Robust Control Task 1 Memo - Aided Tracker

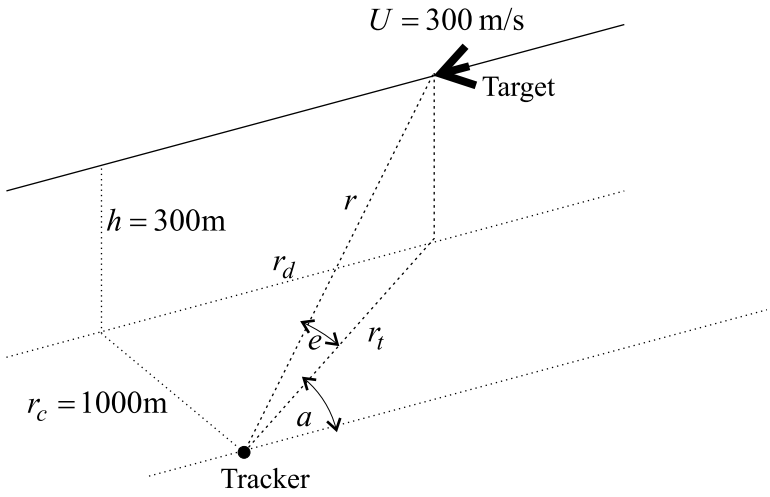
J Treurnicht

August 23, 2004

1 Tracker

1.1 Geometry, Rates and Accelerations

From the geometry



The expression of angles, reates and acceleration is from Garnell and East

$$\begin{aligned}
\dot{a} &= U \sin \frac{a}{r_t} = \frac{U \sin^2 a}{r_c} \\
\ddot{a} &= \frac{U^2 \sin 2a}{r^2 \cos^2 e} \\
\dot{e} &= \frac{U}{r} \sin e \cos a \\
\ddot{e} &= -\frac{U^2}{r^2} \tan e \left(1 - \cos^2 a(1 + 2\cos^2 e)\right)
\end{aligned}$$

We can also easily just differentiate the angle $a = \tan^{-1}\frac{y}{x}$, giving

$$\dot{a} = \frac{1}{1+\left(\frac{y}{x}\right)^2} \times \frac{-y}{x^2} \times \dot{x} = \frac{Uy}{x^2+y^2}$$

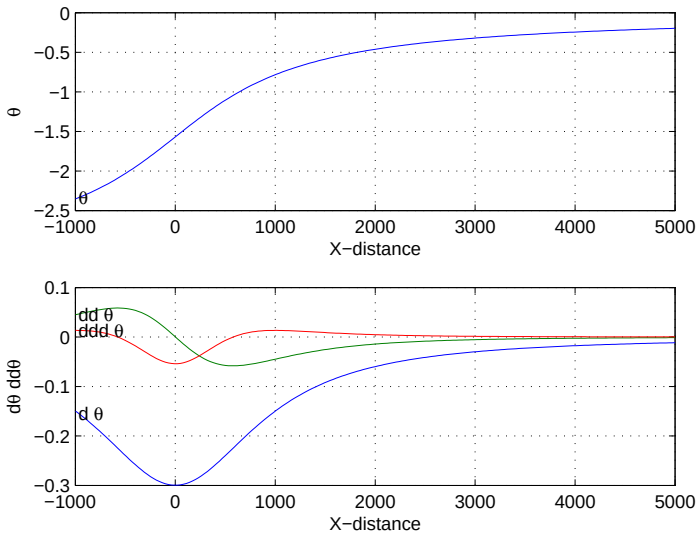
and

$$\ddot{a} = \frac{-Uy}{\left(x^2+y^2\right)^2} \times 2x \times \dot{x} = \frac{2U^2xy}{\left(x^2+y^2\right)^2}$$

