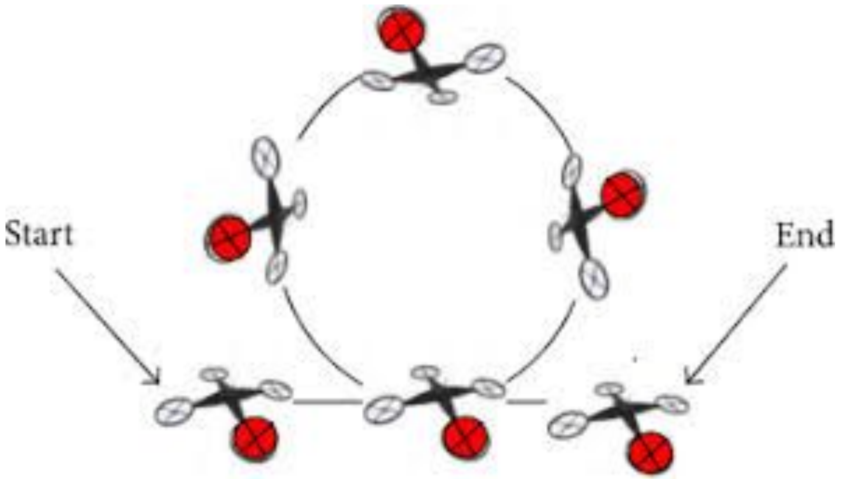
Viva Exam

Gaurav Kumar
190334
AE691A Course Viva
Instructor: Prof. Mangal Kothari

Optimal flip maneuver of variable pitch quadcopter



$$J = \frac{1}{2}\int_0^{T_f} v^2 \, dt$$

co marks

$$\dot{\theta} = q$$

$$J\dot{\omega} + \omega \times J\omega = m$$

$$M = [l, m, n]$$

$$T = \nu k \left(c_{T_1} + c_{T_2} + c_{T_3} + c_{T_4}\right)$$

$$l = \nu k l \left(c_{T_1} - c_{T_2} - c_{T_3} + c_{T_4}\right)$$

$$m = \nu k l \left(c_{T_1} + c_{T_2} - c_{T_3} - c_{T_4}\right)$$

$$n = \nu \frac{kR}{\sqrt{2}} \left(c_{T_1}^{3/2} - c_{T_2}^{3/2} + c_{T_3}^{3/2} - c_{T_4}^{3/2}\right)$$

$$\nu \in \{+1, -1\}$$

$$J = \text{diag}\,(0.012, 0.026, 0.038)$$

$$K = \rho \pi R^2 v_{tip}^2, \quad m = 1.4\,kg$$

$$l = 0.18m, \quad R = 0.14m, \quad \Omega = 418.9\,rad/s$$

Solve it.

Assumptions

1. ψ and ϕ Euler angle is maintained at 0 throughout. Hence,

$$\phi = 0$$

$$\psi = 0$$

$$\dot{\phi} = 0$$

$$\dot{\psi} = 0$$

Using Euler equation

$$
\begin{aligned}
m &= J_y \dot{q} \\
&= J_y \ddot{\theta}
\end{aligned}
$$

Now $\theta(0) = 0$ and $\theta(T) = -\pi$, hence using kinematics

$$
\begin{aligned}
-\pi &= \frac{1}{2}\ddot{\theta}T^2 \\
\Rightarrow \ddot{\theta} &= -\frac{2\pi}{T^2}
\end{aligned}
$$

Final Constraints

$$N = Mg$$

$$\Rightarrow C_{T_1} + C_{T_2} + C_{T_3} + C_{T_4} = \frac{Mg}{k}$$

$$m = J_y \frac{-2\pi}{T^2}$$

$$\Rightarrow C_{T_1} + C_{T_2} - C_{T_3} - C_{T_4} = \frac{-2\pi J_y}{klT^2}$$

OCP

$$J = \frac{1}{2}\int_0^T C_{T_1}^2 + C_{T_2}^2 + C_{T_3}^2 + C_{T_4}^2 \, dt$$

S.T

$$C_{T_1} + C_{T_2} + C_{T_3} + C_{T_4} = \frac{Mg}{k}$$

$$C_{T_1} + C_{T_2} - C_{T_3} - C_{T_4} = \frac{-2\pi J_y}{klT^2}$$

MATLAB R2020a - academic use

Editor - C:\Users\grvfe\Downloads\inverse_pitch.m*

```matlab
k= 1.225*pi*(0.14^4)*(418.9^2);
l=0.18;
I_y= 0.026;
W=1.4*9.8;
C1=W/k;
t=2;
C2=-(2*pi/t^2)*(I_y)/(k*l);
fun = @(x)x(1)^2+x(2)^2+x(3)^2+x(4)^2;
x0=[0;0;0;0];
A=[];
b=[];
Aeq=[1 1 1 1; 1 1 -1 -1];
beq=[C1; C2];
x=fmincon(fun,x0,A,b,Aeq,beq);
```

