Control Theory homework 1

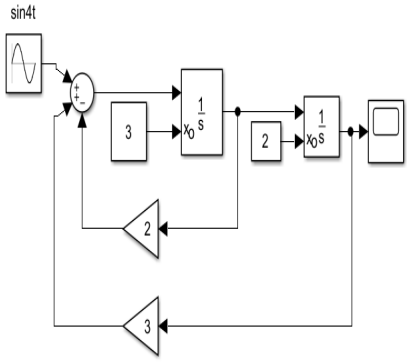
Kamil Hayrullin
BS18-02, variant 'o'

1 Solve second order differential equation:

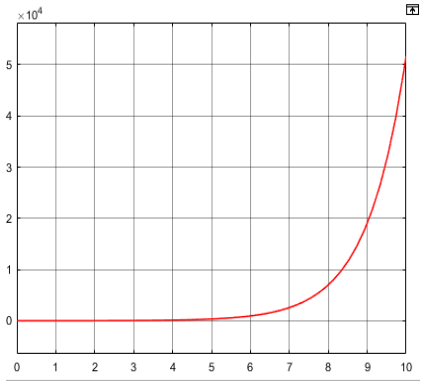
$$x'' + 2x' - 3x = sin4t \tag{1}$$

$$x'(0) = 3, x(0) = 2 \tag{2}$$

(A) Draw a schema in Simulink (do not use transfer func block).



(a) Simulink schema



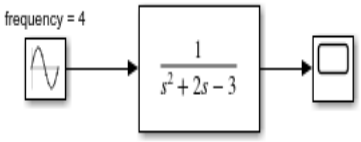
(b) Plot of Simulink shema

(B) Draw a schema in Simulink (use transfer function block).

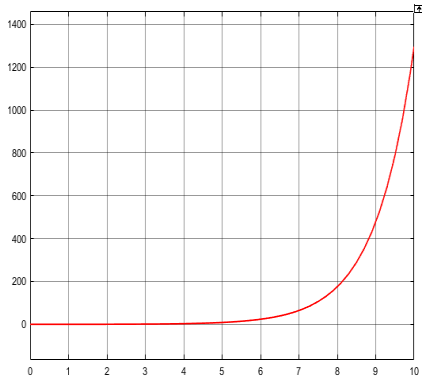
Let's calculate transfer function of equation (1):

$$s^2X(s) + 2sX(s) - 3X(s) = sin4s \tag{3}$$

$$X(s)/sin4s = 1/(s^2 + 2s - 3) \tag{4}$$



(a) Simulink schema



(b) Plot of Simulink shema

(C) Solving differential equation using dsolve in matlab.

```
syms x(t) t
equation = dsolve('D2x(t)+2*Dx(t)-3*x(t)=sin(4*t)', 'x(0) = 2', 'Dx(0) = 3', 't')
pretty(equation)
ezplot(equation,[0,10])
```

