Lattice reconstruction using TBT data with Kalman filters

C.Y. Tan*

Fermi National Accelerator Laboratory,

P.O. Box 500, Batavia, IL 60510-5011, USA.

(Dated: June 21, 2019)

Abstract

I will show how the machine lattice can be constructed from TBT data with the joint Kalman filter (JKF). The JKF algorithm finds a fit to the TBT data so that the beta functions can be constructed and lattice errors can be identified. Another feature of JKF is that noise from the BPMs, pulse to pulse variations of the injected beam, error tolerances of the results can all be taken into account. As a demonstration of JKF, I will show how the JKF can reconstruct the lattice and identify the source of the error when it is applied to TBT (turn by turn) data generated from a toy FODO lattice with a quad strength error.

* cytan@fnal.gov

I. INTRODUCTION

The Kalman filter (invented in the 1960s [1]) is a very well-known method used in HEP track reconstruction[2], aircraft feedback control[3], biological systems[4] etc. The power of the Kalman filter is that it is a self-correcting algorithm. Intuitively, the Kalman filter takes place in two steps: *a priori* knowledge and a model to predict an outcome. And then, it uses the *a posteriori* knowledge of an actual measurement to correct itself before making the next prediction.

In the context of lattice construction from turn by turn (TBT) BPM data, the necessary parameters that have to be determined from the TBT data are the transverse position and angle of the beam at every BPM together with the errors in the real world implementation of the lattice like quadrupole strengths, rotations of beam line elements etc.

Notice that since BPMs only return position information, the remaining phase space parameters and implementation errors will have to be constructed by the Kalman filter from the TBT data. This is only possible when there is a lattice model that is not too far from reality. In this paper, I will use the joint Kalman filter algorithm (JKF) for analysing the TBT data. Note: there is another algorithm called the dual Kalman filter (DKF) which I think should be used in the future. The DKF requires a smaller matrix inversion compared to JKF.

II. STATE VECTORS

The parameters that I will need to find with the Kalman filter are contained in an object called the state vector $\boldsymbol{X}$. It contains both the transverse phase space description of the beam, $\boldsymbol{x}$ and the real-world parameters errors $\boldsymbol{\theta}$

$$\boldsymbol{X}^{(i)}(n) = \begin{pmatrix} \boldsymbol{x}^{(i)}(n) \\ \boldsymbol{\theta}^{(i)} \end{pmatrix} \tag{1}$$

where (i) means that $\boldsymbol{X}$, $\boldsymbol{x}$ and $\boldsymbol{\theta}$ are measured at BPM i at the nth turn. Notice that I have made $\boldsymbol{\theta}$ independent of n because errors like quadrupole strengths and element rotations should be constants.

A. Phase space vectors

In the TBT method, the beam is pinged and then allowed to "freely" oscillate. The ping can be either horizontal or vertical. And after N turns, I can write down N vectors that represent the phase space at BPM i:

$$\boldsymbol{x}^{(i)}(0) = \begin{pmatrix} x_0 \\ x'_0 \\ \delta_0 \\ y_0 \\ y'_0 \end{pmatrix}^{(i)} \quad \boldsymbol{x}^{(i)}(1) = \begin{pmatrix} x_1 \\ x'_1 \\ \delta_1 \\ y_1 \\ y'_1 \end{pmatrix}^{(i)} \quad \ldots \quad \boldsymbol{x}^{(i)}(N-1) = \begin{pmatrix} x_{N-1} \\ x'_{N-1} \\ \delta_{N-1} \\ y_{N-1} \\ y'_{N-1} \end{pmatrix}^{(i)} \tag{2}$$

where $(x_n, y_n)^{(i)}$ are the horizontal and vertical position of the bunch, $(x'_n, y'_n)^{(i)}$ are the horizontal and vertical angle of the bunch at BPM i at turn n. And $\delta_n^{(i)} = (\delta p/p)_n^{(i)}$ is the fractional momentum error at at BPM i at turn n. For this paper, only the horizontal plane is affected by dispersion. Thus when I say horizontal phase space, I mean (x, x', δ) and vertical phase space is (y, y').

B. Parametric errors

In the real world implementation of the lattice, there will be errors. These errors can be discovered from the TBT data provided that the distortions caused by them are not large, i.e. these errors should be perturbations of the model lattice. Examples of parametric errors are BPM rotation errors, BPM calibration errors and quad strength errors, etc.

At each BPM i, I will parametrize each of these errors into a vector $\boldsymbol{\theta}^{(i)}$. For the purposes of this analysis, I will assume that $\boldsymbol{\theta}^{(i)}$ is <u>stationary</u>. With this assumption, the propagation matrix between turns for $\boldsymbol{\theta}^{(i)}$ is simply the identity matrix.

III. PROPAGATION WITH PARAMETRIC ERRORS

I will divide up the discussion about how the bunch is propagated from from BPM i to $i+1$ into two parts below. The first is the ideal case where there are no parametric errors. The second is when parametric errors are included.

