```
import numpy as np
def gram_schmidt(a):
    q = []
    for i in range(len(a)):
        #orthogonalization
        q_tilde = a[i]
        for j in range(len(q)):
            q_tilde = q_tilde - (q[j] @ a[i])*q[j]
        #Test for dependennce
        if np.sqrt(sum(q_tilde**2)) <= 1e-10:
            print("Vectors are linearly dependent.")
            print("GS algorithm terminates at iteration ", i+1)
            return q
        #Normalization
        else:
            q_tilde = q_tilde / np.sqrt(sum(q_tilde**2))
            q.append(q_tilde)
    print("Vectors are linearly independent.")
    return q

a=np.array([(1,-2,1,-1),(1,1,3,-1),(-3,7,1,3)])
q=gram_schmidt(a)
print(q)
#Test orthonormality
print("Norm of q[0] :", (sum(q[0]**2))**0.5)
print('Inner product of q[0] and q[1] :', q[0] @ q[1])
print("Inner product of q[0] and q[2] :", q[0] @ q[2])
print("Norm of q[1] :", (sum(q[1]**2))**0.5)
print("Inner product of q[1] and q[2] :", q[1] @ q[2])
print("Norm of q[2] :", (sum(q[2]**2))**0.5)
Vectors are linearly independent.
[array([ 0.37796447, -0.75592895,  0.37796447, -0.37796447]), array([ 0.17
457431,  0.56736651,  0.7855844 , -0.17457431]), array([-0.57154761, -0.32
659863,  0.48989795,  0.57154761])]
Norm of q[0] : 0.9999999999999999
Inner product of q[0] and q[1] : 1.1102230246251565e-16
Inner product of q[0] and q[2] : -7.216449660063518e-16
Norm of q[1] : 1.0
Inner product of q[1] and q[2] : 4.996003610813204e-16
Norm of q[2] : 1.0
```