Universität Stuttgart

Institute of Geodesy

Kalman Filtering
Implementation with Matlab

Study Report in the Field of Study

Geodesy and Geoinformatics

at Universität Stuttgart

Rachel Kleinbauer

Helsinki, November 2004

Adviser:

Prof. Dr.-Ing.habil. Dr.tech.h.c.mult Dr.-Ing.E.h.mult Erik W. Grafarend
Universität Stuttgart

Prof. Martin Vermeer
Helsinki University of Technology

Abstract

In 1960 and 1961 Rudolf Emil Kalman published his papers on a recursive predictive filter that is based on the use of state space techniques and recursive algorithms and therewith he revolutionized the field of estimation. Since that time the so-called Kalman filter has been the subject of extensive research and application.

The Kalman filter estimates the state of a dynamic system, even if the precise form of the system is unknown. The filter is very powerful in the sense that it supports estimations of past, present, and even future states.

Within the scope of this study thesis it was the task to program a Kalman filter in Matlab. The intention is to give the students of the course “Methods of Navigation” an understanding of the Kalman filter by providing them with its practical aspects.

The composition includes a description of the standard Kalman filter and its algorithm with the two main steps, the prediction step and the correction step. Furthermore the extended Kalman filter is discussed, which represents the conversion of the Kalman filter to nonlinear systems. In the end the program was executed to calculate the orbit of a geostationary satellite as an example.

Keywords: filter, Kalman gain matrix, prediction, dynamic model, state vector

Schlüsselwörter: Filter, Kalman Korrekturmatrix, Vorhersage, dynamisches Modell, Zustandsvekor

Contents

Chapter 1

Introduction

1.1 Motivation

Navigation systems become standard, as they are sold more and more, particularly with new cars. Most of these navigation systems use no longer only the Global Positioning System (GPS) but also an inertial navigation system (INS) to help the driver find his way. Together the two systems complement each other and permit improved navigation accuracy and reliability especially when GPS is degraded or interrupted for example because of buildings or tunnels.

And for this application *the Kalman filter* provides the basis. It constitutes a tool for correcting the predicted INS trajectory with GPS measurements.

Also the determination of a reference orbit for these GPS satellites and correcting it with the data from the GPS control stations is a very important application of the Kalman filter.

But these are only two examples of the wide variety of fields where Kalman filtering plays an important role. The application areas span from aerospace, to marine navigation, demographic modelling, weather science, manufacturing and many others. Because the Kalman filter is very effective and useful for such a large class of problems, it has been subject of extensive research.

Within the scope of this study thesis I programmed a Kalman filter in Matlab that is meant to give the students an understanding of the Kalman filter by providing them with its practical aspects. This programme will be used in the course „Methods of Navigation“ that the students can discover how the Kalman filter works by observing it in action.

1.2 Preview

At the beginning, in chapter two the basic information is provided. The question “What is a Kalman filter” is answered and the basic components of the Kalman filter are explained.

The third chapter is about the formulas of the standard Kalman filter, which is a linear filter. It describes the two main steps of the Kalman filter.

The fourth chapter shows how these formulas are transferred to nonlinear systems, which leads to the so-called Extended Kalman filter.

In chapter five the essential formulas of both the standard Kalman filter and the Extended Kalman filter are summarized in a table.

Chapter six describes the implementation of the Kalman filter in Matlab with some illustrative sections of the Matlab source code.

The programmed Kalman filter is applied in chapter 7 to the example of a geostationary orbit.

And finally chapter 8 represents the closing with conclusions and prospects.

Chapter 2

General Information

2.1 Kalman Filter

Rudolf Emil Kalman was born in Budapest, Hungary, on May 19, 1930. He had the idea of the Kalman filter for the first time in in the year 1958. In 1960 and 1961 he published his papers on the Kalman filter and therewith he revolutionized the field of estimation.

The Kalman filter is a recursive predictive filter that is based on the use of state space techniques and recursive algorithms. It estimates the state of a dynamic system. This dynamic system can be disturbed by some noise, mostly assumed as white noise. To improve the estimated state the Kalman filter uses measurements that are related to the state but disturbed as well.

Thus the Kalman filter consists of two steps:

1. the prediction
2. the correction

In the first step the state is predicted with the *dynamic model*. In the second step it is corrected with the *observation model*, so that the error covariance of the estimator is minimized. In this sense it is an optimal estimator.

This procedure is repeated for each time step with the state of the previous time step as initial value. Therefore the Kalman filter is called a recursive filter.

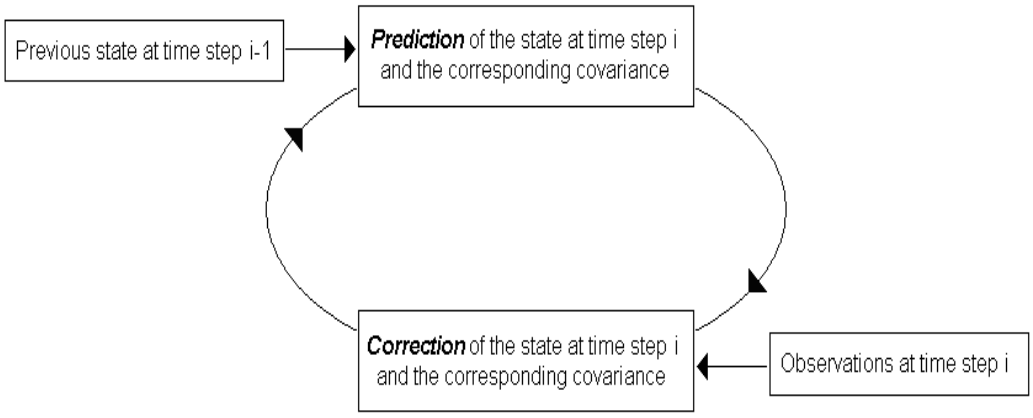

Figure 1: circuit of the Kalman filter

The basic components of the Kalman filter are the state vector, the dynamic model and the observation model, which are described below.