A. Ideal transport with process noise

I define $\boldsymbol{M}_{i\to i+1}$ to be the matrix <u>without any parametric errors</u> that transports the beam from BPM i to BPM $(i+1)$ with the condition that $(i+1)\to 1$ if $i=K$. The entire TBT transport sequence for N turns and K BPMs is as follows

$$\left.\begin{aligned}
\boldsymbol{x}^{(2)}(0) &= \boldsymbol{M}_{1\to 2}\boldsymbol{x}^{(1)}(0) + \boldsymbol{w}^{(1)}(0) \quad \text{Start at turn 0}\\
\boldsymbol{x}^{(3)}(0) &= \boldsymbol{M}_{2\to 3}\boldsymbol{x}^{(2)}(0) + \boldsymbol{w}^{(2)}(0)\\
&\vdots\\
\boldsymbol{x}^{(K)}(0) &= \boldsymbol{M}_{K-1\to K}\boldsymbol{x}^{(K-1)}(0) + \boldsymbol{w}^{(K-1)}(0) \quad \text{Turn 0 is done}\\
\boldsymbol{x}^{(1)}(1) &= \boldsymbol{M}_{K\to 1}\boldsymbol{x}^{(K)}(0) + \boldsymbol{w}^{(1)}(1) \quad \text{Next is turn 1}\\
&\vdots\\
\boldsymbol{x}^{(K)}(N-1) &= \boldsymbol{M}_{K-1\to K}\boldsymbol{x}^{(K-1)}(N-1) + \boldsymbol{w}^{(K-1)}(N-1) \quad \text{Done!}
\end{aligned}\right\} \tag{3}$$

Since the phase space vector has dimension 5, $\boldsymbol{M}_{i\to i+1}$ must be a 5×5 matrix

$$\boldsymbol{M}_{i\to i+1}\boldsymbol{x}^{(i)}(n) = \begin{pmatrix} m_{11} & m_{12} & m_{13} & 0 & 0\\ m_{21} & m_{22} & m_{23} & 0 & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & m_{44} & m_{45}\\ 0 & 0 & 0 & m_{54} & m_{55} \end{pmatrix}_{i\to i+1} \begin{pmatrix} x_n\\ x'_n\\ \delta_n\\ y_n\\ y'_n \end{pmatrix}^{(i)} \tag{4}$$

where $\boldsymbol{w}^{(i)}(n)$ is called the *process* noise of the system. The process noise here can come from the pulse to pulse injection errors of the beam into Booster.

B. Parametric errors

Next, I have to include parametric errors into the TBT transport sequence. Eq. 3 is modified as follows:

$$\left.\begin{aligned}
\boldsymbol{X}^{(2)}(0) &= \boldsymbol{F}_{1\to 2}[\boldsymbol{X}^{(1)}(0), \boldsymbol{W}^{(1)}(0)] \quad \text{Start at turn 0}\\
&\vdots\\
\boldsymbol{X}^{(K)}(0) &= \boldsymbol{F}_{K-1\to K}[\boldsymbol{X}^{(K-1)}(0), \boldsymbol{W}^{(K-1)}(0)] \quad \text{Turn 0 is done}\\
&\vdots\\
\boldsymbol{X}^{(K)}(N-1) &= \boldsymbol{F}_{K-1\to K}[\boldsymbol{X}^{(K-1)}(N-1), \boldsymbol{W}^{(K-1)}(N-1)] \quad \text{Done!}
\end{aligned}\right\} \tag{5}$$

where $\boldsymbol{W}$ is the process noise that affects both $\boldsymbol{x}$ and $\boldsymbol{\theta}$.

As for $\boldsymbol{F}_{i\to i+1}()$ (that includes $\boldsymbol{M}_{i\to i+1}$), I will show an example of how they look like in Section IV D.

IV. JOINT KALMAN FILTER ALGORITHM

In this section, I will describe the JKF algorithm that I will apply to the TBT data. I will not derive the JKF algorithm because it is covered elsewhere.[5–7] As I had stated in the *Introduction*, the JKF consists of two steps that I will discuss below. However, before I start I will introduce new notation for the state vector to differentiate its value at the *a priori* or "prediction' step and the *a posteriori* or "correction" step. Here's my notation:

$$\left.\begin{array}{ll} \overline{\boldsymbol{X}}^{(i)}(n): & \text{state vector at BPM } i \text{ at turn } n \text{ in the prediction step.} \\ & \text{This vector is called the } \underline{\text{predicted}} \text{ state vector.} \\ \widehat{\boldsymbol{X}}^{(i)}(n): & \text{state vector at BPM } i \text{ at turn } n \text{ after the correction step.} \\ & \text{This vector is called the } \underline{\text{estimated}} \text{ state vector.} \end{array}\right\} \tag{6}$$

A. *A priori* or prediction step

In the prediction part, I have the following at BPM i at turn n

$$\overline{\boldsymbol{X}}^{(i)}(n) = \boldsymbol{F}_{i-1\to i}[\widehat{\boldsymbol{X}}^{(i-1)}(n-1), \boldsymbol{W}^{(i-1)}(n-1)] \tag{7}$$

i.e. the above equation gives the predicted state vector at BPM i given the estimated state vector from BPM $i-1$. Note: $\overline{\boldsymbol{X}}^{(i)}$ is clearly model dependent because it uses a model to make a prediction.

