A Toy Model for Computing of Viability Kernel: Car on the Hill

Rüştü Erciyes Karakaya

August 2022

1 Problem Definition

Figure 1: Landscape

As seen in Figure-1, there is a car on the platform and x_1 = s and g(s) are used for its the *longitudinal and vertical position* on the hill, respectively, and $x_2 = \dot{s}$ stands for the its *longitudinal velocity*. Then the landscape can be represented by the parametric function as follows:

$$g(s) = \frac{\frac{-1.1}{1.2}cos(1.2s)+\frac{1.2}{1.1}cos(1.1s)}{2}, \text{ when State Vector := x=} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} s \\ \dot{s} \end{pmatrix}$$

After these, we give the initial states for the car and then observe what's happening at finite time. Of course, this observation depends on some set of ODEs which we call *Evolution Function*; namely, evolution function determine what's happening at finite time and we need this function with initial state for observing the simulation. In this toy problem, our *evolution function* is the following:

$$f(x(t),u) = \dot{x} = \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} x_2 \\ -9.81sin(\frac{dg(x_1)}{dx_1}) - 0.7x_2 + u \end{pmatrix}, \; u \; \epsilon \text{ [-2,2]}$$

As can observe from state vector, derivation of x_1 is x_2, however the derivation of x_2 requires more complicated explanation. Actually, $\dot{x_2}$ is the derivation of velocity, namely acceleration, $\ddot{s}$, and the coefficients -9.81 and -0.7 is related with gravity and friction of wind respectively. As a last, u is the control parameter[1], which can be considered as gear in this case and it can intervene the

[1] Actually, this can give the idea for optimal control theoretical aspect of the problem.

acceleration in a certain extent and determined by user.
At the end of the day; we have nonlinear dynamical system, state vector and control input in general for the all this type of problems.

Aim: The car must stay on landscape, i.e $s \in [0, 12]$.
Hint: Namely, you have to solve initial value problem for nonlinear dynamical system $f(x(t), u)$ and check x_1 whether stay between 0 and 12 during the process of evolution at a finite time.

2 Method

Actually, we have infinitely many x_1 and x_2 as initial state for the IVP problem of $f(x(t), u)$; that's why we select x_1 between 0 and 12, and x_2 between -6 and 6 with a specific step size. The Figure-2 below means that if evolution function starts with the initial states in the green-region then it will be stayed on landscape during given finite time and the remaining part represents the case that car drop out the landscape, which is blue (or red). Initial states in the Figure-2 are the points of (x_1, x_2) and as stated, we can not solve infinitely many IVP Problem but instead, we can follow the the policy below as a numerical method to determine outcome of the IVP for the given dynamical system as colour coded map like figure-II. The policy is the following:[2][3]

We execute the nonlinear dynamical evolution f(x(t):= initial state, u) at finite time and observe,

• if $f(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, u)$, $f(\begin{pmatrix} x_1+1 \\ x_2 \end{pmatrix}, u)$,$f(\begin{pmatrix} x_1 \\ x_2+1 \end{pmatrix}, u)$,$f(\begin{pmatrix} x_1+1 \\ x_2+1 \end{pmatrix}, u)$, all them, stay on landscape;
then we can assume that the finite time process $f(x(t), u)$ with the selected initial states-points from the square, which is shaped by these four points above, will stay on the landscape and this square colored as green.

• if $f(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, u)$, $f(\begin{pmatrix} x_1+1 \\ x_2 \end{pmatrix}, u)$,$f(\begin{pmatrix} x_1 \\ x_2+1 \end{pmatrix}, u)$,$f(\begin{pmatrix} x_1+1 \\ x_2+1 \end{pmatrix}, u)$, all them, drop out of the landscape;
then we can assume that the finite time process $f(x(t), u)$ with the selected initial states-points from the square, which is shaped by these four points above, will out of the landscape and this square colored as blue.

• Else; set the step size, like h=0.1, check $f(x(t), u)$ with all points x=$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ from x_1:h:x_1+1 and x_2:h:x_2+1 and determine whether is colored as blue or green.

[2]You can see the algorithm design and implementation clearly in MATLAB part, at end of the document.

[3]In this case we set step size as 0.1 and used unit square for assumption; however, they can be set accordingly for precision but time of computation must also be considered.

