Kalman Filter for the Resolution of Parameterized Partial Differential Equations

Abdoul Aziz Sarr

August 2025

Contents

1 Introduction

Parameterized partial differential equations (PDEs) are pervasive in modeling physical phenomena, particularly in mechanics, thermal sciences, and engineering. These problems depend not only on spatial and temporal variables but also on parameters (physical constants, domain shapes, boundary conditions) that influence their behavior.

The **classical Kalman filter**, introduced in the 1960s, is a recursive least-squares optimal state estimation algorithm based on linear noisy models. It is especially used to fuse information from a numerical model (prediction) and often noisy measurements (observation). Applied to PDE resolution, the Kalman filter enables real-time assimilation of experimental or synthetic data within a numerical simulation. This improves the quality and robustness of the obtained solutions, particularly in the presence of uncertainties or incomplete measurements.

In this study, we focus on the heat equation:

$$\frac{\partial u}{\partial t} - \nabla \cdot (k \nabla u) = f \quad \text{in} \quad \Omega, \tag{1}$$

where u is the temperature, k the thermal diffusivity, f a source term, and Ω the spatial domain. The numerical resolution uses the `Feel++` library, which provides a powerful framework for finite element discretization, linear system assembly, post-processing, and extension to data assimilation methods.

1.1 Numerical Tools: the Feel++ Library

Feel++ (Finite Element Embedded Library in C++) is an open-source platform dedicated to solving PDE problems using the finite element method. It offers:

- A Python interface for rapid modeling (meshes, boundary conditions, solvers, post-processing),
- Multi-dimensional (2D/3D), multi-physics support, and easy parameter integration,
- Advanced features for data assimilation, optimization, and uncertainty quantification.

In this report, Feel++ is used for geometry generation, discretization of the heat equation, spatio-temporal integration, and export of results necessary for Kalman filter analysis.

2 Kalman Filter for PDE Resolution

The Kalman filter aims to improve the estimation of the state of a dynamic system by correcting the model prediction via smart integration of noisy measurements.

2.1 Mathematical Formulation of the Kalman Filter

Applied scheme (time index k):

Prediction step (model):

$$X_{k|k-1} = A_k X_{k-1|k-1} + B_k u_k, \tag{2}$$

$$P_{k|k-1} = A_k P_{k-1|k-1} A_k^\top + Q_k, \tag{3}$$

where

- $X_{k|k-1}$: predicted state (here, discretized temperature),
- A_k: model matrix (from finite element discretization + time scheme),
- P: prediction error covariance matrix,
- Q_k: process noise covariance matrix,
- $B_k u_k$: known control or source term (often zero here).

Observation step (assimilation):

$$Z_k = H_k X_{k|k-1} + v_k, \tag{4}$$

where

- Z_k: vector of recorded measurements,
- H_k: observation matrix (selects nodes with available measurements),
- v_k: measurement noise with covariance R_k.

Update step (correction):

$$K_k = P_{k|k-1} H_k^\top \left(H_k P_{k|k-1} H_k^\top + R_k \right)^{-1}, \tag{5}$$

$$X_{k|k} = X_{k|k-1} + K_k (Z_k - H_k X_{k|k-1}), \tag{6}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}. \tag{7}$$

Summary of used matrices:

- X_k: state vector (values of u at nodes),
- P_k: error covariance matrix of the state,
- Q_k: process noise covariance,
- H_k: observation matrix (nodes where measurement is done),
- R_k: measurement noise covariance,
- K_k: Kalman gain matrix at step k.

2.2 Kalman Filter Algorithm

The general algorithm used in this work is:

1. Initialization: initial state X_0, initial covariance P_0.
2. For each time step k:
   (a) Model prediction: calculate $X_{k|k-1}$ and $P_{k|k-1}$ by numerical resolution (Feel++).
   (b) Observation: acquisition (or simulation) of noisy measurements Z_k at certain nodes.
   (c) Calculation of Kalman gain K_k and correction of prediction by Z_k.
   (d) Update of error covariance: compute $P_{k|k}$.

This algorithm fits seamlessly within temporal loops of PDE resolution at each time step.

