Tutorial 10

Kalman and Particle filters

H. R. B. Orlande[1,*], M. J. Colaço[1], G. S. Dulikravich[2], F. L. V. Vianna[3], W. B. da Silva[1,4], H. M. da Fonseca[1,4] and O. Fudym[4]

[1]Department of Mechanical Engineering, Politécnica/COPPE, Federal University of Rio de Janeiro, UFRJ, Cid. Universitaria, Cx. Postal: 68503, Rio de Janeiro, RJ, 21941-972, Brazil, helcio@mecanica.ufrj.br, colaco@ufrj.br, wellingtonuff@yahoo.com.br, hmassardf@gmail.com
[2]Department of Mechanical and Materials Engineering, Florida International University, 10555 West Flagler Street, EC 3462, Miami, Florida 33174, U.S.A., dulikrav@fiu.edu
[3]Department of Subsea Technology, Petrobras Research and Development Center – CENPES, Av. Horácio Macedo, 950, Cidade Universitária, Ilha do Fundão, 21941-915, Rio de Janeiro, RJ, Brazil, fvianna@petrobras.com.br
[4]Université de Toulouse ; Mines Albi ; CNRS; Centre RAPSODEE, Campus Jarlard, F-81013 Albi cedex 09, France, olivier.fudym@enstimac.fr

Abstract. In this tutorial we present a general description of state estimation problems within the Bayesian framework. State estimation problems are addressed, in which evolution and measurement stochastic models are used to predict the dynamic behavior of physical systems. The application of two Bayesian filters to linear and non-linear unsteady heat conduction problems is demonstrated, namely: a) the Kalman filter, and b) the Particle Filter through the sampling importance resampling algorithm.

1. Introduction

State estimation problems, also designated *as nonstationary inverse problems* [1], are of great interest in innumerable practical applications. In such kinds of problems, the available measured data is used together with prior knowledge about the physical phenomena and the measuring devices, in order to sequentially produce estimates of the desired dynamic variables. This is accomplished in such a manner that the error is minimized statistically [2]. For example, the position of an aircraft can be estimated through the time-integration of its velocity components since departure. However, it may also be measured with a GPS system and an altimeter. State estimation problems deal with the combination of the model prediction (integration of the velocity components that contain errors due to the velocity measurements) and the GPS and altimeter measurements that are also uncertain, in order to obtain more accurate estimations of the system variables (aircraft position).

State estimation problems are solved with the so-called Bayesian filters [1,2]. In the Bayesian approach to statistics, an attempt is made to utilize all available information in order to reduce the amount of uncertainty present in an inferential or decision-making problem. As new information is obtained, it is combined with previous information to form the basis for statistical procedures. The formal mechanism used to combine the new information with the previously available information is known as Bayes' theorem [1,3].

The most widely known Bayesian filter method is the Kalman filter [1,2,4-9]. However, the application of the Kalman filter is limited to linear models with additive Gaussian noises. Extensions of the Kalman filter were developed in the past for less restrictive cases by using linearization techniques [1,3,6,7,8]. Similarly, Monte Carlo methods have been developed in order to represent the posterior density in terms of random samples and associated weights. Such Monte Carlo methods,

* To whom any correspondence should be addressed.

usually denoted as particle filters among other designations found in the literature, do not require the restrictive hypotheses of the Kalman filter. Hence, particle filters can be applied to non-linear models with non-Gaussian errors [1,4,8-18].

In this tutorial we present the Kalman filter and the Sampling Importance Resampling (SIR) algorithm of the Particle filter. These Bayesian filters are used here to predict the temperature in a medium where the heat conduction model and temperature measurements contain errors. Linear and non-linear heat conduction problems are examined, as well as Gaussian and non-Gaussian noises. Before focusing on the heat conduction applications of interest, the state estimation problem is defined and the Kalman and Particle filters are described below. Recent applications of the Kalman filter and of the Particle filter by our group can be found in [19-27].

2. State Estimation Problem

In order to define the state estimation problem, consider a model for the evolution of the vector **x** in the form

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}) \tag{1.a}$$

where the subscript $k = 1, 2, \ldots$, denotes a time instant t_k in a dynamic problem. The vector $\mathbf{x} \in R^{n_x}$ is called the *state vector* and contains the variables to be dynamically estimated. This vector advances in accordance with the *state evolution model* given by equation (1.a), where **f** is, in the general case, a non-linear function of the state variables **x** and of the *state noise* vector $\mathbf{v} \in R^{n_v}$.

Consider also that measurements $\mathbf{z}_k \in R^{n_z}$ are available at t_k, $k = 1, 2, \ldots$. The measurements are related to the state variables **x** through the general, possibly non-linear, function **h** in the form

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{n}_k) \tag{1.b}$$

where $\mathbf{n} \in R^{n_n}$ is the *measurement noise*. Equation (1.b) is referred to as the *observation (measurement) model*.

The state estimation problem aims at obtaining information about $\mathbf{x}_k$ based on the state evolution model (1.a) and on the measurements $\mathbf{z}_{1:k} = \{\mathbf{z}_i, i = 1, \ldots, k\}$ given by the observation model (1.b) [1-18].

The *evolution-observation model* given by equations (1.a,b) are based on the following assumptions [1,4]:

(i) The sequence $\mathbf{x}_k$ for $k = 1, 2, \ldots$, is a Markovian process, that is,

$$\pi(\mathbf{x}_k | \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1}) = \pi(\mathbf{x}_k | \mathbf{x}_{k-1}) \tag{2.a}$$

(ii) The sequence $\mathbf{z}_k$ for $k = 1, 2, \ldots$, is a Markovian process with respect to the history of $\mathbf{x}_k$, that is,

$$\pi(\mathbf{z}_k | \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k) = \pi(\mathbf{z}_k | \mathbf{x}_k) \tag{2.b}$$

(iii) The sequence $\mathbf{x}_k$ depends on the past observations only through its own history, that is,

$$\pi(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_{1:k-1}) = \pi(\mathbf{x}_k | \mathbf{x}_{k-1}) \tag{2.c}$$

where $\pi(\mathbf{a}|\mathbf{b})$ denotes the conditional probability of **a** when **b** is given.

In addition, for the evolution-observation model given by equations (1.a,b) it is assumed that for $i \neq j$ the noise vectors $\mathbf{v}_i$ and $\mathbf{v}_j$, as well as $\mathbf{n}_i$ and $\mathbf{n}_j$, are mutually independent and also mutually independent of the initial state $\mathbf{x}_0$. The vectors $\mathbf{v}_i$ and $\mathbf{n}_j$ are also mutually independent for all *i* and *j* [1].

Different problems can be considered with the above evolution-observation model, namely [1]:

(i) The *prediction problem*, concerned with the determination of $\pi(\mathbf{x}_k|\mathbf{z}_{1:k-1})$;

(ii) The *filtering problem*, concerned with the determination of $\pi(\mathbf{x}_k|\mathbf{z}_{1:k})$;

(iii) The *fixed-lag smoothing problem*, concerned with the determination of $\pi(\mathbf{x}_k|\mathbf{z}_{1:k+p})$, where $p \geq 1$ is the fixed lag;

(iv) The *whole-domain smoothing problem*, concerned with the determination of $\pi(\mathbf{x}_k|\mathbf{z}_{1:K})$, where $\mathbf{z}_{1:K} = \{\mathbf{z}_i, i = 1,\ldots,K\}$ is the complete sequence of measurements.

This paper deals only with the filtering problem. By assuming that $\pi(\mathbf{x}_0|\mathbf{z}_0) = \pi(\mathbf{x}_0)$ is available, the posterior probability density $\pi(\mathbf{x}_k|\mathbf{z}_{1:k})$ is then obtained with Bayesian filters in two steps [1-18]: *prediction and update*, as illustrated in figure 1. The Kalman filter and the particle filter used in this tutorial are discussed below.

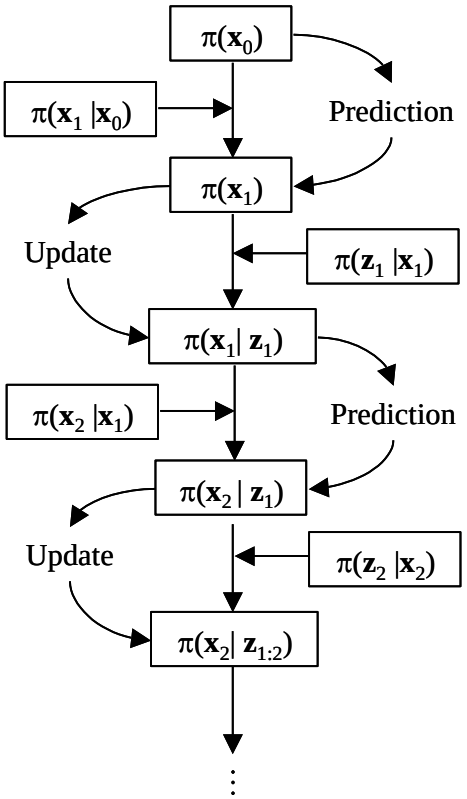


Figure 1. Prediction and update steps for the Bayesian filter [1]

3. The Kalman Filter

For the application of the Kalman filter it is assumed that the evolution and observation models given by equations (1.a,b) are linear. Also, it is assumed that the noises in such models are Gaussian, with known means and covariances, and that they are additive. Therefore, the posterior density $\pi(\mathbf{x}_k \,|\mathbf{z}_{1:k})$ at t_k, $k = 1, 2, \ldots$ is Gaussian and the Kalman filter results in the *optimal solution* to the state estimation problem, that is, the posterior density is calculated exactly [1,2,4-9]. With the foregoing hypotheses, the evolution and observation models can be written respectively as:

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{s}_k + \mathbf{v}_{k-1} \tag{3.a}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{n}_k \tag{3.b}$$

where **F** and **H** are known matrices for the linear evolutions of the state **x** and of the observation **z**, respectively, and **s** is a known vector of inputs. By assuming that the noises **v** and **n** have zero means and covariance matrices **Q** and **R**, respectively, the prediction and update steps of the Kalman filter are given by [1,2,4-9]:

Prediction:

$$\mathbf{x}_k^- = \mathbf{F}_k \hat{\mathbf{x}}_{k-1} + \mathbf{s}_k \tag{4.a}$$

$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k \tag{4.b}$$

Update:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k \right)^{-1} \tag{5.a}$$

$$\hat{\mathbf{x}}_k = \mathbf{x}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \mathbf{x}_k^-) \tag{5.b}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \mathbf{P}_k^- \tag{5.c}$$

The matrix **K** is called Kalman's gain matrix. Notice above that after predicting the state variable **x** and its covariance matrix **P** with equations (4.a,b), *a posteriori* estimates for such quantities are obtained in the update step with the utilization of the measurements **z**. The superscript "^" above indicates the estimate of the state vector.

For other cases for which the hypotheses of linear Gaussian evolution-observation models are not valid, the use of the Kalman filter does not result in optimal solutions because the posterior density is not analytic. The application of Monte Carlo techniques then appears as the most general and robust approach to non-linear and/or non-Gaussian distributions [1,4,8-18]. This is the case despite the availability of the so-called extended Kalman filter and its variations, which generally involves a linearization of the problem. A Monte Carlo filter is described below.

4. The Particle Filter

The *Particle Filter Method* [1,4,8-18] is a Monte Carlo technique for the solution of the state estimation problem. The particle filter is also known as the bootstrap filter, condensation algorithm, interacting particle approximations and survival of the fittest [8]. The key idea is to represent the required posterior density function by a set of random samples (particles) with associated weights, and to compute the estimates based on these samples and weights. As the number of samples becomes very large, this Monte Carlo characterization becomes an equivalent representation of the posterior probability function, and the solution approaches the optimal Bayesian estimate.

We present below the so-called *Sequential Importance Sampling* (SIS) algorithm for the particle filter, which includes a resampling step at each instant, as described in detail in references [8,9]. The SIS algorithm makes use of an *importance density*, which is a density proposed to represent another one that cannot be exactly computed, that is, the sought posterior density in the present case. Then, samples are drawn from the importance density instead of the actual density.

Let $\{\mathbf{x}_{0:k}^i, i=0,\ldots,N\}$ be the particles with associated weights $\{w_k^i, i=0,\ldots,N\}$ and $\mathbf{x}_{0:k}=\{\mathbf{x}_j, j=0,\ldots,k\}$ be the set of all states up to t_k, where N is the number of particles. The weights are normalized so that $\sum_{i=1}^{N} w_k^i = 1$. Then, the posterior density at t_k can be discretely approximated by:

$$\pi(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N} w_k^i \,\delta\left(\mathbf{x}_{0:k} - \mathbf{x}_{0:k}^i\right) \tag{6.a}$$

where $\delta(.)$ is the Dirac delta function. By taking hypotheses (2.a-c) into account, the posterior density (6.a) can be written as [9]:

$$\pi(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N} w_k^i \,\delta\left(\mathbf{x}_k - \mathbf{x}_k^i\right) \tag{6.b}$$

A common problem with the SIS particle filter is the degeneracy phenomenon, where after a few states all but one particle will have negligible weight [1,4,8-18]. This degeneracy implies that a large computational effort is devoted to updating particles whose contribution to the approximation of the posterior density function is almost zero. This problem can be overcome by increasing the number of particles, or more efficiently by appropriately selecting the importance density as the prior density $\pi(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$. In addition, the use of the resampling technique is recommended to avoid the degeneracy of the particles.

Resampling involves a mapping of the random measure $\{\mathbf{x}_k^i, w_k^i\}$ into a random measure $\{\mathbf{x}_k^{i*}, N^{-1}\}$ with uniform weights. It can be performed if the number of effective particles with large weights falls below a certain threshold number. Alternatively, resampling can also be applied indistinctively at every instant t_k, as in the *Sampling Importance Resampling* (SIR) algorithm described in [8,9] and illustrated by figure 2. Such algorithm can be summarized in the following steps, as applied to the system evolution from t_{k-1} to t_k [8,9]:

Step 1. For i=1,…, N draw new particles $\mathbf{x}_k^i$ from the prior density $\pi(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$ and then use the likelihood density to calculate the correspondent weights $w_k^i = \pi(\mathbf{z}_k|\mathbf{x}_k^i)$.

Step 2. Calculate the total weight $T_w = \sum_{i=1}^{N} w_k^i$ and then normalize the particle weights, that is, for i=1,…, N let $w_k^i = T_w^{-1} w_k^i$.

Step 3. Resample the particles as follows:

Step 3.1. Construct the cumulative sum of weights (CSW) by computing $c_i = c_{i-1} + w_k^i$ for i = 1,…, N, with $c_0 = 0$.

Step 3.2. Let i = 1 and draw a starting point u_1 from the uniform distribution $U[0, N^{-1}]$.

Step 3.3. For j = 1, …, N

- Move along the CSW by making $u_j = u_1 + N^{-1}(j-1)$.
- While $u_j > c_i$ make $i = i + 1$.
- Assign samples $\mathbf{x}_k^j = \mathbf{x}_k^i$.
- Assign weights $w_k^j = N^{-1}$.

