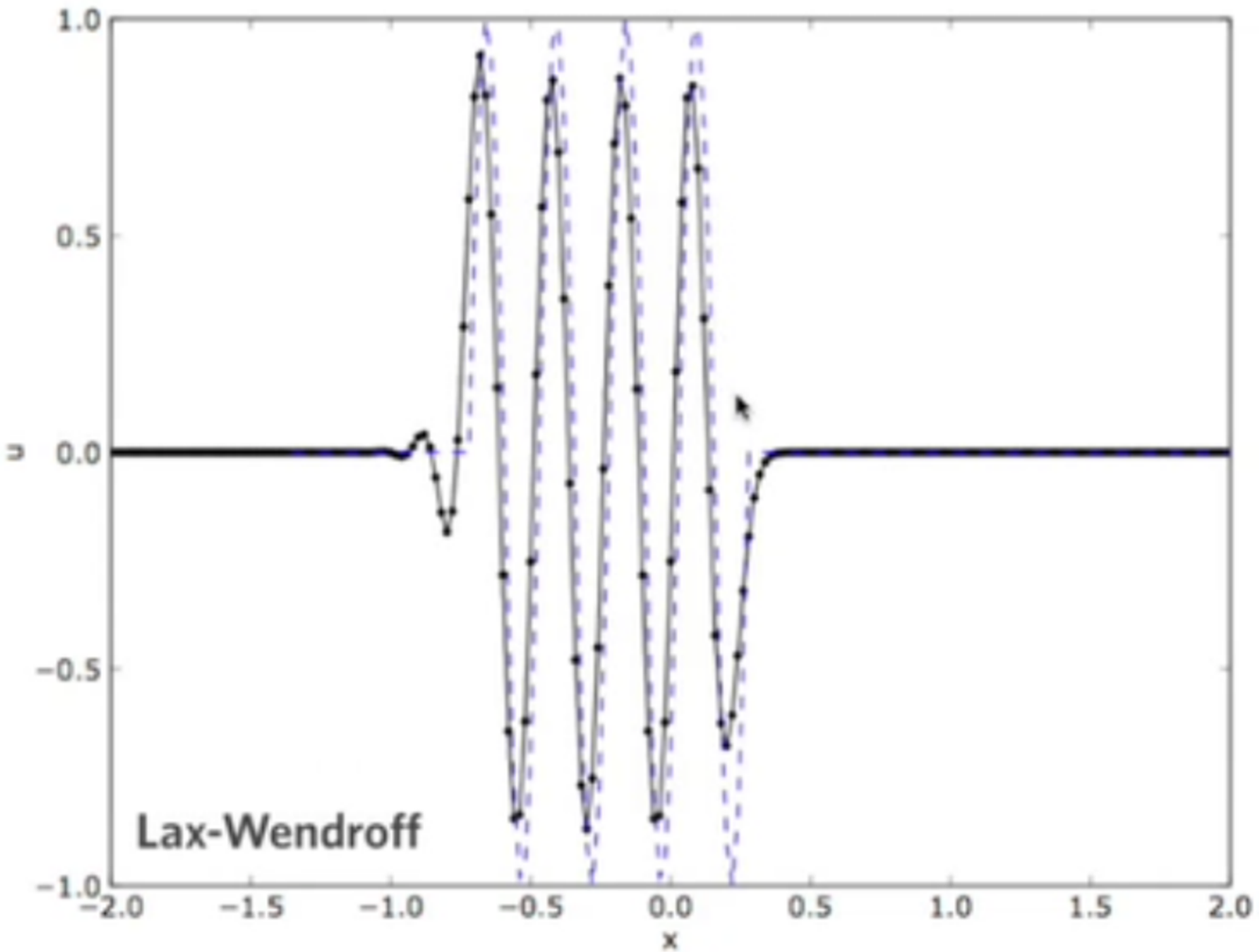
# *Test 3* : sinusoidal wave packet, double frequency