2.2 State Vector

The state vector contains the variables of interest. It describes the state of the dynamic system and represents its degrees of freedom. The variables in the state vector cannot be measured directly but they can be inferred from values that are measurable.

Elements of the state vector can be e.g. position, velocity, orientation angles, etc. A very simple example is a train that is driving with a constant velocity on a straight rail. In this case the train has two degrees of freedom, the distance s and the velocity $\dot{s} = v$. Thus the state vector is

$$\underline{x} = \begin{bmatrix} s \\ v \end{bmatrix}$$

The state vector has two values at the same time, that is the *a priori* value, the predicted value before the update, and the *a posteriori* value, the corrected value after the update. In the following the a priori value is marked by $\underline{x}^-$ and the a posteriori value by $\underline{x}^+$.

2.3 Dynamic Model

The dynamic model describes the transformation of the state vector over time. It can usually be represented by a system of differential equations.

$$\dot{\underline{x}}(t) = \frac{d}{dt}\underline{x}(t) = f(\underline{x}(t), \underline{m}(t)) \tag{2.1}$$

In the linear case this can easily be rewritten as

$$\dot{\underline{x}}(t) = F \cdot \underline{x}(t) + \underline{n}(t) \tag{2.2}$$

where F is the dynamic matrix and is constant, $\underline{x}(t)$ is the state vector and $\underline{n}(t)$ is the dynamic noise which is usually assumed as white noise and has the covariance matrix $Q(t)$.

There are systems that cannot be modelled by differential equations, but these systems are not discussed in this thesis.

2.4 Observation Model

The observation model represents the relationship between the state and the measurements. In the linear case the measurements can be described by a system of linear equations, which depend on the state variables. Usually the observations are made at discrete time steps t_i

$$\underline{l}(t_i) = h(\underline{x}(t_i), \underline{v}(t_i)) \tag{2.3}$$

The vector form of this system is

$$\underline{l}(t_i) = H \cdot \underline{x}(t_i) + \underline{w}(t_i) \tag{2.4}$$

where $\underline{l}(t_i)$ is the vector of the observations at the epoch t_i, H is the observation matrix and $\underline{w}(t_i)$ is the noise of the measurement process with the covariance matrix $R(t_i)$. Like the dynamic matrix, in a linear system the observation matrix H is a constant matrix as well.

Chapter 3

Kalman Filter Algorithm

3.1 Prediction

Like mentioned before, the prediction is the first step of the Kalman filter. The predicted state, or better the a priori state is calculated by neglecting the dynamic noise and solving the differential equations that describe the dynamic model

$$\dot{\underline{x}}^{-}(t) = F \cdot \underline{x}^{-}(t) \tag{3.1}$$

The state vector at time t can be expressed by a Taylor series with respect to an approximate state $\underline{x}^{-}(t_0)$.

$$\underline{x}^{-}(t) = \underline{x}^{-}(t_0) + \dot{\underline{x}}^{-}(t_0)(t-t_0) + \tfrac{1}{2}\ddot{\underline{x}}^{-}(t_0)(t-t_0)^2 + \ldots$$

by using equation (3.1) this can be rewritten as

$$\underline{x}^{-}(t) = \underline{x}^{-}(t_0) + F \cdot \underline{x}^{-}(t_0)(t-t_0) + \tfrac{1}{2}F^2\underline{x}^{-}(t_0)(t-t_0)^2 + \ldots$$

Thus the solution $\underline{x}^{-}(t)$ of the differential equations, in other words the actual predicted state is a linear combination of the initial state $\underline{x}^{-}(t_0)$

$$\underline{x}^{-}(t) = \Phi_0^t \cdot \underline{x}^{-}(t_0) \tag{3.2}$$

Φ_0^t is called the *state transition matrix*, which transforms any initial state $\underline{x}(t_0)$ to its corresponding state $\underline{x}(t)$ at time t.

from the equations (3.1) and (3.2):

$$\dot{\underline{x}}^{-}(t) = F \cdot \underline{x}^{-}(t) = F \cdot \Phi_0^t \cdot \underline{x}^{-}(t_0) \tag{3.3}$$

and by using (3.2) again, one can see

$$\dot{\underline{x}}^{-}(t) = \frac{d}{dt}\underline{x}^{-}(t) = \frac{d}{dt}\left[\Phi_0^t \cdot \underline{x}^{-}(t_0)\right] = \left[\frac{d}{dt}\Phi_0^t\right] \cdot \underline{x}^{-}(t_0) \tag{3.4}$$

By comparing (3.3) and (3.4) it follows that:

$$\frac{d}{dt}\Phi_0^t = F \cdot \Phi_0^t \tag{3.5}$$

with the initial matrix $\Phi_0^0 = \mathrm{I}$, because $\underline{x}(t_0) = \mathrm{I} \cdot \underline{x}(t_0)$

And now the covariance matrix $P^{-}(t_i)$ of the predicted state vector is obtained with the law of error propagation

$$P^{-}(t_i) = \Phi_{t_{i-1}}^{t_i} \cdot P(t_{i-1}) \cdot (\Phi_{t_{i-1}}^{t_i})^T + Q$$

In the more generalized form, where also the covariance matrix of the noise Q is a function of time, the covariance matrix is

$$P^{-}(t_i) = \Phi_{t_{i-1}}^{t_i} \cdot P(t_{i-1}) \cdot (\Phi_{t_{i-1}}^{t_i})^T + \int_{ti-1}^{ti} Q(t)dt \tag{3.6}$$

