15 Saturation and Antiwindup Strategies

Often times, the allowable values of the control input $u(t)$ are limited. For instance, there may be constant absolute upper and lower bounds, $\underline{u}$, $\overline{u}$ such that

$$\underline{u} \leq u(t) \leq \overline{u}$$

must hold for all t. As examples,

- the ailerons on an airplane wing may only deflect about 30° before they reach structural stops
- In a car (the cruise-control example) there is a maximum throttle opening

Other types of constraints are possible: in the airplane example, the ailerons (along with other movable surfaces) have maximum rates of movement. On a fighter plane, for instance, there may be a physical limit on the ailerons of 250°/sec.

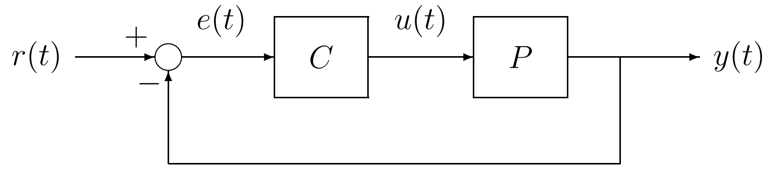
This can lead to performance, which is worse than the situation with no limits, for two reasons:

1. Ultimate response-times of the system are usually longer, since the actual inputs are generally smaller. This is a simple fact-of-life, and must be accepted.
2. The control strategy does not take this possibility into account, and has poor behavior at instances when the inputs are limited. This is **much worse** than the first reason, and must be remedied.

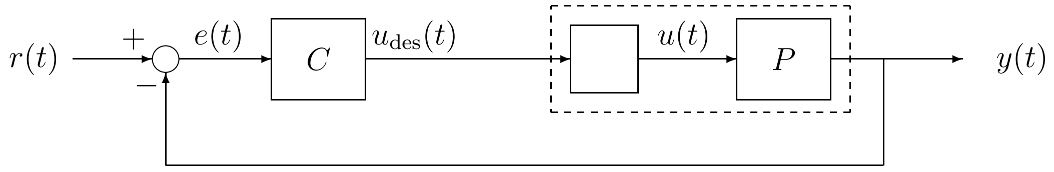
In all of these different cases, the control action (ie., $u(t)$) is limited, or constrained. Explicitly designing the control law to respect these limiting constraints is a very difficult mathematical problem. Often times, the constraints are dealt with using special tricks, and intuition, often on a problem-by-problem basis.

15.1 Saturation

We will consider a special case of the general problem. In the *ideal* situation, there are no limits on the control action, and the block diagram of the closed-loop system appears as usual, as below



The nonideal, or *real*, scenario is that there is a "saturating" element in between the controller and the plant, which limits the control action, regardless of the control system's output.



The behavior of the saturating element is simply

$$u(t) = \begin{array}{lll} U_{\max} & \text{if} & u_{\text{des}}(t) \geq U_{\max} \\ u_{\text{des}}(t) & \text{if} & U_{\min} < u_{\text{des}}(t) < U_{\max} \\ U_{\min} & \text{if} & u_{\text{des}}(t) \leq U_{\min} \end{array}$$

Note that one manner to think about this situation is that the controller "asks" for a desired control input, though the actuator can only "deliver" inputs that satisfy certain problem-dependent constraints.

Our goal is to learn how to implement the controller with extra logic so that when u is in bounds, the performance is the ideal performance, and for scenarios where u reaches its limits, the presence of the saturating element should not wreak too much havoc on the performance of the closed-loop system.

First, to see that this may cause problems, consider the cruise-control problem, with PI control. A new m-file `carpis.m` is given below. It has 5 extra parameters. They are, in order, $m, K_P, K_I, U_{\min}, U_{\max}$. This file simply limits the actual throttle signal to the car. It is written to clearly separate the car equations from the controller equations.