$$\sigma = 0.8$$
$$\Delta x = 0.02$$
$$n_{\text{steps}} = 80$$



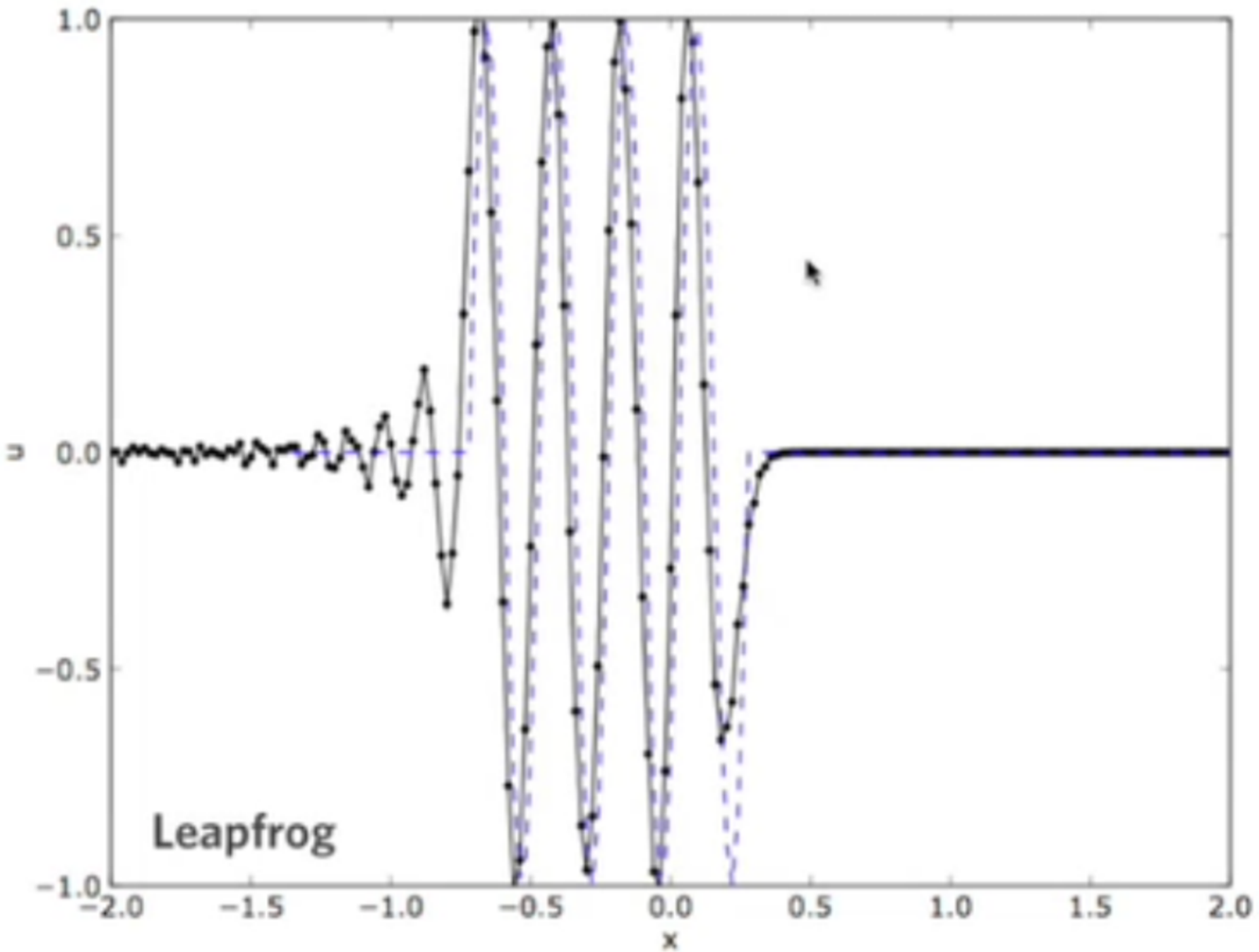
# *Test 2* : sinusoidal wave packet

$$\sigma = 0.8$$
$$\Delta x = 0.02$$
$$n_{\text{steps}} = 80$$



# *Test 2* : sinusoidal wave packet

$$\sigma = 0.8$$
$$\Delta x = 0.02$$
$$n_{\text{steps}} = 80$$



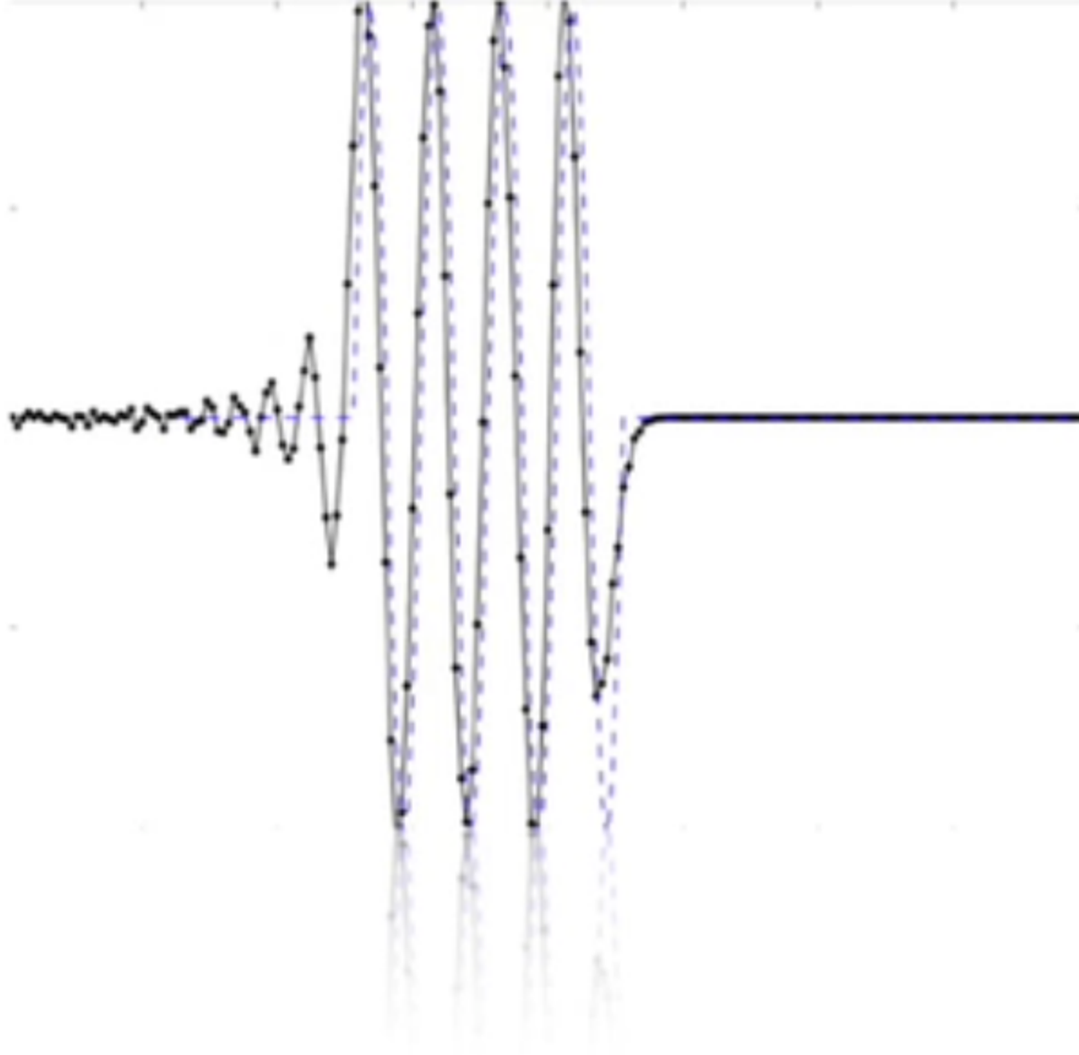
# Summary

- 1st order schemes ... have poor accuracy.
- 2nd order schemes ... provide better accuracy
  but generate numerical oscillations where the solution is not smooth
  - Oscillations stronger with leapfrog.

* Numerical errors are sensitive to the frequency of the solution $u(x,t)$

Results from simple 1D model ....
however they are representative real flow situations
2D, 3D etc.

# Spectral analysis of numerical errors



- Recall for von Neumann analysis ...

we introduced a Fourier decomposition of the solution :

$$u_i^n = \sum_{j=-N}^{N} V_j^n \, e^{I k_j x_i} \quad ; \quad x_i = i\Delta x$$

A single harmonic is $\longrightarrow \quad (u_i^n)_k = V^n \, e^{Ik(i\Delta x)}$

We defined an amplification factor $\longrightarrow \quad G = V^{n+1}/V^n$

... a function of the scheme parameters and of phase angle $\phi$, but not of "n".

- Von Neumann stability condition $|G| \leq 1 \quad \forall \phi_j = k_j \Delta x$

Now — want additional information on the errors