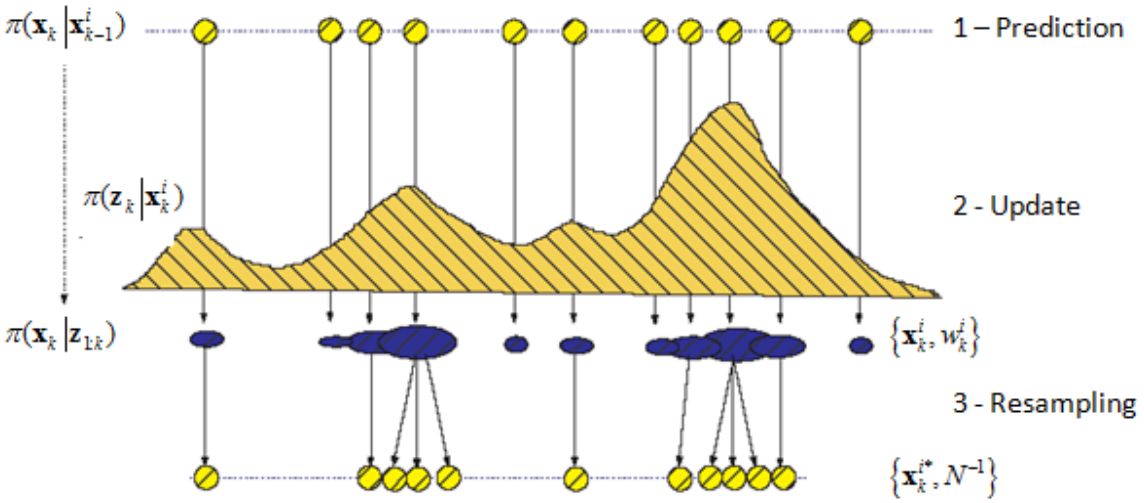


Figure 2. Representation of the *Sampling Importance Resampling* (SIR) algorithm of the Particle filter

Although the resampling step reduces the effects of the degeneracy problem, it may lead to a loss of diversity and the resultant sample can contain many repeated particles. This problem, known as sample impoverishment, can be severe in the case of small process noise. In this situation, all particles collapse to a single particle within few instants t_k [1,8,9,18]. Another drawback of the particle filter is related to the large computational cost due to the Monte Carlo method, which may not allow its application to complicated physical problems. On the other hand, more involved algorithms than the one presented above have been developed, which can reduce the number of particles required for appropriate representation of the posterior density, thus resulting in the reduction of associated computational times. Also, algorithms capable of simultaneously estimating state variables and parameters are available [9,18].

5. Applications

For the sake of understanding the steps required for the application of the Kalman and particle filters, the Appendix A of this Tutorial presents in detail the solution of simple inverse heat transfer problems involving a lumped system. The MATLAB code used for the solution of these problems is also presented in Appendix A. In this session we apply the Bayesian filters described above to the estimation of the transient temperature field in heat conducting media, by using simulated measurements. Linear and non-linear heat conduction problems are addressed, as well as different models for the noises. The problems under study are described below and the results obtained with the Kalman and particle filters are discussed.

5.1. Linear Heat Conduction Problem

Consider heat conduction in a semi-infinite one-dimensional medium, initially at the uniform temperature T^*. The temperature at boundary $x = 0$ is kept at $T = 0$ °C. Physical properties are constant and there is no heat generation in the medium. The formulation for this problem is given by:

$$\frac{1}{\alpha}\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} \quad \text{in } x > 0, \text{ for } t > 0 \tag{7.a}$$

$$T = 0 \quad \text{at } x=0, \text{ for } t > 0 \tag{7.b}$$

$$T = T^* \quad \text{for } t=0, \text{ in } x > 0 \tag{7.c}$$

The analytical solution for this problem is given by [28]:

$$T(x,t) = T^* erf\left(\frac{x}{\sqrt{4\alpha t}}\right) \tag{8}$$

The discretization of equation (7.a) by using explicit finite-differences results in:

$$T_i^{k+1} = rT_{i-1}^k + (1-2r)T_i^k + rT_{i+1}^k \tag{9}$$

where the superscript k denotes the time step, the subscript i denotes the finite-difference node and

$$r = \frac{\alpha \Delta t}{(\Delta x)^2} \tag{10}$$

For the solution of problem (7.a-c) with finite-differences we have to impose a boundary condition at some fictitious surface at $x=L$. We assume L sufficiently large for the time range of interest, so that the finite domain behaves as a semi-infinite medium. Temperature at the surface $x=L$ was assumed to be T^*.

The finite-difference solution for problem (7.a-c) can be written as:

$$\mathbf{T}^{k+1} = \mathbf{F}\mathbf{T}^k + \mathbf{S} \tag{11}$$

where

$$\mathbf{T} = \begin{bmatrix} T_1 \\ \vdots \\ T_{NT} \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} (1-2r) & r & & & & \\ r & (1-2r) & r & & & \\ & \ddots & \ddots & \ddots & & \\ & & r & (1-2r) & r & \\ & & & r & (1-2r) & \end{bmatrix} \qquad \mathbf{S} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ rT^* \end{bmatrix} \tag{12.a-c}$$

In the equations above NT is the number of internal nodes in the finite-difference discretization. Matrix **F** is NTxNT and vectors **T** and **S** are NTx1.

Equation (11) is in appropriate form for the application of the Kalman filter (see equation 3.a). In the present example it is assumed that state and measured variables are the transient temperatures inside the medium at the equidistant finite difference nodes. Therefore, matrix **H** for the observation model (see equation 3.b) is the identity matrix.

The medium is considered to be concrete, with thermal diffusivity a = 4.9x10^{-7} m^2/s. The standard deviation for the measurement errors is considered constant and equal to 2 ^{o}C. The effects of the standard deviation of the errors in the state model are examined below. Figures 3.a and 3.b present the exact temperatures and the measured temperatures in the region, respectively. The final time is taken as 250 seconds and measurements are supposed available in the region every 1 second. The thickness of the medium is considered to be L = 0.1 m and the region is discretized with NT = 50 internal nodes.

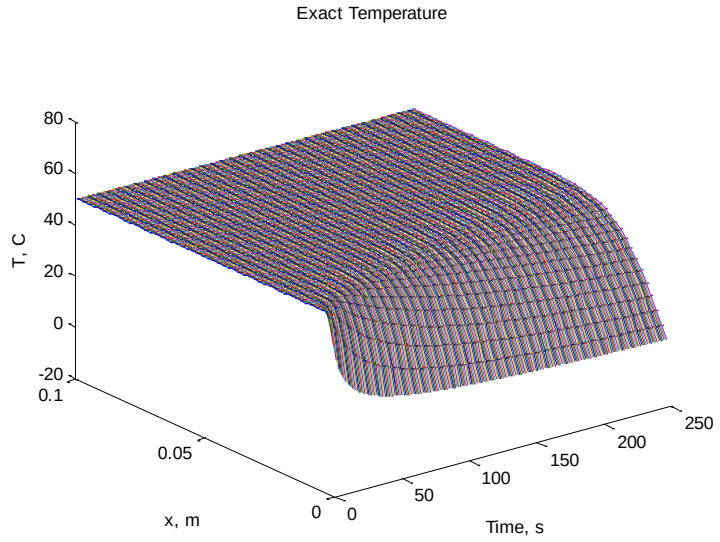


Figure 3.a – Exact temperatures

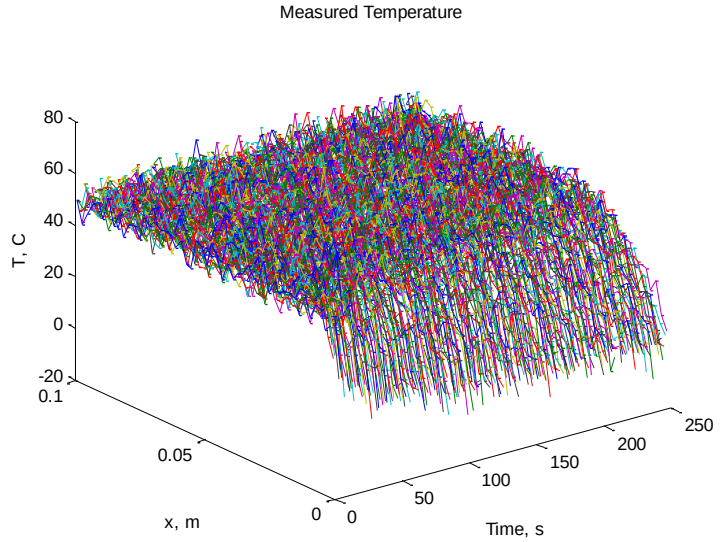
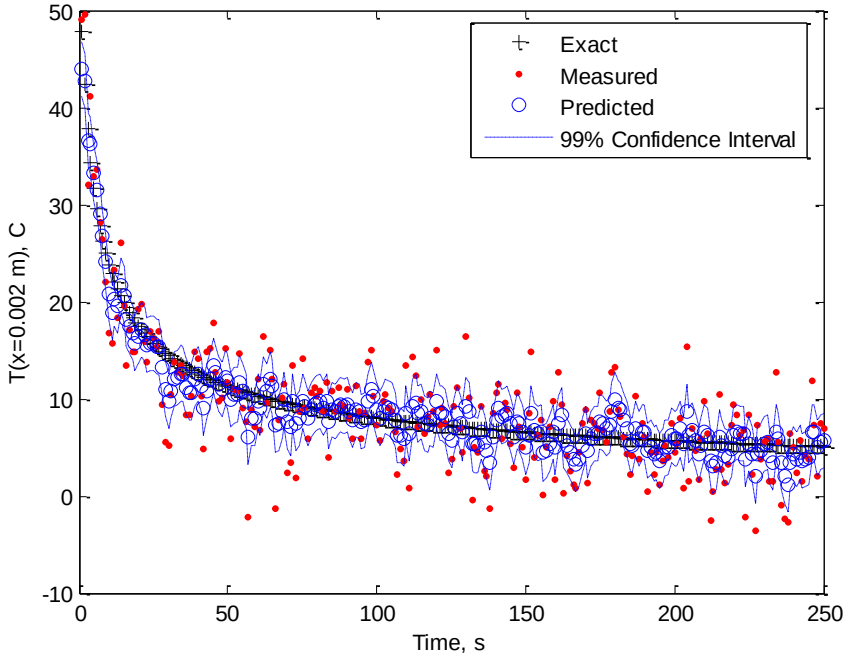
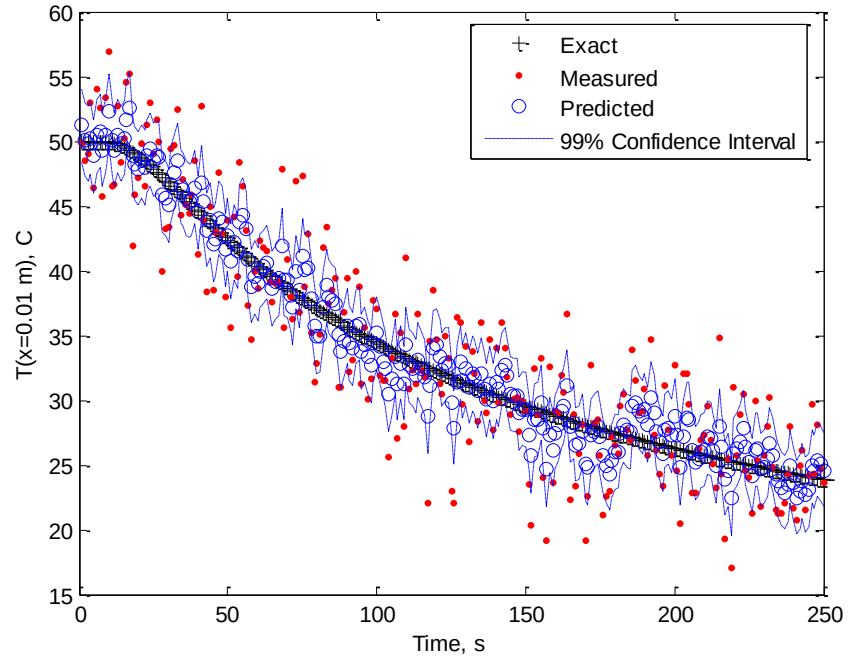


Figure 3.b – Measured temperatures containing Gaussian errors with standard deviation of 2 ^{o}C

Figures 4.a,b present a comparison of exact, measured and predicted temperatures at positions x = 0.002 m and x = 0.01 m, respectively. The predicted temperatures were obtained with the Kalman filter. The 99% confidence intervals for the predicted temperatures are also presented in these figures. The results presented in figures 4.a,b were obtained with a standard deviation for the evolution model errors of 1 ^{o}C. These figures clearly show a great improvement in the predicted temperatures as compared to the measured temperatures at different positions inside the medium. Note that predicted temperatures are much closer to the exact ones than the measurements. In fact, if the model errors are reduced, the predicted temperatures tend to follow the model more closely. This is exemplified in figures 5.a,b, where the standard deviation of the evolution model errors was reduced to 0.5 ^{o}C. On the other hand, if such errors are large, the predictions of the Kalman filter tend to follow the measurements instead of the model. Such a fact is clear from the analysis of figures 6.a,b, which present the results for a standard deviation for the evolution model errors of 2 ^{o}C.

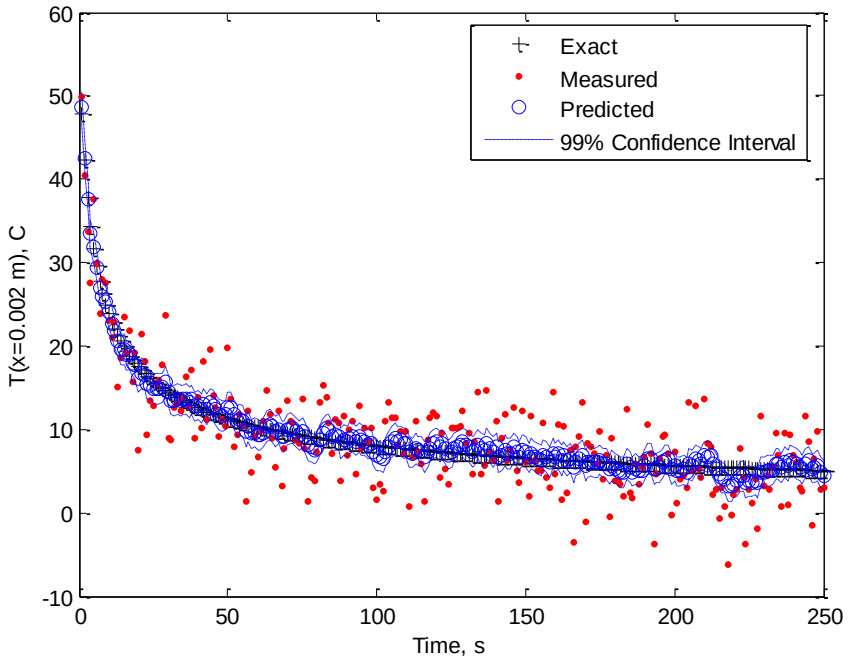


(a)

