1) What MPC brings to the table?

The ability to use prediction to control the output to better align the predicted outcome. In utilizing predictions, MPC can better respond to changes in the environment or disturbances. It can also handle multivariable inputs and outputs. Another key feature of MPC is that its decisions for control actions are constantly updated as new data arrives. This allows the current input to be optimized as data comes in.

2) In your opinion when should we use state space models and when should we use transfer functions?

State space models should be used for representing more complex systems. It gives an overview of what the system entails, such as if you need to manage multiple states.
For transfer functions, it's should be used for simpler systems, such as a PID controller. It allows you to see how an input will affect the output in a SISO system. Great for just focusing on what your output is.

3) In your own word explain what is a state of a system? What is state-space representation?

The state of the system is pretty much what are the internals of the system at a specific time. Let's say the system contains a variety of sensors that are collecting data as time moves forward. The state of the system at time=0 will be different at time=5. Controlling temperature? At time 0, the temperature is at 50 degrees Fahrenheit, while at time 5 its at 60 degrees Fahrenheit. Temperature is a state in this case.
As for state space representation, it uses these state variables to show how the system changes over time. Typically has a representation of how each state variable changes, and how the output will change according to these states.

$\ddot{y}(t) = y(t) + u(t) = x_1(t) + u(t)$

4) $\ddot{y}(t) - y(t) = u(t)$

Let $x_1(t) = y(t)$ $\quad \dot{x}_1(t) = \dot{y}(t) = \underline{x_2(t)}$

$x_2(t) = \dot{y}(t)$ $\quad \dot{x}_2(t) = \ddot{y}(t) = \underline{x_1(t) + u(t)}$

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = Ax(t) + Bu(t)$$

$$y(t) = Cx(t)$$

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

Problem 4

```
%Verification of the continuous state space model.
A = [0 1;1 0];
B = [0;1];
C = [1 0];
D = [0];
%State space model in time.
sys = ss(A,B,C,D)
```

```
sys =

  A =
       x1  x2
   x1   0   1
   x2   1   0

  B =
       u1
   x1   0
   x2   1

  C =
       x1  x2
   y1   1   0

  D =
       u1
   y1   0

Continuous-time state-space model.
Model Properties
```

```
%State space model in discrete. Sample time unspecified
%sysd = ss(A,B,C,D,-1)
```

```
%State space model converted from continuous to discrete.
%Sampling time of 1 second
sysdd = c2d(sys,1)
```

```
sysdd =

  A =
          x1     x2
   x1  1.543  1.175
   x2  1.175  1.543

  B =
           u1
   x1  0.5431
   x2   1.175

  C =
       x1  x2
   y1   1   0

  D =
       u1
   y1   0
```

```
Sample time: 1 seconds
Discrete-time state-space model.
Model Properties
```

```
%Problem 5
clc;
clear;
close all;
%State space matrices definition
A = [0.6 0.2; -0.4 -0.1];
B = [0.2; 0.2];
C = [1 0];
D = 0;

%System definition, Sample time at 0.05s
sys = ss(A,B,C,D,0.05);
%Initial Condition
x0 = [1;0];
%Time interval to 5 seconds
t = 0:0.05:5;
%input u(k) = 5. size of input set to equal size of time
u = 5 * ones(size(t));
%lsimplot(sys,u,t,x0);
%getting x(k). lsimplot just returns y
[y,tOut,x] = lsim(sys,u,t,x0);

plot(tOut,x);
legend("x1(k)","x2(k)");
```