3.2 Correction

In the correction step the predicted state vector $\underline{x}^-(t_i)$ is improved with observations made at the epoch t_i, thus the a posteriori state has the form

$$\underline{x}^+(t_i) = \underline{x}^-(t_i) + \Delta\underline{x}(t_i)$$

with the covariance matrix

$$P^+(t_i) = P^-(t_i) + \Delta P(t_i)$$

As said before the Kalman filter is an optimal filter, this means that the state variances in the state covariance matrix P^+ are minimized. As P^- is already known from the prediction step it follows that ΔP is minimized.

$$\Delta P(t_i) = E\left[\Delta\underline{x}(t_i)\Delta\underline{x}(t_i)^T\right]$$

this condition is complied with

$$\Delta\underline{x}(t_i) = P^- H^T \left(HP^- H^T + R(t_i)\right)^{-1} \cdot \left(\underline{l}(t_i) - H\,\underline{x}^-(t_i)\right)$$

$$\Rightarrow \qquad \Delta\underline{x}(t_i) = K(t_i) \cdot \left(\underline{l}(t_i) - \underline{l}^-(t_i)\right)$$

with

$$K(t) = P^- H^T \left(HP^- H^T + R(t_i)\right)^{-1}$$

K is called the *gain matrix.* The difference $(\underline{l}(t_i) - \underline{l}^-(t_i))$ is called the *measurement residual*. It reflects the discrepancy between the predicted measurement $\underline{l}^-(t_i) = H\underline{x}^-(t_i)$ and the actual measurement $\underline{l}(t_i)$.

Finally the corrected state is obtained by

$$\underline{x}^+(t_i) = \underline{x}^-(t_i) + K(t_i) \cdot \left(\underline{l}(t_i) - \underline{l}^-(t_i)\right)$$

In this equation the estimated state and the measurements are weighted and combined to calculate the corrected state. That means, if the measurement covariance is much smaller than that of the predicted state, the measurement's weight will be high and the predicted state's will be low. And so the uncertainty can be reduced.

The covariance matrix of the a posteriori state is given with the law of error propagation by

$$\begin{aligned} P^{+}(t_i) &= P^{-}(t_i) - K(t_i)HP^{-}(t_i) \\ &= \left(I - K(t_i)H\right)P^{-}(t_i) \end{aligned}$$

Chapter 4

Extended Kalman Filter

4.1 Nonlinear Systems

So far only linear systems have been considered. But in practice the dynamic or the observation model can be nonlinear.

One approach to the Kalman filter for such nonlinear problems is the so-called Extended Kalman filter, which was discovered by Stanley F. Schmidt. After Kalman presented his results about Kalman filtering, Schmidt immediately began applying it to the space navigation problem for the upcoming Apollo project for manned exploration of the moon. In this process, he invented the extended Kalman filter.

This Kalman filter linearizes about the current estimated state. Thus the system must be represented by continuously differentiable functions.

One disadvantage of this version of the Kalman filter for nonlinear systems is that it needs more time-consuming calculations. The implementation for linear systems can be made more efficient by pre-computing the dynamic matrix F, the state transition matrix Φ and the observation matrix H. But for nonlinear systems, these are functions of the state and consequently change with every time step and cannot be pre-computed.

4.2 Prediction

In the nonlinear case the dynamic matrix F is a function of the state to be estimated. So the predicted state is calculated by solving the differential equations in the form

$$\dot{\underline{x}}^{-}(t) = f(\underline{x}^{-}(t))$$

By representing this equation by a Taylor series with respect to x at the predicted state $\underline{x}^{-}(t_i)$ and assuming that the higher order terms can be neglected, the dynamic matrix $F(t_i)$ can be calculated with

$$F(t_i) = \left.\frac{\partial f(\underline{x})}{\partial \underline{x}}\right|_{\underline{x}=\underline{x}^{-}(ti)}$$

And now the other steps of the prediction can be calculated as shown in chapter 3.1 with the equations (3.5) and (3.6), but it should be noted that, now the used matrices are not constant like in the linear case, but depend on the time step.

$$\frac{d}{dt}\Phi_{ti-1}^{ti} = F(t_i)\cdot\Phi_{ti-1}^{ti}$$

$$P^{-}(t_i) = \Phi_{ti-1}^{ti}\cdot P(t_{i-1})\cdot(\Phi_{ti-1}^{ti})^{T} + \int_{ti-1}^{ti} Q(t)dt$$

4.3 Correction

Like the differential equations in the prediction step the corresponding nonlinear observation equations are linearized with the Taylor series about the predicted state $\underline{x}^{-}(t_i)$ and higher order terms are neglected.
Thus the approximate observation matrix is

$$H(t_i) = \left.\frac{\partial h(\underline{x})}{\partial \underline{x}}\right|_{\underline{x}=\underline{x}^{-}(ti)}$$

In that case the predicted measurement $\underline{l}^-(t_i)$ for calculating the measurement residual $(\underline{l}(t_i) - \underline{l}^-(t_i))$ is

$$\underline{l}^-(t_i) = h(\underline{x}^-(t_i))$$

Further on, we can again use the same formulas to calculate the corrected state and its covariance matrix like in the linear case but with time dependent matrices.

$$x^+(t_i) = x^-(t_i) + K\left(\underline{l}(t_i) - \underline{l}^-(t_i)\right)$$

and

$$P^+(t_i) = \left(I - K(t_i)H(t_i)\right)P^-(t_i)$$

with

$$K(t_i) = P^-H(t_i)^T\left(H(t_i)P^-H(t_i)^T + R(t_i)\right)^{-1}$$

Chapter 5

Summary

The models and the essential implementation equations of the linear and the extended Kalman filter that were derived in section 3 and 4 are summarized below.

