Parallelised State Estimation

Application to Methyl Methacrylate Polymerization Process

Aryan Agal
Energy Science and Engineering
IIT Bombay
Mumbai, Maharashtra
aryanagal98@gmail.com

Chitrangna Bhatt
Energy Science and Engineering
IIT Bombay
Mumbai, Maharashtra
chitrangnabhatt@iitb.ac.in

Siddhesh Pawar
Energy Science and Engineering
IIT Bombay
Mumbai, Maharashtra
17d170011@iitb.ac.in

***Abstract*—Accurately knowing the state of a system is an important problem and is essential for reliable practical implementation of many control strategies. One technique to estimate the state of a system is Kalman Filtering, which finds applications in chemical process control, vehicle navigation, signal processing among many other fields. With the advent of on-board GPUs, it is reasonable to use these to speed up state estimation. This paper implements a Proof-of-Concept of this in PyTorch, a popular deep learning framework, to estimate states of a Methyl Methacrylate Polymerization Process.**

***Index Terms*—Kalman Filter, State Estimation, Parallelization**

I. Introduction

Polymers from a very significant segment of the chemical processing industry. The polymers are manufactured in diverse types of reactors by means of different kinetic mechanisms. The quality, processability, and utility of these products greatly depend on the molecular weight distribution (MWD) of the polymer, which is specified at the synthesis stage [1]. Thus, monitoring and controlling MWD in polymerization reactors is of considerable importance. The MWD cannot be measured directly during the process.

To address this measurement problem, several research directions have been pursued. These include the following: (a) development of faster and more reliable sensors (b) study of the quantitative relations between easily available measurements and product quality indices (for example, between viscosity and average molecular weights), and (c) reliable estimation of the unmeasurable state variables from readily available measurements by using state observers/estimators [2] [3]. This paper looks at reliable state estimation of unmeasurable states from readily available measurements.

An important polymer production method is the so-called free-radical polymerization mechanism which is carried out in continuous stirred tank polymerization reactors. These reactors exhibit a diversity of nonlinear and dynamic behavior. State estimators are deterministic/stochastic dynamic systems that are used to reconstruct the inaccessible but important process state variables, from easily measured variables [2]. This paper uses a Kalman filter(KF) to estimate the state variables from the temperature measurements available. The basic idea of KF is to perform linearization at each step to approximate the nonlinear system as a time varying system, affine in variables, to be estimated and then apply linear filter theory to it.

Prof. S. Gopalakrishnan

II. System Modelling

The methyl methacrylate polymerization process (MMA process) considered is operated under the following assumptions -
1. Perfectly mixed reactor contents
2. Uniform cooling fluid temperature
3. Constant density of cooling fluid
4. Constant volume
5. No gel effect
6. Constant heat capacity

The dynamics of the polymerization process can be defined with equations (1)-(6)

$$\frac{dC_m}{dt} = \frac{F\left(C_{\min} - C_m\right)}{V} - \left(k_p + k_{fm}\right) C_m P_0 \tag{1}$$

$$\frac{dC_I}{dt} = \frac{\left(F_I C_{Iin} - F C_I\right)}{V} - k_I C_I \tag{2}$$

$$\frac{dD_0}{dt} = \left(0.5 k_{tc} + k_{td}\right) P_0^2 + k_{fm} C_m P_0 - \frac{F D_0}{V} \tag{3}$$

$$\frac{dD_1}{dt} = M_m \left(k_p + k_{fm}\right) C_m P_0 - \frac{F D_1}{V} \tag{4}$$

$$\frac{dT}{dt} = \frac{F\left(T_{in} - T\right)}{V} + \frac{(-\Delta H) k_p C_m P_0}{\rho C_p} - \frac{UA\left(T - T_j\right)}{\rho C_p V} \tag{5}$$

$$\frac{dT_j}{dt} = \frac{F_{cw}\left(T_{w0} - T_j\right)}{V_0} + \frac{UA\left(T - T_j\right)}{\rho_w C_{pw} V_0} \tag{6}$$

where, $P_0 = \left(2 f' C_I k_I / \left(k_{td} + k_{tc}\right)\right)^{1/2}$ and the rate constants used in this process are derived using (7).

$$k_s = A_s e^{-E_s/RT} \tag{7}$$

where, s can be substituted with parameters p, fm, I, td, tc and the respective values of E_s and A_s [1], as mentioned in Table 1.