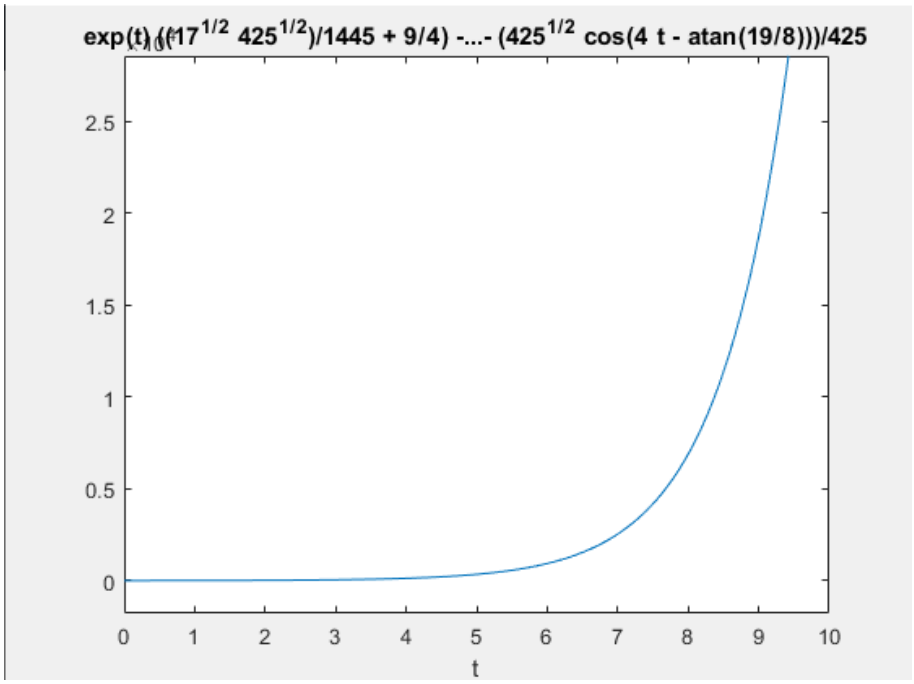


Figure 3: Plot of solution with dsolve

(D) Solving differential equation using Laplace transform in matlab.

```
syms t s x(t) X
equation = diff(x(t),t, 2)+2*diff(x(t),t)-3*x(t)==sin(4*t)
simplify(equation)
F = laplace(equation, t, s)

F = subs(F,[laplace(x, t, s)],[X])
X = solve(F,X)
ans = ilaplace(X)
ans = subs(ans,[x(0), diff(x(t))], [2, 3])
pretty(ans)
ezplot(ans,[0, 10])
```

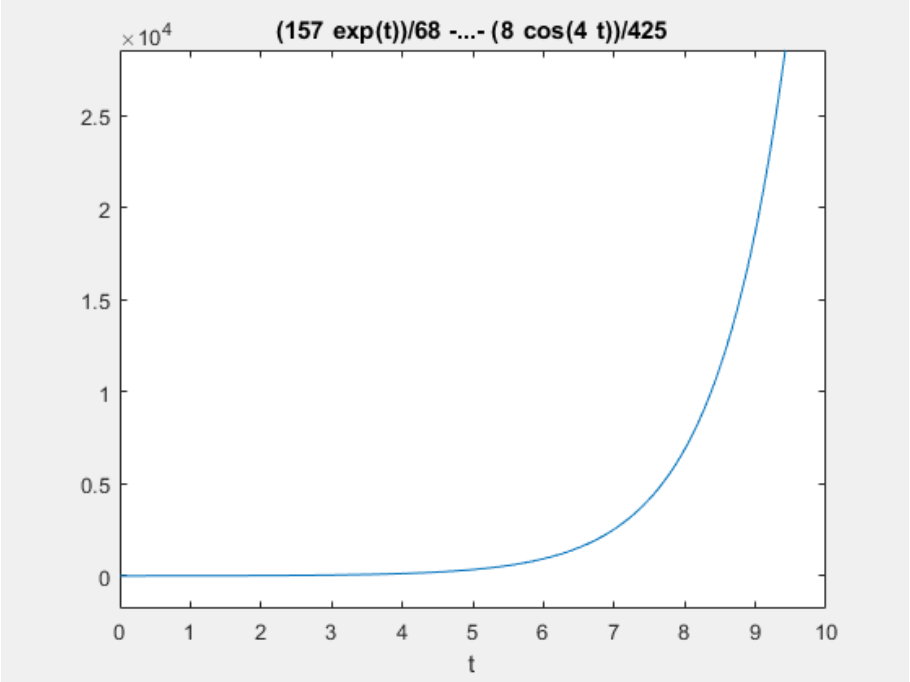


Figure 4: Plot of solution with Laplace transform

2 Find State Space Model of the system.

$$x'' = t + 3 \tag{5}$$

$$y = x + 2x' \tag{6}$$

$$\begin{bmatrix} x' \\ x'' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ x' \end{bmatrix} + \begin{bmatrix} 1 & 3 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix}$$

$$\mathrm{y} = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ x' \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix}$$

3 Find State Space Model of the system.

$$x'''' - 2x''' + x'' - x' + 5 = u1 + u2 \tag{7}$$

$$y = 2x + x' - u1 \tag{8}$$

$$\begin{bmatrix} x' \\ x'' \\ x''' \\ x'''' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ x' \\ x'' \\ x''' \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & -5 \end{bmatrix} \begin{bmatrix} u1 \\ u2 \\ 1 \end{bmatrix}$$

$$\mathrm{y} = \begin{bmatrix} 2 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ x' \\ x'' \\ x''' \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u1 \\ u2 \\ 1 \end{bmatrix}$$