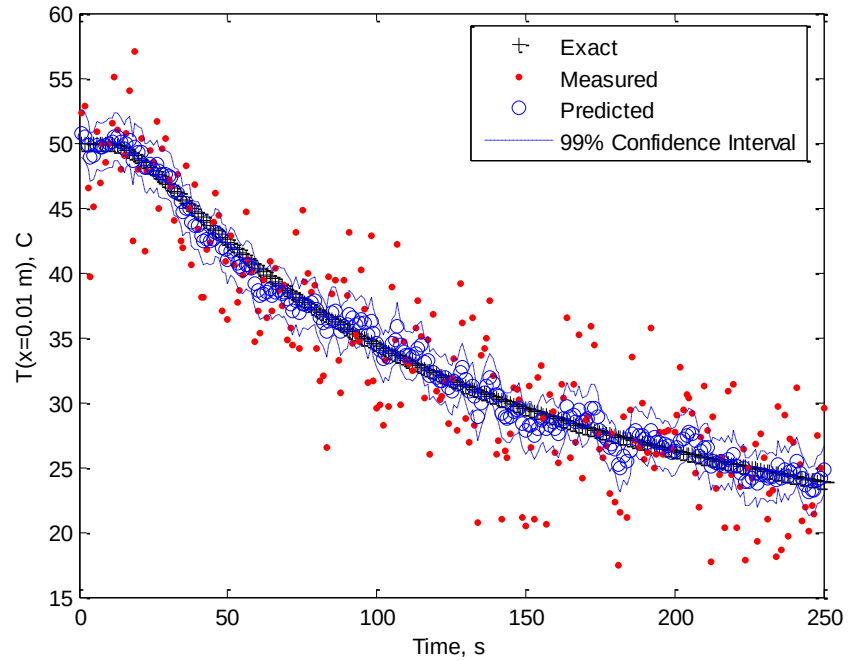


(b)

Figure 4 – Temperature at (a) x = 0.002 m and (b) x = 0.01 m – standard deviation for the evolution model errors of 1 ^{o}C – Kalman filter

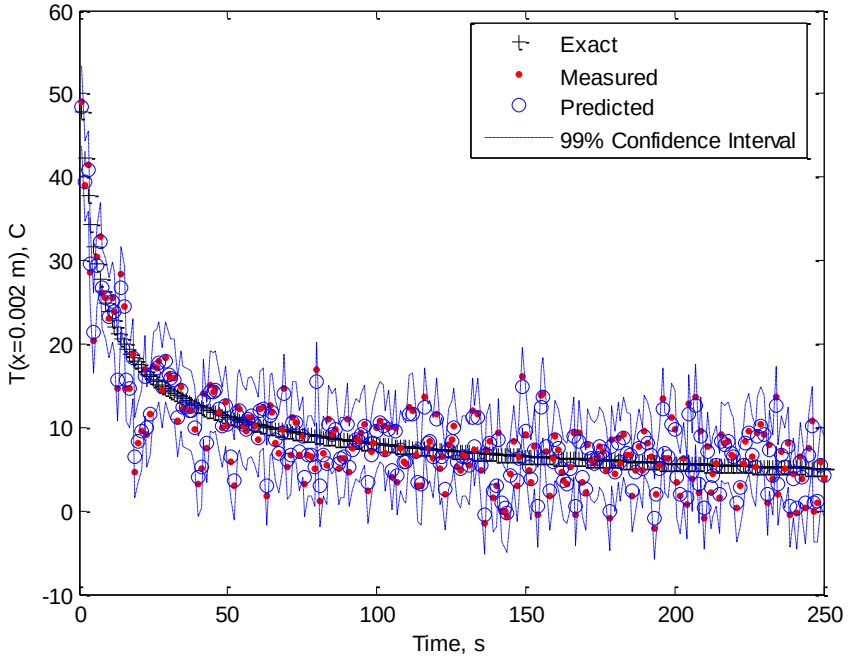


(a)

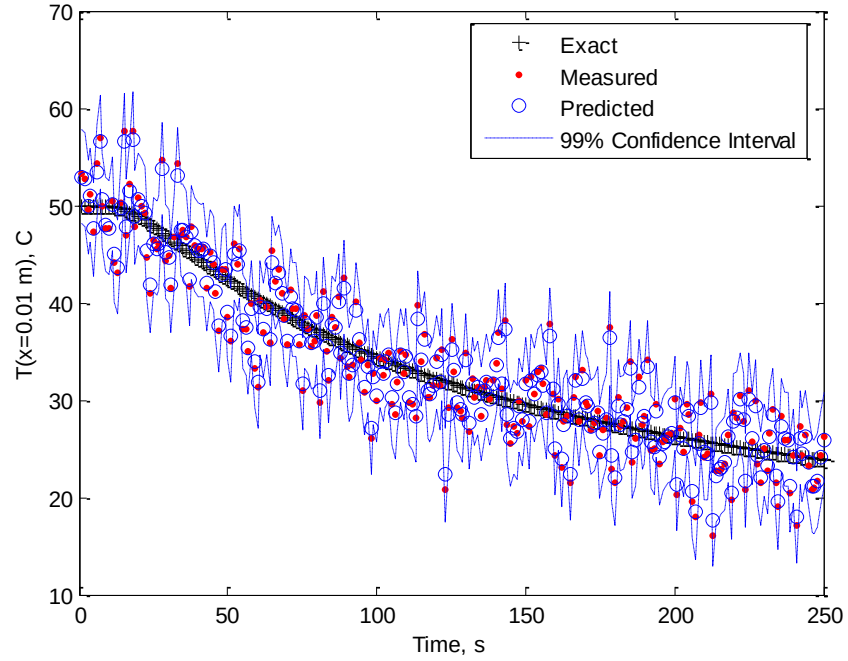


(b)

Figure 5 – Temperature at (a) x = 0.002 m and (b) x = 0.01 m – standard deviation for the evolution model errors of 0.5 ^{o}C – Kalman filter



(a)

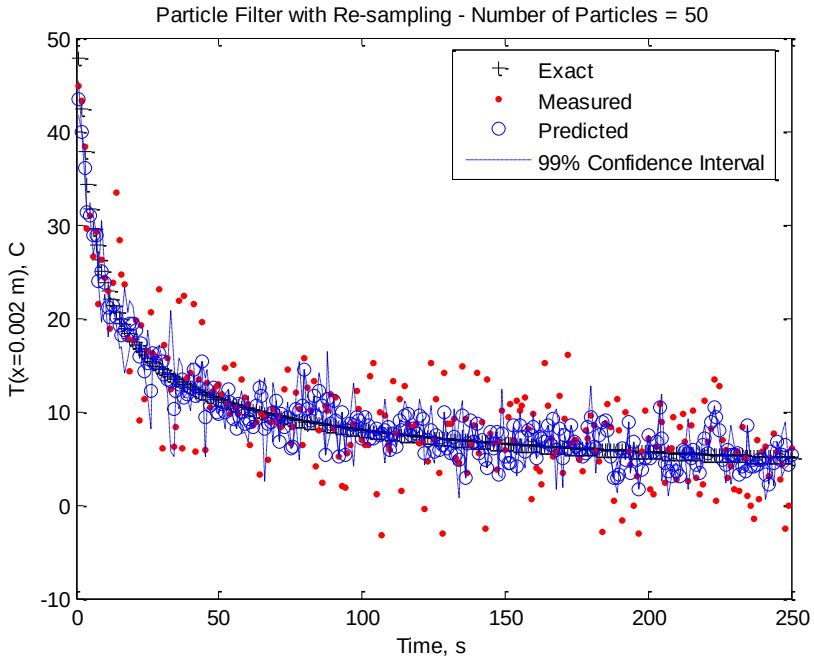


(b)

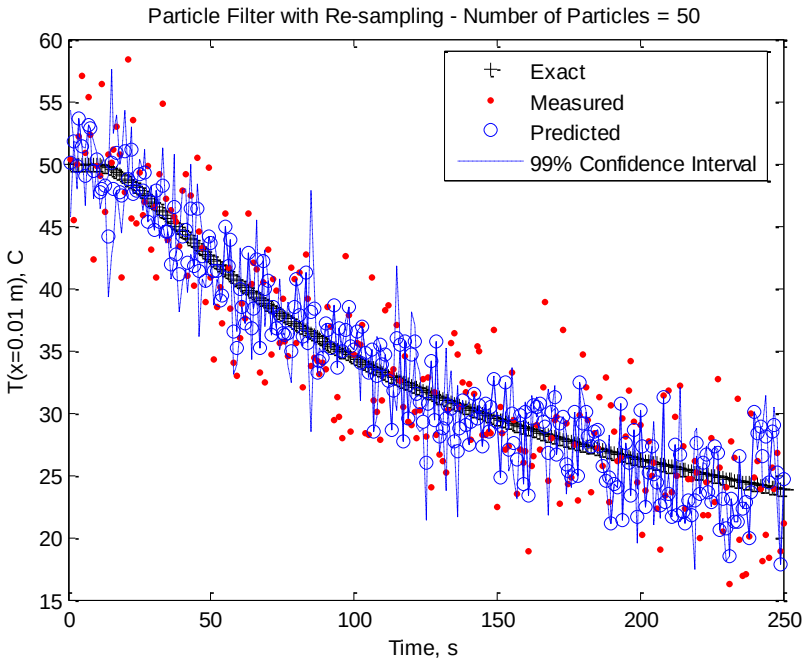
Figure 6 – Temperature at (a) x = 0.002 m and (b) x = 0.01 m – standard deviation for the evolution model errors of 5 ^{o}C – Kalman filter

The application of the particle filter with the sampling importance resampling algorithm described above is presented in figures 7.a,b. These figures present the exact, measured and predicted temperatures at positions x = 0.002 m and x = 0.01 m, respectively. Fifty particles were used in this

case. A comparison of figures 4.a,b and 7.a,b show that the particle filter, similarly to the Kalman filter, provided accurate predictions for the temperature in the region. However, as expected, the computational cost of the particle filter was substantially larger than that of the Kalman filter. For this example, the predictions obtained with the Kalman filter took 0.5 seconds and those obtained with the particle filter took 6.9 seconds on a 1.8 GHz Centrino Duo computer with 1 Mbyte of RAM memory.



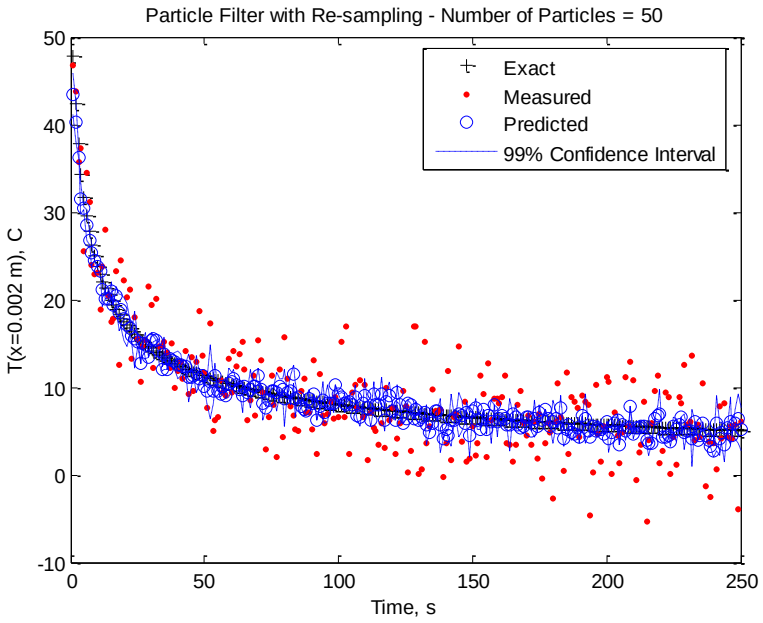
(a)



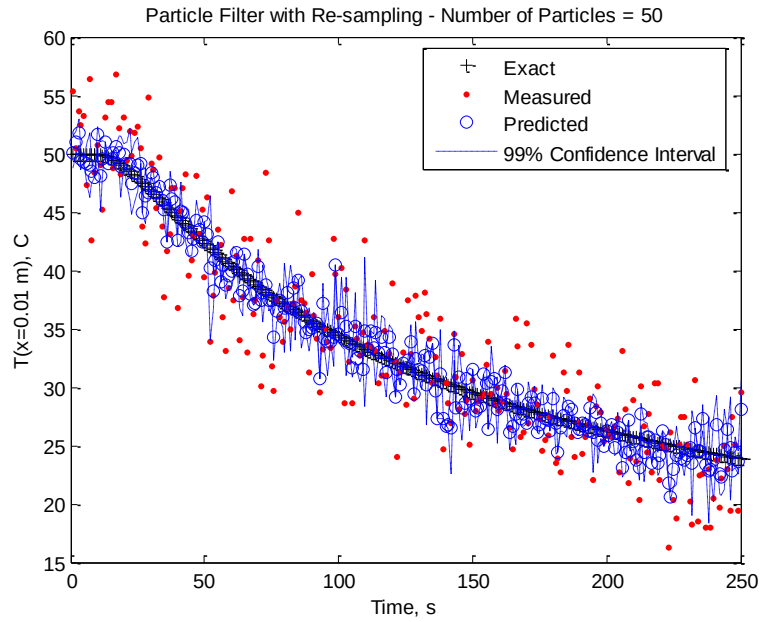
(b)

Figure 7 – Temperature at (a) x = 0.002 m and (b) x = 0.01 m – standard deviation for the evolution model errors of 1 ^{o}C – Particle filter (SIR)

We now examine a case involving uniformly distributed errors in the evolution model, instead of Gaussian errors. For such case, the application of the Kalman filter does not result in optimal solutions. Consequently, only the particle filter was considered for the prediction of the temperatures in the region. The results obtained with errors in the evolution model uniformly distributed in the interval [-1,1] ^{o}C are presented in figures 8.a,b for x = 0.002 m and x = 0.01 m, respectively. These figures show that the particle filter was not affected by the distribution of the errors and results similar to those obtained for the Gaussian distribution (see figures 7.a,b) were obtained.



(a)



(b)

Figure 8 – Temperature at (a) x = 0.002 m and (b) x = 0.01 m – evolution model errors having uniform distribution in [-1,1] ^{o}C – Particle filter (SIR)

5.2. Non-linear Heat Conduction Problem

Consider heat conduction in a one-dimensional medium with thickness L, initially at the uniform temperature T^*. The boundary at $x = 0$ is kept insulated and a constant heat flux q^* is imposed at $x = L$. Thermophysical properties are temperature dependent and there is no heat generation in the medium. The formulation for this problem is given by:

$$C(T)\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left[k(T)\frac{\partial T}{\partial x}\right] \quad \text{in } 0 < x < L, \text{ for } t > 0 \tag{13.a}$$

$$\frac{\partial T}{\partial x} = 0 \quad \text{at } x=0, \text{ for } t > 0 \tag{13.b}$$

$$k(T)\frac{\partial T}{\partial x} = q^* \quad \text{at } x=L, \text{ for } t > 0 \tag{13.c}$$

$$T = T^* \quad \text{for } t=0, \text{ in } x > 0 \tag{13.d}$$

We examine here a physical problem involving the heating of a graphite sample with an oxy-acetylene torch [19]. The temperature dependence of the thermal conductivity, $k(T)$, and volumetric heat capacity, $C(T)$, of the graphite, measured for different temperatures, were curve-fitted (see figure 9) with exponentials of the form:

$$C(T) = A_1 + A_2 e^{-T/A_3} \tag{14.a}$$

$$k(T) = B_1 + B_2 e^{-T/B_3} \tag{14.b}$$

with the adjusted parameters A_1, A_2, A_3, B_1, B_2 and B_3 given in table 1.

