Contents


```
clear
clc

rng(0)
```

```
syms r_bar w_r theta_bar w_theta
syms sigma_r sigma_theta

% polar measurements
r = r_bar + w_r;
theta = theta_bar + w_theta;

% polar covariance
R_i = diag([ sigma_r^2 sigma_theta^2 ]);
R_fn = matlabFunction(R_i);

% cartesian coordinates
x = r * cos(theta);
y = r * sin(theta);

% expected cartesian
x_bar = r_bar * cos(theta_bar);
y_bar = r_bar * sin(theta_bar);
x_bar_fn = matlabFunction(x_bar);
y_bar_fn = matlabFunction(y_bar);

% linearized cartesian about expected values
dx_bar = jacobian(x_bar, [r_bar theta_bar]);
dy_bar = jacobian(y_bar, [r_bar theta_bar]);
dx_bar_fn = matlabFunction(dx_bar);
dy_bar_fn = matlabFunction(dy_bar);

% linearized cartesian covariance
Rc = [dx_bar; dy_bar] * R_i * [dx_bar; dy_bar].';
Rc_fn = matlabFunction(Rc);

% inputs part i
r_bar_val_i = 76;
theta_bar_val_i = -3*pi/180;
sigma_r_val_i = 1;
sigma_theta_val_i = pi/180;

% compute
H = [ dx_bar_fn(r_bar_val_i, theta_bar_val_i);
```

```
      dy_bar_fn(r_bar_val_i, theta_bar_val_i) ];
dr = [ r_bar_val_i - r_bar_val_i ];
dtheta = [ theta_bar_val_i - theta_bar_val_i ];
xy_bar = [ x_bar_fn(r_bar_val_i, theta_bar_val_i); y_bar_fn(r_bar_val_i, theta_bar_val_i) ];
z_lin_i  = xy_bar + H * [dr; dtheta];
P_lin_i = Rc_fn(r_bar_val_i, sigma_r_val_i, sigma_theta_val_i, theta_bar_val_i);

% inputs part ii
r_bar_val_ii = 76;
theta_bar_val_ii = -3*pi/180;
sigma_r_val_ii = 1;
sigma_theta_val_ii = 15*pi/180;

% compute
H = [ dx_bar_fn(r_bar_val_ii, theta_bar_val_ii);
      dy_bar_fn(r_bar_val_ii, theta_bar_val_ii) ];
dr = [ r_bar_val_ii - r_bar_val_ii ];
dtheta = [ theta_bar_val_ii - theta_bar_val_ii ];
xy_bar = [ x_bar_fn(r_bar_val_ii, theta_bar_val_ii); y_bar_fn(r_bar_val_ii, theta_bar_val_ii) ];
z_lin_ii  = xy_bar + H * [dr; dtheta];
P_lin_ii = Rc_fn(r_bar_val_ii, sigma_r_val_ii, sigma_theta_val_ii, theta_bar_val_ii);
```

part b: unscented transform

```
% cholesky factorize
R_i = R_fn(sigma_r_val_i, sigma_theta_val_i);

[z_UT_i, P_UT_i] = unscented_transform(r_bar_val_i, theta_bar_val_i, R_i);

% cholesky factorize
R_ii = R_fn(sigma_r_val_ii, sigma_theta_val_ii);

[z_UT_ii, P_UT_ii] = unscented_transform(r_bar_val_ii, theta_bar_val_ii, R_ii);
```

part c: large random vectors

```
N = 1e7;
```

Case i

```
w = mvnrnd([r_bar_val_i; theta_bar_val_i], R_i, N);
z = [ w(:,1) .* cos(w(:,2)), w(:,1) .* sin(w(:,2)) ];
P_sto_i = cov(z);

disp('Case i:')
disp('Linearized mean and covariance:');
z_lin_i'
P_lin_i

disp('Unscented mean and covariance:');
z_UT_i
P_UT_i

disp('True mean and covariance:');
```

```
mean(z)
P_sto_i
```

```
Case i:
Linearized mean and covariance:

ans =

         75.8958446413476         -3.97753267446373


P_lin_i =

         1.00208022865585        0.0396931273239757
       0.0396931273239757          1.75738998803218

Unscented mean and covariance:

z_UT_i =

         75.8842850327492         -3.97692686226219


P_UT_i =

         1.00234747818872        0.0396791213909831
       0.0396791213909813          1.75739072169245

True mean and covariance:

ans =

         75.8844390709919         -3.97723672573676


P_sto_i =

         1.00165308489859         0.039052110136577
        0.039052110136577            1.756663275305
```

