Linear Model Predictive Control Toolbox

User's Guide

John Bagterp Jørgensen

February 2004

2-control ApS
www.2-control.com

ISBN XX-XXXXX-XX-X
Printed by Book Partner, Nørhaven Digital, Copenhagen, Denmark

Contents

1

State Estimation

Table 1.1. Stationary Estimators for LTI Systems.

DesignStationaryLTIKF	Stationary gain and covariance matrices
MeasurementUpdateStationaryLTIKF	Measurement update filter estimates
TimeUpdateStationaryLTIKF	Time update one-step predictive estimates
StatePredictionStationaryLTIKF	State predictions
MeasurementPredictionStationaryLTIKF	Measurement predictions
OutputPredictionStationaryLTIKF	Output predictions
DesignStationaryLTIPKF	Stationary gain and covariance matrices
MeasurementTimeUpdateStationaryLTIPKF	Measurement and time update
StatePredictionStationaryLTIPKF	State predictions
MeasurementPredictionStationaryLTIPKF	Measurement predictions
OutputPredictionStationaryLTIPKF	Output predictions
DesignSmoothStationaryLTIPKF	Gain matrices for stationary smoother
StateSmoothStationaryLTIPKF	Smoothed state estimates
OutputSmoothStationaryLTIPKF	Smoothed output estimates
StationaryStatePredictionCovariance	Stationary state prediction covariance
StationaryOutputPredictionCovariance	Stationary output prediction covariance
StationaryMeasurementPredictionCovariance	Stationary measurement prediction covariance

Table 1.2. Estimators for LTI systems.

MeasurementUpdateLTIKF	Filtered state and process estimate
TimeUpdateLTIKF	Time update in Kalman filter
StatePredictionLTIKF	State predictions in Kalman filter
MeasurementPredictionLTIKF	Measurement predictions in Kalman filter
OutputPredictionLTIKF	Output predictions in Kalman filter
MeasurementTimeUpdateLTIPKF	Update in predictive Kalman filter
StatePredictionLTIPKF	State prediction
MeasurementPredictionLTIPKF	Measurement prediction
OutputPredictionLTIPKF	Output prediction
StateSmoothLTI	Smoothed state estimates
OutputSmoothLTI	Smoothed output estimates

1.1 Introduction

The state estimation problem is central to model predictive controllers, as the estimator incorporates feedback into the model predictive controller. Further the state estimation is used to generate the prediction equations used by the output regulator in the model predictive controller.

Table 1.3. Estimators for LTI Systems with Missing Observations.

MeasurementUpdateLTIKFWMO	Filtered state and process noise
TimeUpdateLTIKFWMO	One-step prediction in Kalman filter
StatePredictionLTIKFWMO	State predictions
MeasurementPredictionLTIKFWMO	Measurement predictions
OutputPredictionLTIKFWMO	Output predictions
MeasurementTimeUpdateLTIPKFWMO	Update in predictive Kalman filter
StatePredictionLTIPKFWMO	State predictions
MeasurementPredictionLTIPKFWMO	Measurement predictions
OutputPredictionLTIPKFWMO	Output predictions
StateSmoothLTIWMO	Smoothed state estimates
OutputSmoothLTIWMO	Smoothed output estimates
BatchUpdateLTIWMO	Update current one-step estimate given a batch of io-data

Table 1.4. Disturbance Models.

DesignDetectableDisturbanceModel	Designs unstructured disturbance model
AugmentLTISystem	Augment with unstructured disturbance model
AugmentLTISystemWithCovariance	Augment with unstructured disturbance model (with covariances)

The state estimation problem treated here is based on stochastic linear time invariant models. The Linear Model Predictive Control Toolbox contains

1. Stationary estimators for LTI systems.
2. Estimators (non-stationary) for LTI systems.
3. Estimators for LTI systems with missing observations (used for situations with online and laboratory measurements).

In addition, it has facilities for augmentation with a disturbance model such that model-plant mismatch as well as unmeasured disturbances can be estimated without offset. These facilities are essential for offset free predictive control.

A number of utility routines for computation of observability, detectability etc. are also provided.

The smoothing, filtering, and prediction problem is shown in figure 1.1. The filtering problem is used for generating the current initial state and the prediction is used by the predictive controller for selecting the best manipulated input trajectory. In additions the predictions may be used for operation visualization. Smoothing may also be used for operation visualization, but plays no role in the estimator-output regulator loop for linear systems.

Kailath *et al.* (2000) provide a comprehensive introduction to the state estimation problem for linear systems.

1.2 The State Estimation Problem

The various state estimation algorithms considered in this chapter assume unless otherwise stated that the system can be described by a discrete time linear

Table 1.5. Utility Functions.

IsControllable	Tests whether (A, B) is controllable
IsObservable	Tests whether (C, A) is observable
IsStabilizable	Tests whether (A, B) is stabilizable
IsDetectable	Tests whether (C, A) is detectable
IsUnitCircleControllable	Tests whether integrating modes of (A, B) can be controlled
IsUnitCircleDetectable	Tests whether integrating modes of (C, A) can be observed
IsStable	Tests whether the discrete-time matrix A is stable

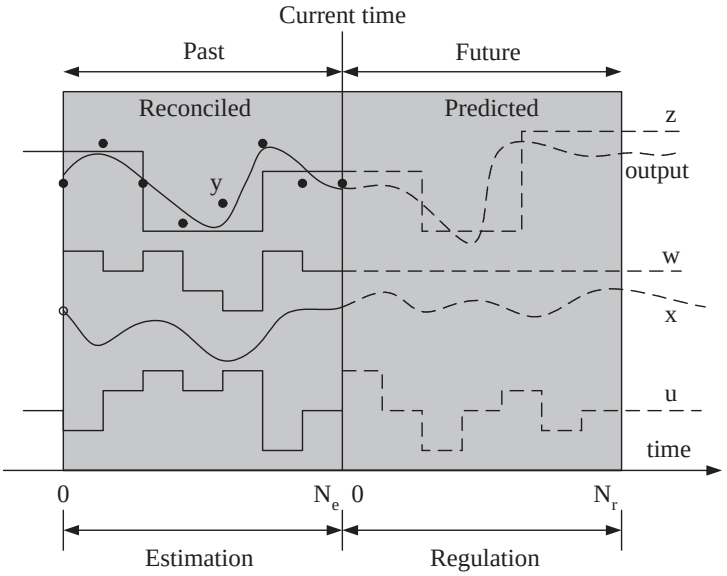

Figure 1.1. Moving horizon estimation and control.

(affine) stochastic state space model

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k u_k + G_k \boldsymbol{w}_k + d_k \tag{1.1a}$$

$$\boldsymbol{y}_k = C_k \boldsymbol{x}_k + \boldsymbol{v}_k + f_k \tag{1.1b}$$

in which the process noise, $\boldsymbol{w}_k$, and the measurement noise, $\boldsymbol{v}_k$ are identically independently normally distributed

$$\begin{bmatrix} \boldsymbol{w}_k \\ \boldsymbol{v}_k \end{bmatrix} \sim N_{iid}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} Q_k & S_k \\ S_k' & R_k \end{bmatrix} \right) \tag{1.1c}$$

and the initial state is assumed to have the normal distribution

$$\boldsymbol{x}_0 \sim N(\hat{x}_{0|-1}, P_{0|-1}) \tag{1.1d}$$

and be independent of the process and measurement noise.

Boldface symbols denote stochastic variables.

For most cases considered, the manipulated variables, u_k, are assumed to be deterministic. However, the situation in which the future u_k are stochastic is discussed as well. The parameters d_k and f_k are assumed to be deterministic exogenous signals known to all estimators considered here.

In a statistical sense the state estimation problem can be stated as the linear-quadratic (least-squares) optimization problem (Jazwinski, 1970)

$$\min_{\{x_k,w_k,v_k\}_{k=0}^{\infty}} \quad \phi = \frac{1}{2}(x_0 - \hat{x}_{0|-1})' P_{0|-1}^{-1}(x_0 - \hat{x}_{0|-1}) + \frac{1}{2}\sum_{k=0}^{\infty} \begin{bmatrix} w_k \\ v_k \end{bmatrix}' \begin{bmatrix} Q_k & S_k \\ S_k' & R_k \end{bmatrix}^{-1} \begin{bmatrix} w_k \\ v_k \end{bmatrix} \tag{1.2a}$$

$$s.t. \qquad x_{k+1} = A_k x_k + B_k u_k + G_k w_k + d_k \qquad k = 0, 1, \ldots \tag{1.2b}$$

$$y_k = C_k x_k + f_k + v_k \qquad k = 0, 1, \ldots, N \tag{1.2c}$$

Let $\left\{\hat{x}_{k|N}, \hat{w}_{k|N}, \hat{v}_{k|N}\right\}_{k=0}^{\infty}$ denote the entire optimal solution profile to the above problem given measurements from discrete time 0 to discrete time N, i.e. to the problem when $\mathcal{Y}_N = \{y_0, y_1, \ldots, y_N\}$ is given. The optimal solution is called the optimal estimate and $\left\{\hat{x}_{k|N}\right\}_{k=0}^{\infty}$ is called the optimal state estimate.

Depending on the time k at which an estimate of the state is required, the state estimation problem may be categorized as

1. **Smoothing**: An estimate, $\hat{x}_{k|N}$ with $k < N$, of the state in the *past* is required.
2. **Filtering**: An estimate, $\hat{x}_{k|N}$ with $k = N$, of the *current* state is required.
3. **Prediction**: An estimate, $\hat{x}_{k|N}$ with $k > N$, of the state in the *future* is required.

Probabilistically, the state estimate problem may be regarded as the conditional mean (Jazwinski, 1970)

$$\hat{x}_{k|N} = E\left\{\boldsymbol{x}_k \,|\, \mathcal{Y}_N\right\} \tag{1.3a}$$

and likewise the process and measurement noise estimates may be interpreted as conditional means

$$\hat{w}_{k|N} = E\left\{\boldsymbol{w}_k \,|\, \mathcal{Y}_N\right\} \tag{1.3b}$$

$$\hat{v}_{k|N} = E\left\{\boldsymbol{v}_k \,|\, \mathcal{Y}_N\right\} \tag{1.3c}$$

The class of problems solved in the linear model predictive control toolbox are for linear (affine) Gaussian systems. For these problems the statistical and the probabilistic estimates are identical.

In predictive control, the filtered estimate is used to incorporate feedback in the controller and the predicted estimates are used in the computation of the optimal control sequence as well as for visualizing the predicted future evolution of the system to the operators.

Smoothed estimates may be used for post processing data and for providing visualization of the most likely past evolution of the system to operators.

1.3 Measurement-Update Time-Update Kalman Filter

The measurement-update time-update Kalman filter contains a true filtering algorithm and a one-step predictor for updating the memory of the filter. At

each sample instant the filtered state $\hat{x}_{k|k}$ and $\hat{w}_{k|k}$ are computed. These estimates form the basis for the prediction made by the model predictive controller in forecasting the future process evolution.

1.3.1 Recursions of the Algorithm

The recursion for a basic time variant Kalman filter with the possibility of missing observations is stated in this subsection. The time variance of the matrices, i.e. the time variance of C_k, S_k, and R_k are important for being able to adjust the filter when some of the values are not measured, e.g. laboratory measurements which are available at a much lower frequency than the online measurements.

At each sampling instant $t_k = kT_s$ for $k = 0, 1, 2, \ldots$ when a new measurement, y_k, becomes available to the estimator, the innovation e_k and the state and process noise filter gains, $K_{fx,k}$ and $K_{fw,k}$ are computed by

$$\hat{y}_{k|k-1} = C_k\hat{x}_{k|k-1} + f_k \tag{1.4a}$$
$$e_k = y_k - \hat{y}_{k|k-1} \tag{1.4b}$$
$$R_{e,k} = C_k P_{k|k-1} C_k' + R_k \tag{1.4c}$$
$$K_{fx,k} = P_{k|k-1} C_k' R_{e,k}^{-1} \tag{1.4d}$$
$$K_{fw,k} = S_k R_{e,k}^{-1} \tag{1.4e}$$

All matrices and vectors are time variant. This allows for missing observations, e.g. laboratory measurements that are not available. The dimensions of the matrices C_k, R_k and S_k as well as the vector f_k are adjusted according to which measurements are available and the innovation as well as the gain matrices are computed with appropriate dimension allowing for a missing observation.

The filtered estimates of the process noise and states, given the measurements $\mathcal{Y}_k = \{y_l\}_{l=0}^{k}$, are computed by

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_{fx,k} e_k \tag{1.5a}$$
$$\hat{w}_{k|k} = K_{fw,k} e_k \tag{1.5b}$$

The estimates $\hat{x}_{k|k}$ and $\hat{w}_{k|k}$ are used by the predictive dynamic regulator (controller) to predict the future evolution of the process. Hence, when this information has been computed the regulation algorithm may be invoked and the optimal control u_k computed and implemented.

When u_k has been computed the Kalman filter is prepared for the next measurement by computing a one-step prediction of the states

$$\hat{x}_{k+1|k} = A_k\hat{x}_{k|k} + B_k u_k + G_k\hat{w}_{k|k} + d_k \tag{1.6}$$

and a one-step prediction of the associated covariance

$$P_{k|k} = P_{k|k-1} - K_{fx,k} R_{e,k} K_{fx,k}' \tag{1.7a}$$
$$Q_{k|k} = Q_k - K_{fw,k} R_{e,k} K_{fw,k}' \tag{1.7b}$$
$$P_{k+1|k} = A_k P_{k|k} A_k' + G_k Q_{k|k} G_k' - A_k K_{fx,k} S_k' G_k' - G_k S_k K_{fx,k}' A_k' \tag{1.7c}$$

Computation of $\hat{x}_{k|k}$, $\hat{w}_{k|k}$, $P_{k|k}$, and $Q_{k|k}$ is the measurement-update, while calculation of $\hat{x}_{k+1|k}$ and $P_{k+1|k}$ is the time update.

The conditional stochastic variables $\boldsymbol{x}_k|\mathcal{Y}_k$, $\boldsymbol{w}_k|\mathcal{Y}_k$, and $\boldsymbol{w}_k|\mathcal{Y}_k$ have the distributions

$$\boldsymbol{x}_k|\mathcal{Y}_k \sim N(\hat{x}_{k|k}, P_{k|k}) \tag{1.8a}$$
$$\boldsymbol{w}_k|\mathcal{Y}_k \sim N(\hat{w}_{k|k}, Q_{k|k}) \tag{1.8b}$$
$$\boldsymbol{x}_{k+1|k}|\mathcal{Y}_k \sim N(\hat{x}_{k+1|k}, P_{k+1|k}) \tag{1.8c}$$

These distribution may be used for computing confidence intervals for the estimates $\hat{x}_{k|k}$, $\hat{w}_{k|k}$, and $\hat{x}_{k+1|k}$, respectively.

1.3.2 Application in a Predictive Controller

When the filtered estimate is applied in a moving horizon predictive controller with sampling time T_s, the implicit assumption is that the computational time for the estimator as well as the dynamic output regulator is negligible. At each sample cycle, the following steps are conducted

1. Read measurements, y_k (these may be filtered values of measurements sampled at a much higher frequency than the sampling time of the moving horizon controller).
2. Compute the filtered state, $\hat{x}_{k|k}$, and process noise $\hat{w}_{k|k}$.
3. Compute the optimal open loop control sequence $\left\{\hat{u}_{k+j|k}\right\}_{j=0}^{N-1}$ as the solution to the output regulation problem. N is the control horizon. The predictors described later in this chapter are employed by the predictive controller.
4. Implement $u_k = \hat{u}_{k+j|k}$ on the process.
5. Compute the one-step predicted state, $\hat{x}_{k+1|k}$, and its covariance, $P_{k+1|k}$.
6. Store $\hat{x}_{k+1|k}$ and $P_{k+1|k}$.

The computations sketched above has been decomposed into an output computation, e.g. computation of u_k, and a state update, e.g. computation of $\hat{x}_{k+1|k}$ and $P_{k+1|k}$, as described in Wittenmark *et al.* (2002). The idea behind this decomposition is to compute and implement u_k as fast as possible.

1.4 One-Step Predictive Kalman Filter

When the computing time of the predictive controller cannot be neglected, a feasible alternative is to implement $u_k = \hat{u}_{k|k-1}$ at the beginning of each sample interval and subsequently compute $\hat{u}_{k+1|k}$ to be implemented at the next sample interval. The implicit assumption is that the estimation and output regulation problem can be solved within one sample time. In this approach the control algorithm introduces a delay of one sample time to the overall system.

The estimation part of this procedure may be conducted doing a filtering and a one-step prediction as described above. However, a more efficient approach in this situation is to employ the one-step predictive Kalman filter.

1.4.1 Algorithm

Again the time variance in the matrices C_k, S_k, and R_k may be used to handle situations with missing observations.

At each cycle the covariance of the innovations, $R_{e,k}$, and the predictive Kalman gain, $K_{p,k}$, are computed by

$$R_{e,k} = C_k P_{k|k-1} C_k' + R_k \tag{1.9a}$$

$$K_{p,k} = (A_k P_{k|k-1} C_k' + G_k S_k) R_{e,k}^{-1} \tag{1.9b}$$

Subsequently, the one-step predicted state is computed according to the following sequence of operations

$$\hat{y}_{k|k-1} = C_k \hat{x}_{k|k-1} + f_k \tag{1.10a}$$

$$e_k = y_k - \hat{y}_{k|k-1} \tag{1.10b}$$

$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k-1} + B_k u_k + d_k + K_{p,k} e_k \tag{1.10c}$$

The covariance, $P_{k+1|k}$, of the one step prediction is computed by

$$P_{k+1|k} = A_k P_{k|k-1} A_k' + G_k Q_k G_k' - K_{p,k} R_{e,k} K_{p,k}' \tag{1.11}$$

Note that the one-step predicted process noise is zero, $\hat{w}_{k+1|k} = 0$. Hence, $\hat{x}_{k+1|k}$ (and $\hat{w}_{k+1|k} = 0$ is sufficient information for computation of $u_{k+1} = \hat{u}_{k+1|k}$ using the algorithm for the dynamic output regulator.