2.3 Robustness Analysis and Importance of the Filter

The Kalman filter provides adaptive correction of numerical predictions by integrating uncertainty related to model errors and measurement noise. Essential advantages in the context of parameterized PDE resolution include:

- Kalman correction allows consistent solutions even if the numerical model alone has approximations or measurements are noisy or partial.
- Data assimilation reduces overall error and improves numerical stability, making the method robust to parameter uncertainties, initial conditions, or mesh variations.
- Comparing resolution with and without Kalman filter shows that incorporating observations leads to solutions less sensitive to noise and closer to observed reality or exact solutions.

3 Tools for Solution Resolution and Visualization

This section details setting up the computational environment and resolution strategy for parameterized heat equations. We present the conjoint use of Feel++ for discretization and numerical resolution, and the integration of the Kalman filter for data assimilation and refining approximate solutions.

3.1 PDE Resolution

```
import sys
import feelpp.core as fppc
import feelpp.toolboxes.core as tb
import pandas as pd
import numpy as np
import pyvista as pv
from feelpp.HeatEquation import HeatEquationKalmanSolver,
   runHeatEquationPk, runConvergenceAnalysis, plot_convergence,
   custom_cmap

sys.argv = ["feelpp_app"]
e = fppc.Environment(
    sys.argv,
    opts=tb.toolboxes_options("coefficient-form-pdes", "cfpdes"),
    config=fppc.localRepository('feelpp_cfpde')
)

# Transient heat equation problem with Kalman filter
u_exact = 'sin(pi*x)*sin(pi*y) + 0.1*t + 0.5'
grad_u_exact = '{pi*cos(pi*x)*sin(pi*y), pi*sin(pi*x)*cos(pi*y)}'
rhs = '0.1 + 2*pi*pi*sin(pi*x)*sin(pi*y)'
k_value = '1'  # Thermal diffusivity k = 1
u0_initial_condition = 'sin(pi*x)*sin(pi*y) + 0.5'
g_dirichlet_boundary = '0.1*t + 0.5'
gN_neumann_boundary = '0'
```

```
24
25 # Time simulation parameters
26 dt = 0.01
27 T_end = 0.5
28 theta_scheme = 0.5
29 measurement_frequency = 5
30 noise_std_dev_measurement = 0.05
31 Q_process_noise_magnitude = 1e-4
32
33 HESolver = HeatEquationKalmanSolver(dim=2, order=1)
34 kalman_results_per_h = []
35
36 for h_val in mesh_sizes:
37     print(f"\n--- Run for h = {h_val} ---")
38     current_HESolver = HeatEquationKalmanSolver(dim=2, order=1)
39     current_HESolver.h = h_val
40     current_HESolver.u_exact = u_exact
41     current_HESolver.grad_u_exact = grad_u_exact
42     current_HESolver.rhs = rhs
43     current_HESolver.k_param = k_value
44     current_HESolver.g = g_dirichlet_boundary
45     current_HESolver.gN = gN_neumann_boundary
46     current_HESolver.u0 = u0_initial_condition
47     current_HESolver.T_end = T_end
48     current_HESolver(
49         h=h_val, rhs=rhs, k_param=k_value,
            g=g_dirichlet_boundary,
50         gN=gN_neumann_boundary, u0=u0_initial_condition, dt=dt,
            T_end=T_end,
51         theta_scheme=theta_scheme,
            measurement_frequency=measurement_frequency,
52         noise_std_dev_measurement=noise_std_dev_measurement,
53         Q_process_noise_magnitude=Q_process_noise_magnitude,
54         plot=1, u_exact=u_exact, grad_u_exact=grad_u_exact
55     )
```

Listing 1: Environment setup and parameter initialization for solving the parameterized heat equation with Feel++ and Kalman filter

4 Visualization of PDE Results with the Kalman Filter

This section presents the visualization results of the solution to the partial differential equation with the application of the Kalman filter for different mesh sizes h. Each figure compares the Kalman filter estimate, the Feel++ model prediction, and the exact solution at time $t = 0.50$.