Command Window

```
>> inverse_pitch

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decre
feasible directions, to within the value of the optimality toleran
and constraints are satisfied to within the value of the constrain

<stopping criteria details>

x =

    0.0130
    0.0130
    0.0134
    0.0134

>>
```

Workspace

Name	Value
A	[]
Aeq	[1,1,1,1;1,1,-1,-1]
ans	1x1 sym
b	[]
beq	[0.0529;-8.7458e...
C1	0.0529
C2	-8.7458e-04
fun	@(x)x(1)^2+x(2)^2...
I_y	0.0260
k	259.4292
l	0.1800
t	2
W	13.7200
x	[0.0130;0.0130;0....
x0	[0;0;0;0]

Analytical Solution assuming variable pitch

$$J = \frac{1}{2}\int_0^T C_{T_1}^2 + C_{T_2}^2 + C_{T_3}^2 + C_{T_4}^2 dt$$

$$s.t$$

$$\int_0^T \left(C_{T_1} + C_{T_2} + C_{T_3} + C_{T_4} - \frac{Mg}{kT} \right) = 0$$

$$\int_0^T \left(\int_0^T \left(\frac{kl}{J_y} (C_{T_1} + C_{T_2} - C_{T_3} - C_{T_4}) \right) dt + \frac{\pi}{T} \right) dt = 0$$

$$using$$

$$\int_0^T q dt = -\pi$$

Augmented Lagrangian

$$L = \left(C_{T_1}^2 + C_{T_2}^2 + C_{T_3}^2 + C_{T_4}^2\right)$$
$$- \lambda \left(C_{T_1} + C_{T_2} + C_{T_3} + C_{T_4} - \frac{Mg}{kT}\right)$$
$$- \mu \left(\int_0^T \left(\frac{kl}{J_y}(C_{T_1} + C_{T_2} - C_{T_3} - C_{T_4})\right) dt + \frac{\pi}{T}\right)$$

Optimality Conditions

$$\frac{\partial L}{\partial C_T} = 0$$

$$\Rightarrow \begin{pmatrix} 2C_{T_1} \\ 2C_{T_2} \\ 2C_{T_3} \\ 2C_{T_4} \end{pmatrix} - \lambda \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} - \mu \begin{pmatrix} \frac{Tkl}{J_y} \\ \frac{Tkl}{J_y} \\ -\frac{Tkl}{J_y} \\ -\frac{Tkl}{J_y} \end{pmatrix} = 0$$

$$\frac{\partial L}{\partial \lambda} = 0$$

$$\Rightarrow \left(C_{T_1} + C_{T_2} + C_{T_3} + C_{T_4} - \frac{Mg}{kT} \right) = 0$$

$$\frac{\partial L}{\partial \mu} = 0$$

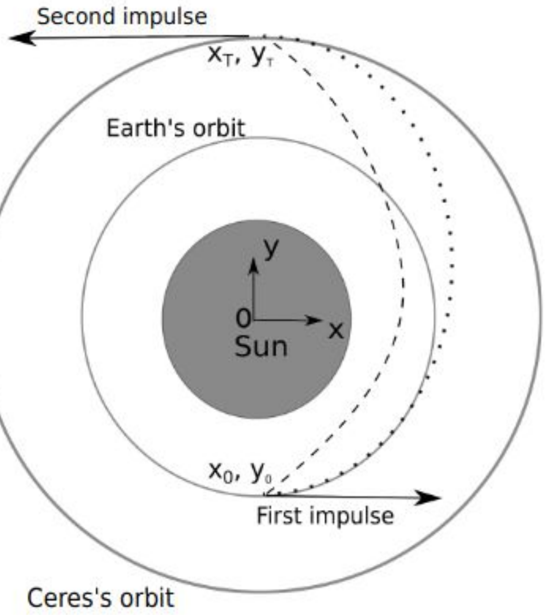
$$\Rightarrow \left(\int_0^T \left(\frac{kl}{J_y} (C_{T_1} + C_{T_2} - C_{T_3} - C_{T_4}) \right) dt + \frac{\pi}{T} \right) = 0$$