Table 1. Parameters of the curve-fitted thermophysical properties

Parameter	**Mean**
A_1 ($Jm^{-3}\,^oC^{-1}$)	5,681,006
A_2 ($Jm^{-3}\,^oC^{-1}$)	−4,813,057
A_3 (^oC)	547.00
B_1 ($Wm^{-1}\,^oC^{-1}$)	24.52
B_2 ($Wm^{-1}\,^oC^{-1}$)	183.05
B_3 (^oC)	277.00

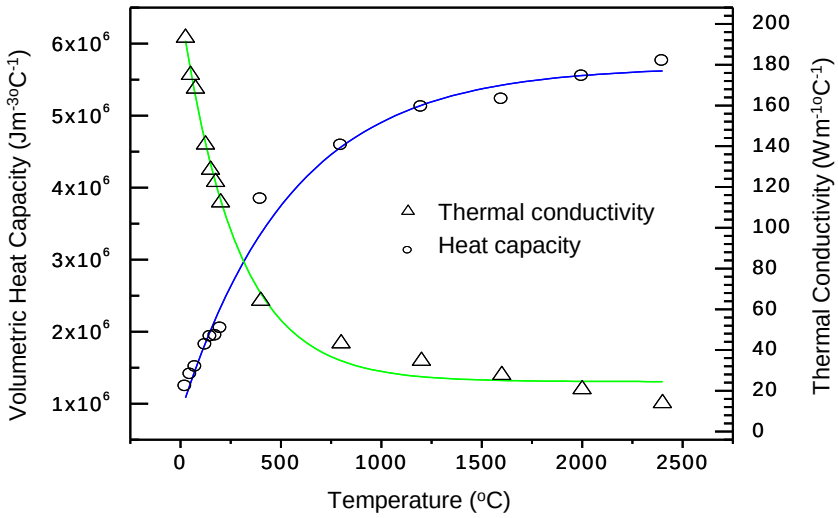


Figure 9. Temperature-dependent thermophysical properties

The sample was supposed to be initially at the room temperature of $T^* = 20\ ^oC$ and the imposed heat flux was $q^* = 10^5\ W/m^2$. The thickness of the slab was taken as $L = 0.01$ m. Finite volumes were used for the solution of this nonlinear heat conduction problem, with the region discretized with 50 equal volumes. The final time was supposed to be 90 seconds and the time step was taken as 1 second. The errors in the state evolution and observation models were supposed to be additive, Gaussian, uncorrelated, with zero mean and constant standard deviations.

In order to examine a very strict case, the standard deviation for the state evolution model was set to 5 ^oC and for the observation model as 10 ^oC. Figures 10.a and 10.b present the exact and measured temperatures, respectively, for such non linear heat conduction problem. It is apparent from the analysis of these figures that extremely large measurement errors were considered for this case.

State evolution and measurement models were used for the prediction of the temperature variation in the region. Due to the nonlinear character of the problem under consideration, only the particle filter, in the form of the SIR algorithm described above, was used for the predictions. Fifty particles were used in such filtering algorithm.

Figures 11.a-d present a comparison of exact, measured and predicted temperatures at the positions $x = 0.0002$ m, 0.001 m, 0.002 m and 0.004 m, respectively. An analysis of these figures reveals the robustness of the particle filter. Despite the very large errors in the evolution and observation models, the predicted average temperatures are in excellent agreement with the exact values. Furthermore, the 99% confidence intervals for the predicted temperatures are significantly smaller than the dispersion of the measurements. Differently from the case analyzed above involving linear heat conduction, sample impoverishment is not observed in figures 11.a-d. This is probably due to the large noise in the state model used for the nonlinear case. The computational time for this case was 61.3 seconds.

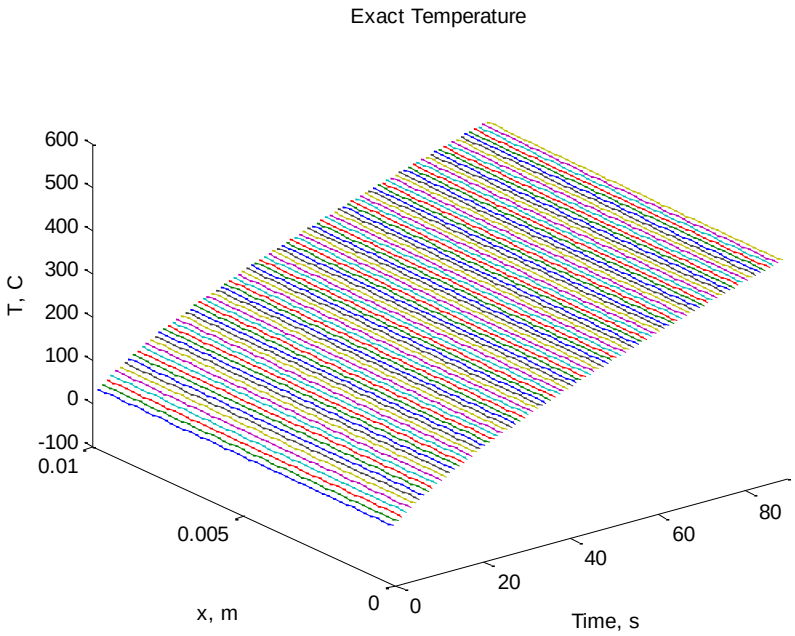


Figure 10.a – Exact temperatures

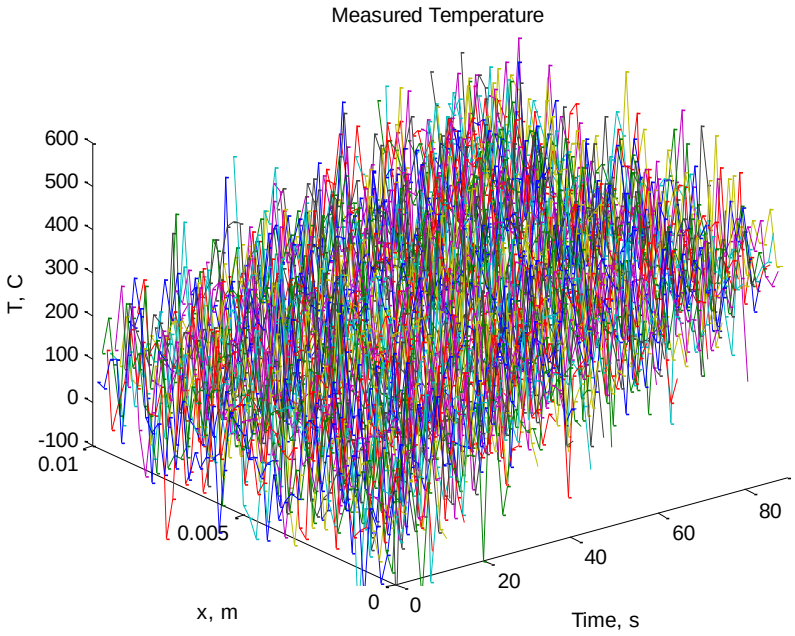


Figure 10.b – Measured temperatures containing Gaussian errors with standard deviation

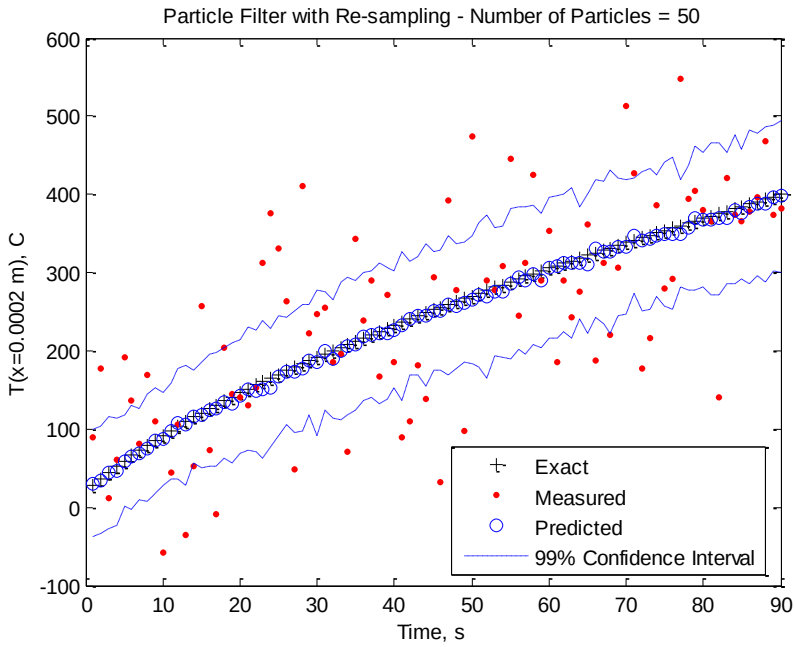


Figure 11.a – Temperature at x = 0.0002 m for the nonlinear heat conduction problem – Particle filter

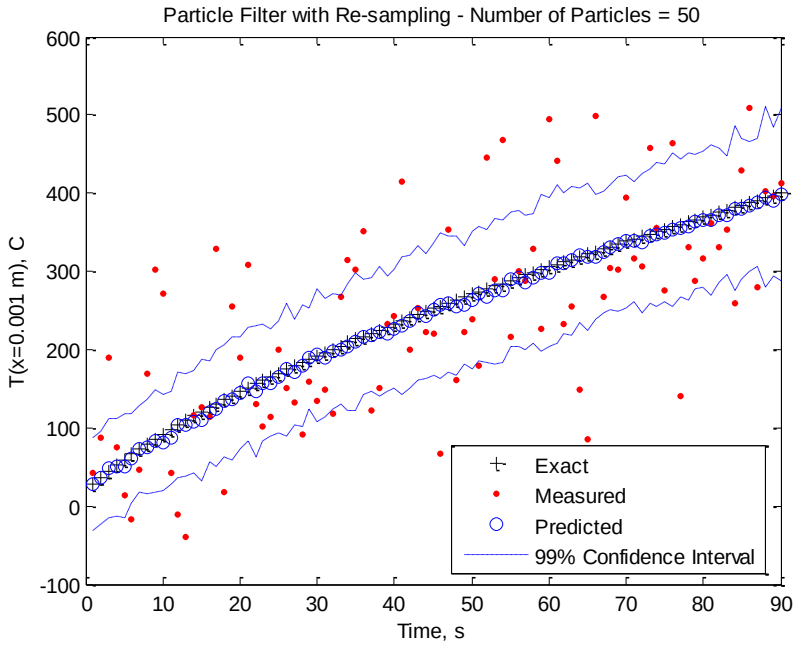


Figure 11.b – Temperature at x = 0.001 m for the nonlinear heat conduction problem – Particle filter

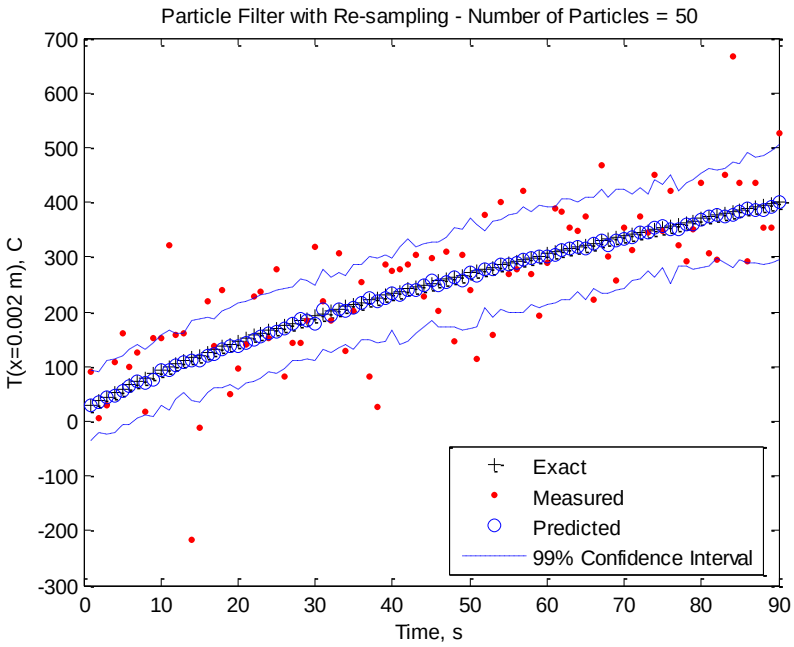


Figure 11.c – Temperature at x = 0.002 m for the nonlinear heat conduction problem – Particle filter

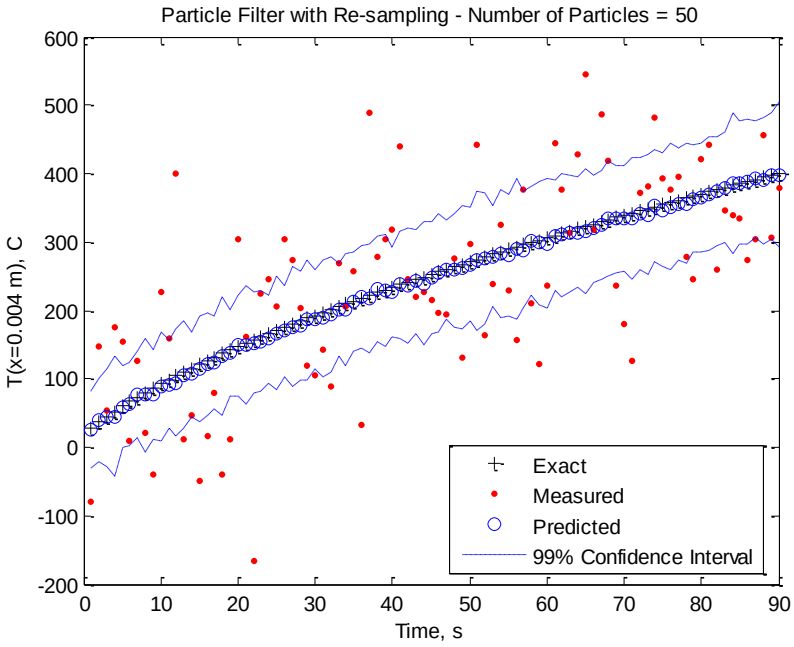


Figure 11.d – Temperature at x = 0.004 m for the nonlinear heat conduction problem – Particle filter

6. Conclusions

This tutorial dealt with the application of the Kalman filter and of the particle filter to the estimation of the transient temperature field in linear and non-linear heat conduction problems. The particle filter was coded in the form of the sampling importance resampling (SIR) algorithm. For linear-Gaussian models, the Kalman filter and the particle filter provided results of similar accuracy. However, the computational cost of the particle filter was significantly larger than that of the Kalman filter. On the other hand, in non-linear and/or non-Gaussian models the basic hypotheses required for the application of the Kalman filter are not valid. The particle filter appears in the literature as an accurate estimation technique of general use, including for non-linear and/or non-Gaussian models. The particle filter was successfully applied above to a non-linear heat conduction problem. Such Monte Carlo technique provided accurate estimation results, even for a strict test-case involving very large errors in the evolution and observation models.

We note that more involved algorithms than the one presented above have been developed for the particle filter, which can reduce the number of particles required for appropriate representation of the posterior density, thus resulting in the reduction of associated computational times. Also, algorithms capable of simultaneously estimating state variables and parameters are available (see [9,18]).