and

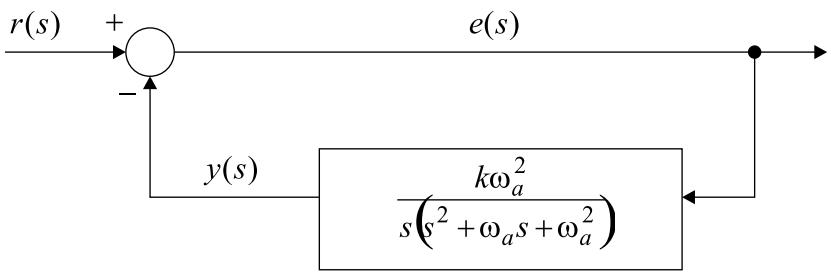
$$a^{(3)} = \frac{2U^2y\dot{x}}{\left(x^2+y^2\right)^2} - \frac{4U^2yx^2}{\left(x^2+y^2\right)^3} \times 2x \times \dot{x} = \frac{2U^3y}{\left(x^2+y^2\right)^2}\left[\frac{4x^2}{x^2+y^2} - 1\right]$$

The azimuth angle, rate, acceleration and jerk is given in the following graph



1.2 Dynamic Coefficients for Tracker

The tracker control loop can be drawn as follows



The transfer function from input $r(s)$ to error $\epsilon(s)$ is given by

$$\frac{\epsilon}{r}(s) = \frac{1}{1 + \frac{k\omega_a^2}{s(s^2+\omega_a s+\omega_a^2)}} = \frac{s^3 + \omega_a s^2 + \omega_a^2 s}{s^3 + \omega_a s^2 + \omega_a^2 s + k\omega_a^2}$$

Get the dynamic error coefficients by long division

$$\begin{array}{r|lll}
 & \frac{s}{k} & +\frac{1}{k\omega_a}\left(1-\frac{\omega_a}{k}\right)s^2 & +\frac{1}{k\omega_a^2}\left(1-\frac{\omega_a}{k}\right)^2 s^3 \quad \dots \quad \dots \\
k\omega_a^2 + \omega_a^2 s + \omega_a s^2 + s^3 & \omega_a^2 s & \omega_a s^2 & s^3 \\
 & \omega_a^2 s & \frac{\omega_a^2 s^2}{k} & \frac{\omega_a s^3}{k} \quad \frac{s^4}{k} \\
\hline
 & & \omega_a\left(1-\frac{\omega_a}{k}\right)s^2 & \left(1-\frac{\omega_a}{k}\right)s^3 \quad \frac{-1}{k}s^4 \\
 & & \omega_a\left(1-\frac{\omega_a}{k}\right)s^2 & \frac{\omega_a}{k}\left(1-\frac{\omega_a}{k}\right)s^3 \quad \frac{1}{k}\left(1-\frac{\omega_a}{k}\right)s^4 \\
\hline
 & & & \left(1-\frac{\omega_a}{k}\right)^2 s^3 \quad \dots
\end{array}$$

so our transfer function becomes

$$\frac{\epsilon}{r}(s) = \frac{s}{k} + \frac{1}{k\omega_a}\left(1-\frac{\omega_a}{k}\right)s^2 + \frac{1}{k\omega_a^2}\left(1-\frac{\omega_a}{k}\right)^2 s^3 + \dots$$

and our error constants (with values $\omega_a = 4\pi,\ k = 2$)

$$\begin{array}{rcllcll}
\frac{1}{k_p} & = & 0 & \rightarrow & k_p & = \infty & = \infty \\
\frac{1}{k_v} & = & \frac{1}{k} & \rightarrow & k_v & = k & = 2 \\
\frac{1}{k_a} & = & \frac{1}{k\omega_a}\left(1-\frac{\omega_a}{k}\right) & \rightarrow & k_a & = \frac{k^2\omega_a}{k-\omega_a} & = -4.76 \\
\frac{1}{k_j} & = & \frac{1}{k\omega_a^2}\left(1-\frac{\omega_a}{k}\right)^2 & \rightarrow & k_j & = \frac{k^3\omega_a^2}{(k-\omega_a)^2} & = 11.32
\end{array}$$

