```
from diffeqpy import de
from juliacall import Main as jl
import numpy as np
import matplotlib.pyplot as plt

'''
def rober(du, u, p, t):
    y1, y2, y3 = u
    k1, k2, k3 = p
    du[0] = -k1 * y1 + k3 * y2 * y3
    du[1] = k1 * y1 - k3 * y2 * y3 - k2 * y2**2
    du[2] = y1 + y2 + y3 - 1
    return

M = np.array([[1.0, 0.0, 0.0],
              [0.0, 1.0, 0.0],
              [0.0, 0.0, 0.0]])

f = de.ODEFunction(rober, mass_matrix = jl.convert(jl.Array,M))

prob_mm = de.ODEProblem(f, [1.0, 0.0, 0.0], (0.0, 1e5), (0.04, 3e7, 1e4))

sol = de.solve(prob_mm, de.RadauIIA5(), reltol = 1e-8, abstol = 1e-8)


plt.plot(sol.t,de.transpose(de.stack(sol.u)))
plt.grid()
plt.show()
'''



C1=1e-6; C2=2e-6; C3=3e-6; C4=4e-6; C5=5e-6
R0=1e3
R1=R2=R3=R4=R5=R6=R7=R8=R9=9e3
Ub=6
uf=0.026; alpha=0.99; beta=1e-6

M = np.array([[-C1,C1,0,0,0,0,0,0],
              [C1,-C1,0,0,0,0,0,0],
              [0,0,-C2,0,0,0,0,0],
              [0,0,0,-C3,C3,0,0,0],
              [0,0,0,C3,-C3,0,0,0],
              [0,0,0,0,0,-C4,0,0],
              [0,0,0,0,0,0,-C5,C5],
              [0,0,0,0,0,0,C5,-C5]])

y0 = [0.,
      Ub/(R2/R1+1.),
      Ub/(R2/R1+1.),
```

```
      Ub,
      Ub/(R6/R5+1.),
      Ub/(R6/R5+1.),
      Ub,
      0.]



def transamp(du, u, p, t):
  U1,U2,U3,U4,U5,U6,U7,U8=u
  uf, alpha, beta, R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, Ub = p
  if (U2-U3)/uf > 300. or (U5-U6)/uf > 300.:
    return
  fac1 = beta*(np.exp((U2-U3)/uf)-1)
  fac2 = beta*(np.exp((U5-U6)/uf)-1)
  du[0] = (U1 - 0.1*np.sin(200*np.pi*t))/R0
  du[1] = U2/R1+(U2-Ub)/R2+(1-alpha)*fac1
  du[2] = U3/R3-fac1
  du[3] = (U4/-Ub)/R4+alpha*fac1
  du[4] = U5/R5+(U5-Ub)/R6+(1-alpha)*fac2
  du[5] = U6/R7-fac2
  du[6] = (U7-Ub)/R8+alpha*fac2
  du[7] = U8/R9
  return


f = de.ODEFunction(transamp, mass_matrix = jl.convert(jl.Array,M))

prob_mm = de.ODEProblem(f, y0, (0.0, 0.2), (uf, alpha, beta, R0, R1, R2, R3, R4,
R5, R6, R7, R8, R9, Ub))
prob_f = de.jit(prob_mm)
sol = de.solve(prob_f, de.Rodas5P(), reltol = 1e-8, abstol = 1e-8)

#for i in range(8):
plt.plot(sol.t,de.transpose(de.stack(sol.u)))
plt.grid()
plt.show()
```