# Control Theory HW 1

Danat Ayazbayev
BS-18-03

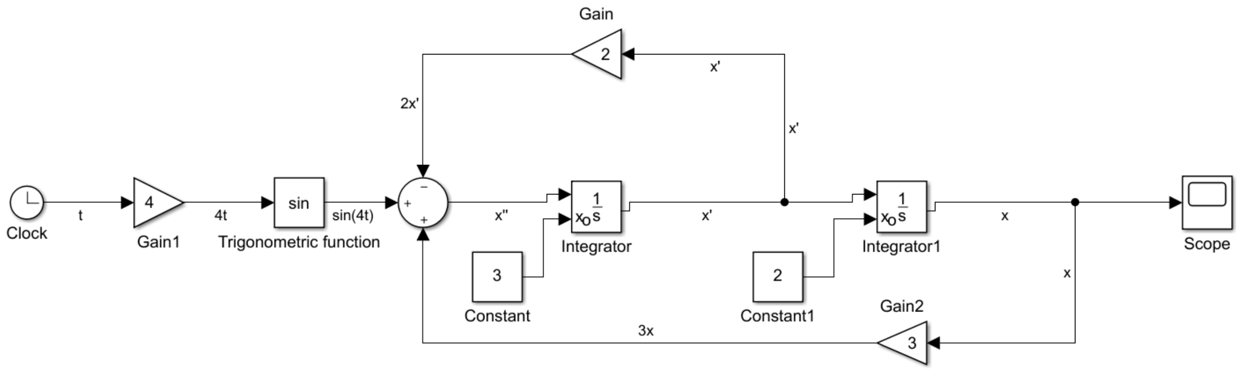
February 2020

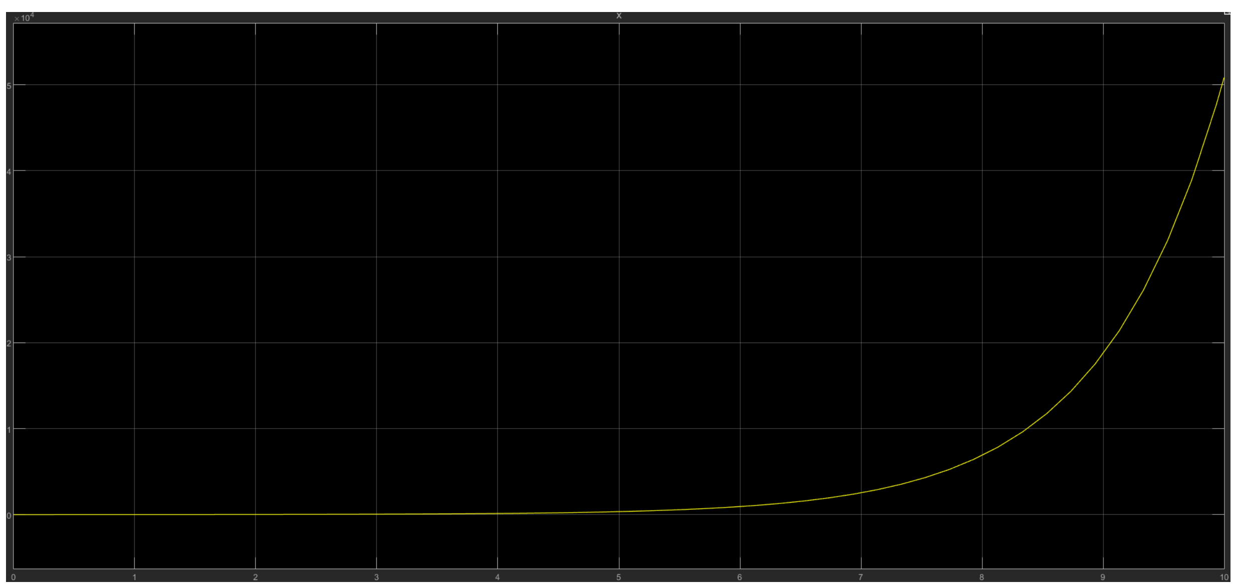
## 1 Preparation

Github repo: https://github.com/AyazbayevD/ControlTheory2020
Variant: o

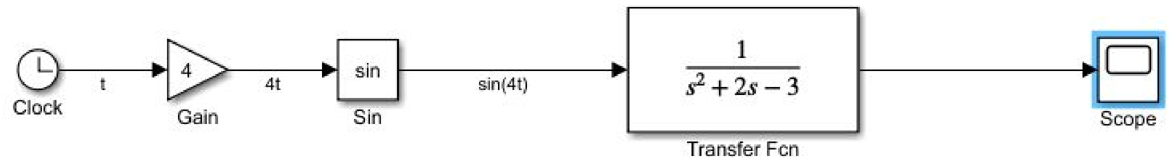
## 2 Solve second order diff equation: $x'' + 2x' - 3x = sin(4t), x'(0) = 3, x(0) = 2$