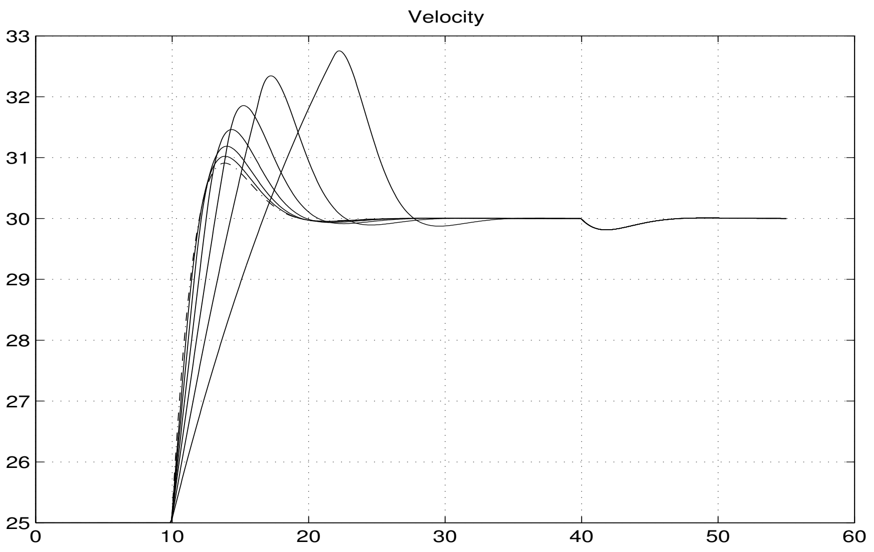
file: carpis.m

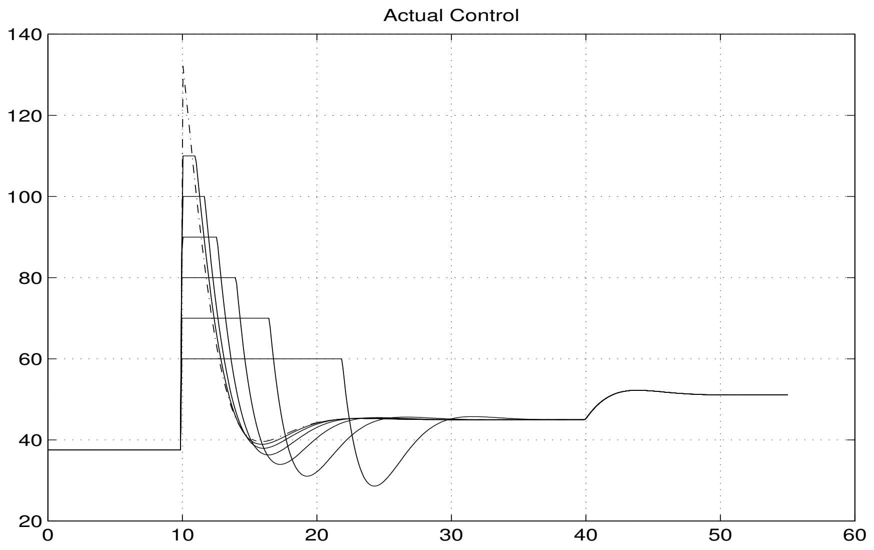
```
function [out1,out2] = carpis(t,x,d,flag,p1,p2,p3,p4,p5)

    % Interpretations x = [v;z], d = [v_des;w], e = [v;u];

    E = 40; alpha = 60; G = -98;
    m = p1; kp = p2; ki = p3; umin = p4; umax = p5;

    x10 = 25; x20 = (alpha*x10)/(E*ki);

    if flag==0
       out1 = [2;0;2;2;0;1];
       out2 = [x10;x20];
    elseif flag==1
       udes = kp*(d(1)-x(1)) + ki*x(2);
       if udes>umax
          u = umax;
       elseif udes<umin
          u = umin;
       else
          u = udes;
       end
       vdot = -alpha*x(1)/m + E*u/m + G*d(2)/m;
       zdot = d(1) - x(1);
       out1 = [vdot;zdot];
    elseif flag==3
       udes = kp*(d(1)-x(1)) + ki*x(2);
       if udes>umax
          u = umax;
       elseif udes<umin
          u = umin;
       else
          u = udes;
       end
       out1 = [x(1);u];
    end
```