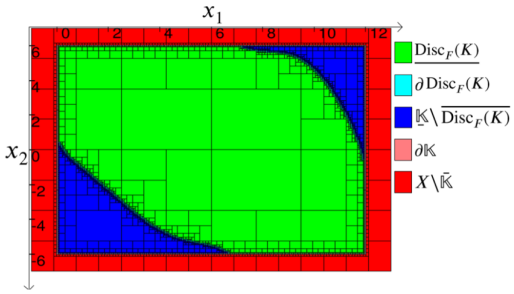


Figure 2: Colour Coded Map: If evolution function starts with the initial states in the green-region then it will be stayed on landscape at given finite time and the remaining part represents the case that car drop out the landscape, which is blue (or red).[5]

Important issue for Numerical Approach: In some cases; there is a huge time consumption, if we attack on the problem by using this algorithm. Instead of these, there are some algorithms to reduce time consumption for creating color coded map.

Suggestion: Lyapunov Function based approach, the policy is the following:

How to find a lyapunov function

We choose a particular control $\mathbf{u} \in \mathbb{U}$. $\mathcal{S}_{\mathbf{u}}$: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ is an autonomous system.
$\mathbf{x}^*$ is an equilibrium point of $\mathcal{S}_{\mathbf{u}} \iff \mathbf{f}(\mathbf{x}^*, \mathbf{u}) = \mathbf{0}$.

- Linearize $\mathcal{S}_{\mathbf{u}}$ around $\mathbf{x}^*$, we get $\mathcal{S}_{\mathbf{u}}^{\mathbf{x}^*}$ defined by $\dot{\tilde{x}} = A\tilde{\mathbf{x}}, \tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$.
- Solve $A^T W + WA = -I$, where W is the unknown amount.
- Check whether W is positive definite.
- If W is positive definite, then $\frac{1}{2}\tilde{\mathbf{x}}^T W \tilde{\mathbf{x}}$ is a Lyapunov function for the linear system, and $\mathbf{x}^*$ is stable.

If we do not find a Lyapunov function for $\mathcal{S}_{\mathbf{u}}^{\mathbf{x}^*}$, we compute the linear system $\mathcal{S}_{ctrl}$ for which $\mathbf{x}^*$ is a stable equilibrium point.

Figure 3: It's taken from *Computing a guaranteed approximation of the viability kernel* Lecture Slides, SMART 2015.

Duality: If we try to obtain color-coded map by using this policy, then we will get most of the green region-but not entire-in a very quick way, however,

[5] Peng, Bowen. (2019). Computation of Discriminating Kernel and Robust Capture Basin with Regulation Map by Interval Methods.

as I stated we loss the region in this case. You can compare the Figure-4 and Figure-2 to see this duality properly.

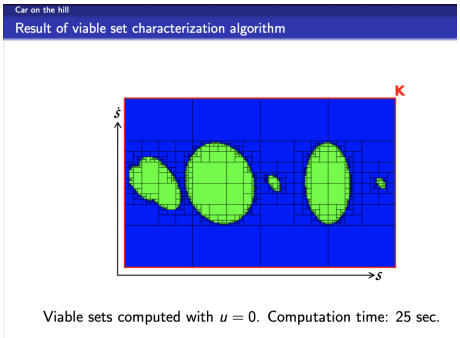


Figure 4: Expected outcome from Lyapunov Function based approach.[6]

3 Viability Theoretical Aspects

What's the viability theory, its aim:[7]

- Viability theory is an area of mathematics that studies the evolution of dynamical systems under constraints on the system state.
- One aim of viability theory is to regulate evolutions under uncertainty in order not only to reach a target in finite time, but also to fulfill constraints (known as viability) until this time.

Some essential definitions:

Viability Kernel:= $Viab_S K$=$\{x(t) \mid \forall t \geq 0, x(t) \epsilon K\}$,
Viability Zone:=K $\subset R^n$, Evolutionary System:=S.

Relation with the our problem: S is the evolutionary system, or nonlinear dynamical system which is $f(x(t), u)$ in our case and its dimension is 2 consist of (x_1,x_2). In this situation, our initial states for starting simulation of system are selected from K $\subset R^2$, namely K is the set (or region) of our initial states. For example, you can consider the K as all rectangle region of Figure-2. After that, obviously, $Viab_S K$ is becoming a green region of Color-coded map, this is the Viable set-kernel (or region).

[6] Computing a guaranteed approximation of the viability kernel Lecture Slides, SMART.
[7] ...taken from document prepared by Prof.Ozgur Ercetin from Sabanci University.

4 APPENDIX

Literature suggestions:

• Monnet, Dominique Ninin, Jordan Jaulin, Luc. (2016). Computing an Inner and an Outer Approximation of the Viability Kernel.