Solution

$$C_{T_1} = C_{T_2} = \frac{Mg}{4k} - \frac{\pi J_y}{4klT} = 0.0130$$

$$C_{T_3} = C_{T_4} = \frac{Mg}{4k} + \frac{\pi J_y}{4klT} = 0.0134$$

Optimal Orbit Transfer



OCP Formulation

$$J = \frac{1}{2}\int_0^T a_x^2(t) + a_y^2(t)dt$$

$$s.t$$

$$\dot{x} = v_x$$

$$\dot{y} = v_y$$

$$\dot{v}_x = -\frac{GM_{sun}x}{r^3} + a_x(t)$$

$$\dot{v}_y = -\frac{GM_{sun}y}{r^3} + a_y(t)$$

$$r > R_{sun}$$

$$where$$

$$r = \sqrt{x^2 + y^2}$$

Activities   Sublime Text   Nov 26 01:06

~/ACADOtoolkit/examples/my_examples/ae691_ass3.cpp - Sublime Text (UNREGISTERED)

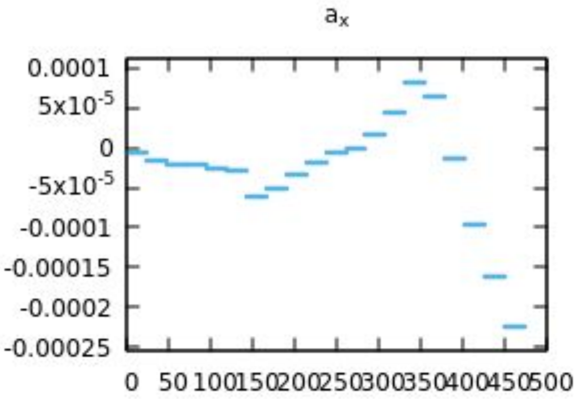
File Edit Selection Find View Goto Tools Project Preferences Help

ae691_ass3.cpp   ghp_jaerCLbDuYPlZ3XxAfV9yluP2NCgXc01MvvP

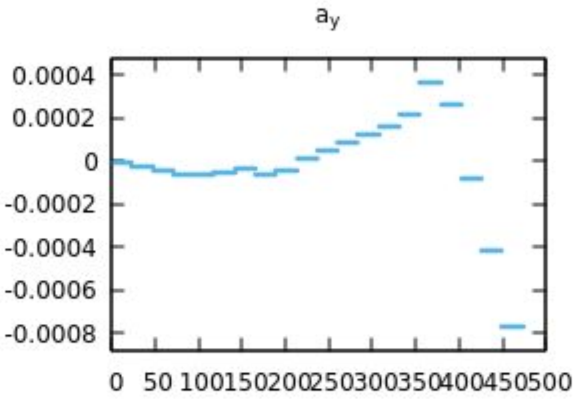
```cpp
#include <acado_toolkit.hpp>
#include <acado_optimal_control.hpp>
#include <acado_gnuplot.hpp>
int main( ){

    USING_NAMESPACE_ACADO

    // INTRODUCE THE VARIABLES:
    // -------------------------
    DifferentialState     x,y,v_x,v_y;
    Control               a_x,a_y  ;
    DifferentialEquation  f   ;
// Constant variables
    const double au = 1.5e8;
    const double G = 1.4761e-34;
    const double M_sun = 2.0e30;
    // DEFINE A DIFFERENTIAL EQUATION:
    // -------------------------------
    f << dot(x) == v_x;
    f << dot(y) == v_y;
    f << dot(v_x) ==(-(G*M_sun*x)/pow(x*x+y*y,(3/2))) + a_x ;
    f<< dot(v_y) == (-(G*M_sun*y)/pow(x*x+y*y,(3/2))) + a_y;
    // DEFINE AN OPTIMAL CONTROL PROBLEM:
    // ----------------------------------
    OCP ocp( 0.0, 473);
    ocp.minimizeLagrangeTerm( a_x*a_x + a_y*a_y  );  // weight this with the physical cost!!!
    ocp.subjectTo( f );

    ocp.subjectTo( AT_START, x ==  0.0 );
    ocp.subjectTo( AT_START, y == -1.0 );
    ocp.subjectTo( AT_START, v_x == 0.01728 );
    ocp.subjectTo( AT_START, v_y == 0.0 );
    ocp.subjectTo( AT_END, x ==  0.0 );
    ocp.subjectTo( AT_END, y == 2.77 );
    ocp.subjectTo( AT_END, v_x == -0.010368 );
    ocp.subjectTo( AT_END, v_y == 0.0 );
    ocp.subjectTo(0.00465047<=pow(x*x+y*y,(1/2))<=4);
//ocp.subjectTo( Sfmin <= Sf <= Sfmax );


    // DEFINE A PLOT WINDOW:
    // ---------------------
    GnuplotWindow window;
        //window.addSubplot( x ,"x"  );
       //window.addSubplot( y ,"y"  );
        //window.addSubplot( v_x ,"v_x"  );
        //window.addSubplot( v_y,"v_y" );
        window.addSubplot( a_x,"a_x" );
        window.addSubplot(a_y,"a_y");
      window.addSubplot(sqrt(a_x*a_x + a_y*a_y), "a");


    // DEFINE AN OPTIMIZATION ALGORITHM AND SOLVE THE OCP:
    // ---------------------------------------------------
    OptimizationAlgorithm algorithm(ocp);
    algorithm.set( KKT_TOLERANCE, 1e-8 );
    algorithm << window;

    algorithm.solve();
   algorithm.getControls("controls.txt");
    return 0; }
```

