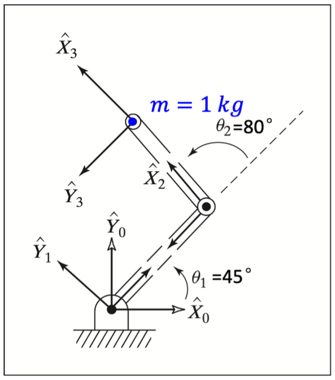
A robotic arm has two links with L_1= 1.2m, and L_2=0.8m. There is a 1 kg mass at the end of the second link. The joint's friction and link mass are ignored. In the beginning, the joints are locked, and the mass is static.



Let's write the equation of static force in manipulation

In [16]:
```
import numpy as np
from numpy import cos, sin, deg2rad ,array, cross
import sympy
from sympy import symbols, Matrix, simplify
```

In [17]:
```
# Variable initialization
theta1 = np.deg2rad(45)
theta2 = np.deg2rad(80)
l1 = 1.2  #m
l2 = 0.8 #m
m= 1 #kg
g = 9.81 #m/s^2
```

In [18]:
```
# Define rotation matrix
def Rot_z(theta):
    R = Matrix([[cos(theta), -sin(theta), 0],
                [sin(theta), cos(theta), 0],
                [0, 0, 1]])
    return R
```

For force transform, ${}^{i}f_i = {}^{i}_{i+1}R^{i+1}f_{i+1}$, we want to find torque τ_1 and τ_2 in link 1,2

In [19]:
```
f_2_in_2 = array([-m*g*sin(np.pi - theta1 - theta2), m*g*cos(np.pi - t
Tau_2_in_2 = np.cross(array([l2, 0, 0]), f_2_in_2)  # Torque at frame
R_1_2 = Rot_z(theta2)  # Rotation from frame 2 to frame 1
f_1_in_1 = R_1_2 @ f_2_in_2  # Force at frame 1 due to frame 2
tau_1_in_1 = R_1_2 ** 2 @ Tau_2_in_2 + np.cross(array([l1, 0, 0]), f_1
print("Torque at joint 2:", Tau_2_in_2[2])
print("Torque at joint 1:", tau_1_in_1[2])
```

```
Torque at joint 2: 4.50142787248301
Torque at joint 1: -3.82263315564503
```

In [ ]:

```

```