	Standard Kalman Filter	***Extended Kalman Filter***
	Dynamic model:	
	$\dot{\underline{x}}(t) = F \cdot \underline{x}(t) + \underline{n}(t)$	$\dot{\underline{x}}(t) = f(\underline{x}(t), \underline{m}(t))$
	Observation model:	
	$\underline{l}(t) = H \cdot \underline{x}(t) + \underline{w}(t)$	$\underline{l}(t) = h(\underline{x}(t), \underline{v}(t))$
Prediction	A priori state:	
	$\dot{\underline{x}}^{-}(t) = F \cdot \underline{x}^{-}(t)$	$\dot{\underline{x}}^{-}(t) = f(\underline{x}^{-}(t))$
	State transition matrix:	
	$\frac{d}{dt}\Phi_{t_{i-1}}^{t_i} = F \cdot \Phi_{t_{i-1}}^{t_i}$	$\frac{d}{dt}\Phi_{t_{i-1}}^{t_i} = F(t_i) \cdot \Phi_{t_{i-1}}^{t_i},\ F(t_i) = \left.\frac{\partial f(\underline{x})}{\partial \underline{x}}\right\|_{\underline{x}=\underline{x}^{-}(ti)}$
	A priori covariance matrix:	
	$P^{-}(t_i) = \Phi_{t_{i-1}}^{t_i} \cdot P^{+}(t_{i-1}) \cdot (\Phi_{t_{i-1}}^{t_i})^T + \int_{t_{i-1}}^{t_i} Q(t)dt$	

Correction

Gain matrix:

$$K(t_i) = P^-(t_i)H^T\left(HP^-(t_i)H^T + R(t_i)\right)^{-1} \quad K(t_i) = P^-(t_i)H(t_i)^T\left(H(t_i)P^-(t_i)H(t_i)^T + R(t_i)\right)^{-1}$$

$$H(t_i) = \left.\frac{\partial h(x)}{\partial x}\right|_{x=\underline{x}^-(ti)}$$

Predicted measurement:

$$\underline{l}^-(t_i) = H\,\underline{x}^-(t_i) \qquad \underline{l}^-(t_i) = h(\underline{x}^-(t_i))$$

A posteriori state:

$$\underline{x}^+(t_i) = \underline{x}^-(t_i) + K\left(\underline{l}(t_i) - \underline{l}^-(t_i)\right)$$

A posteriori covariance matrix:

$$P^+(t_i) = \left(I - K(t_i)H\right)P^-(t_i) \qquad P^+(t_i) = \left(I - K(t_i)H(t_i)\right)P^-(t_i)$$

One has to note that the parameters from the correction step are only defined at the observation times t_i, whereas the parameters from the prediction step exist continuously for all t. To simplify matters, except of the state, they are calculated only at the observation times t_i as well. Of course also the observations can be continuous, but in practice this is rather seldom.

Chapter 6

Implementation

Within the scope of this study thesis a program in Matlab was written. Matlab-version 6.5 Release 13 was used.

In this chapter the implementation is described with some illustrative sections of the Matlab source code.

One condition was that this Kalman filter should be general. That is to say that it should be possible for the user to determine the models. As this condition was more important than the computing time, the algorithm of the extended Kalman filter is used, because by using linear equations, it becomes just the standard Kalman filter since

$$F = \frac{\partial f(\underline{x})}{\partial \underline{x}} \quad \text{and} \quad H = \frac{\partial h(\underline{x})}{\partial \underline{x}}$$

whereas F and H do not depend on the state in the linear case.

In the following source code sections the a priori state and its covariance matrix are marked by an ‘m’ for minus and the corresponding a posteriori parameters are marked by a ‘p’ for plus. ‘i’ specifies the time step.

6.1 Extraction of the Data

The first step of the program is the reading of the required data from text files that the user has to prepare before using the program. How the user has to format the text files is shown in the folder ‘format’.

In this way the following data is read in:

state vector *	x_sym
initial value of the state vector	x0
its covariance matrix	P0
differential equations of the dynamic model *	f
covariance matrix of the dynamic noise *	Q
equations of the observation model *	h
observations	l
their covariance matrices	R
the time steps of the observations	t

Due to their dependency on the variables of the state vector, much of this data is symbolic expressions. The concerned data is marked by *.

Furthermore, the user can choose between ‘no plot’, ‘2D plot’ or ‘3D plot’ and define the axes.

6.2 Prediction

6.2.1 A Priori State

As described in 4.2, the continuous state from the time span t_{i-1} to t is obtained by solving the differential equation. In Matlab this can be realized with the ordinary differential equation solvers (ode).

In general, ode45, which is based on an explicit Runge-Kutta formula, is the best function to apply for most problems. It is a one-step solver and for computing $\underline{x}(t_i)$, it needs only the solution at the preceding time step $\underline{x}(t_{i-1})$.

a priori state:

```
% solving the differential equation f to get the a priori
state
% vector xm:
tspan = [ti(i-1) ti(i)];
[T1 X] = ode45(@diff_eq,tspan,x,opt,x_sym,f);
% Each row in the solution array X corresponds to a time
returned in the column vector T1

k1 = length(T1);
xm = (X(k1,:))'
```

Source code section 1

with ode45 the system of differential equations dxdt = f(x(t)) is integrated over `tspan` from time `t(i-1)` to `t(i)` with initial conditions `x`, which is the previous state vector $\underline{x}^{+}(t_{i-1})$. Each row in the solution array `X` corresponds to a time returned in the column vector `T1`. Thus the last row of this solution array is the a priori state `xm` of the time step `t(i)` that is $\underline{x}^{-}(t_i)$.