The transfer function from azimuth angle input to tracking error becomes

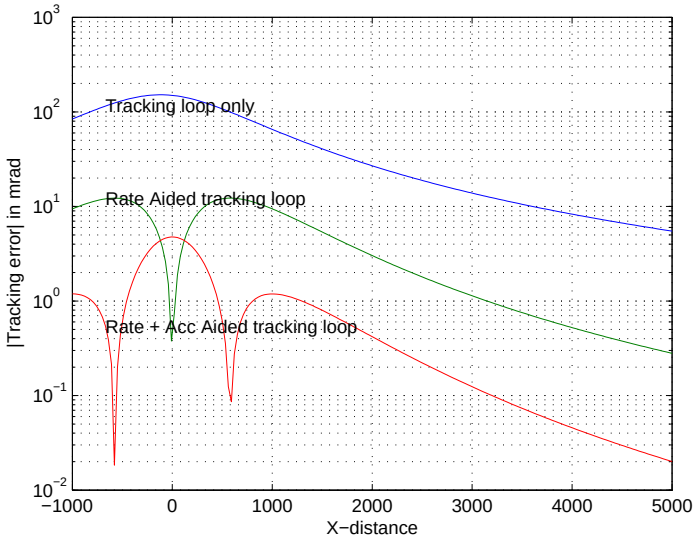
$$\frac{\epsilon}{a}(s) \approx \frac{\dot{a}}{2} - \frac{\ddot{a}}{4.76} = \frac{Ur_c}{2(r_d^2 + r_c^2)} + \frac{2U^2 r_c r_d}{4.76(r_d^2 + r_c^2)^2} = \frac{150\,000}{10^6 + r_d^2} + \frac{37\,815\,126 r_d}{(10^6 + r_d^2)^2}$$

We can now construct a simple table

r_d	$\epsilon_{\dot{a}}$	$\epsilon_{\ddot{a}}$	ϵ_T
4000	0.0088	0.00052	0.0093
3000	0.0150	0.00113	0.0016
2000	0.0300	0.00302	0.0330
1000	0.0750	0.00945	0.0845
0	0.1500	0	0.1500
−1000	0.0750	−0.00945	0.0655

The contribution of the velocity error is clearly greater than the contribution of the acceleration error (almost 10×). If we can generate a feedforward signal that emulates either $\dot{a}$ or better still $\dot{a} + \ddot{a}$, then we can eliminate the majority of the errors.

Graphically, the error has the following form, showing the potential for improvement



2 Kalman Filter

2.1 Feedforward Calculation

For a stationary cartesian Kalman Filter we can generate

$$\begin{aligned}\hat{\dot{a}}_s &= \frac{-\hat{\dot{x}}\hat{y}}{\hat{x}^2+\hat{y}^2} \\ \hat{\ddot{a}}_s &= \frac{2\hat{\dot{x}}^2\hat{x}\hat{y}}{(\hat{x}^2+\hat{y}^2)^2}\end{aligned}$$

so the feedforward equation becomes

$$a_{f,s} = \frac{1}{k_v}\frac{-\hat{\dot{x}}\hat{y}}{\hat{x}^2+\hat{y}^2} + \frac{1}{k_a}\frac{2\hat{\dot{x}}^2\hat{x}\hat{y}}{(\hat{x}^2+\hat{y}^2)^2}$$

For a rotating Kalman Filter, the x-axis is pointed towards the target – which means that $x > 0,\ x \gg |y|$. In this case we might get velocities in all axes. Again doing the differentiation