The stochastic conditional variable $\boldsymbol{x}_{k+1}|\mathcal{Y}_k$ has the distribution

$$\boldsymbol{x}_{k+1}|\mathcal{Y}_k \sim N(\hat{x}_{k+1|k}, P_{k+1|k}) \tag{1.12}$$

which may be used in the computation of confidence intervals for the estimate $\hat{x}_{k+1|k}$.

1.4.2 Application in a Predictive Controller

The sequence of operations conducted periodically with periods of one sample-time, T_s, of the moving horizon predictive controller using a one-step predictive Kalman filter is

1. Implement $u_k = \hat{u}_{k|k-1}$ on the process.
2. Read measurements, y_k.
3. Compute the innovation, e_k, the one-step prediction of the state, $\hat{x}_{k+1|k}$, and its covariance $P_{k+1|k}$.
4. Compute the optimal open loop control sequence $\{\hat{u}_{k+1+j|k}\}_{j=0}^{N-1} = \mu(\hat{x}_{k+1|k}, 0)$ (using $\hat{w}_{k+1|k} = 0$) as the solution to the output regulation problem.
5. Store $\hat{u}_{k+1|k}$, $\hat{x}_{k+1|k}$, and $P_{k+1|k}$.

The initial control $u_0 = \hat{u}_{0|-1}$ may be computed in advance by solving the output regulation problem using $\hat{x}_{0|-1}$, e.g. $\left\{\hat{u}_{j|-1}\right\}_{j=0}^{N-1} = \mu(\hat{x}_{0|-1}, 0)$.

1.5 Open Loop Prediction

The equations for the prediction of the system are used by the predictive controller in the dynamic output regulation problem. In this application the prediction functions are not called explicitly, but the construction of the predictive controller is based on these equations.

Another application of the prediction may be for visualization of the predicted future process evaluation to operators. For this purpose the prediction functions may be needed.

1.5.1 One-Step State Prediction

The one-step state prediction may be obtained by either of the following methods

1. The measurement-update time-update Kalman filter (see section 1.3).
2. The one-step predictive Kalman filter (see section 1.4).

1.5.1.1 Measurement-Update Time-Update One-Step Prediction

Due to the time variance of the matrices, they can include situations with missing observations.

The equations for the Kalman gains in the measurement update are

$$R_{e,k} = C_k P_{k|k-1} C_k' + R_k \tag{1.13a}$$
$$K_{fx,k} = P_{k|k-1} C_k' R_{e,k}^{-1} \tag{1.13b}$$
$$K_{fw,k} = S_k R_{e,k}^{-1} \tag{1.13c}$$

which may be used in computing the one-step prediction of the states by the equations

$$\hat{y}_{k|k-1} = C_k \hat{x}_{k|k-1} + f_k \tag{1.14a}$$
$$e_k = y_k - \hat{y}_{k|k-1} \tag{1.14b}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_{fx,k} e_k \tag{1.14c}$$
$$\hat{w}_{k|k} = K_{fw,k} e_k \tag{1.14d}$$
$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k} + B_k u_k + G_k \hat{w}_{k|k} + d_k \tag{1.14e}$$

If the covariance of the prediction is desired as well, it may be computed by

$$P_{k|k} = P_{k|k-1} - K_{fx,k} R_{e,k} K_{fx,k}' \tag{1.15a}$$
$$Q_{k|k} = Q_k - K_{fw,k} R_{e,k} K_{fw,k}' \tag{1.15b}$$
$$P_{k+1|k} = A_k P_{k|k} A_k' + G_k Q_{k|k} G_k' - A_k K_{fx,k} S_k' G_k' - G_k S_k K_{fx,k}' A_k' \tag{1.15c}$$

The covariance $P_{k+1|k}$ may be used for computation of a confidence interval for the prediction $\hat{x}_{k+1|k}$ using that $\boldsymbol{x}_{k+1}|\mathcal{Y}_k$ is distributed as

$$\boldsymbol{x}_{k+1}|\mathcal{Y}_k \sim N(\hat{x}_{k+1|k}, P_{k+1|k}) \tag{1.16}$$

1.5.1.2 One-Step Predictive Kalman filter

A more direct way to compute the one-step predicted value of the state is by the one-step predictive Kalman filter. As the matrices are time variant, situations with missing observations are accommodated.

The predictive Kalman gain, $K_{p,k}$, is computed by

$$R_{e,k} = C_k P_{k|k-1} C_k' + R_k \tag{1.17a}$$

$$K_{p,k} = (A_k P_{k|k-1} C_k' + G_k S_k) R_{e,k}^{-1} \tag{1.17b}$$

which is used to compute $\hat{x}_{k+1|k}$ using the equations

$$\hat{y}_{k|k-1} = C_k \hat{x}_{k|k-1} + f_k \tag{1.18a}$$

$$e_k = y_k - \hat{y}_{k|k-1} \tag{1.18b}$$

$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k-1} + B_k u_k + d_k + K_{p,k} e_k \tag{1.18c}$$

The covariance, $P_{k+1|k}$, of the one step prediction is computed by

$$P_{k+1|k} = A_k P_{k|k-1} A_k' + G_k Q_k G_k' - K_{p,k} R_{e,k} K_{p,k}' \tag{1.19}$$

and may be used to establish confidence intervals of the prediction.

The stochastic conditional variable $\boldsymbol{x}_{k+1}|\mathcal{Y}_k$ has the distribution

$$\boldsymbol{x}_{k+1}|\mathcal{Y}_k \sim N(\hat{x}_{k+1|k}, P_{k+1|k}) \tag{1.20}$$

which may be used for computation of confidence intervals for the estimate $\hat{x}_{k+1|k}$.

1.5.2 (j+1)-Step State Predictions ($j > 0$)

$(j+1)$-step state prediction with $j > 0$ assumes that $\hat{x}_{k+1|k}$ has been computed. If covariances are desired in addition it also assumes that $P_{k+1|k}$ has been computed.

Notice that $\hat{w}_{k+j|k} = 0$ for $j > 0$. Consequently a (j+1)-step state prediction may be computed by recursively applying the formula

$$\hat{x}_{k+j+1|k} = A_{k+j} \hat{x}_{k+j|k} + B_{k+j} \hat{u}_{k+j|k} + d_{k+j} \qquad j > 0 \tag{1.21}$$

If desired, the associated covariances may be computed recursively using the equation

$$P_{k+j+1|k} = A_{k+j} P_{k+j|k} A_{k+j}' + G_{k+j} Q_{k+j} G_{k+j}' \qquad j > 0 \tag{1.22}$$

In this formula for the covariance we have assumed that $u_{k+j} = \hat{u}_{k+j|k}$ is a deterministic signal, i.e. that its covariance is zero. Due to the normality of the stochastic variables and the linearity of the system, $\boldsymbol{x}_{k+j+1}|\mathcal{Y}_k$ has the distribution

$$x_{k+1+j}|\mathcal{Y}_k \sim N(\hat{x}_{k+1+j|k}, P_{k+1+j|k}) \tag{1.23}$$

1.5.3 Measurement Predictions

It is obvious that $\hat{y}_{k|k} = y_k$ with zero covariance.
For $j > 0$ the predicted measurements are

$$\hat{y}_{k+j|k} = C_{k+j}\hat{x}_{k+j|k} + f_{k+j} \qquad j > 0 \tag{1.24}$$

and the associated covariance is

$$\Theta_{k+j|k} = C_{k+j}P_{k+j|k}C'_{k+j} + R_{k+j} \qquad j > 0 \tag{1.25}$$

The conditional variable $\boldsymbol{y}_{k+j}|\mathcal{Y}_k$ has the distribution

$$\boldsymbol{y}_{k+j}|\mathcal{Y}_k \sim N(\hat{y}_{k+j|k}, \Theta_{k+j|k}) \qquad j > 0 \tag{1.26}$$

This property may be used in computing confidence intervals for the prediction.

1.5.4 Output Predictions

Often one is also interested in an output signal, $\boldsymbol{z}_k$, which is related to the states, $\boldsymbol{x}_k$, by

$$\boldsymbol{z}_k = H_k\boldsymbol{x}_k + h_k \tag{1.27}$$

The predicted (and filtered) output $\hat{z}_{k+j|k}$ for $j \geq 0$ may be computed by

$$\hat{z}_{k+j|k} = H_{k+j}\hat{x}_{k+j|k} + h_{k+j} \qquad j \geq 0 \tag{1.28}$$

The associated covariance may be computed by

$$\Xi_{k+j|k} = H_{k+j}P_{k+j|k}H'_{k+j} \qquad j \geq 0 \tag{1.29}$$

The conditional variable $\boldsymbol{z}_{k+j}|\mathcal{Y}_k$ has the distribution

$$\boldsymbol{z}_{k+j}|\mathcal{Y}_k \sim N(\hat{z}_{k+j|k}, \Xi_{k+j|k}) \qquad j \geq 0 \tag{1.30}$$

This property may be used for computing confidence intervals for the prediction, $\hat{z}_{k+j|k}$ for $j \geq 0$.

1.5.4.1 Application of Output Predictions

Output signals may be used for

1. Representation of measurements without measurement noise, e.g. $\boldsymbol{z}_k = C_k\boldsymbol{x}_k + g_k$ and $\boldsymbol{y}_k = \boldsymbol{z}_k + \boldsymbol{v}_k = C_k\boldsymbol{x}_k + f_k + \boldsymbol{v}_k$.
2. Representation of a subset of the measurements without the measurement noise, e.g. $\boldsymbol{z}_k = F(C_k\boldsymbol{x}_k + f_k)$ and $\boldsymbol{y}_k = C_k\boldsymbol{x}_k + f_k + \boldsymbol{v}_k$.

1.6 Smoothing

Past system trajectories may be computed by smoothing. For linear systems smoothing is not needed for the controller as filtering and prediction is sufficient. Smoothing is merely used to compute a trajectory of the past which may be visualized to operators.

1.6.1 Bryson-Frazier Smoothing Algorithm

The smoothing algorithm described is the Bryson-Frazier smoothing algorithm. The boundary conditions used by the algorithm are the initial values $\hat{x}_{0|-1}$ and $P_{0|-1}$ given by the model specification as well as

$$\lambda_{N+1|N} = 0 \tag{1.31a}$$
$$\Lambda_{N+1|N} = 0 \tag{1.31b}$$

For $k = 0, 1, \ldots, N-1$ compute

$$R_{e,k} = C_k P_{k|k-1} C_k' + R_k \tag{1.32a}$$
$$K_{p,k} = (A_k P_{k|k-1} C_k' + G_k S_k) R_{e,k}^{-1} \tag{1.32b}$$
$$P_{k+1|k} = A_k P_{k|k-1} A_k' + G_k Q_k G_k' - K_{p,k} R_{e,k} K_{p,k}' \tag{1.32c}$$
$$\hat{y}_{k|k-1} = C_k \hat{x}_{k|k-1} + f_k \tag{1.32d}$$
$$e_k = y_k - \hat{y}_{k|k-1} \tag{1.32e}$$
$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k-1} + B_k u_k + d_k + K_{p,k} e_k \tag{1.32f}$$

and subsequently the smoothed states $\left\{\hat{x}_{k|N}\right\}_{k=0}^{N}$ are computed by the backward recursion for $k = N, N-1, \ldots, 0$

$$\lambda_{k|N} = (A_k - K_{p,k} C_k)' \lambda_{k+1|N} + C_k' R_{e,k}^{-1} e_k \tag{1.33a}$$
$$\hat{x}_{k|N} = \hat{x}_{k|k-1} + P_{k|k-1} \lambda_{k|N} \tag{1.33b}$$

If the smoothed states and their covariance is desired the following backward recursions are applied for $k = N, N+1, \ldots, 0$

$$F_k = A_k - K_{p,k} C_k \tag{1.34a}$$
$$\lambda_{k|N} = F_k' \lambda_{k+1|N} + C_k' R_{e,k}^{-1} e_k \tag{1.34b}$$
$$\Lambda_{k|N} = F_k' \Lambda_{k+1|N} F_k + C_k' R_{e,k}^{-1} C_k \tag{1.34c}$$
$$\hat{x}_{k|N} = \hat{x}_{k|k-1} + P_{k|k-1} \lambda_{k|N} \tag{1.34d}$$
$$P_{k|N} = P_{k|k-1} - P_{k|k-1} \Lambda_{k|N} P_{k|k-1} \tag{1.34e}$$

The smoothed process noise, $\left\{\hat{w}_{k|N}\right\}_{k=0}^{N}$, and its covariance, $\left\{Q_{k|N}\right\}_{k=0}^{N}$, may be computed by doing the following computations in addition to (1.34) at each time instant k

$$E_k = Q_k G_k' - S_k K_{p,k}' \tag{1.35a}$$
$$\hat{w}_{k|N} = E_k \lambda_{k+1|N} + S_k R_{e,k}^{-1} e_k \tag{1.35b}$$
$$Q_{k|N} = Q_k - S_k R_{e,k}^{-1} S_k' - E_k \Lambda_{k+1|N} E_k' \tag{1.35c}$$

The conditional stochastic variable $\boldsymbol{x}_k | \mathcal{Y}_N$ has the distribution

$$\boldsymbol{x}_k | \mathcal{Y}_N \sim N(\hat{x}_{k|N}, P_{k|N}) \tag{1.36}$$

This information about the distribution, may be used in computation of confidence interval for the smoothed states, $\hat{x}_{k|N}$.

1.6.1.1 Output Smoothing

In many case, the primary interest is not in the state but some outputs, $\boldsymbol{z}_k$. Let the output be related to the states by

$$\boldsymbol{z}_k = H_k \boldsymbol{x}_k + h_k \tag{1.37}$$

The smoothed output variable estimate, $\hat{z}_{k|N}$, and the associated covariance, $\Xi_{k|N}$, may be computed by

$$\hat{z}_{k|N} = H_k \hat{x}_{k|N} + h_k \tag{1.38a}$$
$$\Xi_{k|N} = H_k P_{k|N} H_k' \tag{1.38b}$$

$\boldsymbol{z}_k|\mathcal{Y}_N$ has the distribution

$$\boldsymbol{z}_k|\mathcal{Y}_N \sim N(\hat{z}_{k|N}, \Xi_{k|N}) \tag{1.39}$$

which may be used in the computation of confidence intervals for the smoothed output estimate, $\hat{z}_{k|N}$.

1.7 Numerical Implementation

Currently, the recursions are implemented in a straight forward manner. Symmetry of $P_{k+1|k}$ is enforced by

$$P_{k+1|k} \leftarrow \frac{P_{k+1|k} + P_{k+1|k}'}{2} \tag{1.40}$$

This tends to stabilize the recursions.

Future version should in addition contain a square-root version in which only half of the symmetric matrix is stored in memory (Kailath *et al.*, 2000).

1.8 Linear Time Invariant Models

In this section we specialize the treatment to linear (affine) time invariant models, i.e. models with $(A, B, G, d, C, f, Q, S, R)$ independent of time. By using time invariant models, situations with missing observations cannot be handled. Linear time invariant models are well suited for analyzing the asymptotic behavior of the estimators and to provide design guidelines. These design guidelines is useful for the time variant models as well. In addition computational efficient stationary estimators can be developed for linear time invariant systems.

The models considered are discrete-time stochastic linear time invariant difference equations of the form

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + Bu_k + G\boldsymbol{w}_k + d \tag{1.41a}$$
$$\boldsymbol{y}_k = C\boldsymbol{x}_k + \boldsymbol{v}_k + f \tag{1.41b}$$

in which the process noise, $\boldsymbol{w}_k$, and the measurement noise, $\boldsymbol{v}_k$, are distributed as

$$\begin{bmatrix} \boldsymbol{w}_k \\ \boldsymbol{v}_k \end{bmatrix} \sim N_{iid}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} Q & S \\ S' & R \end{bmatrix} \right) \tag{1.41c}$$

and the initial state has the distribution

$$\boldsymbol{x}_0 \sim N(\hat{x}_{0|-1}, P_{0|-1}) \tag{1.41d}$$

In addition, outputs of the form

$$\boldsymbol{z}_k = H\boldsymbol{x}_k + h \tag{1.42}$$

are occasionally considered.

1.8.1 Discrete Algebraic Riccati Equation (DARE)

Consider the following recursions in the one-step predictive Kalman filter

$$R_{e,k} = CP_{k|k-1}C' + R \tag{1.43a}$$
$$K_{p,k} = (AP_{k|k-1}C' + GS)R_{e,k}^{-1} \tag{1.43b}$$
$$P_{k+1|k} = AP_{k|k-1}A' + GQG' - K_{p,k}R_{e,k}K_{p,k}' \tag{1.43c}$$

which implies that the recursion for $P_{k+1|k}$ may be expressed as

$$\begin{aligned} P_{k+1|k} = {} & AP_{k|k-1}A' + GQG' \\ & - (AP_{k|k-1}C' + GS)(CP_{k|k-1}C' + R)^{-1}(AP_{k|k-1}C' + GS)' \end{aligned} \tag{1.44}$$

If the sequence of matrices $\left\{P_{k|k-1}\right\}$ converges to a matrix P, this implies $P = P_{k+1|k} = P_{k|k-1}$ for $k \to \infty$ and the recursions (1.43) may be expressed as

$$R_e = CPC' + R \tag{1.45a}$$
$$K_p = (APC' + GS)R_e^{-1} \tag{1.45b}$$
$$P = APA' + GQG' - K_pR_eK_p' \tag{1.45c}$$

which imply that P satisfies

$$P = APA' + GQG' - (APC' + GS)(CPC' + R)^{-1}(APC' + GS)' \tag{1.46}$$

This equation is a discrete algebraic Riccati equation. In the following subsections we will summarize conditions for existence of a solution, P, to this equation and sufficient conditions for convergence of the matrices $\left\{P_{k|k-1}\right\}$ generated by (1.43) to the solution P of the discrete algebraic Riccati equation (1.46).

1.8.1.1 Case: $S = 0$, $R > 0$

In the case $S = 0$, the discrete algebraic Riccati equation (DARE) (1.46) becomes

$$\begin{aligned} P &= APA' + GQG' - APC'(CPC' + R)^{-1}(APC')' \\ &= APA' + GQG' - APC'(CPC' + R)^{-1}CPA' \end{aligned} \tag{1.47}$$

The following properties are known for the discrete algebraic Riccati equation (1.47):

1. DARE (1.47) has a stabilizing solution if and only if (A, C) is detectable and $(A, GQ^{1/2})$ is controllable on the unit circle. Any such solution is unique and also positive definite.