The predicted covariance is also calculated in this step and it is given by

$$\overline{\boldsymbol{\Sigma}}^{(i)}(n) = \widehat{\boldsymbol{A}}^{(i)}(n-1)\widehat{\boldsymbol{\Sigma}}^{(i)}(n-1)[\widehat{\boldsymbol{A}}^{(i)}(n-1)]^T + \boldsymbol{Q}(n-1) \tag{8}$$

where

$$\widehat{\boldsymbol{A}}^{(i)}(n) = \left.\frac{d\boldsymbol{F}_{i-1\to i}[\boldsymbol{X}, \boldsymbol{W}^{(i-1)}(n-1)]}{d\boldsymbol{X}}\right|_{\boldsymbol{X}=\widehat{\boldsymbol{X}}^{(i-1)}(n-1)} \tag{9}$$

is the Jacobian of $\boldsymbol{F}_{i-1\to i}$ evaluated with the value of the estimated state vector $\widehat{\boldsymbol{X}}^{(i-1)}(n-1)$

at BPM $i-1$ from the previous turn; and

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{\Sigma}_{\boldsymbol{w}}^2 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^2 \end{pmatrix} \tag{10}$$

is the covariance matrix of the process noise. Here, $\boldsymbol{\Sigma}_{\boldsymbol{w}}^2$ is the covarance of the process noise, $\boldsymbol{w}$, that affects the phase space part of the state vector. $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^2$ is the covarance of the process noise that affects the parameters part of the state vector. (For TBT, my guess is that $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^2$ should be interpreted as the variance of the tolerance.)

B. *A posteriori* or correction step

In the *a poteriori* or correction step, the algorithm corrects the predicted state vector $\overline{\boldsymbol{X}}^{(i)}(n)$ from the previous step with a measurement. In the TBT case, the correction comes from the position of the beam, $\zeta_n^{(i)}$ measured by BPM i at turn n,

$$\boldsymbol{\zeta}^{(i)}(n) = \begin{pmatrix} \xi_n \\ \Delta_n \\ \eta_n \end{pmatrix}^{(i)} \tag{11}$$

where ξ_n is the x position, Δ_n is the dispersion, and η_n is the y position measured by BPM i at turn n.

The predicted position $\bar{\boldsymbol{z}}^{(i)}(n)$ is found by projecting out the transverse position coordinates from $\overline{\boldsymbol{X}}^{(i)}(n)$ with matrix $\boldsymbol{H}$

$$\bar{\boldsymbol{z}}^{(i)}(n) = \boldsymbol{H}\overline{\boldsymbol{X}}^{(i)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 \end{pmatrix} \begin{pmatrix} x_n \\ x'_n \\ \delta_n \\ y_n \\ y'_n \\ \theta_1 \\ \vdots \\ \theta_M \end{pmatrix}^{(i)} = \begin{pmatrix} x_n \\ \delta_n \\ y_n \end{pmatrix}^{(i)} \tag{12}$$

whre I have assumed that there are M parametric errors.

The Kalman gain $\boldsymbol{K}$ is

$$\boldsymbol{K} = \overline{\boldsymbol{\Sigma}}^{(i)}(n)\boldsymbol{H}^T \left(\boldsymbol{H}\overline{\boldsymbol{\Sigma}}^{(i)}(n)\boldsymbol{H}^T + \boldsymbol{R}\right)^{-1} \tag{13}$$

where $\boldsymbol{R}$ is the error covariance vector of the BPM measurement, i.e.

$$\boldsymbol{R} = \begin{pmatrix} \sigma_\xi^2 \\ \sigma_\Delta^2 \\ \sigma_\eta^2 \end{pmatrix} \tag{14}$$

With the above, the estimated state vector, $\widehat{\boldsymbol{X}}^{(i)}(n)$, is

$$\widehat{\boldsymbol{X}}^{(i)}(n) = \overline{\boldsymbol{X}}^{(i)}(n) + \boldsymbol{K}(\boldsymbol{\zeta}^{(i)}(n) - \boldsymbol{z}^{(i)}(n)) \tag{15}$$

and the estimated error covariance matrix is

$$\widehat{\boldsymbol{\Sigma}}^{(i)}(n) = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\overline{\boldsymbol{\Sigma}}^{(i)}(n) \tag{16}$$

C. Application to TBT data

Here my pseudo-code for analyzing the TBT data. Note: the iterative counter range for both n and i is not the same as what I used in `Mathematica` because counting has to start from "1".

1. <u>Initialization</u> At initialization, I will set the estimated state vector at BPM 1 to the measured values for position and dispersion, and the angles and parametric errors to zero, i.e.

$$\widehat{\boldsymbol{X}}^{(i)}(0) = \begin{pmatrix} \xi_0 \\ 0 \\ \Delta_0 \\ \eta_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}^{(1)} \tag{17}$$

The estimated covariance errors have to be large here because the angles and parametric errors are probably not zero. I have set it to a $(5+M)\times(5+M)$ identity matrix:

$$\widehat{\boldsymbol{\Sigma}}^{(1)}(0) = \boldsymbol{I}_{(5+M)\times(5+M)} \tag{18}$$

2. Iterate For each turn $n = 1$ to N do

   (a) For each BPM $i = 2$ to $K+1$ do

      i. Check to see if the transport should wrap from BPM K back to BPM 1. See Eq. 3.

