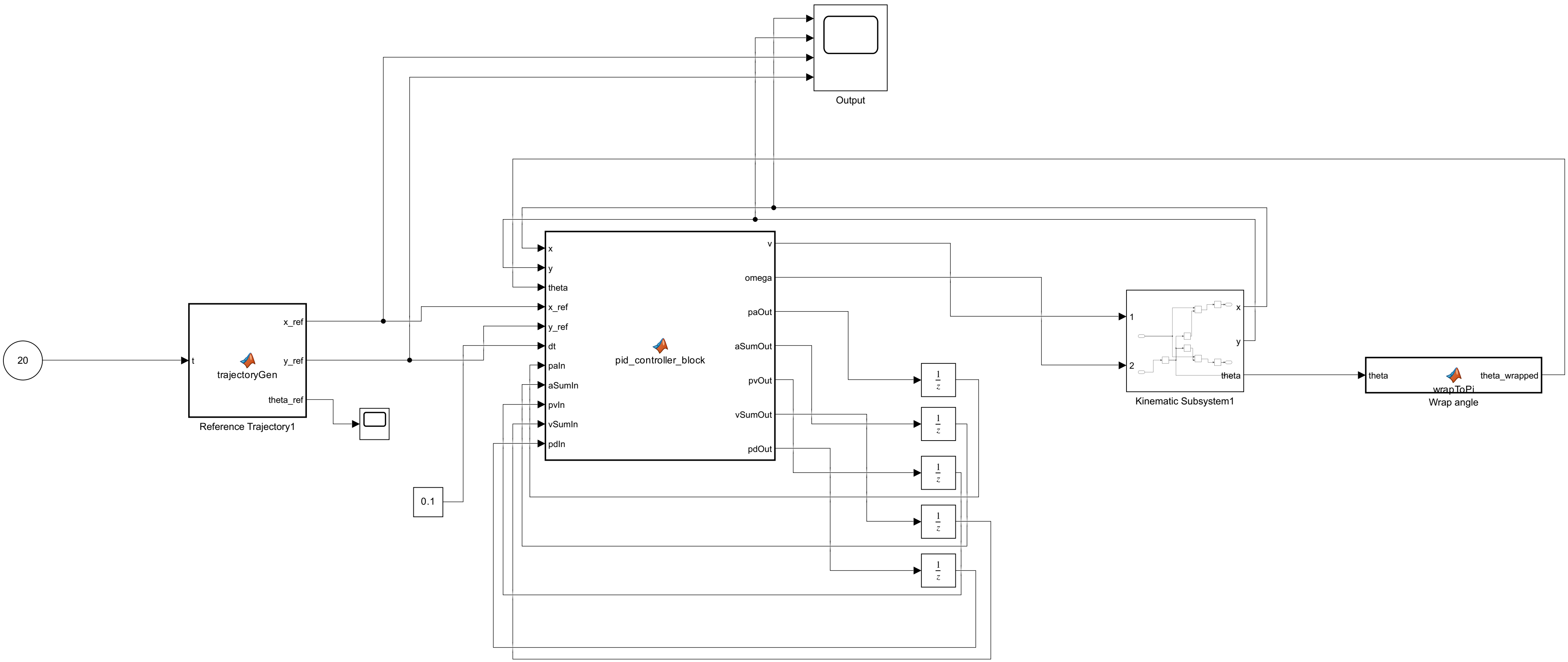
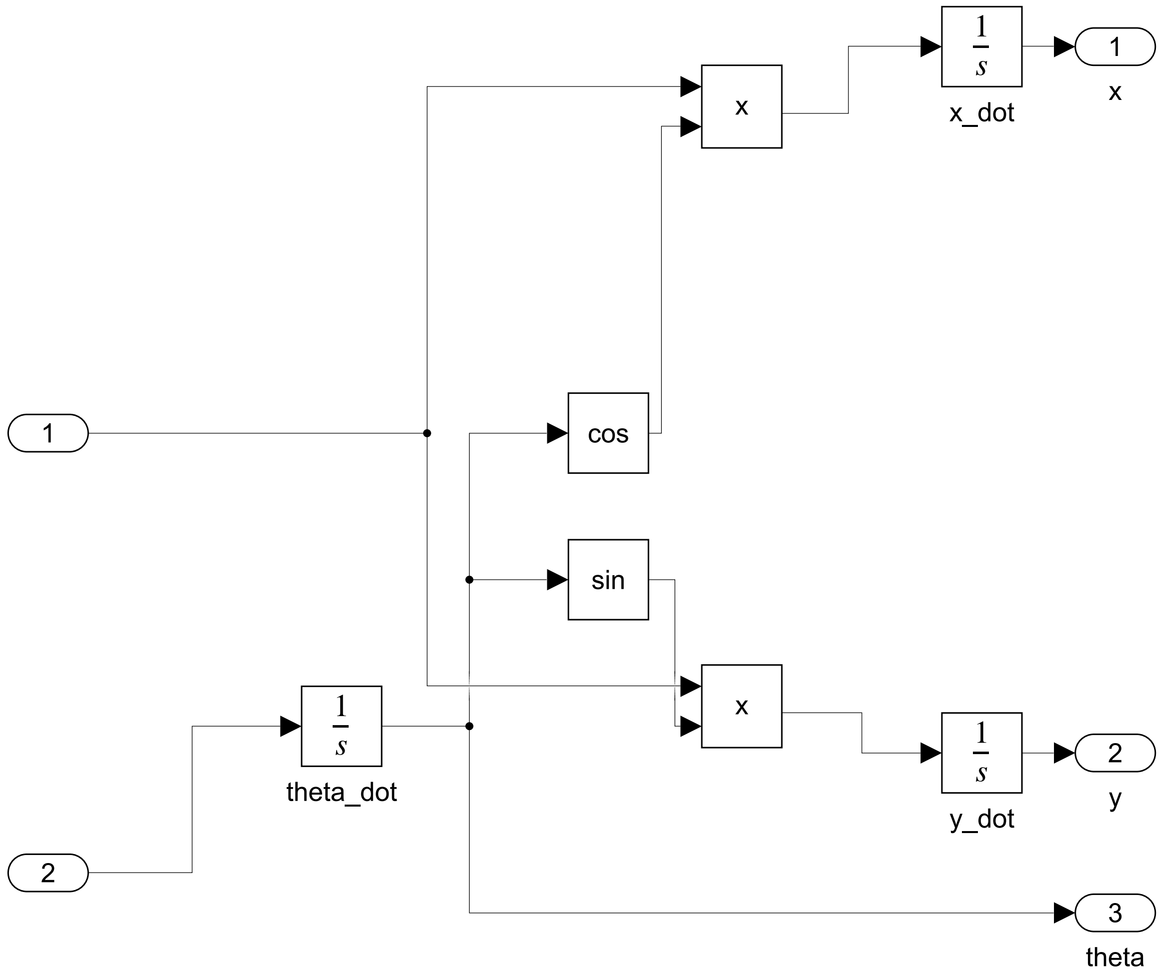


Output
20
t
trajectoryGen
x_ref
y_ref
theta_ref
Reference Trajectory1
0.1
x
y
theta
x_ref
y_ref
dt
paIn
aSumIn
pvIn
vSumIn
pdIn
pid_controller_block
v
omega
paOut
aSumOut
pvOut
vSumOut
pdOut
1
2
x
y
theta
Kinematic Subsystem1
theta
wrapToPi
theta_wrapped
Wrap angle


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

```
function [v, omega, paOut, aSumOut, pvOut, vSumOut, pdOut] = pid_controller_block(x, y, theta, x_ref, y_ref, dt, paIn, aSumIn, pvI
%#codegen
robotState = [x; y; theta];
targetPoint = [x_ref; y_ref];
[v, omega, paOut, aSumOut, pvOut, vSumOut, pdOut] = PIDController(robotState, targetPoint, dt, paIn, aSumIn, pvIn, vSumIn, pdIn);

    function [v, omega, prevAngleError, angleErrorSum, prevVError, vErrorSum, prevDistance] = PIDController(robotState, targetPoin
    % Unpack robot state
    x = robotState(1);
    y = robotState(2);
    theta = robotState(3);

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

    % Desired forward velocity
    v_desired = 5.0;

    %% --- Angular control (omega) ---
    angle_to_target = atan2(y_target - y, x_target - x);
    angle_error = atan2(sin(angle_to_target - theta), cos(angle_to_target - theta));