2. Assume that (A, C) is detectable and $(A, GQ^{1/2})$ is controllable on the unit circle. Then the solution of DARE will have only one positive semi-definite solution *if and only if* $(A, GQ^{1/2})$ is stabilizable. The unique positive semi-definite solution of the DARE (1.47) also defines its stabilizing solution.

A stabilizing solution is a solution for which $A - K_p C$ is stable. This implies that the corresponding steady-state one-step predictive Kalman filter is stable.

According to Kailath *et al.* (2000) and Zhou *et al.* (1996), the rank tests for stabilizability, detectability, and controllability on the unit circle may be conducted as

1. Let $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$. The system (A, B) is stabilizable if and only if rank $\begin{bmatrix}\lambda I - A & B\end{bmatrix} = n$ at all unstable eigenvalues, λ, of A.

2. Let $A \in \mathbb{R}^{n\times n}$ and $G \in \mathbb{R}^{n\times m}$. The system (A, B) is controllable on the unit circle if and only if rank $\begin{bmatrix}\lambda I - A & B\end{bmatrix} = n$ for all eigenvalues, λ, of A with $|\lambda| = 1$.

3. Let $A \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{p\times n}$. The system (C, A) is detectable if and only if rank $\begin{bmatrix}\lambda I - A \\ C\end{bmatrix} = n$ at all unstable eigenvalues, λ, of A.

The convergence of the sequence of matrices $\left\{P_{k|k-1}\right\}$ generated by the Riccati recursions (1.43) is assured by the *sufficient* (but not necessary) conditions stated in the following proposition.

Proposition 1.8.1 (Sufficient Conditions for Convergence)
Assume

1. (A, C) is detectable.

2. $(A, GQ^{1/2})$ is stabilizable.

3. $P_{0|-1} \geq 0$.

Let $\left\{P_{k|k-1}\right\}$ be a sequence of matrices generated by

$$R_{e,k} = CP_{k|k-1}C' + R \tag{1.48a}$$

$$K_{p,k} = AP_{k|k-1}C'R_{e,k}^{-1} \tag{1.48b}$$

$$P_{k+1|k} = AP_{k|k-1}A' + GQG' - K_{p,k}R_{e,k}K_{p,k}' \tag{1.48c}$$

Then

$$\lim_{k\to\infty} P_{k|k-1} = P \tag{1.49}$$

in which P is the solution of the discrete algebraic Riccati equation (1.47).

By assumption 1) and 2) this solution, P, is unique, positive semi-definite, and stabilizing (i.e. $A - K_p C$ is stable).

■

1.8.1.2 Case: $S \neq 0$, $R > 0$

In the case $S \neq 0$, the discrete algebraic Riccati equation (1.46) to be solved is

$$P = APA' + GQG' - (APC' + GS)(CPC' + R)^{-1}(APC' + GS)' \tag{1.50}$$

Since $R > 0$, we may define A_s and Q_s as

$$A_s = A - GSR^{-1}C \tag{1.51a}$$
$$Q_s = Q - SR^{-1}S \tag{1.51b}$$

Using these definitions of A_s and Q_s, the discrete time algebraic Riccati equation (1.46) may be expressed in the same structural form as the discrete time algebraic Riccati equation (1.47) for the case when $S = 0$, i.e.

$$\begin{aligned} P &= APA' + GQG' - (APC' + GS)(CPC' + R)^{-1}(APC' + GS)' \\ &= A_s P A_s' + GQ_s G' - A_s PC'(CPC' + R)^{-1} CPA_s' \end{aligned} \tag{1.52}$$

Consequently, the conditions for existence of solutions to the discrete algebraic Riccati equation (1.46) can be stated in terms of A_s and Q_s using the conditions for the case when $S = 0$.

The following properties are known for the discrete algebraic Riccati equation (1.46):

1. DARE (1.46) has a stabilizing solution if and only if (A_s, C) is detectable and $(A_s, GQ_s^{1/2})$ is controllable on the unit circle. Any such solution is unique and also positive definite.

2. Assume that (A_s, C) is detectable and $(A_s, GQ_s^{1/2})$ is controllable on the unit circle. Then the solution of DARE (1.46) will have only one positive semi-definite solution *if and only if* $(A_s, GQ_s^{1/2})$ is stabilizable. The unique positive semi-definite solution of the DARE (1.46) also defines its stabilizing solution.

The convergence of the sequence of matrices $\left\{P_{k|k-1}\right\}$ generated by the Riccati recursions (1.43) is assured by the *sufficient* (but not necessary) conditions stated in the following proposition.

Proposition 1.8.2 (Sufficient Conditions for Convergence)
Let

$$A_s = A - GSR^{-1}C \tag{1.53a}$$
$$Q_s = Q - SR^{-1}S \tag{1.53b}$$

and assume that $R > 0$. Assume

1. (A_s, C) is detectable.
2. $(A_s, GQ_s^{1/2})$ is stabilizable.
3. $P_{0|-1} \geq 0$.

Let $\left\{P_{k|k-1}\right\}$ be a sequence of matrices generated by

$$R_{e,k} = CP_{k|k-1}C' + R \tag{1.54a}$$
$$K_{p,k} = (AP_{k|k-1}C' + GS)R_{e,k}^{-1} \tag{1.54b}$$
$$P_{k+1|k} = AP_{k|k-1}A' + GQG' - K_{p,k}R_{e,k}K_{p,k}' \tag{1.54c}$$

Then

$$\lim_{k\to\infty} P_{k|k-1} = P \tag{1.55}$$

in which P is the solution of the discrete algebraic Riccati equation (1.46).

By assumption 1) and 2) this solution, P, is unique, positive semi-definite, and stabilizing (i.e. $A - K_pC$ is stable).

■

This proposition implies that under the conditions stated in the proposition, the predictive Kalman gain, $K_{p,k}$, converges to K_p, which may be computed using the solution, P, of the discrete algebraic Riccati equation (1.46)

$$R_e = CPC' + R \tag{1.56a}$$
$$K_p = (APC' + GS)R_e^{-1} \tag{1.56b}$$

The online computations of the one-step predictive Kalman filter reduces to

$$\hat{y}_{k|k-1} = C\hat{x}_{k|k-1} + f \tag{1.57a}$$
$$e_k = y_k - \hat{y}_{k|k-1} \tag{1.57b}$$
$$\hat{x}_{k+1|k} = A\hat{x}_{k|k-1} + Bu_k + d + K_pe_k \tag{1.57c}$$

and the resulting filter is guaranteed to be stable as $A - K_pC$ is stable.

1.8.2 Measurement-Update Time-Update Kalman Filter

The measurement-update time-update Kalman filter recursion in the linear time invariant case are the same as in the linear time variant case, except that the matrices $(A_k, B_k, G_k, d_k, C_k, f_k, Q_k, S_k, R_k)$ are all time invariant.

The filter gain for the states, $K_{fx,k}$, and the filter gain for the process noise, $K_{fw,k}$, are computed by

$$R_{e,k} = CP_{k|k-1}C' + R \tag{1.58a}$$
$$K_{fx,k} = P_{k|k-1}C'R_{e,k}^{-1} \tag{1.58b}$$
$$K_{fw,k} = SR_{e,k}^{-1} \tag{1.58c}$$

The filtered state estimate, $\hat{x}_{k|k}$, and the filtered process noise estimate, $\hat{w}_{k|k}$ are computed using the recursion

$$\hat{y}_{k|k-1} = C\hat{x}_{k|k-1} + f \tag{1.59a}$$
$$e_k = y_k - \hat{y}_{k|k-1} \tag{1.59b}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_{fx,k}e_k \tag{1.59c}$$
$$\hat{w}_{k|k} = K_{fw,k}e_k \tag{1.59d}$$

The one-step prediction estimates of the states, $\hat{x}_{k+1|k}$, is

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k + G\hat{w}_{k|k} + d \tag{1.60}$$

The covariance associated with the filtered state, $P_{k|k}$, the filtered process noise, $Q_{k|k}$, and the one step predicted state, $P_{k+1|k}$, are computed by

$$P_{k|k} = P_{k|k-1} - K_{fx,k}R_{e,k}K'_{fx,k} \tag{1.61a}$$
$$Q_{k|k} = Q_k - K_{fw,k}R_{e,k}K'_{fw,k} \tag{1.61b}$$
$$P_{k+1|k} = AP_{k|k}A' + GQ_{k|k}G' - AK_{fx,k}S'G' - GSK'_{fx,k}A' \tag{1.61c}$$

The relevant conditional variables have the distributions

$$\boldsymbol{x}_k|\mathcal{Y}_k \sim N(\hat{x}_{k|k}, P_{k|k}) \tag{1.62a}$$
$$\boldsymbol{w}_k|\mathcal{Y}_k \sim N(\hat{w}_{k|k}, Q_{k|k}) \tag{1.62b}$$
$$\boldsymbol{x}_{k+1}|\mathcal{Y}_k \sim N(\hat{x}_{k+1|k}, P_{k+1|k}) \tag{1.62c}$$

which may be used in computing confidence intervals for the corresponding estimates.

1.8.2.1 Stationary Case

The recursions for the linear time invariant measurement-update time-update Kalman filter converges to a stationary solution provided the conditions in proposition 1.8.2 are satisfied.

Use the notation

$$\lim_{k\to\infty} P_{k|k-1} = P \tag{1.63a}$$
$$\lim_{k\to\infty} R_{e,k} = R_e \tag{1.63b}$$
$$\lim_{k\to\infty} K_{fx,k} = K_{fx} \tag{1.63c}$$
$$\lim_{k\to\infty} K_{fw,k} = K_{fw} \tag{1.63d}$$
$$\lim_{k\to\infty} P_{k|k} = P_f \tag{1.63e}$$
$$\lim_{k\to\infty} Q_{k|k} = Q_f \tag{1.63f}$$

P is obtained as the solution of the discrete algebraic Riccati equation

$$P = APA' + GQG' - (APC' + GS)(CPC' + R)^{-1}(APC' + GS)' \tag{1.64}$$

which may be used in computing the stationary state and process noise filter gains

$$R_e = CPC' + R \tag{1.65a}$$
$$K_{fx} = PC'R_e^{-1} \tag{1.65b}$$
$$K_{fw} = SR_e^{-1} \tag{1.65c}$$

as well as the covariances, P_f and Q_f, associated with the filtered estimates

$$P_f = P - K_{fx}R_eK'_{fx} \tag{1.66a}$$
$$Q_f = Q - K_{fw}R_eK'_{fw} \tag{1.66b}$$

These matrices may be computed off-line.

The online computations reduce to the state and process noise filter equations

$$\hat{y}_{k|k-1} = C\hat{x}_{k|k-1} + f \tag{1.67a}$$
$$e_k = y_k - \hat{y}_{k|k-1} \tag{1.67b}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_{fx}e_k \tag{1.67c}$$
$$\hat{w}_{k|k} = K_{fw}e_k \tag{1.67d}$$

as well as the equation for the one-step prediction of the states

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k + G\hat{w}_{k|k} + d \tag{1.68}$$

Confidence intervals for the estimates may be computing knowing that the associated conditional variables have the distributions

$$\boldsymbol{x}_k|\mathcal{Y}_k \sim N(\hat{x}_{k|k}, P_f) \tag{1.69a}$$
$$\boldsymbol{w}_k|\mathcal{Y}_k \sim N(\hat{w}_{k|k}, Q_f) \tag{1.69b}$$
$$\boldsymbol{x}_{k+1}|\mathcal{Y}_k \sim N(\hat{x}_{k+1|k}, P) \tag{1.69c}$$

1.8.3 One-Step Predictive Kalman Filter

The recursions for the one-step predictive Kalman filter in the linear time invariant case are stated. They are identical to the corresponding recursions in the time variant case, except that the model matrices are time invariant.

The predictive filter gain, $K_{p,k}$, is computed by

$$R_{e,k} = CP_{k|k-1}C' + R \tag{1.70a}$$
$$K_{p,k} = (AP_{k|k-1}C' + GS)R_{e,k}^{-1} \tag{1.70b}$$

This gain is used in computing the filtered one-step prediction of the states

$$\hat{y}_{k|k-1} = C\hat{x}_{k|k-1} + f \tag{1.71a}$$
$$e_k = y_k - \hat{y}_{k|k-1} \tag{1.71b}$$
$$\hat{x}_{k+1|k} = A\hat{x}_{k|k-1} + Bu_k + d + K_{p,k}e_k \tag{1.71c}$$

The covariance, $P_{k+1|k}$, associated to the one step prediction is computed by

$$P_{k+1|k} = AP_{k|k-1}A' + GQ_kG' - K_{p,k}R_{e,k}K'_{p,k} \tag{1.72}$$

The distribution of the associated conditional stochastic variable

$$\boldsymbol{x}_{k+1}|\mathcal{Y}_k \sim N(\hat{x}_{k+1|k}, P_{k+1|k}) \tag{1.73}$$

may be used in computing a confidence interval for $\hat{x}_{k+1|k}$.

1.8.3.1 Stationary One-Step Predicitive Kalman Filter

The recursions for the linear time invariant one-step predictive Kalman filter converges to a stationary solution provided the conditions in proposition 1.8.2 are satisfied.

Use the notation

$$\lim_{k\to\infty} P_{k|k-1} = P \tag{1.74a}$$

$$\lim_{k\to\infty} R_{e,k} = R_e \tag{1.74b}$$

$$\lim_{k\to\infty} K_{p,k} = K_p \tag{1.74c}$$

P is obtained as the solution of the discrete algebraic Riccati equation

$$P = APA' + GQG' - (APC' + GS)(CPC' + R)^{-1}(APC' + GS)' \tag{1.75}$$

which may be used in computation of

$$R_e = CPC' + R \tag{1.76a}$$

$$K_p = (APC' + GS)R_e^{-1} \tag{1.76b}$$

The matrices P, R_e, and K_p may be computed off-line.

The online computations in the stationary predictive Kalman filter reduces to

$$\hat{y}_{k|k-1} = C\hat{x}_{k|k-1} + f \tag{1.77a}$$

$$e_k = y_k - \hat{y}_{k|k-1} \tag{1.77b}$$

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k-1} + Bu_k + d + K_pe_k \tag{1.77c}$$

The distribution

$$\boldsymbol{x}_{k+1}|\mathcal{Y}_k \sim N(\hat{x}_{k+1|k}, P) \tag{1.78}$$

may be used in computing confidence intervals for $\hat{x}_{k+1|k}$.

1.8.4 State Predictions

In the linear time invariant case the states are predicted by

$$\hat{x}_{k+1+j|k} = A\hat{x}_{k+j|k} + B\hat{u}_{k+j|k} + d \qquad j > 0 \tag{1.79}$$

and the associtated covariance computed by

$$P_{k+1+j|k} = AP_{k+j|k}A' + GQG' \qquad j > 0 \tag{1.80}$$

The fact that $\boldsymbol{x}_{k+j}|\mathcal{Y}_k$ has the distribution

$$\boldsymbol{x}_{k+j}|\mathcal{Y}_k \sim N(\hat{x}_{k+j|k}, P_{k+j|k}) \tag{1.81}$$

may be used to compute confidence intervals for the estimate, $\hat{x}_{k+j|k}$.

1.8.4.1 Stationary State Predictions

In the case when A is stable, the recursions for the covariance associated with the predictions converge

$$\lim_{j\to\infty} P_{k+j|k} = P_{ol} \tag{1.82}$$

and P_{ol} may be computed as the solution of the discrete-time Lyapunov equation

$$P_{ol} = AP_{ol}A' + GQG' = AP_{ol}A' + \bar{Q} \tag{1.83}$$

in which we have defined $\bar{Q} = GQG'$.

The associated estimates are

$$\hat{x}_{k+1+j|k} = A\hat{x}_{k+j|k} + B\hat{u}_{k+j|k} + d \qquad j > 0 \tag{1.84}$$

The fact that $\boldsymbol{x}_{k+j}|\mathcal{Y}_k$ has the distribution

$$\boldsymbol{x}_{k+j}|\mathcal{Y}_k \sim N(\hat{x}_{k+j|k}, P_{ol}) \qquad j \to \infty \quad (A \text{ stable}) \tag{1.85}$$

in the limit $j \to \infty$ may be used to compute confidence intervals for the estimate, $\hat{x}_{k+j|k}$.

1.8.5 Measurement Prediction

It is obvious that $\hat{y}_{k|k} = y_k$ with zero covariance.

For $j > 0$ the predicted measurements are

$$\hat{y}_{k+j|k} = C\hat{x}_{k+j|k} + f \qquad j > 0 \tag{1.86}$$

and the associated covariance is

$$\Theta_{k+j|k} = CP_{k+j|k}C' + R \qquad j > 0 \tag{1.87}$$

The conditional variable $\boldsymbol{y}_{k+j}|\mathcal{Y}_k$ has the distribution

$$\boldsymbol{y}_{k+j}|\mathcal{Y}_k \sim N(\hat{y}_{k+j|k}, \Theta_{k+j|k}) \qquad j > 0 \tag{1.88}$$

This property may be used in computing confidence intervals for the prediction.

1.8.5.1 Stationary Measurement Prediction

When A is stable, the covariance $\Theta_{k+j|k}$ converges to

$$\lim_{j\to\infty} \Theta_{k+j|k} = \Theta_{ol} \tag{1.89}$$

with

$$\Theta_{ol} = CP_{ol}C' + R \tag{1.90}$$

The predicted measurements are

$$\hat{y}_{k+j|k} = C\hat{x}_{k+j|k} + f \qquad j > 0 \tag{1.91}$$

and Θ_{ol} may be used for computing confidence intervals of $\hat{u}_{i+j|k}$ in the limit $j \to \infty$ as

$$\boldsymbol{y}_{k+j}|\mathcal{Y}_k \sim N(\hat{y}_{k+j|k}, \Theta_{ol}) \qquad j \to \infty \quad (A \text{ stable}) \tag{1.92}$$

1.8.6 Output Prediction

The predicted (and filtered) output $\hat{z}_{k+j|k}$ for $j \geq 0$ may be computed by

$$\hat{z}_{k+j|k} = H\hat{x}_{k+j|k} + h \qquad j \geq 0 \tag{1.93}$$

The associated covariance may be computed by

$$\Xi_{k+j|k} = HP_{k+j|k}H' \qquad j \geq 0 \tag{1.94}$$

The conditional variable $\boldsymbol{z}_{k+j}|\mathcal{Y}_k$ has the distribution

$$\boldsymbol{z}_{k+j}|\mathcal{Y}_k \sim N(\hat{z}_{k+j|k}, \Xi_{k+j|k}) \qquad j \geq 0 \tag{1.95}$$

This property may be used for computing confidence intervals for the prediction, $\hat{z}_{k+j|k}$ for $j \geq 0$.

