```
#%%
# Imports/Setup
import sympy as sp
from sympy import eye, sin, cos, diff, Matrix, symbols, Function,
pretty_print, simplify, init_printing, latex, sqrt
from sympy.physics.vector import dynamicsymbols
from sympy.physics.vector.printing import vpprint, vlatex
from IPython.display import Math, display

# Define symbols
g, m, k1, k2, t, F, b, theta = symbols('g, m, k1, k2, t, F, b, theta')
z = dynamicsymbols('z')
zdot = z.diff(t)
zddot = zdot.diff(t)
```

```
#%%
# Redefine new EOM
tempEOM = m*zddot + k1*z + k2*z**3 - (1/sqrt(2))*m*g - F+b*zdot
temp = sp.solve(tempEOM, (zddot))
zdd_EOM = simplify(temp[0])
display(Math(vlatex(zdd_EOM)))
```

```
#%%
# SVF
vars = [(b, 0.1), (g, 9.8), (theta, 45), (m, 0.5), (k1, 0.05), (k2, 0.02)]

svf = Matrix([[zdot],[zdd_EOM]])
states = Matrix([[z], [zdot]])
inputs = Matrix([[F]])
display(Math(vlatex(svf)))
```

```
#%%
# Linearize

'''
    Setting SVF to zero:
    --------------------
    zdot = 0
    F_eq = k1*z_eq + k2*z_eq^3 - (sqrt(2)/2) * mg
    z_eq = our choice


    After we linearize and set theta_eq = 0:
    ----------------------------------------
    zdot_eq = 0
    z_eq = 0
    F_eq = -(sqrt(2)/2) * mg
'''

A = svf.jacobian(states)
```

```
B = svf.jacobian(inputs)

ze = symbols('z_e')

A_ze = A.subs([(z, ze)])
B_ze = B.subs([(z, ze)])

A = A_ze.subs([(ze, 0)])
B = B_ze.subs([(ze, 0)])

A_lin = A.subs(vars)
B_lin = B.subs(vars)

display("Linear EOMs (A) then (B):")
display(Math(vlatex(A_lin)))
display(Math(vlatex(B_lin)))

#%%
# Getting T.F.
C = Matrix([[1, 0]])
D = Matrix([[0]])
I = eye(2)
s = symbols('s')

TF = simplify(C @ (s*I - A).inv() @ B + D)
TF_lin = TF[0].subs(vars)

display("TF:")
display(Math(vlatex(TF[0])))
display(Math(vlatex(TF_lin)))

#%%
# Finding Y/R
R, kp, kd = symbols('R, k_p, k_d')

eq = z - TF[0]*((kp)*(R-z) - (kd*s*z))
result = sp.solve(eq, z)
YR = simplify(result[0]/R)
display(Math(vlatex(YR)))

#%%
# Display State Space
display("SS: A, B, C, D respectivly")
display(Math(vlatex(A_lin)))
display(Math(vlatex(B_lin)))
display(Math(vlatex(C)))
display(Math(vlatex(D)))
```