4.1 Results for mesh size $h = 0.1$

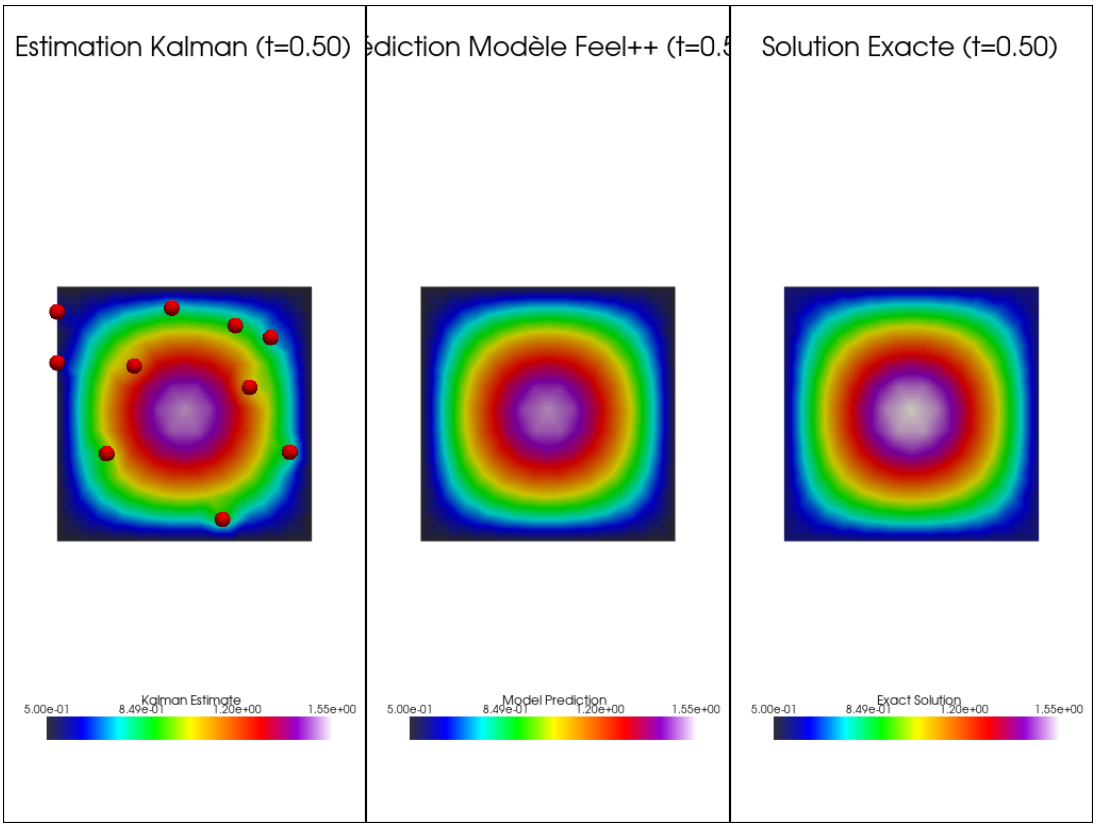

Figure 1: Results visualization for $h = 0.1$ at $t = 0.50$: Kalman estimate, Feel++ model prediction and exact solution.

For the relatively coarse mesh size $h = 0.1$, the Kalman estimate presents a value distribution close to the exact solution. The red points on the Kalman estimate map indicate measurement locations. Despite the coarser mesh, the filter manages to correct the model prediction (which is already visually very close to the exact solution) by integrating these observations. Visual comparison shows that the Feel++ model solution is of good quality, and the Kalman filter refines this solution especially around measurement points, thus reducing potential discrepancies due to model or mesh uncertainties.