Wish to know more about the time dependence of $V^n$

- consider the analytical solution of $u_t + c\,u_x = 0$ ($\tilde{u}$)

Fourier decomposition $\rightarrow$ $\tilde{u}_i^n = \hat{V}\, e^{Ik(x_i - ct^n)}$

Rewrite with $c = \tilde{\omega}/k$ $\rightarrow$ $\tilde{u}_i^n = \hat{V}\, e^{Ikx_i}\, e^{-I\tilde{\omega}t^n}$

A single harmonic is $(\tilde{u}_i^n)_k = \hat{V}(k)\, e^{Ik(i\Delta x)}\, e^{-I\tilde{\omega}(n\Delta t)}$ (*)

with $\hat{V}(k)$ from the initial condition $u(x, t=0) = u_0(x)$

$$\hat{V}(k) = \frac{1}{2L}\int_{-L}^{L} u_0(x)\, e^{-Ikx}\, dx$$

$\rightarrow$ Assume that I.C. is represented exactly on the mesh (except for round-off error)

Numerical amplitude represented similarly to (*)

$$V^n = \hat{V}(k)\, e^{-I\omega(n\Delta t)} = \hat{V}(k)\left(e^{-I\omega\Delta t}\right)^n$$

$e^{-I\omega\Delta t} = G$ ← I.C.

Plane waves $\tilde{\omega} = \tilde{\omega}(k)$ $\rightarrow$ is called = Dispersion relation =

Now write: $V^n = G\,V^{n-1} = (G)^2 V^{n-2} = \ldots = (G)^n V^0 = (G)^n \cdot \hat{V}(k)$

$\Rightarrow \quad G = e^{-I\omega\Delta t} \quad \rightarrow$ this defines $\omega(k)$

Similarly with the analytical solution:

$$\tilde{V}^n = \left(e^{-I\tilde{\omega}\Delta t}\right)^n \hat{V}(k) =: \left(\tilde{G}\right)^n \hat{V}(k)$$

↑ Exact amplification factor

— Note that $\omega$ is a complex function.

