

# **EMBEDDED SYSTEMS**

## **CHAPTER -7**

### **CONTROL SYSTEM**

# 7. Control System [3 Hrs.]

7.1 Open-loop and Close-Loop  
control System overview

7.2 Control System and PID  
Controllers

7.3 Software coding of a PID  
Controller

7.4 PID Tuning

# Control System

- Control physical system's output
  - By setting physical system's input

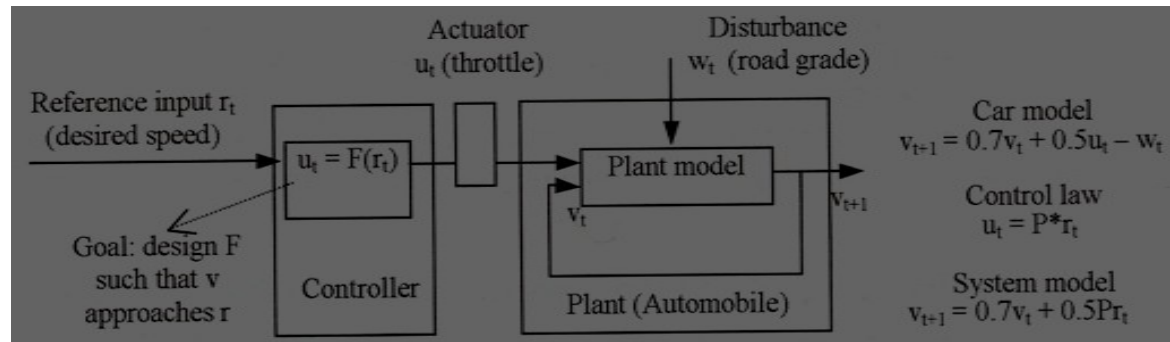
- Tracking

- E.g.

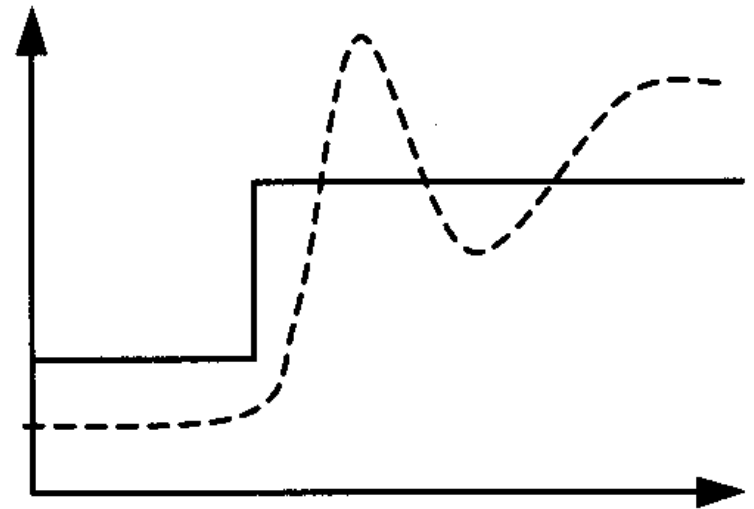
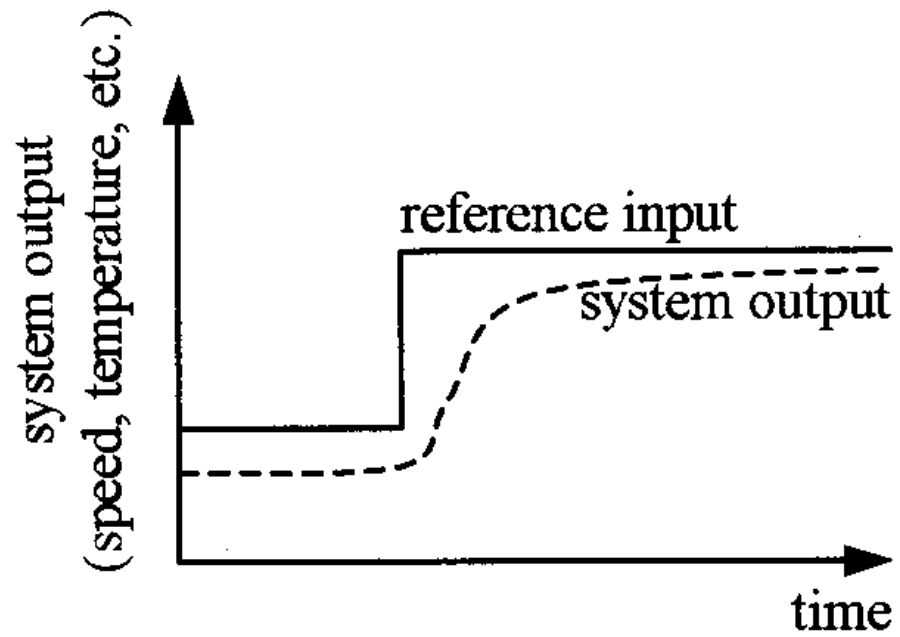
- Cruise control
- Thermostat control
- Disk drive control
- Aircraft altitude control

- Difficulty due to

- Disturbance: wind, road, tire, brake; opening/closing door...
- Human interface: feel good, feel right...



# Tracking



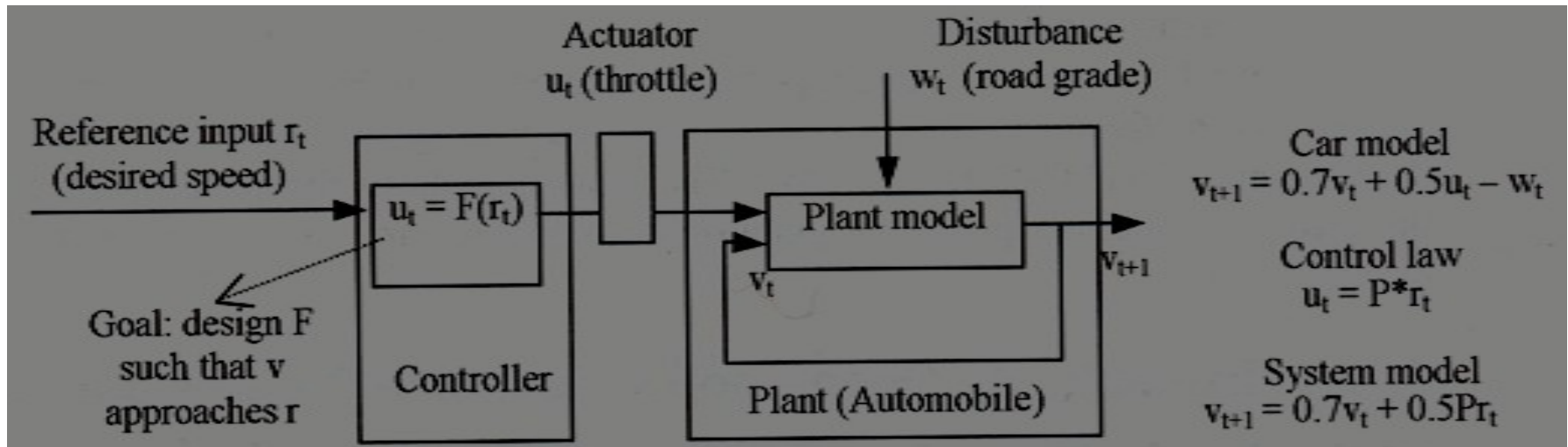
# Open-Loop Control Systems

- Plant
  - Physical system to be controlled
    - Car, plane, disk, heater,...
- Actuator
  - Device to control the plant
    - Throttle, wing flap, disk motor,...
- Controller
  - Designed product to control the plant

$V_t$  – car's current speed

$U_t$  – throttle position

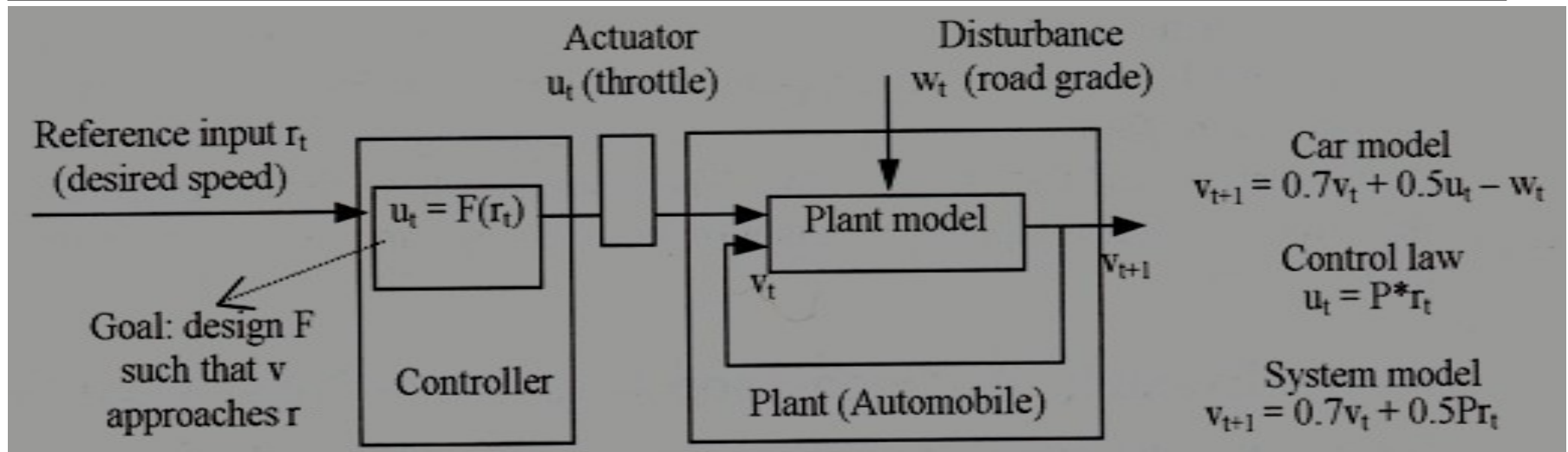
$V_{t+1}$  – car's speed one sec. later



# Open-Loop Control Systems

- **Output**
  - The aspect of the physical system we are interested in
    - Speed, disk location, temperature
- **Reference**
  - The value we want to see at output
    - Desired speed, desired location, desired temperature
- **Disturbance**
  - Uncontrollable input to the plant imposed by environment
    - Wind, bumping the disk drive, door opening