4.2 Results for mesh size $h = 0.05$

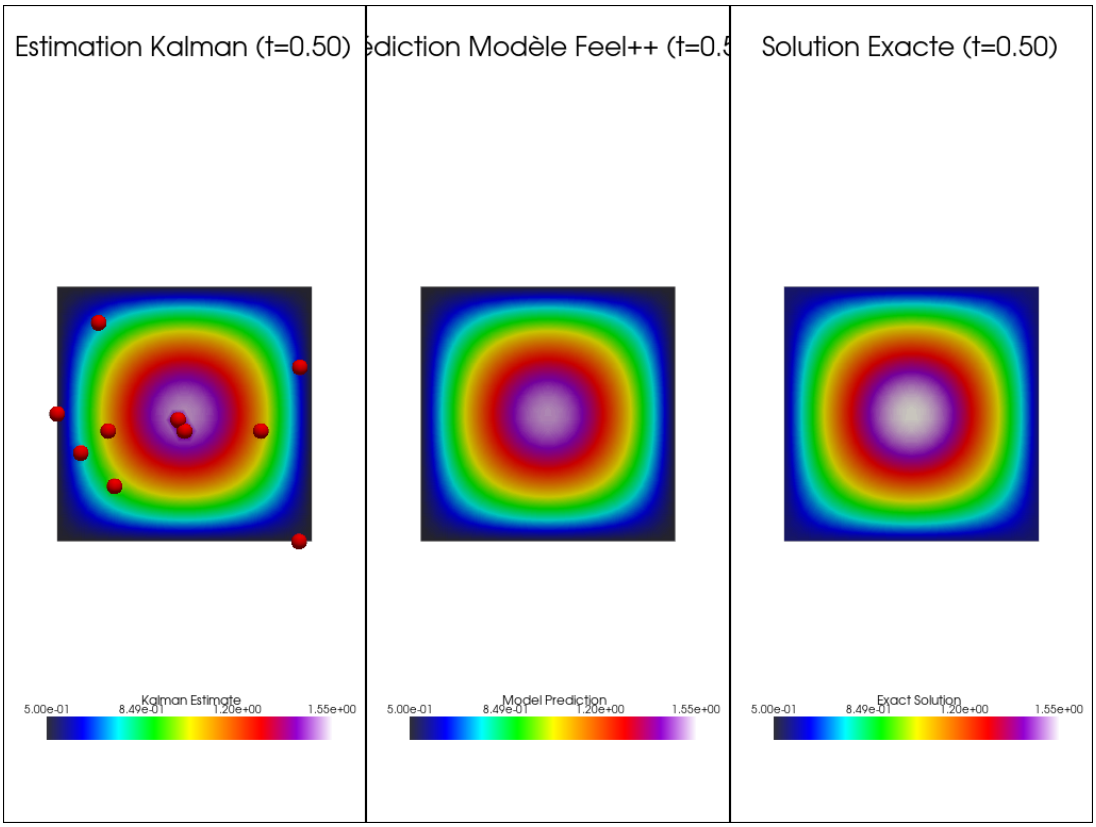

Figure 2: Results visualization for $h = 0.05$ at $t = 0.50$: Kalman estimate, Feel++ model prediction and exact solution.

With a finer mesh size $h = 0.05$, the Feel++ model prediction becomes even more accurate and visually almost identical to the exact solution. The Kalman estimate continues to follow this trend, and differences with the exact solution are minimal. The measurement points (in red) remain on the Kalman estimate map but their influence on correction is less visually apparent because the model solution is already very good. This shows that the more intrinsically accurate the model (due to finer mesh), the more the Kalman filter focuses on managing residual noise and uncertainties, allowing a quick convergence to the exact solution.

