Optimal Control of Unit Mass (New Norms)

ECE367 PS05 Problem 5.6 -- Aman Bhargava

We use the same setup from last time, except with different norms regulating the 'fuel' consumption:

$$||p||_1 = \sum_i |p_i|, \text{ and } ||p||_\infty = \max_i |p_i|$$

Part A

- Consider $||p||_1$ and find optimal solution using computer solver.
- Plot optimal force, position, and velocity.
- What do you observe, and how do they contrast to the l_2 solution?

Part A Answers

We observe a very sparse output vector. The 'engines' fire only once at the beginning and once at the end.

This is very expected from the l_1 norm bec

Part B

- Consider $||p||_\infty$ and find optimal solution using computer solver.
- Plot optimal force, position, and velocity.
- What do you observe, and how do they contrast to the l_2 and l_1 solution? Does it make sense?

Part C

See code below

In [1]:
```
### IMPORT BOX ###

using LinearAlgebra
using Plots
using JuMP
import GLPK
```

In [2]:
```
#######################
# GENERATING MATRIX M #
#######################
ary_size = 10 # Resolution of our control system's control.
M = zeros(ary_size,ary_size)
```

```
row = zeros(1,ary_size)
row[1] = 0.5

for i = 1:ary_size
    M[i,:] = row
    for j = 1:ary_size-1
        row[ary_size-j+1] = row[ary_size-j]
    end
    row[1] += 1
end

M
```

Out[2]:

```
10×10 Array{Float64,2}:
 0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 1.5  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 2.5  1.5  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 3.5  2.5  1.5  0.5  0.0  0.0  0.0  0.0  0.0  0.0
 4.5  3.5  2.5  1.5  0.5  0.0  0.0  0.0  0.0  0.0
 5.5  4.5  3.5  2.5  1.5  0.5  0.0  0.0  0.0  0.0
 6.5  5.5  4.5  3.5  2.5  1.5  0.5  0.0  0.0  0.0
 7.5  6.5  5.5  4.5  3.5  2.5  1.5  0.5  0.0  0.0
 8.5  7.5  6.5  5.5  4.5  3.5  2.5  1.5  0.5  0.0
 9.5  8.5  7.5  6.5  5.5  4.5  3.5  2.5  1.5  0.5
```

In [3]:

```
# Helper matrix to determine speed vector given control force vector #

B = zeros(ary_size,ary_size)

row = zeros(1,ary_size)
row[1] = 1

for i = 1:ary_size
    B[i,:] = row
    for j = 1:ary_size-1
        row[ary_size-j+1] = row[ary_size-j]
    end
end

B
```

Out[3]:

```
10×10 Array{Float64,2}:
 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 1.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 1.0  1.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
 1.0  1.0  1.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0
 1.0  1.0  1.0  1.0  1.0  1.0  0.0  0.0  0.0  0.0
 1.0  1.0  1.0  1.0  1.0  1.0  1.0  0.0  0.0  0.0
 1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  0.0  0.0
 1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  0.0
 1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
```

In [4]:

```
# Creating constraint matrix and running least squares to find smallest l2 norm

A = zeros(2,ary_size)
A[1,:] = M[10,:]
A[2,:] = ones(1,ary_size)

y = [1; 0]

A
```

Out[4]:

```
2×10 Array{Float64,2}:
 9.5  8.5  7.5  6.5  5.5  4.5  3.5  2.5  1.5  0.5
 1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
```

