34 Dead-Reckoning Error

An unmanned, untethered underwater vehicle operating near large metallic marine structures has to navigate without a magnetic compass, and in some cases without any acoustic ranging available. In these cases, a viable approach is dead-reckoning based on the yaw gyro and on the body-referenced velocity over ground, from a Doppler acoustic sensor. Specifically, if θ is yaw angle, U is forward speed, and $[x, y]$ is the position in a global frame of reference, dead-reckoning entails propagating

$$\dot{x} = U \cos\theta$$
$$\dot{y} = U \sin\theta.$$

Let us consider a yaw rate gyro with a noise variance of $0.0025 rad^2/s^2$; it is Gaussian noise. The speed sensor is also subject to Gaussian noise, but this has variance $0.0001 m^2/s^2$. The yaw rate sensor is measuring $r = \dot{\theta}$ and the speed sensor is measuring U.

The platform on which these sensors is mounted goes through a *known* circular trajectory, with $U = 1.5 m/s$ and $r = \pi/150\ rad/s$, and the sensors are sampled five times per second.

Question: Assuming that at time zero, there is no error in x, y, θ, what are the mean and standard deviation of the x, y, and range ($\sqrt{x^2 + y^2}$) errors at the end of one complete circle? Please also confirm that your errors are small with zero sensor noise - this should easily be several meters or less, even if you use a basic forward Euler integration rule.

As indicated in the attached code, this is a Monte Carlo problem, where we inject a random number to go with with every noisy measurement. The figure below shows twenty realizations, and the scatter in the endpoints. You see that the paths get progressively worse as we move around the circle and more erroneous measurements are used in the integrations.

For 10,000 realizations, statistics computed are:

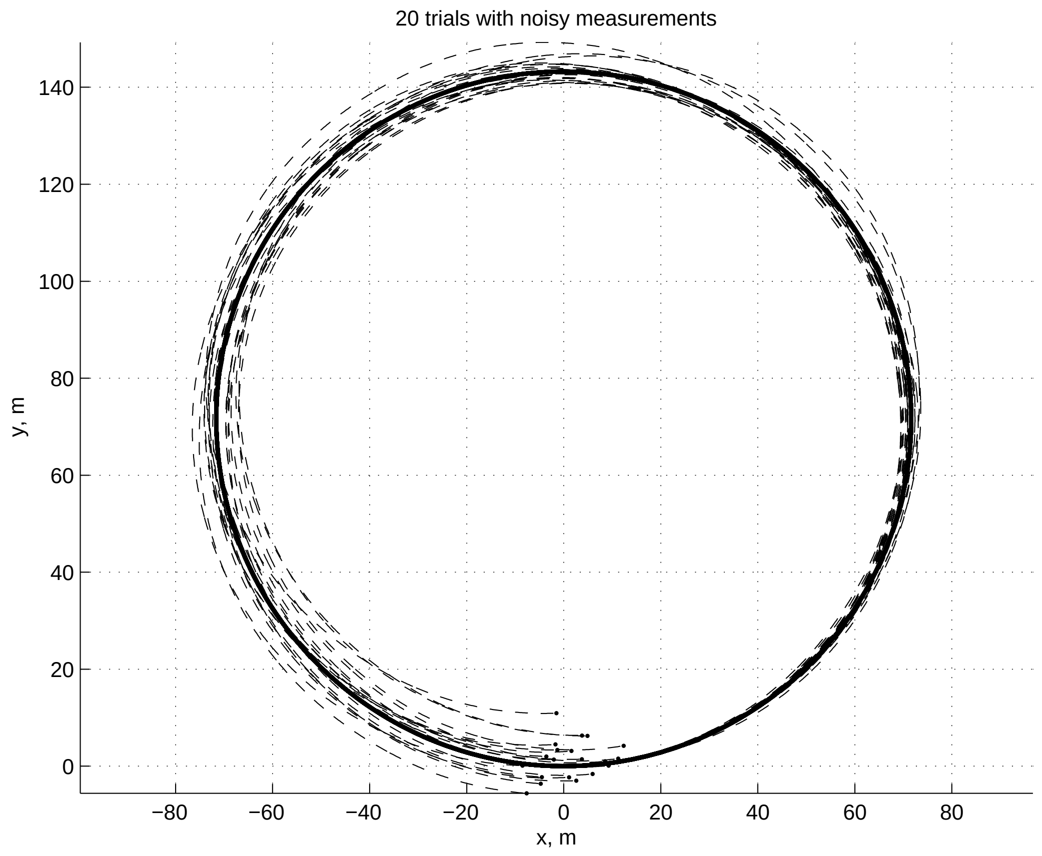
x-error:	*-0.03 m mean*	*6.81 m standard deviation*
y-error:	*0.15 m mean*	*3.88 m standard deviation*
range error:	*6.81 m mean*	*3.89 m standard deviation.*

The mean x and y errors should be close to zero, but there are not enough realizations to show this clearly.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dead reckoning.