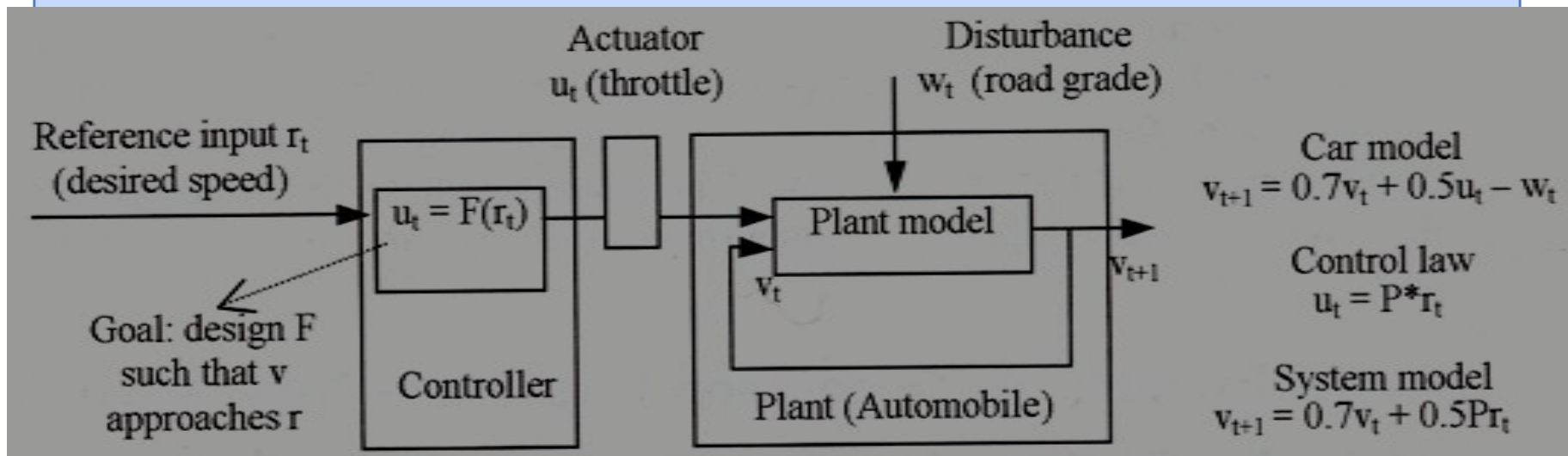
$V_t$  – car's current speed  
 $U_t$  – throttle position  
 $V_{t+1}$  – car's speed one sec. later



# Other Characteristics of open loop

- Feed-forward control
- Delay in actual change of the output
- Controller doesn't know how well thing goes
- Simple
- Best use for predictable systems

$V_t$  – car's current speed  
 $U_t$  – throttle position  
 $V_{t+1}$  – car's speed one sec. later



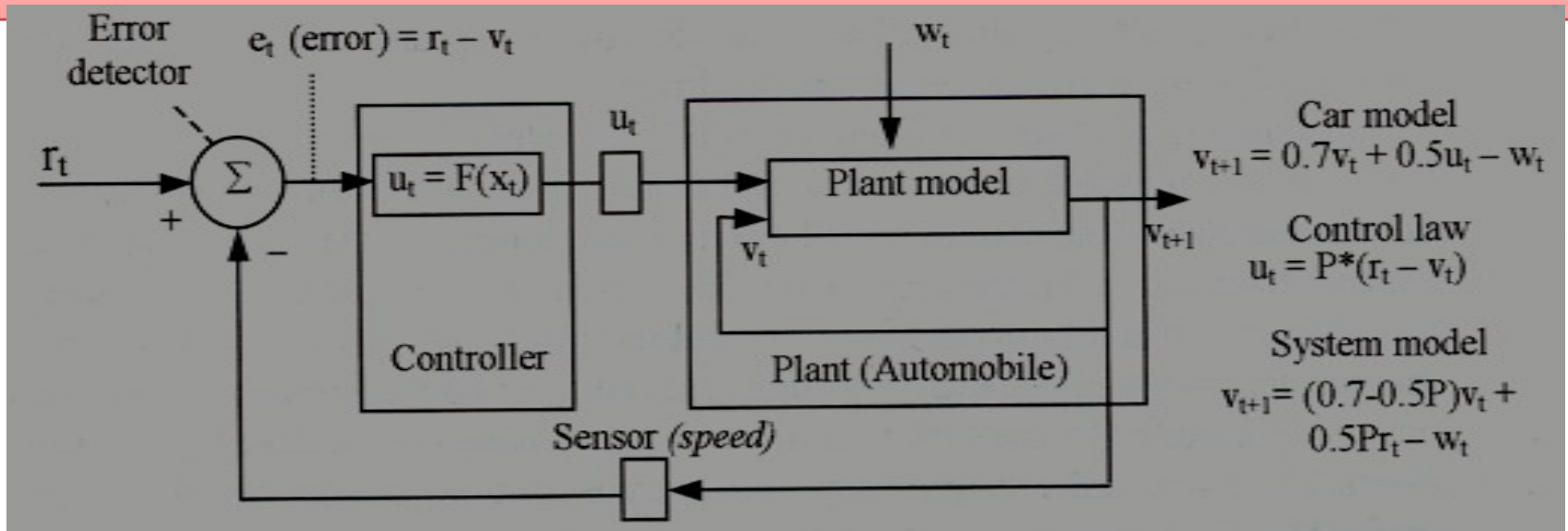
# Close Loop Control Systems

- **Sensor**
  - Measure the plant output
- **Error detector**
  - Detect Error
- **Feedback control systems**
- **Minimize tracking error**

$V_t$  – car's current speed

$U_t$  – throttle position

$V_{t+1}$  – car's speed one sec. later





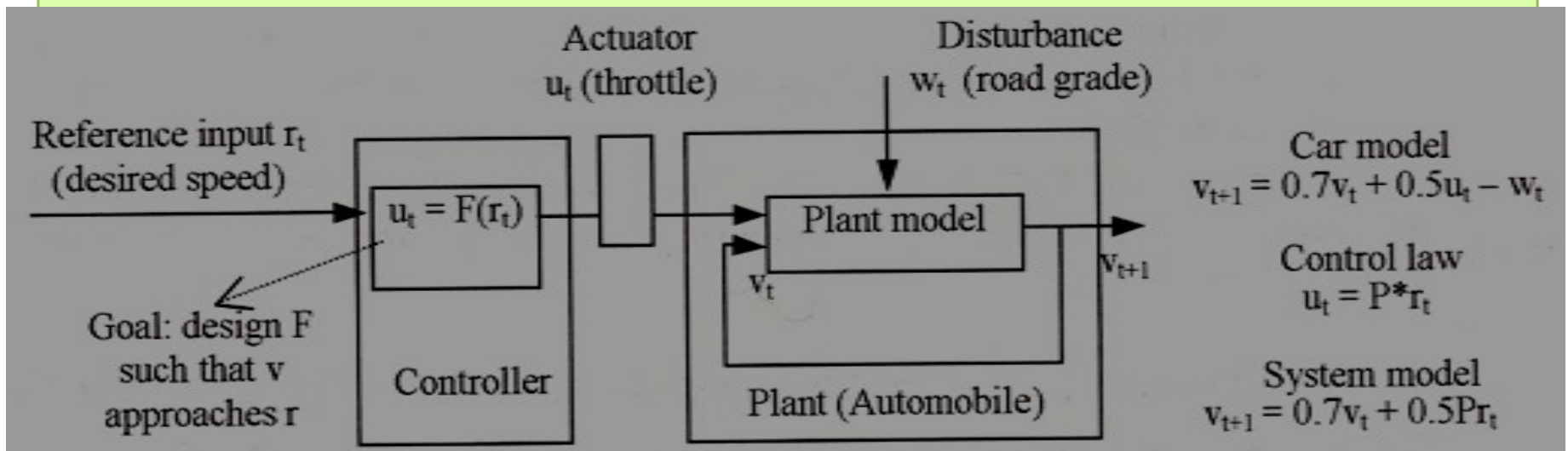
# Designing Open Loop Control System

- Develop a model of the plant
- Develop a controller
- Analyze the controller
- Consider Disturbance
- Determine Performance
- Example: Open Loop Cruise Control System

$V_t$  – car's current speed

$U_t$  – throttle position

$V_{t+1}$  – car's speed one sec. later



# Model of the Plant

- **May not be necessary**
  - Can be done through experimenting and tuning
- **But,**
  - Can make it easier to design
  - May be useful for deriving the controller
- **Example: throttle that goes from 0 to 45 degree**
  - On flat surface at 50 mph, open the throttle to 40 degree
  - Wait 1 “time unit”
  - Measure the speed, let’s say 55 mph
  - Then the following equation satisfy the above scenario
    - $v_{t+1} = 0.7 * v_t + 0.5 * u_t$
    - $55 = 0.7 * 50 + 0.5 * 40$
  - IF the equation holds for all other scenario
    - Then we have a model of the plant

