Kalman analysis of TBT data

Data

Data comes from fodo_error.nb

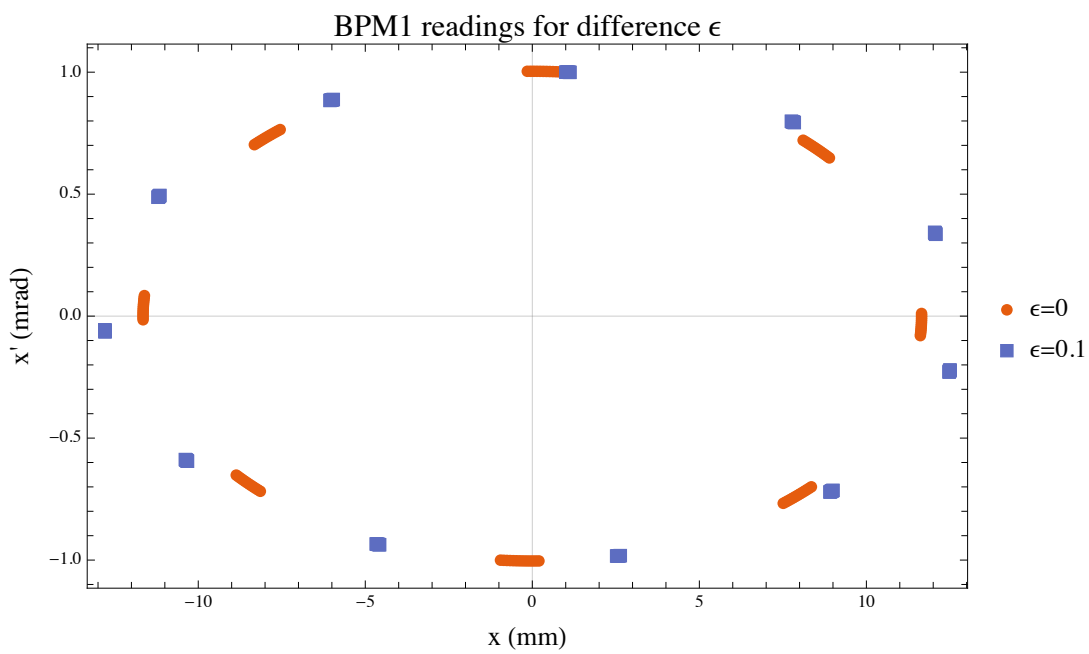
In[126]:=
```
dir = "~/expt/booster/lattice/math/";
```

BPM1

In[127]:=
```
BPM1ideal = Import[dir <> "bpm1_ideal.dat"];
BPM1err = Import[dir <> "bpm1_err.dat"];
```

In[129]:=
```
ListPlot[{BPM1ideal, BPM1err}, PlotMarkers → Automatic, PlotTheme → "Scientific",
 FrameLabel → {Style["x (mm)", 14], Style["x' (mrad)", 14]},
 PlotLabel → Style["BPM1 readings for difference ϵ", 16],
 PlotLegends → {"ϵ=0", "ϵ=0.1"}, ImageSize → 500]
```

Out[129]=

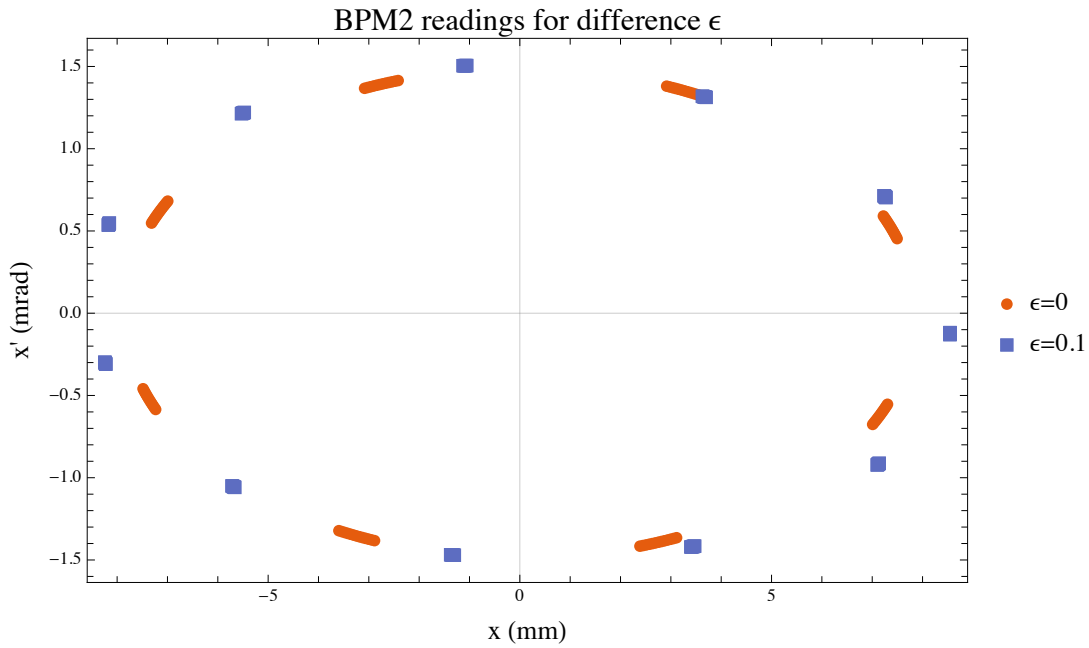


BPM2

In[130]:=
```
BPM2ideal = Import[dir <> "bpm2_ideal.dat"];
BPM2err = Import[dir <> "bpm2_err.dat"];
```