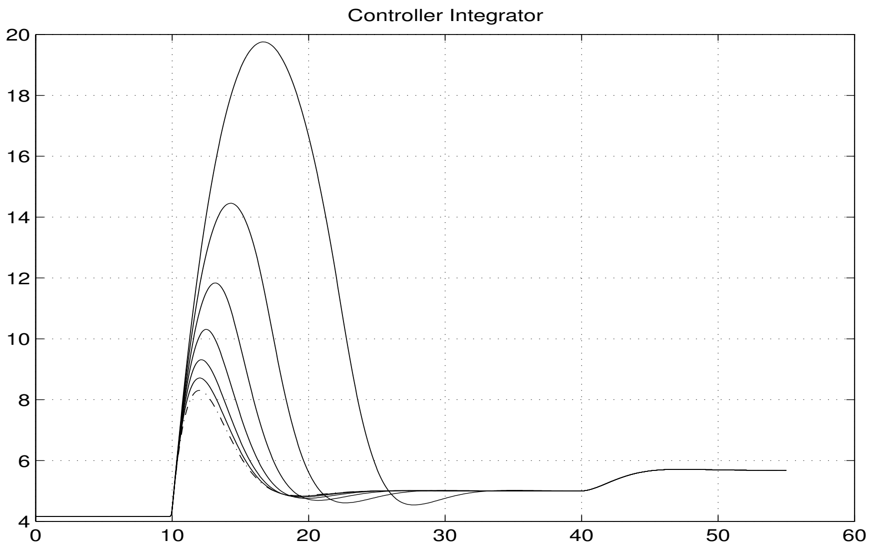
Suppose that we choose as a nominal design $\omega_n := 0.6, \xi := 0.7$, and choose gains K_P and K_I in terms of the design equations in section 8. A simulation of the system without saturation can be done using `inf` and `-inf` gives a decent response, as we have seen earlier, as shown below. Suppose that a saturating element is actually in place. Take $U_{\min} := -U_{\max}$ (so the saturating element is symmetric, which may or may not be realistic in a car example...), and simulate the system for several different values of $U_{\max}$.

The code below will do everything just described. It is best to put this in a file, say `carsim.m`.

```
≫ E = 40; alpha = 60; m = 1000; wn = 0.6; xi = 0.7;
≫ TF = 80;
≫ timespan = [T0 TF];
≫ simopts = simset('solver','ode45');
≫ kp = (2*xi*wn*m - alpha)/E;
≫ ki = m*wn*wn/E;
≫ umax = inf; umin = -inf;
≫ dt = [0 25 0; 9.9 25 0;10 30 0;39.9 30 0;40 30 2.5;80 30 2.5];
≫ intopt = [1e-3;0.001;.1;1;0;2];
≫ [tdata,xdata,edata] = sim('carpis',timespan,simopts,dt,m,kp,ki,umin,umax);
≫ figure(1)
≫ plot(tdata,edata(:,1),'-.'); title('Velocity')
≫ hold on
≫ figure(2)
≫ plot(tdata,edata(:,2),'-.'); title('Actual Control')
≫ hold on
≫ figure(3)
≫ plot(tdata,xdata(:,2),'-.'); title('Controller Integrator')
≫ hold on
≫ satvec = [110 100 90 80 70 60];
≫ for i=1:length(satvec)
≫    umin = -satvec(i);
≫    umax = satvec(i);
≫    [tdata,xdata,edata] = sim('carpis',timespan,simopts,dt,m,kp,ki,umin,umax);
≫    figure(1); plot(tdata,edata(:,1)); hold on
≫    figure(2); plot(tdata,edata(:,2)); hold on
≫    figure(3); plot(tdata,xdata(:,2)); hold on
≫ end
≫ figure(1); grid; hold off
≫ figure(2); grid; hold off
≫ figure(3); grid; hold off
```


Velocity
33
32
31
30
29
28
27
26
25
0
10
20
30
40
50
60


Actual Control
140
120
100
80
60
40
20
0
10
20
30
40
50
60


Controller Integrator
20
18
16
14
12
10
8
6
4
0
10
20
30
40
50
60

Note that as $U_{\max}$ decreases, the closed-loop performance degrades quite ungracefully. *Even though the maximum acceleration is limited, and the slope is decreased, the overshoot actually gets worse!.*

The problem is that once the controller output $u(t)$ exceeds $U_{\max}$, it is of no use for the variable z to keep increasing, since increases its value, which causes increases in $u(t) = K_P\left[v_{\rm des}(t) - v(t)\right] + K_I z(t)$ occur with no benefit – the maximum force, that associated with $U_{\max}$ is already being applied. In fact, not only is there no benefit to increasing the variable z in this scenario, there is actually a significant drawback. Once z is increased beyond a useful value, decreasing it has no immediate effect, since the value of $K_P\left[v_{\rm des}(t) - v(t)\right] + K_I z(t)$ is still well above the threshold $U_{\max}$. So, the control circuitry wastes valuable time getting z back down to a useful level, at which changes in it actually lead to changes in the true applied force.