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



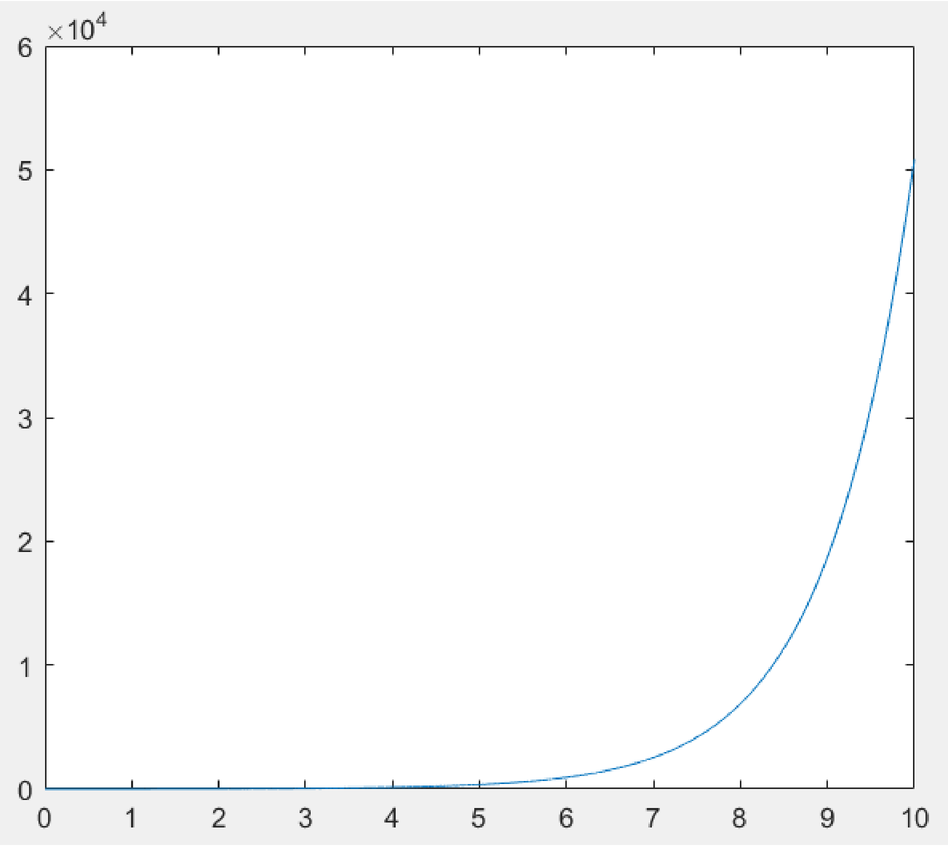



### 2.b Draw a schema in Simulink (use transfer func block)




### 2.c Solve diff equation with matlab function (for example dsolve) and draw a plot in matlab

```
syms x(t)
Dx = diff(x, t);
eqn = diff(x, t, 2) == sin(4 * t) - Dx * 2 + x * 3;
cond = [x(0) == 2, Dx(0) == 3];
xSol(t) = dsolve(eqn, cond);
xSol(t) = simplify(xSol(t));
disp(xSol(t));
var = 0:0.001:10;
plot(var, xSol(var));
```




## 2.d Solve diff equation with Laplace transform in matlab

```
syms x(t) s
Dx = diff(x, t);
eqn = diff(x, t, 2) == sin(4 * t) - Dx * 2 + x * 3;
eqnLT = laplace(eqn, t, s);
syms x_LT
eqnLT = subs(eqnLT, laplace(x, t, s), x_LT);
x_LT = solve(eqnLT, x_LT);
xSol = ilaplace(x_LT, s, t);
xSol = simplify(xSol);
vars = [x(0), Dx(0)];
values = [2, 3];
xSol(t) = subs(xSol, vars, values);
disp(xSol(t));
```

