*Pour mémoire : programmes python3 des programmes Matlab du livre de référence de Trefethen "Spectral methods in Matlab" (Denise Vella-Chemla, juillet 2023)*

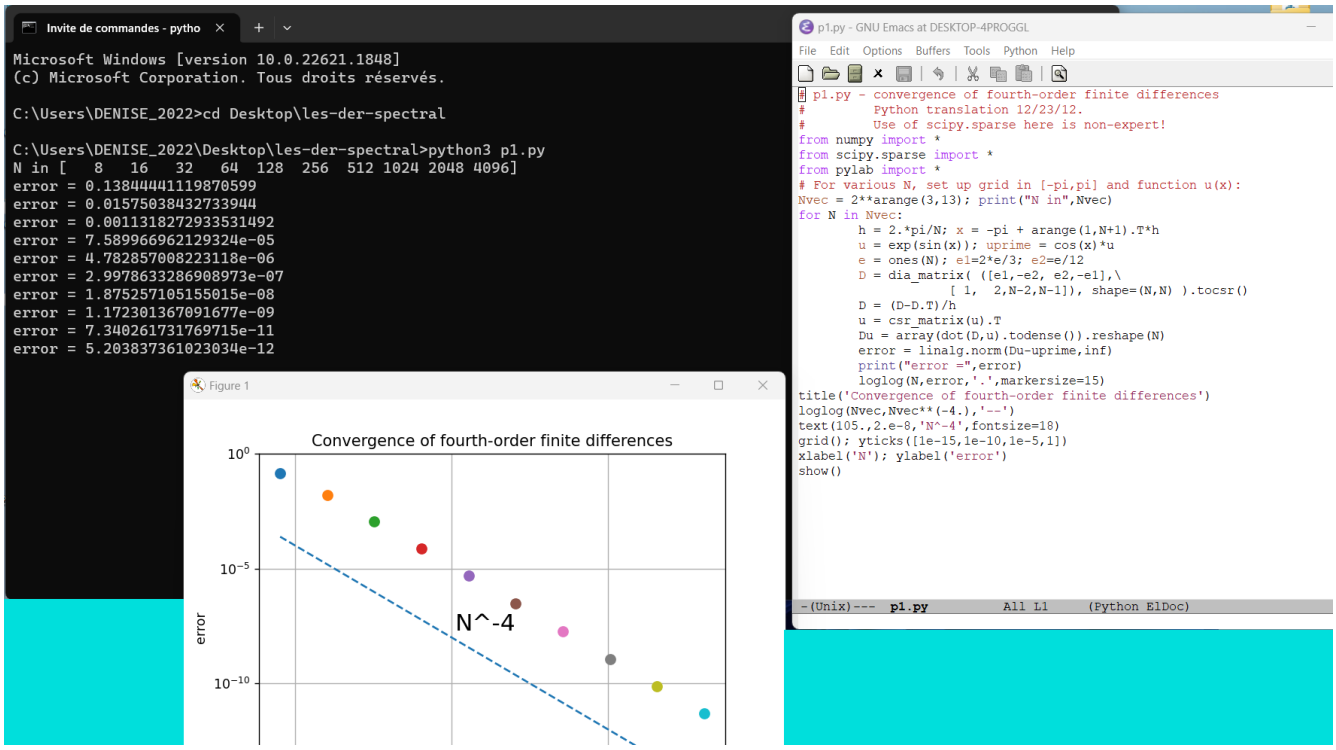
On essaie de se former tant bien que mal à l'analyse spectrale. Dans ce but, on a repris les programmes des deux pages ci-dessous pour qu'ils tournent de façon autonome en python3 et on les a accolés à leur résultat, pour mémoire.

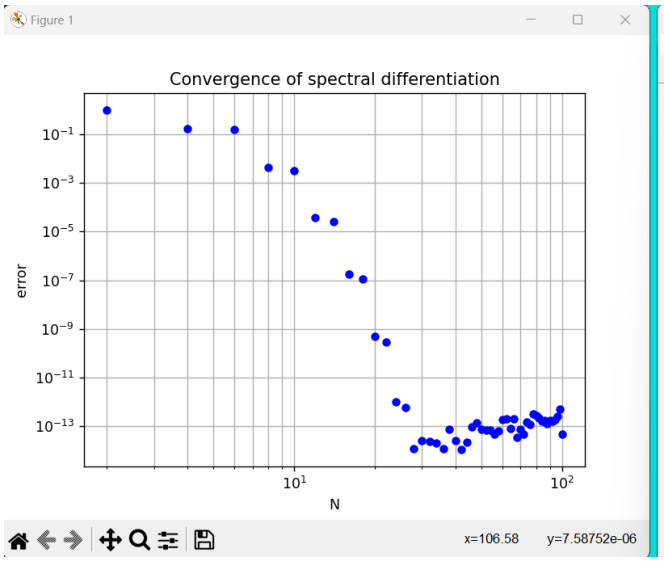
- les programmes (auteur inconnu : JR ?)

  https://blue.math.buffalo.edu/438/trefethen_spectral/all_py_files/ ;

- les programmes de Praveen Chandrashekar du Centre de Mathématiques appliquées de Bangalore en Inde

  http://cpraveen.github.io/teaching/chebpy.html ;

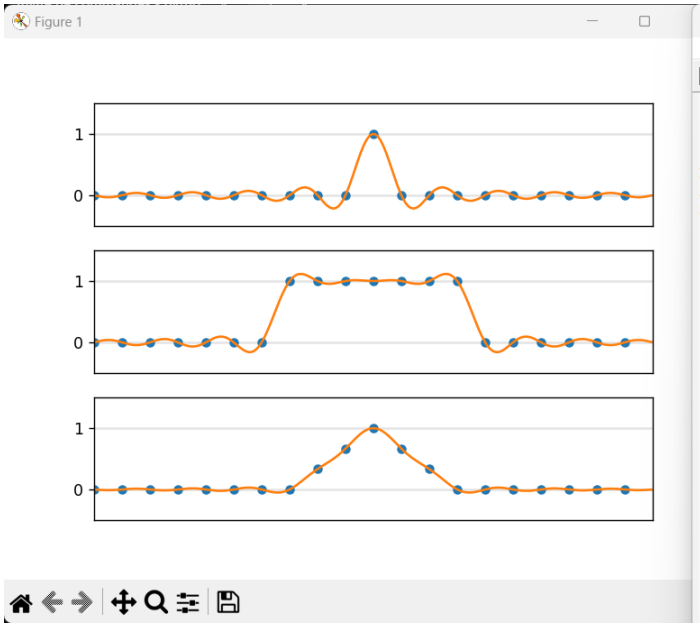





p2.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

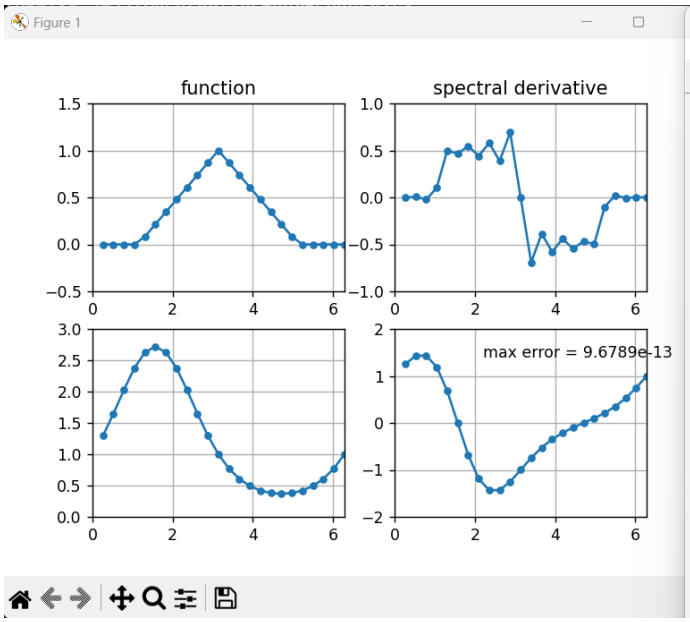
```
# p2.py - convergence of periodic spectral method (compare p1.m)
# JR translation 12/12/12

from numpy import *
from scipy.linalg import toeplitz
#from pylab import title, grid, loglog, xlabel, ylabel, show
from pylab import *

# For various N (even), set up grid as before:

for N in range(2,101,2):
        h = 2.*pi/N;
        x = -pi + arange(0,N).T*h # (including left endpoint instead of right)
        u = exp(sin(x)); uprime = cos(x)*u;
        # Construct spectral differentiation matrix:
        i = array( range(1,N) )
        column   = hstack(( [0.], .5*(-1)**i/tan(i*h/2.) ))
        row      = hstack(( [0.], column[N-1:0:-1]       ))
        D = toeplitz(column,row)
        error = linalg.norm(dot(D,u)-uprime,inf)
        loglog(N,error,'bo',markersize=5)

grid(True,which='both')
xlabel('N'); ylabel('error')
title('Convergence of spectral differentiation')
show()
```




p3.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

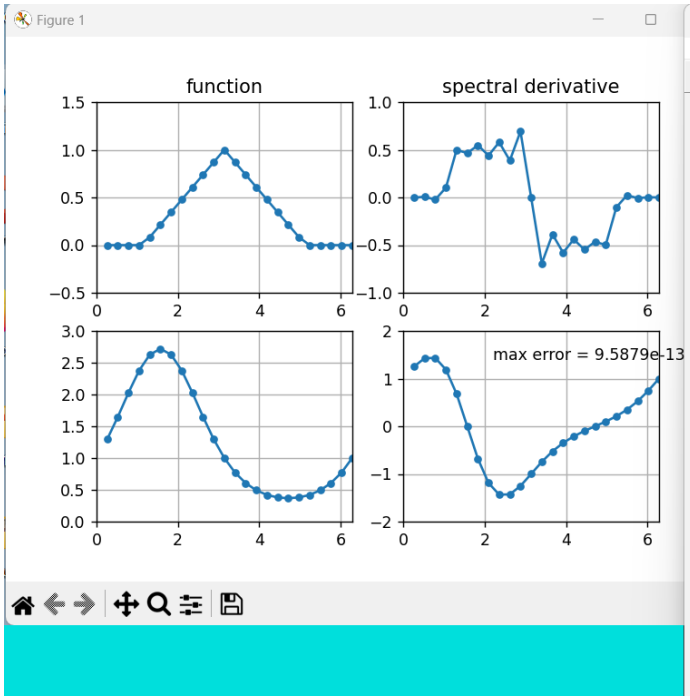
```
# p3.py - band-limited interpolation
# JR translation, 12/13/12
from numpy import *
from pylab import *
h = 1.; xmax = 10.
x = arange(-xmax,xmax,h)                # computational grid
xx = arange(-xmax-h/20,xmax+h/20,h/10)  # plotting grid
for pl in range(3):
        subplot(3,1,pl+1)
        if pl==0: v = x==0                      # delta function
        if pl==1: v = abs(x)<=3.                # square wave
        if pl==2: v = maximum(0.,1.-abs(x)/3.)  # hat function
        v = array(v,dtype=float) # convert boolean to float
        plot([-xmax,xmax],[0.,0.],'k',alpha=0.1)
        plot([-xmax,xmax],[1.,1.],'k',alpha=0.1)
        plot(x,v,'.',markersize=10)
        p = zeros_like(xx)
        for i,xi in enumerate(x):
                p += v[i]*sin(pi*(xx-xi)/h)/(pi*(xx-xi)/h)
        plot(xx,p)
        xlim(-xmax,xmax); ylim(-.5,1.5)
        xticks([]); yticks([0.,1.])
show()
```




