[1]:
```
import numpy as np
from matplotlib import pyplot as plt
from math import sqrt, cos, sin, pi
```

Course: Computational Electrodynamics and Applications

Tutor: Theodoros Samaras

Hand-in date: April 21, 2020

Student: Konstantinos Foutzopoulos < kfoutzop@auth.gr >

0.1 Study of numerical dispersion in 2D FDTD

- The code required for the study is given below. The first method is used purely for studying the numerical dispersion in a uniform grid, whereas the second is the general scheme. In the case where a uniform grid is used the two methods should give the same results.

[2]:
```
def k_in_ug(phi, S, D, eps=10**-6, nm=16):
    '''
    Return k for a uniform grid given (,S,D). [Taflove2000]
    '''
    x = 2*pi
    A, B, C = D*cos(phi)/2, D*sin(phi)/2, (1/S**2)*sin(pi*S/N)**2
    d = (sin(A*x)**2 + sin(B*x)**2 - C)/ \
            (A*sin(2*A*x) + B*sin(2*B*x))
    while abs(d) > eps and nm > 0:
        x -= d
        d = (sin(A*x)**2 + sin(B*x)**2 - C)/ \
                (A*sin(2*A*x) + B*sin(2*B*x))
        nm -= 1

    return x
```

[3]:
```
def k_in_nug(phi, S, Dx, Dy, eps=10**-6, nm=16):
    '''
    Return k for a non-uniform grid given (,S,Dx,Dy). [GedneyEE624]
    For Dx=Dy=D, k is the same found using the "uniform grid" scheme.
    '''
    x = 2*pi
    Dt = S/sqrt((1/Dx)**2+(1/Dy)**2) #CFL
    iDx2, iDy2, iDt2 = 1/Dx**2, 1/Dy**2, 1/Dt**2
    w = 2*pi       # b/c:  = 2c/, c==1
    A, B, C = cos(phi)*Dx/2, sin(phi)*Dy/2, iDt2*sin(w*Dt/2)**2
    d = (iDx2*sin(A*x)**2 + iDy2*sin(B*x)**2 - C)/ \
            (iDx2*A*sin(2*A*x) + iDy2*B*sin(2*B*x))
    while abs(d) > eps and nm > 0:
        x -= d
        d = (iDx2*sin(A*x)**2 + iDy2*sin(B*x)**2 - C)/ \
```

```
                (iDx2*A*sin(2*A*x) + iDy2*B*sin(2*B*x))
        nm -= 1

    return x
```

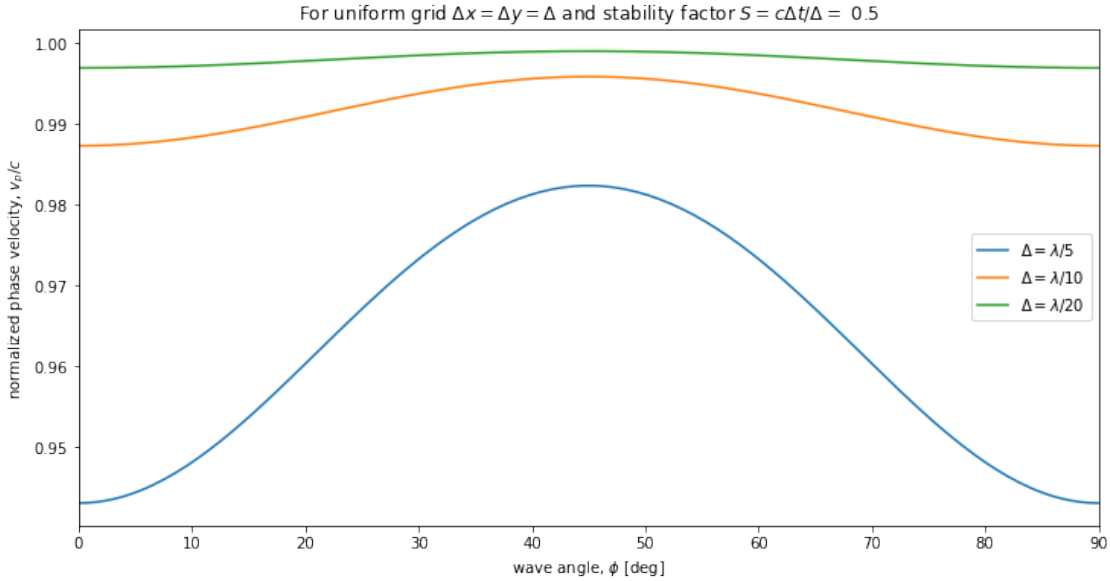
[4]:
```
deg = pi/180
x = np.linspace(0, 90, 180)
```