$$\begin{aligned}\dot{a} &= \frac{1}{1+\left(\frac{y}{x}\right)^2}\times\frac{\dot{y}}{x} - \frac{y\dot{x}}{x^2} = \frac{x\dot{y}-y\dot{x}}{x^2+y^2} \approx \frac{\dot{y}}{x} \\ \ddot{a} &= \frac{\dot{x}\dot{y}+x\ddot{y}-\dot{y}\dot{x}-y\ddot{x}}{x^2+y^2} - \frac{(x\dot{y}-y\dot{x})(2x\dot{x}+2y\dot{y})}{(x^2+y^2)^2} \\ &= \frac{(x\ddot{y}-y\ddot{x})(x^2+y^2)-(x\dot{y}-y\dot{x})(2x\dot{x}+2y\dot{y})}{(x^2+y^2)^2} \approx \frac{x\ddot{y}-2\dot{x}\dot{y}}{x^2}\end{aligned}$$

giving our estimates

$$\hat{\dot{a}}_r = \frac{\hat{\dot{y}}}{\hat{x}}$$

$$\ddot{\hat{a}}_r = \frac{\hat{x}\ddot{\hat{y}} - 2\dot{\hat{x}}\dot{\hat{y}}}{\hat{x}^2}$$

so the feedforward equation becomes

$$a_{f,r} = \frac{1}{k_v}\frac{\dot{\hat{y}}}{\hat{x}} + \frac{1}{k_a}\frac{\hat{x}\ddot{\hat{y}} - 2\dot{\hat{x}}\dot{\hat{y}}}{\hat{x}^2}$$

2.2 Measurement Equation

The measurement equation for the stationary filter is

$$\begin{aligned} x_m &= r_m \cos(a_m)\cos(e_m) \\ y_m &= r_m \cos(a_m)\sin(e_m) \\ z_m &= r_m \sin(e_m) \end{aligned}$$

and the measurement noise is calculated using

$$\begin{aligned} R_x &= (\sigma_r \cos(a_m)\cos(e_m))^2 + (\sigma_a r_m \sin(a_m))^2 \\ R_y &= (\sigma_r \sin(a_m)\cos(e_m))^2 + (\sigma_a r_m \cos(a_m))^2 \end{aligned}$$

The measurement equation for the rotating filter is

$$\begin{aligned} x_m &= r_m \\ y_m &= r_m \sin(a_m) \\ z_m &= r_m \sin(e_m) \end{aligned}$$

and the measurement noise is different

$$\begin{aligned} R_x &= \sigma_r^2 \\ R_y &= \sigma_a^2 r_m^2 \end{aligned}$$

2.3 Kalman Filter Equations

The Kalman filter equations are standard for both cases with

Measurement Time The state, gain and covariance are updated at measurement time

$$\begin{aligned} K(k) &= P^-(k)H^T\left[HP^-(k)H^T + R\right]^{-1} \\ P(k) &= (I - KH)P^-(k) \\ x(k) &= x^-(k) + K(k)\left[y(k) - Hx^-(k)\right] \end{aligned}$$

Predictor The state and covariance are updated before the next measurement time

$$\begin{aligned} x^-(k+1) &= \Phi x(k) \\ P^-(k+1) &= \Phi P(k)\Phi + Q \end{aligned}$$

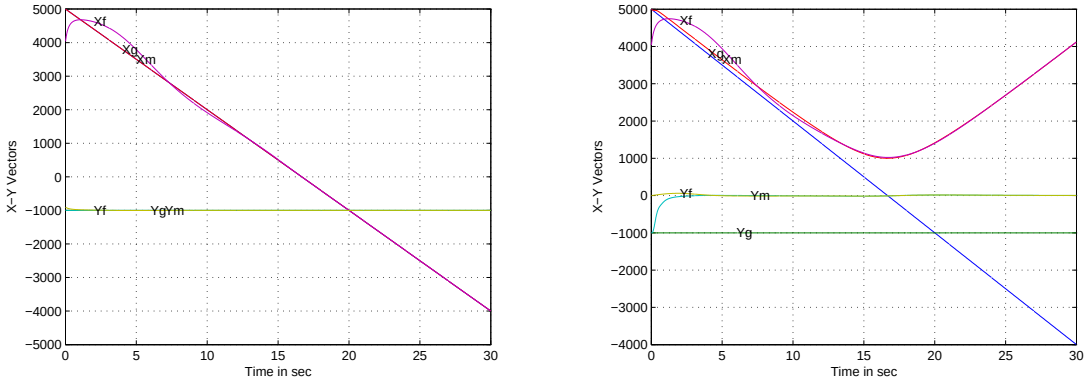
In the case of the rotating filter, the state is also turned by the attitude matrix incremental angles ωT