So $G = |G| e^{-I\Phi}$ and $V^n = GV^{n-1} = |G| e^{-I\Phi} V^{n-1}$

* The error in amplitude is:

$$\epsilon_D = \frac{|G|}{|\tilde{G}|}$$

} Diffusion (or dissipation) error

* The error on phase of the solution is

$$\epsilon_\phi = \frac{\Phi}{\tilde{\Phi}}$$

} Dispersion error

$\tilde{\Phi} = kc\Delta t$

(For convection-dominated flows, but for pure diffusion problem $\tilde{\Phi} = 0$, use the definition $\epsilon_\phi = \Phi - \tilde{\Phi}$

- Error analysis for *hyperbolic* problems $u_t + c\,u_x = 0$

The exact solution for a wave form (*) is $\tilde{\omega} = ck$

and $\tilde{u} = \hat{V}\, e^{Ikx}\, e^{-Ikct}$

* Exact amplification factor $|\tilde{G}| = 1$

and $\tilde{\Phi} = ck\Delta t = c\,\frac{\Delta t}{\Delta x}\,.\,k\Delta x = \sigma\phi$

$\therefore \tilde{G} = e^{-I\sigma\phi}$ $\longrightarrow$ Exact solution propagates without change in amplitude

* Numerical solution : initial wave damped by a factor $|G|$ each $\Delta t$

Diffusion error is $\epsilon_D = |G|$

Phase of num. soln. defines a numerical convection speed ;

$c_{num} = \Phi / k\Delta t$ and since $\tilde{\Phi} = ck\Delta t = \sigma\phi$

$\Rightarrow c_{num} = \frac{c\,\Phi}{\sigma\phi}$

Dispersion error :

$\epsilon_\phi = \frac{\Phi}{ck\Delta t} = \frac{\Phi}{\sigma\phi} = \frac{c_{num}}{c}$

- Error analysis for ***hyperbolic*** problems $u_t + c\,u_x = 0$

- When dispersion error is larger than 1, $\epsilon_\phi > 1$,
  the phase error is a "leading error"
  – the numerical convection speed $c_{num}$ is larger than the exact $c$
  & computed solution moves faster than physical one
- When $\epsilon_\phi < 1$, the phase error is a "lagging error"
  & computed solution travels at lower velocity than the physical one,

REMARK Note that accuracy requires $|G|$ to be as close to 1 as possible, but stability requires $|G| < 1$

To maintain stability — always diffusion error.

## ① — Analysis of *1st order upwind*

We found $G = 1 - 2\sigma \sin^2 \phi/2 - I\sigma \sin\phi$

so $\xi \equiv Re(G) = 1 - 2\sigma \sin^2 \phi/2 = (1-\sigma) + \sigma\cos\phi$

$\eta = -Im(G) = -\sigma \sin\phi$

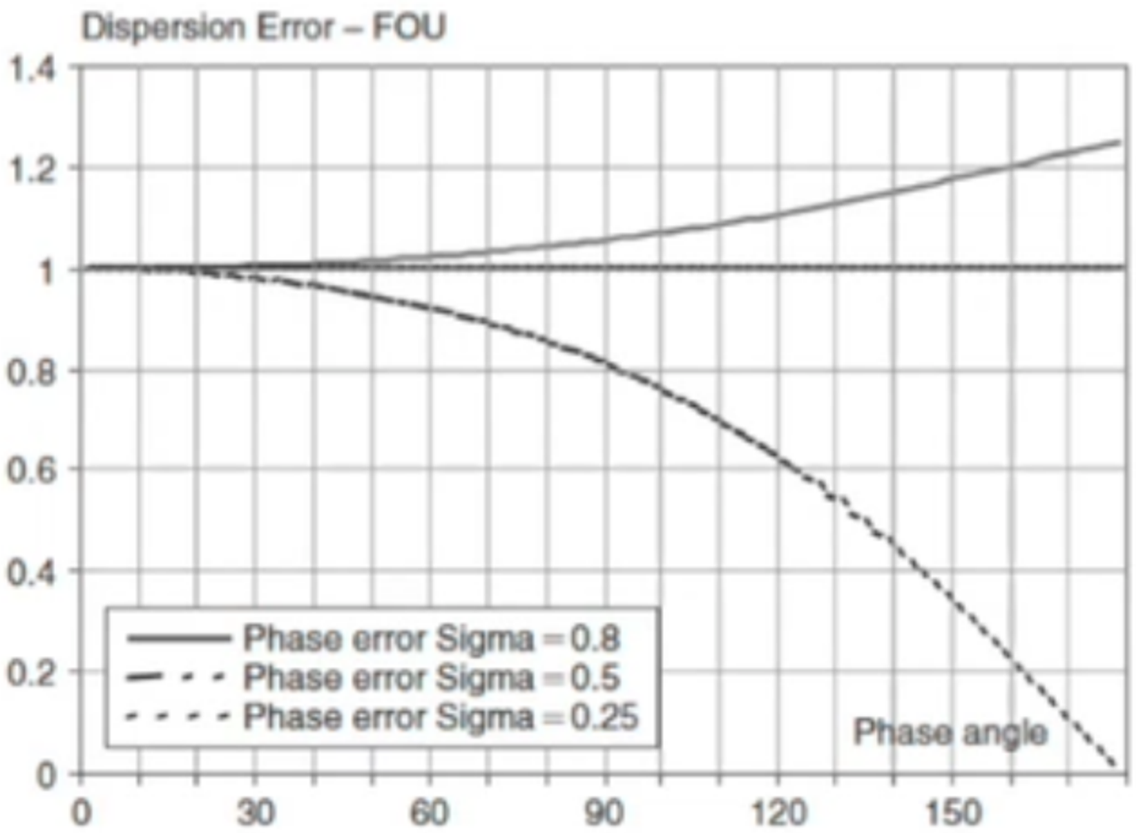
and $|G| = [(1-\sigma+\sigma\cos\phi)^2 + \sigma^2 \sin^2\phi]^{1/2} = \epsilon_D$