4 Write a function in python that converts any ODE (power n) to the state space representation

link to this code on repl.it
https://repl.it/repls/NavajowhiteDigitalProfessional

```
import numpy as math

def StateSpaceModel():
  # enter degree of the polynomial
  n = int(input("Enter degree of polynomail: "))
  frm = int(input("Generate coefficients in range from (int): "))
  to = int(input("Generate coefficients in range to (int): "))

  # generate random coefficients: [a0 a1 ... ak]
  a = math.random.randint(frm, to, n)
  print("Coefficients are:", a)
  # create b coefficient
  b = math.random.randint(frm, to)
  #divide all coefitients by coef. of polynomial of biggest degree
```

```
  a_norm = a[1:] / a[0]

  A = math.zeros((n-1, n-1))  # state matrix
  A[0 , 0:] = -a_norm #put coef. to matrix A
  A[1:, 0:(n-2)] = math.eye(n-2) #diagonal matrix with ones of size n -2

  B = math.zeros((n-1)) #put zeros to B matrix
  B[0] = b/a[0]             # calculate B matrix
  print("b coefficient: ",b)
  print("State space representation:")
  print("x' = ")
  print(A)
  print("x + ")
  print(B)
  return
```

5 Write functions in python that solves ODE and its state space representation. Test your functions on the ODE from task2. Draw plots.

Link to this code on collab:
https://colab.research.google.com/drive/10BoOVXyyugrwR0WKDw0ysiYhzHdx-Obe

```
import numpy as math
import matplotlib.pyplot as plt
from scipy.integrate import odeint

#ODE
def ODE(U, t):
    return [U[1], -2*U[1] +3*U[0] + math.sin(2*t)]
#initial conditions
U0 = [2, 3]
#from 0 to 10
ts = math.linspace(0, 10, 200)
Us = odeint(ODE, U0, ts)
xs = Us[:,0]

plt.xlabel("t")
plt.ylabel("x")
plt.plot(ts,xs);

#State space
#creating matrix, n = degree of polynomial
n = 3
```

```
a = [-3.0, 2.0, 1.0]
a = math.flip(a)
a_norm = a[1:] / a[0]
A = math.zeros((n-1, n-1))
A[0 , 0:] = -a_norm
A[1:, 0:(n-2)] = math.eye(n-2)

def StateSpace(U, t):
    return A.dot(U) + [math.sin(2*t), 0]  # solution with b
#from 0 to 10
U0 = [2, 3]
#initial condition
ts = math.linspace(0, 10, 200)
Us = odeint(StateSpace, U0, ts)
xs = Us[:,0]

plt.xlabel("t")
plt.ylabel("x")
plt.plot(ts,xs);
```

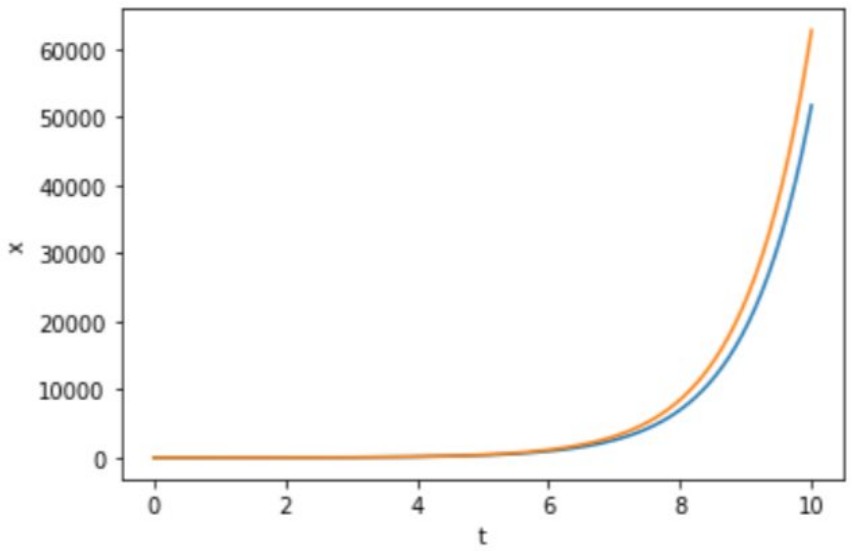


Figure 5: Blue is 'ODE', Red is 'StateSpace'

function diverges and unstable