III. Kalman Filter

Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm to produce estimates of unknown variables that tend to be more accurate than those based on a single measurement alone. It uses a series of measurements

TABLE I: Nomenclature

F	= flow rate of the monomer,
C_{min}	= concentration of the monomer in the feed,
C_m	= concentration of monomer inside the reactor,
V	= volume of the reactor,
F_I	= flow rate of the initiator,
C_{Iin}	= initiator concentration in the feed F_I,
C_I	= concentration of the initiator inside the reactor,
T_{in}	= temperature of the monomer,
T	= reactor temperature,
T_j	= jacket temperature,
ρ	= density of the monomer,
C_p	= heat capacity of the monomer,
D_i	= polymer moments,
F_{cw}	= cooling water flow rate,
T_{w0}	= cooling water temperature,
V_0	= volume of the jacket,
ρ	= density of the cooling water, and
C_{pw}	= heat capacity of the cooling water

observed over time, containing statistical noise and other inaccuracies, and estimates a joint probability distribution over the variables for each timeframe. In the problem statement under consideration, six states are defined - C_m, C_I, D_0, D_I, T and T_j. To estimate these states using a Kalman filter, however, discrete-time linear model is required [4], as opposed to the differential dynamics equations mentioned in Eq. (1)-(6). To obtain a discrete-time linear model these differential equations are firstly expanded using Taylor series for multi-variable functions, about a steady state point.

$$\frac{dX}{dt} = f(X, U, D) \tag{8}$$

$$\begin{aligned}\frac{dX}{dt} \approx f(X^*, U^*, D^*) + A(X - X^*) \\ + B(U - U^*) + H(D - D^*)\end{aligned} \tag{9}$$

Where X denotes the states, U is the physical input, and D is the process noise, which is assumed to be drawn from a zero mean multivariate normal distribution, with covariance Q. Further, deviation variables are defined as in (10).

$$x = X - X^*, u = U - U^*, d = D - D^* \tag{10}$$

Substituting these in Eq.(9), Eq. (11) is obtained

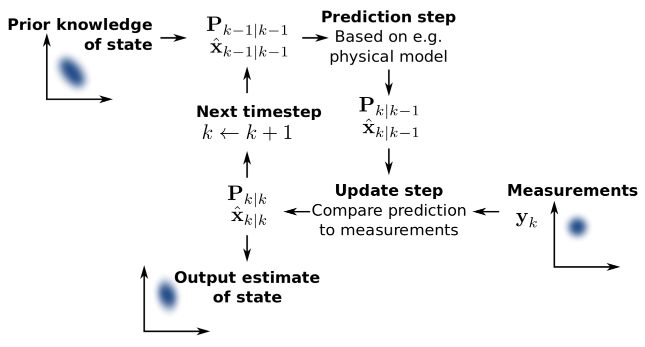
$$\frac{dx}{dt} \approx f(X^*, U^*, D^*) + Ax + Bu + Hd \tag{11}$$

Since the expansion is done about the steady state point, $f(X^*, U^*, D^*) = 0$. The linear system thus obtained is:

$$\frac{dx}{dt} = Ax + Bu + Hd \tag{12}$$

Which when integrated over a discrete time period, yields

$$x(k+1) = \phi x(k) + \gamma_u u(k) + \gamma_d d(k) \tag{13}$$



Prior knowledge of state
$\mathbf{P}_{k-1|k-1}$ $\hat{\mathbf{x}}_{k-1|k-1}$
Prediction step
Based on e.g. physical model
$\mathbf{P}_{k|k-1}$ $\hat{\mathbf{x}}_{k|k-1}$
Next timestep
$k \leftarrow k+1$
$\mathbf{P}_{k|k}$ $\hat{\mathbf{x}}_{k|k}$
Update step
Compare prediction to measurements
Measurements
$\mathbf{y}_k$
Output estimate of state

Fig. 1: Kalman Filter flowchart [4]

Further, the predicted state and measurement are used to predict the final true state, each contributing according to kalman gain. Figure 1 shows Kalman Filter in the form of a flowchart.