p4.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

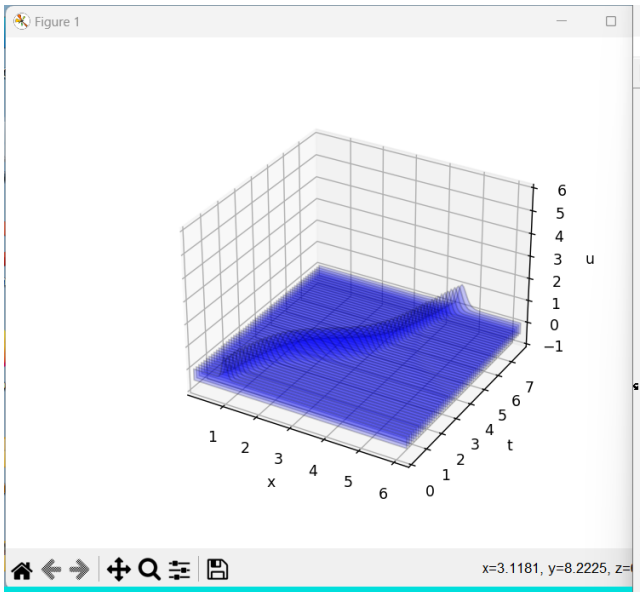
```
# p4.m - periodic spectral differentiation
# Translation 12/13/12
from numpy import *
from scipy.linalg import toeplitz
from pylab import *

# Set up grid and differentiation matrix:
N = 24; h = 2*pi/N; x = h*arange(1,N+1);
i = array( range(1,N) )
column   = hstack(( [0.], .5*(-1)**i/tan(i*h/2.) ))
row      = hstack(( [0.], column[N-1:0:-1]       ))
D = toeplitz(column,row)

# Differentiation of a hat function:
v = maximum(0.,1.-abs(x-pi)/2.)
subplot(2,2,1); plot(x,v,'.-',markersize=8)
xlim(0,2*pi); ylim(-0.5,1.5); grid(); title('function')
subplot(2,2,2); plot(x,dot(D,v),'.-',markersize=8)
xlim(0,2*pi); ylim(-1.,1.); grid(); title('spectral derivative')

# Differentiation of exp(sin(x)):
v = exp(sin(x)); vprime = cos(x)*v;
subplot(2,2,3); plot(x,v,'.-',markersize=8)
xlim(0,2*pi); ylim(0.,3.); grid()
subplot(2,2,4); plot(x,dot(D,v),'.-',markersize=8)
xlim(0,2*pi); ylim(-2.,2.); grid()
error = linalg.norm(dot(D,v)-vprime,inf);
text(2.2,1.4,'max error = '+str('%.4e' % error))
show()
```




p5.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

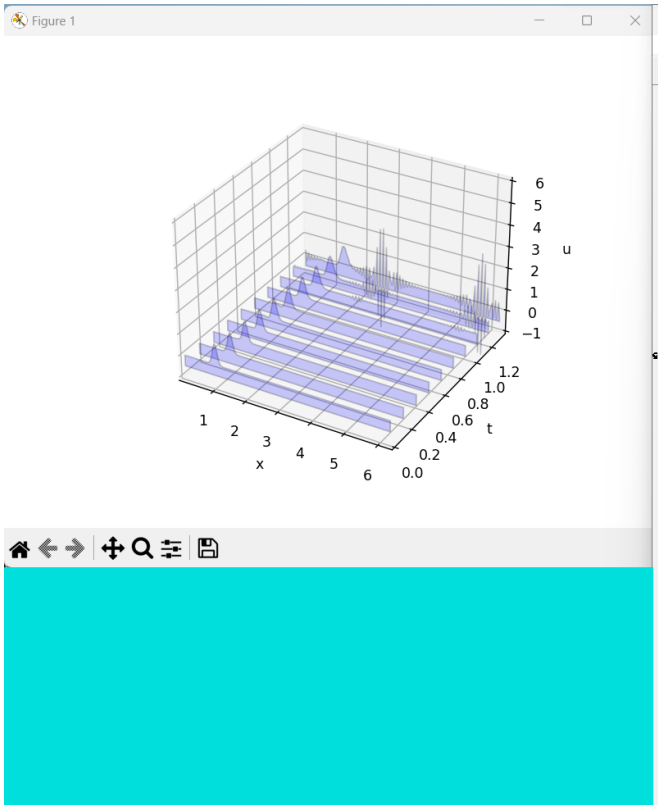
```
# p5.py - repetition of p4.py via FFT
#         For complex v, delete "real" commands.
#         Python/NumPy translation 12/14/12

from numpy import *
from numpy.fft import fft,ifft
from pylab import *

# Differentiation of a hat function:
N = 24; h = 2*pi/N; x = h*arange(1,N+1)
v = maximum(0.,1.-abs(x-pi)/2.); v_hat = fft(v)
w_hat = 1j*hstack(( range(N//2),[0.],range(-N//2+1,0) ))*v_hat
w = real(ifft(w_hat))
subplot(2,2,1), plot(x,v,'.-',markersize=8)
#axis([0 2*pi -.5 1.5]), grid on, title('function')
xlim(0,2*pi); ylim(-0.5,1.5); grid(); title('function')
subplot(2,2,2), plot(x,w,'.-',markersize=8)
xlim(0,2*pi); ylim(-1.,1.); grid(); title('spectral derivative')

# Differentiation of exp(sin(x)):
v = exp(sin(x)); vprime = cos(x)*v
v_hat = fft(v)
w_hat = 1j*hstack(( range(N//2),[0.],range(-N//2+1,0) ))*v_hat
w = real(ifft(w_hat))
subplot(2,2,3); plot(x,v,'.-',markersize=8)
xlim(0,2*pi); ylim(0.,3.); grid()
subplot(2,2,4); plot(x,w,'.-',markersize=8)
xlim(0,2*pi); ylim(-2.,2.); grid()
error = linalg.norm(w-vprime,inf)
text(2.2,1.4,'max error = '+str('%.4e' % error))

show()
```




p6.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

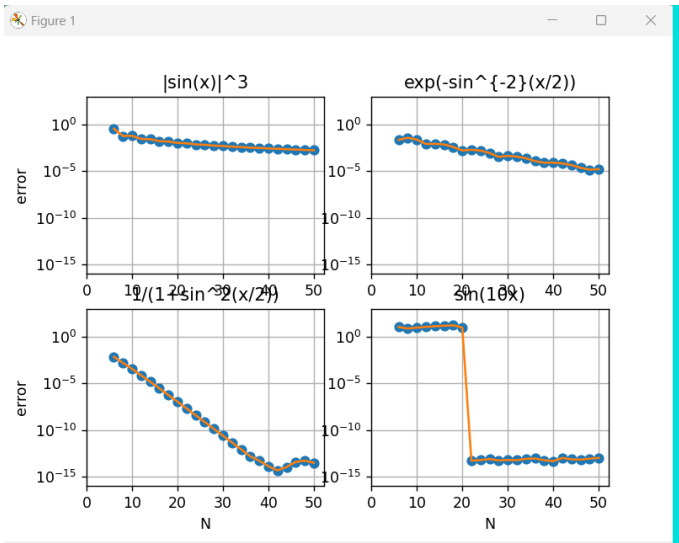
```
import numpy as np ; from numpy import * ; from numpy.fft import fft,ifft
import matplotlib.pyplot as plt ; from mpl_toolkits import mplot3d
import matplotlib ; from matplotlib.colors import to_rgba
from matplotlib.collections import PolyCollection

def waterfall(x,t,u,labels=['x','t','u'],slabthickness=0.5,zrange=[-1.,6.]):
        fig = plt.figure()
        ax = plt.axes(projection='3d')
        cc = lambda arg: matplotlib.colors.to_rgba(arg, alpha=1.0)
        xs = np.hstack(( x,x[-1],x[0] ))
        verts = []
        zs = t
        baseline  = 0.        -                          slabthickness
        for ti,z in enumerate(zs):
                ys = np.hstack(( u[ti,:],baseline,baseline ))
                verts.append(list(zip(xs, ys)))
        poly = PolyCollection(verts, edgecolors = 'black', facecolors = [cc('blue
e')], alpha=.6)
        poly.set_alpha(0.2)
        ax.add_collection3d(poly, zs=zs, zdir='y')
        ax.set_xlabel(labels[0]) ; ax.set_xlim3d(min(xs),max(xs))
        ax.set_ylabel(labels[1]) ; ax.set_ylim3d(min(zs),max(zs))
        ax.set_zlabel(labels[2])
        ax.set_zlim3d(zrange[0],zrange[1])

N=128; h = 2*pi/N; x = h*arange(1,N+1); t = 0.; dt = h/4
c = .2 + sin(x-1)**2
v    = exp(-100*(x      -1)**2)
vold = exp(-100*(x-.2*dt-1)**2)
tmax = 8.; tplot = .15
plotgap = int(round(tplot/dt)); dt = tplot/plotgap
nplots  = int(round(tmax/tplot))
data = vstack(( v, zeros((nplots,N)) )); tdata = [t]
for i in range(nplots):
        for n in range(plotgap):
                t += dt;
                v_hat = fft(v)
                w_hat = 1j*hstack(( range(N//2),[0.],range(-N//2+1,0) ))*v_hat
                w = real(ifft(w_hat))
                vnew = vold - 2*dt*c*w; vold = v.copy(); v = vnew
        data[i+1,:] = v; tdata.append(t)
waterfall(x,tdata,data)
plt.show()
```