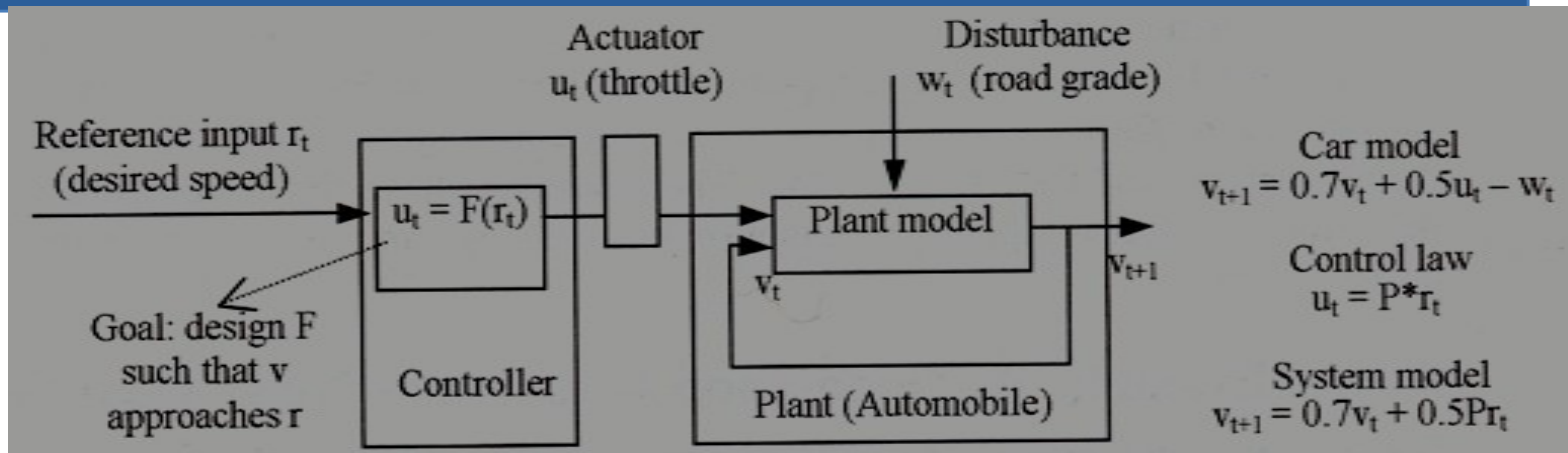
# Designing the Controller

- Assuming we want to use a simple linear function
  - $u_t = F(r_t) = P * r_t$
  - $r_t$  is the desired speed,  $P$  is a constant that the designer must specify.
- Linear proportional controller
- $v_{t+1} = 0.7 * v_t + 0.5 * u_t = 0.7 * v_t + 0.5P * r_t$
- Let  $v_{t+1} = v_t$  at steady state =  $v_{ss}$
- $v_{ss} = 0.7 * v_{ss} + 0.5P * r_t$
- At steady state, we want  $v_{ss} = r_t$
- $P = 0.6$ 
  - I.e.  $u_t = 0.6 * r_t$

$V_t$  – car's current speed

$U_t$  – throttle position

$V_{t+1}$  – car's speed one sec. later



# Analyzing the Controller

- Let  $v_0=20\text{mph}$ ,  $r_0=50\text{mph}$
- $v_{t+1}=0.7*v_t+0.5(0.6)*r_t=0.7*v_t+0.3*50=0.7*v_t+15$
- Throttle position is  $0.6*50=30$  degree

Time (t)	$v_t$
0	20.00
1	29.00
2	35.30
3	39.71
4	42.80
5	44.96
6	46.47
7	47.53
8	48.27
9	48.79
10	49.15
11	49.41
12	49.58

# Considering the Disturbance

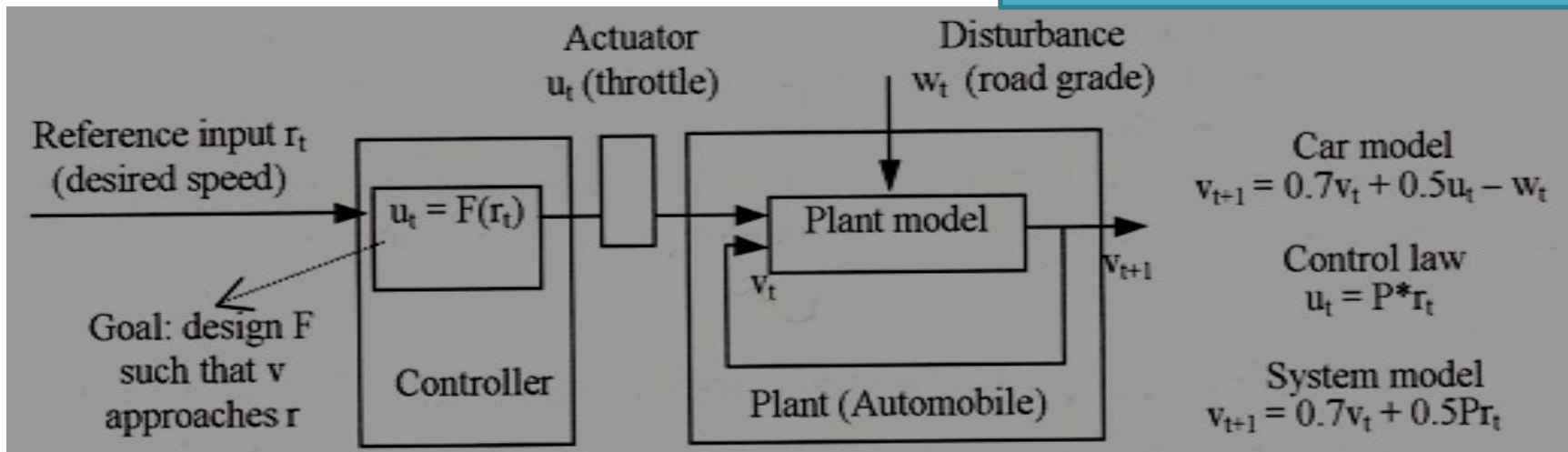
- Assume road grade can affect the speed
  - From  $-5\text{mph}$  to  $+5\text{ mph}$
  - $v_{t+1}=0.7*v_t+10$
  - $v_{t+1}=0.7*v_t+20$

Time (t)	$v_t$	$v_t$ for $w = +5$	$v_t$ for $w = -5$
0	20.00	20.00	20.00
1	29.00	24.00	34.00
2	35.30	26.80	43.80
3	39.71	28.76	50.66
4	42.80	30.13	55.46
5	44.96	31.09	58.82
6	46.47	31.76	61.18
7	47.53	32.24	62.82
8	48.27	32.56	63.98
9	48.79	32.80	64.78
10	49.15	32.96	65.35
11	49.41	33.07	65.74
12	49.58	33.15	66.02