The system of differential equations must be coded as a function that the ode solver can use. In this case the function is called diff_eq. This function returns a column vector corresponding to f(x(t)).

function for the differential equation:

```
function dxdt = diff_eq(t,x,x_sym,f)

% the differential equations are calculated with the values
x:
f = subs(f,x_sym,sym(x));
f = sym(f);

dxdt = eval(f);
```

Source code section 2

The first two arguments of the function must be `t` and `x`, even if one of them is not used.

6.2.2 State Transition Matrix

To solve the differential equation of the state transition matrix the ode45 is used again. But there is the problem that ode45 only works in the case of vectors and not in the case of matrices. So we must split the state transition matrix in vectors i.e. the differential equation is solved column by column. This is permissible because matrix multiplication in general can be done column by column. For example each column of C, which is the solution of the matrix product

$$C = A \cdot B$$

can be calculated

$$c_j = A \cdot b_j$$

where j is the column number.

In this context, we can calculate each column of the wanted state transition matrix by solving the differential equation in the form

$$\left[\frac{d\Phi}{dt}\right]_j = F \cdot \Phi_j$$

where j again is the number of the column.

With this the ode solver is used like above. The differential equation is solved for each column, after which the last row of the solution array `Phi_trans` is the corresponding column of the wanted state transition matrix $\Phi(t_i)$.

But before solving the differential equations for the state transition matrix, the dynamic matrix F needs to be evaluated. In the nonlinear case F depends on $\underline{x}(t_i)$ and has to be calculated at the predicted state $\underline{x}^-(t_i)$ like mentioned in chapter 4.2.

state transition matrix

```
% in the case the dynamic model is nonlinear F needs to be
calculated
% with xm
F = subs(F_dyn,x_sym,sym(xm));
F = sym(F); F = eval(F)

for j=1:n_s
    [T2 Phi_trans] = ode45(@diff_eq_P,tspan,Phi(:,j),opt,F);
                                k2 = length(T2);
                                Phi(:,j) = (Phi_trans(k2,:))';
end
```

Source code section 3

6.2.3 Covariance of the A Priori State

The covariance Q of the dynamic noise may depend on the state variables as well, so it must be calculated at the predicted state $\underline{x}^{-}(t_i)$, before the a priori Covariance matrix can be computed just like shown in 4.2.

Covariance matrix of the a priori state

```
Q = subs(Q_sym,x_sym,sym(xm));
Q = sym(Q); Q = eval(Q);
Pm = sym(Phi*P*Phi' + int(Q,t,ti(i-1),ti(i)));
Pm = eval(Pm)
```

Source code section 4

6.3 Correction

6.3.1 Gain Matrix

Before the gain matrix can be computed, the observation matrix must be determined. In the linear case this is just the Jacobian matrix of the observation equations h, but in the nonlinear case it still depends on the state variables. H could simply be calculated at the a priori state like shown in 4.3. But in some cases it might enhance the results to iterate the observation matrix. Within the following iteration the state, at which the observation matrix H is evaluated, is quickly improved with a simple least squares adjustment that minimizes the improvements of the observations. So the H matrix that is calculated at this improved state is closer to the *real* observations.

Iteration of the observation matrix in function ObsMatrix

```
while max(abs(dx)) > 0.00001
    k=k+1;
    % calculation of H and l0 with the (improved) initial
values:
    H = subs(J,x_sym,sym(x));
    H = sym(H); H = eval(H);

    l0 = subs(h,x_sym,sym(x));
    l0 = sym(l0); l0 =eval(l0);

    dl = l-l0;
    dx = pinv(H'*Rinv*H)*H'*Rinv*dl;

    % improving the state vector
    x = x+dx;

    if k == 20
        dx=[0.000001];
        Iteration = 0;
        sprintf('Iteration in "ObsMatrix.m" aborted!\n
observation matrix H will be calculated at the approximation
state xm\n')
    end
end
```

Source code section 5

One must note, that the standard least squares adjustment used in the iteration is not the Kalman correction formula. As shown in chapter 3.2, with this formula the predicted state and the measurements are weighted and combined to calculate the corrected state. But in this iteration the intention is to calculate the observation matrix H so that it is closer to the observations and less affected by the approximate state. And this is realized in the iteration by minimizing the improvements of the observations.

If the observation matrix H doesn't converge within this iteration, it is just computed at the a priori state $\underline{x}^{-}(t_i)$.

But this iteration can pose a new problem. Namely, when not all elements of the state vector are in the observation equations, the Jacobian matrix of h respecting the state x has columns that are zero i.e. the matrix has no full rank and therewith the equation system $dx = (H' R_{inv} H)^{-1} H R_{inv} dl$ from the standard least squares adjustment in the iteration has no finite solution. Hence instead of the inverse the Moore-Penrose pseudoinverse is used to solve the equation system.

And then the gain matrix is computed with that observation matrix H, which emanates from the function ObsMatrix.

Gain matrix

```
H = ObsMatrix(l(:,i),R(:,:,i),h,x_sym,xm)
K = Pm*H'*(H*Pm*H'+R(:,:,i))^(-1);
```

Source code section 6

6.3.2 A Posteriori State and Covariance

And finally the predicted measurement and with it the a posteriori state and its covariance matrix can be computed with the corresponding formulas.

a posteriori state and its Covariance matrix

```
% predicted measurement:
lm = subs(h,x_sym,sym(xm));
lm = sym(lm); lm = eval(lm);

% measurement residual:
y = l(:,i)-lm       % each column of l corresponds to a time

% a posteriori state vector and the corresponding covariance
matrix:
xp(:,i) = xm + K*y; x = xp(:,i)
P = Pm - K*H*Pm
```

Source code section 7

And for the next time step `xp` will be the initial state `x`.