1.8.6.1 Stationary Output Prediction

In the case when A is stable, the limit

$$\lim_{j\to\infty} \Xi_{k+j|k} = \Xi_{ol} \tag{1.96}$$

exists and may be computed by

$$\Xi_{ol} = HP_{ol}H' \tag{1.97}$$

The predicted (and filtered) output $\hat{z}_{k+j|k}$ for $j \geq 0$ are computed by

$$\hat{z}_{k+j|k} = H\hat{x}_{k+j|k} + h \qquad j \geq 0 \tag{1.98}$$

in which the recursion for computation of $\hat{x}_{k+j|k}$ is utilized.
In the limit $j \to \infty$, the conditional variable $\boldsymbol{z}_{k+j}|\mathcal{Y}_k$ has the distribution

$$\boldsymbol{z}_{k+j}|\mathcal{Y}_k \sim N(\hat{z}_{k+j|k}, \Xi_{ol}) \qquad j \to \infty \qquad (A \text{ stable}) \tag{1.99}$$

This property may be used for computing confidence intervals for the prediction, $\hat{z}_{k+j|k}$ for $j \to \infty$.

1.8.7 Smoothing

The smoothing algorithm described is the Bryson-Frazier smoothing algorithm.
The boundary conditions used by the algorithm are the initial values $\hat{x}_{0|-1}$ and $P_{0|-1}$ given by the model specification as well as

$$\lambda_{N+1|N} = 0 \tag{1.100a}$$
$$\Lambda_{N+1|N} = 0 \tag{1.100b}$$

For $k = 0, 1, \ldots, N$ compute

$$R_{e,k} = CP_{k|k-1}C' + R \tag{1.101a}$$
$$K_{p,k} = (AP_{k|k-1}C' + GS)R_{e,k}^{-1} \tag{1.101b}$$
$$P_{k+1|k} = AP_{k|k-1}A' + GQG' - K_{p,k}R_{e,k}K_{p,k}' \tag{1.101c}$$
$$\hat{y}_{k|k-1} = C\hat{x}_{k|k-1} + f \tag{1.101d}$$
$$e_k = y_k - \hat{y}_{k|k-1} \tag{1.101e}$$
$$\hat{x}_{k+1|k} = A\hat{x}_{k|k-1} + Bu_k + d + K_{p,k}e_k \tag{1.101f}$$

and subsequently the smoothed states $\left\{\hat{x}_{k|N}\right\}_{k=0}^{N}$ are computed by the backward recursion for $k = N, N-1, \ldots, 0$

$$\lambda_{k|N} = (A - K_{p,k}C)'\lambda_{k+1|N} + C'R_{e,k}^{-1}e_k \tag{1.102a}$$
$$\hat{x}_{k|N} = \hat{x}_{k|k-1} + P_{k|k-1}\lambda_{k|N} \tag{1.102b}$$

If the smoothed states and their covariance is desired the following backward recursions are applied for $k = N, N+1, \ldots, 0$

$$F_k = A - K_{p,k}C \tag{1.103a}$$
$$\lambda_{k|N} = F_k'\lambda_{k+1|N} + C'R_{e,k}^{-1}e_k \tag{1.103b}$$
$$\Lambda_{k|N} = F_k'\Lambda_{k+1|N}F_k + C'R_{e,k}^{-1}C \tag{1.103c}$$
$$\hat{x}_{k|N} = \hat{x}_{k|k-1} + P_{k|k-1}\lambda_{k|N} \tag{1.103d}$$
$$P_{k|N} = P_{k|k-1} - P_{k|k-1}\Lambda_{k|N}P_{k|k-1} \tag{1.103e}$$

The smoothed process noise, $\left\{\hat{w}_{k|N}\right\}_{k=0}^{N}$, and its covariance, $\left\{Q_{k|N}\right\}_{k=0}^{N}$, may be computed by doing the following computations in addition to (1.103) at each time instant k

$$E_k = QG' - SK_{p,k}' \tag{1.104a}$$
$$\hat{w}_{k|N} = E_k\lambda_{k+1|N} + SR_{e,k}^{-1}e_k \tag{1.104b}$$
$$Q_{k|N} = Q - SR_{e,k}^{-1}S' - E_k\Lambda_{k+1|N}E_k' \tag{1.104c}$$

The conditional stochastic variable $\boldsymbol{x}_k|\mathcal{Y}_N$ has the distribution

$$\boldsymbol{x}_k|\mathcal{Y}_N \sim N(\hat{x}_{k|N}, P_{k|N}) \tag{1.105}$$

This information about the distribution, may be used in computation of confidence interval for the smoothed states, $\hat{x}_{k|N}$.

The smoothed output variable estimate, $\hat{z}_{k|N}$, and the associated covariance, $\Xi_{k|N}$, may be computed by

$$\hat{z}_{k|N} = H\hat{x}_{k|N} + h \tag{1.106a}$$
$$\Xi_{k|N} = HP_{k|N}H' \tag{1.106b}$$

$\boldsymbol{z}_k|\mathcal{Y}_N$ has the distribution

$$\boldsymbol{z}_k|\mathcal{Y}_N \sim N(\hat{z}_{k|N}, \Xi_{k|N}) \tag{1.107}$$

which may be used in the computation of confidence intervals for the smoothed output estimate, $\hat{z}_{k|N}$.

1.8.7.1 Stationary Smoothing

The recursions for the linear time invariant smoother converges to a stationary solution provided the conditions in proposition 1.8.2 are satisfied.

Use the notation

$$\lim_{k\to\infty} P_{k|k-1} = P \tag{1.108a}$$
$$\lim_{k\to\infty} R_{e,k} = R_e = CPC' + R \tag{1.108b}$$
$$\lim_{k\to\infty} K_{p,k} = K_p = (APC' + GS)R_e^{-1} \tag{1.108c}$$
$$\lim_{k\to\infty} F_k = F = A - K_pC \tag{1.108d}$$
$$\lim_{N\to\infty} \Lambda_{k|N} = \Lambda \qquad k \ll N \tag{1.108e}$$
$$\lim_{N\to\infty} P_{k|N} = P_s = P - P\Lambda P \qquad k \ll N \tag{1.108f}$$
$$\lim_{k\to\infty} E_k = QG' - SK_p \tag{1.108g}$$
$$\lim_{N\to\infty} Q_{k|N} = Q_s = Q - SR_e^{-1}S' - E\Lambda E' \qquad k \ll N \tag{1.108h}$$

P is obtained as the solution of the discrete algebraic Riccati equation

$$P = APA' + GQG' - (APC' + GS)(CPC' + R)^{-1}(APC' + GS)' \tag{1.109}$$

Λ is obtained as the limit $N \to \infty$ and $k = N - j$ for $j \to \infty$, i.e. $k \ll N$, of

$$\Lambda_{k|N} = F_k'\Lambda_{k+1|N}F_k + C'R_{e,k}^{-1}C \tag{1.110}$$

By construction $F = A - K_pC$ is stable. Hence Λ may be obtained as the solution to the discrete Lyapunov equation

$$\Lambda = F'\Lambda F + C'R_e^{-1}C \tag{1.111}$$

Define

$$K_{s\lambda} = CR_e^{-1} \tag{1.112a}$$
$$K_{sw} = SR_e^{-1} \tag{1.112b}$$

Then the forward recursion $k = 0, 1, ..., N$ becomes

$$\hat{y}_{k|k-1} = C\hat{x}_{k|k-1} + f \tag{1.113a}$$
$$e_k = y_k - \hat{y}_{k|k-1} \tag{1.113b}$$
$$\hat{x}_{k+1|k} = A\hat{x}_{k|k-1} + Bu_k + d + K_pe_k \tag{1.113c}$$

and the backward recursions $k = N, N-1, ..., 0$ with $\lambda_{N+1|N} = 0$ can be stated as

$$\lambda_{k|N} = F\lambda_{k+1|N} + K_{s\lambda}e_k \tag{1.114a}$$
$$\hat{x}_{k|N} = \hat{x}_{k|k-1} + P\lambda_{k|N} \tag{1.114b}$$

The smoothed process noise may be obtained as part of the backward recursion

$$\hat{w}_{k|N} = E\lambda_{k+1|N} + K_{sw}e_k \tag{1.115}$$

The distributions

$$\boldsymbol{x}_{k|N} \sim N(\hat{x}_{k|N}, P_s) \tag{1.116a}$$
$$\boldsymbol{w}_{k|N} \sim N(\hat{w}_{k|N}, Q_s) \tag{1.116b}$$

may be used for computation of confidence intervals of the smoothed estimates.

1.9 Disturbance Models and Offset Free Estimation

In the presence of unmeasured disturbances or model-plant mismatch some kind of disturbance model is needed to guarantee offset free control by the predictive controller. This topic has been discussed by Muske and Badgwell (2002), Pannocchia and Rawlings (2003), and **?**).

The methodologies provided are almost identical. They both require that the original system is augmented with as many states as there is measured variables and that the augmented system is detectable. In addition to detectability it is a requirement that the original system is stabilizable.

For classic PID-control, it is well known that P-control is not sufficient for steady state offset free control. PI-control is needed for that. In the same way, model predictive control with just the nominal model is insufficient for steady offset free control in the case of unmeasured disturbances or any other model-plant mismatch.

In this section different disturbance models are discussed. In addition, *sufficient* conditions for detectability of the augmented system and offset free control of the combined estimator and regulator are listed. The closed-loop controller performance is directly related to how accurately the disturbance model represents the actual disturbance entering the process (Francis and Wonham, 1976).

This section provides tests for deciding whether a given disturbance model is sufficient for offset free predictive control as well as guidelines for the design of such disturbance models. However, it does provide constructive methods for the synthesis of these disturbance models. This construction is an open issue and need to be addressed for easing the use of the linear model predictive control toolbox.

1.9.1 Structured Disturbance Model

Muske and Badgwell (2002) discuss the structured disturbance models and design guidelines to assure offset free predictive control. They consider an aug-

mented model with the structure

$$\begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{s}_{k+1} \\ \boldsymbol{q}_{k+1} \end{bmatrix} = \begin{bmatrix} A & G_s & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{s}_k \\ \boldsymbol{q}_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} u_k + \begin{bmatrix} G & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_k \\ \boldsymbol{\eta}_k \\ \boldsymbol{\zeta}_k \end{bmatrix} + \begin{bmatrix} d_k \\ 0 \\ 0 \end{bmatrix} \tag{1.117a}$$

$$\boldsymbol{y}_k = \begin{bmatrix} C & 0 & G_q \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{s}_k \\ \boldsymbol{q}_k \end{bmatrix} + \boldsymbol{v}_k + f_k \tag{1.117b}$$

in which s_k is state disturbances and q_k are output disturbances. They assume that the objective is to control the measured variables to some target. Their presentation of the topic, relies on the implicit assumption that the number of manipulated variables equals the number of measured variables. Hence, the treatment replies to the square case.

It is assumed that (C, A) is detectable, and in order to have offset free control that (A, B) is stabilizable.

Offset free estimation requires that the augmented model is detectable, i.e. that

$$\text{rank} \begin{bmatrix} \lambda I - A & -G_s & 0 \\ 0 & \lambda I - I & 0 \\ 0 & 0 & \lambda I - I \\ C & 0 & G_q \end{bmatrix} = n + n_s + n_q \quad \forall \lambda = \text{eig} \left(\begin{bmatrix} A & G_s & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \right) : |\lambda| \geq 1 \tag{1.118}$$

The main results used to assert that a given disturbance model leads to an augmented model that is detectable and that such a model may be used to obtain offset free control are

1. The augmented system is detectable *if and only if*
   1) The original system (C, A) is detectable
   2) $\text{rank} \begin{bmatrix} I - A & -G_s & 0 \\ C & 0 & G_q \end{bmatrix} = n + n_s + n_q$
2. There exists a detectable augmented system with $n_s + n_q = p$ for all detectable (C, A), i.e. if the original system is detectable then it is always possible to construct a disturbance model of dimension p such that the augmented system is detectable.

These two properties imply

1. G_d and G_p must both have full column rank.
2. The total number of augmented disturbance states must not exceed the number of outputs.
3. The number of augmented state disturbances must not exceed the number of linearly independent outputs.
4. $\mathcal{R}(G_d)$ (the range of G_d) must not contain any unobservable modes of A.

5. The default output disturbance model $G_p = I$ is not suitable for integrating systems.

For systems **without integrating modes** we have

1. The output disturbance model, $G_p = I$ and $G_d = []$, leads to a detectable augmented system.

2. If the outputs of the original system model are linearly independent, a state disturbance model will be detectable.

1.9.1.1 Offset Free Predictive Control

Muske and Badgwell (2002) show that it is sufficient for offset free predictive control that the following conditions are satisfied

1. The closed loop system is asymptotically stable.

2. (A, B) is stabilizable and (C, A) is detectable.

3. The total number of augmented integrating states is equal to the number of measurements, $n_s + n_q = p$.

4. The augmented system is detectable (the conditions for this are discussed above).

5. No inequality constraints are strictly active at steady state.

In practice it can be tested whether a given disturbance model will give offset free control by testing whether the augmented system matrix is detectable.

1.9.2 Unstructured Disturbance Model

The structured disturbance model has been analyzed for square systems. It may probably be extended to non-square systems as well but yet no analytic results for such cases.

Pannocchia and Rawlings (2003) analyzed a non-structured disturbance model and its ability to give offset free control. Their analysis applied to square as well as non-square systems. The key result is: *A number of integrating disturbances equal to the number of measured variables are sufficient to guarantee offset free predictive control.* This result apply to square and non-square systems. In particular, it should be noted that if the number of measured variables, p, is greater than the number of manimpulated variables, m, then it is *not* sufficient to have only m integrators to have offset free control in m controlled variables. The controlled variables must be in the space spanned by the measurement equations.

The augmented model studied by Pannocchia and Rawlings (2003) is

$$\begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{s}_{k+1} \end{bmatrix} = \begin{bmatrix} A & B_s \\ 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{s}_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_k \\ \boldsymbol{\eta}_k \end{bmatrix} + \begin{bmatrix} d_k \\ 0 \end{bmatrix} \tag{1.119a}$$

$$\boldsymbol{y}_k = \begin{bmatrix} C & C_s \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{s}_k \end{bmatrix} + \boldsymbol{v}_k + f_k \tag{1.119b}$$

in which $\boldsymbol{s}_k \in R^{n_s}$ is the integrating disturbance states.

To be able to estimate the states of the model without error asymptotically, i.e. offset free estimation, the system must be detectable. In general detectability of the augmented system may be checked by the condition

$$\operatorname{rank} \begin{bmatrix} \lambda I - A & -B_s \\ 0 & \lambda I - I \\ C & C_s \end{bmatrix} = n + n_s \quad \forall \lambda = \operatorname{eig} \left(\begin{bmatrix} A & B_s \\ 0 & I \end{bmatrix} \right) : |\lambda| \geq 1 \tag{1.120}$$

Using the structure of the augmented system, this condition can be reformulated; the augmented system is detectable if and only if the following two conditions are satisfied

1. The non-augmented system is detectable, i.e. (C, A) is detectable.
2. $\operatorname{rank} \begin{bmatrix} I - A & -B_s \\ C & C_s \end{bmatrix} = n + n_s$

1.9.2.1 Offset Free Predictive Control

The variables to be controlled are a subset of the measurement noise-free measurements, i.e.

$$\boldsymbol{z}_k = H \left(\begin{bmatrix} C & C_s \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{s}_k \end{bmatrix} + f_k \right) \tag{1.121}$$

Let $\boldsymbol{z}_k \in \mathbb{R}^{n_c}$. Obviously, the number of controlled objective, n_c, has to be less or equal to the number of manipulated variables, m, in order to obtain offset free control. A necessary condition for offset free control can be stated as

$$\operatorname{rank} \begin{bmatrix} I - A & -B \\ HC & 0 \end{bmatrix} = n + n_c \tag{1.122}$$

Consequently, assume that (1.122) is satisfied. This assumption implies

1. We are able to compensate for the steady state disturbances in the controlled variables, z_k, i.e. to keep the controlled variables at target if the disturbances are known.
2. The number of controlled variables cannot exceed either the number of manipulated variables, m, or the number of measurements, p.
3. H has to be full row rank, i.e. the controlled variables must be independent of each other.

The sufficient conditions for offset free predictive control using the unstructured disturbance model can be stated as

1. rank $\begin{bmatrix} I-A & -B \\ HC & 0 \end{bmatrix} = n + n_c$, in which n_c is the number of controlled variables and n is the state dimension of the original system.
2. Augment the model with a number integrating disturbances equal to the number of measurements ($n_s = p$).
3. Choose B_s and C_s such that rank $\begin{bmatrix} I-A & -B_s \\ C & C_s \end{bmatrix} = n + p$.
4. If the closed loop system, i.e. the system consisting of estimator-regulator-plant, is stable, and constraints are not strictly active at steady state, there is zero offset in the controlled variables, $\boldsymbol{z}_k$.

In the special case that the system has no integrators, an output disturbance model, i.e. $B_s = 0$ and $C_s = I$, will provide offset free control.

1.9.2.2 Relation to the Structured Disturbance Model

The unstructured disturbance model of Pannocchia and Rawlings (2003) contains the structured disturbance model of Muske and Badgwell (2002) as a special case. The matrices in the unstructured disturbance model are related to the matrices in the structured disturbance model by

$$B_s = \begin{bmatrix} G_s & 0 \end{bmatrix} \qquad C_s = \begin{bmatrix} 0 & G_q \end{bmatrix} \tag{1.123}$$

1.9.2.3 Construction of a Detectable Disturbance Model

One way to construct an unstructured disturbance model that is guaranteed to detectable is:

1. QR-factorize $\begin{bmatrix} I-A \\ C \end{bmatrix}$: $\begin{bmatrix} I-A \\ C \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}$
2. Set $\begin{bmatrix} -B_s \\ C_s \end{bmatrix} = Q_2$, i.e. $B_s = -Q_2(1:n,:)$ and $C_s = Q_2(n+1:n+p,:)$.