Case ii

```
w = mvnrnd([r_bar_val_ii; theta_bar_val_ii], R_ii, N);
z = [ w(:,1) .* cos(w(:,2)), w(:,1) .* sin(w(:,2)) ];
P_sto_ii = cov(z);

disp('Case ii:')
disp('Linearized mean and covariance:');
z_lin_ii'
P_lin_ii

disp('Unscented mean and covariance:');
z_UT_ii
P_UT_ii
```

```
disp('True mean and covariance:');
mean(z)
P_sto_ii
```

```
Case ii:
Linearized mean and covariance:

ans =

          75.8958446413476         -3.97753267446373


P_lin_ii =

          2.08159916631931          20.6381415338717
          20.6381415338717          394.799199588487

Unscented mean and covariance:

z_UT_ii =

          73.2949350737035         -3.84122478100471


P_UT_ii =

          15.6110670950729          19.9290912253928
            19.92909122532          394.836341251303

True mean and covariance:

ans =

          73.3390054525815         -3.83663171984719


P_sto_ii =

          14.5962973188707           18.584811702349
           18.584811702349          369.136213687784
```

Comment on your results

```
disp('The unscented mean is much closer to the true mean than the linearized mean. ');
disp('The unscented covariance is also smaller than the the linearized covariance. ');
disp('Norm(linearized covariance) - norm(unscented covariance) = ')
norm(P_lin_ii) - norm(P_UT_ii)

disp('The unscented transform produces more accurate results than linearization with smaller uncertainty.')
```

```
The unscented mean is much closer to the true mean than the linearized mean.
The unscented covariance is also smaller than the the linearized covariance.
```

```
Norm(linearized covariance) - norm(unscented covariance) =

ans =

      1.80888077920827e-05

The unscented transform produces more accurate results than linearization with smaller uncertainty.
```

subfunctions

```
function [z_bar, Pzz] = unscented_transform(r_bar_val, theta_bar_val, P)
s = chol(P)';

% obtain lambda
a = 10^-3;
b = 2;
k = 0;
nx = 2;
nv = 0;
nz = 2;
% lambda = a^2 * (nx + 1/2) - nx;
lambda = a^2 * (nx + nv + k) - (nx + nv);

% build sigma points (still polar coordinates)
rtheta_bar = [r_bar_val; theta_bar_val];
R = rtheta_bar' ;
for i = 1 : nx
    Ri = rtheta_bar + sqrt( nx + lambda ) * s(:,i);
    R = [ R; Ri' ];
end
for i = nx + 1 : 2*nx
    Ri = rtheta_bar - sqrt( nx + lambda) * s(:, i-nx);
    R = [ R; Ri' ];
end

% push sigma points through measurement model -->
z = @(sigma) [sigma(1) * cos(sigma(2)); sigma(1) * sin(sigma(2))];
for i = 1 : 2*nx + 1
    Z(i,:) = z(R(i,:));
end

% determine weighting
w_0m = lambda / (nx + lambda);
w_im = 1 / ( 2*(nx + lambda) );
w_0c = lambda / (nx + lambda) + 1 - a^2 + b;
w_ic = 1 / ( 2*(nx + lambda) );

% recombine with the weighting
z_bar = w_0m * Z(1,:);
for i = 2 : 2*nx+1
    z_bar = z_bar + w_ic * Z(i,:);
end

% covariance
ztilde = Z - z_bar; ztilde = ztilde';
Pzz = zeros(size(nz));
for i = 1:length(ztilde)
```

```
    if i == 1
        w = w_0c;
    else
        w = w_ic;
    end
    Pzz = Pzz + w * ztilde(:,i) * ztilde(:,i)';
end

end
```

Published with MATLAB® R2020a