In[132]:=
```
ListPlot[{BPM2ideal, BPM2err}, PlotMarkers → Automatic, PlotTheme → "Scientific",
 FrameLabel → {Style["x (mm)", 14], Style["x' (mrad)", 14]},
 PlotLabel → Style["BPM2 readings for difference ϵ", 16],
 PlotLegends → {"ϵ=0", "ϵ=0.1"}, ImageSize → 500]
```

Out[132]=

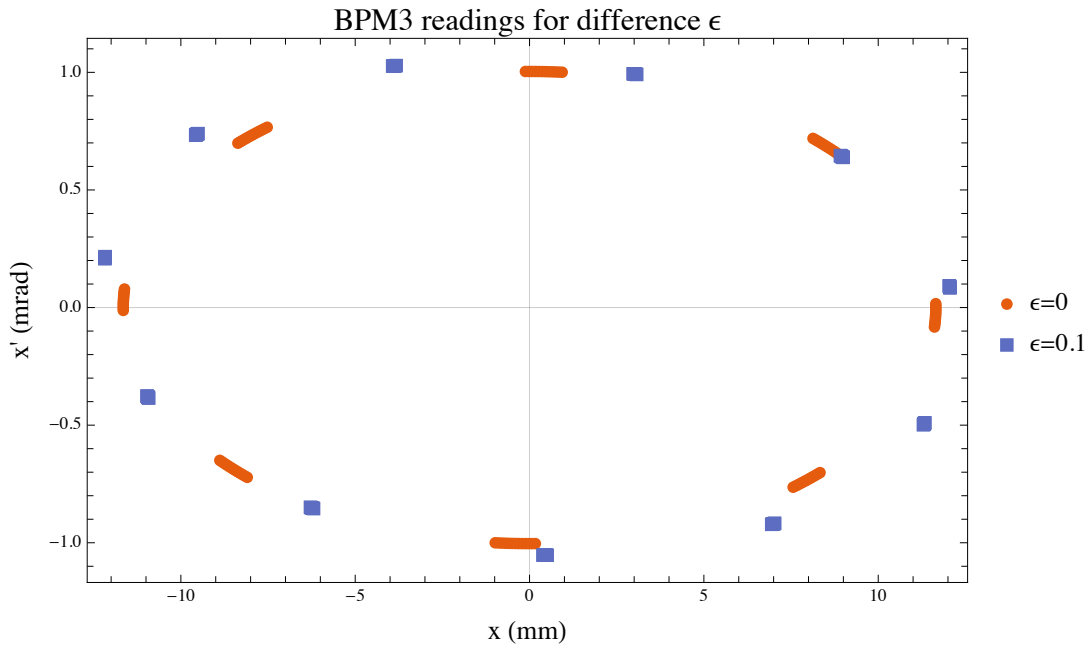


BPM3

In[133]:=
```
BPM3ideal = Import[dir <> "bpm3_ideal.dat"];
BPM3err = Import[dir <> "bpm3_err.dat"];
```

In[135]:=
```
ListPlot[{BPM3ideal, BPM3err}, PlotMarkers → Automatic, PlotTheme → "Scientific",
 FrameLabel → {Style["x (mm)", 14], Style["x' (mrad)", 14]},
 PlotLabel → Style["BPM3 readings for difference ϵ", 16],
 PlotLegends → {"ϵ=0", "ϵ=0.1"}, ImageSize → 500]
```