      ii. Prediction Calculate the predicted state vector and covarance error. An example of $\boldsymbol{F}$ can be found in the example shown in section IV D

$$\left.\begin{aligned} \overline{\boldsymbol{X}}^{(i)}(n) &= \boldsymbol{F}_{i-1\to i}[\widehat{\boldsymbol{X}}^{(i-1)}(n-1), \boldsymbol{W}^{(i-1)}(n-1)] \\ \overline{\boldsymbol{\Sigma}}^{(i)}(n) &= \widehat{\boldsymbol{A}}^{(i)}(n-1)\widehat{\boldsymbol{\Sigma}}^{(i)}(n-1)[\widehat{\boldsymbol{A}}^{(i)}(n-1)]^T + \boldsymbol{Q}(n-1) \end{aligned}\right\} \tag{19}$$

      iii. Calculate Kalman gain Caclulate the Kalman gain

$$\boldsymbol{K} = \overline{\boldsymbol{\Sigma}}^{(i)}(n)\boldsymbol{H}^T\left(\boldsymbol{H}\overline{\boldsymbol{\Sigma}}^{(i)}(n)\boldsymbol{H}^T + \boldsymbol{R}\right)^{-1} \tag{20}$$

      iv. Correction Calculate the estimated state vector and covariance error

$$\left.\begin{aligned} \widehat{\boldsymbol{X}}^{(i)}(n) &= \overline{\boldsymbol{X}}^{(i)}(n) + \boldsymbol{K}(\boldsymbol{\zeta}^{(i)}(n) - \boldsymbol{z}^{(i)}(n)) \\ \widehat{\boldsymbol{\Sigma}}^{(i)}(n) &= (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\overline{\boldsymbol{\Sigma}}^{(i)}(n) \end{aligned}\right\} \tag{21}$$

   (b) end of i loop.

3. end of n loop.

D. FODO example

In this example, I have made a simple 3 cell FODO lattice. For this toy model, I will only look at the (x, x') phase space with zero δ.

The strengths of the quads, length of the quads and the drift space are summarized in Table. I. I have placed a BPM at the centre of each quad magnet. See Fig. 1.

The comparison between the β function of an ideal FODO and the lattice with the red D quad (Q4) strength increased by 10% is shown in Fig. 2.

Quad strength k	Length of quad ℓ_q	length of drift ℓ_d
0.54102 m^{-2}	0.5 m	2.5 m

TABLE I. The parameters of a FODO cell.

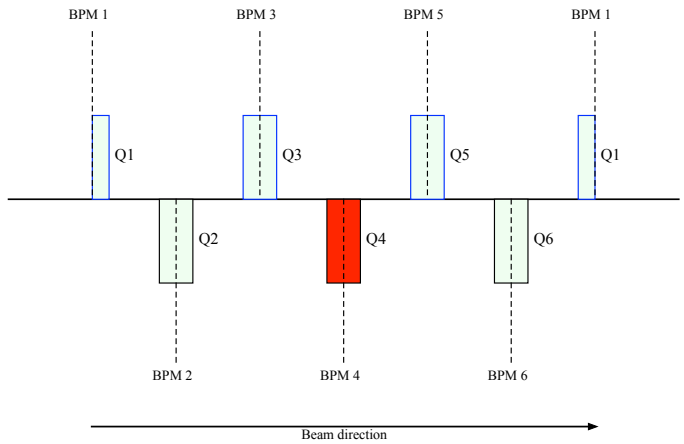

FIG. 1. This is the 3 cell FODO lattice that I have used for my example. There is a BPM at the centre of each quad. The injected beam enters at BPM 1. The red D quad (Q4) is the one where I have increased its k value by 10%.

1. *TBT data*

I generated the TBT data by assuming that $(x_0, x_0') = (0.001, 0.001)$, i.e. 1 mm offset and 1 mrad angle. I added Gaussian noise that has $\sigma = 0.05$ mm to the x data to simulate measurement noise from the BPM.

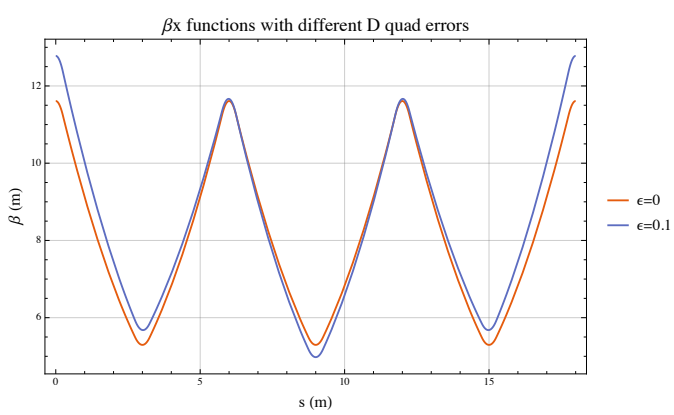

FIG. 2. This is the comparison between the β functions of the ideal FODO lattice and when the red D quad (Q4) strength is increased by 10%.

An example of the TBT data that I generated at BPM 1 is shown in Fig. 3. I do this for BPMs 2 to 6.