## 3 Find State Space Model of the system: $x'' = t + 3, y = x + 2x'$

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

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

## 4 Find State Space Model of the system: $x'''' - 2x''' + x'' - x' + 5 = u_1 + u_2, y = 2x + x' - u_1$

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

$$y = \begin{bmatrix} 2 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ x' \\ x'' \\ x''' \end{bmatrix} - \begin{bmatrix} 0 \end{bmatrix} u_1$$

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

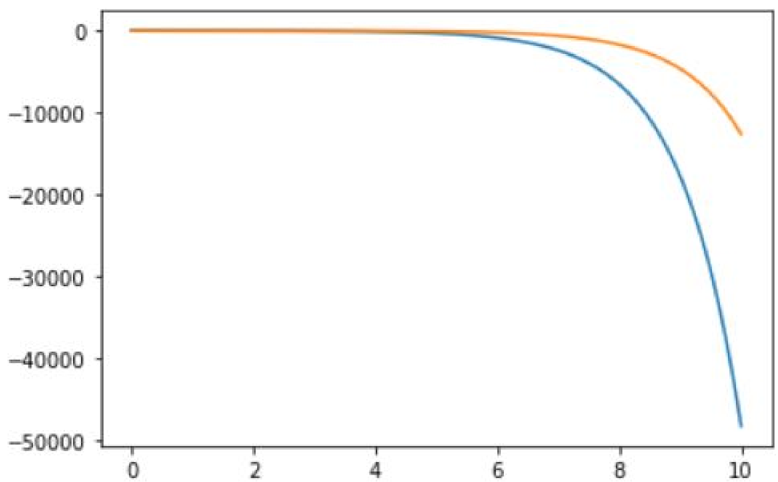
```python
def ss(a, n):  # function takes array and its size as arguments
    A = []
    row = []
    for i in range(n):
        row.append(-a[n - i - 1] / a[n])
    A.append(row)
    for i in range(n - 1):
        row = []
        for j in range(n):
            if j == i:
                row.append(1)
            else:
                row.append(0)
        A.append(row)
    return A
```

## 6 Write functions in python that solves ODE and its state space representation. Test your functions on the ODE from task2. Draw plots. Use odeint from scipy.integrate library. Is the ODE stable? Does its solution converges or diverges?

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


def ss(a, n):  # function takes array and its size as arguments
    A = []
    row = []
    for i in range(n):
        row.append(-a[n - i - 1] / a[n])
    A.append(row)
    for i in range(n - 1):
        row = []
        for j in range(n):
            if j == i:
                row.append(1)
            else:
                row.append(0)
        A.append(row)
    return A


def size(A: list):  # return size of matrix
    n = len(A)
    if n == 0:
        return [0, 0]
    else:
        m = len(A[0])
    for i in range(len(A)):
        if len(A[i]) != m:
            print("Incorrect matrix")
            raise RuntimeError
    return [n, m]


```

```python
def sum(A, B):
    an, am = size(A)
    bn, bm = size(B)
    if an != bn and am != bm:
        print("Incorrect matrix")
        raise RuntimeError
    sum = []
    for i in range(an):
        row = []
        for j in range(am):
            row.append(A[i][j] + B[i][j])
        sum.append(row)
    return sum


def mult(A, B):
    an, am = size(A)
    bn, bm = size(B)
    if am != bn:
        print("Incorrect matrix")
        raise RuntimeError
    product = []
    for i in range(an):
        row = []
        for j in range(bm):
            sum = 0
            for g in range(am):
                sum += A[i][g] * B[g][j]
            row.append(sum)
        product.append(row)
    return product


def linear_method(x, t, a: float, b: float, d):
    x1, x2 = x
    dx1dx2 = [-a * x1 - b * x2 - d(t), x1]
    return dx1dx2


def state_space(x, t, a, n, func):
    b = [[-func(t)], [0]]
    X = [[x[0]], [x[1]]]
    A = ss(a, n)
    prod = mult(A, X)
    res = sum(prod, b)
    return [res[0][0], res[1][0]]
```

```python

x0 = [0, -3]
t = np.linspace(0, 10, 200)
func = lambda t: -np.sin(4 * t)
ode = odeint(linear_method, x0, t, args=(2, -3, func))
y1 = []
for val in ode:
    y1.append(val[1])
ode = odeint(state_space, x0, t, args=([1, 2, -3], 2, func))
ys = []
y2 = []
for val in ode:
    y2.append(val[1])

plt.plot(t, y1, label="Linear")
plt.plot(t, y2, label="State Space")
plt.show()
```




ODE is stable
Solution diverges