Out[135]=

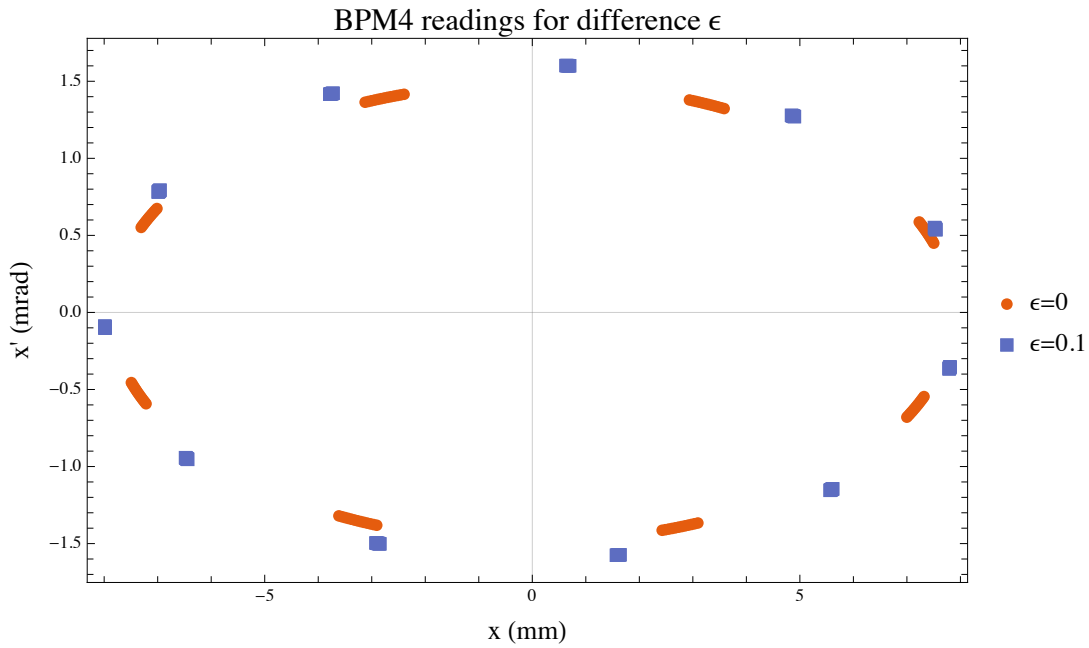


BPM4

In[136]:=
```
BPM4ideal = Import[dir <> "bpm4_ideal.dat"];
BPM4err = Import[dir <> "bpm4_err.dat"];
```

In[138]:=
```
ListPlot[{BPM4ideal, BPM4err}, PlotMarkers → Automatic, PlotTheme → "Scientific",
 FrameLabel → {Style["x (mm)", 14], Style["x' (mrad)", 14]},
 PlotLabel → Style["BPM4 readings for difference ϵ", 16],
 PlotLegends → {"ϵ=0", "ϵ=0.1"}, ImageSize → 500]
```

Out[138]=

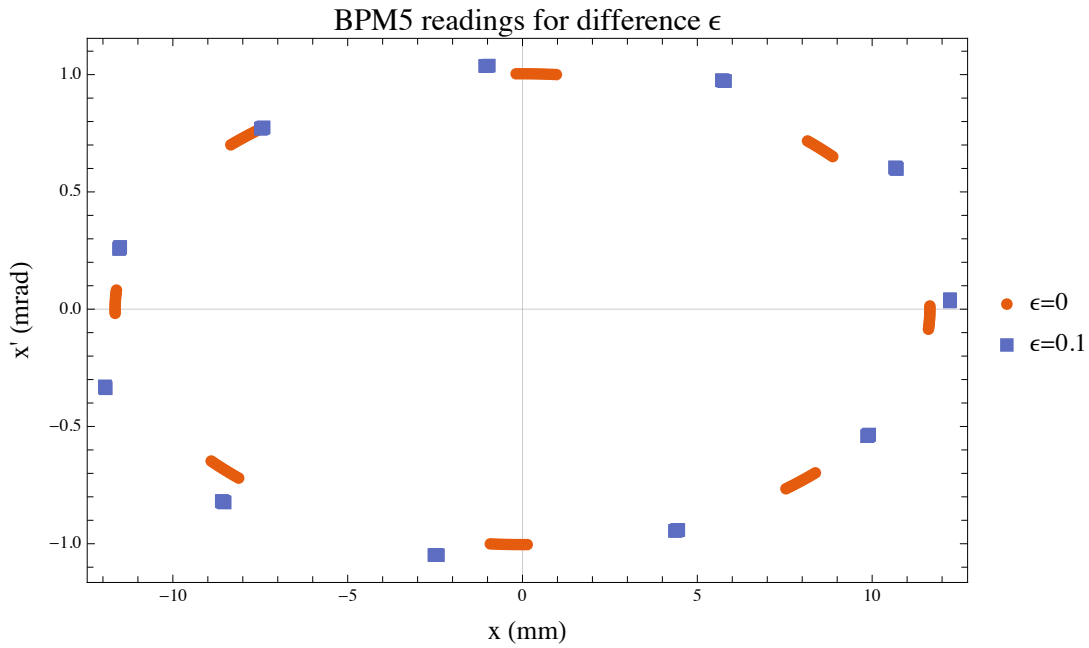


BPM5

In[139]:=
```
BPM5ideal = Import[dir <> "bpm5_ideal.dat"];
BPM5err = Import[dir <> "bpm5_err.dat"];
```

In[141]:=
```
ListPlot[{BPM5ideal, BPM5err}, PlotMarkers → Automatic, PlotTheme → "Scientific",
 FrameLabel → {Style["x (mm)", 14], Style["x' (mrad)", 14]},
 PlotLabel → Style["BPM5 readings for difference ϵ", 16],
 PlotLegends → {"ϵ=0", "ϵ=0.1"}, ImageSize → 500]
```