To initialize a Kalman Filter, first some simulation parameters are provided, state noise Q and measurement noise R, which require some engineering insight to set heuristically. Moreover, an initial guess of the state $\hat{x}_{0|0}$ and covariance of this state $P_{0|0}$ must be provided. Availability of process model is assumed. Then further three steps are followed : (a)Prediction (b)Gain Computation (c)Update. The prediction step involves simply using the filtered states $\hat{x}_{k-1|k-1}$ and $P_{k-1|k-1}$, and running them through the provided process model. Equations (14)-(15) show how we obtain the resultant $\hat{x}_{k|k-1}$ and $P_{k|k-1}$.

$$\hat{x}_{k|k-1} = \phi \hat{x}_{k-1|k-1} + \Gamma_u u(k) \tag{14}$$

$$P_{k|k-1} = \phi P_{k-1|k-1} \phi^T + \Gamma_d Q_d \Gamma_d^T \tag{15}$$

This step is followed by the Kalman Gain computation given in Equation (16).

$$L^*(k) = P_{k|k-1} C^T [C P_{k|k-1} C^T + R]^{-1} \tag{16}$$

Now to optimally combine predictions with the information with Kalman Filter, we use Equations (17)-(19).

$$e(k) = [y(k) - C\hat{x}_{k|k-1}] \tag{17}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + L^*(k) e(k) \tag{18}$$

$$P_{k|k} = [I - L^*(k) C] P_{k|k-1} \tag{19}$$

This process is recursive, with the newly obtained $\hat{x}_{k|k}$ and $P_{k|k}$ are sent to the filter again to optimally combine these estimates with new measurements. At the same time, the current state can now be sent to our control strategy, so as to control the system. [4]

IV. Implementation Details

The main framework used was Pytorch.

The operating parameters used in the process simulation are mentioned in Table II and those for simulation are available in Table III. $x(0)$ denotes the initial state for true state generation; R, Q are the co-variance matrices for

measurement variable Y and state X; initial condition for estimators are given by $\hat{x}_{0|0}$ and $P_{0|0}$.
For the purpose of this simulation, 1000 measurements (variables T and T_j) are taken at an interval of 28.8 seconds, amounting to 8 hours. The rest of the states (C_m, C_I, D_0, and D_I) are unmeasured. The two inputs of F, feed flow rate, and F_{cw}, cooling water flow rate, are kept constant at values of 1 m^3/h and 0.159 m^3/h respectively. The required values and functions are defined in the file 'model.py' and 'util.py'.
There were three major parts that were implemented:

A. Kalman Filter

Implementation of kalman filter was done using the f.solve solver in the scipy library and the torch grad library. Firstly, a steady state operating point was found by differentiating through the torch differentiation and then equating it to zero to get the steady state point. From the steady state point the system was linearized. Then Jacobian of the linearized system was generated by using the torch autograd function. By using the Jacobian, a continuous time linear perturbation model was developed for the system. From the continuous model, a discrete model was generated using the cont2discrete functionality in the scipy library, as needed for kalman filter implementation.

B. State data generator and addition of noise

Each sample indicates measurements taken after time period of 0.008 hours from the reactor The data was generated by selecting an initial value of state of reactor at time t=0. The initial values were chosen from A.Silva-Beard,1999 [1]. The for every time step k+1, the differential equation of state was solved, by integrating it from k to k+1 time period, to get the true value of state. To this true value, state noise was added by sampling from multivariate normal distribution having co-varaiances as taken from the reference [1], to get the measured value of the states. This value was then used in state equations to get the temperature value (these are basically physical equations). Finally, measurement noise, which was sampled from multivariate normal distribution having variances as provided by the measurement noise vector (which was taken from reference), was added to the temperature values to get measured value of temperature. This temperature value was the one that was fed to the Kalman filter.

C. Application of Kalman filter to estimate the state

This included application of the kalman filter to the noisy data to predict the true states of the system. Basic matrix multiplication from the torch library was used to implement the matrix equations given in the previous section to iteratively solve for the true states. The estimated states and measurements, after the addition of noise, predict the final state. The contribution from both of these is based on Kalman Gain value.