The model obtained in this way is guaranteed to be detectable. However, for a given process it may not reflect the actual disturbances in the best possible way. This is an identification issue.

1.9.3 Missing Observations

When a measurement fall out temporarily, there seems to be no reason to adjust the disturbance model; even though the augmented system is in principle non-detectable while the measurement is not available.

An open issue for the disturbance model is how to proceed when measurements fall out permanently. In the case when the original model has no integrators, and an output disturbance model is applied, an intuitive approach is to switch off the integrating disturbance states associated with the missing measurement.

1.10 Online and Laboratory Measurements

A frequent situation occurring for industrial chemical processes, is the situation in which online measurements are available to the control system without delay at a high and regular frequency, while some key variables are measured in a laboratory and only available to the control system with a long delay and at a low and irregular frequency.

Assume that the sample time of the control system is T_s, and that the online measurements (possibly filtered) are in principle available with intervals of T_s. If some sensor falls out, this can be handled by the Kalman filter with time variant models as well by adjusting the appropriate measurement matrices. The laboratory measurements are available at a much lower frequency. There is a long delay from the time when the process is sampled to the time when the result of the laboratory analysis is available to the control system. For simplicity we assume that the samples for laboratory analysis is *taken from the process* at integer multiples of the control system sample time, T_s. If this is not the case, the time when the sample is set to the closest integer multiple of the control cycle time.

In the control system, we operate with an estimation window of N_e sample times. N_e must be selected large enough.

At time

$$kT_s \leq t < (k+1)T_s \tag{1.124}$$

when measurement y_k has been processed, the memory of the estimator is

$$(\hat{x}_{k+1|k}, P_{k+1|k}) \tag{1.125a}$$

$$(\hat{x}_{k+1-N_e|k-N_e}, P_{k+1-N_e|k-N_e}) \tag{1.125b}$$

which is the current one step prediction, and the one step prediction N_e samples in the past.

Assume that a sample is taken for laboratory analysis at time $t = k_{lab}T_s$ and that the current time, t, at which the result of the laboratory analysis is entered into the control system is $kT_s \leq t < (k+1)T_s$. Assume that $k - N_e + 1 \leq k_{lab} \leq k$, i.e. that the sample is taken in the estimation window of interest; if this is not case discard the value as it is too old to provide any useful online information about the current state of the process. When $k - N_e + 1 \leq k_{lab} \leq k$, the measurement vector, $y_{k_{lab}}$, is supplied with the missing value for the laboratory measurement (and an appropriate flag is set to indicate that the value is available). When this has been done, the one-step predictive Kalman filter with time variant models is run from time $k - N_e + 1$ to time k with the measurements $\{y_i\}_{i=k-N_e+1}^{k}$ provided. In this way the current estimate $\hat{x}_{k+1|k}$ and its associated covariance $P_{k+1|k}$ are updated such that the laboratory measurement is taken into account.

This procedure requires that historical measurements $\{y_i\}_{i=k-N_e+1}^{k}$ and manipulated variables $\{u_i\}_{i=k-N_e+1}^{k}$ are stored and available to the control system, when the current one-step predictions are to be updated incorporating laboratory measurements.

1.11 Matlab Implementation

1.11.1 Stationary Estimators for LTI Systems

```
% STATIONARY ESTIMATORS FOR LINEAR TIME INVARIANT SYSTEMS
% ------------------------------------------------------------------------
%   DesignStationaryLTIKF                    Computes stationary gain and covariance matrices
%   MeasurementUpdateStationaryLTIKF        Measurement update filter estimates
%   TimeUpdateStationaryLTIKF               Time update one-step predictive estimates
%   StatePredictionStationaryLTIKF          State predictions
%   MeasurementPredictionStationaryLTIKF    Measurement predictions
%   OutputPredictionStationaryLTIKF         Output predictions
%
%   DesignStationaryLTIPKF                  Computes stationary gain and covariance matrix
%   MeasurementTimeUpdateStationarLTIPKF    Measurement and time update in predictive Kalman filter
%   StatePredictionStationaryLTIPKF         State predictions based on one-step prediction
%   MeasurementPredictionStationaryLTIPKF   Measurement predictions based on one-step prediction
%   OutputPredictionStationaryLTIPKF        Output predictions based on one-step prediction
%
%   DesignSmoothStationaryLTIPKF            Computes gain matrices for a stationary smoother
%   StateSmoothStationaryLTIPKF             Smoothed state estimates in LTI model
%   OutputSmoothStationaryLTIPKF            Smoothed output estimates in LTI model
%
%   StationaryStatePredictionCovariance         Computes the stationary state prediction covariance
%   StationaryOutputPredictionCovariance        Computes the stationary output prediction covariance
%   StationaryMeasurementPredictionCovariance   Computes the stationary measurement prediction covariance
```

1.11.1.1 DesignStationaryLTIKF

```
%
% DesignStationaryLTIKF     Computes the gain and covariance matrices for a
%                           stationary measurement-update time-update
%                           Kalman filter
%
%
%                           LTI system considered
%
%                              x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                              y[k]   = C*x[k] + v[k] + f
%
%                              <w[k],w[k]> = Q, <v[k],v[k]> = R, <w[k],v[k]>=S
%
%                           The gain matrices are computed for a stationary
%                           meausement update time update Kalman filter.
%
%
% Syntax:   [Kfx,Kfw,Pf,Qf,P,Re]=DesignStationaryLTIKF(A,B,G,C,Q,S,R)
%
%                   A,B,G   :   Matrices in state transition equation of
%                               the LTI system
%                   C       :   Matrix in the measurement equation
%                   Q,S,R   :   Covariance matrices
%
%                   Kfx     :   Gain matrix for the filtered state estimate
%                   Kfw     :   Gain matrix for the filtered process noise
%                               estimate
%                   Pf      :   Covariance matrix associated with the
%                               filtered state estimate
%                   Qf      :   Covariance matrix associated with the
%                               filtered process noise estimate
%                   P       :   Covariance matrix associated with the
%                               one-step state prediction
%                   Re      :   Covariance matrix associated with the
%                               innovations
```

1.11.1.2 MeasurementUpdateStationaryLTIKF

```
%
% MeasurementUpdateStationaryLTIKF  Computes the filtered state and process
%                                   noise estimate using a stationary measurement update
%                                   time update Kalman filter
%
%                                   LTI system considerd
%
%                                      x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                                      y[k]   = C*x[k] + v[k] + f
%
%                                     <w[k],w[k]> = Q, <v[k],v[k]> = R, <w[k],v[k]>=S
%
%
%
%
% Syntax:   [xf,wf]=MeasurementUpdateStationaryLTIKF(y,C,f,xhat,Kfx,Kfw)
%
```

```
%               y       :   Measurement, y = y[k]
%               C,f     :   Matrices in measurement equation
%               xhat    :   One step prediction, xhat = x[k|k-1]
%               Kfx     :   Kalman filter gain for the filtered state
%                           estimate
%               Kfw     :   Kalman filter for the filtered process noise
%                           estimate
%
%               xf      :   Filtered state estimate, xf = x[k|k]
%               wf      :   Filtered process noise estimate, wf = w[k|k]
```

1.11.1.3 TimeUpdateStationaryLTIKF

```
%
% TimeUpdateStationaryLTIKF One-step state prediction with a stationary
%                           measurement update time update Kalman filter
%                           in an LTI system.
%
%                           LTI system considerd
%
%                               x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                               y[k]   = C*x[k] + v[k] + f
%
%                               <w[k],w[k]> = Q, <v[k],v[k]> = R, <w[k],v[k]>=S
%
%
%
% Syntax: xhat = TimeUpdateStationaryLTIKF(A,B,G,d,xf,u,wf)
%
%           A,B,G,d :   Matrices in
%           xf      :   Filtered states, xf = x[k|k]
%           u       :   Process input, u = u[k]
%           wf      :   Filtered process noise, wf = w[k|k]
%
%           xhat    :   One-step state prediction, xhat = x[k+1|k]
```

1.11.1.4 StatePredictionStationaryLTIKF

```
%
% StatePredictionStationaryLTIKF    State predictions using a stationary
%                                   measurement update time update Kalman filter
%                                   for an LTI system.
%
%                                   LTI system considerd
%
%                                    x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                                    y[k]   = C*x[k] + v[k] + f
%
%                                    <w[k],w[k]> = Q, <v[k],v[k]> = R, <w[k],v[k]>=S
%
%                                   The estmimated states x[k+j|k] are computed for
%                                   j=0,1, ..., Np
%
%
%
% Syntax: Xpred = StatePredictionStationaryLTIKF(A,B,G,d,xf,wf,Upred,Np)
%
%               A,B,G,d     :   Matrices in LTI state transition equation
%               xf          :   Filtered state estimate, xf = x[k|k]
%               wf          :   Filtered process noise estimate, wf = w[k|k]
%               Upred       :   Predicted process inputs, Upred(:,j+1) = u[k+j],
%                               j=0,1,...,Np-1
%               Np          :   Prediction horizon
%
%               Xpred       :   Array of predicted states
%                               Xpred(:,j+1) = x[k+j|k], j=0,1,...,Np
```

1.11.1.5 MeasurementPredictionStationaryLTIPKF

```
%
% MeasurementPredictionStationaryLTIKF  Measurement predictions using a
%                                       stationary measurement update time
%                                       update Kalman filter for an LTI
%                                       system.
%
%                                   LTI system considerd
%
%                                    x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                                    y[k]   = C*x[k] + v[k] + f
%
%                                    <w[k],w[k]> = Q, <v[k],v[k]> = R, <w[k],v[k]>=S
%
%                                   The estmimated measurements y[k+j|k] are computed for
%                                   j=0,1, ..., Np
```

```
%
%
%
% Syntax: Ypred = MeasurementPredictionStationaryLTIKF(A,B,G,d,C,f,xf,wf,Upred,y,Np)
%
%               A,B,G,d     :   Matrices in LTI state transition equation
%               C,f         :   Matrices in LTI measurement equation
%               xf          :   Filtered state estimate, xf = x[k|k]
%               wf          :   Filtered process noise estimate, wf = w[k|k]
%               Upred       :   Predicted process inputs, Upred(:,j+1) = u[k+j],
%                               j=0,1,...,Np-1
%               y           :   Measured value, y = y[k]
%               Np          :   Prediction horizon
%
%               Ypred       :   Array of predicted measurementes
%                               Ypred(:,j+1) = y[k+j|k], j=0,1,...,Np
```

1.11.1.6 OutputPredictionStationaryLTIKF

```
%
% OutputPredictionStationaryLTIKF   Output predictions using a
%                                   stationary measurement update time
%                                   update Kalman filter for an LTI system.
%
%                                   LTI system considerd
%
%                                    x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                                    y[k]   = C*x[k] + v[k] + f
%                                    z[k]   = H*x[k] + h
%
%                                    <w[k],w[k]> = Q, <v[k],v[k]> = R, <w[k],v[k]>=S
%
%                                   The estmimated measurements z[k+j|k] are computed for
%                                   j=0,1, ..., Np
%
%
%
% Syntax: Zpred = OutputPredictionStationaryLTIKF(A,B,G,d,H,h,xf,wf,Upred,Np)
%
%               A,B,G,d     :   Matrices in LTI state transition equation
%               C,f         :   Matrices in LTI measurement equation
%               xf          :   Filtered state estimate, xf = x[k|k]
%               wf          :   Filtered process noise estimate, wf = w[k|k]
%               Upred       :   Predicted process inputs, Upred(:,j+1) = u[k+j],
%                               j=0,1,...,Np-1
%               Np          :   Prediction horizon
%
%               Zpred       :   Array of predicted measurementes
%                               Zpred(:,j+1) = z[k+j|k], j=0,1,...,Np
```

1.11.1.7 DesignStationaryLTIPKF

```
%
% DesignStationaryLTIPKF    Computes the gain matrix and covariance
%                           matrices for a stationary linear time invariant
%                           predictive Kalman filter
%
%                           LTI system considered
%
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%
%                           <w[k],w[k]> = Q, <v[k],v[k]> = R, <w[k],v[k]> = S
%
%
%
%
% Syntax:  [Kp,P,Re]=DesignStationaryLTIPKF(A,B,G,C,Q,S,R)
%
%               A,B,G   :   Matrices in LTI state transition equation
%               C       :   Matrix in LTI measurement equation
%               Q,S,R   :   Covariance matrices in LTI system
%
%               Kp      :   Predictive Kalman gain
%               P       :   Covariance associated to one-step prediction
%               Re      :   Covariance of the innovations
```

1.11.1.8 MeasurementTimeUpdateStationaryLTIPKF

```
%
% MeasurementTimeUpdateStationaryLTIPKF
%                   Computes the combined measurement-time update for a
%                   stationary linear time invariant predictive Kalman
%                   filter.
```

```
%
%                   LTI system considered
%
%                   x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                   y[k]   = C*x[k] + v[k] + f
%
%                   <w[k],w[k]> = Q, <v[k],v[k]> = R, <w[k],v[k]> = S
%
%
%
% Syntax: xhatp = MeasurementTimeUpdateStationarLTIPKF(A,B,G,d,C,f,Kp,xhat,u,y)
%
%               A,B,G,d :   Matrices in LTI state transition equation
%               C,f     :   Matrices in LTI measurement equation
%               Kp      :   Stationary predictive Kalman gain
%                           (computed by DesignStationaryLTIPKF)
%               xhat    :   Current one-step state prediction, xhat = x[k|k-1]
%               u       :   Process input, u = u[k]
%               y       :   Measurement, y = y[k]
%
%               xhatp   :   One-step prediction, xhatp = x[k+1|k]
```

1.11.1.9 StatePredictionStationaryLTIPKF

```
%
% StatePredictionStationaryLTIPKF
%                       Computes state predictions for discrete-time k =
%                       1, 2, ..., Np in LTI systems. The estimates are
%                       based on the one-step predictive state estimate.
%
%                      The LTI model is
%
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%
%                      in which
%
%                           <w[k],w[k]> = Q
%                           <w[k],v[k]> = S
%                           <v[k],v[k]> = R
%
%                       The state estimate
%                       given the previous information Y = (y(0), y(1),
%                       ..., y(k)) is:
%
%                           xhat[k+1|k] = xhat
%
%                       The state estimates x[k+j|k] for j=1,...,Np
%                       are computed. It is assumed that u[k+j]=u[k+j|k],
%                       (j=1,2, ..., Np-1) which is specified in Upred.
%
%
% Syntax:   Xpred = StatePredictionStationaryLTIPKF(A,B,G,d,xhat,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           xhat      :   One-step predictive state estimate, xhat = x[k+1|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 1,2, ..., Np-1
%           Np      :     Prediction horizon
%
%           Xpred   :     Xpred(:,j+1) = x[k+j|k]   j=1, ..., Np
```

1.11.1.10 MeasurementPredictionStationaryLTIPKF

```
%
% MeasurementPredictionStationaryLTIPKF
%                               Computes measurement predictions for discrete-time k =
%                               1, 2, ..., Np in LTI systems. The estimates are
%                               based on the one-step predictive state estimate.
%
%
%                      The LTI model is
%
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%
%                      in which
%
%                           <w[k],w[k]> = Q
%                           <w[k],v[k]> = S
%                           <v[k],v[k]> = R
%
%                       The state estimate
%                       given the previous information Y = (y(0), y(1),
%                       ..., y(k)) is:
%
%                           xhat[k+1|k] = xhat
%
```

```
%                       The measurement estimates y[k+j|k] for j=1,...,Np
%                       are computed. It is assumed that u[k+j]=u[k+j|k],
%                       (j=1,2, ..., Np-1) which is specified in Upred.
%
%
% Syntax:   Ypred = MeausurementPredictionStationaryLTIPKF(A,B,G,d,C,f,xhat,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           C,f     :     LTI system matrices in measurement equation
%           xhat    :     One-step predictive state estimate, xhat = x[k+1|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 1,2, ..., Np-1
%           Np      :     Prediction horizon
%
%           Ypred   :     Ypred(:,j+1) = y[k+j|k]   j=1, ..., Np
```

1.11.1.11 OutputPredictionStationaryLTIPKF

```
%
% OutputPredictionStationaryLTIPKF
%                         Computes output predictions for discrete-time k =
%                         1, 2, ..., Np in LTI systems. The estimates are
%                         based on the one-step predictive state estimate.
%
%
%                       The LTI model is
%
%                            x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                            y[k]   = C*x[k] + v[k] + f
%                            z[k]   = H*x[k] + h
%
%                       in which
%
%                            <w[k],w[k]> = Q
%                            <w[k],v[k]> = S
%                            <v[k],v[k]> = R
%
%                        The state estimate
%                        given the previous information Y = (y(0), y(1),
%                        ..., y(k)) is:
%
%                            xhat[k+1|k] = xhat
%
%                        The output estimates z[k+j|k] for j=1,...,Np
%                        are computed. It is assumed that u[k+j]=u[k+j|k],
%                        (j=1,2, ..., Np-1) which is specified in Upred.
%
%
% Syntax:   Zpred = OutputPredictionStationaryLTIPKF(A,B,G,d,H,h,xhat,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           H,h     :     LTI system matrices in output equation
%           xhat    :     One-step predictive state estimate, xhat = x[k+1|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 1,2, ..., Np-1
%           Np      :     Prediction horizon
%
%           Zpred   :     Zpred(:,j+1) = z[k+j|k]   j=1, ..., Np
```

1.11.1.12 DesignSmoothStationaryLTIPKF

```
%
% DesignSmoothStationaryLTIPKF  Computes gain matrices for stationary
%                               smoothing of an LTI system.
%
%
%                       The LTI model is
%
%                            x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                            y[k]   = C*x[k] + v[k] + f
%                            z[k]   = H*x[k] + h
%
%                       in which
%
%                            <w[k],w[k]> = Q
%                            <w[k],v[k]> = S
%                            <v[k],v[k]> = R
%
%
%
%
% Syntax:   [Kp,P,FT,Klambda,HP]=DesignSmoothStationaryLTIPKF(A,B,G,C,H,Q,S,R)
%
%               A,B,G   :   Matrices in LTI state transition equation
%               C       :   Matrix in measurement equation
%               H       :   Matrix in output equation (may be empty)
%               Q,S,R   :   Covariance matrices in LTI system
%
%               Kp      :   Predictive Kalman gain (used by state and output smoother)
```

```
%               P       :   Predictive Kalman covariance (used by state smoother)
%               FT      :   FT = (A-Kp*C)' (used by state and output smoother)
%               Klambda :   Klambda = C'*(C*P*C'+R)^{-1} (used by state and output smoother)
%               HP      :   H*P (used by output smoother)
```