4.3 Results for mesh size $h = 0.025$

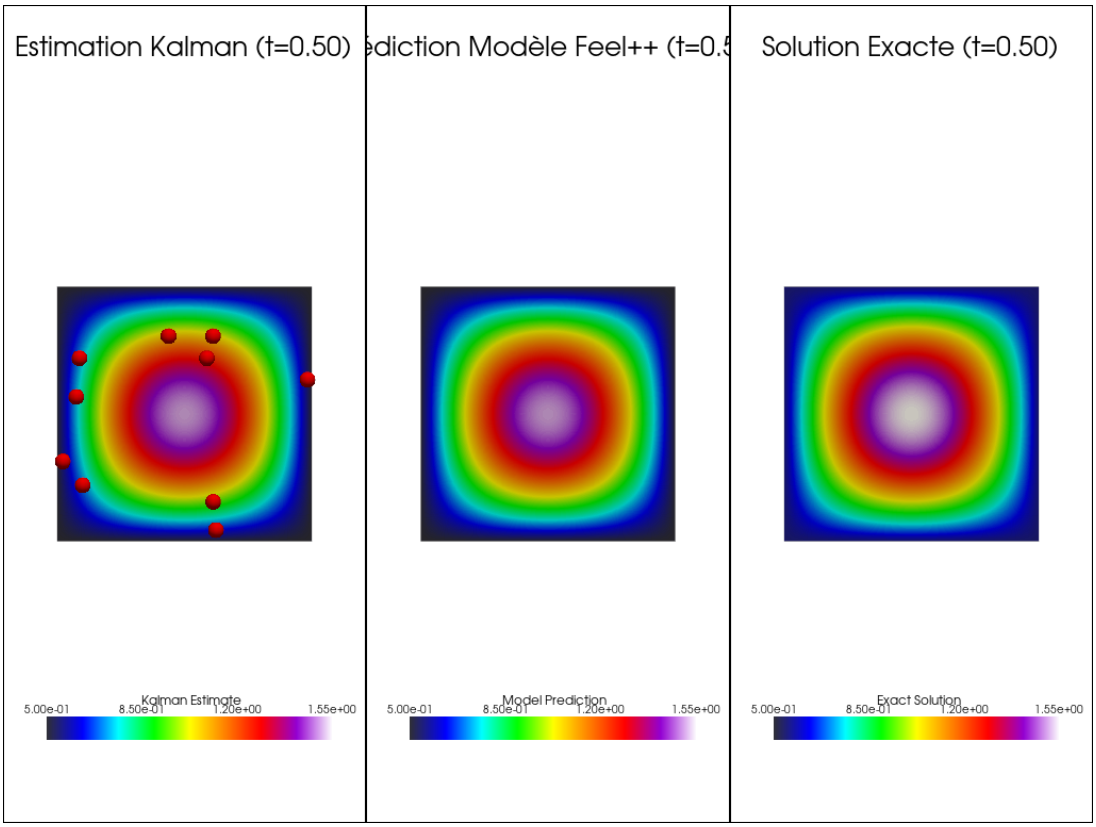

Figure 3: Results visualization for $h = 0.025$ at $t = 0.50$: Kalman estimate, Feel++ model prediction and exact solution.

For the finest mesh size $h = 0.025$, the three representations – Kalman estimate, Feel++ model prediction, and exact solution – are almost indistinguishable to the naked eye. This indicates excellent convergence of the finite element model toward the exact solution as the mesh refines. In this scenario, the Kalman filter's role is to guarantee robustness against measurement and process noise, even if the discretization error is already very low. The corrections brought by the filter are subtle but essential to maintain the optimality of the estimate in the presence of uncertainties, ensuring a smooth and reliable solution.

4.4 General Commentary on Smoothing and Robustness of the Kalman Filter

All these visualizations highlight the Kalman filter's capability to produce smoothed and robust estimates for PDE resolution, even in presence of potentially noisy measurement data (indicated by red points).

Smoothing: The Kalman estimate provides a color map that is very coherent and smooth, showing no artifacts or irregularities that raw measurements could introduce if naively integrated. The Kalman filter, by merging the model prediction with observations, manages to "smooth" the data, mitigating measurement noise impact while correcting model deviations. This results in soft color transitions and well-defined contours, similar to the exact solution.

Robustness: The filter's robustness is demonstrated by its ability to maintain a high-quality estimate even on coarser meshes ($h = 0.1$). As the mesh refines, the model prediction improves, but the Kalman filter retains a crucial role by assimilating residual uncertainties and noise. It ensures the obtained solution remains reliable and close to physical reality, even if the model has approximations or if measurements are incomplete or corrupted. The filter acts as a continuous regularization and correction mechanism, essential for realistic engineering and scientific simulations.

5 Conclusion

In summary, the Kalman filter proves efficient by providing an optimal solution that leverages both numerical model precision and observation reliability. Whether the mesh is coarse or very fine, it guarantees rapid convergence, noise attenuation, and robustness to uncertainty, making it relevant for a wide range of applied contexts.