7. Acknowledgements

The financial support provided by FAPERJ, CAPES and CNPq, Brazilian agencies for the fostering of science, is greatly appreciated

8. References

1. Kaipio, J. and Somersalo, E., 2004, *Statistical and Computational Inverse Problems*, Applied Mathematical Sciences 160, Springer-Verlag.
2. Maybeck, P., 1979, *Stochastic models, estimation and control*, Academic Press, New York.
3. Winkler, R., 2003, *An Introduction to Bayesian Inference and Decision*, Probabilistic Publishing, Gainsville.
4. Kaipio, J., Duncan S., Seppanen, A., Somersalo, E., Voutilainen, A., 2005, *State Estimation for Process Imaging*, Chapter in *Handbook of Process Imaging for Automatic Control*, editors: David Scott and Hugh McCann, CRC Press.
5. Kalman, R., 1960, A New Approach to Linear Filtering and Prediction Problems, *ASME J. Basic Engineering*, vol. 82, pp. 35-45.
6. Sorenson, H., 1970, Least-squares estimation: from Gauss to Kalman, *IEEE Spectrum*, vol. 7, pp. 63-68.
7. Welch, G. and Bishop, G., 2006, *An Introduction to the Kalman Filter*, UNC-Chapel Hill, TR 95-041.
8. Arulampalam, S., Maskell, S., Gordon, N., Clapp, T., 2001, A Tutorial on Particle Filters for on-line Non-linear/Non-Gaussian Bayesian Tracking, *IEEE Trans. Signal Processing*, vol. 50, pp. 174-188.
9. Ristic, B., Arulampalam, S., Gordon, N., 2004, *Beyond the Kalman Filter*, Artech House, Boston.
10. Doucet, A., Godsill, S., Andrieu, C., 2000, On sequential Monte Carlo sampling methods for Bayesian filtering, *Statistics and Computing*, vol. 10, pp. 197-208.
11. Liu, J and Chen, R., 1998, Sequential Monte Carlo methods for dynamical systems, *J. American Statistical Association*, vol. 93, pp. 1032-1044.
12. Andrieu, C., Doucet, A., Robert, C., 2004, Computational advances for and from Bayesian analysis, *Statistical Science*, vol. 19, pp. 118-127.
13. Johansen, A. Doucet, A., 2008, A note on auxiliary particle filters, *Statistics and Probability Letters*, to appear.
14. Carpenter, J., Clifford, P., Fearnhead, P, 1999, An improved particle filter for non-linear problems, *IEEE Proc. Part F: Radar and Sonar Navigation*, vol. 146, pp. 2-7.

15. Del Moral, P., Doucet, A., Jasra, A., 2007, Sequential Monte Carlo for Bayesian Computation, *Bayesian Statistics*, vol. 8, pp. 1-34.
16. Del Moral, P., Doucet, A., Jasra, A., 2006, Sequential Monte Carlo samplers, *J. R. Statistical Society*, vol. 68, pp. 411-436.
17. Andrieu, C., Doucet, Sumeetpal, S., Tadic, V., 2004, Particle methods for charge detection, system identification and control, *Proceedings of IEEE*, vol. 92, pp. 423-438.
18. Doucet, A., Freitas, N and Gordon, N., 2001, *Sequential Monte Carlo Methods in Practice*, Springer, New York.
19. Orlande, H., Dulikravich, G. and Colaço, M., 2008, Application of Bayesian Filters to Heat Conduction Problem, *EngOpt 2008 – International Conference on Eng. Optimization*, (ed: Herskovitz), June 1-5, Rio de Janeiro, Brazil.
20. Silva, W. B. Orlande, H. R. B.; Colaço, M. J., 2010, .Evaluation of Bayesian Filters Applied to Heat Conduction Problems, *2nd International Conference on Engineering Optimization*, Lisboa.
21. Silva, W. B. Orlande, H. R. B.; Colaço, M. J., Fudym, O., 2011, Application Of Bayesian Filters To A One-Dimensional Solidification Problem, *21st Brazilian Congress of Mechanical Engineering*, Natal, RN, Brazil.
22. Colaço, M. J., Orlande, H. R. B.; Silva, W. B., Dulikravich, G., 2011, Application Of A Bayesian Filter To Estimate Unknown Heat Fluxes In A Natural Convection Problem, *ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, *IDETC/CIE 2011*, August 29-31, Washington, DC.
23. Fonseca, H., Orlande, H. R. B., Fudym, O., Sepulveda, F., 2010, Kalman Filtering For Transient Source Term Mapping From Infrared Images, *Inverse Problems, Design And Optimization Symposium*, João Pessoa.
24. Vianna, F., Orlande, H. R. B., Dulikravich, G., 2010, Optimal Heating Control To Prevent Solid Deposits In Pipelines, *V European Conference On Computational Fluid Dynamics - ECCOMAS CFD* 2010
25. Vianna, F., Orlande, H. R. B., Dulikravich, G., 2010, Pipeline Heating Method Based On Optimal Control And State Estimation, *Brazilian Congress Of Thermal Sciences And Engineering*, Uberlândia.
26. Vianna, F., Orlande, H. R. B., Dulikravich, G, 2010, Temperature Field Prediction Of A Multilayered Composite Pipeline Based On The Particle Filter Method, *14th International Heat Transfer Conference*, Washington.
27. Vianna, F., Orlande, H. R. B., Dulikravich, G, 2009, Estimation Of The Temperature Field In Pipelines By Using The Kalman Filter, *2nd International Congress of Serbian Society of Mechanics* (IConSSM 2009)
28. Ozisik, M., 1993, *Heat Conduction*, Wiley, New York.

Appendix A

In this appendix, for didactical purposes, we present the solution of a simple state estimation problem in heat transfer, involving a lumped system. The problem is formulated and solved below, by using the Kalman filter and the Particle filter, coded with the SIR algorithm. Two different cases are examined and a MATLAB code is included.

Physical Problem and Mathematical Formulation

We consider a slab of thickness L, initially at the uniform temperature T_0, which is subjected to a uniform heat flux $q(t)$ over one of its surfaces, while the other exchanges heat by convection and linearized radiation with a medium at a temperature T_∞ with a heat transfer coefficient h (see figure A.1). Temperature gradients are neglected inside the slab and a lumped formulation is used. The formulation for this problem is given by [1]:

$$\frac{d\theta(t)}{dt} + m\theta(t) = \frac{mq(t)}{h} \qquad \text{for } t > 0 \qquad \text{(A.1.a)}$$

$$\theta = \theta_0 \qquad \text{for } t = 0 \qquad \text{(A.1.b)}$$

where

$$\theta(t) = T(t) - T_\infty \qquad \text{(A.2.a)}$$

$$\theta_0 = T_0 - T_\infty \qquad \text{(A.2.b)}$$

$$m = \frac{h}{\rho c L} \qquad \text{(A.2.c)}$$

and ρ is the density and c is the specific heat of the homogeneous material of the slab.

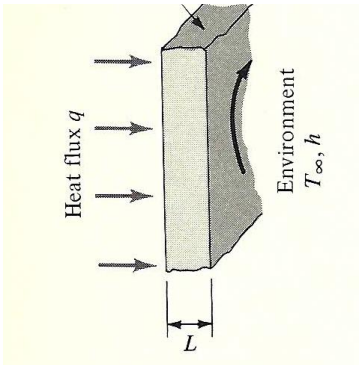


Figure A.1. Physical problem [1]

Two illustrative cases are now examined, namely: (i) Heat Flux $q(t) = q_0$ constant and deterministically known; and (ii) Heat Flux $q(t) = q_0 f(t)$ with unknown time variation. Results are obtained for these two cases, by assuming that the plate is made of aluminum ($\rho = 2707$ kgm^{-3}, $c =$ 896 Jkg^{-1}K^{-1}), with thickness $L = 0.03$ m, $q_0 = 8000$ Wm^{-2}, $T_\infty = 20$ ^{o}C, $h = 50$ Wm^{-2}K^{-1} and $T_0 = 50$ ^{o}C [1]. Measurements of the transient temperature of the slab are assumed available. These measurements contain additive, uncorrelated, Gaussian errors, with zero mean and a constant standard deviation σ_z. The errors in the state evolution model are also supposed to be additive, uncorrelated, Gaussian, with zero mean and a constant standard deviation σ_θ.

(i) Heat Flux $q(t) = q_0$ constant and deterministically known

The analytical solution for this problem is given by:

$$\theta(t) = \theta_0 e^{-mt} + \frac{q_0}{h}(1 - e^{-mt}) \tag{A.3}$$

The only state variable in this case is the temperature $\theta(t_k) = \theta_k$ since the applied heat flux q_0 is constant and deterministically known, as the other parameters appearing in the formulation. By using a forward finite-differences approximation for the time derivative in equation (A.1.a), we obtain:

$$\theta_k = (1 - m\Delta t)\theta_{k-1} + \frac{mq_0}{h}\Delta t \tag{A.4}$$

Therefore, the state and observation models given by equations (3.a,b) are obtained with:

$$\mathbf{x}_k = [\theta_k] \qquad \mathbf{F}_k = [(1 - m\Delta t)] \qquad \mathbf{s}_k = \left[m\frac{q_0}{h}\Delta t \right] \qquad \mathbf{H}_k = [1] \quad \mathbf{Q}_k = [\sigma_\theta^2] \qquad \mathbf{R}_k = [\sigma_z^2] \tag{A5.a-f}$$

Figure A.2 presents a comparison of the analytical and numerical solutions given by equations (A.3) and (A.4), respectively, for Δt = 1 s. The agreement between these two solutions is perfect in the graph scale. Simulated measurements with a standard deviation of σ_z = 1 °C are also shown in this figure.

Figures A.3 and A.4 present the estimated temperatures obtained with the Kalman filter and the particle filter, respectively, for a standard deviation of the evolution model σ_θ = 1 °C. These two figures illustrate the capabilities of both filters examined here; the results obtained with them are similar and they are capable of accurately recovering the exact temperature. The number of particles used in the SIR algorithm of the particle filter was 200.

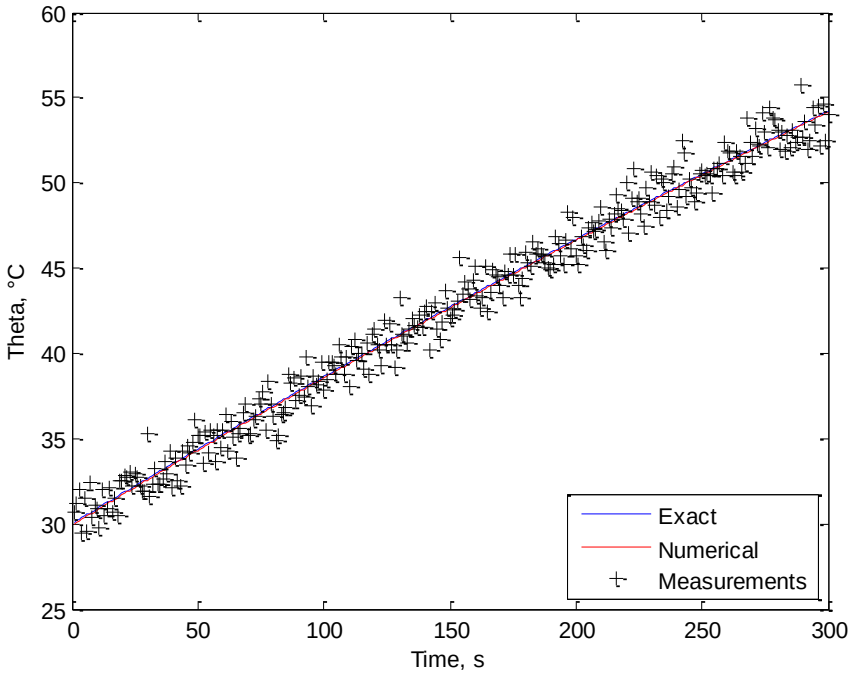


Figure A.2. Simulated measurements and comparison of analytical and numerical solutions

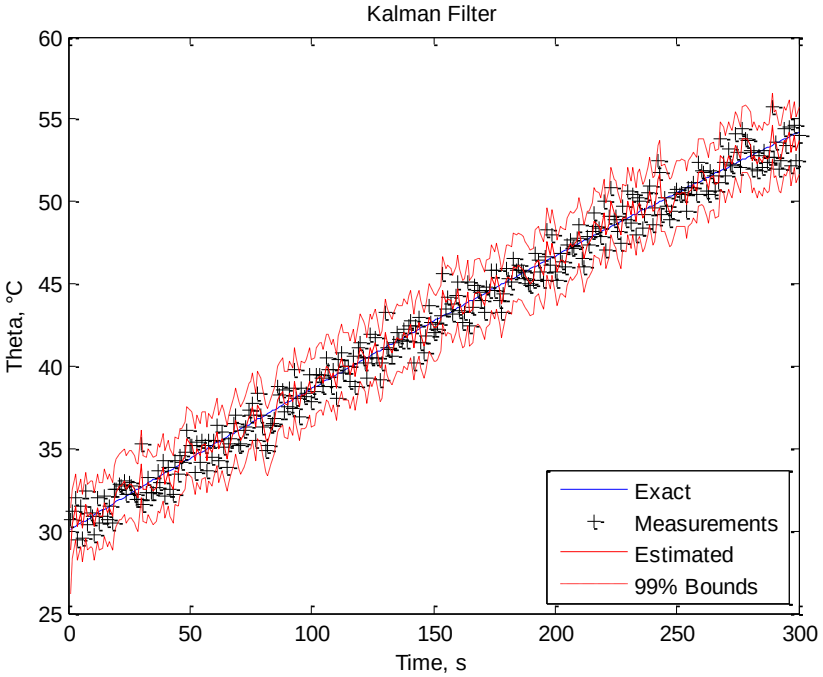


Figure A.3. Solution with the Kalman filter

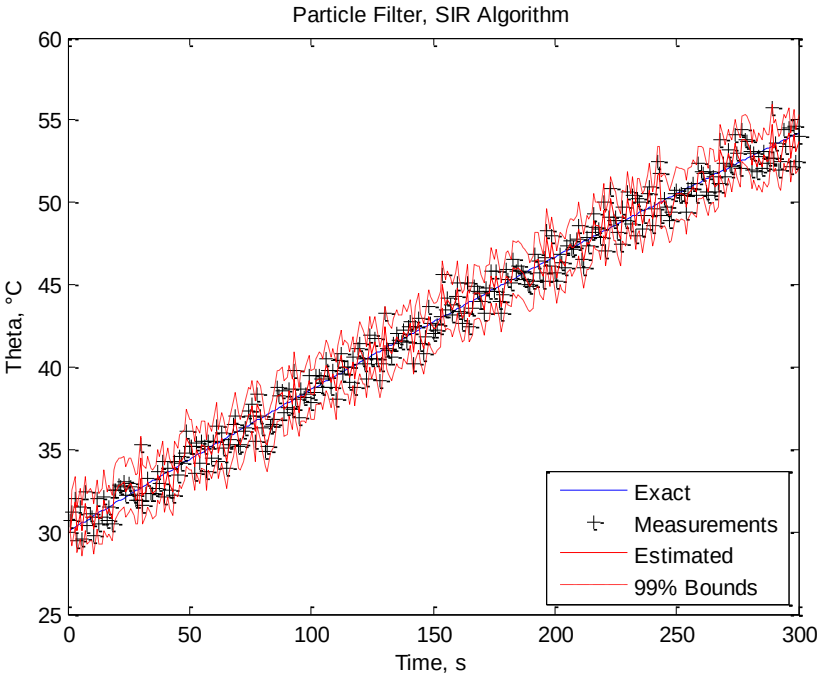


Figure A.4. Solution with the Particle filter

(ii) Heat Flux $q(t) = q_0 f(t)$ with unknown time variation

The analytical solution for this problem is given by:

$$\theta(t) = e^{-mt}\left\{\theta_0 + \frac{mq_0}{h}\int_{t'=0}^{t} e^{mt'} f(t')\,dt'\right\} \tag{A.6}$$

In this case, the state variables are given by the temperature $\theta(t_k) = \theta_k$ and the function that gives the time variation of the applied heat flux, that is, $f(t_k) = f_k$ .As in the case examined above, the applied heat flux q_0 is constant and deterministically known, as the other parameters appearing in the formulation. By using a forward finite-differences approximation for the time derivative in equation (A.1.a), we obtain the equation for the evolution of the state variable $\theta(t_k) = \theta_k$:

$$\theta_k = (1 - m\Delta t)\theta_{k-1} + \left(\frac{mq_0}{h}\Delta t\right) f_{k-1} \tag{A.7}$$

A random walk model is used for the state variable $f(t_k) = f_k$, which is given in the form:

$$f_k = f_{k-1} + \varepsilon_{k-1} \tag{A.8}$$

where ε_{k-1} is Gaussian with zero mean and constant standard deviation σ_{rw}.