p6u.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

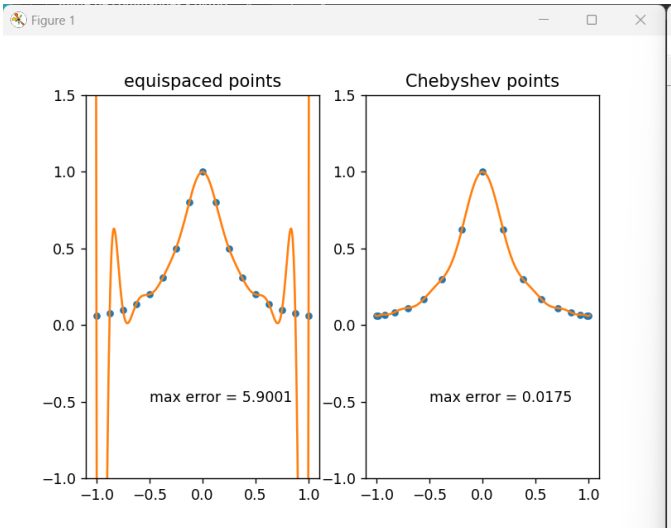
```
import numpy as np ; from numpy import *
from numpy.fft import fft,ifft
import matplotlib ; import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d ; from matplotlib.colors import to_rgba
from matplotlib.collections import PolyCollection

def waterfall(x,t,u,labels=['x','t','u'],slabthickness=0.5,zrange=[-1.,6.]):
        fig = plt.figure()
        ax = plt.axes(projection='3d')
        cc = lambda arg: matplotlib.colors.to_rgba(arg, alpha=1.0)
        xs = np.hstack(( x,x[-1],x[0] ))
        verts = [] ; zs = t ; baseline  = 0.-slabthickness
        for ti,z in enumerate(zs):
                ys = np.hstack(( u[ti,:],baseline,baseline ))
                verts.append(list(zip(xs, ys)))
        poly = PolyCollection(verts, edgecolors = 'black', facecolors = [cc('blue
e')], alpha=.4)
        poly.set_alpha(0.2)
        ax.add_collection3d(poly, zs=zs, zdir='y')
        ax.set_xlabel(labels[0]) ; ax.set_xlim3d(min(xs),max(xs))
        ax.set_ylabel(labels[1]) ; ax.set_ylim3d(min(zs),max(zs))
        ax.set_zlabel(labels[2]) ; ax.set_zlim3d(zrange[0],zrange[1])

N=128; h = 2*pi/N; x = h*arange(1,N+1)
c = .2 + sin(x-1)**2
t = 0.; dt = 1.9/N
v    = exp(-100*(x      -1)**2)
vold = exp(-100*(x-.2*dt-1)**2)
tmax = 1.4; tplot = .15
plotgap = int(round(tplot/dt)); dt = tplot/plotgap
nplots  = int(round(tmax/tplot))
data = vstack(( v, zeros((nplots,N)) )); tdata = [t]
for i in range(nplots):
        for n in range(plotgap):
                t += dt;
                v_hat = fft(v)
                w_hat = 1j*hstack(( range(N//2),[0.],range(-N//2+1,0) ))*v_hat
                w = real(ifft(w_hat))
                vnew = vold - 2*dt*c*w;                 # leap frog formula
                vold = v.copy(); v = vnew
        data[i+1,:] = v; tdata.append(t)
ax = waterfall(x,tdata,data)
plt.show()
```




```
p7.py - GNU Emacs at DESKTOP-4PROGGL
File  Edit  Options  Buffers  Tools  Python  Help
from numpy import *
from scipy.linalg import toeplitz
from pylab import *
set_printoptions(linewidth=200,precision=5)
# Compute derivatives for various values of N:
Nmax = 50; E = zeros((4,Nmax//2-2))
Ns = range(6,Nmax+2,2)
for N in Ns:
        h = 2*pi/N; x = h*arange(1,N+1)
        i = array( range(1,N) )
        column   = hstack(( [0.], .5*(-1)**i/tan(i*h/2.) ))
        row      = hstack(( [0.], column[N-1:0:-1]       ))
        D = toeplitz(column,row)
        #print D
        v = abs(sin(x))**3
        # 3rd deriv in BV
        vprime = 3*sin(x)*cos(x)*abs(sin(x))
        E[0,N//2-3] = linalg.norm(dot(D,v)-vprime,inf)
        v = exp(-sin(x/2)**(-2))
# C-infinity
        vprime = .5*v*sin(x)/sin(x/2)**4
        E[1,N//2-3] = linalg.norm(dot(D,v)-vprime,inf)
        v = 1/(1+sin(x/2)**2)
# analytic in a strip
        vprime = -sin(x/2)*cos(x/2)*v**2
        E[2,N//2-3] = linalg.norm(dot(D,v)-vprime,inf)
        v = sin(10*x); vprime = 10*cos(10*x)            # band-limited
        E[3,N//2-3] = linalg.norm(dot(D,v)-vprime,inf)

# Plot results:
titles = ['|sin(x)|^3','exp(-sin^{-2}(x/2))',\
          '1/(1+sin^2(x/2))','sin(10x)'      ]
for iplot in [0,1,2,3]:
        subplot(2,2,iplot+1)
        semilogy(Ns,E[iplot,:],'.',markersize=12)
        plot(Ns,E[iplot,:])
        ylim(1e-16, 1e3)
        xticks(arange(0,Nmax+10,10));
        if iplot>1: xlabel('N')
        yticks(10.**(arange(-15,5,5)));
        if iplot%2==0: ylabel('error')
        grid()
        title(titles[iplot])
show()
```

```
C:\Users\DENISE_2022\Desktop\les-der-spectral>python3 p8.py
6 [ 0.46147291699547  7.49413462105052  7.72091605300656 28.8324837783401 ]
12 [0.97813728129861 3.17160532064719 4.4559352911668  8.92452905811993]
18 [0.9999700014993  3.00064406679582 4.99259532440077  7.03957189798149]
24 [0.99999999762905 3.00000009841086 4.99999796527329 7.00002499815655]
30 [0.99999999999998 3.00000000000073 4.9999999999756  7.0000000005086 ]
36 [1.               3.               4.99999999999999 7.              ]

C:\Users\DENISE_2022\Desktop\les-der-spectral>
```

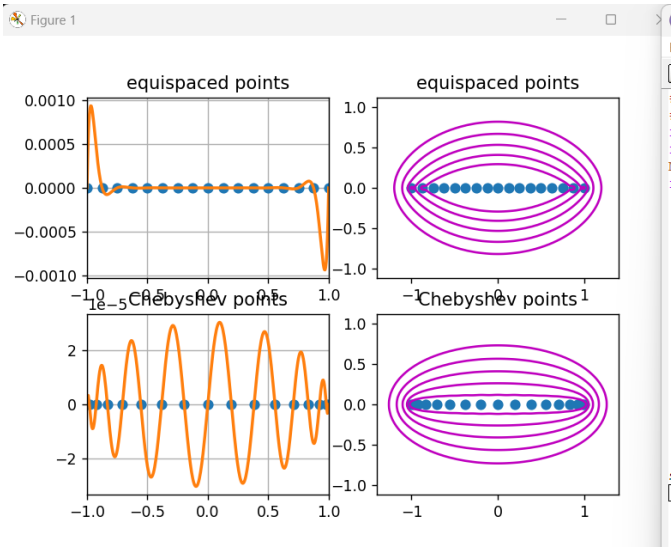
```
# p8.py - eigenvalues of harmonic oscillator -u"+x^2 u on R
#         Python translation 12/19/12
from numpy import *
from scipy.linalg import toeplitz
set_printoptions(linewidth=200,precision=14)

L = 8.                                  # domain is [-L L], periodic
for N in range(6,42,6):
        h = 2*pi/N; x = h*arange(1,N+1); x = L*(x-pi)/pi
        i = array( range(1,N) )
        column = hstack(( [-pi**2/(3*h**2)-1./6.] ,\
                          -.5*(-1)**i/sin(h*i/2.)**2 ))
        D2 = (pi/L)**2*toeplitz(column)  # 2nd-order differentiation
        eigenvalues = sort(linalg.eigvals(-D2 + diag(x**2)))
        print(N, eigenvalues[0:4])
```




p9.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

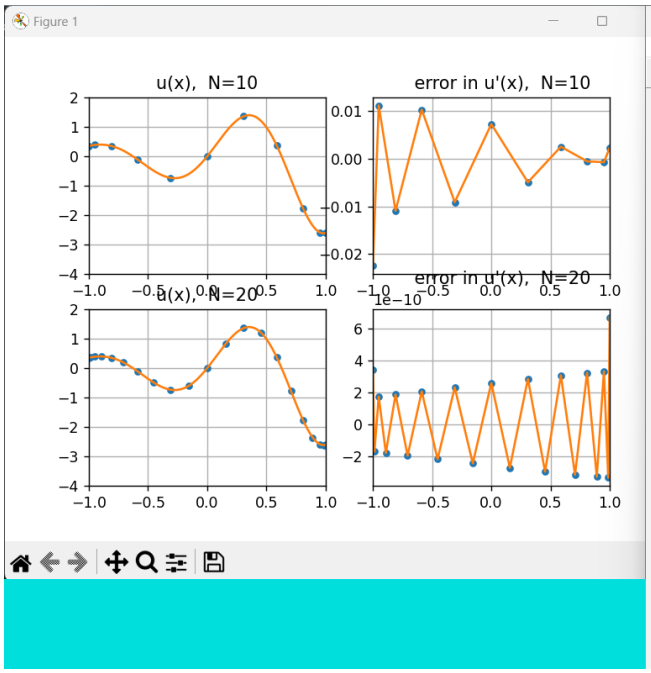
```
# p9.py - polynomial interpolation in equispaced and Chebyshev pts
# Python translation 12/17/12
from numpy import *
from pylab import *
N = 16
xx = arange(-1.01,1.015,.005) #-1.01:.005:1.01
for i in [1,2]:
        if i==1: s = 'equispaced points'; x = -1. + 2.*arange(0,N+1)/N
        if i==2: s =  'Chebyshev points'; x = cos(pi*arange(0,N+1)/N)
        subplot(1,2,i)
        u  = 1/(1+16*x**2 )
        uu = 1/(1+16*xx**2)
        p = polyfit(x,u,N)              # interpolation
        pp = polyval(p,xx)              # evaluation of interpolant
        plot(x,u,'.',markersize=8)
        plot(xx,pp)
        xlim(-1.1,1.1); ylim(-1,1.5); title(s)
        error = linalg.norm(uu-pp,inf)
        text(-.5,-.5,'max error = '+str('%.4f' % error) )
show()
```




p10.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

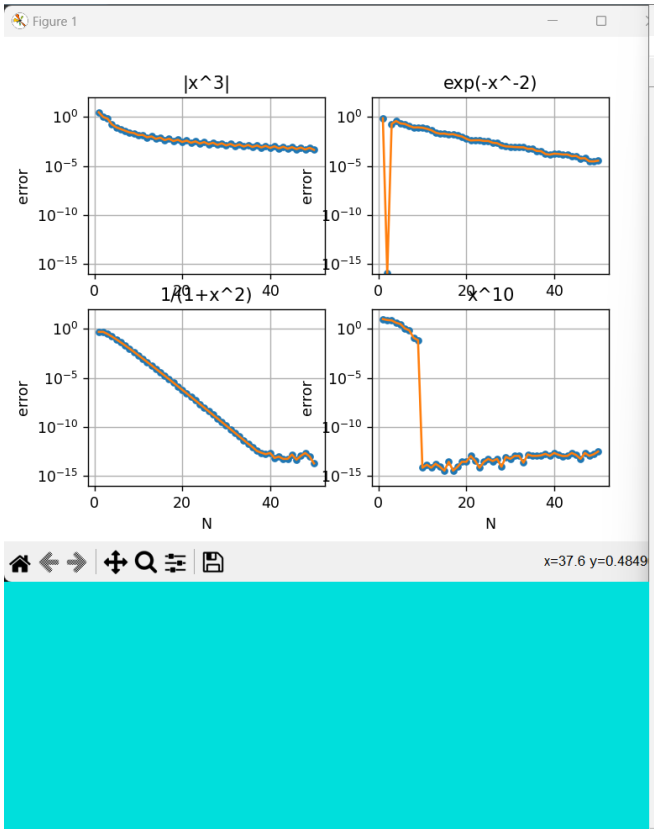
```
# p10.py - polynomials and corresponding equipotential curves
#          Translation started 12/20/12
from numpy import *
from pylab import *
N = 16
for i in [0,1]:
        if i==0: s = 'equispaced points'; x = linspace(-1.,1.,N+1)
        if i==1: s = 'Chebyshev points' ; x = cos(linspace(0,pi,N+1))
        p = poly(x)
        # Plot p(x) over [-1,1]:
        xx = arange(-1.,1.005,.005); pp = polyval(p,xx)
        subplot(2,2,int(2*i+1))
        plot(x,0*x,'.',markersize=12)
        plot(xx,pp,linewidth=2); grid(); xlim(-1.,1.)
        xticks([-1.,-.5,0,.5,1]); title(s)
        # Plot equipotential curves:
        subplot(2,2,int(2*i+2))
        plot(real(x),imag(x),'.',markersize=12)
        xgrid = arange(-1.4,1.42,.02); ygrid = arange(-1.12,1.14,.02)
        xx,yy = meshgrid(xgrid,ygrid); zz = xx+1j*yy
        pp = polyval(p,zz); levels = 10.**arange(-4,1)
        contour(xx,yy,abs(pp),levels,colors=['m']); title(s)
show()
```