Out[141]=

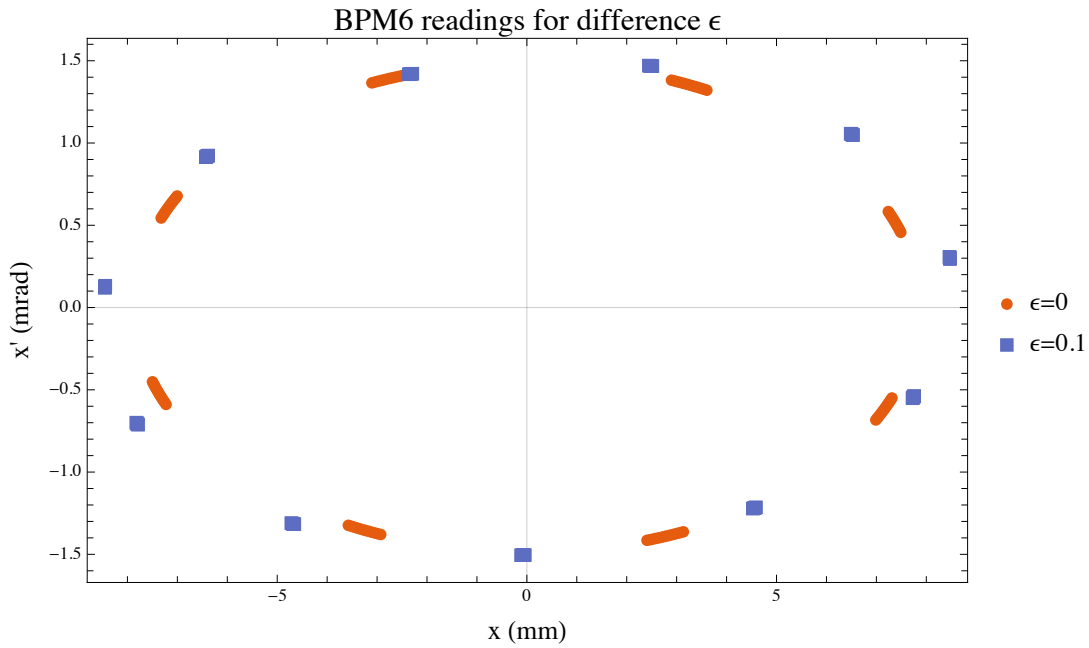


BPM6

In[142]:=
```
BPM6ideal = Import[dir <> "bpm6_ideal.dat"];
BPM6err = Import[dir <> "bpm6_err.dat"];
```

In[144]:=
```
ListPlot[{BPM6ideal, BPM6err}, PlotMarkers → Automatic, PlotTheme → "Scientific",
 FrameLabel → {Style["x (mm)", 14], Style["x' (mrad)", 14]},
 PlotLabel → Style["BPM6 readings for difference ϵ", 16],
 PlotLegends → {"ϵ=0", "ϵ=0.1"}, ImageSize → 500]
```

Out[144]=



Add noise to BPM position data

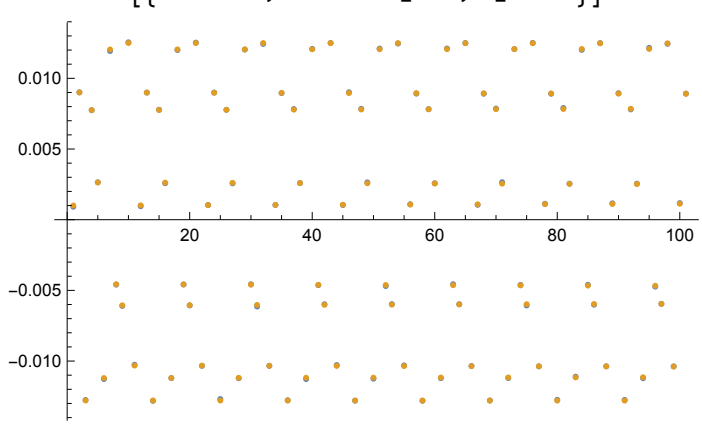
In[145]:=
```
σBPM = 0.05; (*mm*)
```

Convert input data to metres and add noise

In[146]:=
```
x1noise = (#[[1]] + σBPM RandomVariate[NormalDistribution[]] & /@ BPM1err) 10^-3; (*m*)
```

In[147]:=
```
ListPlot[{x1noise, BPM1err[[All, 1]] 10^-3}]
```

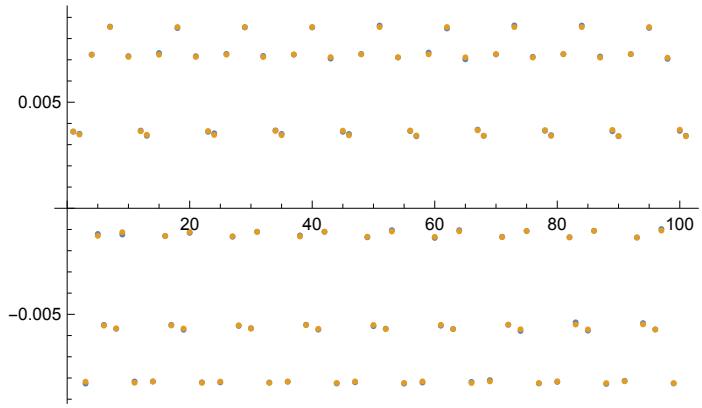
Out[147]=



In[148]:=
```
x2noise = (#[[1]] + σBPM RandomVariate[NormalDistribution[]] & /@ BPM2err) 10^-3; (*m*)
```