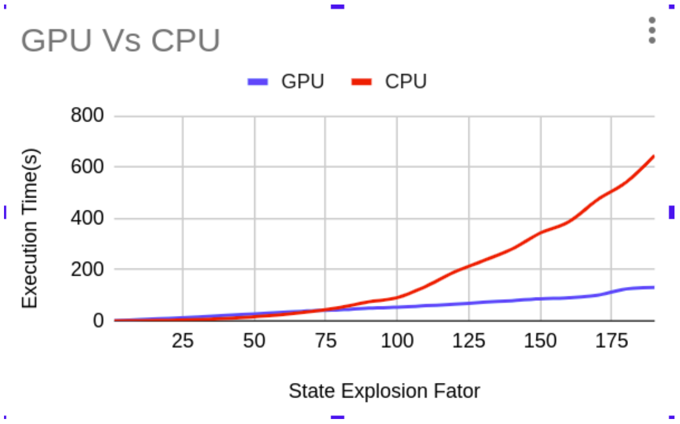


Fig. 2: Explosion factor vs Compute Time

V. Results and Discussions

Multiple experiments were carried out in order to highlight the effectiveness of the parallel computing through the use of GPU. The data for the simulation was generated for samples separated by 0.008h and 1000 such samples were generated. On the data that we generated, Kalman filter was applied for estimation of the states. When,we ran the experiments for 6 states, the time required for CPU (0.33s) was less than that required for GPU (1.013s). So we increased the number of states, by a factor that we would refer to as state explosion factor (M). The increase in states was done by replicating the state vector M number of times and then concatenating the vectors, to a vector of size 6*M, similarly the noise matrices were replicated M number of times and were then attached to get block diagonal matrices of size (6M)*(6M). These block diagonal matrices were then used as co-variance matrices for multivariate normal distribution. The **most important observation** of this project was that, the CPU compute time was less than the GPU compute time for up-to state explosion factor of 80 (that is number of states = 480), as shown in the Table IV After that the difference between the CPU and GPU compute time grew exponentially as shown in Figure 2. The less time required for the CPU can be explained by the fact that the size of the matrix is so small that no parallelization is required and the overhead in this case becomes the time to transfer data from RAM to GPU. [3]
The true states and the estimated states of the Kalman filters is shown in Figure 3,

VI. Conclusion and Future Work

State estimation was performed, for various number of states, in this project. Most important insight obtained was regarding the choice of CPU and GPU based on size of matrices that are used for computations. In state estimation problems where, the online (real time) calculations have matrix size less than 500*500, CPU should be used, this is typically the case for most of process engineering applications. However when the state space is large like

TABLE II: Operating Parameters for MMA process

$f' = 0.58$	$F_I = 0.0032\ \mathrm{m}^3/\mathrm{h}$
$V = 0.1\ \mathrm{m}^3$	$A = 2\ \mathrm{m}^2$
$V_0 = 0.02\ \mathrm{m}^3$	$T_s = 0.008\ \mathrm{h}$
$T_{w0} = 293.2\ \mathrm{K}$	$T_{in} = 350\ \mathrm{K}$
$\rho = 866\ \mathrm{kg}/\mathrm{m}^3$	$R = 8.314\ \mathrm{kJ}/(\mathrm{kgmol.K})$
$\rho_w = 1000\ \mathrm{kg}/\mathrm{m}^3$	$M_m = 100.12\ \mathrm{kg}/\mathrm{kgmol}$
$C_{\mathrm{min}} = 6.4678 \mathrm{kgmol}/\mathrm{m}^3$	$C_{pw} = 4.2\ \mathrm{kJ}/(\mathrm{kg.K})$
$C_{Iin} = 8.0 \mathrm{kgmol}/\mathrm{m}^3$	$C_p = 2.0\ \mathrm{kJ}/(\mathrm{kg} \cdot \mathrm{K})$
$U = 720\ \mathrm{kJ}/\left(\mathrm{h.K} \cdot \mathrm{m}^2\right)$	$-\Delta H = 57800\ \mathrm{kJ}/\mathrm{kgmol}$
$A_I = 3.792 \times 10^{18} \mathrm{l}/\mathrm{h}$	$E_I = 1.2877 \times 10^5\ \mathrm{kJ}/\mathrm{kgmol}$
$A_{tc} = 3.8223 \times 10^{10}\ \mathrm{m}^3/(\mathrm{kgmol.h})$	$E_{tc} = 2.9422 \times 10^3\ \mathrm{kJ}/\mathrm{kgmol}$
$A_{td} = 3.1457 \times 10^{11}\ \mathrm{m}^3/(\mathrm{kgmol.h})$	$E_{td} = 2.9422 \times 10^3\ \mathrm{kJ}/\mathrm{kgmol}$
$A_{fm} = 1.0067 \times 10^{15}\ \mathrm{m}^3/(\mathrm{kgmol.h})$	$E_{fm} = 7.4478 \times 10^4\ \mathrm{kJ}/\mathrm{kgmol}$
$A_p = 1.77 \times 10^9\ \mathrm{m}^3/(\mathrm{kgmol.h})$	$E_p = 1.8283 \times 10^4\ \mathrm{kJ}/\mathrm{kgmol}$

