OPTIMIZATION THEORY

CDO

Sergio Barbero, Maaz Ashiq

Introduction to the problem

We want to implement a method of direct dynamical optimization with additional constraints.

In order to make it we start from these formulas:

$$x_{n+1} = Ax_n^2 + Bu_n^2$$

$$J = \sum_{n=0}^{n-1} \left(Q\, x_n^2 + Ru_n^2 \right)$$

Notice that in our case we have to calculate $\bar{J}$, our final performance index, with our penalty.

With the parameters directly passed to the program A, B, Q, R, x0

Implementation

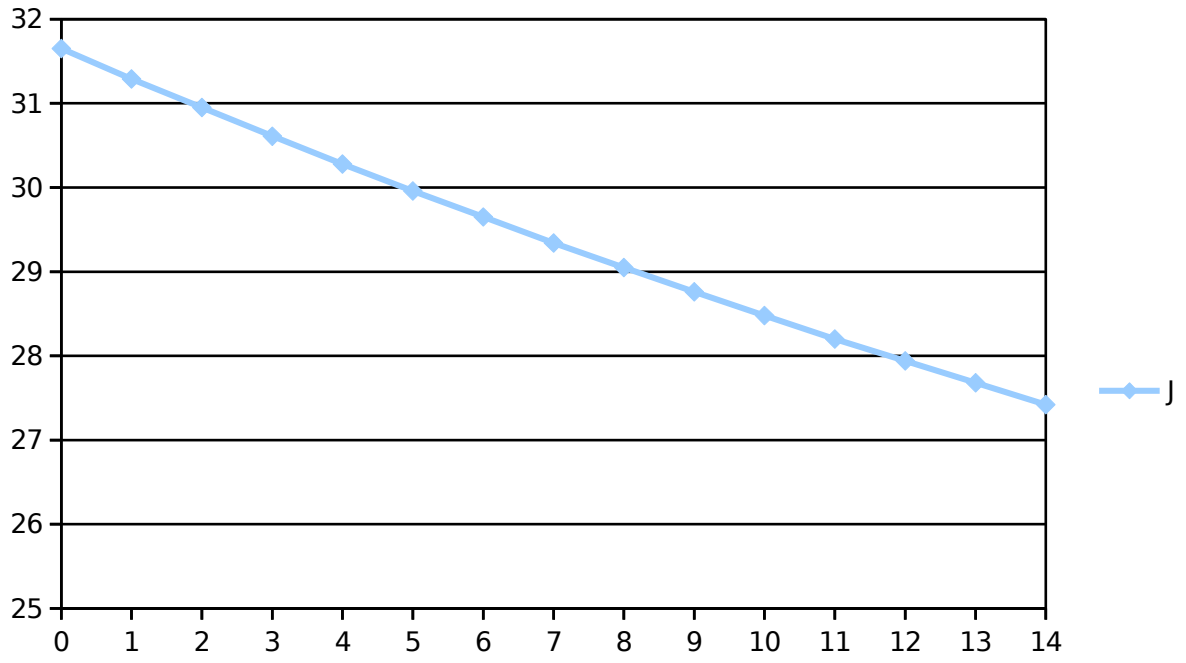
This is the main function, it runs all the procedure of our algorithm

```
def dynamicOptimization(self, A, B, Q, R, x0, n, k):
    t1 = 1.31
    t2 = 2.31
    toptimal = 0.001
    v1 = -1
    v2 = 2
    x = [[0 for x in range(n)] for y in range(k)]
    u = [[0 for x in range(n)] for y in range(k)]
    b = [[0 for x in range(n)] for y in range(k)]
    p = [[0 for x in range(n+1)] for y in range(k)]
    Ji = [0 for x in range(k)]
    u[0] = self.initial
    for i in range(0, k):
        x[i][0] = x0
        for j in range(1, n):
            x[i][j] = x[i][j - 1] + u[i][j - 1]

        for j in reversed(range(0, n+1)):
            if j == n :
                p[i][j] = 0
            else:
                p[i][j] = 2*Q*x[i][j] + p[i][j+1]*2*A*x[i][j]

        for j in range(0, n):
            b[i][j] = 2*R*u[i][j] + t1*(2*R*u[i][j] - 2*R*v1) + t2*(2*R*u[i][j] - 2*R*v2) + p[i][j + 1]
        for j in range(0, n):
           if i + 1 < k:
               u[i + 1][j] = u[i][j] - toptimal * b[i][j]

        sum = 0
        for j in range(n):
            sum += x[i][j]**2 + u[i][j]**2 + t1*(u[i][j] - v1)*max(0, u[i][j] - v1)+ t2*(-1*u[i][j]-v2)*max(0, -1*u[i][j]- v2)
        Ji[i] = sum

        #print("U: ", *u[i] , "x: ", *x[i], "J: ", Ji[i])

    self.U = u
    self.J = Ji
    self.X = x
```