Therefore, the state and observation models given by equations (3.a,b) are obtained with:

$$\mathbf{x}_k = \begin{bmatrix}\theta_k \\ f_k\end{bmatrix} \qquad \mathbf{F}_k = \begin{bmatrix}(1 - m\Delta t) & \frac{mq_0}{h}\Delta t \\ 0 & 1\end{bmatrix} \qquad \mathbf{s}_k = \begin{bmatrix}0 \\ 0\end{bmatrix} \tag{A.9.a-c}$$

$$\mathbf{H}_k = [1\ 0] \qquad \mathbf{Q}_k = \begin{bmatrix}\sigma_\theta^2 & 0 \\ 0 & \sigma_{rw}^2\end{bmatrix} \qquad \mathbf{R}_k = [\sigma_z^2] \tag{A.9.d-f}$$

For the cases studied here, two different functions were examined for the time variation of the applied heat flux, specifically, a step function in the form:

$$f(t) = \begin{cases} 1, & 0 < t \le \dfrac{t_{final}}{2} \\ 0, & \dfrac{t_{final}}{2} < t < t_{final} \end{cases} \tag{A.10}$$

and a ramp function in the form

$$f(t) = t \Big/ t_{final} \tag{A.11}$$

Figure A.5 presents a comparison of the analytical and numerical solutions given by equations (A.6) and (A.7), respectively, for the step variation of the heat flux. Figure A.6 presents a similar comparison for the ramp variation of the heat flux. For both cases, we took Δt = 1 s. The agreement between these two solutions for both cases is perfect in the graph scale. Simulated measurements with a standard deviation of σ_z = 1 ^{o}C are also shown in these figures.

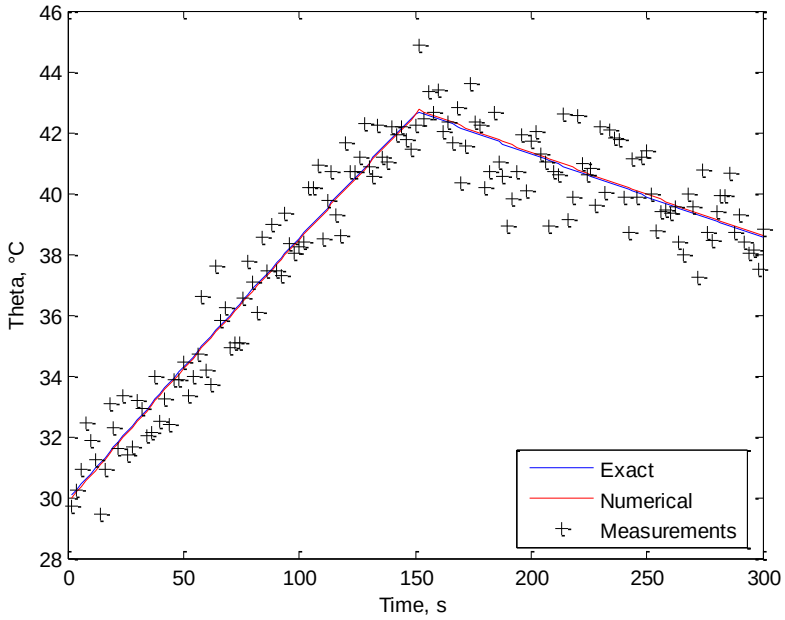


Figure A.5. Simulated measurements and comparison of analytical and numerical solutions for the step function

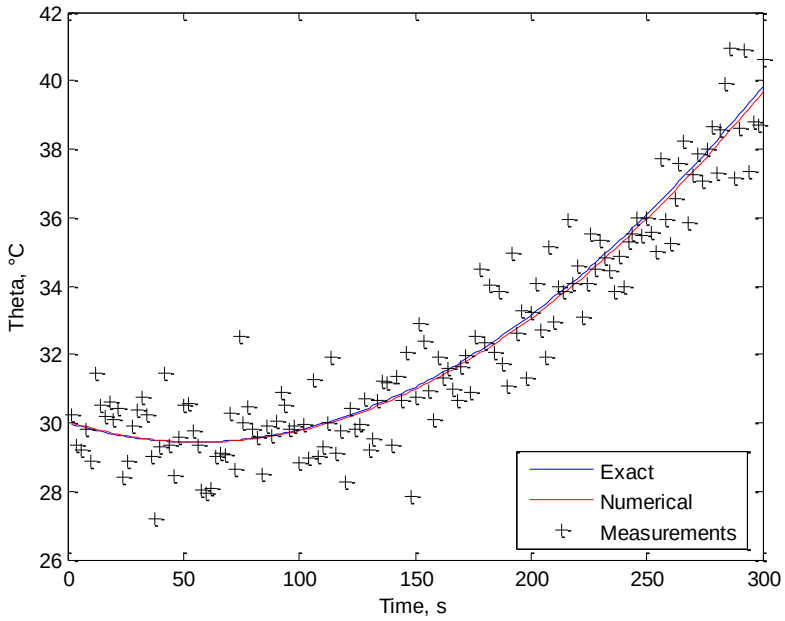


Figure A.6. Simulated measurements and comparison of analytical and numerical solutions for the ramp function

The results presented below were all obtained with $\sigma_z = \sigma_\theta = 1$ °C, while the standard deviation used in the random walk model of $f(t)$ in equation (A.8) was $\sigma_{rw} = 0.3$. The number of particles used in the particle filter was 500.

We first present the results obtained for the step variation of the heat flux. Figures A.7 and A.8 show the estimated temperatures obtained with the Kalman filter and the particle filter, respectively. Figures A.9 and A.10 show the estimated time variation of the heat flux obtained with the Kalman filter and the particle filter, respectively.

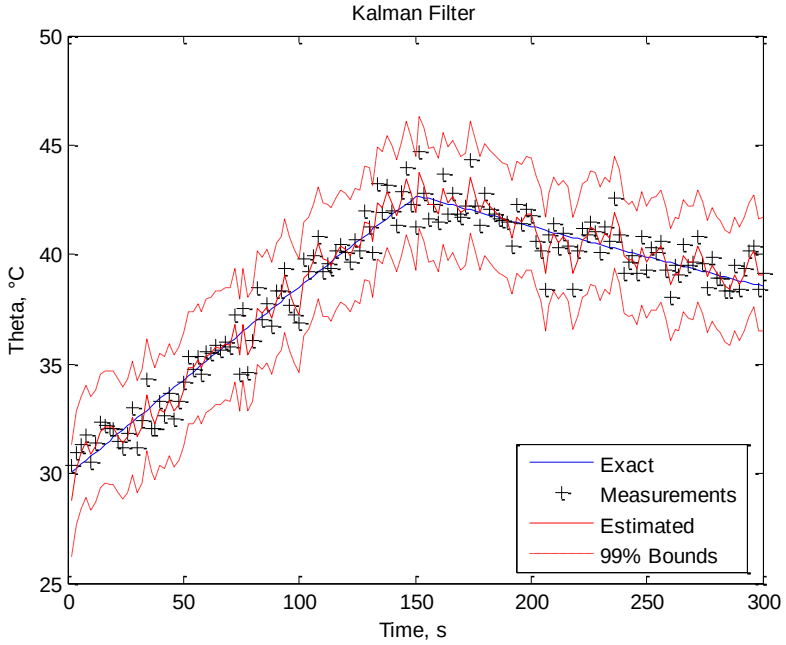


Figure A.7. Solution with the Kalman filter: θ(*t*) estimation for the step function.

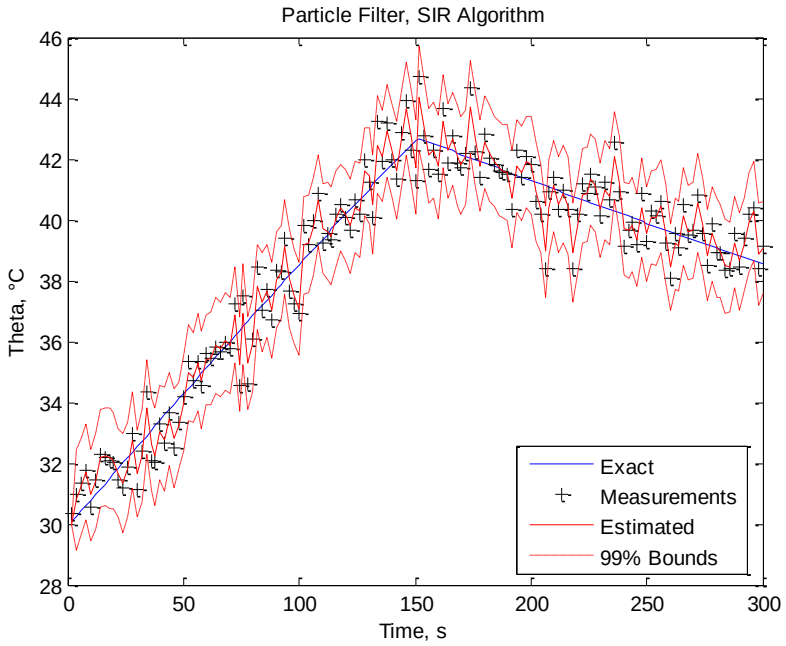


Figure A.8. Solution with the Particle filter: θ(*t*) estimation for the step function.

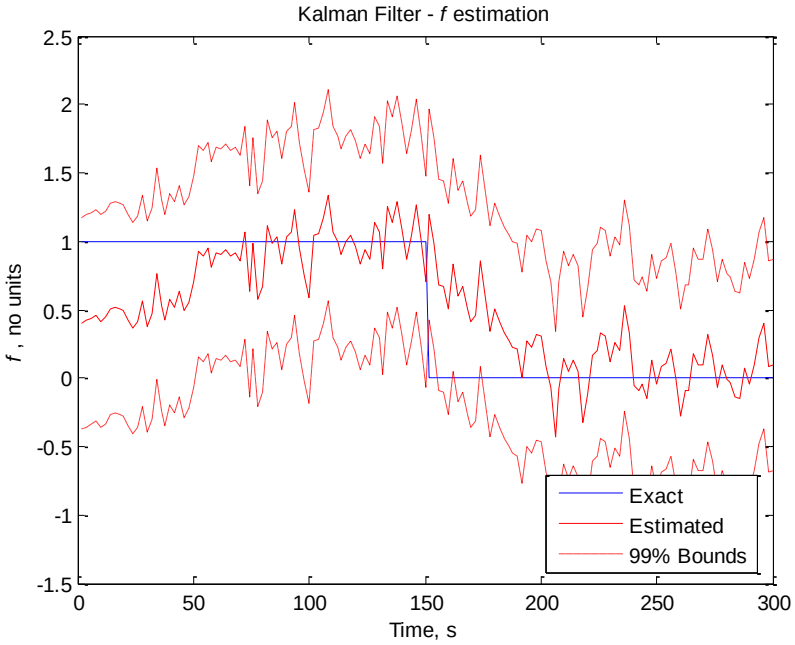


Figure A.9. Solution with the Kalman filter: $f(t)$ estimation for the step function.

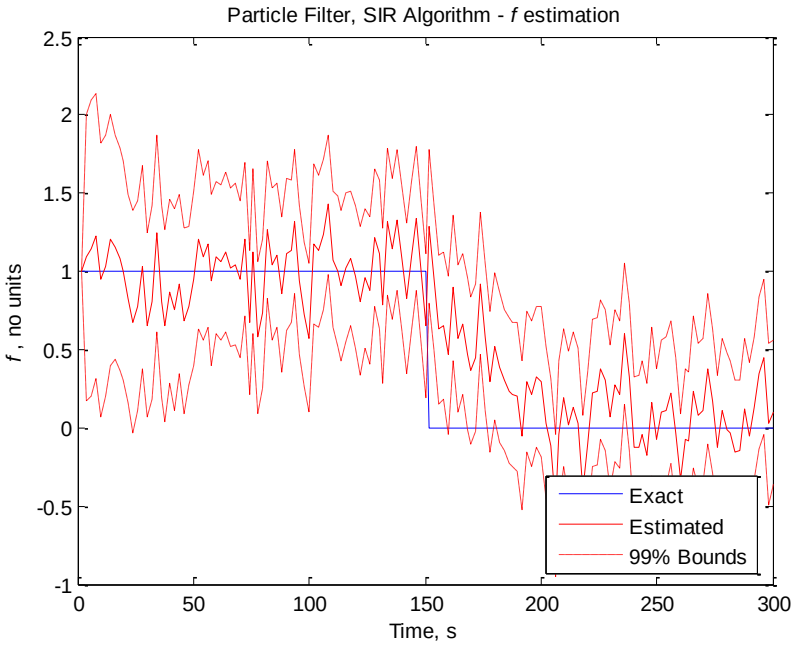


Figure A.10. Solution with the Particle filter: $f(t)$ estimation for the step function.

The results obtained for the ramp variation of the heat flux are presented in figures A.11-A.14. Figures A.11 and A.12 show the estimated temperatures obtained with the Kalman filter and the particle filter, respectively. Figures A.13 and A.14 show the estimated time variation of the heat flux obtained with the Kalman filter and the particle filter, respectively

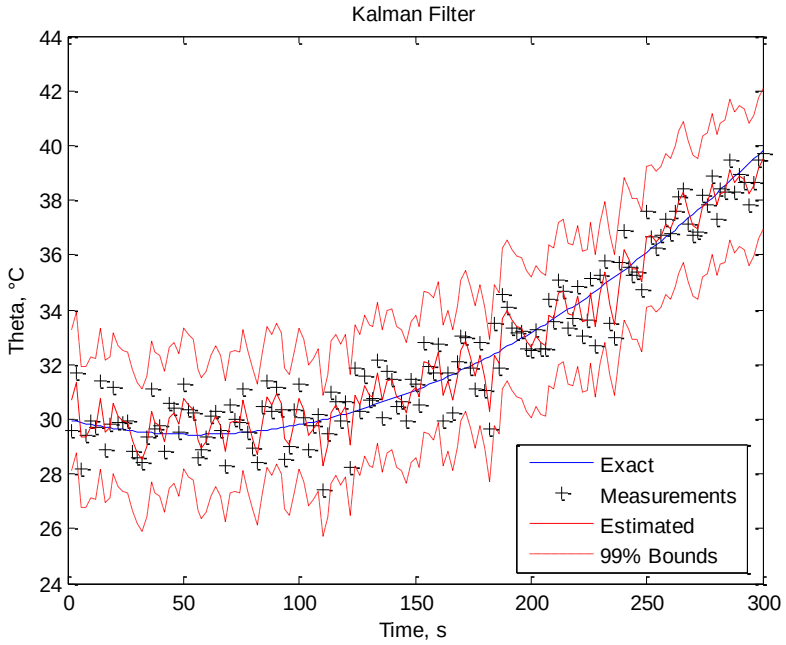


Figure A.11. Solution with the Kalman filter: θ(*t*) estimation for the ramp function.