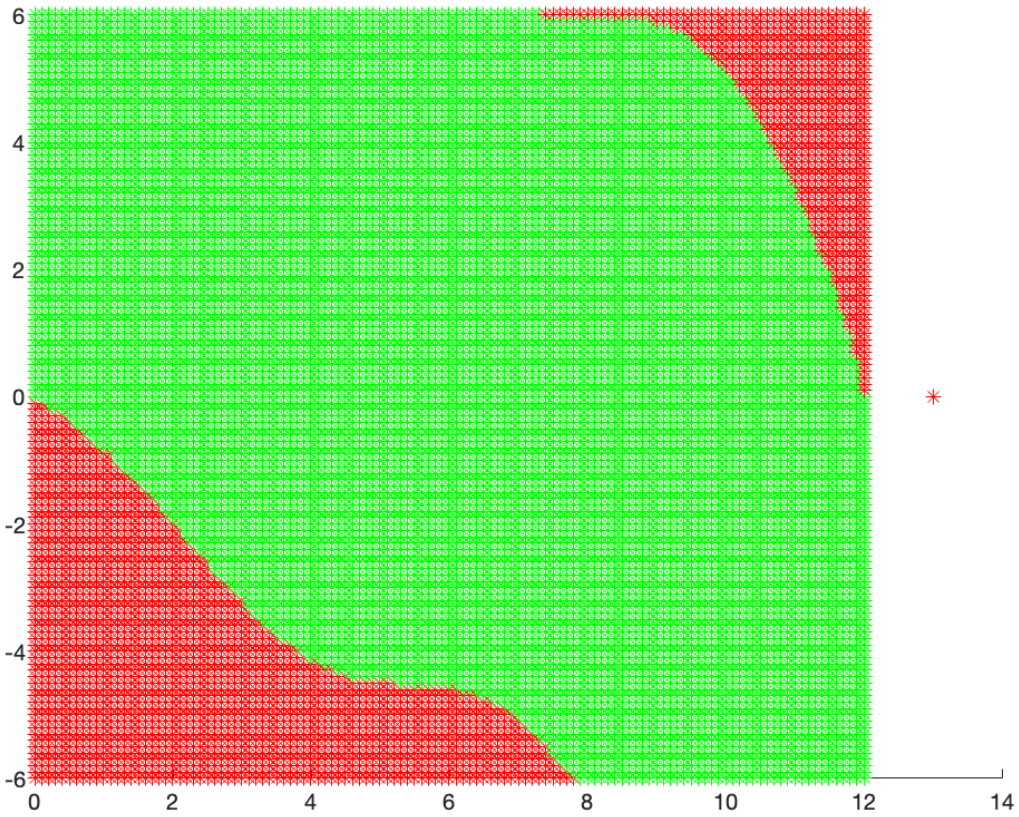
• Peng, Bowen. (2019). Computation of Discriminating Kernel and Robust Capture Basin with Regulation Map by Interval Methods. International Journal of Robust and Nonlinear Control. 29. 10.1002/rnc.4577.

• Aubin, Jean-Pierre and Bayen, Alexandre and Saint-Pierre, Patrick. (2011). Viability Theory: New Directions.

For the MATLAB part of the APPENDIX, please visit the following pages!

