Control Theory.
Assignment One.
Polina Turischeva. Group 3. Variant n

Task 1

Preparation is done

Task 2

B

Initial equation: $x'' - x' - 2x = sin2t - 3$, $x'(0) = 2$, $x(0) = 5$
Initial conditions are neglected for transfer function. We change the primes of power n to s in power of n. $u(t) = 2sin(t) - 3$, as there are no $u(t)$ derivatives, hence, there will be 1 in the nominator. $H(s) = \frac{Y(s)}{U(s)} = \frac{1}{s^2 - s - 2}$
The direction of the solution is the other as it does not take into account initial conditions.

C

Matlab code for ODE solution:

```
syms x(t)
eqn = diff(x,t,2) == diff(x,t) +2*x-3+sin(2*t);
Dx = diff(x,t);
cond = [x(0)==5, Dx(0)==2];
xSol(t) = dsolve(eqn,cond)
```

Matlab output:

```
xSol(t) =(23*exp(-t))/15 + (23*exp(2*t))/12 +
+(10^(1/2)*cos(2*t + atan(3)))/20 + 3/2
```

D

Matlab code for ODE solution with Laplace Transform:

```
clc; clearvars; syms t s Y x(t) Dx(t)
assume([t Y] > 0);
Dx = diff(x, t);
D2x = diff(Dx, t);
LS = sin(2*t)-3;
EQN=D2x-Dx-2*x-LS;
LEQN=laplace(EQN,t,s);
LT_Y=subs(LEQN,laplace(x,t,s),Y);
% x(0) = 5
LT_Y=subs(LT_Y, x(0), 5);
% x (0)= 2
LT_Y=subs(LT_Y, subs(diff(x(t), t), t, 0), 2);
xs=solve(LT_Y,Y);
x=ilaplace(xs,s,t)
ezplot(x); shg
```