In[149]:=
```
ListPlot[{x2noise, BPM2err[[All, 1]] 10^-3 }]
```

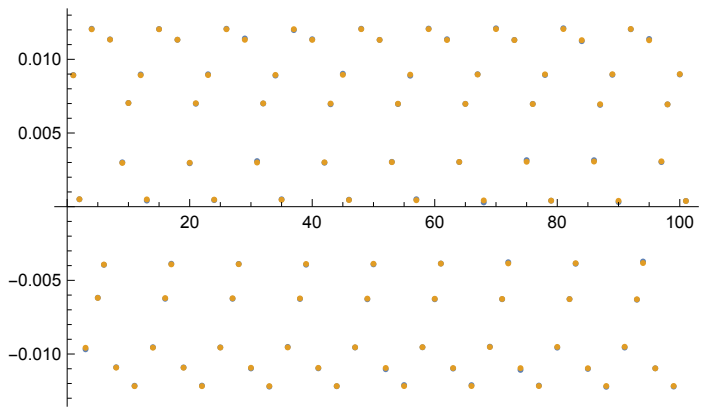
Out[149]=



In[150]:=
```
x3noise = (#[[1]] + σBPM RandomVariate[NormalDistribution[]] & /@ BPM3err) 10^-3; (*m*)
```

In[151]:=
```
ListPlot[{x3noise, BPM3err[[All, 1]] 10^-3 }]
```

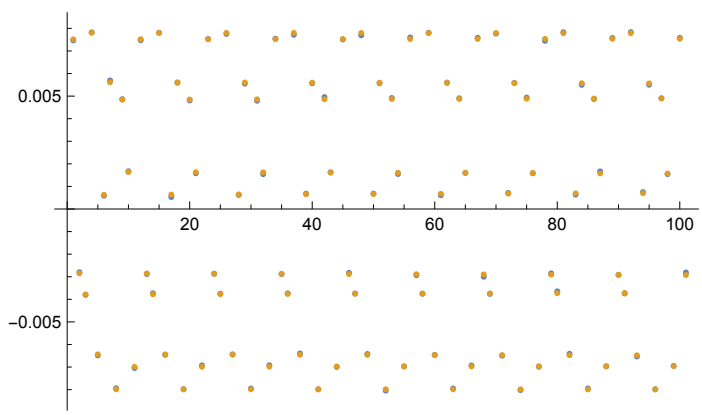
Out[151]=



In[152]:=
```
x4noise = (#[[1]] + σBPM RandomVariate[NormalDistribution[]] & /@ BPM4err) 10^-3; (*m*)
```

In[153]:=
```
ListPlot[{x4noise, BPM4err[[All, 1]] 10^-3 }]
```

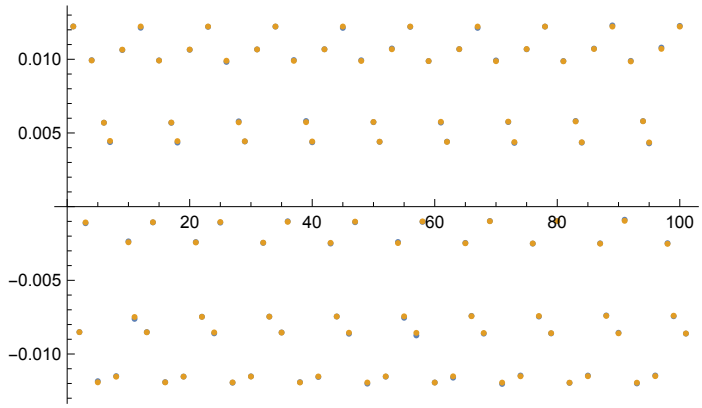
Out[153]=



In[154]:=
```
x5noise = (#[[1]] + σBPM RandomVariate[NormalDistribution[]] & /@ BPM5err) 10^-3; (*m*)
```

In[155]:=
```
ListPlot[{x5noise, BPM5err[[All, 1]] 10^-3 }]
```

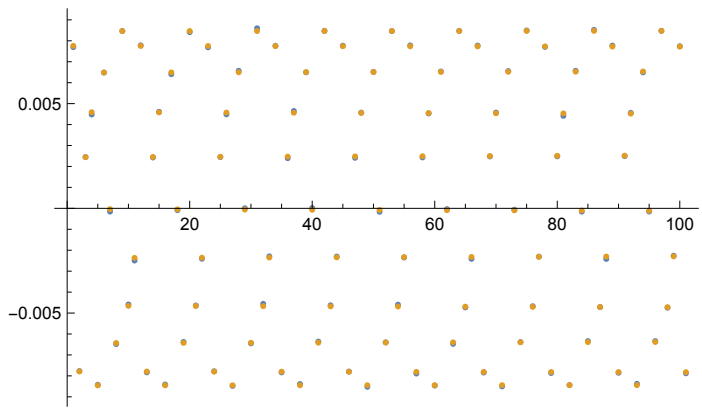
Out[155]=



In[156]:=
```
x6noise = (#[[1]] + σBPM RandomVariate[NormalDistribution[]] & /@ BPM6err) 10^-3; (*m*)
```