Line 62, Column 1   stable (22)   Spaces: 4   C++

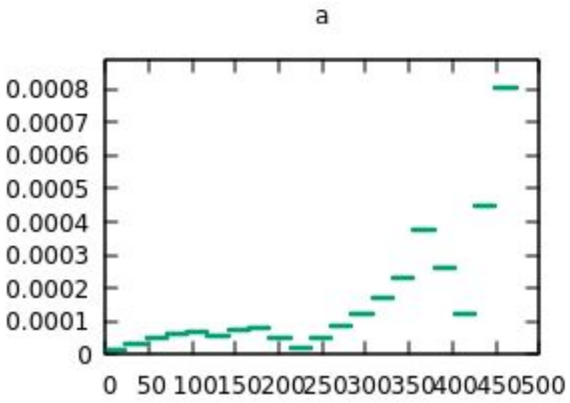
Plots



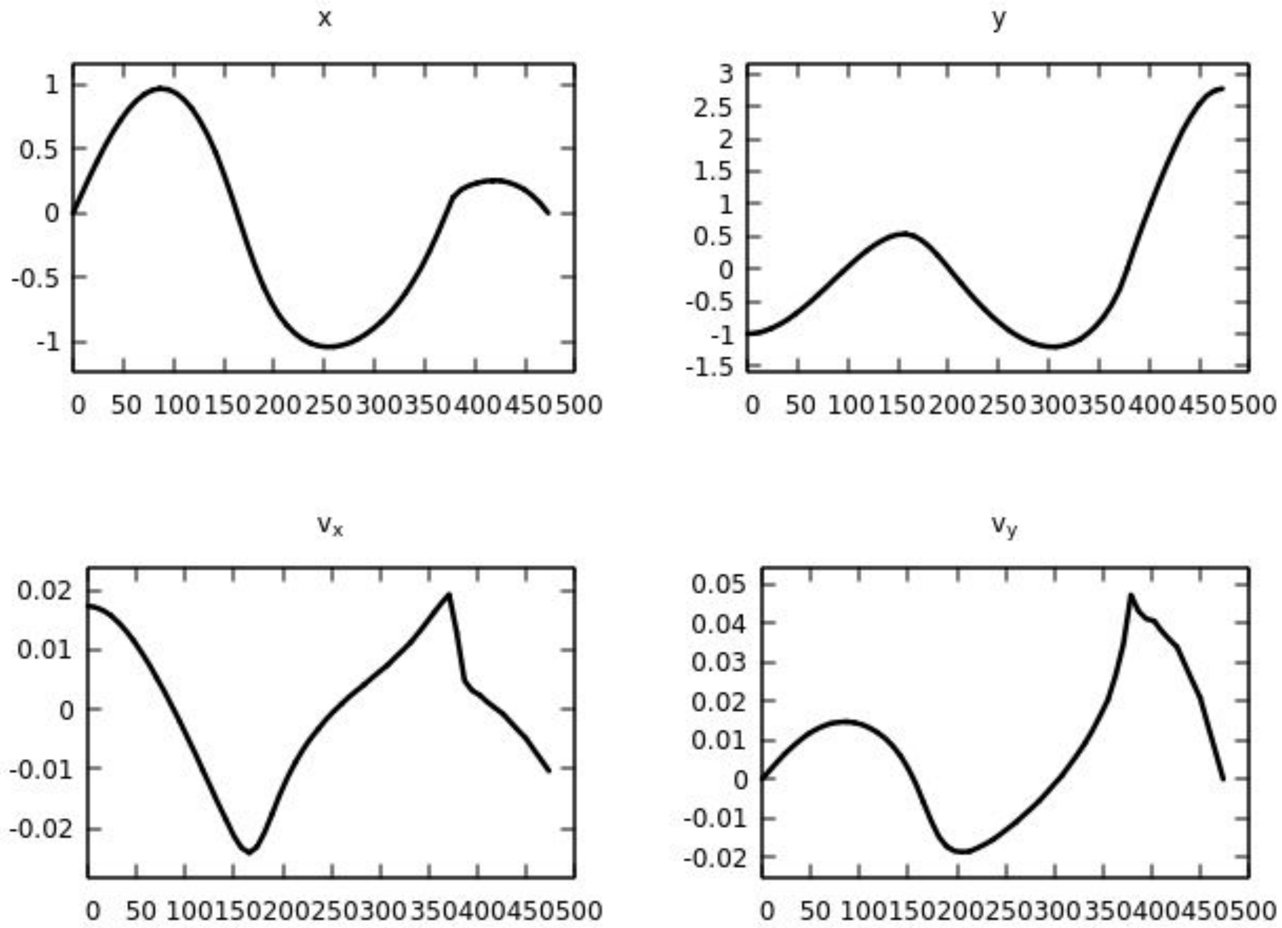
a_x
0.0001
5x10-5
0
-5x10-5
-0.0001
-0.00015
-0.0002
-0.00025
0 50 100 150 200 250 300 350 400 450 500



a_y
0.0004
0.0002
0
-0.0002
-0.0004
-0.0006
-0.0008
0 50 100 150 200 250 300 350 400 450 500



a
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0
0 50 100 150 200 250 300 350 400 450 500


x
1
0.5
0
-0.5
-1
0 50 100 150 200 250 300 350 400 450 500
y