TABLE III: Simulation Parameters for MMA process

$x(0) = [5.9655\ 0.0249\ 0.0020\ 50.3287\ 351.4013\ 332.90774]^T$
$Q = diag\{3.2028 \times 10^6; 6.2 \times 10^{12}; 25 \times 10^{14}; 25.28 \times 10^6; 12.34 \times 10^4; 11.08 \times 10^4\}$
$R = diag\{2.5 \times 10^1 2.5 \times 10^1\}$; $\hat{x}_{0\|0} = 1.025 \times x_{0\|0}$; $P_{0\|0} = 10 \times Q$

TABLE IV: Compute Times for various State Explosion factors

State Explosion Factor	GPU Time(s)	CPU Time(s)
1	1.013	0.33
10	4.96	0.9
20	10.17	2.45
30	15.59	5.28
40	21.27	9.48
50	27.48	16.53
60	33.4	24.72
70	39.74	37.05
80	43.41	52.75
90	49.89	74.61
100	53.46	90.92

more than 1000, such as in robotic applications, GPUs should be preferred. One immediate extension of this work can be using the values of the estimated states to control the system using model predictive control or other control strategy. Parallelized implementation becomes very important in such cases as the state estimation is done online (real time) and the control actions are also taken in the real time. Another extension could be dividing the system into parts and defining the state variables for each of the parts,currently our system assumes that the state variables are uniform across the system, which is not the case in most of large scale reactors. Then state estimation can be carried with the help of more powerful non linear state estimators like neural networks in order to model the interaction between the parts of the system. These can be used to predict adverse phenomena such as Trommsdorff–Norrish [2] effect which occurs due to localized increases in viscosity of the polymerizing system. The localized changes are the functions of the state of the state of system as well as the history of the states of the system.

VII. Code

The code was run on Google Colab, available here. The main code is also available as an iPython notebook titled ”Large State Estimation with Kalman Filter.ipynb”. To run the code, you must provide additionally the files titled ”model.py” and ”util.py” in the same directory as the notebook. Note that the respository being cloned from GitHub does exactly this (first cell of the notebook). You may avoid this by uploading the two files manually, provided in the submission.

VIII. Acknowledgment

We would like to thank Prof. S. Gopalakrishnan for giving us the opportunity to work on this project.

References

[1] Silva-Beard, Andrea, and Antonio Flores-Tlacuahuac. ”Effect of process design/operation on the steady-state operability of a methyl methacrylate polymerization reactor.” Industrial & engineering chemistry research 38.12 (1999): 4790-4804.

[2] O'Shaughnessy, Ben, and Jane Yu. ”Autoacceleration in free radical polymerization. 1. Conversion.” Macromolecules 27.18 (1994): 5067-5078.

[3] Dobravec, Tomaž, and Patricio Bulić. ”Comparing CPU and GPU implementations of a simple matrix multiplication algorithm.” Int. Jour. of Computer and Electrical Engineering 9.2 (2017): 430-438.