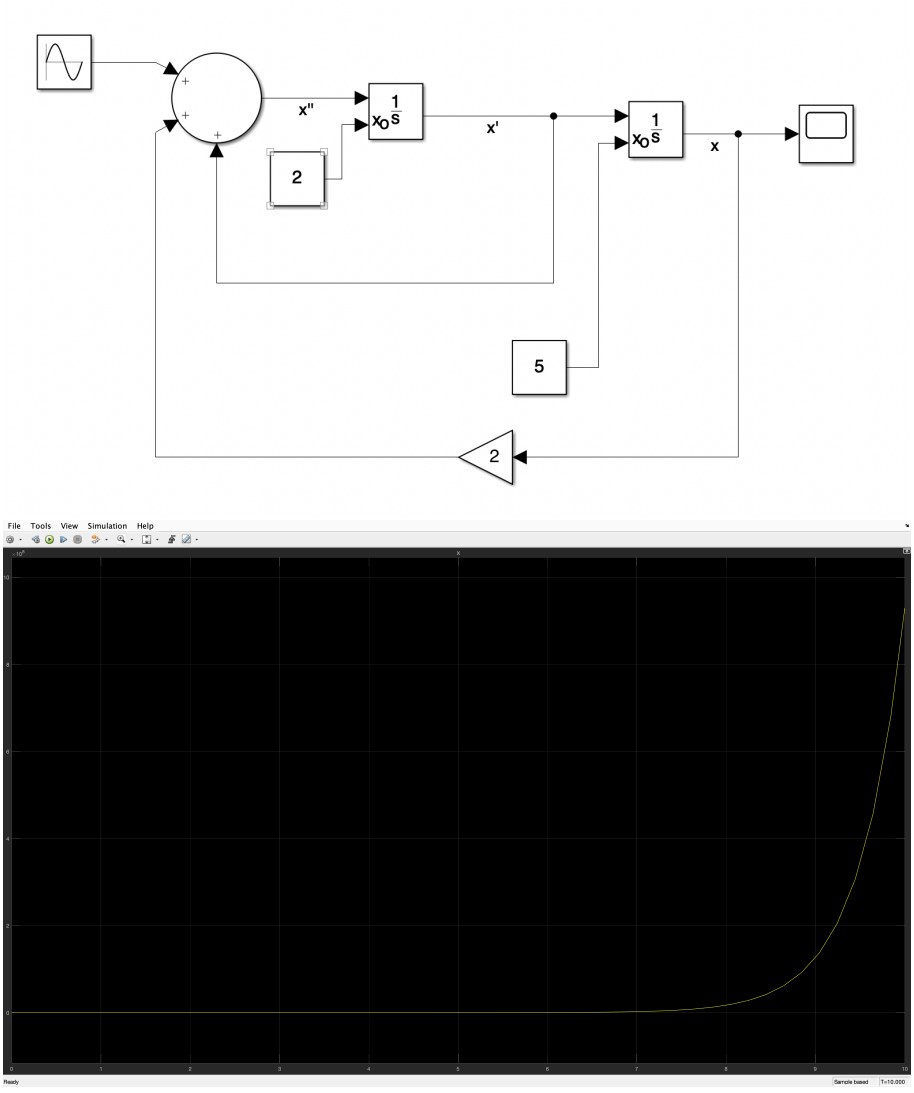

Figure 1: Solution for 2A

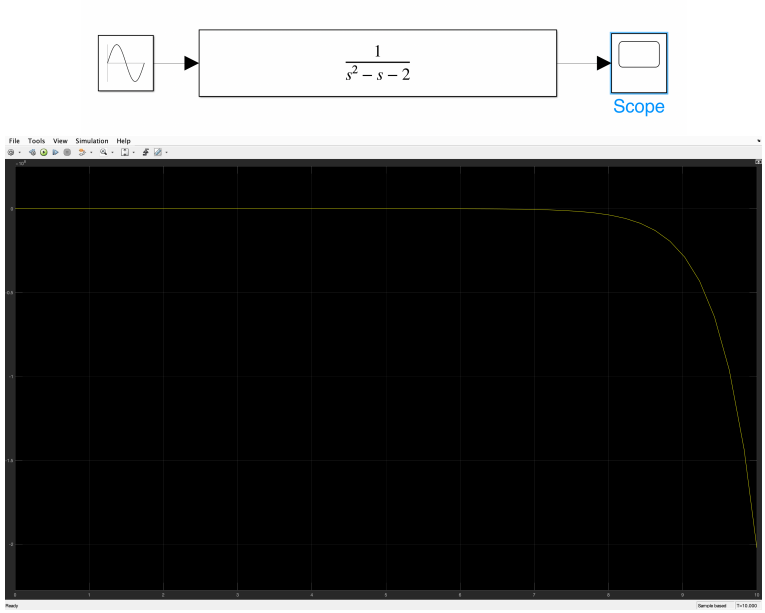

Figure 2: Solution for 2B

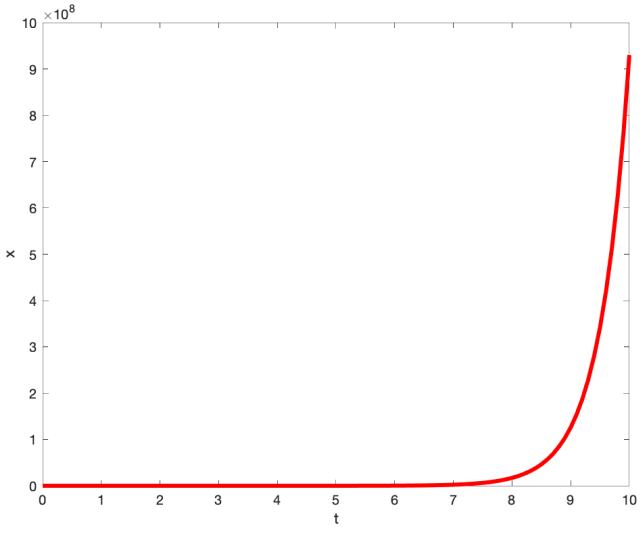

Figure 3: Solution for 2C

Task 3

$x'' + 3x' + 3x = t$, $y = x' + 2t \Rightarrow x'' = -3x' - 3x + 2y$

$$a = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x \\ x' \end{bmatrix} \Rightarrow a' = \begin{bmatrix} 0 & 1 \\ -3 & -3 \end{bmatrix} a + \begin{bmatrix} 0 \\ 1 \end{bmatrix} t$$

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

Task 4

$3x'''' + 2x''' - x'' + 2x' = u_1 + 5u_2 + 3$, $y = x' + u_2$

$$a' = \begin{bmatrix} x \\ x' \\ x'' \\ x''' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{2}{3} & -\frac{1}{3} & -\frac{2}{3} & 0 \end{bmatrix} a + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 5 \end{bmatrix} u_2 + 3$$

Task 5

```
import numpy as np

# the input should be array of coefficients for primes
# in decreasing order
# if there is a missing prime => there should be 0
# b is a string?
def toStateSpace(a, b):
  n = len(a)
  A = np.eye(n-1, k=1)
  lastLine = [(c*(-1)/a[0]) for c in a[1:]]
  lastLine.reverse()
  A[n-2]=lastLine
  u = np.zeros((n-1, 1))
  u[n-2] = [1/a[0]]
  return A, u
```

Task 6

Code with solving functions:

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

# coef - coefficients on the right part, before derivates of y(t)
# coef[0] is coef before y
# coef[1] is coef before y'
# u = u(t)

def task6(coef, init_values, u):
  # solution for ODE
  t = np.linspace(0, 10, 100)
  # print(t)
  sol = odeint(model, init_values, t, args=(coef, u))
  y_sol = sol[:,0]