Chapter 7

Example

As an example the program was executed to calculate the orbit of a geostationary satellite. These geostationary orbits are used for most commercial communication satellites, for TV-transmissions and for meteorological satellites like METEOSAT. Geostationary orbits can only be achieved very close to the ring of 35 786 km above the equator with an orbital speed around 11 068 km/h.

7.1 Dynamic Model and Initial Values

The motion of an object in the earth's gravitational field can be described by the following differential equations:

$$\frac{d^2}{dt^2}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = -\frac{GM}{\left(\sqrt{x^2 + y^2 + z^2}\right)^3}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

where the noise elements n_x, n_y and n_z represent for example the irregularities of the Earth's gravitational field or the influence of the solar pressure.

As these are second-order differential equations and the Kalman filter needs first-order differential equations, the model must be changed so that it fulfils the requirements. This can be realized by transforming one second-order differential equation to a system of two first-order differential equations. For this purpose the state vector $\underline{x}$ must be extended with its first derivation $\underline{\dot{x}}$.

In this example the additional components are

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

This causes the following equation system

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} - \frac{GM}{\left(\sqrt{x^2 + y^2 + z^2}\right)^3}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ n_x \\ n_y \\ n_z \end{bmatrix}$$

In this way, one can reduce the form of any system of higher order differential equations to an equivalent system of first-order differential equations.

This system without the noise vector is now used to calculate the a priori state where G is the gravitational constant and M is the mass of the earth:

$$G = 6{,}6742 \cdot 10^{-11} \frac{m^3}{kg \cdot s^2}$$
$$M = 5{,}9736 \cdot 10^{24}\, kg$$

As the computation of the actual a priori state needs the previous state vector as initial value, one must assume values for the elements of the state vector at the start time t_0.

For a geostationary satellite one may say that the z-coordinate is around zero and that it stays in this range. Therewith the velocity in z-direction v_z is around zero as well. When we now assume that the satellite is situated on the x-axis at the start of its observation, also the y-coordinate is zero and the x-coordinate is the distance from the origin, the centre of the earth. And with the mean earth's radius of 6370 km and the height of the satellite above equator around 36000 km the distance between origin and satellite is around 42370 km. At this point the satellite moves only in y-direction, therefore v_y is around 11068 km/h and v_x is near zero.

So the initial state is

$$\underline{x}_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ v_{x0} \\ v_{y0} \\ v_{z0} \end{bmatrix} = \begin{bmatrix} 42370\,km \\ 0 \\ 0 \\ 0 \\ 11068\,km/h \\ 0 \end{bmatrix}$$

Let's say that this vector is known with the covariance matrix

$$P_0 = \begin{bmatrix} 50\,km^2 & & & & & \\ & 50\,km^2 & & & 0 & \\ & & 50\,km^2 & & & \\ & & & 1\frac{km^2}{h^2} & & \\ & 0 & & & 1\frac{km^2}{h^2} & \\ & & & & & 1\frac{km^2}{h^2} \end{bmatrix}$$

and that the covariance matrix of the system noise is

$$Q = \begin{bmatrix} 0 & & & & & \\ & 0 & & & 0 & \\ & & 0 & & & \\ & & & 0{,}001\frac{km^2}{h^4} & & \\ & 0 & & & 0{,}001\frac{km^2}{h^4} & \\ & & & & & 0{,}001\frac{km^2}{h^4} \end{bmatrix}$$

7.2 Observation Model and Observations

For a geostationary satellite the revolution time is $T = 24\,\text{h}$. Only one revolution is observed. Every three hours the distance from the origin to the satellite is observed.

Thus the observation equation is

$$l = \sqrt{x^2 + y^2 + z^2}$$

Let's say that the distance is constantly 42156 km and its variance is 0,001 km^2 so the observation array is $[42156]_{1x8}$ km with one element of the array corresponds to one time step.

7.3 Results

In the following table the results for one revolution are shown. For every time step the a priori values are faced with the a posteriori values. The units of the coordinates are km and of the velocity km/h.

$t_i = 3\,\text{h}$						
$\underline{x}^-$	P^-					
30072	93.2300	60.3695	0	21.5350	25.9892	0
29926	60.3695	92.6435	0	25.9892	21.2825	0
0	0	0	32.5665	0	0	-4.5807
-7772	21.5350	25.9892	0	8.6114	6.4003	0
7860	25.9892	21.2825	0	6.4003	8.5492	0
0	0	0	-4.5807	0	0	2.1799
$\underline{x}^+$	P^+					
29881	16.2047	-16.2826	0	-2.2791	2.2905	0
29736	-16.2826	16.3629	0	2.2905	-2.3013	0
0	0	0	32.5665	0	0	-4.5807
-7831	-2.2791	2.2905	0	1.2487	-0.9267	0
7801	2.2905	-2.3013	0	-0.9267	1.2577	0
0	0	0	-4.5807	0	0	2.1799