$= [1 - 4\sigma(1-\sigma)\sin^2 \phi/2]^{1/2}$

phase given by $\tan\phi = -\dfrac{Im(G)}{Re(G)}$

Phase error : $\epsilon_\phi = \dfrac{\tan^{-1}[(\sigma\sin\phi)/(1-\sigma+\sigma\cos\phi)]}{\sigma\phi}$

- *1st order upwind*




For
$\sigma > 0.5$ :
$\epsilon_\phi > 1 \Rightarrow$ leading

$\sigma = 0.5$ :
$\epsilon_\phi = 1$

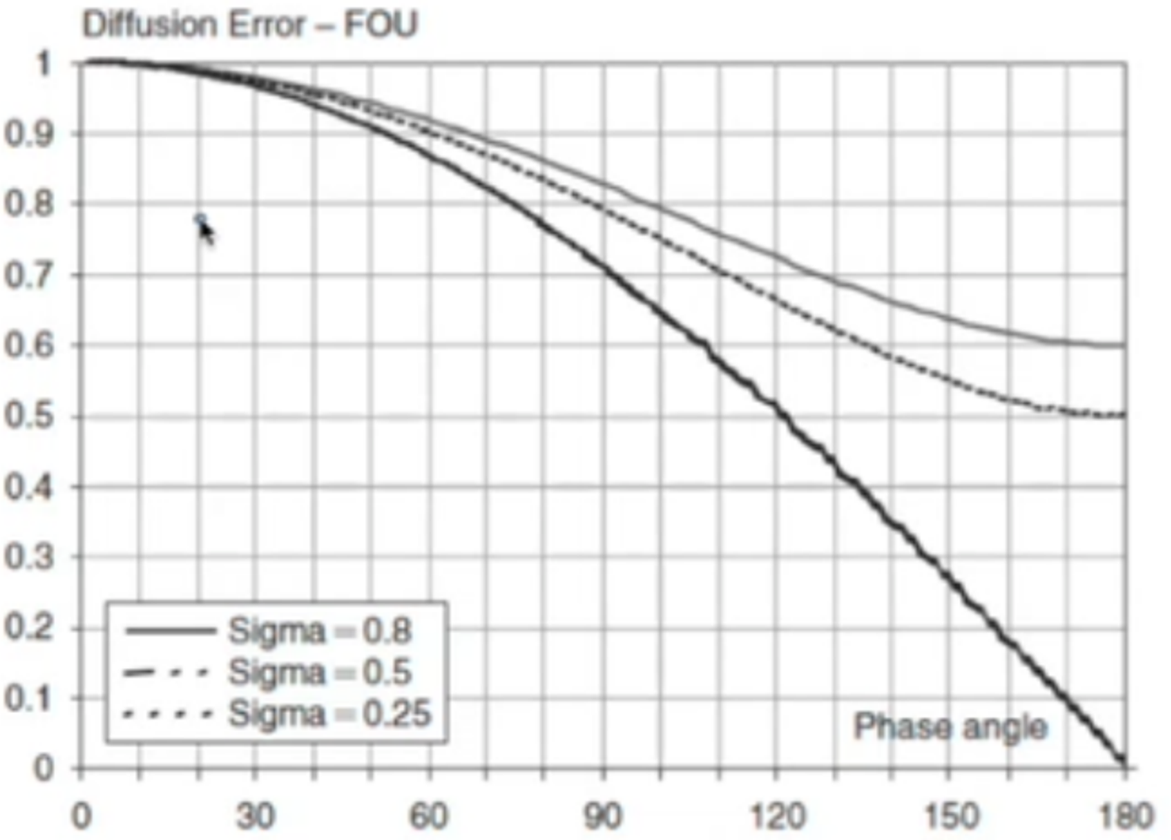
For
$\sigma < 0.5$ ;
$\epsilon_\phi < 1 \Rightarrow$ lagging