p11.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

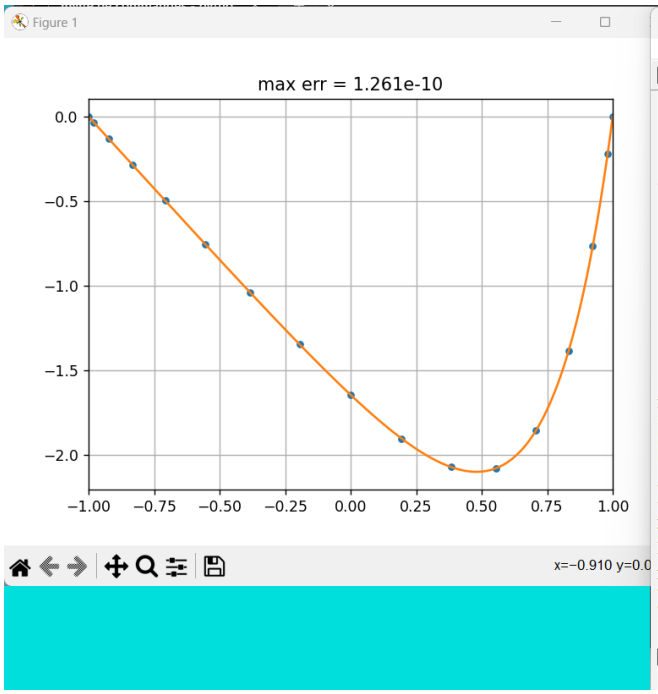
```
# p11.py - Chebyshev differentation of a smooth function
#          Python/NumPy translation by JR 12/22/12
from numpy import *
from pylab import *

def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1), [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)

xx = arange(-1,1.01,.01); uu = exp(xx)*sin(5*xx)
for N in [10, 20]:
        D,x = cheb(N); u = exp(x)*sin(5*x)
        subplot(2,2,2*(N==20)+1)
        plot(x,u,'.',markersize=8); grid()
        plot(xx,uu)
        xlim(-1,1); ylim(-4,2); title('u(x),  N='+str(N))
        error = dot(D,u) - exp(x)*(sin(5*x)+5*cos(5*x))
        subplot(2,2,2*(N==20)+2)
        plot(x,error,'.',markersize=8); grid()
        plot(x,error)
        xlim(-1,1); title("    error in u'(x),  N="+str(N))
show()
```




p12.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

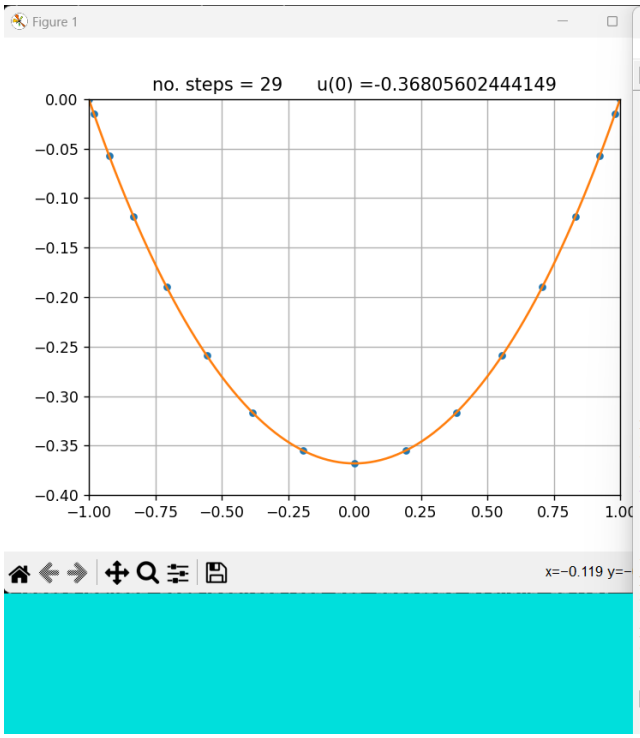
```
# p12.py - accuracy of Chebyshev spectral differentiation
#          (compare p7.py)
#          Python/NumPy translation 12/22/12
from numpy import *
from pylab import *

def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1), [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)

# Compute derivatives for various values of N:
Nmax = 50; E = zeros((4,Nmax))
for N in range(1,Nmax+1):
        D,x = cheb(N)
        v = abs(x)**3;      vprime = 3*x*abs(x) # 3rd deriv in BV
        E[0,N-1] = linalg.norm(dot(D,v)-vprime,inf)
        v = exp(-x**(-2)); vprime = 2*v/x**3   # C-infinity
        E[1,N-1] = linalg.norm(dot(D,v)-vprime,inf)
        v = 1/(1+x**2);     vprime = -2*x*v**2  # analytic in [-1,1]
        E[2,N-1] = linalg.norm(dot(D,v)-vprime,inf)
        v = x**10;          vprime = 10*x**9    # polynomial
        E[3,N-1] = linalg.norm(dot(D,v)-vprime,inf)

# Plot results:
titles = ['|x^3|','exp(-x^-2)','1/(1+x^2)','x^10']
for iplot in range(4):
        subplot(2,2,iplot+1)
        semilogy(range(1,Nmax+1),E[iplot,:],'.',markersize=8)
        plot(range(1,Nmax+1),E[iplot,:])
        ylim(1e-16,100); grid()
        yticks(10.**arange(-15,5,5))
        if iplot>1: xlabel('N');
        ylabel('error'); title(titles[iplot])
show()
```




p13.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

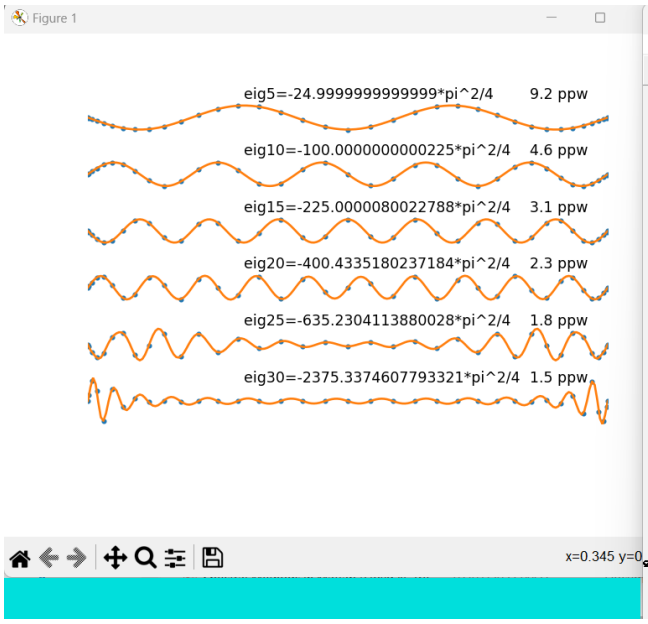
```
# p13.py - solve linear BVP u_xx = exp(4x), u(-1)=u(1)=0
#          Python/NumPy translation 12/23/12
from numpy import *
from pylab import *

def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1), [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)

N = 16
D,x = cheb(N)
D2 = dot(D,D)
D2 = D2[1:-1,1:-1]                # boundary conditions
f = exp(4*x[1:-1])
u = linalg.solve(D2,f)            # Poisson eq. solved here
u = hstack(([0.],u,[0.]))
plot(x,u,'.',markersize=8)
xx = arange(-1,1.01,.01)
uu = polyval(polyfit(x,u,N),xx)   # interpolate grid data
plot(xx,uu)
grid(); xlim(-1,1)
exact = ( exp(4*xx) - sinh(4)*xx - cosh(4) )/16
title('max err = '+str('%.3e' % norm(uu-exact,inf)),fontsize=12)
show()
```

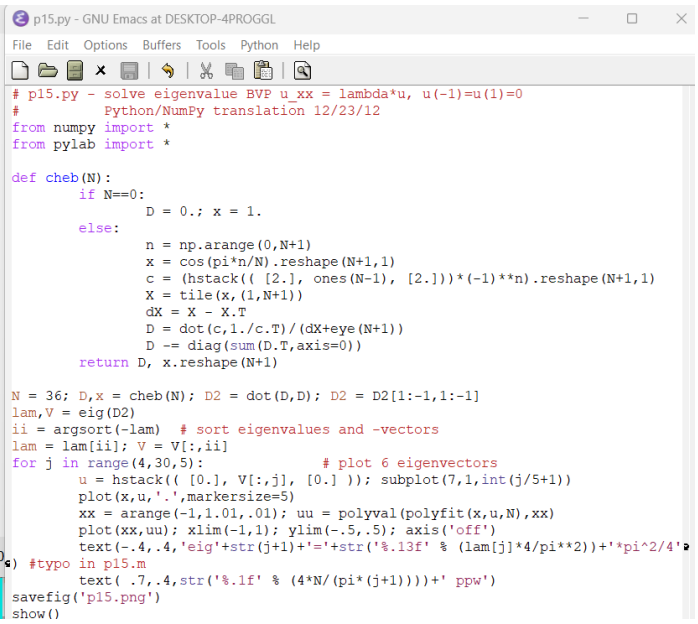



p14.py - GNU Emacs at DESKTOP-4PROGGL

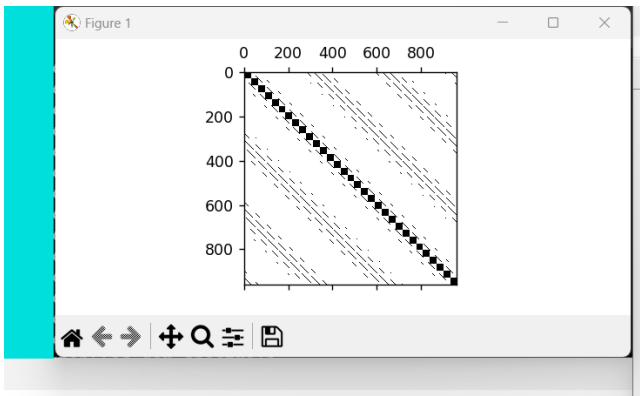
File Edit Options Buffers Tools Python Help

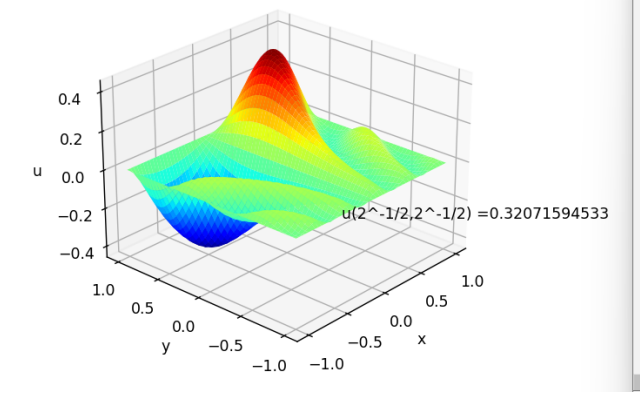
```
# p14.py - solve nonlinear BVP u_xx = exp(u), u(-1)=u(1)=0
#          (compare p13.py)
#          Python/NumPy translation 12/23/12
from numpy import *
from pylab import *

def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1), [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)

N = 16
D,x = cheb(N); D2 = dot(D,D); D2 = D2[1:-1,1:-1]
u = zeros(N-1)
change = 1.; it = 0
while change > 1e-15:   # fixed-point iteration
        unew = linalg.solve(D2,exp(u))
        change = linalg.norm(unew-u,inf)
        u = unew; it += 1
u = hstack(([0.],u,[0.]))
plot(x,u,'.',markersize=8)
xx = arange(-1,1.01,.01)
uu = polyval(polyfit(x,u,N),xx)    # interpolate grid data
plot(xx,uu)
grid(); xlim(-1,1); ylim(-0.4,0)
title('no. steps = '+str(it)+'      u(0) ='+str('%.14f' % u[N//2]) )
show()
```