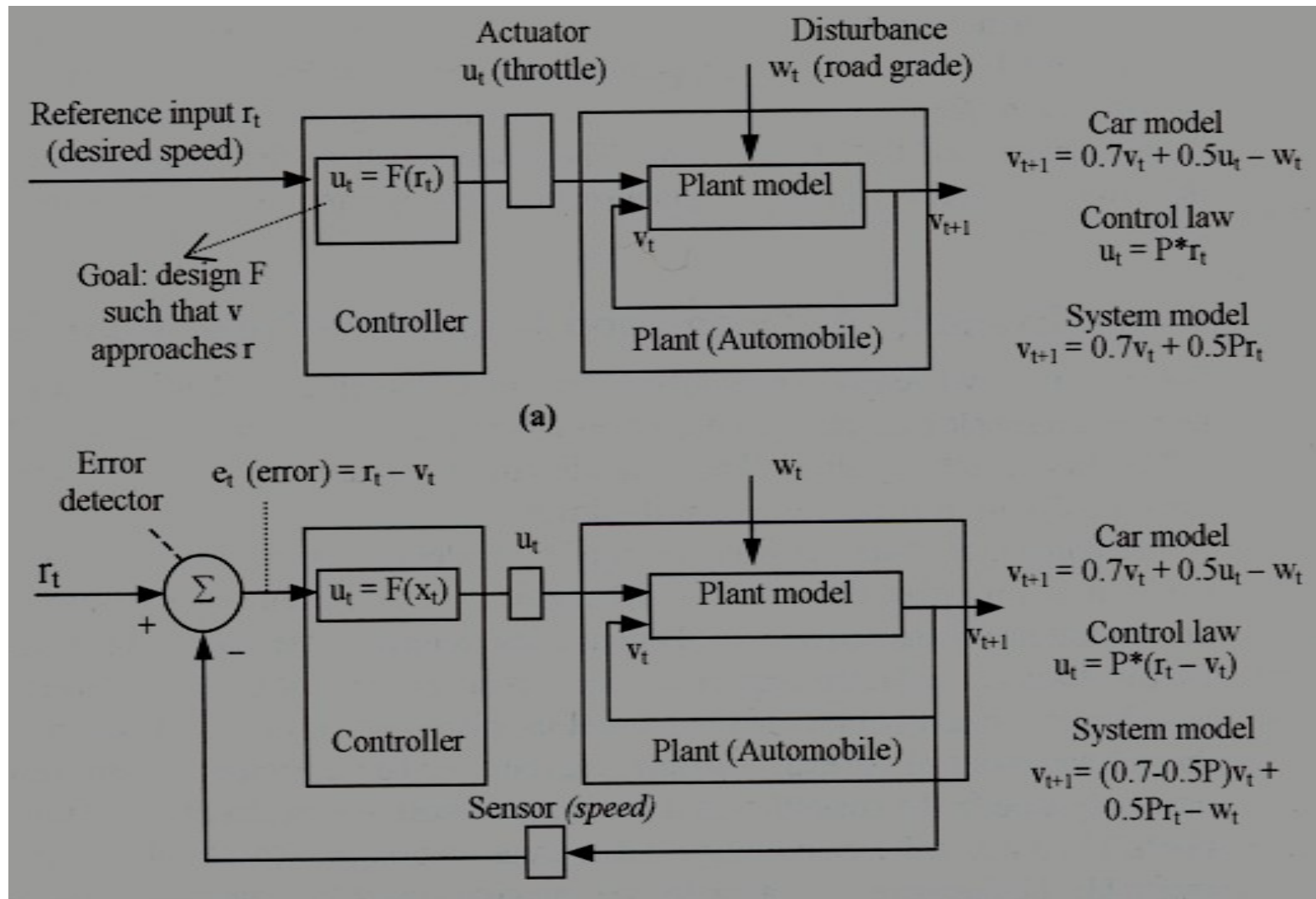
# Determining Performance

- $V_{t+1} = 0.7 * v_t + 0.5P * r_0 - w_0$
- $v_1 = 0.7 * v_0 + 0.5P * r_0 - w_0$
- $v_2 = 0.7 * (0.7 * v_0 + 0.5P * r_0 - w_0) + 0.5P * r_0 - w_0 = 0.7 * 0.7 * v_0 + (0.7 + 1.0) * 0.5P * r_0 - (0.7 + 1.0)w_0$
- $v^t = 0.7^t * v_0 + (0.7^{t-1} + 0.7^{t-2} + \dots + 0.7 + 1.0)(0.5P * r_0 - w_0)$
- Coefficient of  $v_t$  determines rate of decay of  $v_0$ 
  - $>1$  or  $<-1$ ,  $v_t$  will grow without bound
  - $<0$ ,  $v_t$  will oscillate

$V_t$  – car's current speed  
 $U_t$  – throttle position  
 $V_{t+1}$  – car's speed one sec. later



# Designing Close Loop Control System

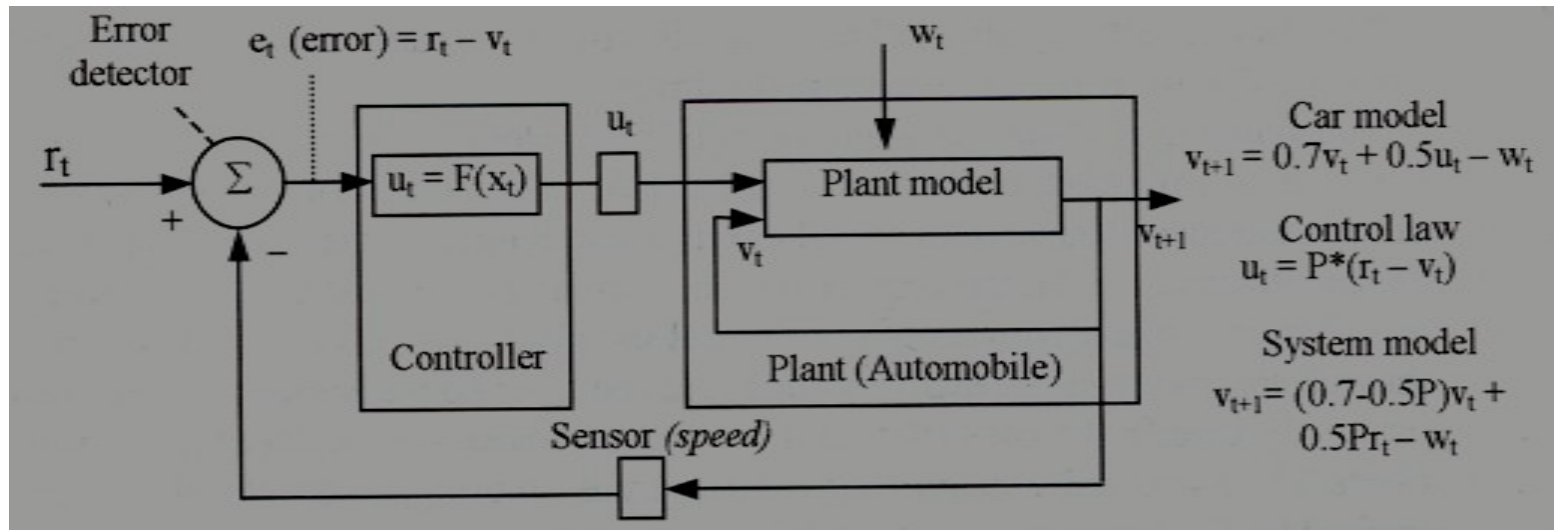


# Stability

- $u_t = P * (r_t - v_t)$
- $v_{t+1} = 0.7v_t + 0.5u_t - w_t = 0.7v_t + 0.5P*(r_t - v_t) - w_t$   
 $= (0.7 - 0.5P)*v_t + 0.5P*r_t - w_t$
- $v^t = (0.7 - 0.5P)^t * v_0 + ((0.7 - 0.5P)^{t-1} + (0.7 - 0.5P)^{t-2} + \dots + 0.7 - 0.5P + 1.0)(0.5P*r_0 - w_0)$

- Stability constraint (I.e. convergence) requires  
 $|0.7 - 0.5P| < 1$   
 $-1 < 0.7 - 0.5P < 1$   
 $-0.6 < P < 3.4$

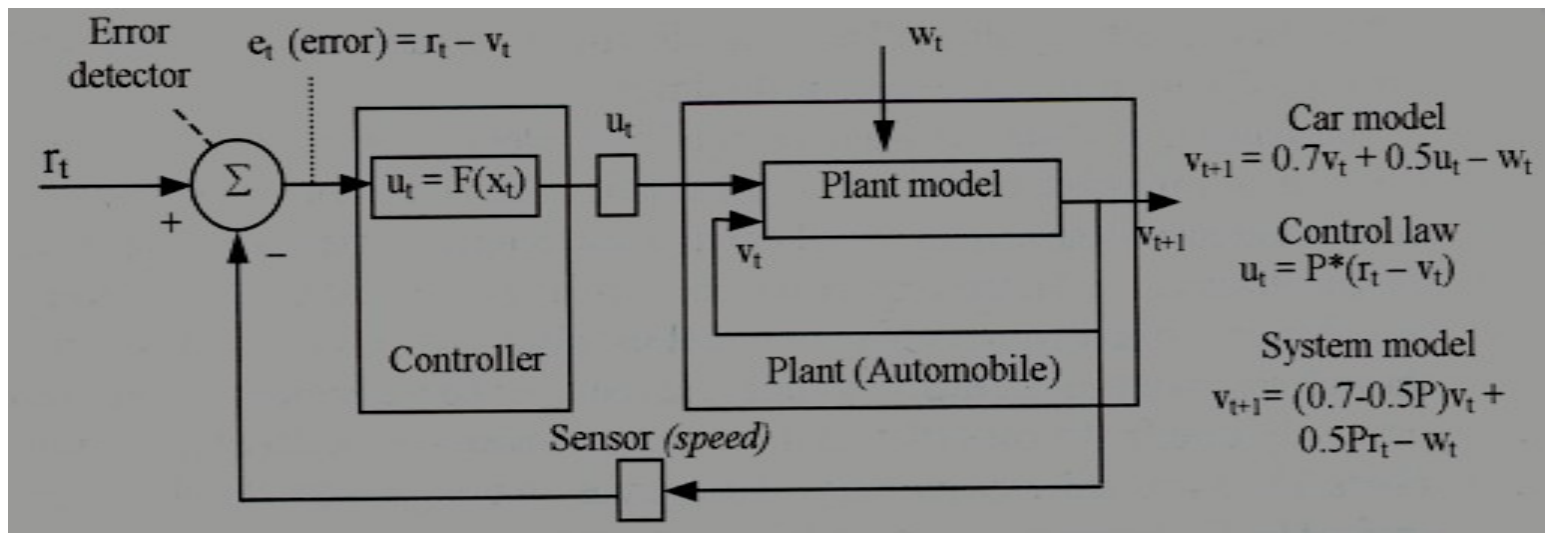
$V_t$  – car's current speed  
 $U_t$  – throttle position  
 $V_{t+1}$  – car's speed one sec. later