- *1st order upwind*

Test #2

$k = 4\pi$

$\phi = \frac{\pi}{12.5}$



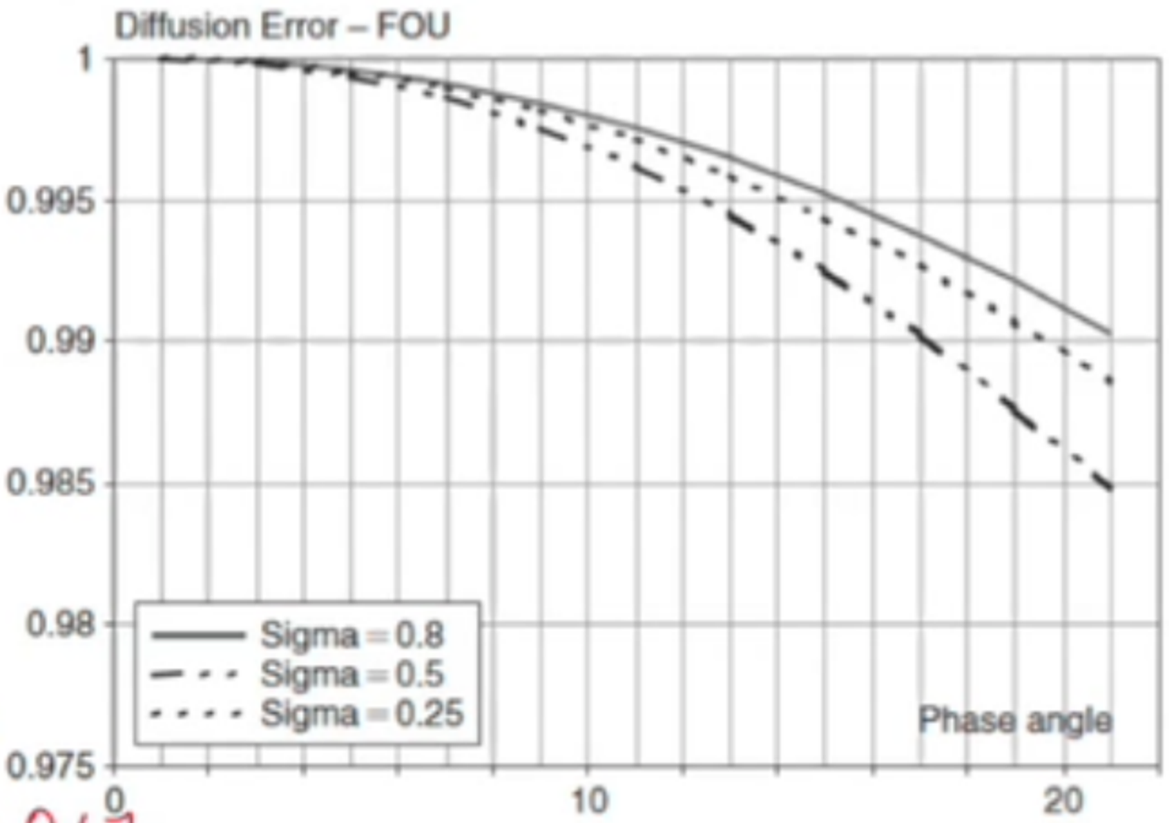

⟶ Increase in frequency

$\phi = k\Delta x$ & $k = \frac{2\pi}{\lambda}$ so $\phi = \frac{2\pi}{\lambda}\Delta x$

Highest frequency represented on mesh, shortest wavelength $\lambda = 2\Delta x$

$\Rightarrow \phi = \pi$

- 1st order upwind




Test #2

$k = 4\pi$

$\phi = \frac{\pi}{12.5}$

get ≈ diffusion error 0.995

For 80 time steps

$|G|^n = (0.995)^{80} = 0.67$

Test #3

$\phi = \pi/6.25 = 28.8°$

$|G|^n = (0.98)^{80}$

$= 0.2$ !!

An error of only 1%

$|G| = (0.99)$

After 100 steps leads to amplitude error of 36.6%

⟶ Increase in frequency

$\phi = k\Delta x$ & $k = \frac{2\pi}{\lambda}$ so $\phi = \frac{2\pi}{\lambda}\Delta x$