l_1 Solution

In [5]:

```julia
model = Model(GLPK.Optimizer)

clctr = ones(10)

@variable(model, p[1:10])
@variable(model, t[1:10]) # Helper variable

# @objective(model, Min, t[1]+t[2]+t[3]+t[4]+t[5]+t[6]+t[7]+t[8]+t[9]+t[10])
@objective(model, Min, sum(t))

@constraint(model, sum(p) == 0)

@constraint(model, 9.5p[1]+8.5p[2]+7.5p[3]+6.5p[4]+5.5p[5]+4.5p[6]+3.5p[7]+2.5p[

@constraint(model, p .<= t)
@constraint(model, p .>= -t)

@constraint(model, t .>= 0)


print(model)
```

```
Min t[1] + t[2] + t[3] + t[4] + t[5] + t[6] + t[7] + t[8] + t[9] + t[10]
Subject to
 p[1] + p[2] + p[3] + p[4] + p[5] + p[6] + p[7] + p[8] + p[9] + p[10] = 0.0
 9.5 p[1] + 8.5 p[2] + 7.5 p[3] + 6.5 p[4] + 5.5 p[5] + 4.5 p[6] + 3.5 p[7] + 2.
5 p[8] + 1.5 p[9] + 0.5 p[10] = 1.0
 p[1] + t[1] ≥ 0.0
 p[2] + t[2] ≥ 0.0
 p[3] + t[3] ≥ 0.0
 p[4] + t[4] ≥ 0.0
 p[5] + t[5] ≥ 0.0
 p[6] + t[6] ≥ 0.0
 p[7] + t[7] ≥ 0.0
 p[8] + t[8] ≥ 0.0
 p[9] + t[9] ≥ 0.0
 p[10] + t[10] ≥ 0.0
 t[1] ≥ 0.0
 t[2] ≥ 0.0
 t[3] ≥ 0.0
 t[4] ≥ 0.0
 t[5] ≥ 0.0
 t[6] ≥ 0.0
 t[7] ≥ 0.0
 t[8] ≥ 0.0
 t[9] ≥ 0.0
 t[10] ≥ 0.0
 p[1] - t[1] ≤ 0.0
 p[2] - t[2] ≤ 0.0
 p[3] - t[3] ≤ 0.0
 p[4] - t[4] ≤ 0.0
 p[5] - t[5] ≤ 0.0
 p[6] - t[6] ≤ 0.0
 p[7] - t[7] ≤ 0.0
 p[8] - t[8] ≤ 0.0
```

```
 p[9] - t[9] ≤ 0.0
 p[10] - t[10] ≤ 0.0
```

In [6]:

```
optimize!(model)
println("Termination status : ", termination_status(model))
println("Primal status      : ", primal_status(model))
```

```
Termination status : OPTIMAL
Primal status      : FEASIBLE_POINT
```

In [7]:

```
obj_value = objective_value(model)
p = value.(p)
x = M*p
```

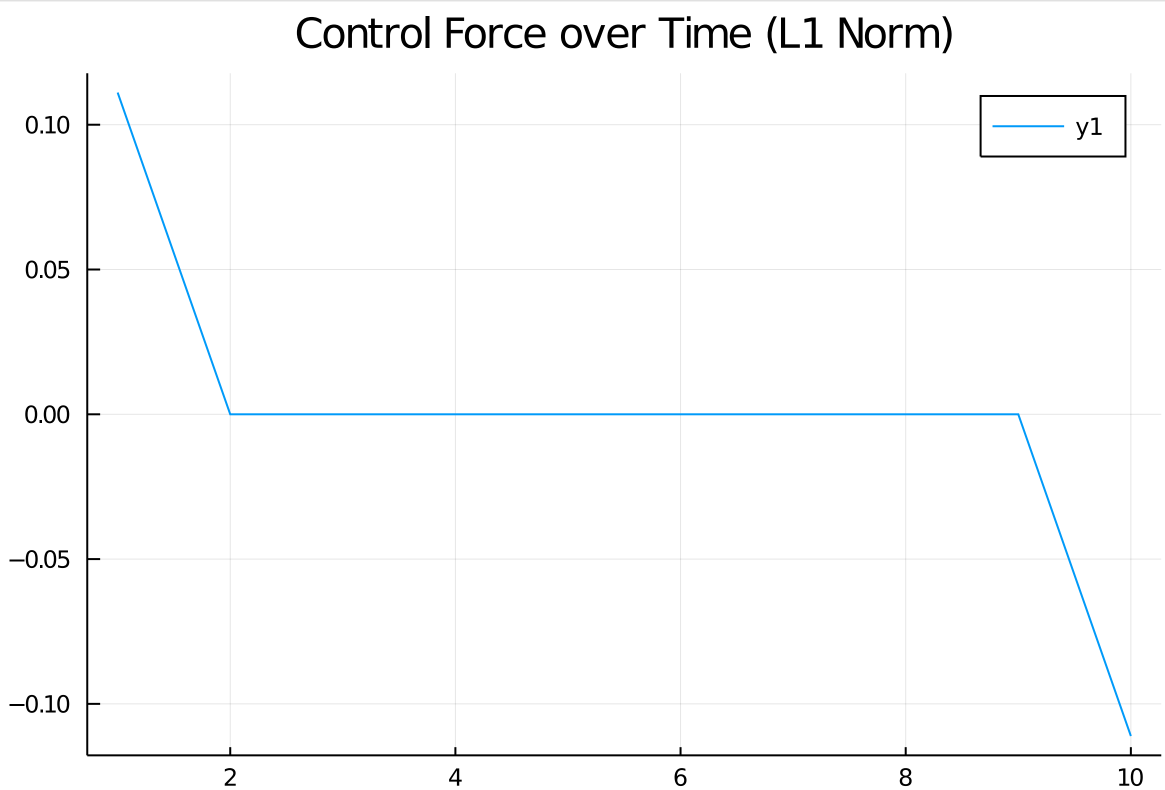
Out[7]:

```
10-element Array{Float64,1}:
 0.05555555555555555
 0.16666666666666666
 0.2777777777777778
 0.38888888888888884
 0.5
 0.611111111111111
 0.7222222222222222
 0.8333333333333333
 0.9444444444444444
 1.0
```