$$x^-(k+1) = A(\omega T)x^-(k+1)$$

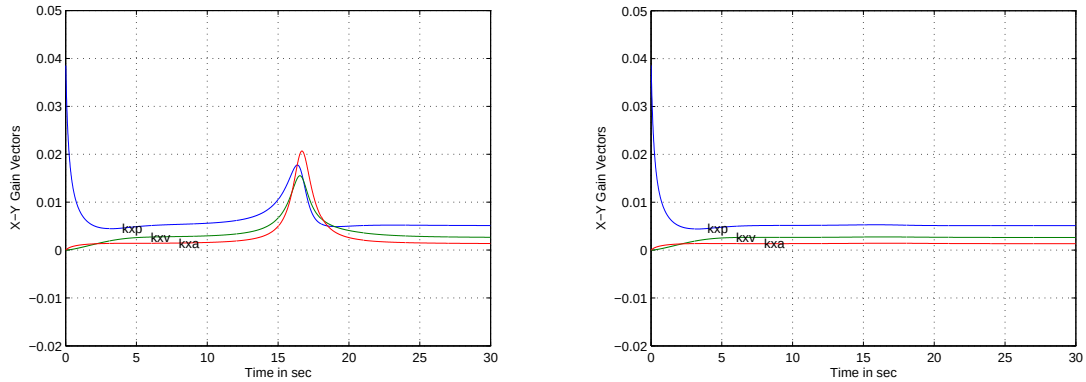
The matrices $Q, Phi, P(0)$ can be found from Singer.

3 Results

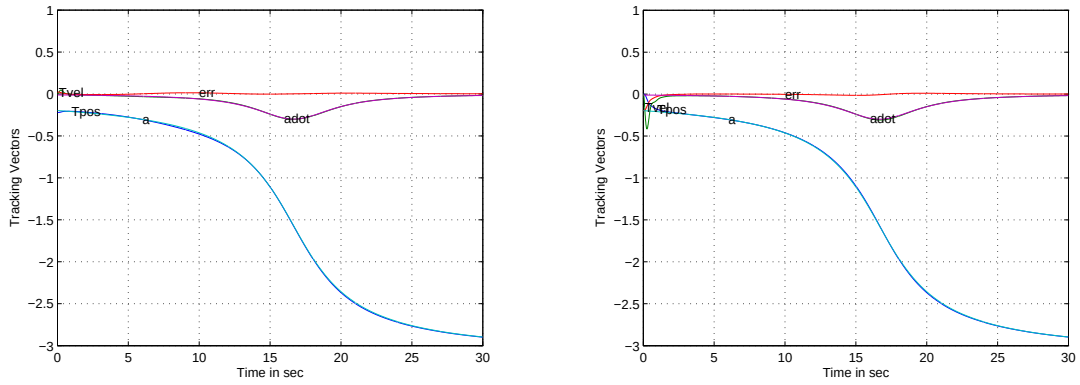
Results, using a discrete implementation with sampling period $T = 0.01$ is shown below (left for stationary case and right for rotating case)