Highest frequency represented on mesh, shortest wavelength $\lambda = 2\Delta x$

$\Rightarrow \phi = \pi$

## ② — Analysis of *Lax-Friedrichs*

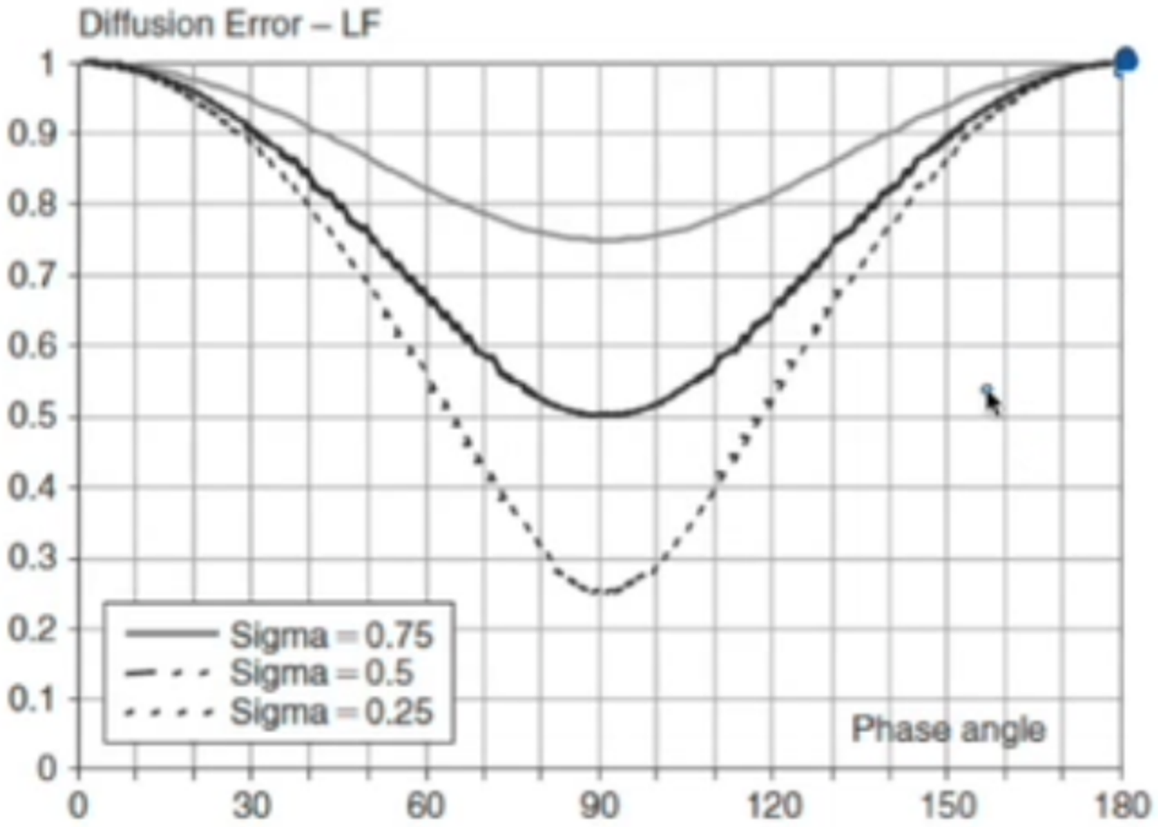
$$|G| = \left[ \cos^2\phi + \sigma^2 \sin^2\phi \right]^{1/2}$$

$$\Phi = \tan^{-1}(\sigma \tan\phi)$$

* Dissipation error : $\epsilon_D = \left[ \cos^2\phi + \sigma^2 \sin^2\phi \right]^{1/2}$

* Dispersion error : $\epsilon_\phi = \dfrac{\Phi}{\sigma\phi} = \dfrac{\tan^{-1}(\sigma \tan\phi)}{\sigma\phi}$

- *Lax-Friedrichs*

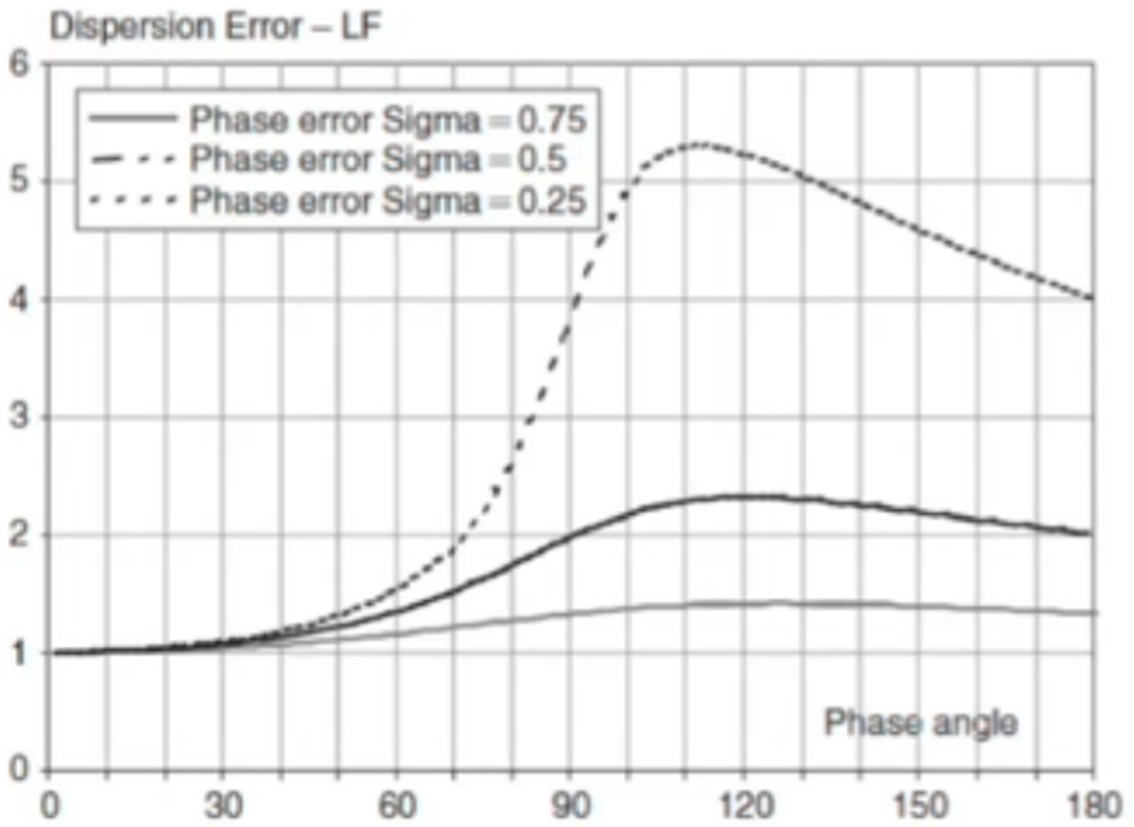



No diffusion error for $\phi = \pi$

wavelength $2\Delta x$

error if $\lambda = 2\Delta x$ will not be damped

- *Lax-Friedrichs*




$\epsilon_\phi > 1$

"leading"