In this case, the accepted terminology is that "the integral controller wound up, causing unnecessary overshoot and oscillations."

In order to avoid *integral wind-up*, we need some additional circuitry or software in the control logic which essentially "turns off the controller integrator" in certain situations.

15.2 Anti-Windup PI control action

For a general PI controller, let $r(t)$ denote the reference signal, which is the *desired value of the plant output*, and let $y(t)$ denote the *actual plant output*. The PI controller equations are

$$\begin{aligned} \dot{z}(t) &= r(t) - y(t) \\ u_{\rm des}(t) &= K_P\left[r(t) - y(t)\right] + K_I z(t) \end{aligned} \tag{71}$$

The saturating element maps the desired control input into the actual control input,

$$u(t) = \begin{array}{lll} U_{\max} & \text{if} & u_{\rm des}(t) \geq U_{\max} \\ u_{\rm des}(t) & \text{if} & U_{\min} < u_{\rm des}(t) < U_{\max} \\ U_{\min} & \text{if} & u_{\rm des}(t) \leq U_{\min} \end{array}$$

Suppose (for clarity) that $K_I > 0$. You need to rearrange some inequalities if $K_I < 0$. One viable strategy is

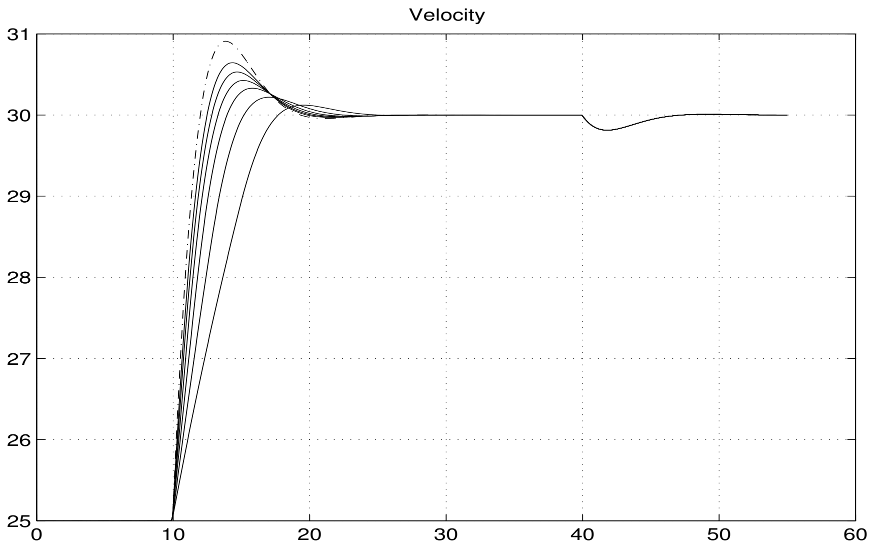
- If u is in-bounds, run as usual
- If u is at $U_{\max}$, and $r - y > 0$, running as usual would increase z, increasing $u_{\rm des}$, at no benefit, only screwing up the ability to come off (since decreasing z would initially then have no benefit). So, stop integrator.
- If u is at $U_{\max}$, and $r - y < 0$, continue integrating.

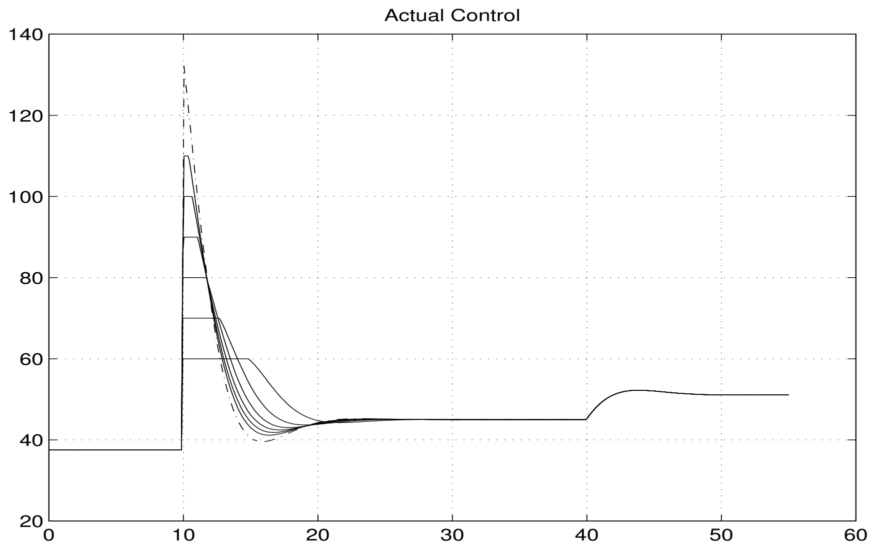
- If u is at $U_{\min}$, and $r - y > 0$, run as usual
- If u is at $U_{\min}$, and $r - y < 0$, stop integrator.