$t_2 = 6\,\mathrm{h}$						
$\underline{x}^-$	P^-					
-78	23.9078	-11.0423	0	2.7856	-4.8808	0
41986	-11.0423	68.4490	0	-1.0336	26.2773	0
0	0	0	31.5209	0	0	4.4499
-11098	2.7856	-1.0336	0	1.1409	-0.9422	0
-80	-4.8808	26.2773	0	-0.9422	10.3770	0
0	0	0	4.4499	0	0	2.2151
$\underline{x}^+$	P^+					
-106	22.1132	0.0411	0	2.6175	-0.6257	0
42156	0.0411	0.0011	0	0.0049	-0.0008	0
0	0	0	31.5209	0	0	4.4499
-11101	2.6175	0.0049	0	1.1252	-0.5435	0
-15	-0.6257	-0.0008	0	-0.5435	0.2885	0
0	0	0	4.4499	0	0	2.2151
$t_3 = 9\,\mathrm{h}$						
$\underline{x}^-$	P^-					
-30046	54.5046	-9.0798	0	4.4684	-12.5291	0
29716	-9.0798	476.5877	0	-107.1142	94.2455	0
0	0	0	14.6957	0	0	-0.0584
-7823	4.4684	-107.1142	0	24.1833	-21.6615	0
-7837	-12.5291	94.2455	0	-21.6615	20.7579	0
0	0	0	-0.0584	0	0	3.4042
$\underline{x}^+$	P^+					
-30029	47.0222	47.5445	0	-8.5441	-0.0665	0
29587	47.5445	48.0746	0	-8.6395	-0.0669	0
0	0	0	14.6957	0	0	-0.0584
-7794	-8.5441	-8.6395	0	1.5532	0.0120	0
-7866	-0.0665	-0.0669	0	0.0120	0.0005	0
0	0	0	-0.0584	0	0	3.4042
$t_4 = 12\,\mathrm{h}$						
$\underline{x}^-$	P^-					
	1.0e+003 *					
-42243						
-350	3.4832	-4.6581	0	1.4861	0.1355	0
0	-4.6581	6.2294	0	-1.9874	-0.1812	0
67	0	0	0.0147	0	0	0.0001
-11050	1.4861	-1.9874	0	0.6341	0.0578	0
0	0.1355	-0.1812	0	0.0578	0.0053	0
	0	0	0.0001	0	0	0.0034
$\underline{x}^+$	P^+					
-42154	0.0010	-0.0023	0	0.0005	0.0001	0
-471	-0.0023	0.1132	0	-0.0114	-0.0103	0
0	0	0	14.7478	0	0	0.0865
105	0.0005	-0.0114	0	0.0016	0.0010	0
-11046	0.0001	-0.0103	0	0.0010	0.0013	0
0	0	0	0.0865	0	0	3.3928

$t_5 = 15\,\mathrm{h}$						
$\underline{x}^-$	P^-					
-29457	1.5022	-0.9029	0	0.1126	0.3016	0
-30149	-0.9029	0.6831	0	-0.0409	-0.1852	0
0	0	0	32.8787	0	0	4.4902
7918	0.1126	-0.0409	0	0.0139	0.0219	0
-7705	0.3016	-0.1852	0	0.0219	0.0610	0
0	0	0	4.4902	0	0	2.1351
$\underline{x}^+$	P^+					
-29469	0.6028	-0.5859	0	0.0025	0.1272	0
-30145	-0.5859	0.5713	0	-0.0021	-0.1237	0
0	0	0	32.8787	0	0	4.4902
7916	0.0025	-0.0021	0	0.0004	0.0005	0
-7708	0.1272	-0.1237	0	0.0005	0.0272	0
0	0	0	4.4902	0	0	2.1351
$t_6 = 18\,\mathrm{h}$						
$\underline{x}^-$	P^-					
590	110.7664	29.6598	0	-8.3071	19.6877	0
-42042	29.6598	8.1447	0	-2.2026	5.3251	0
0	0	0	30.6971	0	0	-4.6430
11076	-8.3071	-2.2026	0	0.6256	-1.4708	0
203	19.6877	5.3251	0	-1.4708	3.5136	0
0	0	0	-4.6430	0	0	2.3326
$\underline{x}^+$	P^+					
168	3.0761	0.0470	0	-0.3140	0.3424	0
-42158	0.0470	0.0017	0	-0.0047	0.0055	0
0	0	0	30.6971	0	0	-4.6430
11107	-0.3140	-0.0047	0	0.0324	-0.0349	0
127	0.3424	0.0055	0	-0.0349	0.0385	0
0	0	0	-4.6430	0	0	2.3326
$t_7 = 21\,\mathrm{h}$						
$\underline{x}^-$	P^-					
30077	460.8544	98.3627	0	-34.4698	-65.8387	0
-29341	98.3627	21.6793	0	-7.4826	-13.9265	0
0	0	0	48.7009	0	0	-0.5355
7778	-34.4698	-7.4826	0	2.6015	4.9014	0
7981	-65.8387	-13.9265	0	4.9014	9.4292	0
0	0	0	-0.5355	0	0	1.0338
$\underline{x}^+$	P^+					
30320	1.0409	1.0645	0	-0.2322	0.0059	0
-29289	1.0645	1.0906	0	-0.2378	0.0064	0
0	0	0	48.7009	0	0	-0.5355
7760	-0.2322	-0.2378	0	0.0522	-0.0014	0
7947	0.0059	0.0064	0	-0.0014	0.0004	0
0	0	0	-0.5355	0	0	1.0338
$t_8 = 24\,\mathrm{h}$						
$\underline{x}^-$	P^-					
42392	511.0893	-159.2552	0	142.8863	-38.9394	0
767	-159.2552	50.0587	0	-44.5714	12.0875	0
0	0	0	49.1491	0	0	-0.3718
-107	142.8863	-44.5714	0	39.9526	-10.8813	0
11043	-38.9394	12.0875	0	-10.8813	2.9719	0
0	0	0	-0.3718	0	0	1.0217

$\underline{x}^+$	P^+					
42148	0.0012	-0.0083	0	0.0012	0.0007	0
843	-0.0083	0.4398	0	-0.0506	-0.0456	0
0	0	0	49.1491	0	0	-0.3718
-175	0.0012	-0.0506	0	0.0062	0.0052	0
11062	0.0007	-0.0456	0	0.0052	0.0050	0
0	0	0	-0.3718	0	0	1.0217

The biggest difference between the predicted and the corrected state occurs at the time step $t_6 = 18\,\text{h}$. At this time step it is around

$$\Delta x(t_6) = \begin{bmatrix} -422\,km \\ -116\,km \\ 0\,km \\ 31\,km/h \\ -76\,km/h \\ 0\,km/h \end{bmatrix}$$

At the time step $t_5 = 15\,\text{h}$ the improvements are the smallest with

$$\Delta x(t_5) = \begin{bmatrix} -12\,km \\ 4\,km \\ 0\,km \\ -1\,km/h \\ -2\,km/h \\ 0\,km/h \end{bmatrix}$$

Furthermore it is shown how the changes, made by the observations in the correction step, affect the accuracy. If the covariance matrix of the observations is smaller than that of the predicted state, the states covariance matrix is improved as one can see in the table.