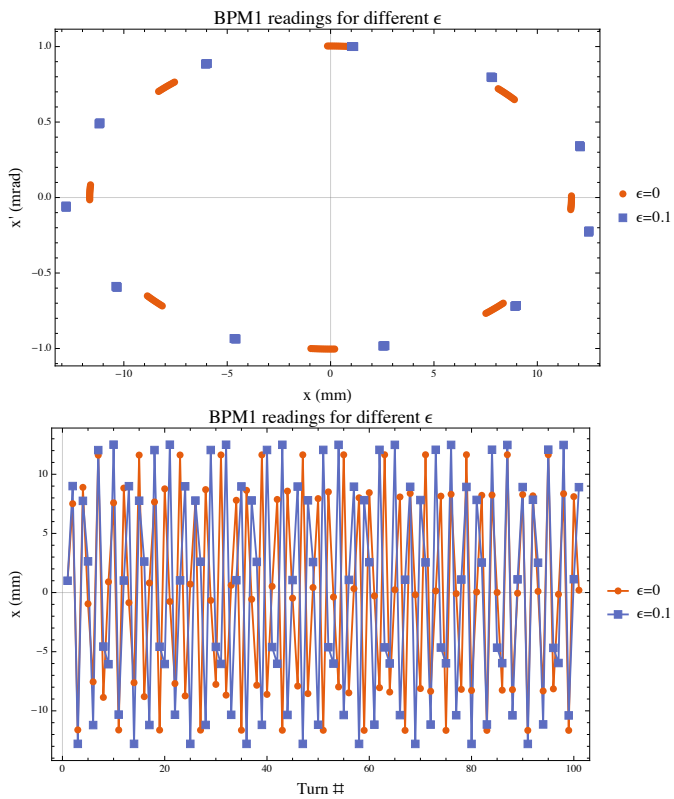

FIG. 3. This is the position and phase space at BPM 1 for 100 turns when the initial state is $(x_0, x_0') = (0.001, 0.001)$. I have plotted the results for the ideal lattice and when the red D quad (Q4) strength is increased by 10%.

2. *The fitting model*

The fitting model consists of thin quads that are shown in Fig. 4. For simplicity, (or my laziness), I have put two thin corrector quads per quad. These quads are parametrized as

$$\Delta Q_{\text{err}} = \begin{pmatrix} 1 & 0 \\ -\theta & 0 \end{pmatrix} \tag{22}$$

so that $\theta = 1/(\text{focal length})$ of the thin quad. The location of each of these thin corrector quads are shown in Fig. 4.

And because of the odd-ball way I have made the thin lenses, the focal length from the

combination of these two lenses can be found with the following formula

$$\Delta f = \frac{f_\theta(f_\theta - \ell_q)}{2f_\theta - \ell_q} + \frac{l_q}{2} \tag{23}$$

where $f_\theta = -1/\theta$ and the value of l_q is shown in Table I. Note: I have to add $l_q/2$ to the first term because the first term is the focal length w.r.t. the downstream thin lens. To bring it back to the centre of the quad, I have to add $l_q/2$.

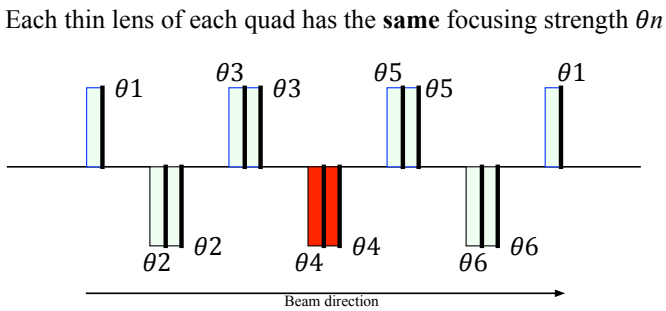

FIG. 4. I have placed 2 thin lenses per quadrupole. These strength of these thin lenses will come from the TBT data.

3. *Kalman filtering*

One advantage of the Kalman filter is that noise is taken into account. Table II summarizes the σ of the Gaussian noise used for the algorithm. I have used a non-zero value for the process noise variance, i.e. injection pulse to pulse variation, but I have set $\boldsymbol{W}^{(i)} = \mathbf{0}$ for the analysis — I think this should be okay.

I have set the following initial values to start the Kalman filter:

$$\widehat{\boldsymbol{X}}^{(1)}(0) = \begin{pmatrix} \xi^{(1)}(0) = 0.001 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \hat{\boldsymbol{\Sigma}}^{(1)}(0) = \boldsymbol{I}_{8\times 8} \tag{24}$$

Description	Symbol	Value
BPM read noise	σ_ξ	0.05 mm
Injection pulse to pulse variation (process noise)	$\boldsymbol{\Sigma_W}$	$\begin{pmatrix} 0.01 \text{ mm} \\ 0.01 \text{ mrad} \end{pmatrix}$
Quad focus strength error $\theta^{(i)}$ error tolerance	$\boldsymbol{\Sigma_\theta}$	$\begin{pmatrix} 0.01 \times 10^{-3} \text{ m}^{-1} \\ \vdots \\ 0.01 \times 10^{-3} \text{ m}^{-1} \end{pmatrix}$

TABLE II. The σ of the Gaussian noise and tolerances used for the Kalman filter. I am calling $\boldsymbol{\Sigma_\theta}$ the error tolerance because I don't think it makes sense to call it "noise" here.