% FSH MIT Mechanical Engineering April 2008

clear all;
clc;
```



```
dt = 0.2 ; % time step, s
U = 1.5 ; % speed, m/s
r = 2*pi/300 ; % yaw rate, rad/s
Uvar = 0.0025 ; % speed sensor variance, (m/s)^2
Rvar = 0.0001 ; % yaw rate variance (rad/s)^2
N = 20 ;    % number of trials (a small number if showing plots,
            % or a big number if you want good statistics)
plotFlag = 1 ; % set to one to show plots

T = 2*pi/r ; % time to complete an entire circle

figure(1);clf;hold off;
figure(2);clf;hold off;

tic ;
for ii = 1:N, % loop through the number of realizations

    % initialize the various states - first time step
    x(1) = 0 ;  % x-position
    y(1) = 0 ; % y-position
    th(1) = 0 ; % heading
    xp(1) = 0 ; % "pure" x-position : created without sensor noise
    yp(1) = 0 ; % "pure" y-position
    thp(1) = 0 ; % "pure" heading

    rMeasure(1) = sqrt(Rvar)*randn + r ; % first measurement of r
    UMeasure(1) = sqrt(Uvar)*randn + U ; % first measurement of U

    ct = 1 ;
    for tt = dt:dt:T,   % step through all the times needed to complete
        % a circle

        ct = ct + 1 ;

        UMeasure(ct) = sqrt(Uvar)*randn + U ; % get measurements
        rMeasure(ct) = sqrt(Rvar)*randn + r ;

        % propagate the states based on the noisy measurements
        % (using a one-shot modified Euler step)

%        th(ct) = th(ct-1) + dt*rMeasure(ct)/2 + dt*rMeasure(ct-1)/2 ;

%        x(ct) = x(ct-1) + dt*UMeasure(ct)*cos(th(ct))/2 + ...
%            dt*UMeasure(ct-1)*cos(th(ct-1))/2;

%        y(ct) = y(ct-1) + dt*UMeasure(ct)*sin(th(ct))/2 + ...
%            dt*UMeasure(ct-1)*sin(th(ct-1))/2;

        % a regular forward Euler - testing testing!!!*******
        % this is what most of the students did S08
        th(ct) = th(ct-1) + dt*rMeasure(ct-1) ;
        x(ct) = x(ct-1) + ...
            dt*UMeasure(ct-1)*cos(th(ct-1));
        y(ct) = y(ct-1) + ...
            dt*UMeasure(ct-1)*sin(th(ct-1));


        % for the first realization, compute also a "pure"
        % trajectory, i.e., without sensor noise - this will
        % confirm that our integration method is OK.
        if ii == 1,
            thp(ct) = thp(ct-1) + dt*r ;
            xp(ct) = xp(ct-1) + dt*U*cos(thp(ct))/2 + ...
                dt*U*cos(thp(ct-1))/2;

            yp(ct) = yp(ct-1) + dt*U*sin(thp(ct))/2 + ...
                dt*U*sin(thp(ct-1))/2;
        end;

    end;

    tvec = 0:dt:dt*(length(x)-1) ;

    % collect the errors for each realization
    xerr(ii) = x(end);
    yerr(ii) = y(end);
    rerr(ii) = sqrt(x(end)^2 + y(end)^2);

    % here are the errors with the "pure" trajectory
    xperr = xp(end);
    yperr = yp(end) ;
    rperr = sqrt(xperr(end)^2 + yperr(end)^2);

    if plotFlag,
        figure(1);hold on;
        plot(tvec,x,tvec,y,tvec,th*20) ;
        legend('x, m','y, m','\theta*20, rad/s',3);
        xlabel('sec');

        figure(2);hold on;
        plot(x,y,'--',x(end),y(end),'k.');
        plot(xp,yp,'r','LineWidth',2);
        axis('equal');
        grid on;
        xlabel('x, m');
        ylabel('y, m');
        title(sprintf('%d trials with noisy measurements',N));

        disp(sprintf('ii:  %d     Range Error:  %g m.', ii, rerr(ii)));

    end;

end;
disp(sprintf('Elapsed time: %g sec.', toc));
disp(sprintf('[xerr yerr rangeErr] with NO NOISE:  %g %g %g m.',...
    xperr,yperr,rperr));
disp(sprintf('Trials:  %d', length(xerr)));
disp(sprintf('Mean of [xerr yerr rangeErr]:  %g %g %g m.',mean(xerr),...
    mean(yerr), mean(rerr)));
disp(sprintf('Stddev of [xerr yerr rangeErr]:  %g %g %g m.',std(xerr),...
    std(yerr), std(rerr)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```