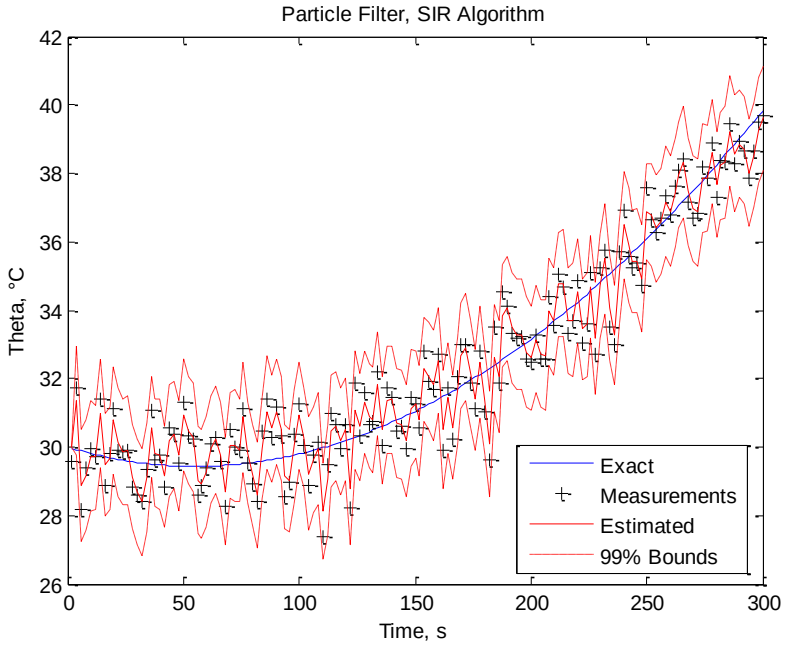


Figure A.12. Solution with the Particle filter: θ(*t*) estimation for the ramp function.

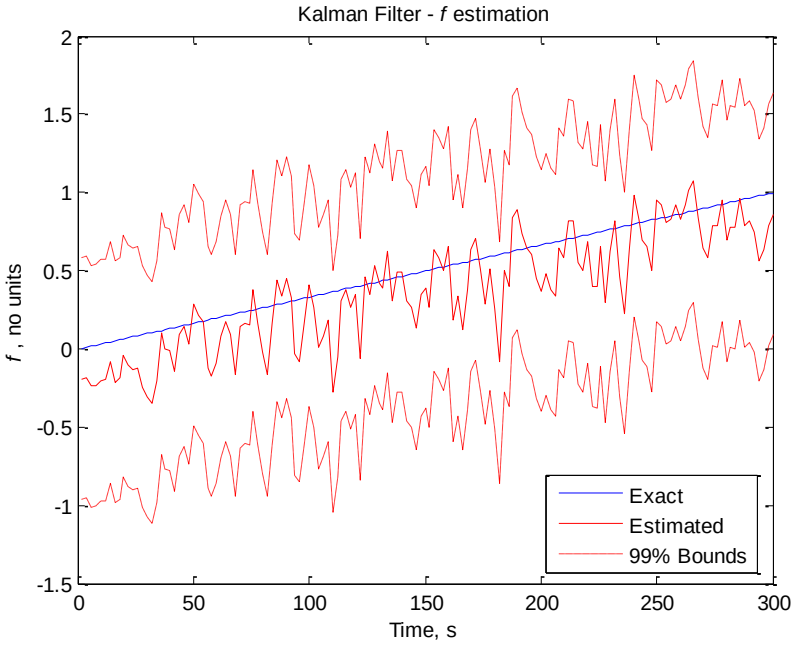


Figure A.13. Solution with the Kalman filter: $f(t)$ estimation for the ramp function.

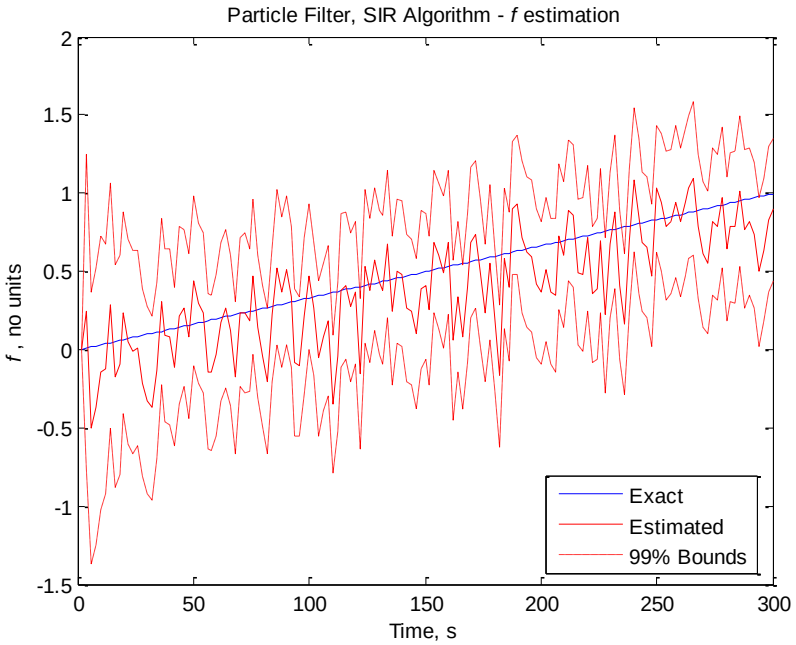


Figure A.14. Solution with the Particle filter: $f(t)$ estimation for the ramp function.

Figures A.7-A.14 illustrate the capabilities of both filters examined here, for the simultaneous estimation of the transient temperature and of the time variation of the applied heat flux. Note that measurements are available only for the temperature, and that a non-informative random walk model was used for the variation of the heat flux. For both variations of the heat flux, and with both filters, the recovered heat fluxes show oscillations due to the ill-posed character of the inverse problem (see figures A.9 and A.10 for the step function and A.13 and A.14 for the ramp function). Such oscillations can be reduced by decreasing the standard deviation used for the random walk model for $f(t)$ (see equation A.8), resulting in smoother recovered functions. On the other hand, fast variations

cannot be resolved. This fact is illustrated in figures A.15-A.18 for the step variation of the heat flux, where σ_{rw} was taken as 0.1, instead of 0.3 as above (see also figures A.7-A.10).

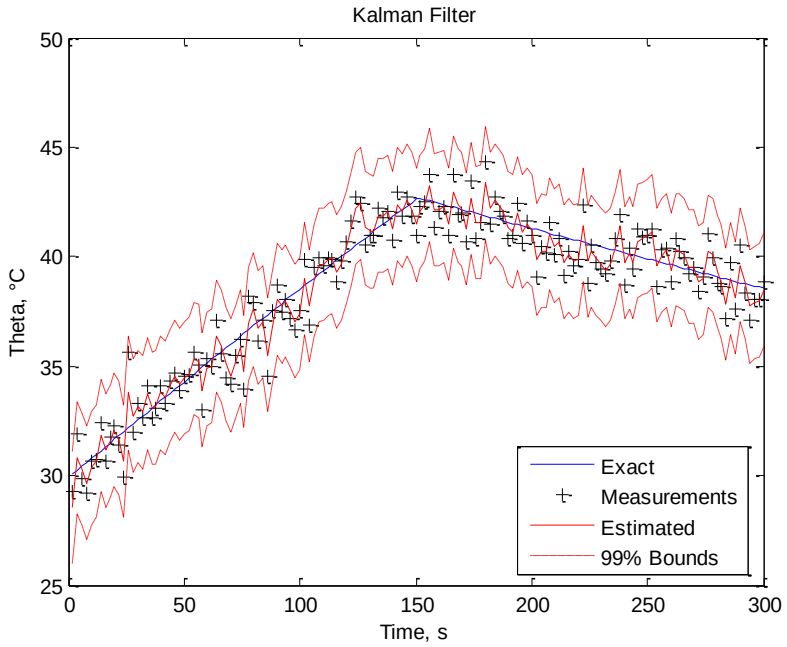


Figure A.15. Solution with the Kalman filter: θ(t) estimation for the step function.

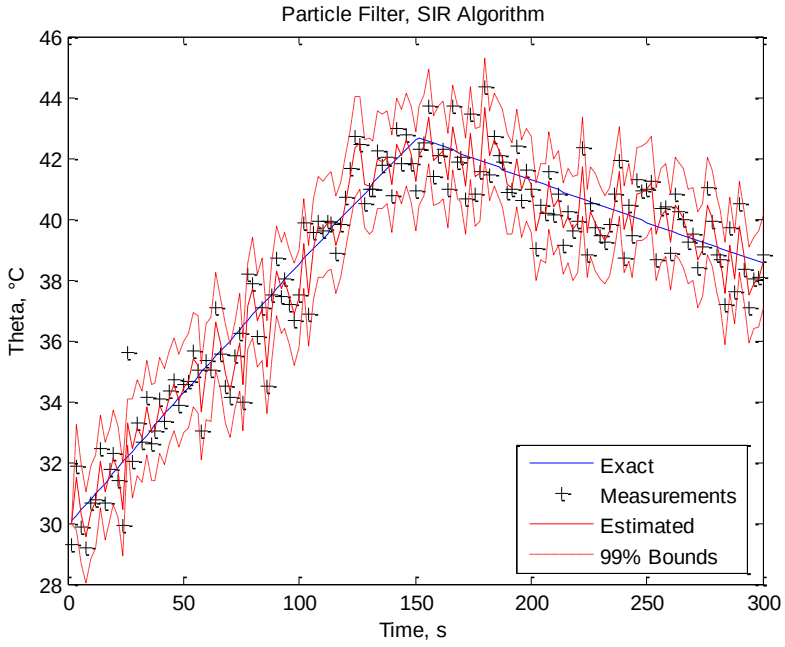


Figure A.16. Solution with the Particle filter: θ(t) estimation for the step function.

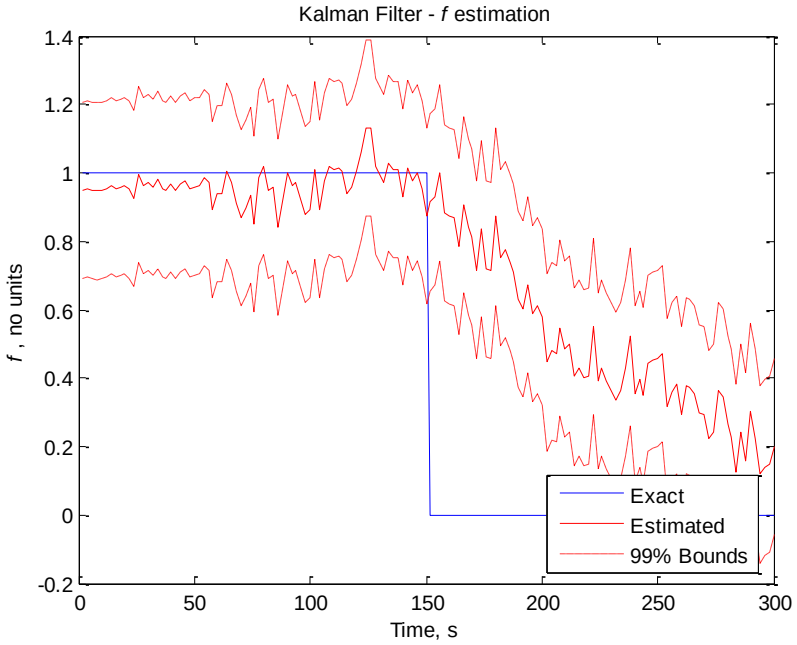


Figure A.17. Solution with the Kalman filter: $f(t)$ estimation for the step function.

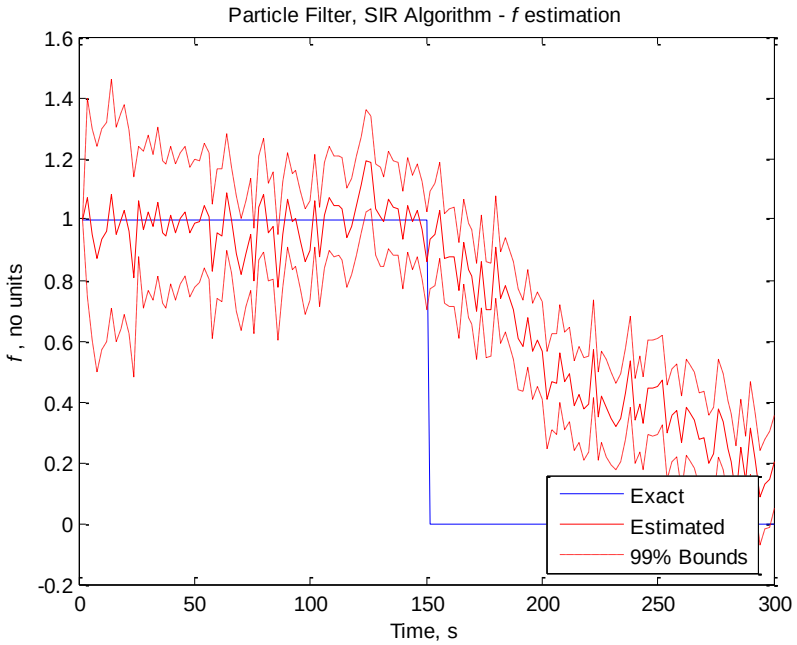


Figure A.18. Solution with the Particle filter: $f(t)$ estimation for the step function.

The MATLAB code used for the cases presented above can be found at the end of this Appendix.