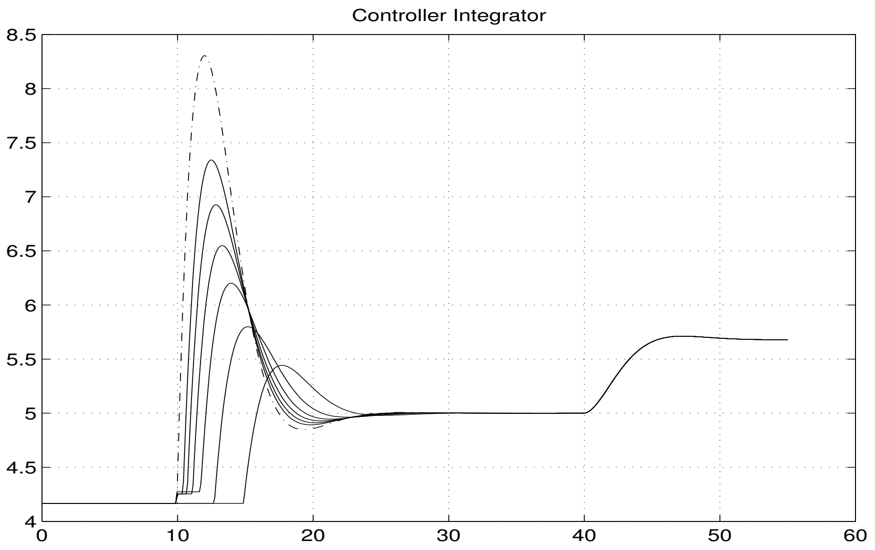
Mathematically, this is

$$\begin{array}{lcl}
u_{\text{des}}(t) & := & K_P\,[r(t) - y(t)] \; + \; K_I z(t) \\
\text{if } u_{\text{des}}(t) \geq U_{\max} \text{ and } r(t) \geq y(t) & \Rightarrow & \dot{z}(t) = 0 \\
\text{elseif } u_{\text{des}}(t) \geq U_{\max} \text{ and } r(t) \leq y(t) & \Rightarrow & \dot{z}(t) = r(t) - y(t) \\
\text{if } u_{\text{des}}(t) \leq U_{\min} \text{ and } r(t) \leq y(t) & \Rightarrow & \dot{z}(t) = 0 \\
\text{elseif } u_{\text{des}}(t) \leq U_{\min} \text{ and } r(t) \geq y(t) & \Rightarrow & \dot{z}(t) = r(t) - y(t)
\end{array} \tag{72}$$

Doing so yields much better performance. The plots are shown on the next page.


Velocity
31
30
29
28
27
26
25
0
10
20
30
40
50
60

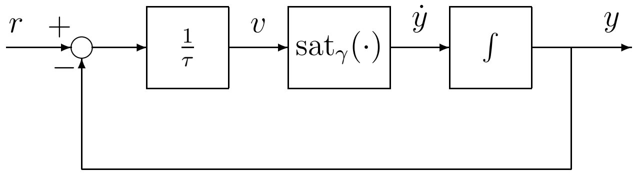

Actual Control
140
120
100
80
60
40
20
0
10
20
30
40
50
60


Controller Integrator
8.5
8
7.5
7
6.5
6
5.5
5
4.5
4
0
10
20
30
40
50
60

15.3 Problems

1. In simple mathematical models for actuators (like the motor/aileron control surface on an airplane), often we would like to capture the fact that the output of the actuator (the angular position of the flap) cannot move faster than a certain limit, but otherwise responds like a first-order system.