In [8]:

```
Plots.plot(p)
Plots.title!("Control Force over Time (L1 Norm)")
```

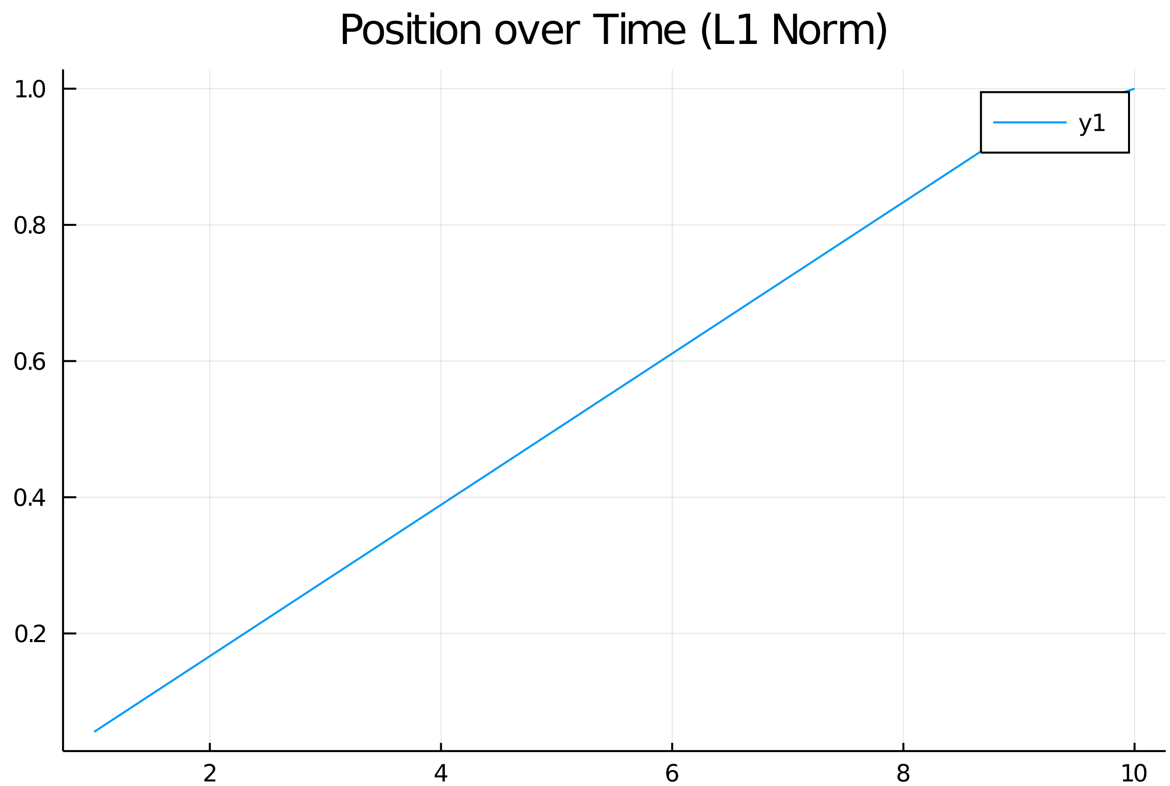
Out[8]:



In [9]:

```
# Plotting position over time (A)

Plots.plot(x)
Plots.title!("Position over Time (L1 Norm)")
```

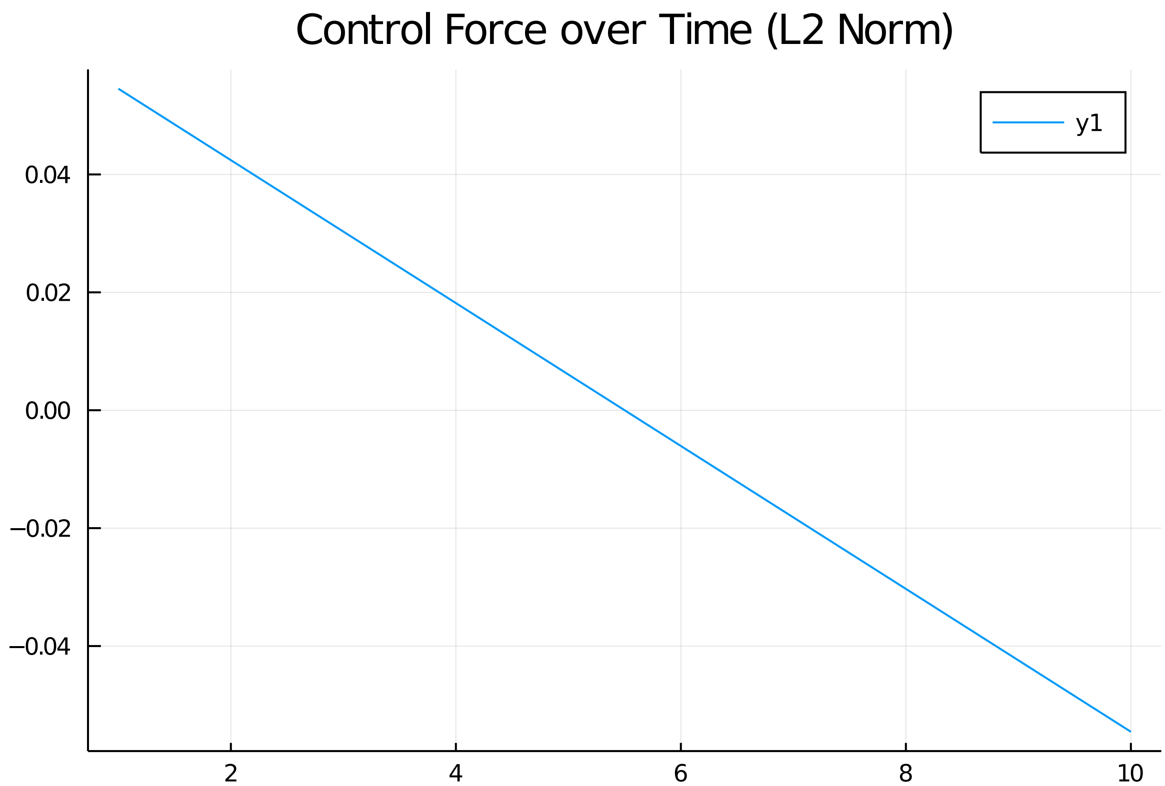
Out[9]:



l_2 Solution

In [10]:
```
# Running least squares to solve for p

p = transpose(A)*inv(A*transpose(A))*y
# Calculating corresponding position vector

x = M*p

# Plotting control force vector over time

Plots.plot(p)
Plots.title!("Control Force over Time (L2 Norm)")
```