# ③ — Analysis of *Lax-Wendroff*

$$|G| = \left[1 - 4\sigma^2(1-\sigma^2)\sin^4\phi/2\right]^{1/2} \rightarrow \text{dissipation error}$$

& phase error $\epsilon_\phi = \dfrac{\tan^{-1}\left[(\sigma\sin\phi)/(1-2\sigma^2\sin^2\phi/2)\right]}{\sigma\phi}$

= accuracy region = where $\epsilon_D \approx 1$

Test #3

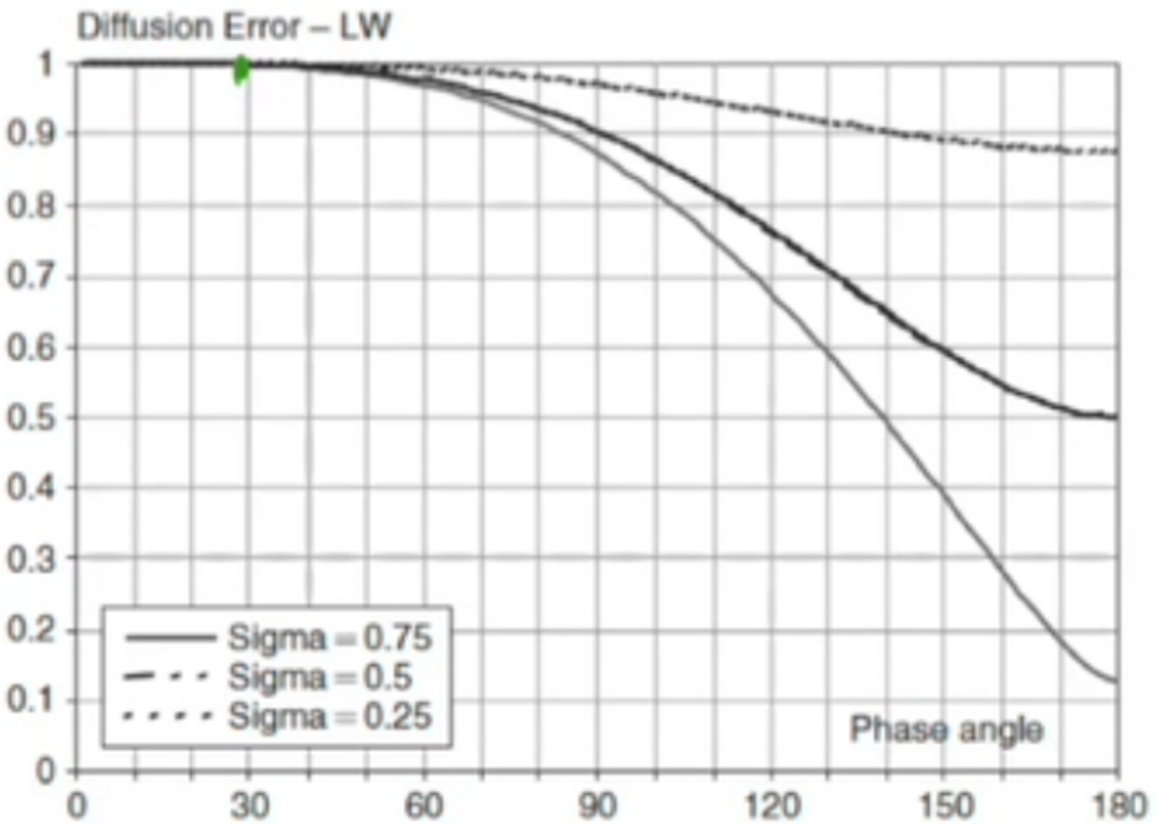
$\phi = \pi/6.25 = 28.8°$

for $\sigma = 0.8$ gives

$\epsilon_D \simeq 0.9985$

↳ after 80 steps

get ~0.89



# ③ — Analysis of *Lax-Wendroff*

$$|G| = \left[1 - 4\sigma^2(1-\sigma^2)\sin^4\phi/2\right]^{1/2} \rightarrow \text{dissipation error}$$

& phase error $\epsilon_\phi = \dfrac{\tan^{-1}\left[(\sigma\sin\phi)/(1-2\sigma^2\sin^2\phi/2)\right]}{\sigma\phi}$

= accuracy region = where $\epsilon_D \approx 1$

Test #3

$\phi = \pi/6.25 = 28.8°$

for $\sigma = 0.8$ gives

$\epsilon_D \simeq 0.9985$

↳ after 80 steps

get ~0.89