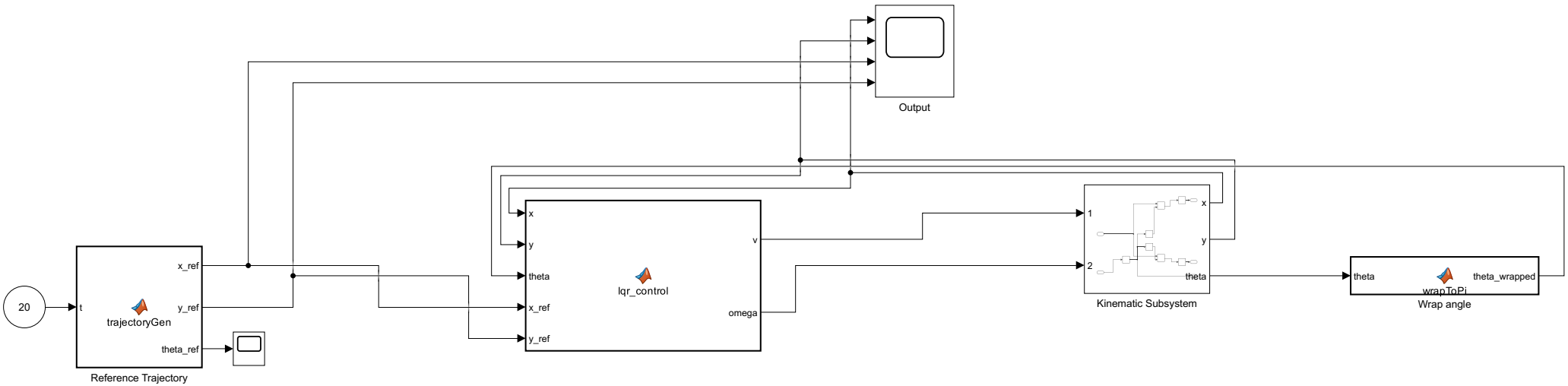
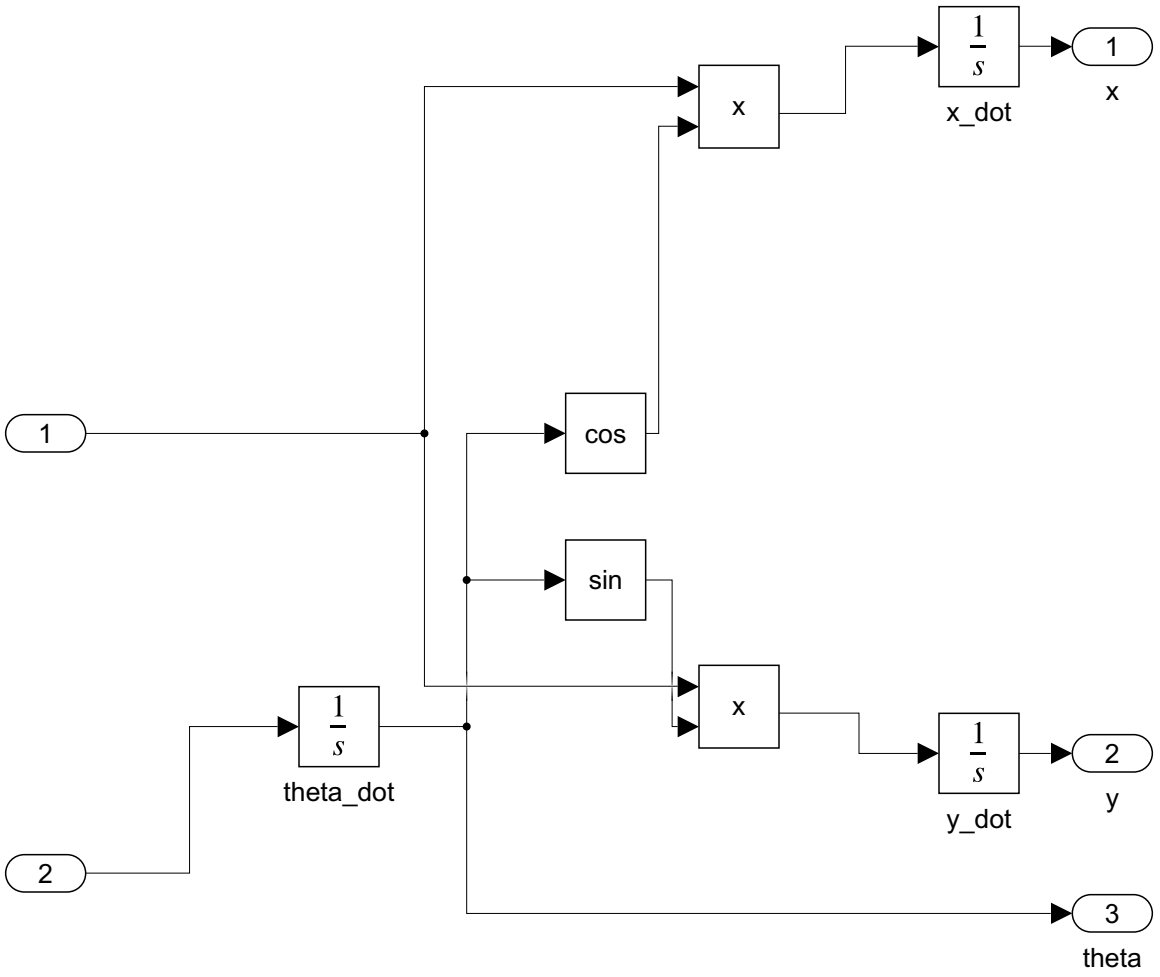

Output
20
t
x_ref
y_ref
theta_ref
trajectoryGen
Reference Trajectory
x
y
theta
x_ref
y_ref
lqr_control
v
omega
1
2
x
y
theta
Kinematic Subsystem
theta
wrapToPi
theta_wrapped
Wrap angle


1
2
cos
sin
x
x
1/s
x_dot
1/s
theta_dot
1/s
y_dot
1
x
2
y
3
theta

```
function [v, omega] = lqr_control(x, y, theta, x_ref, y_ref)
%#codegen
robotState = [x; y; theta];
targetPoint = [x_ref; y_ref];
[v, omega] = LQRController(robotState, targetPoint);

function [v, omega] = LQRController(robotState, targetPoint)
    % Inputs:
    %   robotState = [x; y; theta]
    %   targetPoint = [x_target; y_target]

    x = robotState(1);
    y = robotState(2);
    theta = robotState(3);

    x_target = targetPoint(1);
    y_target = targetPoint(2);

    % LQR Gain matrix (2x3)
    K = [1.5, 0.0, 0.0;
         0.0, 1.5, 2.0];

    % Global error
    dx = x_target - x;
    dy = y_target - y;
    error_global = [dx; dy];

    % Rotation matrix to robot frame
    c = cos(theta);
    s = sin(theta);
    R = [c, s; -s, c];
    error_robot = R * error_global;

    % Theta error
    angle_to_target = atan2(dy, dx);
    theta_error = atan2(sin(angle_to_target - theta), cos(angle_to_target - theta));

    % Form error state
    error_state = [error_robot; theta_error];

    % Control input via LQR gains
    v_r = K(1, :) * error_state;
    v_l = K(2, :) * error_state;

    % Convert to unicycle model control inputs
    v = (v_r + v_l) / 2;
    omega = (v_r - v_l) / 0.2;  % 0.2m wheel base

    % Saturation
    v = max(0.2, min(v, 5.0));
    omega = min(max(omega, -pi), pi);
end
end
```

```
function [x_ref, y_ref, theta_ref] = trajectoryGen(t)
    radius = 5;
    omega_ref = 0.2;
    x_ref = radius * cos(omega_ref * t);
    y_ref = radius * sin(omega_ref * t);
    theta_ref = omega_ref * t + pi/2;
end
```

```
function theta_wrapped = wrapToPi(theta)
    theta_wrapped = mod(theta + pi, 2*pi) - pi;
end
```