The following figure shows the calculated trajectory of the geostationary satellite. The red crosses mark the corrected states at the time steps, where the observations are made.

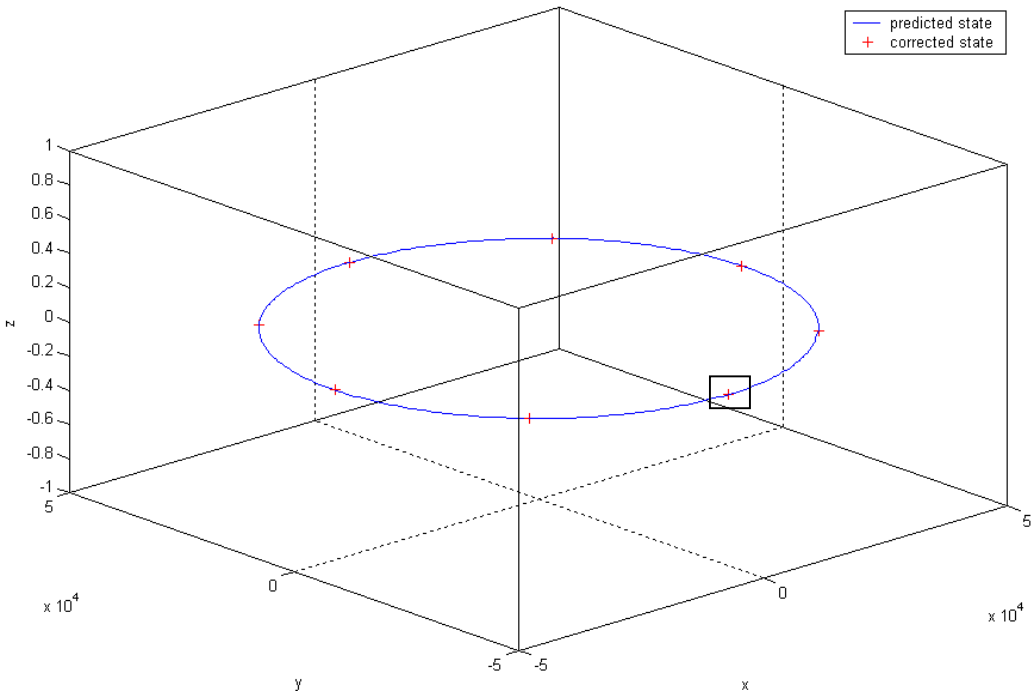

Figure 2: calculated orbit

To pinpoint the correction in a graphic display the plot around the state at the time step t_6 is enlarged.

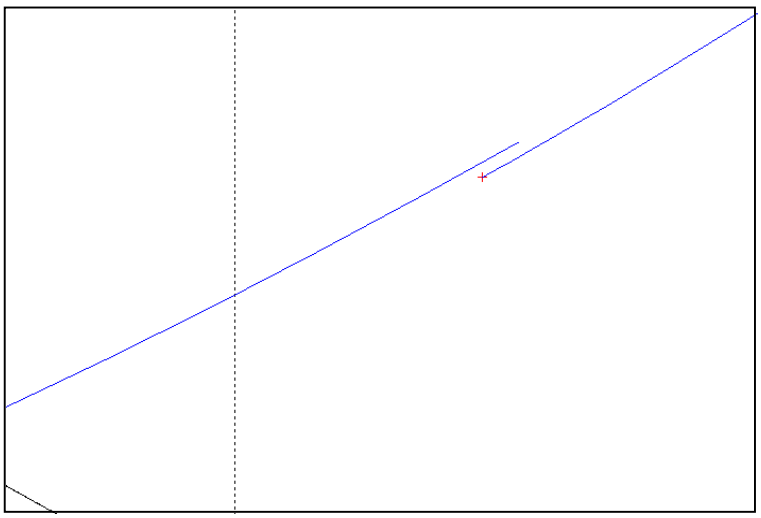

Figure 3:Enlargement

Chapter 8

Conclusion and Prospect

The program that was written in the scope of this study thesis intends to provide a basic understanding of the Kalman filter. Of course this program is too slow for real time computations, because the handling with symbolic expressions in Matlab needs a lot of calculation time.

But in general the recursive nature of the Kalman filter allows for efficient real-time processing. Because of this and because of its apparently simple and easily programmed algorithm, the Kalman filter will continue to play a very important role not only in GPS-based navigation systems.

References

B. Hoffmann-Wellenhof, H. Lichtenegger and J. Collins (1997): *GPS Theory and Practice*, Springer-Verlag Vienna

Martin Vermeer (2004): *Methods of Navigation*, Lecture notes, department of surveying of Helsinki University of Technology (http://www.hut.fi/~mvermeer/nav_en.pdf)

Mohinder S. Grewal, Angus P. Andrews (2001): *Kalman Filtering: Theory and Practice Using Matlab*, John Wiley & Sons Inc 2001

Sorenson, Harold W. (1985): *Kalman filtering: theory and application*, New York

Internet:
http://www.wikipedia.de
http://www.cs.unc.edu/~welch/kalman/