- We create the requested graph for a uniform grid and different grid steps.

[5]:
```
S = 0.5

plt.figure(figsize=(12, 6))
for N in [5, 10, 20]:  # points/0 (D=0/N)
    y = [2*pi/k_in_ug(phi*deg, S, 1/N) for phi in x]
    plt.plot(x, y, label='$\Delta = \lambda$/{}'.format(N))

plt.legend(loc='best')
plt.xlim(0, 90)
plt.title('For uniform grid $\Delta x = \Delta y = \Delta$ ' +\
              'and stability factor ' +\
              '$S = c\Delta t/\Delta=$ {}'.format(S))
plt.xlabel('wave angle, $\phi$ [deg]')
plt.ylabel('normalized phase velocity, $v_p/c$')
plt.show()
```

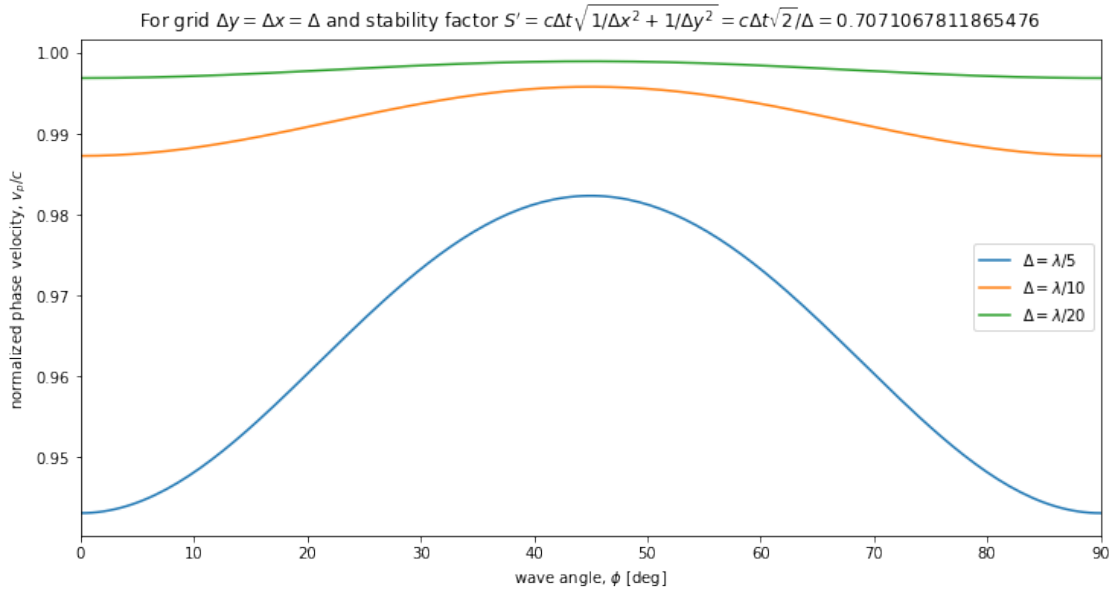


Again the previous, using the generalized scheme.

