Table of Contents

In [1]:
```
import sympy as sp
import numpy as np
np.set_printoptions(threshold=np.inf) #When we want notebook to display everything
```
executed in 6.27s, finished 16:47:27 2021-09-17

In [2]:
```
import ipyvolume as ipv  # Does not come with default anaconda #Used to show 3D graphs
import ipywidgets as widgets
```
executed in 1.17s, finished 16:47:28 2021-09-17

In [3]:
```
p_x, p_y, p_z, th, phi, = sp.symbols(r'p_x p_y p_z \theta \phi')
```
executed in 19ms, finished 16:47:31 2021-09-17

In [4]:
```
A = sp.Matrix([[1,0,0,p_x],
               [0,1,0,p_y],
               [0,0,1,-p_z],
               [0,0,0,1]])
B = sp.Matrix([[sp.cos(th),0,sp.sin(th),0],
               [0,1,0,0],
               [-sp.sin(th),0,sp.cos(th),0],
               [0,0,0,1]])
C = sp.Matrix([[1,0,0,0],
              [0,sp.cos(phi),-sp.sin(phi),0],
              [0,sp.sin(phi),sp.cos(phi),0],
              [0,0,0,1]])

D = sp.Matrix([[1,0,0,-p_x],
               [0,1,0,-p_y],
               [0,0,1, p_z],
               [0,0,0,1]])
```
executed in 71ms, finished 16:47:31 2021-09-17

In [5]:
```
A*B*C*D
```
executed in 1.43s, finished 16:47:34 2021-09-17

Out[5]:

$$\begin{bmatrix} \cos(\theta) & \sin(\phi)\sin(\theta) & \sin(\theta)\cos(\phi) & -p_x\cos(\theta) + p_x - p_y\sin(\phi)\sin(\theta) + p_z\sin(\theta)\cos(\phi \\ 0 & \cos(\phi) & -\sin(\phi) & -p_y\cos(\phi) + p_y - p_z\sin(\phi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) & p_x\sin(\theta) - p_y\sin(\phi)\cos(\theta) + p_z\cos(\phi)\cos(\theta) - p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In [6]:
```python
#Calculate numbers needed for 2D surface plotter

def GetPoints(x,y):
    '''
    input
    x: np array
        the x positions of the grid points

    y: np array
        the y positions of the grid points

    returns
    np array
        with the position of the points going from bottom left to bottom right, then the

    '''
    mesh = np.meshgrid(x,y)
    x_parts = np.reshape(mesh[0],(1,-1))[0]
    y_parts = np.reshape(mesh[1],(1,-1))[0]
    points = np.array([[x_parts[i],y_parts[i]] for i in range(len(x_parts))])
    return np.reshape(points,(1,-1))[0]

##Tests
actual = GetPoints([0,1,2],[0,1])
expected = np.array([0,0,1,0,2,0,0,1,1,1,2,1])
assert max(abs(actual-expected))==0
```

executed in 20ms, finished 16:47:52 2021-09-17

In [7]:
```
def GetIndices(x, y):
    '''
    input
    x: np array
        the x positions of the grid points

    y: np array
        the y positions of the grid points

    returns
    np array
        with the indices of all the linking points to make 3 vertex shapes

    '''
    w = len(x) #width
    def get_pos(x_pos, y_pos):
        return w*y_pos + x_pos #get the index position

    def get_points(x_pos, y_pos):
        return np.array([get_pos(x_pos,y_pos), get_pos(x_pos+1,y_pos), get_pos(x_pos,y_p
                         get_pos(x_pos+1,y_pos), get_pos(x_pos,y_pos+1), get_pos(x_pos+1

    indices = []
    for y_pos in range(len(y)-1):
        for x_pos in range(len(x)-1):
            indices.append(get_points(x_pos, y_pos))

    return np.reshape(np.array(indices),(1,-1))[0]

##Tests
actual = GetIndices([0,1,2],[0,1])
expected = np.array([0,1,3,
                     1,3,4,
                     1,2,4,
                     2,4,5])
assert max(abs(actual-expected)) == 0
```