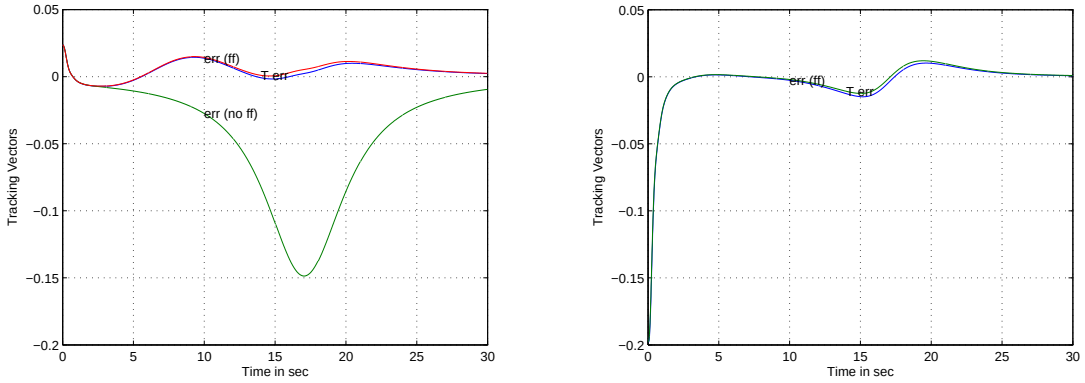
(a) Position (g = geometry, m = measured, f = filtered)



(b) Gain



(c) Angle Vectors



(d) Tracking Error Vectors

The results look fairly similar, the settling of the filter influencing the tracking error somewhat. The peak error before feedforward has been rduced by a factor of 10×. The STD of the error for the target downrange of the three cases are:

Configuration 4000–1000 m	1σ	Units	Notes
No feedforward	14.0	mrad	
Feedforward from Stationary KF	1.7	mrad	
Feedforward from Rotary KF	2.0	mrad	Simple rotation only!

A Stationary Filter

The heart of the Matlab file for the tracker with stationary Kalman Filter is listed here.

```
clear

T = 0.01;

alpha = 2;
sp  = 50;
sa  = 0.001; se = sa; sr  = 5;

M = 3000;
U = -300;
x0 = 5000;
y0 = -1000;
z0 = -300;

Psi = [1, T, (-1 + alpha*T + exp(-alpha*T))/alpha^2
       0, 1, (1 - exp(-alpha*T))/alpha
       0, 0, exp(-alpha*T)];
H   = [1,0,0];

Q = 1 * alpha * [T^5/20, T^4/8, T^3/6
                 T^4/8,  T^3/3, T^2/2
                 T^3/6,  T^2/2, T] * sp^2;

% Plant G(s) = 157.9/(s^2 + 12.57 s + 157.9) X 1/s
% Dig equivalent G(z) =
%       0.007565 z + 0.007255       0.01 z
%      ---------------------- X   --------
%      z^2 - 1.867 z + 0.8819       z - 1

numz = [0, 0.007565, 0.007255];
denz = [1, -1.867,   0.8819];
Ks = 2;

xc = [5000, -300, 0]';
yc = [-1000, 0, 0]';
zc = [-300, 0, 0]';
xp = xc; yp = yc; zp = zc;
Pxp = 100*[sr^2, sr^2/T,       0
       sr^2/T, 2*sr^2/T^2, 0
       0,      0,          0];
% Alternatief - maak onsekerheid hoog in eerste stap
Pxp = [1 0 0; 0 5 0; 0 0 20];
```

```
Pyp = Pxp;
Pzp = Pxp;

ut1 = 0; ut2 = 0;
ot1 = 0; ot2 = 0;
otp = atan2(yc(1),xc(1));
utf1 = 0; utf2 = 0;
otf1 = 0; otf2 = 0;
otfp = otp;

for i = 1:M
    ti(i) = i * T;

    % Geometry equations

    x = x0 + U * i * T;
    y = y0;
    z = z0;

    r = sqrt(x^2 + y^2 + z^2);
    a = atan2(y,x);
    e = asin(-z/r);
    ad = -U*sin(a).*sin(a)./y;
%    ed = ;

    % Measurements

    xm = r * cos(a) * cos(e);
    ym = r * sin(a) * cos(e);
    zm = r * sin(e);

    Rx = (sr * cos(a) * cos(e))^2 + (sa * r * sin(a))^2;
    Ry = (sr * sin(a) * cos(e))^2 + (sa * r * cos(a))^2;
    Rz = (se * sin(e))^2 + (sa * r * cos(e))^2;

    % Update Gain

    Pxzz = H * Pxp * H' + Rx;
    Pyzz = H * Pyp * H' + Ry;
    Pzzz = H * Pzp * H' + Rz;

    Kx = Pxp * H' / Pxzz;
    Ky = Pyp * H' / Pyzz;
    Kz = Pzp * H' / Pzzz;

    Pxc = (eye(3,3) - Kx * H) * Pxp;
```

```
   Pyc = (eye(3,3) - Ky * H) * Pyp;
   Pzc = (eye(3,3) - Kz * H) * Pzp;

%   Kx = [0.02 0.0199 0.0099]';
%   Ky = [0.02 0.0199 0.0099]';
   Kz = [0 0 0]';

   % Get innovation

   ix = xm - H * xp;
   iy = ym - H * yp;
   iz = zm - H * zp;

   xd = Kx * ix;
   yd = Ky * iy;
   zd = Kz * iz;

   xc = xp + xd;
   yc = yp + yd;
   zc = zp + zd;

   % Predictor

   xp = Psi * xc;
   yp = Psi * yc;
   zp = Psi * zc;

   Pxp = Psi * Pxc * Psi + Q;
   Pyp = Psi * Pyc * Psi + Q;
   Pzp = Psi * Pzc * Psi + Q;

   % Calculate feedforward rates

   rsl  = xp(1)^2 + yp(1)^2;
   adf  = -xp(2) * yp(1) / rsl;
   addf = 2 * xp(2)^2 * xp(1) * xp(1) / rsl / rsl;

   % Tracker with feedforward

   taferr = a - otfp;
   utf = Ks * (taferr + 0.5*adf - addf/4.76);
   otf = numz(2) * utf1 + numz(3) * utf2 ...
       - denz(2) * otf1 - denz(3) * otf2;
   otf2 = otf1; utf2 = utf1;
   otf1 = otf;  utf1 = utf;
```

```
    otfp = otfp + 0.01 * otf;

    % Tracker without feedforward

    taerr = a - otp;
    ut = Ks * (taerr);
    ot = numz(2) * ut1 + numz(3) * ut2 ...
        - denz(2) * ot1 - denz(3) * ot2;
    ot2 = ot1; ut2 = ut1;
    ot1 = ot;  ut1 = ut;

    otp = otp + 0.01 * ot;

    % Vectors

    gVec(i,:)  = [x,y,z];
    ...

end
```