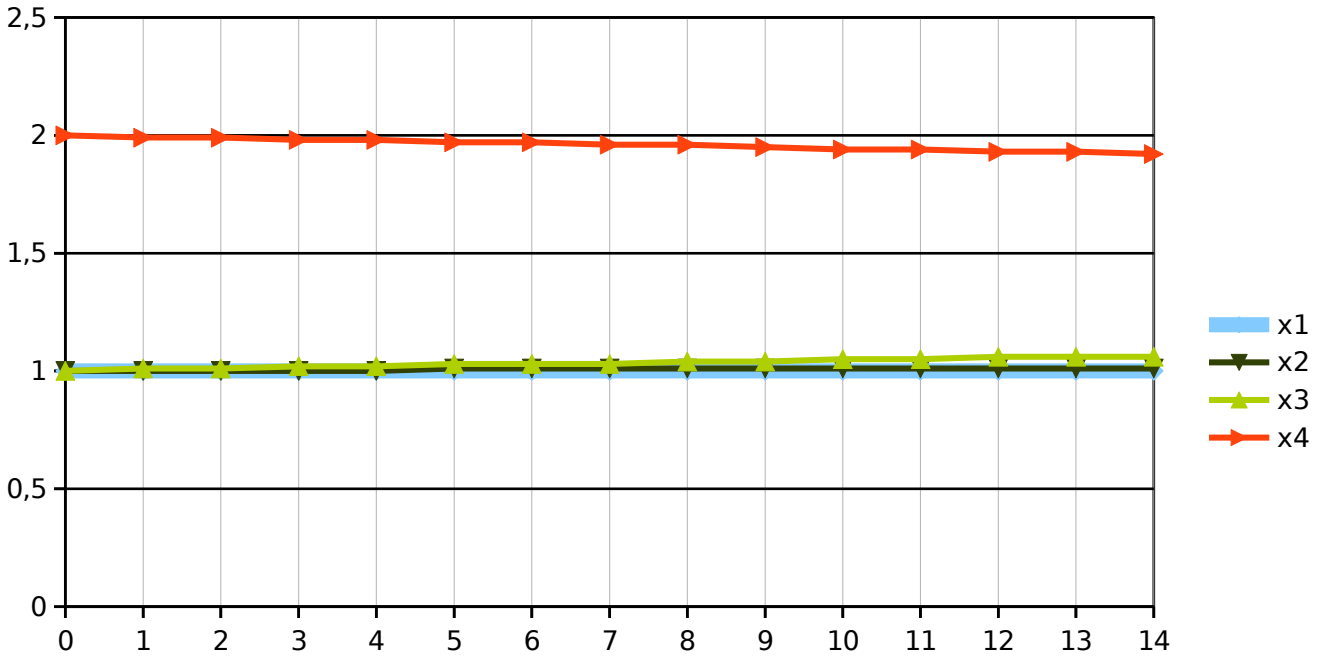
As we can see the code is very similar to the CMDO problem, we had to add the penalty for our J

Results

We run the program under 15 iterations: This the graphic iterations/J:



And this is the graphic for each trajectory:



We can see a gradual minimization of our results for every iteration.