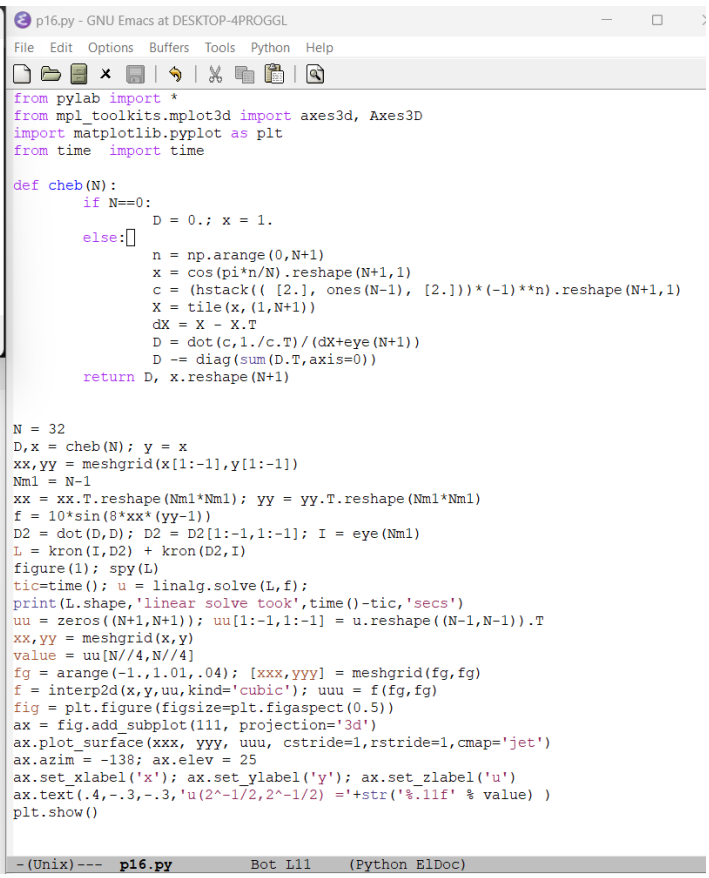


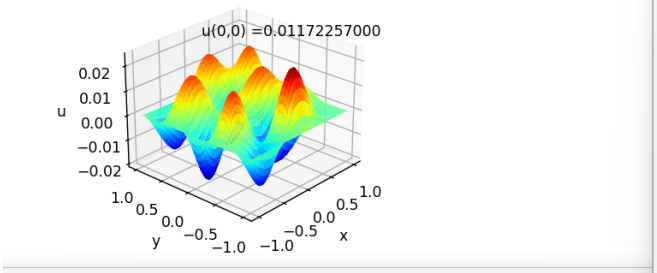

```
# p15.py - solve eigenvalue BVP u_xx = lambda*u, u(-1)=u(1)=0
#          Python/NumPy translation 12/23/12
from numpy import *
from pylab import *

def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1), [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)

N = 36; D,x = cheb(N); D2 = dot(D,D); D2 = D2[1:-1,1:-1]
lam,V = eig(D2)
ii = argsort(-lam)  # sort eigenvalues and -vectors
lam = lam[ii]; V = V[:,ii]
for j in range(4,30,5):              # plot 6 eigenvectors
        u = hstack(( [0.], V[:,j], [0.] )); subplot(7,1,int(j/5+1))
        plot(x,u,'.',markersize=5)
        xx = arange(-1,1.01,.01); uu = polyval(polyfit(x,u,N),xx)
        plot(xx,uu); xlim(-1,1); ylim(-.5,.5); axis('off')
        text(-.4,.4,'eig'+str(j+1)+'='+str('%.13f' % (lam[j]*4/pi**2))+'*pi^2/4'
) #typo in p15.m
        text( .7,.4,str('%.1f' % (4*N/(pi*(j+1))))+' ppw')
savefig('p15.png')
show()
```

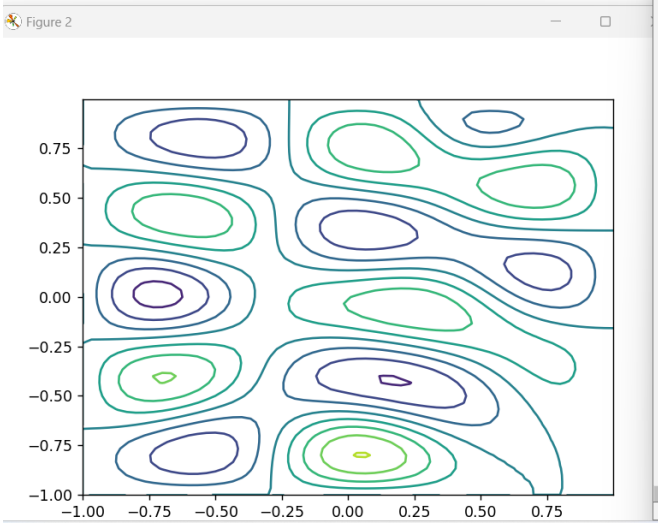



```
from pylab import *
from mpl_toolkits.mplot3d import axes3d, Axes3D
import matplotlib.pyplot as plt
from time  import time

def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1), [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)


N = 32
D,x = cheb(N); y = x
xx,yy = meshgrid(x[1:-1],y[1:-1])
Nm1 = N-1
xx = xx.T.reshape(Nm1*Nm1); yy = yy.T.reshape(Nm1*Nm1)
f = 10*sin(8*xx*(yy-1))
D2 = dot(D,D); D2 = D2[1:-1,1:-1]; I = eye(Nm1)
L = kron(I,D2) + kron(D2,I)
figure(1); spy(L)
tic=time(); u = linalg.solve(L,f);
print(L.shape,'linear solve took',time()-tic,'secs')
uu = zeros((N+1,N+1)); uu[1:-1,1:-1] = u.reshape((N-1,N-1)).T
xx,yy = meshgrid(x,y)
value = uu[N//4,N//4]
fg = arange(-1.,1.01,.04); [xxx,yyy] = meshgrid(fg,fg)
f = interp2d(x,y,uu,kind='cubic'); uuu = f(fg,fg)
fig = plt.figure(figsize=plt.figaspect(0.5))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(xxx, yyy, uuu, cstride=1,rstride=1,cmap='jet')
ax.azim = -138; ax.elev = 25
ax.set_xlabel('x'); ax.set_ylabel('y'); ax.set_zlabel('u')
ax.text(.4,-.3,-.3,'u(2^-1/2,2^-1/2) ='+str('%.11f' % value) )
plt.show()
```



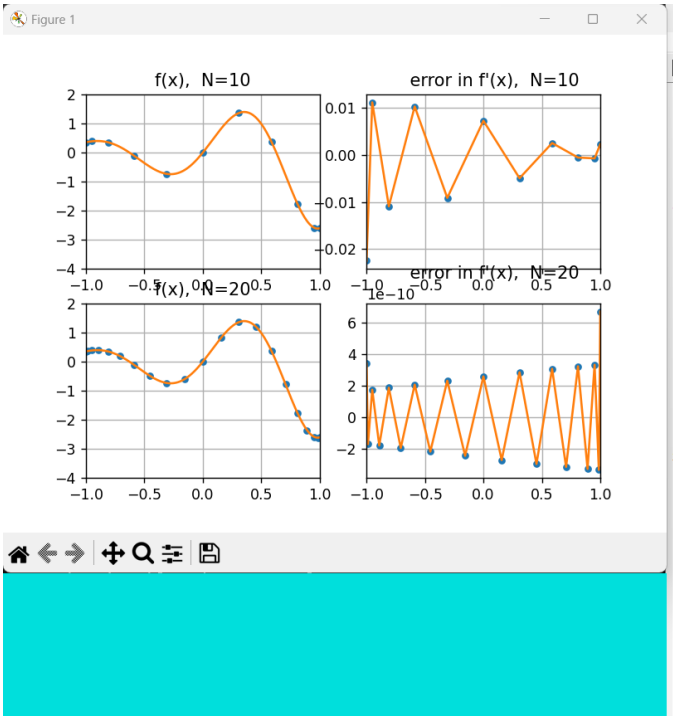

p17.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

```
# p17.py - Helmholtz eq. u_xx + u_yy + (k^2)u = f
#          on [-1,1]x[-1,1]    (compare p16.py)
#          Python translation 12/25/12
from numpy import *
from scipy.interpolate import interp2d
from pylab import *

def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1), [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)

from mpl_toolkits.mplot3d import axes3d, Axes3D
import matplotlib.pyplot as plt

# Set up spectral grid and tensor product Helmholtz operator:
N = 24; D,x = cheb(N); y = x
xx,yy = meshgrid(x[1:-1],y[1:-1])
Nm1 = N-1; xx = xx.T.reshape(Nm1*Nm1); yy = yy.T.reshape(Nm1*Nm1)
f = exp(-10*((yy-1)**2+(xx-.5)**2));
D2 = dot(D,D); D2 = D2[1:-1,1:-1]; I = eye(Nm1)
k = 9.
L = kron(I,D2) + kron(D2,I) + k**2*eye(Nm1*Nm1)
# Solve for u, reshape to 2D grid, and plot:
u = linalg.solve(L,f)
uu = zeros((N+1,N+1)); uu[1:-1,1:-1] = u.reshape((N-1,N-1)).T
xx,yy = meshgrid(x,y)
fg = arange(-1.,1.001,.0333); [xxx,yyy] = meshgrid(fg,fg)
f = interp2d(x,y,uu,kind='cubic'); uuu = f(fg,fg)
fig = plt.figure(figsize=plt.figaspect(0.5))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(xxx, yyy, uuu, cstride=1,rstride=1,cmap='jet')
ax.azim = -138; ax.elev = 25
ax.set_xlabel('x'); ax.set_ylabel('y'); ax.set_zlabel('u')
ax.text(.2,1,.022,'u(0,0) ='+str('%.11f' % uu[N//2,N//2]) )
figure(2);
```

-(Unix)--- **p17.py** Top L20 (Python ElDoc)

Wrote c:/Users/DENISE_2022/Desktop/les-der-spectral/p17.py

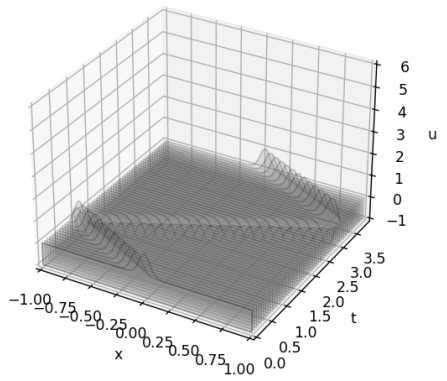



p18.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

```
# p18.py - Chebyshev differentiation via FFT (compare p11.py)
#          Python/NumPy translation 12/26/12
from numpy import *
from numpy.fft import fft,ifft
from pylab import *


def chebfft(v):
        N = len(v)-1;
        if N==0: return 0
        x = cos(arange(0,N+1)*pi/N)
        ii = arange(0,N); iir = arange(1-N,0); iii = array(ii,dtype=int)
        #v = v[:]
        V = hstack((v,v[N-1:0:-1])) # transform x -> theta
        U = real(fft(V))
        W = real(ifft(1j*hstack((ii,[0.],iir))*U))
        w = zeros(N+1)
        w[1:N] = -W[1:N]/sqrt(1-x[1:N]**2)     # transform theta -> x
        w[0] = sum(iii**2*U[iii])/N + .5*N*U[N]
        w[N] = sum((-1)**(iii+1)*ii**2*U[iii])/N + .5*(-1)**(N+1)*N*U[N]
        return w

xx = arange(-1,1.01,.01); ff = exp(xx)*sin(5*xx)
for N in [10, 20]:
        x = cos(pi*arange(0,N+1)/N); f = exp(x)*sin(5*x)
        subplot(2,2,2*(N==20)+1)
        plot(x,f,'.',markersize=8); grid()
        plot(xx,ff)
        xlim(-1,1); ylim(-4,2); title('f(x),  N='+str(N))
        error = chebfft(f) - exp(x)*(sin(5*x)+5*cos(5*x))
        subplot(2,2,2*(N==20)+2)
        plot(x,error,'.',markersize=8); grid()
        plot(x,error)
        xlim(-1,1); title("    error in f'(x),  N="+str(N))
show()
```