In[157]:=
```
ListPlot[{x6noise, BPM6err[[All, 1]] 10^-3 }]
```

Out[157]=



Transport matrices

In[158]:=
```
MQF[K_, l_] := ( Cos[Sqrt[K] l]                  1/Sqrt[K] Sin[Sqrt[K] l]
                 -Sqrt[K] Sin[Sqrt[K] l]         Cos[Sqrt[K] l] );
MQD[K_, l_] := ( Cosh[Sqrt[K] l]                 1/Sqrt[K] Sinh[Sqrt[K] l]
                 Sqrt[K] Sinh[Sqrt[K] l]         Cosh[Sqrt[K] l] );
Mdrift[l_] := ( 1  l
                0  1 );
```

Quad error.

$\theta = 1/f$

In[161]:=
```
MQerr[θ_] := ( 1   0
               -θ  1 );
```

Add in the error

In[162]:=
```
MQFerr[K_, l_, θ_] := MQF[K, l].MQerr[θ];
MQDerr[K_, l_, θ_] := MQD[K, l].MQerr[θ];
```

Lattice

Let the number of FODO elements be three, i.e. I have

FODO|FODO| FODO
#1 | #2 | #3

On each quad, I will also have a quad error matrix whose value the Kalman filter will try to find

In[164]:=
```
Melement = {

  {MQFerr, k, lq/2, θ1}, {Mdrift, ld}, {MQDerr, k, lq/2, θ2},
  {MQDerr, k, lq/2, θ2}, {Mdrift, ld}, {MQFerr, k, lq/2, θ3},


  {MQFerr, k, lq/2, θ3}, {Mdrift, ld}, {MQDerr, k, lq/2, θ4},
  {MQDerr, k, lq/2, θ4}, {Mdrift, ld}, {MQFerr, k, lq/2, θ5},

  {MQFerr, k, lq/2, θ5}, {Mdrift, ld}, {MQDerr, k, lq/2, θ6},
  {MQDerr, k, lq/2, θ6}, {Mdrift, ld}, {MQFerr, k, lq/2, θ1}
 }
```

Out[164]=
```
{{MQFerr, 0.54102, 0.25, θ1}, {Mdrift, 2.5}, {MQDerr, 0.54102, 0.25, θ2},
 {MQDerr, 0.54102, 0.25, θ2}, {Mdrift, 2.5}, {MQFerr, 0.54102, 0.25, θ3},
 {MQFerr, 0.54102, 0.25, θ3}, {Mdrift, 2.5}, {MQDerr, 0.54102, 0.25, θ4},
 {MQDerr, 0.54102, 0.25, θ4}, {Mdrift, 2.5}, {MQFerr, 0.54102, 0.25, θ5},
 {MQFerr, 0.54102, 0.25, θ5}, {Mdrift, 2.5}, {MQDerr, 0.54102, 0.25, θ6},
 {MQDerr, 0.54102, 0.25, θ6}, {Mdrift, 2.5}, {MQFerr, 0.54102, 0.25, θ1}}
```

A matrices

From BPM1 to BPM2

In[165]:=
```
FF1to2 = MQDerr[k, lq/2, θ2].Mdrift[ld].MQFerr[k, lq/2, θ1];
```

In[166]:=
```
F1to2 = FF1to2.(x1
                x2) // Simplify;
```

In[167]:=
```
F11 = F1to2[[1, 1]];
F21 = F1to2[[2, 1]];
```

In[169]:=
```
AA1 =
 ( D[F11, x1]  D[F11, x2]  D[F11, θ1]  D[F11, θ2]  D[F11, θ3]  D[F11, θ4]  D[F11, θ5]  D[F1
   D[F21, x1]  D[F21, x2]  D[F21, θ1]  D[F21, θ2]  D[F21, θ3]  D[F21, θ4]  D[F21, θ5]  D[F2
       0           0           1           0           0           0           0
       0           0           0           1           0           0           0
       0           0           0           0           1           0           0
       0           0           0           0           0           1           0
       0           0           0           0           0           0           1
       0           0           0           0           0           0           0

   )

   // Simplify;
```

From BPM2 to BPM3

In[170]:=
```
FF2to3 = MQFerr[k, lq/2, θ3].Mdrift[ld].MQDerr[k, lq/2, θ2];
```

In[171]:=
```
F2to3 = FF2to3.( x1 ) // Simplify;
                ( x2 )
```

In[172]:=
```
F11 = F2to3[[1, 1]];
F21 = F2to3[[2, 1]];
```

In[174]:=
```
AA2 =
 ( D[F11, x1]  D[F11, x2]  D[F11, θ1]  D[F11, θ2]  D[F11, θ3]  D[F11, θ4]  D[F11, θ5]  D[F1
   D[F21, x1]  D[F21, x2]  D[F21, θ1]  D[F21, θ2]  D[F21, θ3]  D[F21, θ4]  D[F21, θ5]  D[F2
       0           0           1           0           0           0           0
       0           0           0           1           0           0           0
       0           0           0           0           1           0           0
       0           0           0           0           0           1           0
       0           0           0           0           0           0           1
       0           0           0           0           0           0           0

   )

   // Simplify;
```