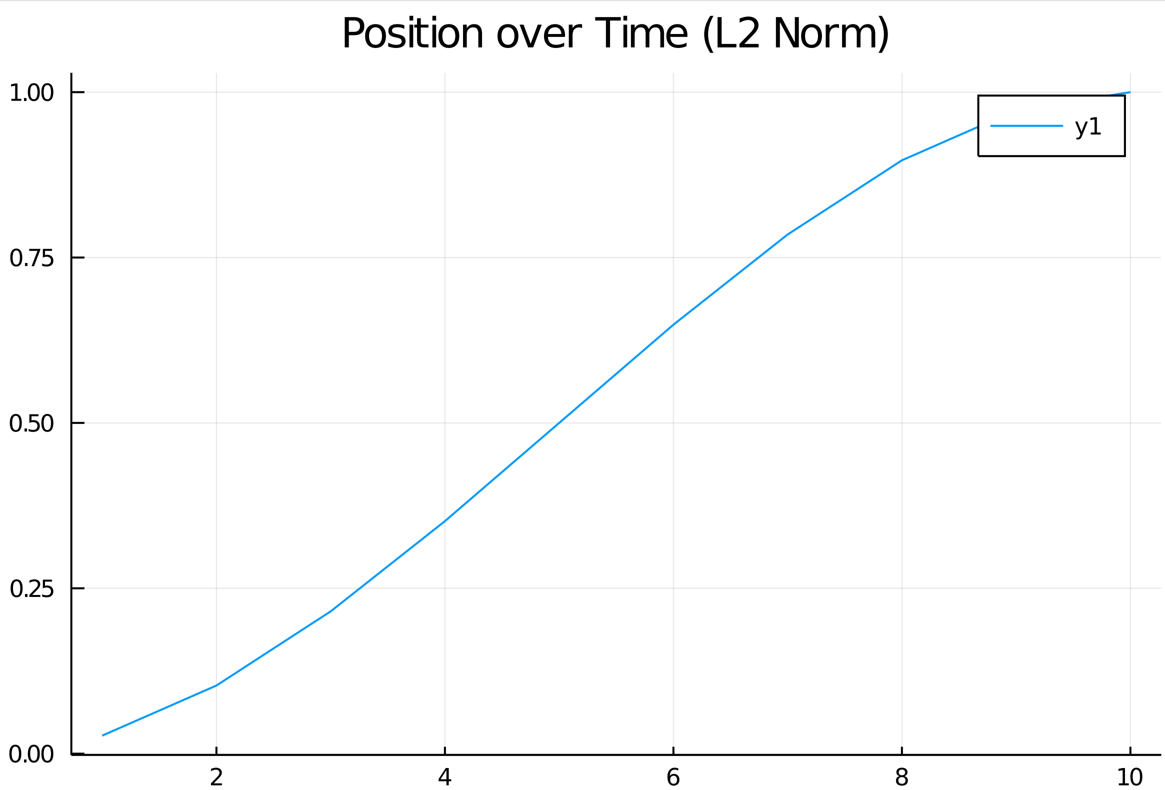
Out[10]:



In [11]:

```
# Plotting position over time (A)

Plots.plot(x)
Plots.title!("Position over Time (L2 Norm)")
```

Out[11]:



l_∞ Solution

In [12]:

```
model = Model(GLPK.Optimizer)

clctr = ones(10)

@variable(model, p[1:10])
@variable(model, t) # Helper variable

@objective(model, Min, t)

@constraint(model, sum(p) == 0)

@constraint(model, 9.5p[1]+8.5p[2]+7.5p[3]+6.5p[4]+5.5p[5]+4.5p[6]+3.5p[7]+2.5p[

@constraint(model, p .<= t)
@constraint(model, p .>= -t)

@constraint(model, t .>= 0)


print(model)
```

```
Min t
Subject to
 p[1] + p[2] + p[3] + p[4] + p[5] + p[6] + p[7] + p[8] + p[9] + p[10] = 0.0
 9.5 p[1] + 8.5 p[2] + 7.5 p[3] + 6.5 p[4] + 5.5 p[5] + 4.5 p[6] + 3.5 p[7] + 2.
5 p[8] + 1.5 p[9] + 0.5 p[10] = 1.0
 p[1] + t ≥ 0.0
 p[2] + t ≥ 0.0
 p[3] + t ≥ 0.0
 p[4] + t ≥ 0.0
 p[5] + t ≥ 0.0
 p[6] + t ≥ 0.0
 p[7] + t ≥ 0.0
 p[8] + t ≥ 0.0
 p[9] + t ≥ 0.0
 p[10] + t ≥ 0.0
 t ≥ 0.0
 p[1] - t ≤ 0.0
 p[2] - t ≤ 0.0
 p[3] - t ≤ 0.0
 p[4] - t ≤ 0.0
 p[5] - t ≤ 0.0
 p[6] - t ≤ 0.0
 p[7] - t ≤ 0.0
 p[8] - t ≤ 0.0
 p[9] - t ≤ 0.0
 p[10] - t ≤ 0.0
```

In [13]:

```
optimize!(model)
println("Termination status : ", termination_status(model))
println("Primal status      : ", primal_status(model))
```

```
Termination status : OPTIMAL
Primal status      : FEASIBLE_POINT
```

In [14]:

```
obj_value = objective_value(model)
p = value.(p)
x = M*p
```

Out[14]:

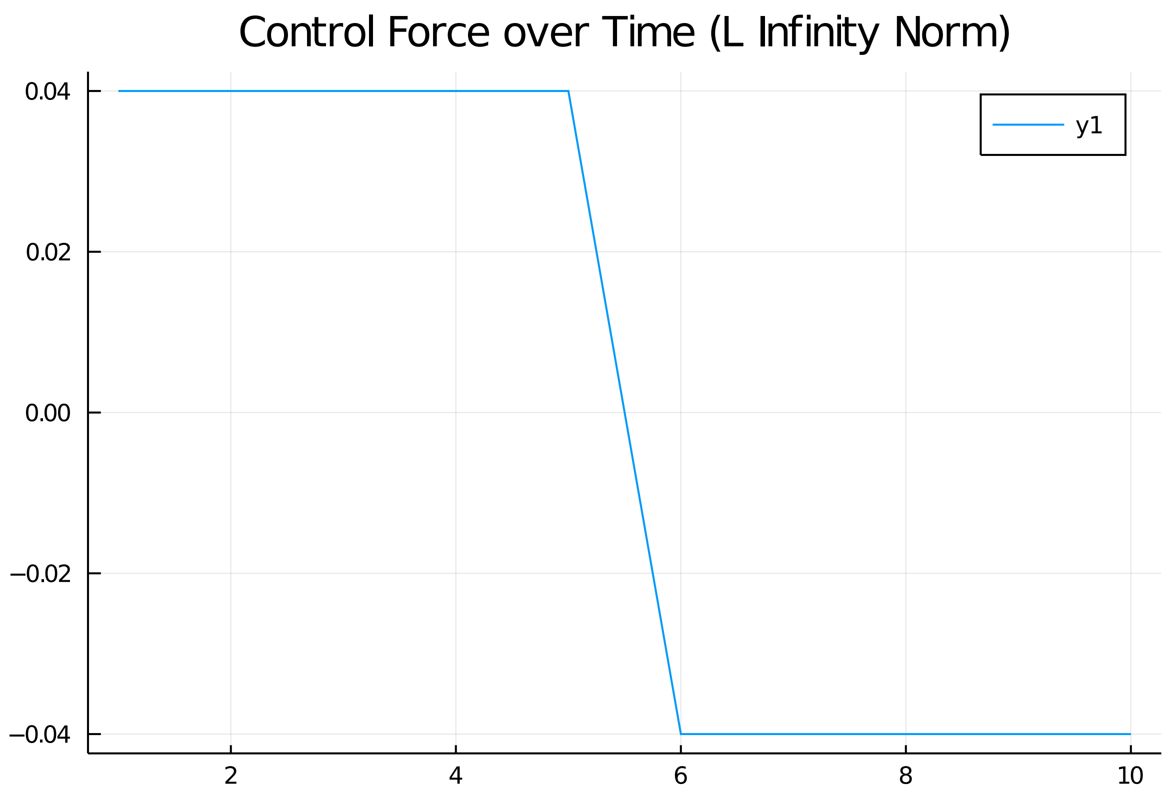
```
10-element Array{Float64,1}:
 0.019999999999999997
 0.07999999999999999
 0.17999999999999997
```

```
0.31999999999999995
0.5
0.6799999999999999
0.82
0.9199999999999998
0.9799999999999995
1.0
```

