COMP 145 UNC-Chapel Hill
Implementation Manual

The Kalman Filter On-line Learning Tool

Greg Welch, client

May 1, 2001

__________________________  ____________________________
Chris Riley, Director         Tom Bodenheimer, Producer

__________________________  ____________________________
Erin Parker, Admin. Leader    John Carpenter, Librarian
1 Introduction

This implementation of the The Kalman Filter On-line Learning Tool is based on two months’ collaboration amongst the team and with the client, Greg Welch. It represents a revision of the earlier design specification dated February 20, 2001.

2 High-Level Design Specification

The high-level design consists of a structural model and a control model.

2.1 Structural Model

The Kalman Filter On-line Learning Tool is composed of the modules shown in Figure 1: a user interface, a Kalman filter engine, a data repository, and a plotting tool. The user interface serves to separate the user from the inner workings of the learning tool.

![Figure 1: Structural Model Diagram.](image)

2.1.1 User Interface

The user interface controls all of the windows or screens presented to the user. Windows include:

- a Main window that displays the current parameters for the Kalman filter simulation
- an Edit screen that allows the user to change the parameters for the Kalman filter simulation
- a Plot screen that displays x-y plots of the estimated and actual states, the square root of the covariance, and the residual
- a Step screen that allows the user to step through a simulation and view intermediate values
- a Help screen that displays instructions on how to use the tool.

The user interface delegates the task of creating the information contained in each window to the other subsystems as necessary.

### 2.1.2 Kalman Filter Engine

The Kalman filter engine implements the Kalman filter equations and is responsible for generating the Kalman filter data for a simulation. It receives a set of parameters from the user interface and returns data covering all time steps of a simulation.

### 2.1.3 Data Repository

The data repository stores a record of the filter’s iterations and the values computed. It provides methods that allow the user interface and the plotting tool to access specific data from a simulation.

### 2.1.4 Plotting Tool

The plotting tool generates x-y plots of the state and error estimates produced by the filter. Plots are created from values accessed through a data repository object. The user interface is responsible for displaying these plots.

### 2.2 Control Model

The user interface acts as the manager of the other subsystems. Its tasks include:

- Displaying the Edit, Plot, Step, and Help windows
- Running a Kalman filter simulation (with the help of the Kalman filter engine)
- Creating x-y plots of the data (with the help of the plotting tool)

These tasks are triggered by user input (e.g., clicking on Plot button).
3 Detailed Design Specification

This section describes the detailed design of the four subsystems (user interface, Kalman filter engine, data repository, and plotting tool) through the use of object and interaction diagrams. All modules are implemented in Java™. A detailed description of each class including instance variables and method interfaces is found in “Java™ API Documentation for the The Kalman Filter On-line Learning Tool”.

3.1 Object Models

3.1.1 User Interface

The object model describing the user interface is shown in Figure 2. The KalmanFilterTool class is the
primary class of the user interface module. It is in charge of displaying a main window which contains the current settings and control buttons. Also contained in the module are implementations of the ActionListener interface for each control button. These listen for user input and take the appropriate action. In each case, a new window is created and displayed to the user.

### 3.1.2 Kalman Filter Engine

The object model describing the Kalman filter engine is shown in Figure 3. The KalmanFilter class is composed of two interfaces: ProcessModel and MeasurementModel. A third interface, Truth, is used by MeasurementModel to generate a signal corresponding to the actual dynamics of the process. An important feature of the design is that the implementation of the Kalman filter equations in the KalmanFilter class is independent of the specific simulation example.

There are several auxiliary classes that are not shown in Figure 3.
• The `ProcessType`, `TruthType`, and `MeasurementType` classes store sets of constants corresponding to specific process, truth, and measurement types, respectively.

• The `KalmanFilterData` class encapsulates the simulation data for a single time step.

• The creation of a `KalmanFilter` object is handled by several factory classes. `KFModelFactory` is an interface that defines the creation of models corresponding to a specific model type. For example,

  ```java
def makeTruth(TruthType type)
  return the Truth object for the given TruthType. The WaterTankModelFactory class is an implementation of this interface for the water tank simulation problem. A different example problem would require a different implementation of the interface. The Kalman filter engine itself is created using the KalmanFilterFactory class.

• The `Matrix` class from the Jama package is used to store all matrices and vectors in the Kalman filter equations.

The complete list of classes and method interfaces is found in “Java™ API Documentation for the The Kalman Filter On-line Learning Tool”.

3.1.3 Data Repository

The data repository module consists of a single class, `DataRepository`. Methods provide access to data for an entire simulation run. For example,

```java
def getEstimate()
returns an array of the estimated state of the system for a complete simulation. These methods are primarily used by the plotting tool. There are additional methods for use by the user interface in stepping through the data that return string representations of the data at individual time steps. For instance,

```java
def getPredState(int timestep)
returns a string representing the predicted state at the given time step. Misceallaneous methods are also provided that return the maximum number of time steps in a simulation, the number of components in the state and measurement vectors, and a string representation of the entire simulation for writing to an external file. Again, “Java™ API Documentation for the The Kalman Filter On-line Learning Tool” contains the complete description of this class and its methods.

3.1.4 Plotting Tool

The plotting tool consists of a single class, `ThreePlot`, that accepts arrays containing values for the predicted and true state, covariance, and residual from a simulation and returns a window containing the three plots. As shown in Figure 4, this class delegates the task of actually creating the plots to the `Plot` class in the PtPlot plotting package.

3.2 Interaction Diagrams

This section provides interaction diagrams outlining the behavior of the The Kalman Filter On-line Learning Tool for various user inputs. Dashed lines indicate the creation of objects. Solid lines indicate method calls. Time increases as you move down the diagram.
3.2.1 Edit

The interaction diagram showing the system behavior for changing the parameters for a simulation is depicted in Figure 5. The user presses the Edit button in the main control window and is then presented with an EditDialog window. Upon selecting the new parameters, the user presses the Apply button which updates the current parameters in the main control window.
3.2.2 Plot

Figure 6 shows the system behavior when the *Plot* button is pressed. When the user presses the *Plot* button in the main control window, a *KalmanFilter* object is created based on the current parameter settings and a simulation is computed. The data from the simulation is then used to create a *DataRepository* object. After querying the *DataRepository* object, a plot window is created and displayed to the user. Note that if the *Plot* button is pressed and the current parameters have not been changed, the steps to create and run a new simulation are omitted.
3.2.3 Step

Figure 7 shows the system behavior when the Step button is pressed. Pressing the Step button triggers the creation of a StepDialog window. For the purposes of this diagram only, it is assumed that a Kalman filter simulation and a corresponding data repository have already been created. Also the Rewind, StepBack, and FastForward listeners are omitted for the sake of brevity.
3.2.4 Help

Figure 8 shows the system behavior when the Help button is pressed. When the user presses the Help button

in the main control window, a HelpDialog window is displayed showing the main help page in HTML. The user may then select hyperlinks which cause the current help page to be updated similar to a web browser.