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0 50 100 150 200 250 300 350 400 450 500
v_x
0.02
0.01
0
-0.01
-0.02
0 50 100 150 200 250 300 350 400 450 500
v_y
0.05
0.04
0.03
0.02
0.01
0
-0.01
-0.02
0 50 100 150 200 250 300 350 400 450 500

Comparison with Hoffman Transfer

1. **Δ**v for Optimal trajectory: -17.567
2. **Δ**v for Hoffman Transfer: -29.2191

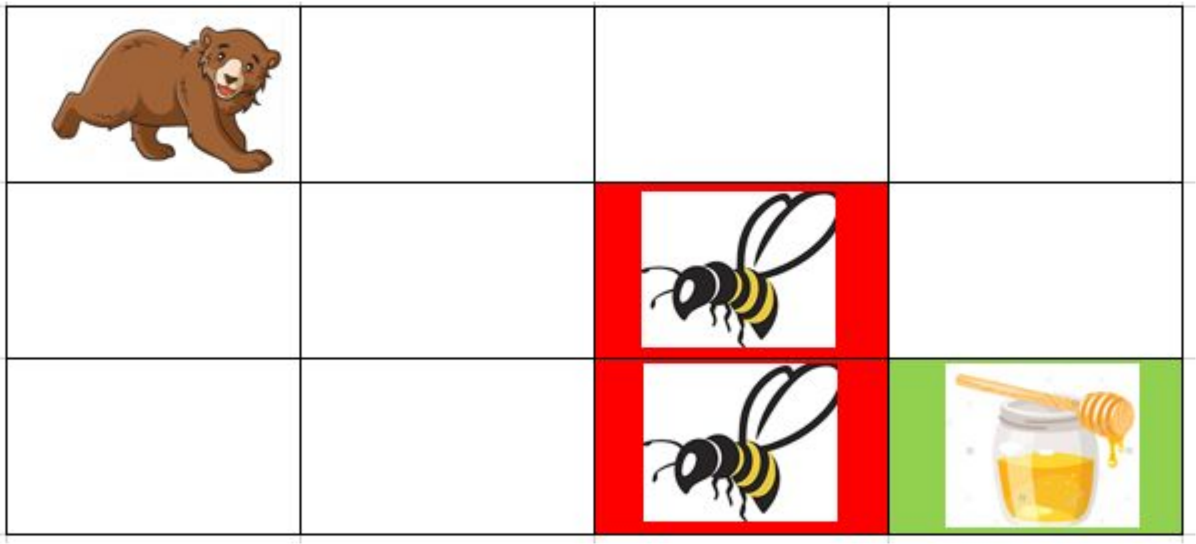
$$\sqrt{1.4761 \cdot \frac{10^{-34} \cdot 2 \cdot 10^{30}}{4.26352 \cdot 10^{-5}}} \left(\sqrt{2 \cdot 3.13995111 \cdot \frac{\quad}{3.1399}} \right. = -1.65675772$$