From BPM3 to BPM4

In[175]:=
```
FF3to4 = MQDerr[k, lq/2, θ4].Mdrift[ld].MQFerr[k, lq/2, θ3];
```

In[176]:=
```
F3to4 = FF3to4.( x1 ) // Simplify;
                ( x2 )
```

In[177]:=
```
F11 = F3to4[[1, 1]];
F21 = F3to4[[2, 1]];
```

In[179]:=
```
AA3 =
 ( D[F11, x1]  D[F11, x2]  D[F11, θ1]  D[F11, θ2]  D[F11, θ3]  D[F11, θ4]  D[F11, θ5]  D[F1
   D[F21, x1]  D[F21, x2]  D[F21, θ1]  D[F21, θ2]  D[F21, θ3]  D[F21, θ4]  D[F21, θ5]  D[F2
       0           0           1           0           0           0           0
       0           0           0           1           0           0           0
       0           0           0           0           1           0           0
       0           0           0           0           0           1           0
       0           0           0           0           0           0           1
       0           0           0           0           0           0           0

   )

   // Simplify;
```

From BPM4 to BPM5

In[180]:=
```
FF4to5 = MQFerr[k, lq/2, θ5].Mdrift[ld].MQDerr[k, lq/2, θ4];
```

In[181]:=
```
F4to5 = FF4to5.( x1 ) // Simplify;
                ( x2 )
```

In[182]:=
```
F11 = F4to5[[1, 1]];
F21 = F4to5[[2, 1]];
```

In[184]:=
```
AA4 =
 ( D[F11, x1]  D[F11, x2]  D[F11, θ1]  D[F11, θ2]  D[F11, θ3]  D[F11, θ4]  D[F11, θ5]  D[F1
   D[F21, x1]  D[F21, x2]  D[F21, θ1]  D[F21, θ2]  D[F21, θ3]  D[F21, θ4]  D[F21, θ5]  D[F2
       0           0           1           0           0           0           0
       0           0           0           1           0           0           0
       0           0           0           0           1           0           0
       0           0           0           0           0           1           0
       0           0           0           0           0           0           1
       0           0           0           0           0           0           0

   )

   // Simplify;
```

From BPM5 to BPM6

In[185]:=
```
FF5to6 = MQDerr[k, lq/2, θ6].Mdrift[ld].MQFerr[k, lq/2, θ5];
```

In[186]:=
```
F5to6 = FF5to6.(x1
                x2) // Simplify;
```

In[187]:=
```
F11 = F5to6[[1, 1]];
F21 = F5to6[[2, 1]];
```

In[189]:=
```
AA5 =
 (D[F11, x1]  D[F11, x2]  D[F11, θ1]  D[F11, θ2]  D[F11, θ3]  D[F11, θ4]  D[F11, θ5]  D[F1
 D[F21, x1]  D[F21, x2]  D[F21, θ1]  D[F21, θ2]  D[F21, θ3]  D[F21, θ4]  D[F21, θ5]  D[F2
      0           0           1           0           0           0           0
      0           0           0           1           0           0           0
      0           0           0           0           1           0           0
      0           0           0           0           0           1           0
      0           0           0           0           0           0           1
      0           0           0           0           0           0           0

   )

   // Simplify;
```

From BPM6 to BPM1

In[190]:=
```
FF6to1 = MQFerr[k, lq/2, θ1].Mdrift[ld].MQDerr[k, lq/2, θ6];
```

In[191]:=
```
F6to1 = FF6to1.(x1
                x2) // Simplify;
```

In[192]:=
```
F11 = F6to1[[1, 1]];
F21 = F6to1[[2, 1]];
```

In[194]:=
```
AA6 =
 (D[F11, x1]  D[F11, x2]  D[F11, θ1]  D[F11, θ2]  D[F11, θ3]  D[F11, θ4]  D[F11, θ5]  D[F1
 D[F21, x1]  D[F21, x2]  D[F21, θ1]  D[F21, θ2]  D[F21, θ3]  D[F21, θ4]  D[F21, θ5]  D[F2
      0           0           1           0           0           0           0
      0           0           0           1           0           0           0
      0           0           0           0           1           0           0
      0           0           0           0           0           1           0
      0           0           0           0           0           0           1
      0           0           0           0           0           0           0

   )

   // Simplify;
```

In[195]:=
```
AA[k_, x_] := Module[{px1, px2, pθ1, pθ2, pθ3, pθ4, pθ5, pθ6, px, AAn},
   AAn = {AA1, AA2, AA3, AA4, AA5, AA6};
   {px1, px2, pθ1, pθ2, pθ3, pθ4, pθ5, pθ6} = Flatten[x];
   Return[AAn[[k]] /. {x1 → px1, x2 → px2,
      θ1 → pθ1, θ2 → pθ2, θ3 → pθ3, θ4 → pθ4, θ5 → pθ5, θ6 → pθ6}];
  ];
```