References

1. Bayazitoglu, Y., Ozisik, M., 1988, *Elements of Heat Transfer*, McGraw Hill, New York.

MATLAB code for Case (i): Heat Flux $q(t) = q_0$ is constant and deterministically known, and (ii): Heat Flux $q(t) = q_0 f(t)$ is a state variable

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Example for the Kalman Filter and the Particle Filter (SIR Algorithm) %
%                                                                          %
%   Slab of thickness L with lumped formulation - Example 2-4, Bayazitoglu%
%   and Ozisik's book - page 26                                            %
%                                                                          %
%  Helcio R. B. Orlande, Henrique M. da Fonseca and Wellington B. da Silva%
%   June 01, 2011                                                          %
%                                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
clear all;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% physical parameters of the medium                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

k = 204;                    % thermal conductivity, W/mK
ro = 2707;                  % density, kg/m^3
cp = 896;                   % spcific heat, J/kgK
lthick = 0.03;              % thickness of the plate, m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% time discretization                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

timefinal=300;                    % final time, s
deltatime=1;                       % time step, s
time=deltatime:deltatime:timefinal; % vector of physical time, s

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% parameters for the equation                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

temp0 = 50;         % initial temperature, oC
q0 = 8000;          % applied heat flux, W/m^2
tempinf=20;         % external temperature, oC
htcoef=50;          % heat transfer coefficient, W/m^2K
mcoef=htcoef/(ro*cp*lthick);
q0h=q0/htcoef;
theta0=temp0-tempinf;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% number of particles for the Particle Filter                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Nparti=200;
Nparti1=1/Nparti;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  std for the measurement error, evolution model and random walk    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

stdmeas=0.3;               % standard deviation of the measurement errors
stdmodel=0.03;             % standard deviation of the evolution model
stdrw=0.03;                % random walk of the estimated parameter

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Choose of the structure of the function f                                %
%                                                                          %
% 1 = Constant function over all the experiment                            %
% 2 = Square time varying function                                         %
% 3 = Ramp function                                                        %
%                                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
 hf_type=3;                                      % choose the function type

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initializing vectors for constant heat flux                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if hf_type==1
    thetanum=zeros(1,length(time));
    xest=zeros(1,length(time));
    P=zeros(1,length(time));
    xestpf=zeros(1,length(time));
    xestsir=zeros(1,length(time));
    stdsir=zeros(1,length(time));
    xpartiold=zeros(1,length(Nparti));
    xpartinew=zeros(1,length(Nparti));
    wparti=zeros(1,length(Nparti));
    xpartires=zeros(1,length(Nparti));
    wpartires=zeros(1,length(Nparti));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initializing vectors for time varying heat flux                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if hf_type==2||hf_type==3
    thetanum=zeros(1,length(time));
    xest=zeros(2,length(time));
    P=zeros(2,2,length(time));
    xestpf=zeros(2,length(time));
    xestsir=zeros(2,length(time));
    stdsir=zeros(2,length(time));
    xpartiold=zeros(2,length(Nparti));
    xpartinew=zeros(2,length(Nparti));
    wparti=zeros(1,length(Nparti));
    xpartires=zeros(2,length(Nparti));
    wpartires=zeros(1,length(Nparti));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% form of function                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%
% constant f   %
%%%%%%%%%%%%%%%%

if hf_type==1
    f=ones(1,(1/deltatime)*timefinal) ;
end

%%%%%%%%%%%%%%%%
% step f       %
%%%%%%%%%%%%%%%%

if hf_type==2
    f=[ones(1,(1/deltatime)*timefinal/2)...
       zeros(1,(1/deltatime)*timefinal/2)];
end

%%%%%%%%%%%%%%%%
% ramp f       %
%%%%%%%%%%%%%%%%

if hf_type==3
    f=(1:deltatime:timefinal)/timefinal ;       % ramp
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Exact Solution                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%
% constant f    %
%%%%%%%%%%%%%%%%

if hf_type==1
    thetaexa = theta0*exp(-mcoef.*time)+q0h.*(1-exp(-mcoef.*time));
end

%%%%%%%%%%%%%%%%
% step f       %
%%%%%%%%%%%%%%%%

if hf_type==2
    Int=[(exp(mcoef.*(1:deltatime:(timefinal/2)))-1)...
         (exp(mcoef.*timefinal/2)-1)*ones(1,(1/deltatime)*timefinal/2)] ;

    thetaexa = exp(-mcoef.*time).*(theta0+q0h.*Int);
end

%%%%%%%%%%%%%%%%
% ramp f       %
%%%%%%%%%%%%%%%%

if hf_type==3
    thetaexa = exp(-mcoef.*time).*(theta0+q0h.*...
               ((1+exp(mcoef.*time).*(mcoef.*time-1))/(timefinal*mcoef)));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Numerical solution  (finite differece)                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

bigq=mcoef*q0h*deltatime;
thetanum(1)=theta0;
for k = 1:(length(time)-1)
    thetanum(k+1)=(1-mcoef*deltatime)*thetanum(k)+bigq*f(k);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Simulated measurements                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

unce=randn(1,length(time));         %random vector. mean=1 and std=0
thetameas=thetaexa+(stdmeas.*unce); %Tmeas=Texa+eps

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Kalman Filter                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% constant f               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if hf_type==1
    F=(1-mcoef*deltatime);
    S=bigq;
    H=1;
    Q=stdmodel^2;
    R=stdmeas^2;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Initial State and its Covariance Matrix                                    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    xminus(1)=theta0+randn*stdmeas;
    xest(1)=xminus(1);
    P(1)=R;

    for k = 1:(length(time)-1)
        %%%%%%%%%%%%%%%%%%%%%%%%
        %   Prediction Kalman %
        %%%%%%%%%%%%%%%%%%%%%%%%
```

```
        xminus=F*xest(k)+S;
        Pminus=F*P(k)*F'+Q;

        %%%%%%%%%%%%%%%%%%%%%%%
        %   Update Kalman     %
        %%%%%%%%%%%%%%%%%%%%%%%

        Gainm=Pminus*H'*inv(H*Pminus*H'+R);  %#ok<*MINV>
        xest(k+1)=xminus+Gainm*(thetameas(k+1)-H*xminus);
        P(k+1)=(eye(1)-Gainm*H)*Pminus;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Confidence interval        %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%

    lbd=xest-2.576.*sqrt(P);
    ubd=xest+2.576.*sqrt(P);
end % of the Kalman Filter estimation for a constant heat flux

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% time varying f            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if hf_type==2 || hf_type==3
    F=[(1-mcoef*deltatime) (mcoef*deltatime*q0h);0 1];
    H=[1 0];
    Q=[stdmodel^2    0    ;...
          0       stdrw^2];
    R=stdmeas^2;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Initial State and its Covariance Matrix                                    %
    % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    xminus=[theta0+randn*stdmeas;
              f(1)+randn*stdrw];
    Pminus=Q;
    xest(:,1)=xminus;
    P(:,:,1)=Pminus;


    for k = 1:(length(time)-1)
        %%%%%%%%%%%%%%%%%%%%%%%%
        %   Prediction Kalman %
        %%%%%%%%%%%%%%%%%%%%%%%%

        xminus=F*xminus;
        Pminus=F*Pminus*F'+Q;

        %%%%%%%%%%%%%%%%%%%%%%%
        %   Update Kalman     %
        %%%%%%%%%%%%%%%%%%%%%%%

        Gainm=Pminus*H'*inv(H*Pminus*H'+R);  %#ok<*MINV>
        xnew=xminus+Gainm*(thetameas(k+1)-H*xminus);
        Pnew=(eye(2)-Gainm*H)*Pminus;

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Storing the predicted variables %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        xest(:,k+1)=xnew;
        P(:,:,k+1)=Pnew;

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Updating auxiliary variables   %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        xminus=xnew;
        Pminus=Pnew;
```

```
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Confidence interval       %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    lbdt=xest(1,:)-2.576.*sqrt(P(1,1));
    ubdt=xest(1,:)+2.576.*sqrt(P(1,1));

    lbdq=xest(2,:)-2.576.*sqrt(P(2,2));
    ubdq=xest(2,:)+2.576.*sqrt(P(2,2));
end %  of the Kalman Filter estimation for a time varying heat flux

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Particle Filter (SIR Algorithm)                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% constant f                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if hf_type==1

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Distribution Initial Time %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    xestsir(1)=theta0;

    for i=1:Nparti
        xpartiold(i)=theta0+randn*stdmodel;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%
    % Advancing Particles  %
    %%%%%%%%%%%%%%%%%%%%%%%%%%

    for k = 1:(length(time)-1)
          for i=1:Nparti
            %%%%%%%%%%%%%%%%%%%%%%
            %   1 - PREDICTION %
            %%%%%%%%%%%%%%%%%%%%%%
            xpartinew(i)=F*xpartiold(i)+S+randn*stdmodel;
            %%%%%%%%%%%%%%%%%%%%%%
            %   Weights         %
            %%%%%%%%%%%%%%%%%%%%%%
            wparti(i)=exp(-((xpartinew(i)-thetameas(k+1))/stdmeas)^2);
          end
        wtotal=sum(wparti);
        wpartin=wparti./wtotal;
        %%%%%%%%%%%%%%%%%
        % 2 - UPDATE   %
        %%%%%%%%%%%%%%%%%
        %xpartiw=xpartinew.*wparti;
        %%%%%%%%%%%%%%%%%%%%%%
        %   Mean at time k+1 %
        %%%%%%%%%%%%%%%%%%%%%%
        xestpf(k+1)=xpartinew*wpartin';
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %   Standard Deviation at time k+1 %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % 3 - RESAMPLING     %
        %%%%%%%%%%%%%%%%%%%%%%
        cresa=zeros(Nparti,1);
        uresa=zeros(Nparti,1);
        cresa(1)=wpartin(1);
        for i=2:Nparti
            cresa(i)=cresa(i-1)+wpartin(i);
        end
        iresa=1;
        uresa(1)=rand*Nparti1;
```

```
        for j=1:Nparti
            uresa(j)=uresa(1)+Nparti1*(j-1);
            while uresa(j) > cresa(iresa)
                iresa=iresa+1;
            end
            xpartires(j)=xpartinew(iresa);
            wpartires(j)=Nparti1;
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Means and standard deviation   %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        xestsir(k+1)=mean(xpartires);
        stdsir(k+1)=std(xpartires);
        xpartiold=xpartires;
        %%%%%%%%%%%%%%%%%%%%%%
        % End of resampling %
        %%%%%%%%%%%%%%%%%%%%%%
        %xpartiold=xpartinew;

    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Confidence interval       %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    lbdsir=xestsir-2.576.*stdsir;
    ubdsir=xestsir+2.576.*stdsir;
end %  of the Particle Filter estimation for a constant heat flux


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% time varying f               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if hf_type==2 || hf_type==3

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Distribution Initial Time %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    xestsir(:,1)=[theta0;f(1)];

    for i=1:Nparti
        xpartiold(1,i)=theta0+randn*stdmodel;
        xpartiold(2,i)=f(1)+randn*stdrw;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%
    % Advancing Particles  %
    %%%%%%%%%%%%%%%%%%%%%%%%

    for k = 1:(length(time)-1)
          for i=1:Nparti
            %%%%%%%%%%%%%%%%%%%%%
            %   1 - PREDICTION %
            %%%%%%%%%%%%%%%%%%%%%
            xpartinew(:,i)=F*xpartiold(:,i)+randn*[stdmodel;stdrw];
            %%%%%%%%%%%%%%%%%%%%%
            %   Weights        %
            %%%%%%%%%%%%%%%%%%%%%
            wparti(i)=exp(-((xpartinew(1,i)-thetameas(k+1))/stdmeas)^2);
          end
        wtotal=sum(wparti);
        wpartin(1,:)=wparti./wtotal;
        %%%%%%%%%%%%%%%%%
        % 2 - UPDATE   %
        %%%%%%%%%%%%%%%%%
        %xpartiw=xpartinew.*wparti;
        %%%%%%%%%%%%%%%%%%%%%%
        %   Mean at time k+1 %
        %%%%%%%%%%%%%%%%%%%%%%
        xestpf(:,k+1)=xpartinew*wpartin';
```

```
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %   Standard Deviation at time k+1 %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % 3 - RESAMPLING     %
        %%%%%%%%%%%%%%%%%%%%%%
        cresa=zeros(Nparti,1);
        uresa=zeros(Nparti,1);
        cresa(1)=wpartin(1);
        for i=2:Nparti
            cresa(i)=cresa(i-1)+wpartin(i);
        end
        iresa=1;
        uresa(1)=rand*Nparti1;
        for j=1:Nparti
            uresa(j)=uresa(1)+Nparti1*(j-1);
            while uresa(j) > cresa(iresa)
                iresa=iresa+1;
            end
            xpartires(:,j)=xpartinew(:,iresa);
            wpartires(j)=Nparti1;
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Means and standard deviation   %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        xestsir(1,k+1)=mean(xpartires(1,:));     % Tempearature
        xestsir(2,k+1)=mean(xpartires(2,:));     % f
        stdsir(1,k+1)=std(xpartires(1,:));       % Temperature
        stdsir(2,k+1)=std(xpartires(2,:));       % f
        xpartiold=xpartires;
        %%%%%%%%%%%%%%%%%%%%%
        % End of resampling %
        %%%%%%%%%%%%%%%%%%%%%
        %xpartiold=xpartinew;

    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Confidence interval       %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    lbdsirt=xestsir(1,:)-2.576.*stdsir(1,:);
    ubdsirt=xestsir(1,:)+2.576.*stdsir(1,:);

    lbdsirf=xestsir(2,:)-2.576.*stdsir(2,:);
    ubdsirf=xestsir(2,:)+2.576.*stdsir(2,:);
end %  of the Particle Filter estimation for a time varying heat flux



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot of the results - constant f                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if hf_type==1
    figure (1)
    plot(time,thetaexa,'-b',time,thetanum,'--r');
    xlabel('Time, s');
    ylabel('Theta, °C');
    legend('Exact','Numerical','Location','SouthEast');

    figure (2)
    plot(time,thetaexa,'-b',time,thetanum,'-r',time,thetameas,'+k');
    xlabel('Time, s');
    ylabel('Theta, °C');
    legend('Exact','Numerical','Measurements','Location','SouthEast');

    figure (3)
    plot(time,thetaexa,'-b',time,thetameas,'+k',time,xest,'-r',time,lbd,'--r',time,ubd,'--
r');
    xlabel('Time, s');
    ylabel('Theta, °C');
    title('Kalman Filter');
```

```
    legend('Exact','Measurements','Estimated','99% Bounds','Location','SouthEast');

    figure (4)
    plot(time,thetaexa,'-b',time,thetameas,'+k',time,xestsir,'-r',time,lbdsir,'--
r',time,ubdsir,'--r');
    xlabel('Time, s');
    ylabel('Theta, °C');
    title('Particle Filter, SIR Algorithm');
    legend('Exact','Measurements','Estimated','99% Bounds','Location','SouthEast');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot of the results - time varying f                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if hf_type==2||hf_type==3
    figure (1)
    plot(time,thetaexa,'-b',time,thetanum,'--r');
    xlabel('Time, s');
    ylabel('Theta, °C');
    legend('Exact','Numerical','Location','SouthEast');

    figure (2)
    plot(time,thetaexa,'-b',time,thetanum,'-r',time,thetameas,'+k');
    xlabel('Time, s');
    ylabel('Theta, °C');
    legend('Exact','Numerical','Measurements','Location','SouthEast');

    figure (3)
    plot(time,thetaexa,'-b',time,thetameas,'+k',time,xest(1,:),'-r',...
         time,lbdt,'--r',time,ubdt,'--r');
    xlabel('Time, s');
    ylabel('Theta, °C');
    title('Kalman Filter');
    legend('Exact','Measurements','Estimated','99% Bounds',...
           'Location','SouthEast');

    figure (4)
    plot(time,thetaexa,'-b',time,thetameas,'+k',time,xestsir(1,:),'-r',...
         time,lbdsirt,'--r',time,ubdsirt,'--r');
    xlabel('Time, s');
    ylabel('Theta, °C');
    title('Particle Filter, SIR Algorithm');
    legend('Exact','Measurements','Estimated','99% Bounds',...
           'Location','SouthEast');

    figure (5)
    plot(time,f,'-b',time,xest(2,:),'-r',time,lbdq,'--r',time,ubdq,'--r');
    xlabel('Time, s');
    ylabel('\itf \rm, no units');
    title('Kalman Filter - \itf \rmestimation');
    legend('Exact','Estimated','99% Bounds','Location','SouthEast');

    figure (6)
    plot(time,f,'-b',time,xestsir(2,:),'-r',time,lbdsirf,'--r',...
         time,ubdsirf,'--r');
    xlabel('Time, s');
    ylabel('\itf \rm, no units');
    title('Particle Filter, SIR Algorithm - \itf \rmestimation');
    legend('Exact','Estimated','99% Bounds','Location','SouthEast');
end
```