1.11.1.13 StateSmoothStationaryLTIPKF

```
%
% StateSmoothStationaryLTIPKF
%                   Compute the stationary smoothed state estimates of linear time
%                   invariant system. Let the current discrete-time be k. Then
%                   the estimates are computed at times k,k-1,...,k-Ne.
%
%
%               The LTI model is
%
%                   x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                   y[k]   = C*x[k] + v[k] + f
%
%               in which
%
%                   <w[k],w[k]> = Q
%                   <w[k],v[k]> = S
%                   <v[k],v[k]> = R
%
%               The state estimate
%               given the previous information
%               Y = (y(k), y(k-1),..., y(k-Ne-1)) is:
%
%                   xhat[k-Ne|k-Ne-1] = xhat0
%
%               The smoothed output estimates x[k-j|k] for j=0,1,...,Ne
%               are computed. It is assumed that u[k-j] and y[k-j] for
%               (j=0,1, ..., Ne) are specified in U and Y, respectively.
%
%
% Syntax:  X = StateSmoothStationaryLTIPKF(A,B,G,d,C,f,Kp,P,FT,Klambda,xhat0,Y,U,Ne)
%
%               A,B,G,d             :   Matrices in dynamic LTI equation
%               C,f                 :   Matrices in measurement equation
%               Kp,P,FT,Klambda     :   Stationary LTI smoother matrices
%                                       (computed by DesignSmoothStationaryLTIPKF)
%               xhat0               :   xhat0 = x[k-Ne|k-Ne-1]
%               Y                   :   Measurements, Y(:,j) = y[k-Ne-1+j],       j=1,2, ..., Ne+1
%               U                   :   Manipulable inputs, U(:,j) = u[k-Ne-1+j], j=1,2, ..., Ne+1
%               Ne                  :   Estimation window
%
%               X                   :   Smoothed state estimates, X(:,j) = x[k-Ne-1+j|k], j=1,2, ..., Ne+1
```

1.11.1.14 OutputSmoothStationaryLTIPKF

```
%
% OutputSmoothStationaryLTIPKF
%                   Compute the stationary smoothed output estimates of linear time
%                   invariant system. Let the current discrete-time be k. Then
%                   the estimates are computed at times k,k-1,...,k-Ne.
%
%
%               The LTI model is
%
%                   x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                   y[k]   = C*x[k] + v[k] + f
%
%               in which
%
%                   <w[k],w[k]> = Q
%                   <w[k],v[k]> = S
%                   <v[k],v[k]> = R
%
%               The output estimate
%               given the previous information
%               Y = (y(k), y(k-1),..., y(k-Ne-1)) is:
%
%                   xhat[k-Ne|k-Ne-1] = xhat0
%
%               The smoothed output estimates z[k-j|k] for j=0,1,...,Ne
%               are computed. It is assumed that u[k-j] and y[k-j] for
%               (j=0,1, ..., Ne) are specified in U and Y, respectively.
%
%
% Syntax:  Z = OutputSmoothStationaryLTIPKF(A,B,G,d,C,f,Kp,HP,FT,Klambda,xhat0,Y,U,Ne)
%
%               A,B,G,d             :   Matrices in dynamic LTI equation
%               C,f                 :   Matrices in measurement equation
%               Kp,HP,FT,Klambda    :   Stationary LTI smoother matrices
%                                       (computed by DesignSmoothStationaryLTIPKF)
%               xhat0               :   xhat0 = x[k-Ne|k-Ne-1]
```

```
%               Y                   :   Measurements, Y(:,j) = y[k-Ne-1+j],       j=1,2, ..., Ne+1
%               U                   :   Manipulable inputs, U(:,j) = u[k-Ne-1+j], j=1,2, ..., Ne+1
%               Ne                  :   Estimation window
%
%               Z                   :   Smoothed state estimates, Z(:,j) = z[k-Ne-1+j|k], j=1,2, ..., Ne+1
```

1.11.1.15 StationaryStatePredictionCovariance

```
%
% StationaryStatePredictionCovariance   Computes the open loop stationary
%                                       state prediction covariance.
%
%
%                      The LTI model is
%
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%
%                      in which
%
%                           <w[k],w[k]> = Q
%                           <w[k],v[k]> = S
%                           <v[k],v[k]> = R
%
%                     The open loop stationary state prediction covariance
%                     is (for A stable)
%
%                     Pol = lim_{j -> \infty} P[k+j|k]
%
%
% Syntax: Pol = StationaryStatePredictionCovariance(A,G,Q)
%
%           A,G,Q   :   LTI system matrices (see above)
%           Pol     :   Stationary state prediction covariance
```

1.11.1.16 StationaryOutputPredictionCovariance

```
%
% StationaryOutputPredictionCovariance   Computes the open loop stationary
%                                        output prediction covariance.
%
%
%                      The LTI model is
%
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%                           z[k]   = H*x[k] + h
%
%                      in which
%
%                           <w[k],w[k]> = Q
%                           <w[k],v[k]> = S
%                           <v[k],v[k]> = R
%
%                     The open loop stationary state prediction covariance
%                     is (for A stable)
%
%                     Pol = lim_{j -> \infty} P[k+j|k]
%                     Polz = H*Pol*H'
%
% Syntax: Polz = StationaryOutputPredictionCovariance(A,G,H,Q)
%
%           A,G,H,Q   :   LTI system matrices (see above)
%           Polz      :   Stationary output prediction covariance
```

1.11.1.17 StationaryMeasurementPredictionCovariance

```
%
% StationaryMeasurementPredictionCovariance   Computes the open loop stationary
%                                             measurement prediction covariance.
%
%
%                      The LTI model is
%
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%
%                      in which
%
%                           <w[k],w[k]> = Q
%                           <w[k],v[k]> = S
%                           <v[k],v[k]> = R
%
%                     The open loop stationary state prediction covariance
```

```
%                     is (for A stable)
%
%                     Pol = lim_{j -> \infty} P[k+j|k]
%                     Poly = C*Pol*C' + R
%
% Syntax: Poly = StationaryMeasurementPredictionCovariance(A,G,C,Q,R)
%
%           A,G,C,Q,R   :   LTI system matrices (see above)
%           Poly        :   Stationary measurement prediction covariance
```

1.11.2 LTI Systems

```
% LINEAR TIME INVARIANT STATE ESTIMATION
% --------------------------------------------------------------------------
%   MeasurementUpdateLTIKF          Compute the filtered state and process noise estimate
%   TimeUpdateLTIKF                 Compute the one-step update in the Kalman filter
%   StatePredictionLTIKF            Compute state prediction using filtered estimates
%   MeasurementPredictionLTIKF      Compute measurement prediction using filtered estimates
%   OutputPredictionLTIKF           Compute output prediction using filtered estimates
%
%   MeasurementTimeUpdateLTIPKF     Compute one-step prediction in predictive Kalman filter
%   StatePredictionLTIPKF           State predictions based one-step prediction
%   MeasurementPredictionLTIPKF     Measurement predictions based on one-step prediction
%   OutputPredictionLTIPKF          Output prediction based on one-step prediction
%
%   StateSmoothLTI                  Smoothed state estimates in LTI model
%   OutputSmoothLTI                 Smoothed output estimates in LTI model
```

1.11.2.1 MeasurementUpdateLTIKF

```
%
% MeasurementUpdateLTIKF    Computes the measurement update in a
%                           measurememement update-time update Kalman filter
%                           for LTI systems
%
%                           The LTI model is
%
%                               x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                               y[k]   = C*x[k] + v[k] + f
%
%                           in which
%
%                               <w[k],w[k]> = Q
%                               <w[k],v[k]> = S
%                               <v[k],v[k]> = R
%
%                           The state estimate and associated covariance
%                           given the previous information Y = (y(0), y(1),
%                           ..., y(k-1)) is:
%
%                               xhat[k|k-1] = xhat, P[k|k-1] = P.
%
%                           The estimates xf = x[k|k] and wf = w[k|k] based
%                           on the measurement y[k]=y is computed by
%                           MeasurementUpdateLTIKF.
%
%                           In addition the covrariances Pf = P[k|k] and Qf
%                           = Q[k|k] are computed as well as the gain
%                           matrices Kfx and Kfw.
%
%
% Syntax: [xf,wf,Pf,Qf,Kfx,Kfw]=MeasurementUpdateLTIKF(C,f,Q,S,R,xhat,P,y)
%
%           C,f     :   Constants in measurement equation
%           Q,S,R   :   Covariances of the noise variables
%           xhat    :   xhat = x[k|k-1], conditional estimate
%           P       :   P = P[k|k-1], conditional covariance
%           y       :   y = y[k], measurement vector
%
%           xf      :   xf = x[k|k], filtered state estimate
%           wf      :   wf = w[k|k], filtered process noise estimate
%           Pf      :   Pf = P[k|k], covariance associated with xf
%           Qf      :   Qf = Q[k|k], covariance associated with wf
%           Kfx     :   Kalman filter gain for the states
%           Kfw     :   Kalman filter gain for the process noise
```

1.11.2.2 TimeUpdateLTIKF

```
%
% TimeUpdateLTIKF   Computes the time update in a
%                   measurememement update time update Kalman filter
%                   for LTI systems
%
%                   The LTI model is
```

```
%
%                       x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                       y[k]   = C*x[k] + v[k] + f
%
%                   in which
%
%                       <w[k],w[k]> = Q
%                       <w[k],v[k]> = S
%                       <v[k],v[k]> = R
%
%                   The state estimate and associated covariance
%                   given the previous information Y = (y(0), y(1),
%                   ..., y(k)) is:
%
%                       xhat[k|k] = xf, P[k|k] = Pf.
%                       what[k|k] = wf, Q[k|k] = Qf.
%
%                   xf, Pf, wf, and Qf may be computed by
%                   MeasurementUpdateLTIKF. In addition Kfx is needed
%                   which may also be computed by MeasurementUpdateLTIKF.
%
%                   The estimate xhat = x[k+1|k] is computed by
%                   TimeUpdateLTIKF. In addition the covrariance P = P[k+1|k]
%                   is computed.
%
%
%
% Syntax: [xhat,P]=TimeUpdateLTIKF(A,B,G,d,xf,u,wf,Pf,Qf,S,Kfx)
%
%           A,B,G,d :   Constants in dynamic LTI system
%           xf      :   Filtered state estimate, xf = x[k|k]
%           u       :   Manipulated variable, u = u[k]
%           wf      :   Filtered process noise, wf = w[k|k]
%           Pf      :   Covariance, Pf = P[k|k]
%           Qf      :   Covariance, Qf = Q[k|k]
%           S       :   Covariance in stochastic LTI model
%           Kfx     :   Kalman filter gain (computed by
%                       MeasurementUpdateLTIKF)
%
%           xhat    :   One-step prediction estimate, xhat = x[k+1|k]
%           P       :   Covariance associated to xhat, P = P[k+1|k]
```

1.11.2.3 StatePredictionLTIKF

```
%
% StatePredictionLTIKF  Computes state predictions for discrete-time k =
%                       0,1, ..., Np in LTI systems. The estimates are based on the filtered
%                       state and process noise estimates.
%
%
%                      The LTI model is
%
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%
%                      in which
%
%                           <w[k],w[k]> = Q
%                           <w[k],v[k]> = S
%                           <v[k],v[k]> = R
%
%                       The state estimate and associated covariance
%                       given the previous information Y = (y(0), y(1),
%                       ..., y(k)) is:
%
%                           xhat[k|k] = xf
%                           what[k|k] = wf
%
%                       The state estimates x[k+j|k] for j=0,1,...,Np
%                       are computed. It is assumed that u[k+j]=u[k+j|k],
%                       which is specified in Upred.
%
%
% Syntax:   Xpred = StatePredictionLTIKF(A,B,G,d,xf,wf,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           xf      :     Filtered state, xf = x[k|k]
%           wf      :     Filtered process noise, wf = w[k|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 0,1, ..., Np-1
%           Np      :     Prediction horizon
%
%           Xpred   :     Xpred(:,j+1) = x[k+j|k]   j=0,1, ..., Np
```

1.11.2.4 MeasurementPredictionLTIKF

```
%
% MeasurementPredictionLTIKF    Computes measurement predictions for discrete-time k =
```

```
%                           0,1, ..., Np in LTI systems. The estimates are based on the filtered
%                           state and process noise estimates.
%
%
%                     The LTI model is
%
%                          x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                          y[k]   = C*x[k] + v[k] + f
%
%
%                     in which
%
%                          <w[k],w[k]> = Q
%                          <w[k],v[k]> = S
%                          <v[k],v[k]> = R
%
%                      The state estimate and associated covariance
%                      given the previous information Y = (y(0), y(1),
%                      ..., y(k)) is:
%
%                          xhat[k|k] = xf
%                          what[k|k] = wf
%
%                      The output estimates y[k+j|k] for j=0,1,...,Np
%                      are computed. It is assumed that u[k+j]=u[k+j|k],
%                      which is specified in Upred.
%
%
% Syntax:   Ypred = MeasurementPredictionLTIKF(A,B,G,d,C,f,xf,wf,Upred,y,Np)
%
%           A,B,G,d :     LTI system matrices
%           C,f     :     Parameters in output equation
%           xf      :     Filtered state, xf = x[k|k]
%           wf      :     Filtered process noise, wf = w[k|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 0,1, ..., Np-1
%           y       :     Measurement, y = y[k]
%           Np      :     Prediction horizon
%
%           Ypred   :     Ypred(:,j+1) = y[k+j|k]   j=0,1, ..., Np
```

1.11.2.5 OutputPredictionLTIKF

```
%
% OutputPredictionLTIKF Computes output predictions for discrete-time k =
%                       0,1, ..., Np in LTI systems. The estimates are based on the filtered
%                       state and process noise estimates.
%
%
%                     The LTI model is
%
%                          x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                          y[k]   = C*x[k] + v[k] + f
%                          z[k]   = H*x[k] + h
%
%
%                     in which
%
%                          <w[k],w[k]> = Q
%                          <w[k],v[k]> = S
%                          <v[k],v[k]> = R
%
%                      The state estimate and associated covariance
%                      given the previous information Y = (y(0), y(1),
%                      ..., y(k)) is:
%
%                          xhat[k|k] = xf
%                          what[k|k] = wf
%
%                      The output estimates z[k+j|k] for j=0,1,...,Np
%                      are computed. It is assumed that u[k+j]=u[k+j|k],
%                      which is specified in Upred.
%
%
% Syntax:   Zpred = OutputPredictionLTIKF(A,B,G,d,H,h,xf,wf,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           H,h     :     Parameters in output equation
%           xf      :     Filtered state, xf = x[k|k]
%           wf      :     Filtered process noise, wf = w[k|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 0,1, ..., Np-1
%           Np      :     Prediction horizon
%
%           Zpred   :     Zpred(:,j+1) = z[k+j|k]   j=0,1, ..., Np
```

1.11.2.6 MeasurementTimeUpdateLTIPKF

```
%
```

```
% MeasurementTimeUpdateLTIPKF   Computes the one-step prediction in the
%                               predictive Kalman filter for LTI systems
%
%                           The LTI model is
%
%                                  x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                                  y[k]   = C*x[k] + v[k] + f
%
%                           in which
%
%                                  <w[k],w[k]> = Q
%                                  <w[k],v[k]> = S
%                                  <v[k],v[k]> = R
%
%                            The state estimate and associated covariance
%                            given the previous information Y = (y(0), y(1),
%                            ..., y(k-1)) is:
%
%                                xhat[k|k-1] = xhat, P[k|k-1] = P.
%
%                            Based on y[k]=y and u[k]=u, the estimates xhatp = x[k+1|k]
%                            and the associated covariance Pp = P[k+1|k] are computed by
%                            MeasurementTimeUpdateLTIPKF.
%
%
%
% Syntax: [xhatp,Pp]=MeasurementTimeUpdateLTIPKF(A,B,G,d,C,f,Q,S,R,xhat,P,y,u)
%
%                   A,B,G,d :   Matrices in LTI state transition equation
%                   C,f     :   Matrices in LTI measurement equation
%                   Q,S,R   :   Covariance matrices in LTI system
%                   xhat    :   xhat = x[k|k-1]
%                   P       :   P = P[k|k-1]
%                   y       :   y = y[k]
%                   u       :   u = u[k]
%
%                   xhatp   :   xhatp = x[k+1|k]
%                   Pp      :   Pp = P[k+1|k]
```

1.11.2.7 StatePredictionLTIPKF

```
%
% StatePredictionLTIPKF  Computes state predictions for discrete-time k =
%                         1, 2, ..., Np in LTI systems. The estimates are
%                         based on the one-step predictive state estimate.
%
%                       The LTI model is
%
%                            x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                            y[k]   = C*x[k] + v[k] + f
%
%                       in which
%
%                            <w[k],w[k]> = Q
%                            <w[k],v[k]> = S
%                            <v[k],v[k]> = R
%
%                        The state estimate and associated covariance
%                        given the previous information Y = (y(0), y(1),
%                        ..., y(k)) is:
%
%                            xhat[k+1|k] = xhat
%
%                        The state estimates x[k+j|k] for j=1,...,Np
%                        are computed. It is assumed that u[k+j]=u[k+j|k],
%                        (j=1,2, ..., Np-1) which is specified in Upred.
%
%
% Syntax:   Xpred = StatePredictionLTIPKF(A,B,G,d,xhat,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           xhat      :   One-step predictive state estimate, xhat = x[k+1|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 1,2, ..., Np-1
%           Np      :     Prediction horizon
%
%           Xpred   :     Xpred(:,j+1) = x[k+j|k]   j=1, ..., Np
```