4. Results

The results of the Kalman fits are shown in the following Figures 5, 6, and 7. The Kalman fit to the "measured" x (noisy ξ BPM data) gives me the estimated value of x'. This result, at least by eye, looks very good when compared to the underlying known model.

From Fig. 7, all the quad strength errors are small except for $Q4$ which is where I had placed the error. Here, it is obvious why the Kalman filter is a good method: (a) it shows how the parameter converges to the "correct" value and (b) it also shows the error of the result.

From the last value of the $\theta 4$ in Fig. 7, I have $\theta 4 = (-0.0141 \pm 0.0005)$ m^{-1}. And by using Eq. 23, I find that the combined thin lens focal length to be

$$f_{\theta 4} = (36 \pm 1) \text{ m} \tag{25}$$

In fact, the red D quad (Q4) error is 10% higher in strength and so the "correct" answer should be

$$\Delta f = 1/(0.1 \times k l_q) = 37 \text{ m} \tag{26}$$

which is the same (barely) as $f_{\theta 4}$. However, the above formula is really only valid for the case when $\Delta f \gg l_q$ which I'm not sure whether I should use this formula here or not. I need to revisit this ...

More interesting is to see how the β function as found by the Kalman filter matches the lattice with the stronger red D quad (Q4). See Fig. 8. From this figure, the match between the actual lattice and the Kalman lattice is within 0.5%.

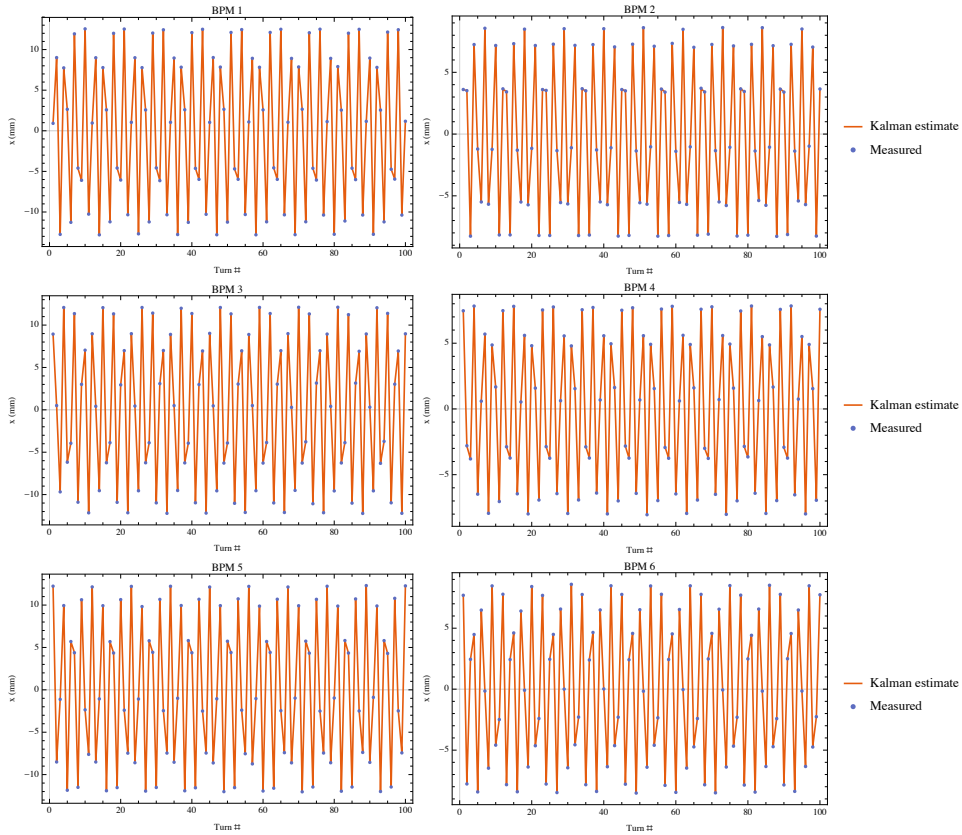

FIG. 5. The Kalman filter found very good parameters to fit to the TBT data.

V. CONCLUSION

From my toy FODO model, I have demonstrated that the Kalman filter does reconstruct the lattice from TBT data. IMO, there are three major advantages:

- The ability to incorporate errors of the measurement and tolerances into the algorithm.
- The ability to identify the source of the lattice error and to calculate its size.
- Visualization of the convergence of the parametric errors.