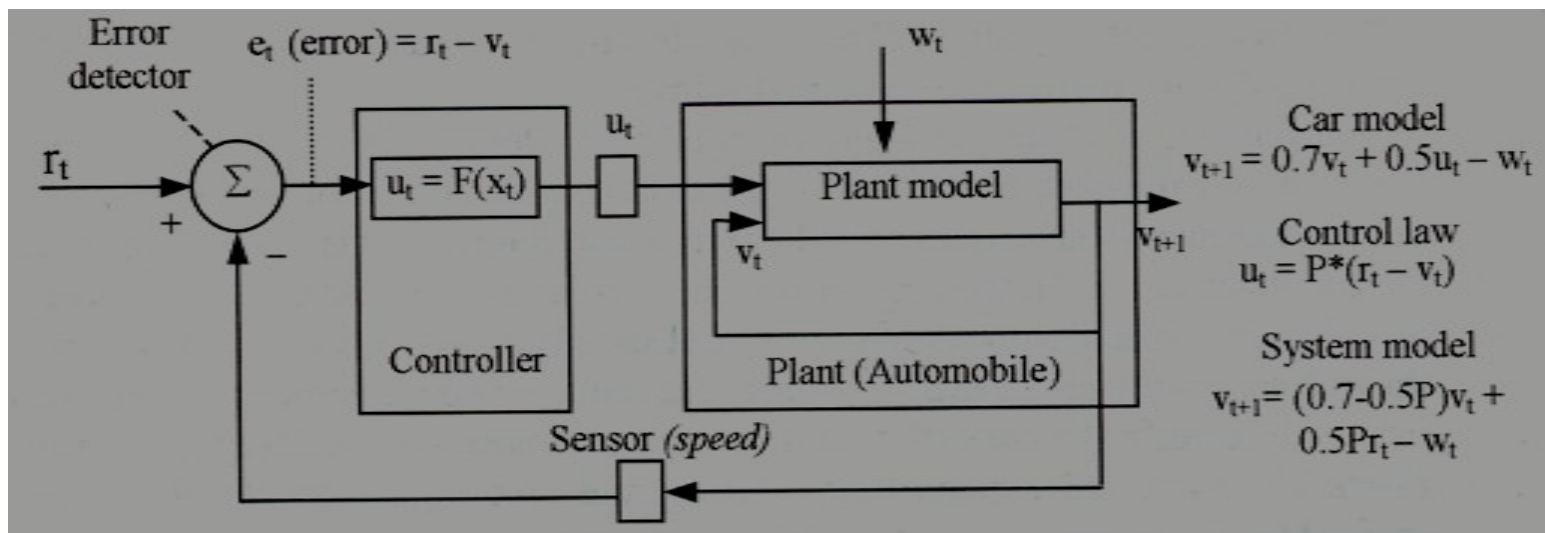
# Reducing effect of $v_0$

- $u_t = P * (r_t - v_t)$
- $v_{t+1} = 0.7v_t + 0.5u_t - w_t = 0.7v_t + 0.5P*(r_t - v_t) - w_t$   
 $= (0.7 - 0.5P)*v_t + 0.5P*r_t - w_t$
- $v^t = (0.7 - 0.5P)^t * v_0 + ((0.7 - 0.5P)^{t-1} + (0.7 - 0.5P)^{t-2} + \dots + 0.7 - 0.5P + 1.0)(0.5P*r_0 - w_0)$
- To reduce the effect of initial condition
  - $0.7 - 0.5P$  as small as possible
  - $P = 1.4$



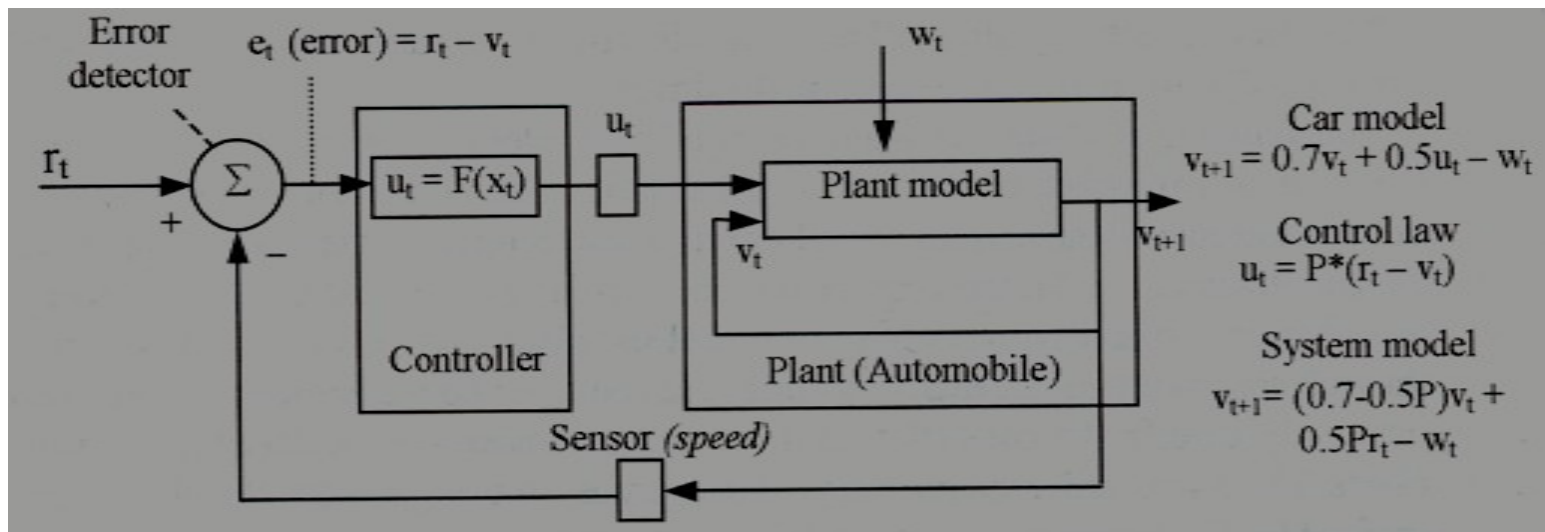
# Avoid Oscillation

- $u_t = P * (r_t - v_t)$
- $v_{t+1} = 0.7v_t + 0.5u_t - w_t = 0.7v_t + 0.5P*(r_t - v_t) - w_t$   
 $= (0.7 - 0.5P)*v_t + 0.5P*r_t - w_t$
- $v^t = (0.7 - 0.5P)^t * v_0 + ((0.7 - 0.5P)^{t-1} + (0.7 - 0.5P)^{t-2} + \dots + 0.7 - 0.5P + 1.0)(0.5P*r_0 - w_0)$
- To avoid oscillation
  - $0.7 - 0.5P \geq 0$
  - $P \leq 1.4$



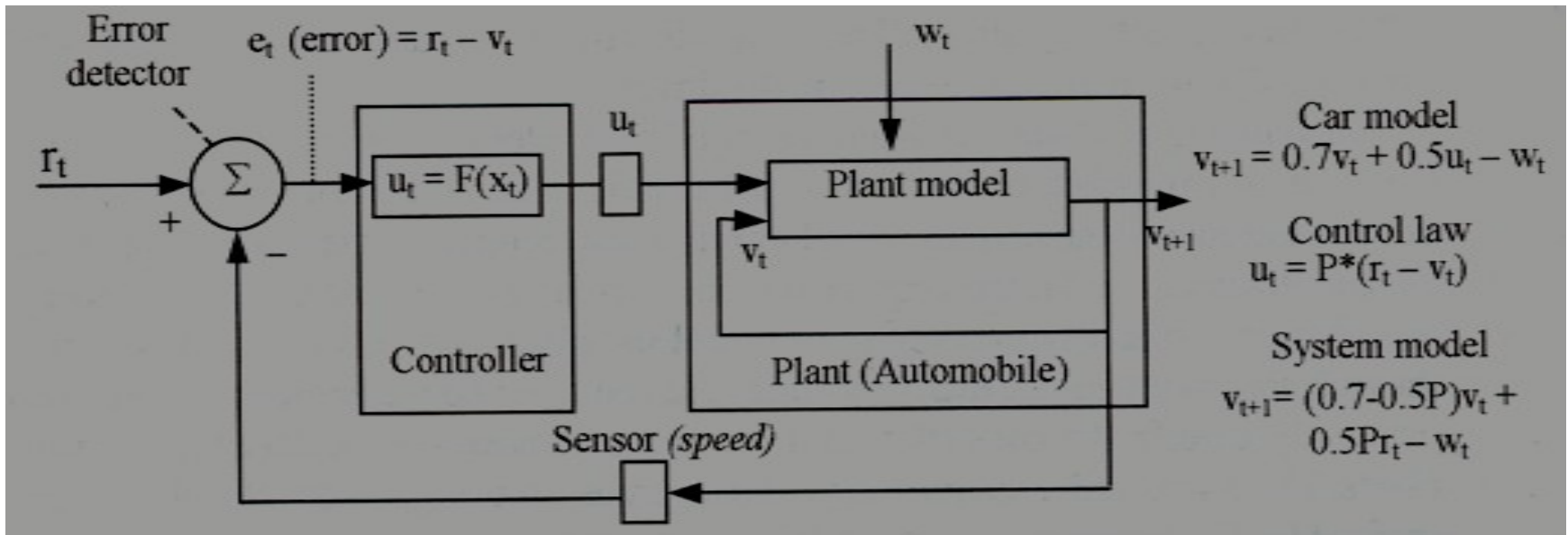
# Perfect Tracking

- $u_t = P * (r_t - v_t)$
- $v_{t+1} = 0.7v_t + 0.5u_t - w_t = 0.7v_t + 0.5P*(r_t - v_t) - w_t$   
 $= (0.7 - 0.5P)*v_t + 0.5P*r_t - w_t$
- $v_{ss} = (0.7 - 0.5P)*v_{ss} + 0.5P*r_0 - w_0$   
 $(1 - 0.7 + 0.5P)v_{ss} = 0.5P*r_0 - w_0$   
 $v_{ss} = (0.5P / (0.3 + 0.5P)) * r_0 - (1.0 / (0.3 + 0.5P)) * w_0$
- To make  $v_{ss}$  as close to  $r_0$  as possible
  - $P$  should be as large as possible



# Close-Loop Design

- $u_t = P * (r_t - v_t)$
- Finally, setting  $P=3.3$ 
  - Stable, track well, some oscillation
  - $u_t = 3.3 * (r_t - v_t)$



# Analyze the controller

- $v_0=20$  mph,  $r_0=50$  mph,  $w=0$
- $v_{t+1} = 0.7v_t + 0.5P*(r_t - v_t) - w$   
 $= 0.7v_t + 0.5*3.3*(50 - v_t)$
- $u_t = P * (r_t - v_t)$   
 $= 3.3 * (50 - v_t)$
- But  $u_t$  range from 0-45
- Controller saturates

Time	$v_t$	$u_t$
0	20.00	99.00
1	63.50	-44.55
2	22.18	91.82
3	61.43	-37.73
4	24.14	85.34
5	59.57	-31.58
6	25.91	79.50
7	57.89	-26.02
8	27.51	74.22
9	56.37	-21.01
10	28.95	69.46
...		
45	44.53	18.06
46	40.20	32.34
47	44.31	18.78
48	40.41	31.66
49	44.11	19.42
50	40.59	31.05
...		
ss	42.31	25.38