1.11.2.8 MeasurementPredictionLTIPKF

```
%
% MeasurementPredictionLTIPKF   Computes measurement predictions for discrete-time k =
%                               1, 2, ..., Np in LTI systems. The estimates are
%                               based on the one-step predictive state estimate.
%
%
%                       The LTI model is
%
```

```
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%
%                       in which
%
%                           <w[k],w[k]> = Q
%                           <w[k],v[k]> = S
%                           <v[k],v[k]> = R
%
%                        The state estimate and associated covariance
%                        given the previous information Y = (y(0), y(1),
%                        ..., y(k)) is:
%
%                           xhat[k+1|k] = xhat
%
%                        The measurement estimates y[k+j|k] for j=1,...,Np
%                        are computed. It is assumed that u[k+j]=u[k+j|k],
%                        (j=1,2, ..., Np-1) which is specified in Upred.
%
%
% Syntax:   Ypred = MeausurementPredictionLTIPKF(A,B,G,d,C,f,xhat,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           C,f     :     LTI system matrices in measurement equation
%           xhat    :     One-step predictive state estimate, xhat = x[k+1|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 1,2, ..., Np-1
%           Np      :     Prediction horizon
%
%           Ypred   :     Ypred(:,j+1) = y[k+j|k]   j=1, ..., Np
```

1.11.2.9 OutputPredictionLTIPKF

```
%
% OutputPredictionLTIPKF Computes output predictions for discrete-time k =
%                         1, 2, ..., Np in LTI systems. The estimates are
%                         based on the one-step predictive state estimate.
%
%
%                       The LTI model is
%
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%                           z[k]   = H*x[k] + h
%
%                       in which
%
%                           <w[k],w[k]> = Q
%                           <w[k],v[k]> = S
%                           <v[k],v[k]> = R
%
%                        The state estimate and associated covariance
%                        given the previous information Y = (y(0), y(1),
%                        ..., y(k)) is:
%
%                           xhat[k+1|k] = xhat
%
%                        The output estimates z[k+j|k] for j=1,...,Np
%                        are computed. It is assumed that u[k+j]=u[k+j|k],
%                        (j=1,2, ..., Np-1) which is specified in Upred.
%
%
% Syntax:   Zpred = OutputPredictionLTIPKF(A,B,G,d,H,h,xhat,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           H,h     :     LTI system matrices in output equation
%           xhat    :     One-step predictive state estimate, xhat = x[k+1|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 1,2, ..., Np-1
%           Np      :     Prediction horizon
%
%           Zpred   :     Zpred(:,j+1) = z[k+j|k]   j=1, ..., Np
```

1.11.2.10 StateSmoothLTI

```
%
% StateSmoothLTI    Compute the smoothed state estimates of linear time
%                   invariant system. Let the current discrete-time be k. Then
%                   the estimates are computed at times k,k-1,...,k-Ne.
%
%
%               The LTI model is
%
%                   x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                   y[k]   = C*x[k] + v[k] + f
%
%               in which
%
%                   <w[k],w[k]> = Q
```

```
%                   <w[k],v[k]> = S
%                   <v[k],v[k]> = R
%
%               The state estimate and associated covariance
%               given the previous information
%               Y = (y(0), y(1),..., y(k-Ne-1)) is:
%
%                   xhat[k-Ne|k-Ne-1] = xhat0
%                   P[k-Ne|k-Ne-1] = P0
%
%               The smoothed state estimates x[k-j|k] for j=0,1,...,Ne
%               are computed. It is assumed that u[k-j] and y[k-j] for
%               (j=0,1, ..., Ne) are specified in U and Y, respectively.
%
%
% Syntax:  X = StateSmoothLTI(A,B,G,d,C,f,Q,S,R,xhat0,P0,Y,U,Ne)
%
%               A,B,G,d     :   Matrices in dynamic LTI equation
%               C,f         :   Matrices in measurement equation
%               Q,S,R       :   Covariance matrices
%               xhat0       :   xhat0 = x[k-Ne|k-Ne-1]
%               Phat0       :   Phat0 = P[k-Ne|k-Ne-1]
%               Y           :   Measurements, Y(:,j) = y[k-Ne-1+j],       j=1,2, ..., Ne+1
%               U           :   Manipulable inputs, U(:,j) = u[k-Ne-1+j], j=1,2, ..., Ne+1
%               Ne          :   Estimation window
%
%               X           :   Smoothed state estimates, X(:,j) = x[k-Ne-1+j|k], j=1,2, ..., Ne+1
```

1.11.2.11 OutputSmoothLTI

```
%
% OutputSmoothLTI   Compute the smoothed output estimates of linear time
%                   invariant system. Let the current discrete-time be k. Then
%                   the estimates are computed at times k,k-1,...,k-Ne.
%
%
%               The LTI model is
%
%                   x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                   y[k]   = C*x[k] + v[k] + f
%                   z[k]   = H*x[k] + h
%
%               in which
%
%                   <w[k],w[k]> = Q
%                   <w[k],v[k]> = S
%                   <v[k],v[k]> = R
%
%               The state estimate and associated covariance
%               given the previous information
%               Y = (y(0), y(1),..., y(k-Ne-1)) is:
%
%                   xhat[k-Ne|k-Ne-1] = xhat0
%                   P[k-Ne|k-Ne-1] = P0
%
%               The smoothed output estimates z[k-j|k] for j=0,1,...,Ne
%               are computed. It is assumed that u[k-j] and y[k-j] for
%               (j=0,1, ..., Ne) are specified in U and Y, respectively.
%
%
% Syntax:  Z = OutputSmoothLTI(A,B,G,d,C,f,H,h,Q,S,R,xhat0,P0,Y,U,Ne)
%
%               A,B,G,d     :   Matrices in dynamic LTI equation
%               C,f         :   Matrices in measurement equation
%               H,h         :   Matirces in output equation
%               Q,S,R       :   Covariance matrices
%               xhat0       :   xhat0 = x[k-Ne|k-Ne-1]
%               Phat0       :   Phat0 = P[k-Ne|k-Ne-1]
%               Y           :   Measurements, Y(:,j) = y[k-Ne-1+j],       j=1,2, ..., Ne+1
%               U           :   Manipulable inputs, U(:,j) = u[k-Ne-1+j], j=1,2, ..., Ne+1
%               Ne          :   Estimation window
%
%               Z           :   Smoothed output estimates, Z(:,j) = z[k-Ne-1+j|k], j=1,2, ..., Ne+1
```

1.11.3 LTI Systems with Missing Observations

```
% LINEAR TIME INVARIANT STATE ESTIMATION WITH MISSING OBSERVATIONS
% ----------------------------------------------------------------------
%   MeasurementUpdateLTIKFWMO       Compute the filtered state and process noise estimate
%   TimeUpdateLTIKFWMO              Compute the one-step update in the Kalman filter
%   StatePredictionLTIKFWMO         Compute state prediction using filtered estimates
%   MeasumentPredictionLTIKFWMO     Compute measurement prediction using filtered estimates
%   OutputPredictionLTIKFWMO        Compute output prediction using filtered estimates
%
%   MeasurementTimeUpdateLTIPKFWMO  Compute one-step prediction in predictive Kalman filter
%   StatePredictionLTIPKFWMO        State predictions based on one-step prediction
%   MeasurementPredictionLTIPKFWMO  Measurement predictions based on one-step prediction
```

```
%   OutputPredictionLTIPKFWMO       Output prediction based on one-step prediction
%
%   StateSmoothLTIWMO               Smoothed state estimates in LTI model with missing observations
%   OutputSmoothLTIWMO              Smoothed output estimates in LTI model with missing observations
%
%   BatchUpdateLTIWMO               Update of current one-step prediction given a batch of past io-data
```

1.11.3.1 MeasurementUpdateLTIKFWMO

```
%
% MeasurementUpdateLTIKFWMO    Computes the measurement update in a
%                              measurememement update-time update Kalman filter
%                              for LTI systems with missing observations.
%
%                           The LTI model is
%
%                               x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                               y[k]   = C*x[k] + v[k] + f
%
%                           in which
%
%                               <w[k],w[k]> = Q
%                               <w[k],v[k]> = S
%                               <v[k],v[k]> = R
%
%                           The state estimate and associated covariance
%                           given the previous information Y = (y(0), y(1),
%                           ..., y(k-1)) is:
%
%                               xhat[k|k-1] = xhat, P[k|k-1] = P.
%
%                           The estimates xf = x[k|k] and wf = w[k|k] based
%                           on the measurement y[k]=y is computed by
%                           MeasurementUpdateLTIKF.
%
%                           In addition the covrariances Pf = P[k|k] and Qf
%                           = Q[k|k] are computed as well as the gain
%                           matrices Kfx and Kfw.
%
%
% Syntax: [xf,wf,Pf,Qf,Kfx,Kfw,Sk]=MeasurementUpdateLTIKFWMO(C,f,Q,S,R,xhat,P,y,yflag)
%
%           C,f     :   Constants in measurement equation
%           Q,S,R   :   Covariances of the noise variables
%           xhat    :   xhat = x[k|k-1], conditional estimate
%           P       :   P = P[k|k-1], conditional covariance
%           y       :   y = y[k], measurement vector
%           yflag   :   0/1 flag indicating whether the corresponding
%                       entry is measured or not.
%                       yflag(i) = 1: entry i is measured
%                       yflag(i) = 0: entry i is not measured
%
%           xf      :   xf = x[k|k], filtered state estimate
%           wf      :   wf = w[k|k], filtered process noise estimate
%           Pf      :   Pf = P[k|k], covariance associated with xf
%           Qf      :   Qf = Q[k|k], covariance associated with wf
%           Kfx     :   Kalman filter gain for the states
%           Kfw     :   Kalman filter gain for the process noise
%           Sk      :   S matrix compatible with missing observations
```

1.11.3.2 TimeUpdateLTIKFWMO

```
%
% TimeUpdateLTIKFWMO   Computes the time update in a
%                      measurememement update time update Kalman filter
%                      for LTI systems with missing observations
%
%                   The LTI model is
%
%                       x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                       y[k]   = C*x[k] + v[k] + f
%
%                   in which
%
%                       <w[k],w[k]> = Q
%                       <w[k],v[k]> = S
%                       <v[k],v[k]> = R
%
%                   The state estimate and associated covariance
%                   given the previous information Y = (y(0), y(1),
%                   ..., y(k)) is:
%
%                       xhat[k|k] = xf, P[k|k] = Pf.
%                       what[k|k] = wf, Q[k|k] = Qf.
%
%                   xf, Pf, wf, and Qf may be computed by
%                   MeasurementUpdateLTIKFWMO. In addition Kfx is needed
```

```
%                   which may also be computed by MeasurementUpdateLTIKFWMO.
%
%                   The estimate xhat = x[k+1|k] is computed by
%                   TimeUpdateLTIKFWMO. In addition the covrariance P = P[k+1|k]
%                   is computed.
%
%
%
% Syntax: [xhat,P]=TimeUpdateLTIKFWMO(A,B,G,d,xf,u,wf,Pf,Qf,Sk,Kfx)
%
%           A,B,G,d :   Constants in dynamic LTI system
%           xf      :   Filtered state estimate, xf = x[k|k]
%           u       :   Manipulated variable, u = u[k]
%           wf      :   Filtered process noise, wf = w[k|k]
%           Pf      :   Covariance, Pf = P[k|k]
%           Qf      :   Covariance, Qf = Q[k|k]
%           Sk      :   Covariance in stochastic LTI model accounting for
%                       missing obsevation (computed by MeasurementUpdateLTIKFWMO)
%           Kfx     :   Kalman filter gain (computed by MeasurementUpdateLTIKFWMO)
%
%           xhat    :   One-step prediction estimate, xhat = x[k+1|k]
%           P       :   Covariance associated to xhat, P = P[k+1|k]
```

1.11.3.3 StatePredictionLTIKFWMO

```
%
% StatePredictionLTIKFWMO  Computes state predictions for discrete-time k =
%                          0,1, ..., Np in LTI systems with missing observations.
%                          The estimates are based on the filtered state and process
%                          noise estimates.
%
%
%                     The LTI model is
%
%                          x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                          y[k]   = C*x[k] + v[k] + f
%
%                     in which
%
%                          <w[k],w[k]> = Q
%                          <w[k],v[k]> = S
%                          <v[k],v[k]> = R
%
%                      The state estimate and associated covariance
%                      given the previous information Y = (y(0), y(1),
%                      ..., y(k)) is:
%
%                          xhat[k|k] = xf
%                          what[k|k] = wf
%
%                      The state estimates x[k+j|k] for j=0,1,...,Np
%                      are computed. It is assumed that u[k+j]=u[k+j|k],
%                      which is specified in Upred.
%
%
% Syntax:   Xpred = StatePredictionLTIKFWMO(A,B,G,d,xf,wf,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           xf      :     Filtered state, xf = x[k|k]
%           wf      :     Filtered process noise, wf = w[k|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 0,1, ..., Np-1
%           Np      :     Prediction horizon
%
%           Xpred   :     Xpred(:,j+1) = x[k+j|k]   j=0,1, ..., Np
```

1.11.3.4 MeasurementPredictionLTIKFWMO

```
%
% MeasurementPredictionLTIKFWMO    Computes measurement predictions for discrete-time k =
%                                  0,1, ..., Np in LTI systems with missing observations.
%                                  The estimates are based on the filtered
%                                  state and process noise estimates.
%
%
%                     The LTI model is
%
%                          x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                          y[k]   = C*x[k] + v[k] + f
%
%
%                     in which
%
%                          <w[k],w[k]> = Q
%                          <w[k],v[k]> = S
%                          <v[k],v[k]> = R
%
%                      The state estimate and associated covariance
```

```
%                       given the previous information Y = (y(0), y(1),
%                       ..., y(k)) is:
%
%                           xhat[k|k] = xf
%                           what[k|k] = wf
%
%                       The output estimates y[k+j|k] for j=0,1,...,Np
%                       are computed. It is assumed that u[k+j]=u[k+j|k],
%                       which is specified in Upred.
%
%
% Syntax:   Ypred = MeasurementPredictionLTIKFWMO(A,B,G,d,C,f,xf,wf,Upred,y,yflag,Np)
%
%           A,B,G,d :     LTI system matrices
%           C,f     :     Parameters in output equation
%           xf      :     Filtered state, xf = x[k|k]
%           wf      :     Filtered process noise, wf = w[k|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 0,1, ..., Np-1
%           y       :     Measurement, y = y[k]
%           yflag   :     0/1 flag indicating whether the corresponding
%                         entry is measured or not.
%                            yflag(i) = 1: entry i is measured
%                            yflag(i) = 0: entry i is not measured
%           Np      :     Prediction horizon
%
%           Ypred   :     Ypred(:,j+1) = y[k+j|k]   j=0,1, ..., Np
```

1.11.3.5 OutputPredictionLTIKFWMO

```
%
% OutputPredictionLTIKFWMO Computes output predictions for discrete-time k =
%                          0,1, ..., Np in LTI systems with missing observations.
%                          The estimates are based on the filtered state and process
%                          noise estimates.
%
%
%                      The LTI model is
%
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%                           z[k]   = H*x[k] + h
%
%
%                      in which
%
%                           <w[k],w[k]> = Q
%                           <w[k],v[k]> = S
%                           <v[k],v[k]> = R
%
%                       The state estimate and associated covariance
%                       given the previous information Y = (y(0), y(1),
%                       ..., y(k)) is:
%
%                           xhat[k|k] = xf
%                           what[k|k] = wf
%
%                       The output estimates z[k+j|k] for j=0,1,...,Np
%                       are computed. It is assumed that u[k+j]=u[k+j|k],
%                       which is specified in Upred.
%
%
% Syntax:   Zpred = OutputPredictionLTIKFWMO(A,B,G,d,H,h,xf,wf,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           H,h     :     Parameters in output equation
%           xf      :     Filtered state, xf = x[k|k]
%           wf      :     Filtered process noise, wf = w[k|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 0,1, ..., Np-1
%           Np      :     Prediction horizon
%
%           Zpred   :     Zpred(:,j+1) = z[k+j|k]   j=0,1, ..., Np
```

1.11.3.6 MeasurementTimeUpdateLTIPKFWMO

```
%
% MeasurementTimeUpdateLTIPKFWMO   Computes the one-step prediction in the
%                                  predictive Kalman filter for LTI
%                                  systems with missing observations.
%
%                          The LTI model is
%
%                                 x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                                 y[k]   = C*x[k] + v[k] + f
%
%                          in which
%
%                                 <w[k],w[k]> = Q
```

```
%                               <w[k],v[k]> = S
%                               <v[k],v[k]> = R
%
%                          The state estimate and associated covariance
%                          given the previous information Y = (y(0), y(1),
%                          ..., y(k-1)) is:
%
%                              xhat[k|k-1] = xhat, P[k|k-1] = P.
%
%                          Based on y[k]=y and u[k]=u, the estimates xhatp = x[k+1|k]
%                          and the associated covariance Pp = P[k+1|k] are computed by
%                          MeasurementTimeUpdateLTIPKFWMO.
%
%
%
% Syntax: [xhatp,Pp]=MeasurementTimeUpdateLTIPKFWMO(A,B,G,d,C,f,Q,S,R,xhat,P,y,yflag,u)
%
%                  A,B,G,d :   Matrices in LTI state transition equation
%                  C,f     :   Matrices in LTI measurement equation
%                  Q,S,R   :   Covariance matrices in LTI system
%                  xhat    :   xhat = x[k|k-1]
%                  P       :   P = P[k|k-1]
%                  y       :   y = y[k]
%                  yflag   :  0/1 flag indicating whether the corresponding
%                             entry is measured or not.
%                                yflag(i) = 1: entry i is measured
%                                yflag(i) = 0: entry i is not measured
%                  u       :   u = u[k]
%
%                  xhatp   :   xhatp = x[k+1|k]
%                  Pp      :   Pp = P[k+1|k]
```

1.11.3.7 StatePredictionLTIPKFWMO

```
%
% StatePredictionLTIPKFWMO  Computes state predictions for discrete-time k =
%                           1, 2, ..., Np in LTI systems with missing obsevations.
%                           The estimates are based on the one-step predictive state estimate.
%
%                      The LTI model is
%
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%
%                      in which
%
%                           <w[k],w[k]> = Q
%                           <w[k],v[k]> = S
%                           <v[k],v[k]> = R
%
%                       The state estimate and associated covariance
%                       given the previous information Y = (y(0), y(1),
%                       ..., y(k)) is:
%
%                           xhat[k+1|k] = xhat
%
%                       The state estimates x[k+j|k] for j=1,...,Np
%                       are computed. It is assumed that u[k+j]=u[k+j|k],
%                       (j=1,2, ..., Np-1) which is specified in Upred.
%
%
% Syntax:   Xpred = StatePredictionLTIPKFWMO(A,B,G,d,xhat,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           xhat      :   One-step predictive state estimate, xhat = x[k+1|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 1,2, ..., Np-1
%           Np      :     Prediction horizon
%
%           Xpred   :     Xpred(:,j+1) = x[k+j|k]   j=1, ..., Np
```