I think to really make sure that the Kalman filter works for lattice reconstruction from TBT data, I have to include finite chromaticity. Including chromaticity means that sextupoles will need to be included in the model. Furthermore, chromaticity will damp out the TBT data which may cause problems with the analysis. When chromaticity is included, I cannot ignore δ like I had in the toy FODO model. To make it usable in real life, skew quads, BPM calibration errors etc. will also have to be included as well. This is why I

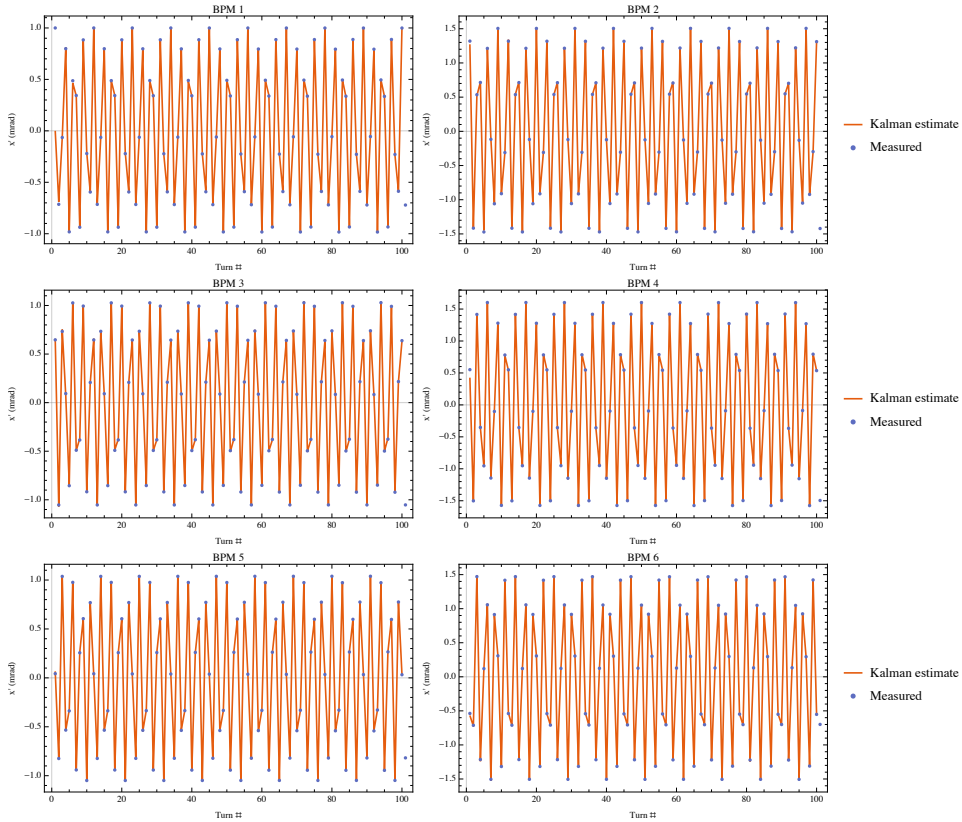

FIG. 6. This compares the Kalman filter results to the model angle at each BPM. I know what the angle should be because I have the exact lattice with the quad error in it.

think the DKF may be better suited for this application rather than JKF because a smaller matrix will need to be inverted.

---

[1] R.E. Kalman. A new approach to linear filtering and prediction problems. *J. Basic Eng.*, 82(D):35–45, 1960.

[2] P. Avery. Applied fitting theory V track fitting using the kalman filter. Technical Report CBX92-39, Cornell University/LNS, 1992.

[3] G.T. Aldrich and W.B. Krabill. An application of kalman techniques to aircraft and missile radar tracking. *AIAA Journal*, 11(7):932–934, 1973.

[4] Z. Ji and M. Brown. Joint state and parameter estimation for biochemical dynamic pathways with iterative extended kalman filter: Comparison with dual state and parameter estimation. *Open Automation and Control Systems Jornal*, 2:69–77, 2009.