# Analyze the controller

- $v_0=20$  mph,  $r_0=50$  mph,  $w=0$
- $v_{t+1} = 0.7v_t + 0.5 * u_t$
- $u_t = 3.3 * (50 - v_t)$ 
  - Saturate at 0, 45
- Oscillation!
  - “feel bad”

Time	$v_t$	$u_t$	$v_t$	$u_t$
0	20.00	99.00	20.00	45.00
1	63.50	-44.55	36.50	44.55
2	22.18	91.82	47.83	7.18
3	61.43	-37.73	37.07	42.68
4	24.14	85.34	47.29	8.95
5	59.57	-31.58	37.58	40.99
6	25.91	79.50	46.80	10.55
7	57.89	-26.02	38.04	39.47
8	27.51	74.22	46.36	12.00
9	56.37	-21.01	38.45	38.10
10	28.95	69.46	45.97	13.31
...				
45	44.53	18.06	41.70	27.39
46	40.20	32.34	42.89	23.48
47	44.31	18.78	41.76	27.20
48	40.41	31.66	42.83	23.66
49	44.11	19.42	41.81	27.02
50	40.59	31.05	42.78	23.83
...				
ss	42.31	25.38	42.31	25.38

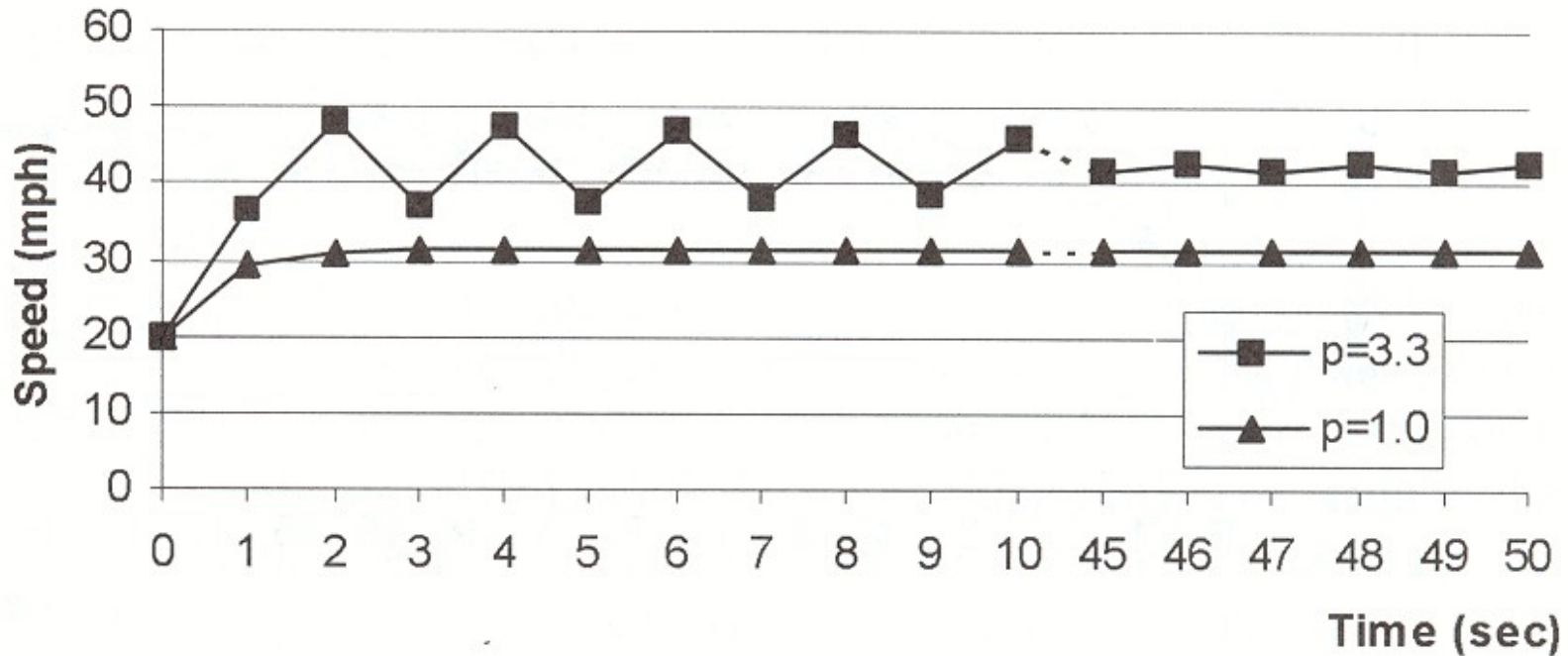
# Analyze the controller

- Set  $P=1.0$  to void oscillation
  - Terrible SS performance

Time	$v_t$	$u_t$	$v_t$	$u_t$	$v_t$	$u_t$
0	20.00	99.00	20.00	45.00	20.00	30.00
1	63.50	-44.55	36.50	44.55	29.00	21.00
2	22.18	91.82	47.83	7.18	30.80	19.20
3	61.43	-37.73	37.07	42.68	31.16	18.84
4	24.14	85.34	47.29	8.95	31.23	18.77
5	59.57	-31.58	37.58	40.99	31.25	18.75
6	25.91	79.50	46.80	10.55	31.25	18.75
7	57.89	-26.02	38.04	39.47	31.25	18.75
8	27.51	74.22	46.36	12.00	31.25	18.75
9	56.37	-21.01	38.45	38.10	31.25	18.75
10	28.95	69.46	45.97	13.31	31.25	18.75
...						
45	44.53	18.06	41.70	27.39	31.25	18.75
46	40.20	32.34	42.89	23.48	31.25	18.75
47	44.31	18.78	41.76	27.20	31.25	18.75
48	40.41	31.66	42.83	23.66	31.25	18.75
49	44.11	19.42	41.81	27.02	31.25	18.75
50	40.59	31.05	42.78	23.83	31.25	18.75
...						
ss	42.31	25.38	42.31	25.38	31.25	18.75



# Analyzing the Controller





# Minimize the effect of disturbance

- $V_{t+1} = 0.7v_t + 0.5 * 3.3 * (r_t - v_t) - w$ 
  - $w = -5$  or  $+5$
- 39.74
  - Close to 42.31
  - Better than
    - 33
    - 66
- Cost
  - SS error
  - oscillation

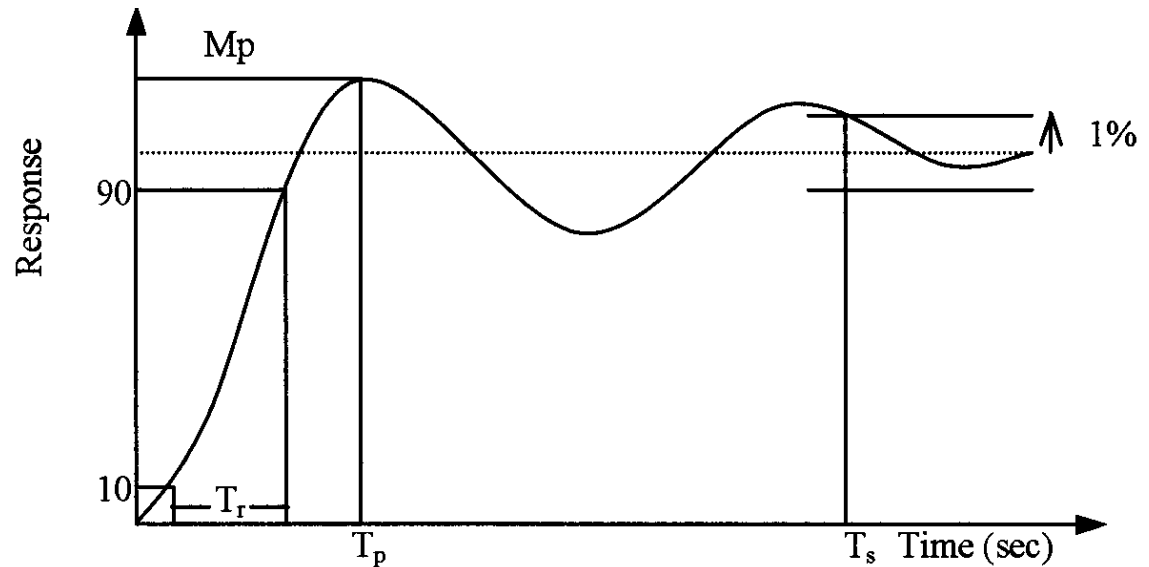
Time	$v_t$	$u_t$	$v_t$	$u_t$
0	20.00	45.00	20.00	45.00
1	31.50	45.00	41.50	28.05
2	39.55	34.49	48.08	6.35
3	39.93	33.24	41.83	26.97
4	39.57	34.42	47.76	7.38
5	39.91	33.30	42.13	25.99
6	39.59	34.37	47.48	8.31
7	39.89	33.35	42.39	25.10
8	39.60	34.32	47.23	9.15
9	39.88	33.40	42.63	24.30
10	39.62	34.27	47.00	9.91
...				
45	39.76	33.78	44.52	18.09
46	39.72	33.91	45.21	15.82
47	39.76	33.78	44.55	17.97
48	39.73	33.91	45.17	15.92
49	39.76	33.79	44.58	17.87
50	39.73	33.90	45.14	16.02
...				
ss	39.74	33.85	44.87	16.92

# General Control System

- Objective
  - Causing output to track a reference even in the presence of
    - Measurement noise
    - Model error
    - Disturbances
- Metrics
  - Stability
    - Output remains bounded
  - Performance
    - How well an output tracks the reference
  - Disturbance rejection
  - Robustness
    - Ability to tolerate modeling error of the plant