In [15]:

```
Plots.plot(p)
Plots.title!("Control Force over Time (L Infinity Norm)")
```

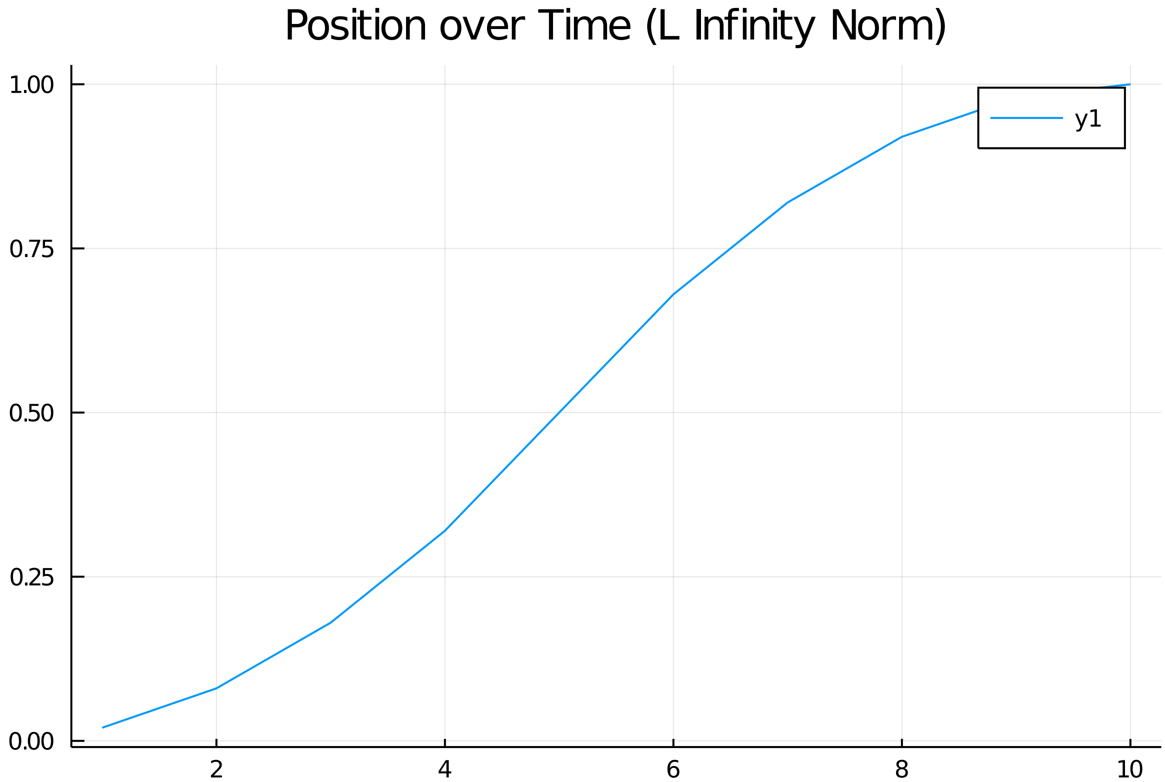
Out[15]:



In [16]:

```
# Plotting position over time (A)

Plots.plot(x)
Plots.title!("Position over Time (L Infinity Norm)")
```

Out[16]:


Position over Time (L Infinity Norm)
1.00
0.75
0.50
0.25
0.00
2
4
6
8
10
y1

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```