[6]:
```
S = 0.5*sqrt(2)
#S = 0.9
#S = 0.1

plt.figure(figsize=(12, 6))
for N in [5, 10, 20]:  # points/0 (D=0/N)
    y = [2*pi/k_in_nug(phi*deg, S, 1/N, 1/N) for phi in x]
    plt.plot(x, y, label='$\Delta = \lambda$/{}'.format(N))

plt.legend(loc='best')
plt.xlim(0, 90)
plt.title('For grid $\Delta y = \Delta x = \Delta$ ' +\
              'and stability factor ' +\
              '$S\' = c\Delta t\sqrt{1/\Delta x^2 + 1/\Delta y^2}$' +\
              '$=c\Delta t\sqrt{2}/\Delta=$' + '{}'.format(S))
plt.xlabel('wave angle, $\phi$ [deg]')
plt.ylabel('normalized phase velocity, $v_p/c$')
plt.show()
```



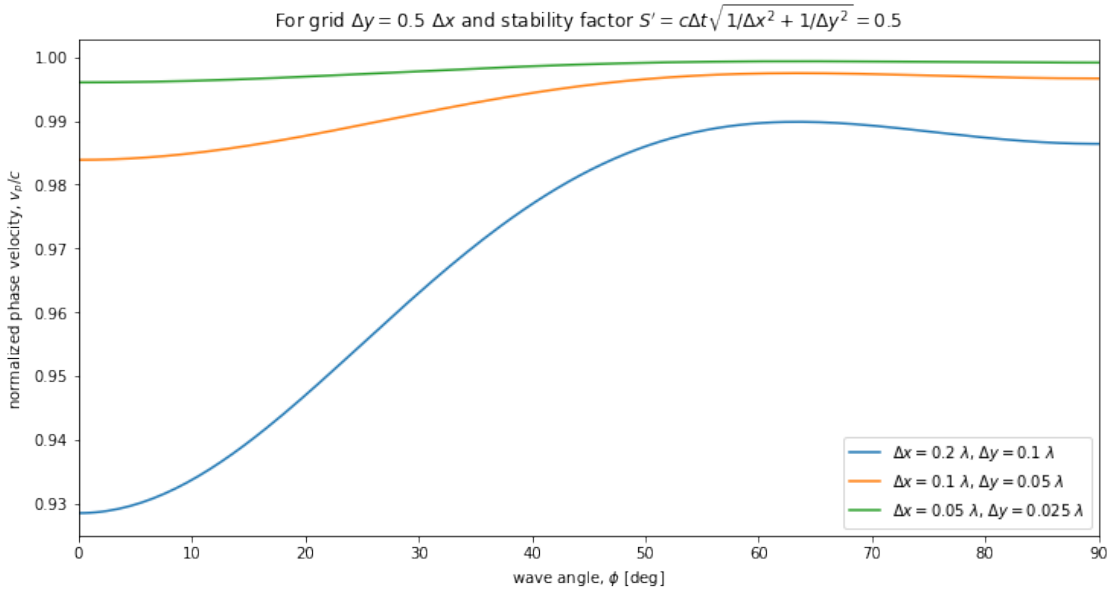
- We create the requested graph for a non-uniform grid and different grid steps.

[7]:
```
S = 0.5
#S = 0.9
a = 0.5

plt.figure(figsize=(12, 6))
for N in [5, 10, 20]:  # points/0 (D=0/N)
```

```
    y = [2*pi/k_in_nug(phi*deg, S, 1/N, a/N) for phi in x]
    plt.plot(x, y, label='$\Delta x = ${} $\lambda$, '.format(1/N) +\
                        '$\Delta y = ${} $\lambda$'.format(a/N))

plt.legend(loc='best')
plt.xlim(0, 90)
plt.title('For grid $\Delta y = ${} $\Delta x$ '.format(a) +\
              'and stability factor ' + \
              '$S\' = c\Delta t\sqrt{1/\Delta x^2 + 1/\Delta y^2}=$' +\
              '{}'.format(S))
plt.xlabel('wave angle, $\phi$ [deg]')
plt.ylabel('normalized phase velocity, $v_p/c$')
plt.show()
```



Bibliography

- [Taflove2000] A. Taflove and S.C. Hagness, Computational Electrodynamics: The Finite-Difference Time-Domain Method, 2nd edition. 2000.
- [GedneyEE624] Gedney Stephen, University of Kentucky, EE624 Notes, Fall 2005, "2-Dimensional FDTD Analysis".