# Performance (generally speaking)

- Rise time
  - Time it takes from 10% to 90%
- Peak time
- Overshoot
  - Percentage by which Peak exceeds final value
- Settling time
  - Time it takes to reach 1% of final value

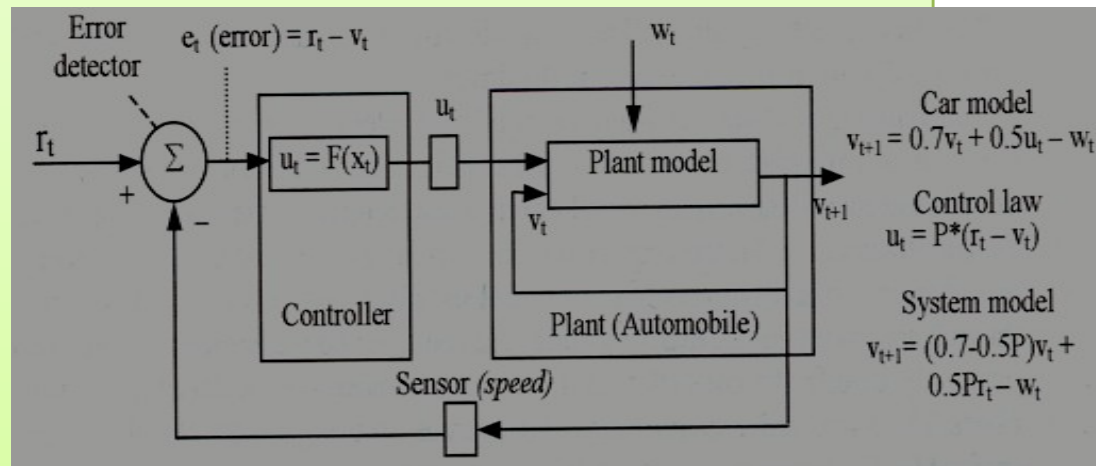


# Plant modeling is difficult

- **May need to be done first**
- **Plant is usually on continuous time**
  - **Not discrete time**
    - E.g. car speed continuously react to throttle position, not at discrete interval
  - **Sampling period must be chosen carefully**
    - To make sure “nothing interesting” happen in between
    - I.e. small enough
- **Plant is usually non-linear**
  - E.g. shock absorber response may need to be 8<sup>th</sup> order differential
- **Iterative development of the plant model and controller**
  - Have a plant model that is “good enough”

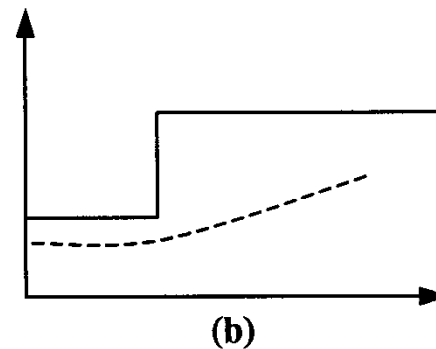
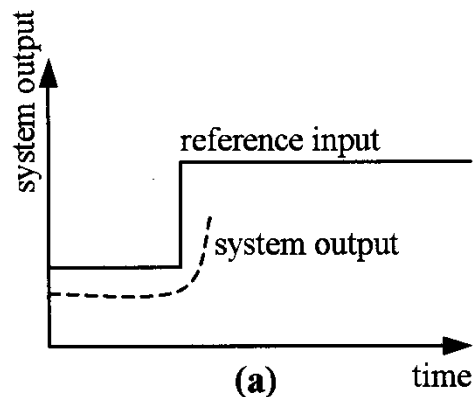
# Controller Design: P

- Proportional controller
  - A controller that multiplies the tracking error by a constant
    - $u_t = P * (r_t - v_t)$
  - Close loop model with a linear plant
    - E.g.  $v_{t+1} = (0.7 - 0.5P) * v_t + 0.5P * r_t - w_t$
- P affects
  - Transient response
    - Stability, oscillation
  - Steady state tracking
    - As large as possible
  - Disturbance rejection
    - As large as possible



# Controller Design: PD

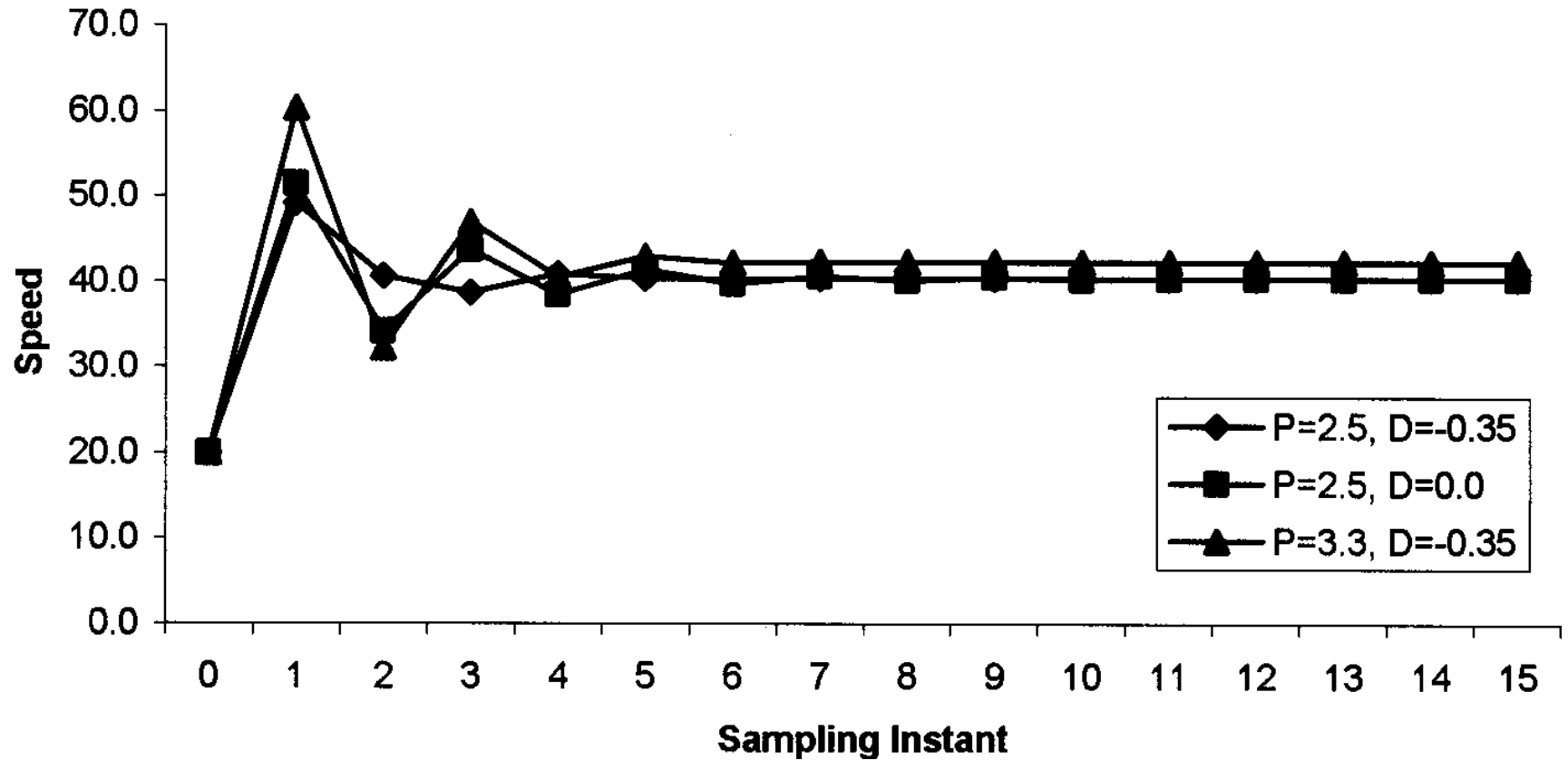
- Proportional and Derivative control
  - $u_t = P * (r_t - v_t) + D * ((r_t - v_t) - (r_{t-1} - v_{t-1})) = P * e_t + D * (e_t - e_{t-1})$
- Consider the size of error over time
- Intuitively
  - Want to “push” more if the error is not reducing fast enough
  - Want to “push” less if the error is reducing really fast



# PD Controller

- Need to keep track of error derivative
- E.g. Cruise controller example
  - $v_{t+1} = 0.7v_t + 0.5u_t - w_t$
  - Let  $u_t = P * e_t + D * (e_t - e_{t-1})$ ,  $e_t = r_t - v_t$
  - $v_{t+1} = 0.7v_t + 0.5 * (P * (r_t - v_t) + D * ((r_t - v_t) - (r_{t-1} - v_{t-1}))) - w_t$
  - $v_{t+1} = (0.7 - 0.5 * (P + D)) * v_t + 0.5D * v_{t-1} + 0.5 * (P + D) * r_t - 0.5D * r_{t-1} - w_t$
  - Assume reference input and disturbance are constant, the steady-state speed is
    - $V_{ss} = (0.5P / (1 - 0.7 + 0.5P)) * r$
    - Does not depend on D!!!
- P can be set for best tracking and disturbance control
- Then D set to control oscillation/overshoot/rate of convergence