1.11.3.8 MeasurementPredictionLTIPKFWMO

```
%
% MeasurementPredictionLTIPKFWMO   Computes measurement predictions for discrete-time k =
%                                  1, 2, ..., Np in LTI systems with missing observations.
%                                  The estimates are based on the one-step predictive state estimate.
%
%
%                      The LTI model is
%
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%
%                      in which
%
%                           <w[k],w[k]> = Q
%                           <w[k],v[k]> = S
```

```
%                           <v[k],v[k]> = R
%
%                       The state estimate and associated covariance
%                       given the previous information Y = (y(0), y(1),
%                       ..., y(k)) is:
%
%                           xhat[k+1|k] = xhat
%
%                       The measurement estimates y[k+j|k] for j=1,...,Np
%                       are computed. It is assumed that u[k+j]=u[k+j|k],
%                       (j=1,2, ..., Np-1) which is specified in Upred.
%
%
% Syntax:   Ypred = MeausurementPredictionLTIPKFWMO(A,B,G,d,C,f,xhat,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           C,f     :     LTI system matrices in measurement equation
%           xhat    :     One-step predictive state estimate, xhat = x[k+1|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 1,2, ..., Np-1
%           Np      :     Prediction horizon
%
%           Ypred   :     Ypred(:,j+1) = y[k+j|k]   j=1, ..., Np
```

1.11.3.9 Output Prediction

```
%
% OutputPredictionLTIPKFWMO Computes output predictions for discrete-time k =
%                           1, 2, ..., Np in LTI systems with missing observations.
%                           The estimates are based on the one-step predictive state estimate.
%
%
%                      The LTI model is
%
%                           x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                           y[k]   = C*x[k] + v[k] + f
%                           z[k]   = H*x[k] + h
%
%                      in which
%
%                           <w[k],w[k]> = Q
%                           <w[k],v[k]> = S
%                           <v[k],v[k]> = R
%
%                       The state estimate and associated covariance
%                       given the previous information Y = (y(0), y(1),
%                       ..., y(k)) is:
%
%                           xhat[k+1|k] = xhat
%
%                       The output estimates z[k+j|k] for j=1,...,Np
%                       are computed. It is assumed that u[k+j]=u[k+j|k],
%                       (j=1,2, ..., Np-1) which is specified in Upred.
%
%
% Syntax:   Zpred = OutputPredictionLTIPKFWMO(A,B,G,d,H,h,xhat,Upred,Np)
%
%           A,B,G,d :     LTI system matrices
%           H,h     :     LTI system matrices in output equation
%           xhat    :     One-step predictive state estimate, xhat = x[k+1|k]
%           Upred   :     Upred(:,j+1) = u[k+j|k] j = 1,2, ..., Np-1
%           Np      :     Prediction horizon
%
%           Zpred   :     Zpred(:,j+1) = z[k+j|k]   j=1, ..., Np
```

1.11.3.10 StateSmoothLTIWMO

```
%
% StateSmoothLTIWMO    Compute the smoothed state estimates of linear time
%                      invariant system with missing observations.
%                      Let the current discrete-time be k. Then
%                      the estimates are computed at times k,k-1,...,k-Ne.
%
%
%               The LTI model is
%
%                   x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                   y[k]   = C*x[k] + v[k] + f
%
%               in which
%
%                   <w[k],w[k]> = Q
%                   <w[k],v[k]> = S
%                   <v[k],v[k]> = R
%
%               The state estimate and associated covariance
%               given the previous information
%               Y = (y(0), y(1),..., y(k-Ne-1)) is:
```

```
%
%                   xhat[k-Ne|k-Ne-1] = xhat0
%                   P[k-Ne|k-Ne-1] = P0
%
%               The smoothed state estimates x[k-j|k] for j=0,1,...,Ne
%               are computed. It is assumed that u[k-j] and y[k-j] for
%               (j=0,1, ..., Ne) are specified in U and Y, respectively.
%
%
% Syntax:  X = StateSmoothLTIWMO(A,B,G,d,C,f,Q,S,R,xhat0,P0,Y,Yflag,U,Ne)
%
%               A,B,G,d     :   Matrices in dynamic LTI equation
%               C,f         :   Matrices in measurement equation
%               Q,S,R       :   Covariance matrices
%               xhat0       :   xhat0 = x[k-Ne|k-Ne-1]
%               Phat0       :   Phat0 = P[k-Ne|k-Ne-1]
%               Y           :   Measurements, Y(:,j) = y[k-Ne-1+j],       j=1,2, ..., Ne+1
%               Yflag       :   1/0 matrix indicating if the measurement in
%                               the corresponding entry is available.
%               U           :   Manipulable inputs, U(:,j) = u[k-Ne-1+j], j=1,2, ..., Ne+1
%               Ne          :   Estimation window
%
%               X           :   Smoothed state estimates, X(:,j) = x[k-Ne-1+j|k], j=1,2, ..., Ne+1
```

1.11.3.11 OutputSmoothLTIWMO

```
%
% OutputSmoothLTIWMO   Compute the smoothed output estimates of a linear time
%                      invariant system with missing observations.
%                      Let the current discrete-time be k. Then
%                      the estimates are computed at times k,k-1,...,k-Ne.
%
%
%               The LTI model is
%
%                   x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
%                   y[k]   = C*x[k] + v[k] + f
%                   z[k]   = H*x[k] + h
%
%               in which
%
%                   <w[k],w[k]> = Q
%                   <w[k],v[k]> = S
%                   <v[k],v[k]> = R
%
%               The state estimate and associated covariance
%               given the previous information
%               Y = (y(0), y(1),..., y(k-Ne-1)) is:
%
%                   xhat[k-Ne|k-Ne-1] = xhat0
%                   P[k-Ne|k-Ne-1] = P0
%
%               The smoothed output estimates z[k-j|k] for j=0,1,...,Ne
%               are computed. It is assumed that u[k-j] and y[k-j] for
%               (j=0,1, ..., Ne) are specified in U and Y, respectively.
%               Yflag specifies whether the corresponding entry in Y
%               corresponds to a valid measurement.
%
%
% Syntax:  Z = OutputSmoothLTIWMO(A,B,G,d,C,f,H,h,Q,S,R,xhat0,P0,Y,Yflag,U,Ne)
%
%               A,B,G,d     :   Matrices in dynamic LTI equation
%               C,f         :   Matrices in measurement equation
%               H,h         :   Matirces in output equation
%               Q,S,R       :   Covariance matrices
%               xhat0       :   xhat0 = x[k-Ne|k-Ne-1]
%               Phat0       :   Phat0 = P[k-Ne|k-Ne-1]
%               Y           :   Measurements, Y(:,j) = y[k-Ne-1+j],       j=1,2, ..., Ne+1
%               Yflag       :   1/0 matrix indicating if the measurement in
%                               the corresponding entry is available. 1:
%                               measurement available. 0: meausrement not
%                               available.
%               U           :   Manipulable inputs, U(:,j) = u[k-Ne-1+j], j=1,2, ..., Ne+1
%               Ne          :   Estimation window
%
%               Z           :   Smoothed output estimates, Z(:,j) = z[k-Ne-1+j|k], j=1,2, ..., Ne+1
```

1.11.3.12 BatchUpdateLTIWMO

```
%
% BatchUpdateLTIWMO  Computes updated one-step prediction estimate and
%                    covariance for a batch of measurements stemming from
%                    an LTI system with missing observations.
%
%              The LTI model is
%
%                   x[k+1] = A*x[k] + B*u[k] + G*w[k] + d
```

```
%                   y[k]   = C*x[k] + v[k] + f
%
%               in which
%
%                   <w[k],w[k]> = Q
%                   <w[k],v[k]> = S
%                   <v[k],v[k]> = R
%
%               Let x0 = x[k+1-Ne|k-Ne] and P0 = P[k+1-Ne|k-Ne].
%               Given the batch of data {y[j],u[j]} for j=k+1-Ne,...,k
%               with possile missing observation, BatchUpdateLTIWMO
%               computes xhatp = x[k+1|k] and Pp = P[k+1|k].
%
%
%
% Syntax: [xhatp,Pp]=BatchUpdateLTIWMO(A,B,G,d,C,f,Q,S,R,x0,P0,Y,Yflag,U,Ne)
%
%                   A,B,G,d :   Matrices in LTI state transition equation
%                   C,f     :   Matrices in measurement equation
%                   Q,S,R   :   Covariance matrices in LTI system
%                   x0      :   x0 = x[k+1-Ne|k-Ne]
%                   P0      :   P0 = P[k+1-Ne|k-Ne]
%                   Y       :   Array with measurements (dim: p x Ne)
%                   Yflag   :   Array with measurement available flag (dim: p x Ne))
%                   U       :   Array with process inputs (dim: n x Ne)
%                   Ne      :   Length of estimation window
%
%                   xhatp   :   xhatp = x[k+1|k]
%                   Pp      :   Pp = P[k+1|k]
```

1.11.4 Disturbance Models

```
% DISTURBANCE MODELS
% ------------------------------------------------------------------------
%   DesignDetectableDisturbanceModel    Designs an unstructured disturbance model such
%                                       that the augmented system is detectable
%   AugmentLTISystem                    Augment an LTI system with an unstructured disturbance model
%   AugmentLTISystemWithCovariance      Augment a stochastic LTI system with an unstructured disturbance model
```

1.11.4.1 DesignDetectableDisturbanceModel

```
%
% DesignDetectableDisturbanceModel  Given a detectable system (C,A) a
%                                   disturbance model with p-integrating
%                                   modes is designed such that the
%                                   augmented system is detectable. p is
%                                   the number of measurements
%
%                   LTI system considered
%
%                       x[k+1] = A*x[k] + B*u[k] + G*w[k] + d + Bs*s[k]
%                       y[k]   = C*x[k] + v[k] + f + Cs*s[k]
%
%                       s[k+1] = s[k] + eta[k]
%
%                   <w[k],w[k]> = Q, <w[k],v[k]> = S, <v[k],v[k]> = R
%                   <eta[k],eta[k]> = Qs, <eta[k],w[k]> = 0, <eta[k], v[k]> = 0
%
%
%
% Syntax: [Bs,Cs] = DesignDetectableDisturbanceModel(A,C)
%
%               A,B     :   Matrices in LTI system (see above)
%               Bs,Cs   :   Disturbance model matrices (see above)
```

1.11.4.2 AugmentLTISystem

```
%
% AugmentLTISystem  Augments an LTI system with an unstructured disturbance
%                   model such that it is represented as a standard state space model.
%
%                   LTI system considered
%
%                       x[k+1] = A*x[k] + B*u[k] + G*w[k] + d + Bs*s[k]
%                       y[k]   = C*x[k] + v[k] + f + Cs*s[k]
%
%                       s[k+1] = s[k] + eta[k]
%
%                   <w[k],w[k]> = Q, <w[k],v[k]> = S, <v[k],v[k]> = R
%                   <eta[k],eta[k]> = Qs, <eta[k],w[k]> = 0, <eta[k], v[k]> = 0
%
%                   Standard model:
%
%                       xbar[k+1] = Abar*xbar[k] + Bbar*u[k] + Gbar*wbar[k] + dbar
```

```
%                       y[k]      = Cbar*xbar[k] + vbar[k] + fbar
%
%
% Syntax: [Abar,Bbar,Gbar,dbar,Cbar,fbar]=AugmentLTISystem(A,B,G,d,C,f,Bs,Cs)
%
%           A,B,G,d     :   Original matrices in LTI state transition equation
%           C,f         :   Original matrices in LTI measurement equation
%           Bs,Cs       :   Matrices in unstructured disturbance model
%
%           Abar,Bbar,Gbar,dbar :   Matrices in augmented LTI state transition equation
%           Cbar,fbar           :   Matrices in augmented LTI measurement equation
```

1.11.4.3 AugmentLTISystemWithCovariance

```
%
% AugmentLTISystemWithCovariance
%                   Augments an LTI system with an unstructured disturbance
%                   model and its covariance such that it is represented as a standard state space model.
%
%                   LTI system considered
%
%                       x[k+1] = A*x[k] + B*u[k] + G*w[k] + d + Bs*s[k]
%                       y[k]   = C*x[k] + v[k] + f + Cs*s[k]
%
%                       s[k+1] = s[k] + eta[k]
%
%                   <w[k],w[k]> = Q, <w[k],v[k]> = S, <v[k],v[k]> = R
%                   <eta[k],eta[k]> = Qs, <eta[k],w[k]> = 0, <eta[k], v[k]> = 0
%
%                   Standard model:
%
%                       xbar[k+1] = Abar*xbar[k] + Bbar*u[k] + Gbar*wbar[k] + dbar
%                       y[k]      = Cbar*xbar[k] + vbar[k] + fbar
%
%
% Syntax: [Abar,Bbar,Gbar,dbar,Cbar,fbar,Qbar,Sbar,Rbar]=AugmentLTISystemWithCovariance(A,B,G,d,C,f,Q,S,R,Bs,Cs,Qs)
%
%           A,B,G,d     :   Original matrices in LTI state transition equation
%           C,f         :   Original matrices in LTI measurement equation
%           Q,S,R       :   Covariance matrices in original LTI model
%           Bs,Cs       :   Matrices in unstructured disturbance model
%           Qs          :   Covariance matrix in unstructured disturbance model
%
%           Abar,Bbar,Gbar,dbar :   Matrices in augmented LTI state transition equation
%           Cbar,fbar           :   Matrices in augmented LTI measurement equation
%           Qbar,Sbar,Rbar      :   Covariance matrices in augmented LTI system
```

1.11.5 Utility Functions

```
% UTILITY FUNCTIONS
% ------------------------------------------------------------------------
%   IsControllable              Tests whether (A,B) is controllable
%   IsObservable                Tests whether (C,A) is observable
%   IsStabilizable              Tests whether (A,B) is stabilizable
%   IsDetectable                Tests whether (C,A) is detectable
%   IsUnitCircleControllable    Tests whether integrating modes of (A,B) can be controlled
%   IsUnitCircleObservable      Tests whether integrating modes of (C,A) can be observed
%   IsStable                    Tests whether the matrix A is discrete-time stable
```

1.11.5.1 IsControllable

```
%
% IsControllable    Tests whether the system (A,B) is controllable
%
%
% Syntax:   flag = IsControllable(A,B)
%
%               A,B :   Discrete-time matrices in LTI system description
%               flag:   Flag indicating whether (A,B) is controllable
%                           flag = 1:   (A,B) is controllable
%                           flag = 0:   (A,B) is not controllable
```

1.11.5.2 IsObservable

```
%
% IsObservable  Tests whether (C,A) is observable
%
%
% Syntax:   flag = IsObservable(C,A)
%
```

```
%               C,A : Discrete-time matrices in LTI system description
%               flag:   Flag indicating whether (C,A) is observable
%                           flag = 1:   (C,A) is observable
%                           flag = 0:   (C,A) is not observable
```

1.11.5.3 IsStabilizable

```
%
% IsStabilizable    Test whether (A,B) is stabilizable
%
%
% Syntax:  flag = IsStabilizable(A,B)
%
%               A,B :   Discrete-time matrices in LTI system description
%               flag:   Flag indicating whether (A,B) is stabilizable
%                           flag = 1:   (A,B) is stabilizable
%                           flag = 0:   (A,B) is not stabilizable
```

1.11.5.4 IsDetectable

```
%
% IsDetectable  Tests whether (C,A) is detectable
%
%
% Syntax:   flag = IsDetectable(C,A)
%
%               C,A : Discrete-time matrices in LTI system description
%               flag:   Flag indicating whether (C,A) is detectable
%                           flag = 1:   (C,A) is detectable
%                           flag = 0:   (C,A) is not detectable
%
```

1.11.5.5 IsUnitCircleControllable

```
%
% IsUnitCircleControllable    Tests whether the system (A,B) is unit circle
%                             controllable
%
%
% Syntax:   flag = IsUnitCicleControllable(A,B)
%
%               A,B :   Discrete-time matrices in LTI system description
%               flag:   Flag indicating whether (A,B) is unit circle controllable
%                           flag = 1:   (A,B) is unit circle controllable
%                           flag = 0:   (A,B) is not circle controllable
```

1.11.5.6 IsUnitCircleObservable

```
%
% IsUnitCircleObservable  Tests whether (C,A) is unit circle observable
%
%
% Syntax:   flag = IsUnitCircleObservable(C,A)
%
%               C,A : Discrete-time matrices in LTI system description
%               flag:   Flag indicating whether (C,A) is unit circle observable
%                           flag = 1:   (C,A) is unit circle observable
%                           flag = 0:   (C,A) is not unit circle observable
```

1.11.5.7 IsStable

```
%
% IsStable  Tests whether the matrix A is stable in a discrete-time sense
%
% Syntax:   flag = IsStable(A)
%
%               A       :   Matrix A in LTI discrete-time system
%               flag    :   Indicator of stability of A
%                               flag = 1:   A is stable
%                               flag = 0:   A is not stable
```

References

Francis, B. A. and Wonham, W. M. (1976). The Internal Model Principle of Control Theory. *Automatica*, **12**, 457–465.

Jazwinski, A. H. (1970). *Stochastic Processes and Filtering Theory*. Academic Press.

Kailath, T.; Sayed, A. H. and Hassibi, B. (2000). *Linear Estimation*. Prentice Hall.

Muske, K. R. and Badgwell, T. A. (2002). Disturbance Modeling for Offset-Free Linear Model Predictive Control. *Journal of Process Control*, **12**, 617–632.

Pannocchia, G. and Rawlings, J. B. (2003). Disturbance Models for Offset-Free Model-Predictive Control. *AIChE J.*, **49**, 426–437.

Wittenmark, B.; Åström, K. J. and Årzén, K.-E. (2002). *Computer Control: An Overview*.

Zhou, K.; Doyle, J. C. and Glover, K. (1996). *Robust and Optimal Control*. Prentice Hall, Upper Saddle River, New Jersey.