B Rotating Filter

Only the differences from the stationary filter are highlighted.

```
...
xc = [5000, -300, 0]';
yc = [0, -100, 0]';
zc = [300, 0, 0]';
xp = xc; yp = yc; zp = zc;
Pxp = 1*[sr^2, sr^2/T,       0
       sr^2/T, 2*sr^2/T^2, 0
       0,      0,          0];
% Alternatief - maak onsekerheid hoog in eerste stap
Pxp = 1*[1 0 0; 0 5 0; 0 0 0];
Pyp = 1*[0 0 0; 0 5 0; 0 0 10];
Pzp = zeros(3,3);
...
    % Measurements

    taferr = a - otfp;
    xm = r * cos(taferr) * cos(e);
    ym = r * sin(taferr) * cos(e);
    zm = r * sin(e);

    Rx = (sr * cos(e))^2;
    Ry = (sa * r * cos(e))^2;
    Rz = (sa * r * sin(e))^2;
    ...
    % Predictor

    xp = Psi * xc;
    yp = Psi * yc;
    zp = Psi * zc;

    Pxp = Psi * Pxc * Psi + Q;
    Pyp = Psi * Pyc * Psi + Q;
    Pzp = Psi * Pzc * Psi + Q;

    xp = xp + yp * ad * T;
    yp = yp - xp * ad * T;

    % Calculate feedforward rates

    adf  = yp(2) / xp(1);
    addf = yp(3) / xp(1) - 2 * xp(2) * yp(2) / xp(1)^2;
    ...
```