# PD Control Example





# PI Control

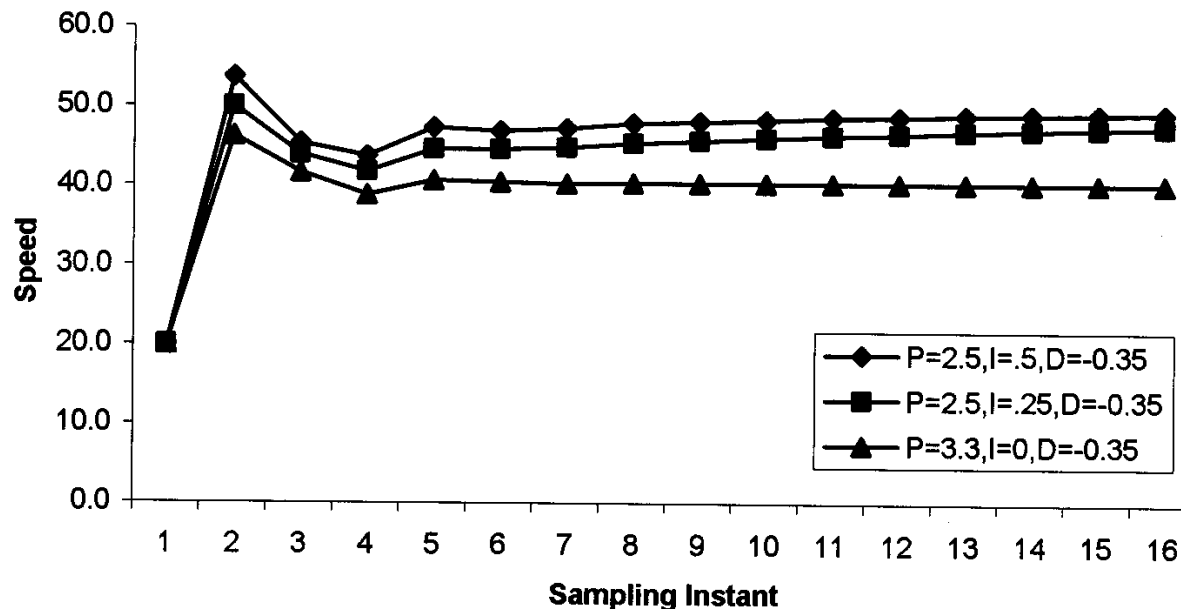
- Proportional plus integral control
  - $u_t = P * e_t + I * (e_0 + e_1 + \dots + e_t)$
- Sum up error over time
  - Ensure reaching desired output, eventually
  - $v_{ss}$  will not be reached until  $e_{ss} = 0$
- Use P to control disturbance
- Use I to ensure steady state convergence and convergence rate

# PID Controller

- Combine Proportional, integral, and derivative control

$$- u_t = P * e_t + I * (e_0 + e_1 + \dots + e_t) + D * (e_t - e_{t-1})$$

- Available off-the shelf



# Software Coding

- Main function loops forever, during each iteration
  - Read plant output sensor
    - May require A2D
  - Read current desired reference input
  - Call PidUpdate, to determine actuator value
  - Set actuator value
    - May require D2A

```
void main()
{
    double sensor_value, actuator_value, error_current;
    PID_DATA pid_data;
    PidInitialize(&pid_data);
    while (1) {
        sensor_value = SensorGetValue();
        reference_value = ReferenceGetValue();
        actuator_value =
            PidUpdate(&pid_data, sensor_value, reference_value);
        ActuatorSetValue(actuator_value);
    }
}
```

# Software Coding (continue)

- Pgain, Dgain, Igain are constants
- sensor\_value\_previous
  - For D control
- error\_sum
  - For I control

```
typedef struct PID_DATA {  
    double Pgain, Dgain, Igain;  
    double sensor_value_previous; // find the derivative  
    double error_sum; // cumulative error  
}
```

# Computation

- $u_t = P * e_t + I * (e_0 + e_1 + \dots + e_t) + D * (e_t - e_{t-1})$

```
double PidUpdate(PID_DATA *pid_data, double sensor_value,
                 double reference_value)
{
    double Pterm, Iterm, Dterm;
    double error, difference;

    error = reference_value - sensor_value;
    Pterm = pid_data->Pgain * error; /* proportional term*/
    pid_data->error_sum += error; /* current + cumulative*/
    // the integral term
    Iterm = pid_data->Igain * pid_data->error_sum;
    difference = pid_data->sensor_value_previous -
                 sensor_value;
    // update for next iteration
    pid_data->sensor_value_previous = sensor_value;
    // the derivative term
    Dterm = pid_data->Dgain * difference;
    return (Pterm + Iterm + Dterm);
```

# PID tuning

- Analytically deriving P, I, D may not be possible
  - E.g. plant not available, or too costly to obtain
- Ad hoc method for getting “reasonable” P, I, D
  - Start with a small P, I=D=0
  - Increase D, until seeing oscillation
    - Reduce D a bit
  - Increase P, until seeing oscillation
    - Reduce D a bit
  - Increase I, until seeing oscillation
- Iterate until can change anything without excessive oscillation

# Practical Issues with Computer-Based Control

- Quantization
- Overflow
- Aliasing
- Computation Delay

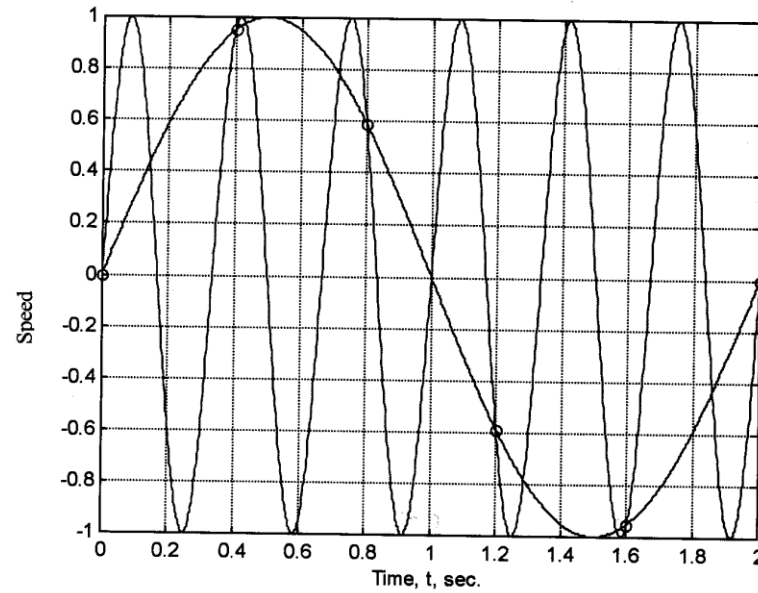
# Quantization & Overflow

- Quantization
  - Can't store 0.36 as 4-bit fractional number
  - Can only store 0.75, 0.59, 0.25, 0.00, -0.25, -0.50, -0.75, -1.00
  - Choose 0.25
    - Result in quantization error of 0.11
- Sources of quantization error
  - Operations, e.g.  $0.50 * 0.25 = 0.125$ 
    - Can use more bits until input/output to the environment/memory
  - A2D converters
- Overflow
  - Can't store  $0.75 + 0.50 = 1.25$  as 4-bit fractional number
- Solutions:
  - Use fix-point representation/operations carefully
    - Time-consuming
  - Use floating-point co-processor
    - Costly



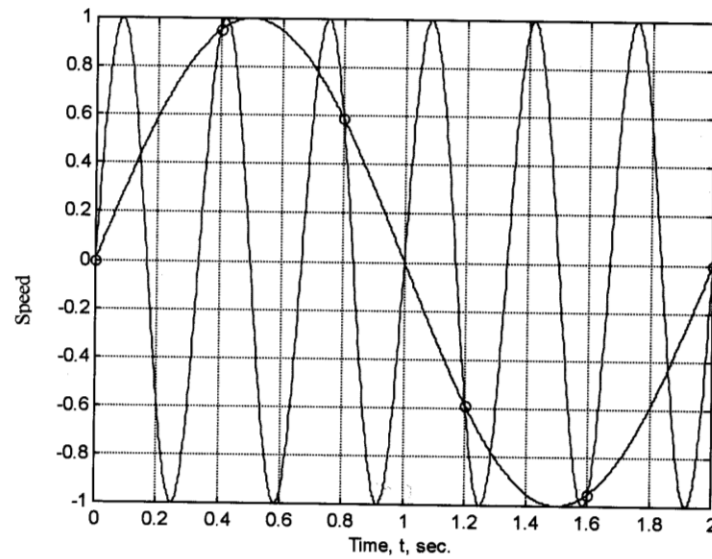
# Aliasing

- Quantization/overflow
  - Due to discrete nature of computer data
- Aliasing
  - Due to discrete nature of sampling



# Aliasing Example

- Sampling at 2.5 Hz, period of 0.4, the following are indistinguishable
  - $y(t)=1.0*\sin(6\pi t)$ , frequency 3 Hz
  - $y(t)=1.0*\sin(\pi t)$ , frequency of 0.5 Hz
- In fact, with sampling frequency of 2.5 Hz
  - Can only correctly sample signal below Nyquist frequency  $2.5/2 = 1.25$  Hz



# Computation Delay

- Inherent delay in processing
  - Actuation occurs later than expected
- Need to characterize implementation delay to make sure it is negligible
- Hardware delay is usually easy to characterize
  - Synchronous design
- Software delay is harder to predict
  - Should organize code carefully so delay is predictable and minimized
  - Write software with predictable timing behavior (be like hardware)
    - Time Trigger Architecture
    - Synchronous Software Language

# Benefit of Computer Control

- Cost!!!
  - Expensive to make analog control immune to
    - Age, temperature, manufacturing error
  - Computer control replace complex analog hardware with complex code
- Programmability!!!
  - Computer Control can be “upgraded”
    - Change in control mode, gain, are easy to do
  - Computer Control can be adaptive to change in plant
    - Due to age, temperature, ...etc
  - “future-proof”
    - Easily adapt to change in standards,..etc