Figure 1

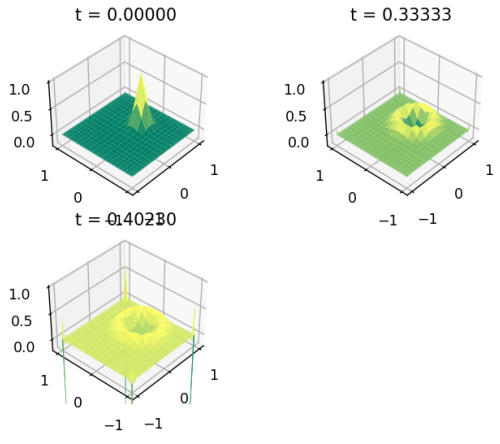



p19.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

```
from mpl_toolkits import mplot3d
from matplotlib.collections import PolyCollection

def chebfft(v):
        N = len(v)-1 ; if N==0: return 0
        x = cos(arange(0,N+1)*pi/N)
        ii = arange(0,N); iir = arange(1-N,0); iii = array(ii,dtype=int)
        V = hstack((v,v[N-1:0:-1])) # transform x -> theta
        U = real(fft(V)) ; W = real(ifft(1j*hstack((ii,[0.],iir))*U))
        w = zeros(N+1)
        w[1:N] = -W[1:N]/sqrt(1-x[1:N]**2)     # transform theta -> x
        w[0] = sum(iii**2*U[iii])/N + .5*N*U[N]
        w[N] = sum((-1)**(iii+1)*ii**2*U[iii])/N + .5*(-1)**(N+1)*N*U[N]
        return w

def waterfall(x,t,u,labels=['x','t','u'],slabthickness=0.5,zrange=[-1.,6.]):
        fig = plt.figure()
        ax = plt.axes(projection='3d')
        cc = lambda arg: matplotlib.colors.to_rgba(arg, alpha=0.7)
        xs = np.hstack(( x,x[-1],x[0] )) ; verts = [] ; zs = t
        baseline  = 0.-slabthickness
        for ti,z in enumerate(zs):
                ys = np.hstack(( u[ti,:],baseline,baseline ))
                verts.append(list(zip(xs, ys)))
        poly = PolyCollection(verts, edgecolors = 'black', facecolors = [cc('gra
'y')], alpha=.6) ; poly.set_alpha(0.2)
        ax.add_collection3d(poly, zs=zs, zdir='y')
        ax.set_xlabel(labels[0]) ; ax.set_xlim3d(min(xs),max(xs))
        ax.set_ylabel(labels[1]) ; ax.set_ylim3d(min(zs),max(zs))
        ax.set_zlabel(labels[2]) ; ax.set_zlim3d(zrange[0],zrange[1])

N = 80; x = cos(pi*arange(0,N+1)/N); dt = 8./N**2
v = exp(-200*x**2); vold = exp(-200*(x-dt)**2)
tmax = 4.; tplot = .075;
plotgap = int(round(tplot/dt)); dt = tplot/plotgap
nplots  = int(round(tmax/tplot))
plotdata = vstack(( v, zeros((nplots,N+1)) )); tdata = [0.]
for i in range(nplots):
        for n in range(plotgap):
                w = chebfft(chebfft(v)); w[0] = 0; w[N] = 0
                vnew = 2*v - vold + dt**2*w; vold = v.copy(); v = vnew
        plotdata[i+1,:] = v; tdata.append(dt*i*plotgap)
waterfall(x,tdata,plotdata,slabthickness=1.)
plt.show()
```

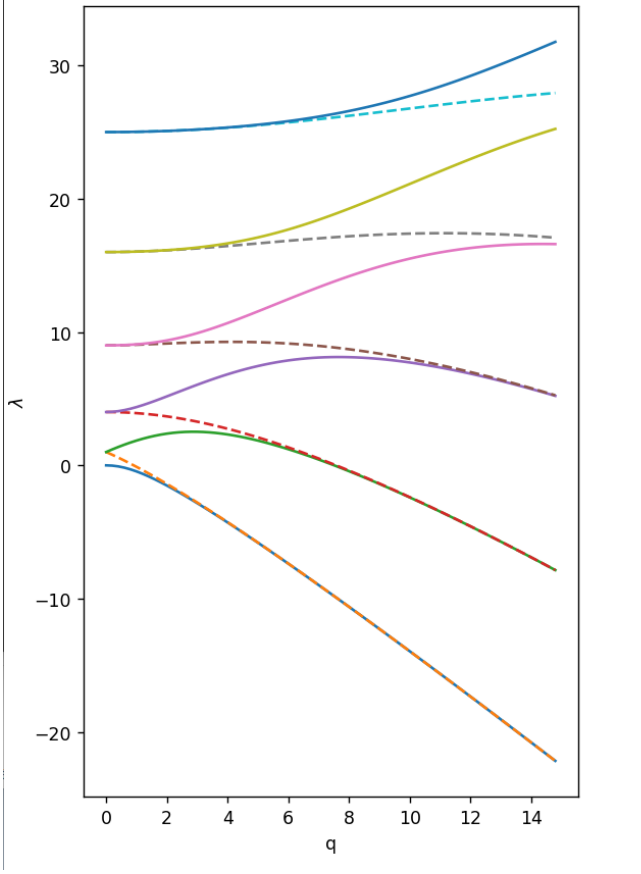
Figure 1




p20u.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

```
set_printoptions(precision=4,linewidth=200)
N = 24; x = cos(pi*arange(0,N+1)/N); y = x; dt = 6.6/N**2
[xx,yy] = meshgrid(x,y)
plotgap = int(round(1./3./dt)); dt = 1./3./plotgap
vv = exp(-40*((xx-.4)**2 + yy**2))
vvold = vv.copy(); vvnew=0*vv
[ay,ax] = meshgrid([.56, .06],[.1, .55])
for n in array(range(3*plotgap+1)):
        t = n*dt
        if (n+.5)%plotgap < 1:      # plots at multiples of t=1/3
                iplot = n/plotgap
                ax = subplot(2,2,int(iplot+1),projection='3d')
                xxxrange = arange(-1,1.01,.05)
                [xxx,yyy] = meshgrid(xxxrange,xxxrange)
                f = interp2d(x,y,vv,kind='cubic');
                vvv = f(xxxrange,xxxrange)
                ax.plot_surface(xx, yy, vv, cstride=1,rstride=1,cmap='summer')
                title('t = '+str('%.5f' % t)); ax.azim = -138; ax.elev = 38; ax.s
'set_zlim3d(-0.2,1)
        uxx = zeros((N+1,N+1)); uyy = zeros_like(uxx)
        ii = range(1,N); iip = range(0,N); iim = range(1-N,0)
        for i in range(1,N):
                v = vv[i,:]; V = hstack(( v, v[N-1:0:-1] ))
                U = real(fft(V))
                W1 = real(ifft(1j*hstack((iip,[0.],iim))*U)) # diff wrt theta
                W2 = real(ifft(-hstack(( range(0,N+1), iim ))**2.*U))
                uxx[i,ii] = W2[ii]/(1-x[ii]**2) - x[ii]* \
                                W1[ii]/(1-x[ii]**2)**(3./2.)
        for j in range(1,N):
                v = vv[:,j]; V = hstack(( v, v[N-1:0:-1] ))
                U = real(fft(V))
                W1 = real(ifft(1j*hstack((iip,[0.],iim))*U)) # diff wrt theta
                W2 = real(ifft(-hstack(( range(0,N+1), iim ))**2.*U))    # diff^
'2 wrt theta
                uyy[ii,j] = W2[ii]/(1-y[ii]**2) - y[ii]* \
                                W1[ii]/(1-y[ii]**2)**(3./2.)
        vvnew = 2*vv - vvold + dt**2*(uxx+uyy)
        vvold = vv.copy(); vv = vvnew
        if max(abs(vv.ravel()  ))>2: break   # ravel is just a new view
ax = subplot(2,2,int(iplot+2),projection='3d')
ax.plot_surface(xx, yy, vv, cstride=1,rstride=1,cmap='summer')
title('t = '+str('%.5f' % t)); ax.azim = -138; ax.elev = 38; ax.set_zlim3d(-0.2,
'1)
show()
```

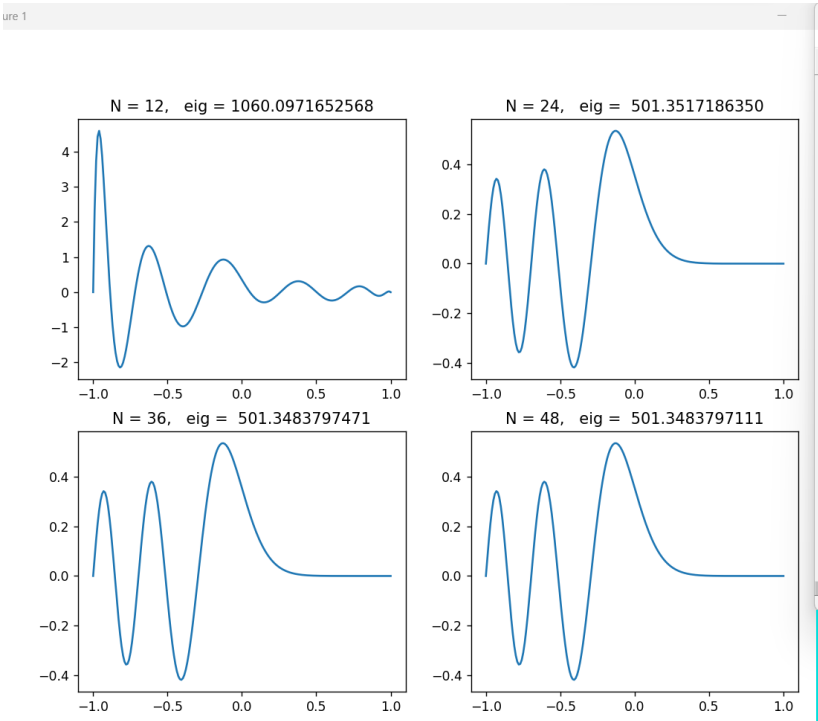



p21.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

```
import matplotlib.pyplot as plt
from numpy import pi,arange,sin,cos,zeros,diag,sort,real
from scipy.linalg import toeplitz
from numpy.linalg import eig
from itertools import cycle
from matplotlib.pyplot import figure,plot,xlabel,ylabel

N = 42; h = 2.0*pi/N; x = h*arange(1,N+1)
col = zeros(N)
col[0] = -pi**2/(3.0*h**2) - 1.0/6.0
col[1:] = -0.5*(-1.0)**arange(1,N)/sin(0.5*h*arange(1,N))**2
D2 = toeplitz(col)

ne = 11 # number of eigenvalues to plot
qq = arange(0.0, 15.0, 0.2)
data= zeros((len(qq),ne))
i = 0
for q in qq:
    evals,evecs = eig(-D2 + 2.0*q*diag(cos(2.0*x)))
    e = real(sort(evals))
    data[i,:] = e[0:ne]
    i = i + 1

figure(figsize=(5,10))
lines=cycle(["-","--"])
for i in range(ne):
    plot(qq,data[:,i],next(lines))
xlabel("q")
ylabel("$\lambda$");
plt.show()
```

-\---  **p21.py**  All L17  (Python ElDoc)

