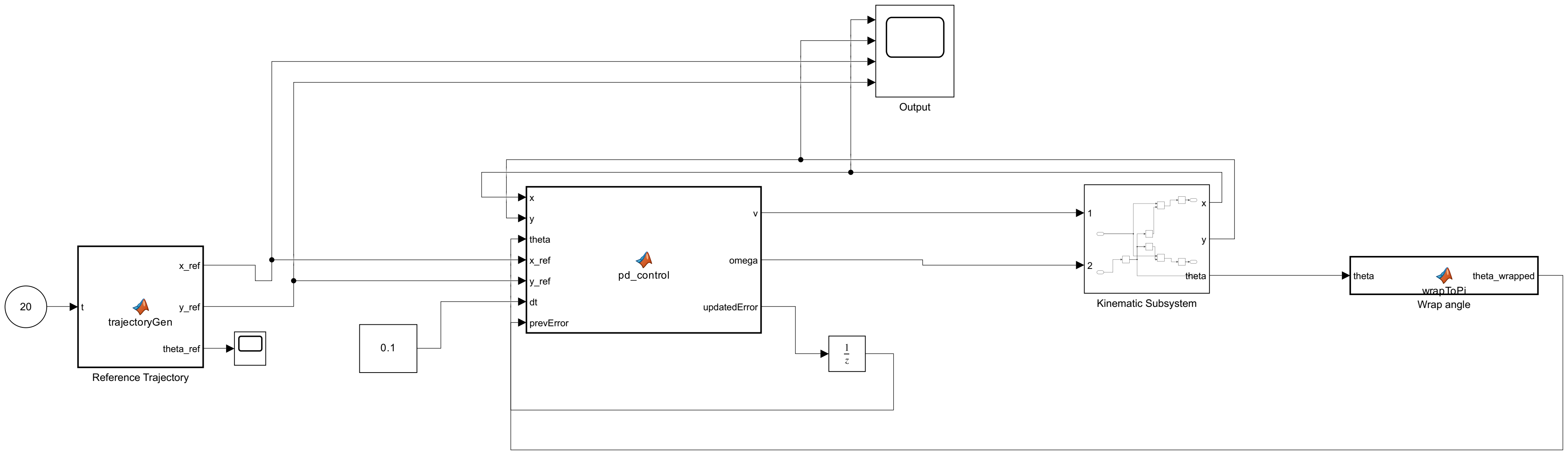
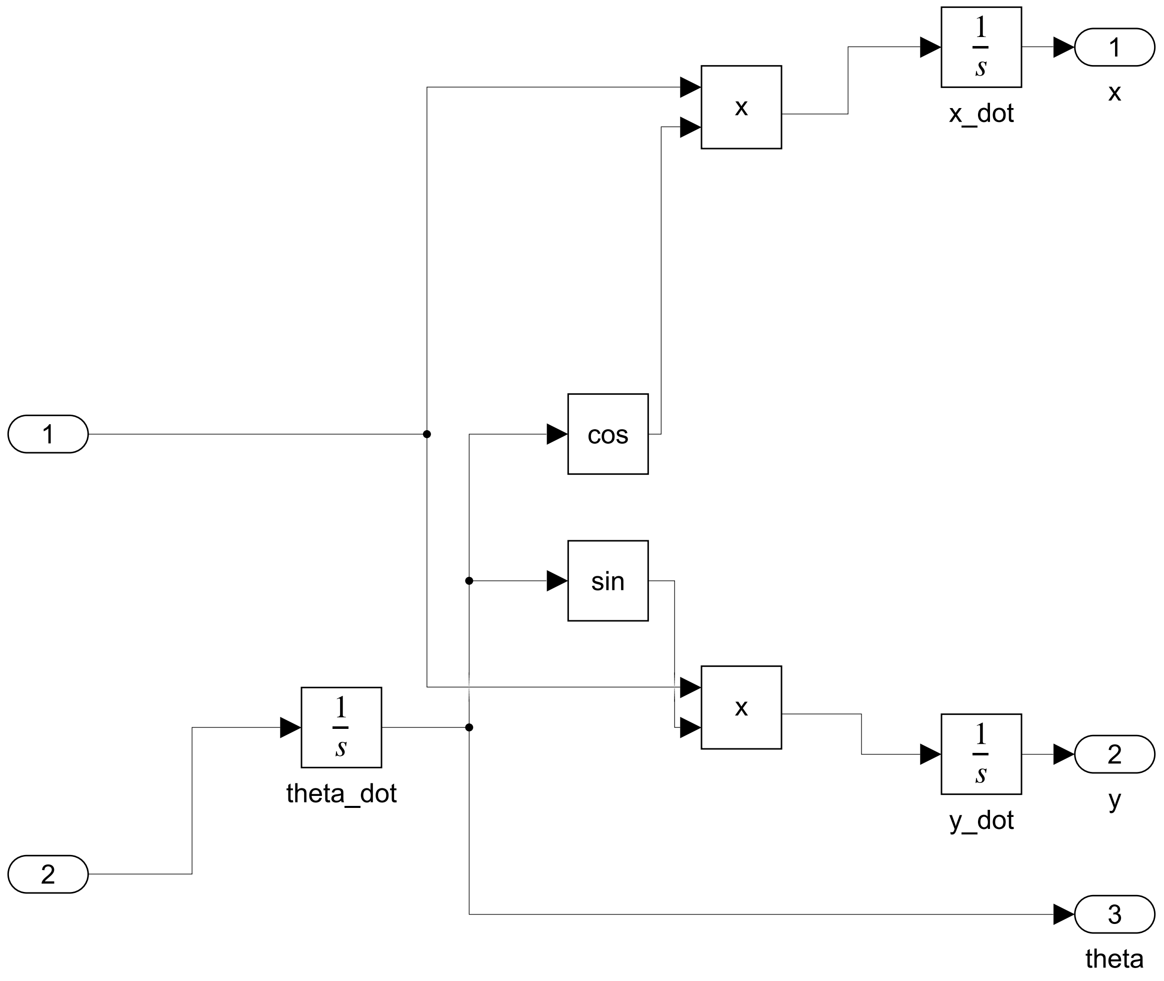

Output
20
t
trajectoryGen
x_ref
y_ref
theta_ref
Reference Trajectory
0.1
x
y
theta
x_ref
y_ref
dt
prevError
pd_control
v
omega
updatedError
1/z
1
2
x
y
theta
Kinematic Subsystem
theta
wrapToPi
theta_wrapped
Wrap angle


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

```
function [v, omega, updatedError] = pd_control(x, y, theta, x_ref, y_ref, dt, prevError)
%#codegen
robotState = [x; y; theta];
targetPoint = [x_ref; y_ref];
[v, omega, updatedError] = PD_Controller(robotState, targetPoint, dt, prevError);

    function [v, omega, updatedPrevAngleError] = PD_Controller(robotState, targetPoint, dt, prevAngleError)
    x = robotState(1); y = robotState(2); theta = robotState(3);
    x_target = targetPoint(1); y_target = targetPoint(2);

    dx = x_target - x;
    dy = y_target - y;
    dist_to_target = sqrt(dx^2 + dy^2);

    angle_to_target = atan2(dy, dx);
    angle_error = atan2(sin(angle_to_target - theta), cos(angle_to_target - theta));
    angle_error_deriv = (angle_error - prevAngleError) / dt;
    updatedPrevAngleError = angle_error;

    % PD Gains
    kp_v = 1.5;
    kp_omega = 2.0;
    kd_omega = 1.0;

    v = min(max(kp_v * dist_to_target, 0.2), 5.0);
    omega = min(max(kp_omega * angle_error + kd_omega * angle_error_deriv, -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
```