  #solytion for state space
  coef.reverse
  a, b = toStateSpace (coef, u)
  # as there is no equation for y(x) we assume y=x

  c = np.zeros((b.shape)).T
  c[0][0] =1
  d =np.array([[0]])
  sys = signal.StateSpace(a, b, c, d)
  t2,y2 = signal.step(sys, init_values, t)
  sol2 = odeint(withStateSpace, init_values, t, args =(a,b, u))
  y_sol2 = sol2[:,0]

  print("A=", a,'\n', "B=", b)
  du = [u(i) for i in t ]
  t3, y3, ev = signal.lsim(sys, du, t, init_values)
```

```
  #drawing
  plt.figure(1, figsize=(10, 8))
  plt.plot(t,y_sol,'r-',linewidth=1,label='ODE Solution')
  plt.plot(t2,y2,'g:',linewidth=2,label='State Space - step ')
  plt.plot(t3,y3,'y-',linewidth=2,label='State Space-lsim')
  plt.plot(t,y_sol2,'b:',linewidth=2,label='State Space-odeint')
  plt.xlabel('Time')
  plt.ylabel('Response (y)')
  plt.legend(loc='best')


def model(init,t, coef, u):
  dydt = init
  intermediate = coef[:len(coef)-1]

  intermediate[:]=[x/((-1)*coef[len(coef)-1]) for x in intermediate]
  intermediate = [(init[i]*intermediate[i]) for i in range(len(init))]
  dydt = list(dydt)
  dydt.append( sum(intermediate) + u(t)/coef[len(coef)-1] )
  dydt.pop(0)
  return dydt

def withStateSpace(x,t, state_m, input_m, u):
  return (state_m.dot(x) + (input_m*(u(t))).T)[0]
```

Code with task 2 testing:

```
from math import sin, cos

# x''-x'-2x=sin2t -3

def u(t):
  return sin(2*t) -3

task6([-2, -1, 1], [5,2], u)
```

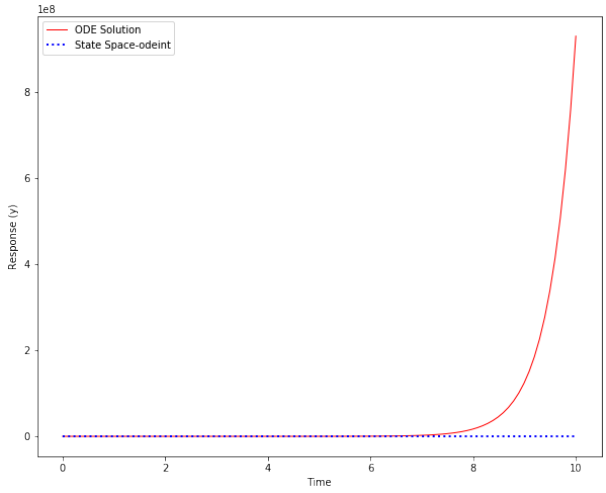

Figure 4: Solution for second task.
State space solution is very different from MatLab and python ODE solution. The curve shape is same but the values are much less. MatLab dsolve solution was same as python ODE solution, hence, I assume it to be correct.

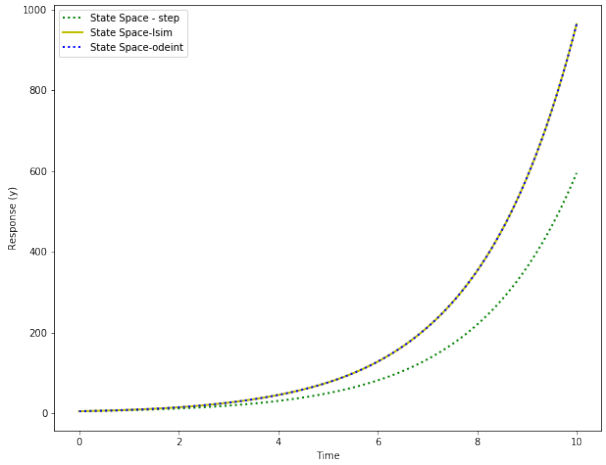

Figure 5: Checking State Space solution. Step solution is different as it does not take initial conditions. StateSpace solutions with odeint and lsim functions are the same, hence I assume them to be correct.

Conclusion: I got two different results for ODE solution and state space solution. I can not explain this results yet and would be happy to receive some answers/hints in the review. From the graph we can conclude that ode solution diverges. Moreover, both solutions have the same - exponential form. As exponent is growing, we may say that for $e^{\lambda t}$ $\lambda > 0$, therefore, every solution is unstable.
If we check the exact solution it actually consists of 2 exponents :

$$x(t) = \frac{1}{60}(92e^{-t} + 155e^{2t} - 9sin(2t) + 3cos(2t) + 90)$$

For big t we can neglect trigonometric function and constant as they are bounded, $92e^{-t}$ can also be neglected as it seeks to 0. Therefore, we get

$$x(t) = \frac{1}{60}155e^{2t} = \frac{23}{16}e^{2t}$$

This confirms our observation from the graph, that the solution is unstable and diverging.