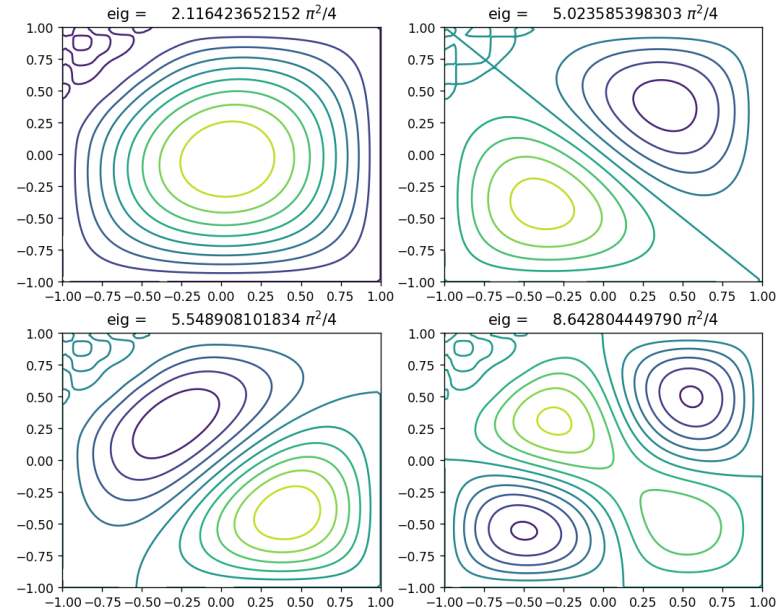



p22.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

```
import matplotlib.pyplot as plt
import numpy as np
from numpy import *
from scipy.linalg import eig
from scipy.special import airy
from matplotlib.pyplot import figure,subplot,plot,title

def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1),
                              [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)

figure(figsize=(10,8))
for N in range(12,60,12):
    D,x = cheb(N); D2 = dot(D,D); D2 = D2[1:N,1:N]
    Lam,V = eig(D2,diag(x[1:N]))
    Lam = real(Lam); ii = where(Lam>0)[0]
    V = real(V[:,ii]); Lam = Lam[ii]
    ii = argsort(Lam); ii=ii[4]; Lam=Lam[ii]
    v = zeros(N+1); v[1:N] = V[:,ii]; v = v/v[N//2]*airy(0.0)[0]
    xx = linspace(-1.0,1.0,200); vv = polyval(polyfit(x,v,N),xx);
    subplot(2,2,N//12); plot(xx,vv)
    title("N = %d,   eig = %15.10f"%(N,Lam));
plt.show()
```

-**- **p22.py** All L15 (Python ElDoc)

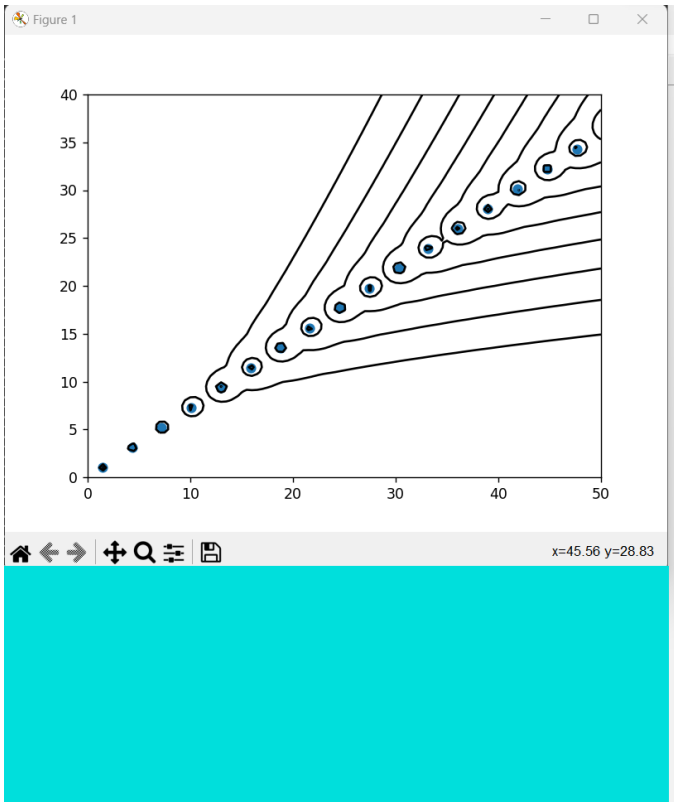



p23.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

```
import matplotlib.pyplot as plt
import numpy as np
from numpy import *
from matplotlib.pyplot import figure,subplot,plot,title,contour
from scipy.linalg import eig,norm
from scipy.interpolate import interp2d

def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1),
                              [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)

# Set up tensor product Laplacian and compute 4 eigenmodes
N = 16; D,x = cheb(N); y = x;
xx,yy = meshgrid(x[1:N],y[1:N])
xx = reshape(xx,(N-1)**2)
yy = reshape(yy,(N-1)**2)
D2 = dot(D,D); D2 = D2[1:N,1:N]; I = eye(N-1)
L = -kron(I,D2) - kron(D2,I)
L = L + diag(exp(20*(yy-xx-1)))
D,V = eig(L); D = real(D); V = real(V)
ii = argsort(D); ii = ii[0:4]; D = D[ii]; V = V[:,ii]
# Reshape them to 2D grid, interpolate to finer grid, and plot
fine = linspace(-1.0,1.0,100,True);
uu = zeros((N+1,N+1));
figure(figsize=(10,10))
for i in range(4):
    uu[1:N,1:N] = reshape(V[:,i],(N-1,N-1))
    uu = uu/norm(uu,inf)
    f = interp2d(x,y,uu,kind='cubic')
    uuu = f(fine,fine)
    subplot(2,2,i+1)
    contour(fine,fine,uuu,10)
    title("eig = %18.12f $\pi^2/4$"%(D[i]/(pi**2/4)))
plt.show()
```

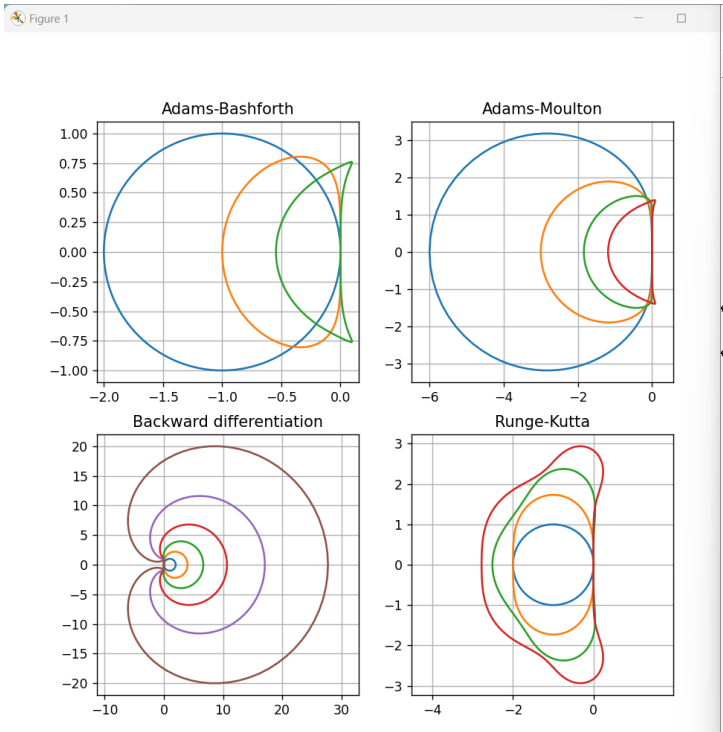
-**- **p23.py** Top L15 (Python ElDoc)




p24.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

```
import matplotlib.pyplot as plt
import numpy as np
from numpy import *
from scipy.linalg import solve,eig,svd,svdvals
from matplotlib.pyplot import figure,plot,title,axis,contour

def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1), [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)

N = 70; D, x = cheb(N); x = x[1:N];
L = 6.0; x = L*x; D = D/L;
A = -dot(D,D);
A = A[1:N,1:N] + (1+3j)*diag(x**2);
lam, v = eig(A)
fig = figure()
plot(real(lam),imag(lam),"o")
axis([0, 50, 0, 40])

h = 0.5 #Smaller the value, finer the plot
x = arange(0,50+h,h); y = arange(0,40+h,h); xx,yy = meshgrid(x,y);
zz = xx + 1j*yy;
I = eye(N-1); sigmin = zeros((len(y),len(x)))
for j in range(0,len(x)):
    for i in range(0,len(y)):
        sigmin[i,j] = min(svdvals(zz[i,j]*I - A));

levels = 10.0**arange(-4.5,0.0,0.5);
contour(x,y,sigmin,levels,colors = 'k');
plt.show()
```




p25.py - GNU Emacs at DESKTOP-4PROGGL

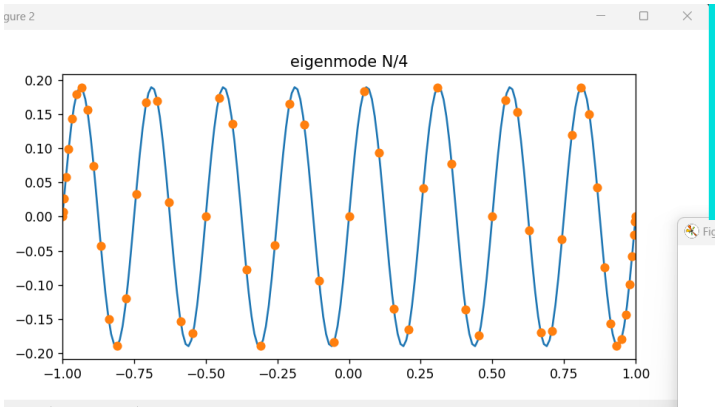
File Edit Options Buffers Tools Python Help

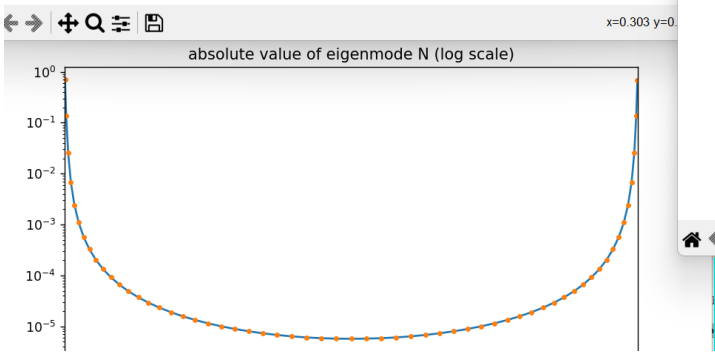
```
import matplotlib.pyplot as plt ; from matplotlib.pyplot import *
from numpy import pi,real,imag,zeros,exp,arange
figure(figsize=(8,8))

subplot(2,2,1)
z = exp(1j*pi*arange(0,201)/100); r = z - 1
s = 1; rr = r/s; plot(real(rr),imag(rr))
s = (3 - 1/z)/2; rr = r/s; plot(real(rr),imag(rr))
s = (23 - 16/z + 5/z**2)/12; rr = r/s; plot(real(rr),imag(rr))
axis('equal'); grid('on') ; title('Adams-Bashforth')

subplot(2,2,2)
s = (5*z + 8 - 1/z)/12; rr = r/s; plot(real(rr),imag(rr))
s = (9*z + 19 - 5/z + 1/z**2)/24; rr = r/s; plot(real(rr),imag(rr))
s = (251*z + 646 - 264/z + 106/z**2 - 19/z**3)/720; rr = r/s; plot(real(rr),imag(rr))
d = 1 - 1/z
s = 1 - d/2 - d**2/12 - d**3/24 - 19*d**4/720 - 3*d**5/160; dd = d/s; plot(real(dd),imag(dd))
axis('equal'); grid('on') ; title('Adams-Moulton')

subplot(2,2,3)
r = 0
for i in range(1,7):
    r = r + d**i/i; plot(real(r),imag(r))
axis('equal'); grid('on') ; title('Backward differentiation')

subplot(2,2,4)
w = 0; W = 1j*zeros(len(z)); W[0] = w;
for i in range(1,len(z)): w = w - (1+w-z[i]); W[i] = w
plot(real(W),imag(W))
w = 0; W = 1j*zeros(len(z)); W[0] = w;
for i in range(1,len(z)): w = w - (1+w+0.5*w**2-z[i]**2)/(1+w); W[i] = w
plot(real(W),imag(W))
w = 0; W = 1j*zeros(len(z)); W[0] = w;
for i in range(1,len(z)):
    w = w - (1+w+0.5*w**2+w**3/6-z[i]**3)/(1+w+0.5*w**2); W[i] = w
plot(real(W),imag(W))
w = 0; W = 1j*zeros(len(z)); W[0] = w;
for i in range(1,len(z)):
    w = w - (1+w+0.5*w**2+w**3/6+w**4/24-z[i]**4)/(1+w+w**2/2+w**3/6); W[i] = w
plot(real(W),imag(W))
axis('equal'); grid('on') ; title('Runge-Kutta');
plt.show()
```