executed in 18ms, finished 16:47:56 2021-09-17

In [8]:
```
x = np.linspace(-2,2,101)#41)
y = np.linspace(-2,2,101)#41)
GetPoints(x,y)
```

executed in 367ms, finished 16:47:58 2021-09-17

Out[8]:
```
array([-2.  , -2.  , -1.96, -2.  , -1.92, -2.  , -1.88, -2.  , -1.84,
       -2.  , -1.8 , -2.  , -1.76, -2.  , -1.72, -2.  , -1.68, -2.  ,
       -1.64, -2.  , -1.6 , -2.  , -1.56, -2.  , -1.52, -2.  , -1.48,
       -2.  , -1.44, -2.  , -1.4 , -2.  , -1.36, -2.  , -1.32, -2.  ,
       -1.28, -2.  , -1.24, -2.  , -1.2 , -2.  , -1.16, -2.  , -1.12,
       -2.  , -1.08, -2.  , -1.04, -2.  , -1.  , -2.  , -0.96, -2.  ,
       -0.92, -2.  , -0.88, -2.  , -0.84, -2.  , -0.8 , -2.  , -0.76,
       -2.  , -0.72, -2.  , -0.68, -2.  , -0.64, -2.  , -0.6 , -2.  ,
       -0.56, -2.  , -0.52, -2.  , -0.48, -2.  , -0.44, -2.  , -0.4 ,
       -2.  , -0.36, -2.  , -0.32, -2.  , -0.28, -2.  , -0.24, -2.  ,
       -0.2 , -2.  , -0.16, -2.  , -0.12, -2.  , -0.08, -2.  , -0.04,
       -2.  ,  0.  , -2.  ,  0.04, -2.  ,  0.08, -2.  ,  0.12, -2.  ,
        0.16, -2.  ,  0.2 , -2.  ,  0.24, -2.  ,  0.28, -2.  ,  0.32,
       -2.  ,  0.36, -2.  ,  0.4 , -2.  ,  0.44, -2.  ,  0.48, -2.  ,
        0.52, -2.  ,  0.56, -2.  ,  0.6 , -2.  ,  0.64, -2.  ,  0.68,
       -2.  ,  0.72, -2.  ,  0.76, -2.  ,  0.8 , -2.  ,  0.84, -2.  ,
        0.88, -2.  ,  0.92, -2.  ,  0.96, -2.  ,  1.  , -2.  ,  1.04,
       -2.  ,  1.08, -2.  ,  1.12, -2.  ,  1.16, -2.  ,  1.2 , -2.  ,
        1.24, -2.  ,  1.28, -2.  ,  1.32, -2.  ,  1.36, -2.  ,  1.4 ,
```

In [9]:
```
GetIndices(x,y)
```

executed in 431ms, finished 16:48:02 2021-09-17

```
21,    22,    122,    22,    122,    123,    22,    23,    123,
23,   123,    124,    23,     24,    124,    24,   124,    125,
24,    25,    125,    25,    125,    126,    25,    26,    126,
26,   126,    127,    26,     27,    127,    27,   127,    128,
27,    28,    128,    28,    128,    129,    28,    29,    129,
29,   129,    130,    29,     30,    130,    30,   130,    131,
30,    31,    131,    31,    131,    132,    31,    32,    132,
32,   132,    133,    32,     33,    133,    33,   133,    134,
33,    34,    134,    34,    134,    135,    34,    35,    135,
35,   135,    136,    35,     36,    136,    36,   136,    137,
36,    37,    137,    37,    137,    138,    37,    38,    138,
38,   138,    139,    38,     39,    139,    39,   139,    140,
39,    40,    140,    40,    140,    141,    40,    41,    141,
41,   141,    142,    41,     42,    142,    42,   142,    143,
42,    43,    143,    43,    143,    144,    43,    44,    144,
44,   144,    145,    44,     45,    145,    45,   145,    146,
45,    46,    146,    46,    146,    147,    46,    47,    147,
47,   147,    148,    47,     48,    148,    48,   148,    149,
48,    49,    149,    49,    149,    150,    49,    50,    150,
50,   150,    151,    50,     51,    151,    51,   151,    152,
```