[4] Wikipedia contributors, ”Kalman filter,” Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Kalman_filter&oldid1022365423 (accessed May 16, 2021).

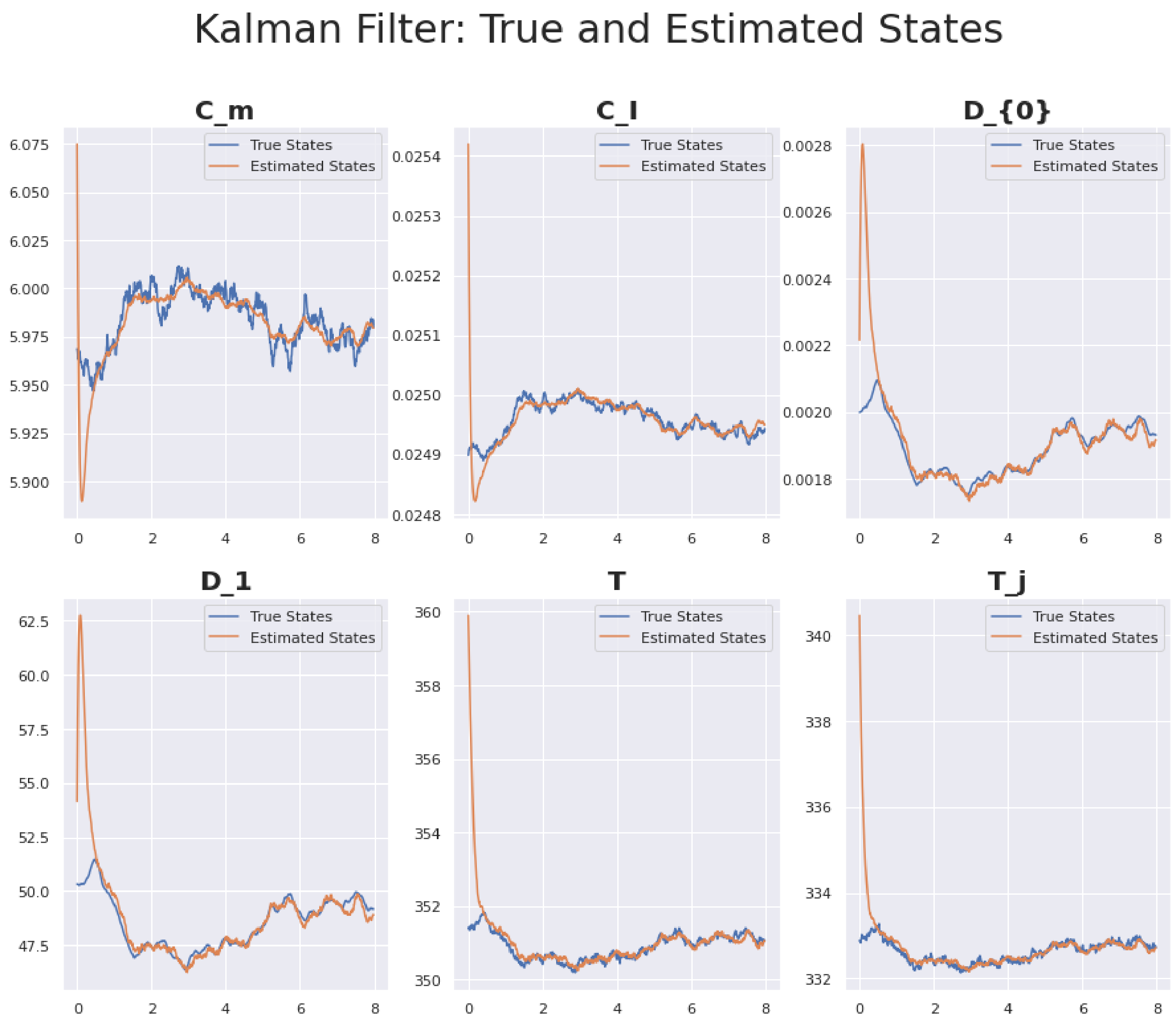


Fig. 3: Kalman Filter Results