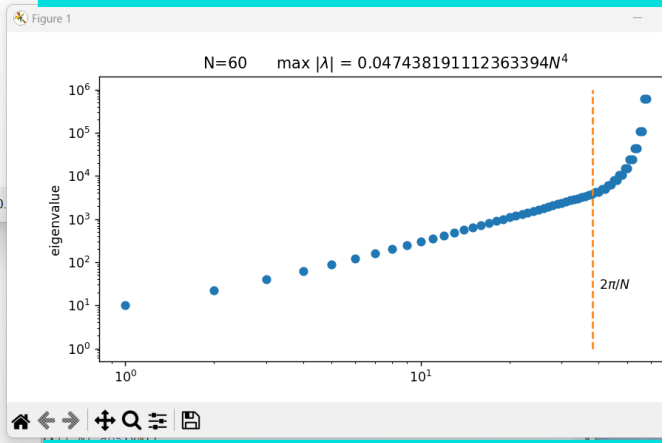
p26.py - GNU Emacs at DESKTOP-4PROGGL

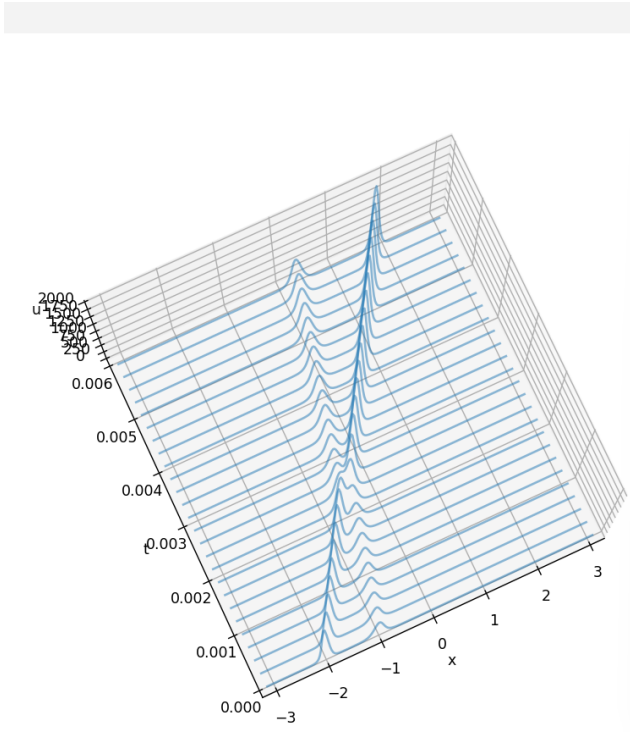
File Edit Options Buffers Tools Python Help

```
def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1), [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)

N = 60; D, x = cheb(N); D2 = dot(D,D); D2 = D2[1:N,1:N]
Lam, V = eig(D2)
ii = argsort(-Lam); e = Lam[ii]; V = V[:,ii]

# Plot eigenvalues
figure(figsize=(8,4))
loglog(-e,'o')
semilogy(2*N/pi*array([1,1]),array([1,1e6]),'--')
ylabel('eigenvalue')
title('N='+str(N)+'      max |$\lambda$| = '+str(max(-e)/N**4)+'$N^4$')
text(2.1*N/pi,24,'$2\pi/N$')

# Plot eigenmode N/4 (physical)
figure(figsize=(8,4))
vN4 = zeros(N+1)
vN4[1:N] = V[:,N//4];
xx = arange(-1.0,1.01,0.01)
vv = polyval(polyfit(x,vN4,N),xx)
plot(xx,vv,'-')
plot(x,vN4,'o')
xlim((-1.0,1.0))
title('eigenmode N/4')

# Plot eigenmode N (nonphysical)
figure(figsize=(8,4))
vN = V[:,N-2]
semilogy(x[1:N],abs(vN))
plot(x[1:N],abs(vN),'.')
xlim((-1.0,1.0))
title('absolute value of eigenmode N (log scale)');
plt.show()
```



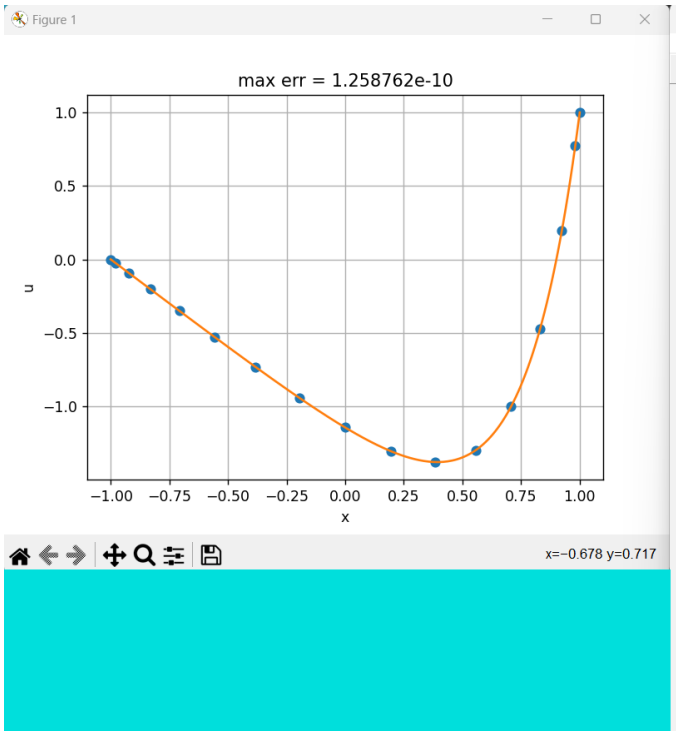




p27.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.collections import LineCollection
from numpy import pi,cosh,exp,round,zeros,arange,real
from numpy.fft import fft,ifft
from matplotlib.pyplot import figure

# Set up grid and differentiation matrix:
N = 256; dt = 0.4/N**2; x = (2*pi/N)*arange(-N/2,N/2);
A, B = 25.0, 16.0
u = 3*A**2/cosh(0.5*A*(x+2))**2 + 3*B**2/cosh(0.5*B*(x+1))**2
v = fft(u);
k = zeros(N); k[0:N//2] = arange(0,N/2); k[N//2+1:] = arange(-N/2+1,0,1)
ik3 = 1j*k**3

# Time-stepping by Runge-Kutta
tmax = 0.006; nplt = int(round((tmax/25)/dt))
nmax = int(round(tmax/dt))
udata = []; udata.append(list(zip(x, u)))
tdata = [0.0]
for n in range(1,nmax+1):
    t = n*dt; g = -0.5j*dt*k
    E = exp(dt*ik3/2); E2 = E**2
    a = g * fft(real(ifft( v          ))**2)
    b = g * fft(real(ifft( E*(v+a/2) ))**2)
    c = g * fft(real(ifft( E*v+b/2   ))**2)
    d = g * fft(real(ifft( E2*v+E*c  ))**2)
    v = E2*v + (E2*a + 2*E*(b+c) + d)/6
    if n%nplt == 0:
        u = real(ifft(v))
        udata.append(list(zip(x, u)))
        tdata.append(t);

fig = figure(figsize=(12,10))
```

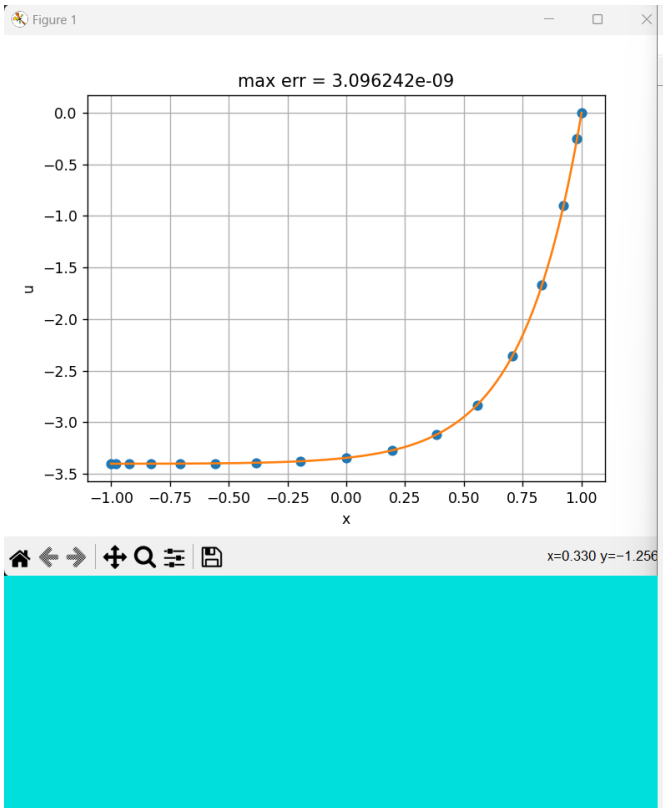
-\--- **p27.py** Top L1 (Python ElDoc)




p32.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

```
import numpy as np
from numpy import *
from numpy.linalg import norm, solve
import matplotlib.pyplot as plt
from matplotlib.pyplot import title,plot,xlabel,ylabel,grid

def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1), [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)

N = 16
D,x = cheb(N)
D2 = dot(D,D)
D2 = D2[1:N,1:N]
f = exp(4.0*x[1:N])
u = solve(D2,f)
s = zeros(N+1)
s[1:N] = u
s = s + (x + 1.0)/2.0 # Correction for bc

xx = linspace(-1.0,1.0,200)
uu = polyval(polyfit(x,s,N),xx)     # interpolate grid data
exact = (exp(4.0*xx) - sinh(4.0)*xx - cosh(4.0))/16.0 + (xx + 1.0)/2.0
maxerr = norm(uu-exact,inf)

title('max err = %e' % maxerr)
plot(x,s,'o',xx,exact)
xlabel('x'); ylabel('u'); grid(True);
plt.show()
```




p33.py - GNU Emacs at DESKTOP-4PROGGL

File Edit Options Buffers Tools Python Help

```
import numpy as np
from numpy import *
from numpy.linalg import norm
from scipy.linalg import solve
import matplotlib.pyplot as plt
from matplotlib.pyplot import *

def cheb(N):
        if N==0:
                D = 0.; x = 1.
        else:
                n = np.arange(0,N+1)
                x = cos(pi*n/N).reshape(N+1,1)
                c = (hstack(( [2.], ones(N-1), [2.]))*(-1)**n).reshape(N+1,1)
                X = tile(x,(1,N+1))
                dX = X - X.T
                D = dot(c,1./c.T)/(dX+eye(N+1))
                D -= diag(sum(D.T,axis=0))
        return D, x.reshape(N+1)

N = 16
# Build matrix
D,x = cheb(N)
D2 = dot(D,D)
D2[N,:] = D[N,:] # Last eqn has neumann bc
D2 = D2[1:N+1,1:N+1]
# RHS
f = zeros(N)
f[0:-1] = exp(4.0*x[1:N])
# Solve
u = solve(D2,f)
s = zeros(N+1)
s[1:N+1] = u
# Compute error
xx = linspace(-1.0,1.0,200)
uu = polyval(polyfit(x,s,N),xx)     # interpolate grid data
exact = (exp(4.0*xx) - 4.0*exp(-4.0)*(xx-1.0) - exp(4.0))/16.0
maxerr = norm(uu-exact,inf)
title('max err = %e' % maxerr)
plot(x,s,'o',xx,exact)
xlabel('x'); ylabel('u'); grid(True);
plt.show()
```