    angle_error_deriv = (angle_error - prevAngleError) / dt;
    angleErrorSum = angleErrorSum + angle_error * dt;
    angleErrorSum = min(max(angleErrorSum, -1.0), 1.0); % Anti-windup
    prevAngleError = angle_error;

    % PID gains for angular velocity
    kp_omega = 2.0;
    ki_omega = 1.0;
    kd_omega = 0.6;
    omega = kp_omega * angle_error + ki_omega * angleErrorSum + kd_omega * angle_error_deriv;
    omega = min(max(omega, -pi), pi);

    %% --- Linear velocity control ---
    dist_to_target = sqrt((x_target - x)^2 + (y_target - y)^2);
    v_current = (prevDistance - dist_to_target) / dt;
    prevDistance = dist_to_target;

    v_error = v_desired - v_current;
    v_error_deriv = (v_error - prevVError) / dt;
    vErrorSum = vErrorSum + v_error * dt;
    vErrorSum = min(max(vErrorSum, -2.0), 2.0); % Anti-windup
    prevVError = v_error;

    % PID gains for linear velocity
    kp_v = 1.0;
    ki_v = 0.5;
    kd_v = 0.2;
    v = kp_v * v_error + ki_v * vErrorSum + kd_v * v_error_deriv;

    % Reduce speed when not aligned with target
    v = v * cos(angle_error);

    % Clamp final velocity
    v = min(max(v, 0.2), 5.0);
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
```