```
from numpy import *
import math
import numpy as np


def NewMark_a_constante(properties, pj, time):

    m =   properties["m"]
    k =   properties["k"]
    c =   properties["c"]
    u0 =  properties["u0"]
    du0 = properties["du0"]
    h = time[1] - time[0]
    intervalos = time[-1]/h

    γ = 0.5
    ß = 0.25

#-------------------Calculos iniciales--------------------

    ddu0    = (pj[0] - c*du0 - k*u0)/(m)
    a1      = (1/ (ß*h**2) )*m   +     (γ/ (ß*h) )*c
    a2      = (1/   (ß*h)  )*m   +     (γ/ß - 1)*c
    a3      = (1/  (2*ß) -1)*m   +     (γ/(2*ß) - 1)*c*h
    kgor    = k + a1

    inte = int(intervalos)
    u       =  zeros((inte+1))
    du      =  zeros((inte+1))
    ddu     =  zeros((inte+1))
    t       =  zeros((inte+1))
    Pi_gor  =  zeros((inte+1))


#-----------Calculos para el tiempo de paso i-------------

    u[0]        = u0
    du[0]       = du0
    ddu[0]      = ddu0
    Pi_gor[0]   = 0
    t[0]        = 0

    for i in range(0,inte):
        Pi_gor[i+1]   = pj[i+1] + a1*u[i] + a2*du[i] + a3*ddu[i]
        u[i+1]        = Pi_gor[i+1] / kgor
        du[i+1]       = γ/(ß*h) * (u[i+1] - u[i])    +  (1- γ/ß)*du[i]    + h*(1-
γ/(2*ß))*ddu[i]
        ddu[i+1]      = (1/(ß*h**2))*(u[i+1] - u[i]) -  (1/(ß*h))*du[i]   - (1/(2*ß) -1) *
ddu[i]
        t[i+1]        =  h + h*i

        Pi_gori     = Pi_gor[i]
        pji         = pj[i]
        ui          = u[i]
        ddui        = ddu[i]
        dui         = du[i]
        ti          = t[i]


    return u, du, ddu

def Diferencia_Central(properties,  pj, time):

    m = properties["m"]
    k = properties["k"]
    c = properties["c"]
    u0 = properties["u0"]
    du0 = properties["du0"]

    h = time[1] - time[0]
    intervalos = int(time[-1]/h)




    p0      = pj[0]
    ddu0    = (p0 - c*du0 - k*u0)/(m)
    u1      = u0 - h*du0 + ((h**2)/2)*ddu0
    kgor    = m/(h**2) + c/(2*h)
    a       = m/(h**2) - c/(2*h)
    b       = k - (2*m)/(h**2)


    u =   zeros((intervalos+2))
    du =  zeros((intervalos+1))
    ddu = zeros((intervalos+1))
    t =   zeros((intervalos+1))


    #Calculos para el tiempo de paso i
    u[0] = u0

    for i in range(0,intervalos+1):
        if i == 0:
            Pi_gor = pj[i] - a*u1 - b*u[i]
            u[i+1] = Pi_gor / kgor

            pji = pj[i]
            ui = u[i]
            ui1 = u[i+1]

            t[i] = h*i


            #print ( f" i = {i} ",f" t = {t[i]} ",f" pj[i] = {pj[i]} ", f" ui-1 = {u1} ",f"
ui = {u[i+1]} ", f" Pi_gor = {Pi_gor} ")

        else:
            Pi_gor = pj[i] - a*u[i-1] - b*u[i]
            u[i+1] = Pi_gor / kgor
            t[i] = h*i


            pji = pj[i]
            ui01 = u[i-1]
            ui = u[i]
            ui1 = u[i+1]
    cc = 0
    for i in range(0,intervalos+1):

        if i == 0:
            du[i]  = du0
            ddu[i] = ddu0

        else:
            du[i]  = (u[i+1]-  u[i-1]       )    /(2*h)
            ddu[i] = (u[i+1]-2*u[i] + u[i-1])    /(h**2)
        c += 1
    u = u[:-1]

    return u, du, ddu

def Rectangular(properties, pj, time):

    m = properties["m"]
    k = properties["k"]
    c = properties["c"]
    u0 = properties["u0"]
    du0 = properties["du0"]


    wn= ((k/m))**0.5      # 1/s
    cr = 2*m*wn           # Kg/s
    ξ =  c/cr             #  -
    wd= wn*(1-ξ**2)**0.5  # 1/s

    h = time[1] - time[0] # s
    intervalos = int(time[-1]/h)

    p0      = pj[0]
    ddu0    = (p0 - c*du0 - k*u0)/(m)

    su_A   = zeros((intervalos+1))
    su_B   = zeros((intervalos+1))
    A      = zeros((intervalos+1))
    B      = zeros((intervalos+1))
    u      = zeros((intervalos+1))
    du     = zeros((intervalos+1))
    ddu    = zeros((intervalos+1))

    c = 0

    for i in range(len(u)):

        su_A[i] = (pj[i])*np.exp(ξ*wn*(i*h))  *np.cos(wd*(i*h))   #N
        su_B[i] = (pj[i])*np.exp(ξ*wn*(i*h))  *np.sin(wd*(i*h))   #N

        if i == 0:
            A[i] = h/(m*wd)*su_A[i]      #N
            B[i] = h/(m*wd)*su_B[i]      #N

        else:
            A[i] = (h/(m*wd))*su_A[i]+A[i-1]
            B[i] = (h/(m*wd))*su_B[i]+B[i-1]



        # if c == 0:
        #     u[0]    = u0
        #     du[0]   = du0
        #     ddu[0]  = ddu0
        #     c+=1


        u[i]   = A[i]*(np.exp(-ξ*wn*time[i])) *  np.sin(wd*time[i]) - B[i]*(np.exp(-
ξ*wn*time[i])) * np.cos(wd*time[i])
        du[i]  = (u[i]- u[i-1])/h
        ddu[i] = (du[i]-du[i-1])/h



    # print (f"m = {m}")
    # print (f"k = {k}")
    # print (f"A = {A}")
    # print (f"B = {B}")
    # print (f"h = {h}")
    # print (f"ξ = {ξ}")
    # print (f"wn = {wn}")

    return u, du, ddu
```