0.0.1 Visualize WebGL Results in Python ¶

In [1]:
```
def read_file(file_name):
    f = open(file_name, "r")
    my_string_data = f.read().splitlines()
    my_Z = []
    for row_string_data in my_string_data:
        row_string_list = row_string_data.split(',')
        my_time_frame_row = []
        for element_index in range(len(row_string_list)-1):
            my_time_frame_row.append(float(row_string_list[element_index]))
        my_Z.append(np.array([my_time_frame_row]))
    return np.array(my_Z)
```

executed in 17ms, finished 13:28:40 2021-09-07

In [2]:
```
#File not available on computing exporting as PDF
Z_Num = read_file("E:\Downloads\Test (17).txt")


Num_xy = len(Z_Num[0][0])**0.5
print(Num_xy)
a = np.linspace(0, 1, num=int(Num_xy), endpoint=True)
b = np.linspace(0, 1, num=int(Num_xy), endpoint=True)
U, V = np.meshgrid(a, b)
X = U
Y = V

ipv.figure()
s = ipv.plot_surface(X, Y, Z_Num, color="orange")
#w = ipv.plot_wireframe(X, Y, Z, color="red")
ipv.animation_control(s, add = True, interval=200)#, sequence_length=2)
#mylink = widgets.link((s, 'sequence_index'), (w, 'sequence_index'))
ipv.show()
```

executed in 169ms, finished 13:28:41 2021-09-07

```
22.0

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-69cb1732bdb4> in <module>
     10 Y = V
     11
---> 12 ipv.figure()
     13 s = ipv.plot_surface(X, Y, Z_Num, color="orange")
     14 #w = ipv.plot_wireframe(X, Y, Z, color="red")

NameError: name 'ipv' is not defined
```

1 Compare Accuracy Numerical (WebGL) and Analytic

In [138]:
```
#Analytic Solution
MAX_N = 100

def Q_n(n_x,n_y):
    return np.pi*(n_x**2+n_y**2)**0.5

def B_n(n_x, n_y):
    return 4*(-1)**(n_x+n_y)/(n_x*n_y*np.pi**2)

def u(x,y,t):
    total = 0
    for n_x in range(1, MAX_N):
        for n_y in range(1, MAX_N):
            total += B_n(n_x,n_y)*np.sin(n_x*np.pi*x)*np.sin(n_y*np.pi*y)*np.cos(Q_n(n_x
    return total
```

executed in 9ms, finished 00:25:38 2021-08-18

In [101]:
```
#Put time and run commands
X = U
Y = V  # The y-axis is the wrong way around
#Z = np.array([[np.real(u(x,y)) for x in a] for y in b])
Z_Ana = np.array([[[u(x,y,t) for x in a] for y in b] for t in [0,1,2,3,4.4,16]])#np.lins

ipv.figure()
s = ipv.plot_surface(X, Y, Z_Ana, color="orange")
#w = ipv.plot_wireframe(X, Y, Z, color="red")
ipv.animation_control(s, add = True, interval=200)
#mylink = widgets.link((s, 'sequence_index'), (w, 'sequence_index'))
ipv.show()
```

executed in 6.69s, finished 23:43:09 2021-08-17

```
VBox(children=(Figure(animation=200.0, camera=PerspectiveCamera(fov=45.0, position=(0.
0, 0.0, 2.0), quaternion…
```

In [ ]:
```
#Compare
```

In [ ]:
```
#Z_Num[0]-Z_Ana[0].reshape(1,-1)
print(u(0.90,0.75,0))
Z_Num[0][0]
```