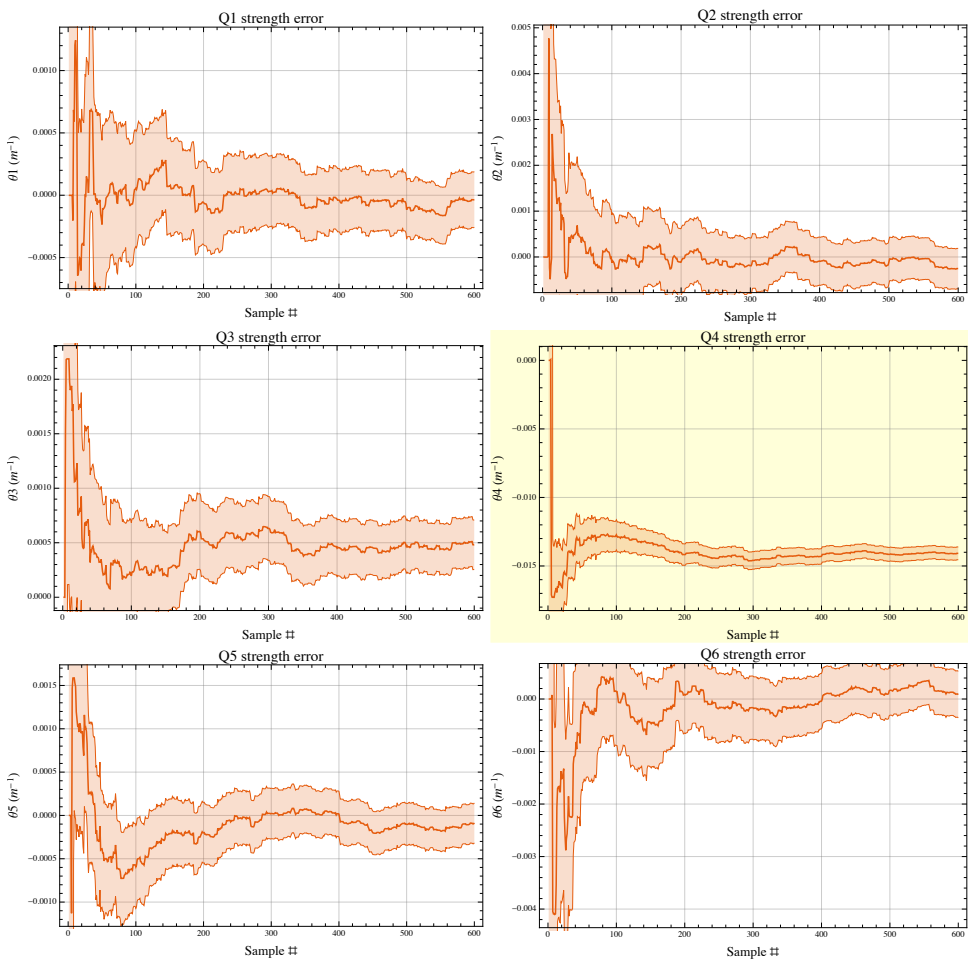

FIG. 7. Here's the quad strength errors found from the fit. There is *some* small value close to zero for all the quads except Q4, which is the red D quad (Q4). The plot that has θ_4 is highlighted in yellow. The $\pm 1\sigma$ error found by the Kalman filter is also drawn here. Note: instead of "Turn #" on the abscissa, I have "Sample #". This is because for each turn, I have 6 BPM readings, i.e 6 samples/turn. Hence, I use sample # here rather than turn #.

[5] G.L. Plett. Lecture notes and recordings for ece5550: Applied kalman filtering. `http://mocha-java.uccs.edu/ECE5550/`, 2014.

[6] G.L. Plett. Dual and joint EKF for simultaneous SOC and SOH estimation. In *CD-ROM Proceedings of the 21st Electric Vehicle Symposium (EVS21)*, 2005.

[7] V. Yadav. Kalman filter. `https://mec560sbu.github.io/2016/10/29/KalmanFilter/`, 2016.

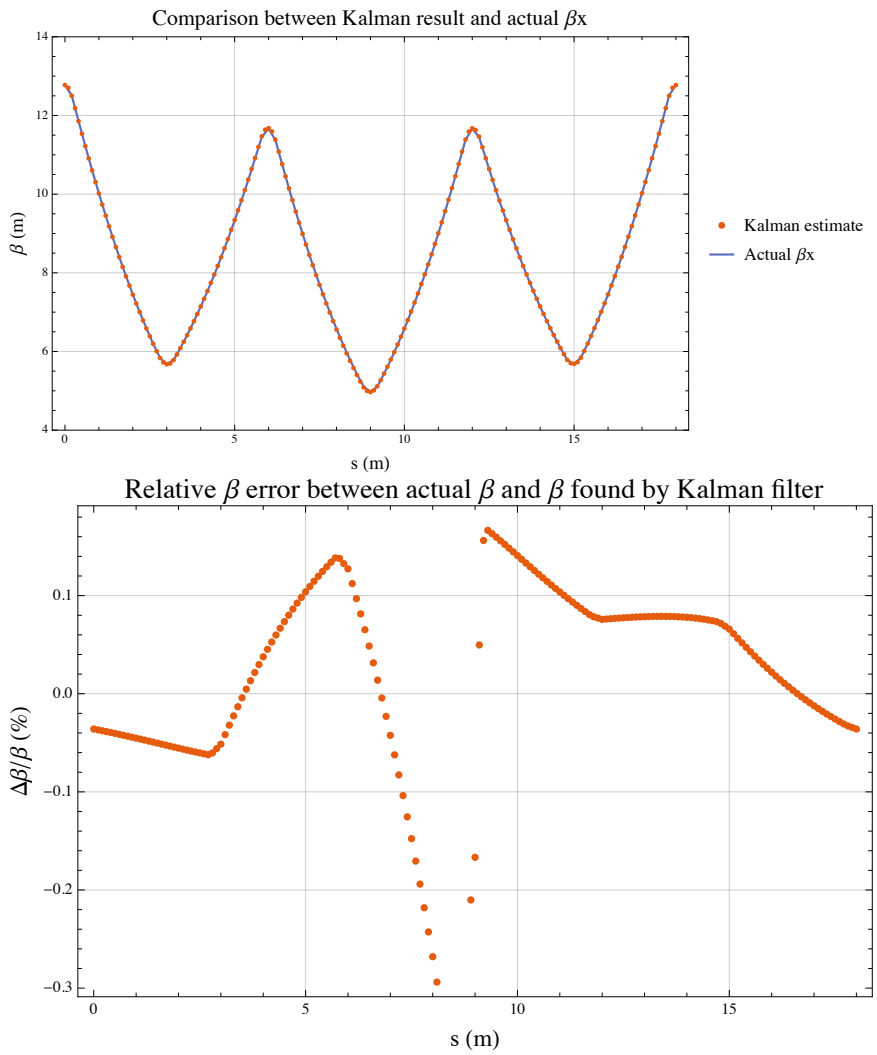

FIG. 8. Here, I compare the actual lattice with the red D quad (Q4) error to what the Kalman filter found. The result is nearly perfect! The relative error is between the two is $< 0.5\%$.