$$\sqrt{1.4761 \cdot \frac{10^{-34} \cdot 2 \cdot 10^{30}}{3.13995111 \cdot 10^{-6}}} \left(1 - \sqrt{2 \cdot 4.26352 \cdot 10} \right. = -27.56238418$$

$$-1.65675772 + -27.56238418 = -29.2191419$$

Value Iteration

Horizontal Move: 80% Probability
Vertical Move : 20% Probability



Pseudo Code

Initialize V arbitrarily, e.g.,$V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
    $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

MATLAB R2020b - academic use

Editor - /home/grvk/ae691/value_iter.m

value_iter.m | next_state.m | inverse_pitch.m

```matlab
gamma = 0.9;
delta= 0.001;
transition_Hori = 0.40;
transition_Verti = 0.10;
states = zeros(3,4);
for i=1:3
    for j=1:4
        states(i,j)=10*i+j;
    end
end
rewards=zeros(3,4);
for i=1:3
    for j=1:4
        rewards(i,j)=0;
    end
end
rewards(3,4)=1;
rewards(2,3)=-1;
rewards(3,3)=-1;
actions = py.list({{'D', 'R'},{'D', 'R', 'L'},{'D', 'L', 'R'},{'D', 'L'}, {'D', 'U', 'R'},{'D', 'R', 'L', 'U'},{},{'D', 'L', 'U'},{'U', 'R'},{'U', 'L', 'R'},{},{}});
A = reshape(states',[1,12]);
V=rewards;
iter=0;
transition = 0;
while true
    theta=0;
    iter=iter+1;
    for i=1:3
        for j=1:4
            c1 = i == 2 && j==3;
            c2= i==3 && j==3;
            c3 = i==3 && j==4;
            if  c1||c2|| c3
            else
                maxi=0;
                old_v = V(i,j);
                v_m=0;
                k=find(A == 10*i+j);
                for m=1:length(actions{k})
                    [p,q]=next_state(i,j,actions{k}{m});
                    if actions{k}{m}=='D' || actions{k}{m}=='U'
                        transition = transition_Verti;
                    end
                    if actions{k}{m}=='L' || actions{k}{m}=='R'
```

Command Window

```
>> value_iter
>> V

V =

    0.0008    0.0022    0.0062    0.0171
    0.0001    0.0020   -1.0000    0.1900
    0.0000    0.0015   -1.0000    1.0000

>>
```

Editor - /home/grvk/ae691/next_state.m

```
function[p,q] = next_state(i,j,ch)
    if ch == 'D'
        p = i+1;
        q=j;
    end
    if ch == 'U'
        p = i-1;
        q=j;
    end
    if ch == 'R'
        q = j+1;
        p=i;
    end
    if ch == 'L'
        q = j-1;
        p=i;
    end
end
```

Command Window

```
>> value_iter
>> V

V =

    0.0008    0.0022    0.0062    0.0171
    0.0001    0.0020   -1.0000    0.1900
    0.0000    0.0015   -1.0000    1.0000

>>
```

Final Policy

0.0008 →	0.0022 →	0.0062 →	0.0171 ↓
0.0001 →	0.0020 ↑	-1.0000	0.1900 ↓
0.0000 →	0.0015 ↑	-1.0000	1.0000

THANK YOU