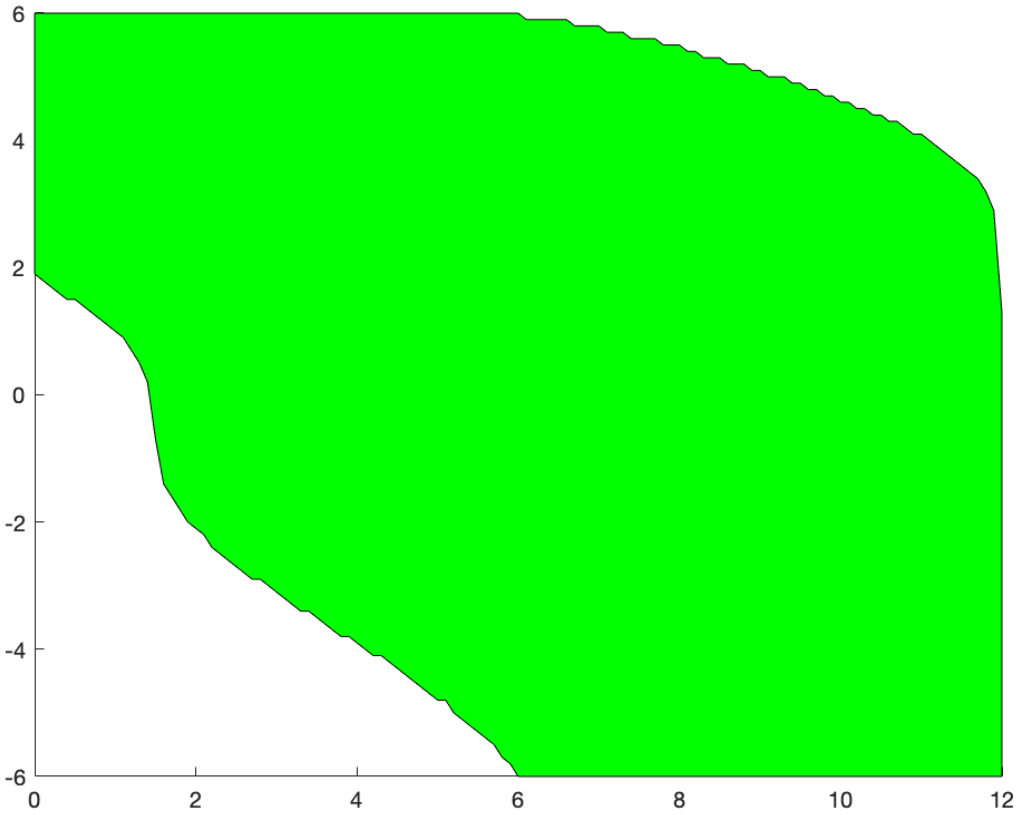
```
%computing the viab kernel by numerical approach
tic
tspan= 0:0.1:30;
%%{
hold all
x1=0;x2=-6;
while x1<12
    x2=-6;
    while x2<6
        [t,y1] = ode45(@vdp_Car,tspan,[x1;x2]);
        [t,y2] = ode45(@vdp_Car,tspan,[x1;x2+1]);
        [t,y3] = ode45(@vdp_Car,tspan,[x1+1;x2]);
        [t,y4] = ode45(@vdp_Car,tspan,[x1+1;x2+1]);
        A1=(0>y1(:,1)); B1=(y1(:,1)>12);
        A2=(0>y2(:,1)); B2=(y2(:,1)>12);
        A3=(0>y3(:,1)); B3=(y3(:,1)>12);
        A4=(0>y4(:,1)); B4=(y4(:,1)>12);
        if A1+B1+A2+B2+A3+B3+A4+B4==0
            bool=true;m=10*x1+1;
            for i=x1:0.1:x1+1
                n=10*(x2+6)+1;
                for j=x2:0.1:x2+1
                    plot(i,j,'*',"Color",'g');
                    table(m,n)=bool;
                    n=n+1;
                end
                m=m+1;
            end
        elseif ((A1+B1~=0)&(A2+B2~=0)&(A3+B3~=0)&(A4+B4~=0))
            bool=false;m=10*x1+1;
            for i=x1:0.1:x1+1
                n=10*(x2+6)+1;
                for j=x2:0.1:x2+1
                    plot(i,j,'*',"Color",'r');
                    table(m,n)=bool;
                    n=n+1;
                end
                m=m+1;
            end
        else
            m=10*x1+1;
            for i=x1:0.1:x1+1
                n=10*(x2+6)+1;
                for j=x2:0.1:x2+1
                    [t,y] = ode45(@vdp_Car,tspan,[i;j]);
```

```
                    A=(0>y(:,1)); B=(y(:,1)>12);
                    bool=false;
                    if A+B==0
                        bool=true;
                        plot(i,j,'*',"Color",'g');
                    else
                        plot(i,j,'*',"Color",'r');
                    end
                    table(m,n)=bool;
                    n=n+1;
                end
                m=m+1;
            end
        end
        x2=x2+1;
    end
    x1=x1+1;
end
%table
plot(13,0,'*',"Color",'r')
hold off
```




```
s_dot=-6:0.1:6;
for i=1:length(table(1,:))
    a=find(table(:,i)==1,1,"first");
    s_dot1(i)=s_dot(a);
    b=find(table(:,i)==1,1,"last");
    s_dot2(i)=s_dot(b);
end

s=0:0.1:12;
figure
hold all
plot(s,s_dot1)
plot(s,s_dot2)
patch([s fliplr(s)], [s_dot1 fliplr(s_dot2)], 'g')
hold off
```




```
%[t,y] = ode45(@vdp_Car,tspan,[2.35;-4]);
toc
```

Elapsed time is 31.962863 seconds.

```
function dydt = vdp_Car(t,y)
dydt = [y(2); -9.81*sin(0.55*sin(1.2*y(1))-0.6*sin(1.1*y(1)))-0.7*y(2)];
end
```