   (a) A block diagram of a model for this behavior is shown below. Assume that $\tau > 0$. The saturation block has the static (nondynamic) relationship shown in its block - the output is the input for small inputs, but beyond a certain range, the output is limited.



$$\text{sat}_\gamma(v) = \begin{matrix} \gamma & \text{if} & v > \gamma \\ v & \text{if} & -\gamma \leq v \leq \gamma \\ -\gamma & \text{if} & v < -\gamma \end{matrix}$$

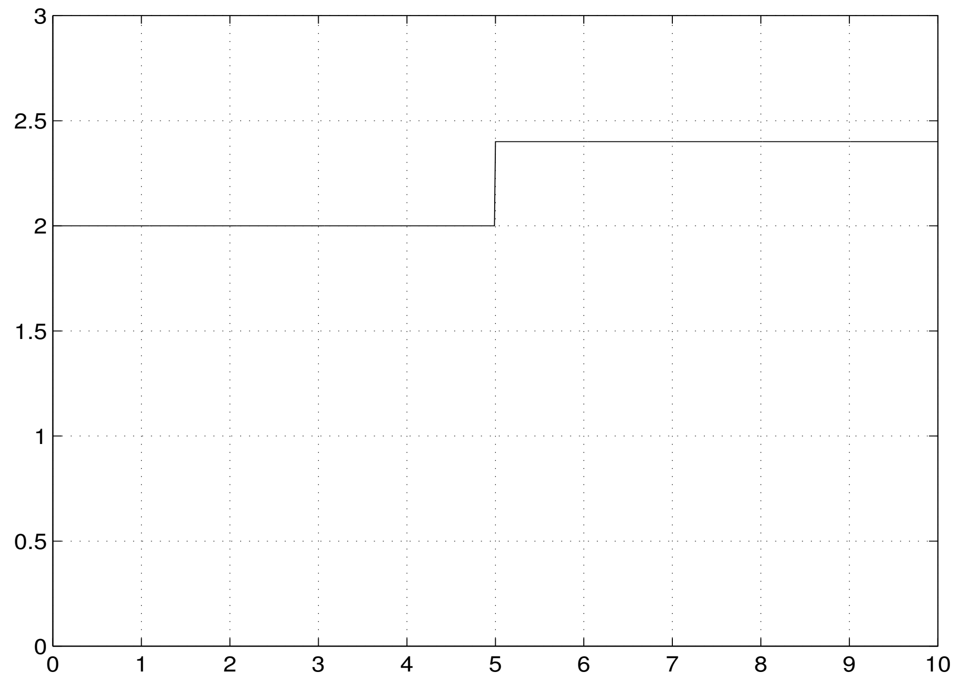
   There are actually three (3) different regimes in which this model behaves differently. For each of the cases below, explain what $\dot{y}(t)$ is in each regime:

   i. What is $\dot{y}(t)$ when $r(t) - \tau\gamma < y(t) < r(t) + \tau\gamma$
   ii. What is $\dot{y}(t)$ when $y(t) < r(t) - \tau\gamma$
   iii. What is $\dot{y}(t)$ when $r(t) + \tau\gamma < y(t)$
   iv. Define $V(t) := [r(t) - y(t)]^2$. Suppose $r(t)$ is constant, $\bar{r}$. Show that regardless of what regime we are in, $\dot{V}(t) \leq 0$, and in fact, $\dot{V}(t) = 0 \Leftrightarrow y(t) = \bar{r}$, and $y(t) \neq \bar{r} \Leftrightarrow \dot{V}(t) < 0$.

   (b) Starting from zero (0) initial conditions, determine the steady-state value of y due to a step input $r(t) = \bar{r}$, where $\bar{r}$ is some fixed real number. Think carefully how y will transition from its initial condition of $y(0) = 0$ to its final value.

   (c) An input signal $r(t)$ is shown below. On the same graph, draw the resulting output $y(t)$, starting from initial condition $y(0) = 0$. Do this <u>without</u> the computer, using your results in part (1a), for the following cases

   i. $\tau = 0.5, \gamma = 0.4$.
   ii. $\tau = 0.5, \gamma = 0.7$.
   iii. $\tau = 0.5, \gamma = 1.0$.
   iv. $\tau = 0.5, \gamma = 1.4$.
   v. $\tau = 0.5, \gamma = 5.0$.


3
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6
7
8
9
10