State and parameter matrix

Assume that the state vector is

$$xs = \begin{pmatrix} x1 \\ x2 \\ \theta 1 \\ \theta 2 \\ \theta 3 \\ \theta 4 \\ \theta 5 \\ \theta 6 \end{pmatrix}$$

x1: position

x2: x’

θn: the error in the strength of the quad

H, h and FF matrices

Predicted values

In[196]:=
```
h[xpred_] := {xpred[[1, 1]]}; (* return the position*)
```

In[197]:=
```
H[xpred_] := Return[( 1 0 0 0 0 0 0 0 )]; (* project out position*)
```

Set up the transport matrices that take the beam from one bpm to the next

In[198]:=
```
FFtrans = {FF1to2, FF2to3, FF3to4, FF4to5, FF5to6, FF6to1};
```

In[199]:=
```
FF[ k_, xprev_] := Module[{px1, px2, pθ1, pθ2, pθ3, pθ4, pθ5, pθ6, px, M},
   {px1, px2, pθ1, pθ2, pθ3, pθ4, pθ5, pθ6} = Flatten[xprev];
   M = FFtrans[[k]];
   px = M.(px1, px2) /. { θ1 → pθ1, θ2 → pθ2, θ3 → pθ3, θ4 → pθ4, θ5 → pθ5, θ6 → pθ6} ;
   Return[{{px[[1, 1]], px[[2, 1]], pθ1, pθ2, pθ3, pθ4, pθ5, pθ6}}^T];
  ];
```

Kalman filter

Covariance errors

In[200]:=
```
QQx1 = (1 0 0 0 0 0 0 0); QQx2 = (0 1 0 0 0 0 0 0);
```

In[201]:=
```
QQθ1 = (0 0 1 0 0 0 0 0); QQθ2 = (0 0 0 1 0 0 0 0); QQθ3 = (0 0 0 0 1 0 0 0); QQθ4 = (0 0 0 0 0 1 0 0); QQθ5 = (0 0 0 0 0 0 1 0); QQθ6 = (0 0 0 0 0 0 0 1);
```

In[202]:=
```
jx = (0.01 × 10^-3)^2; (*variance of x and x', guess*)
```

In[203]:=
```
jθ = (0.01 × 10^-3)^2; (*variance of θn, guess*)
```

In[204]:=
```
QQ = jx (QQx1.QQx1^T + QQx2.QQx2^T) +
    jθ (QQθ1.QQθ1^T + QQθ2.QQθ2^T + QQθ3.QQθ3^T + QQθ4.QQθ4^T + QQθ5.QQθ5^T + QQθ6.QQθ6^T);
```

BPM measurement error

In[205]:=
```
RR = {{(σBPM 10^-3)^2}}
```

Out[205]= $\{\{2.5 \times 10^{-9}\}\}$

Nominal values for the elements

Quad strength and length

In[206]:=
```
k = 0.54102;
lq = 0.5;
```

Length of drift space

In[208]:=
```
ld = 2.5;
```

Initialization

Noisy measurement data, in sequence:

BPM1, BPM2, BPM3, ..., BPM6

I made it so that

observations[[n]] is the set of BPM data at turn n.

observations⟦n⟧⟦k⟧ is the position of the beam at BPM k at turn n

In[209]:=
```
observations = Table[{{x1noise⟦i⟧}, {x2noise⟦i⟧}, {x3noise⟦i⟧},
    {x4noise⟦i⟧}, {x5noise⟦i⟧}, {x6noise⟦i⟧}}, {i, Length[x1noise]}];
```

In[210]:=
```
nx = Length[QQ];
ny = Length[RR];
Nturns = Length[observations];
Nbpms = 6;
```

Identity matrix

In[214]:=
```
Inx = IdentityMatrix[nx];
```

Allocate results

In[215]:=
```
xpred = Table[ ConstantArray[0, {Nbpms, nx}], Nturns];
(*prediction of state vector for each bpm*)
Ppred = Table[ConstantArray[0, {Nbpms, nx, nx}], Nturns];
(*prediction error covariance matrix for each bpm*)
xest = Table[ConstantArray[0, {Nbpms, nx}], Nturns];
(*estimation of state vector for each bpm*)
Pest = Table[ConstantArray[0, {Nbpms, nx, nx}], Nturns];
(*estimation error covariance matrix for each bpm*)
KK = Table[ConstantArray[0, {Nbpms, nx, ny}], Nturns];
(*Kalman gain for each bpm*)
```

starting state vector for BPM1. I only know the position, we everything else is zero

In[220]:=
```
xinit = ( observations⟦1⟧⟦1⟧⟦1⟧ )
        (          0            )
        (          0            )
        (          0            )
        (          0            )
        (          0            )
        (          0            )
        (          0            ) ;
```

In[221]:=
```
Pinit = IdentityMatrix[Length[xinit]];
```

Set initial prediction

In[222]:=
```
xpred⟦1⟧⟦1⟧ = xinit;
Ppred⟦1⟧⟦1⟧ = Pinit;

xest⟦1⟧⟦1⟧ = xinit;
Pest⟦1⟧⟦1⟧ = Pinit;
PPest = Pinit;
```

Iterate

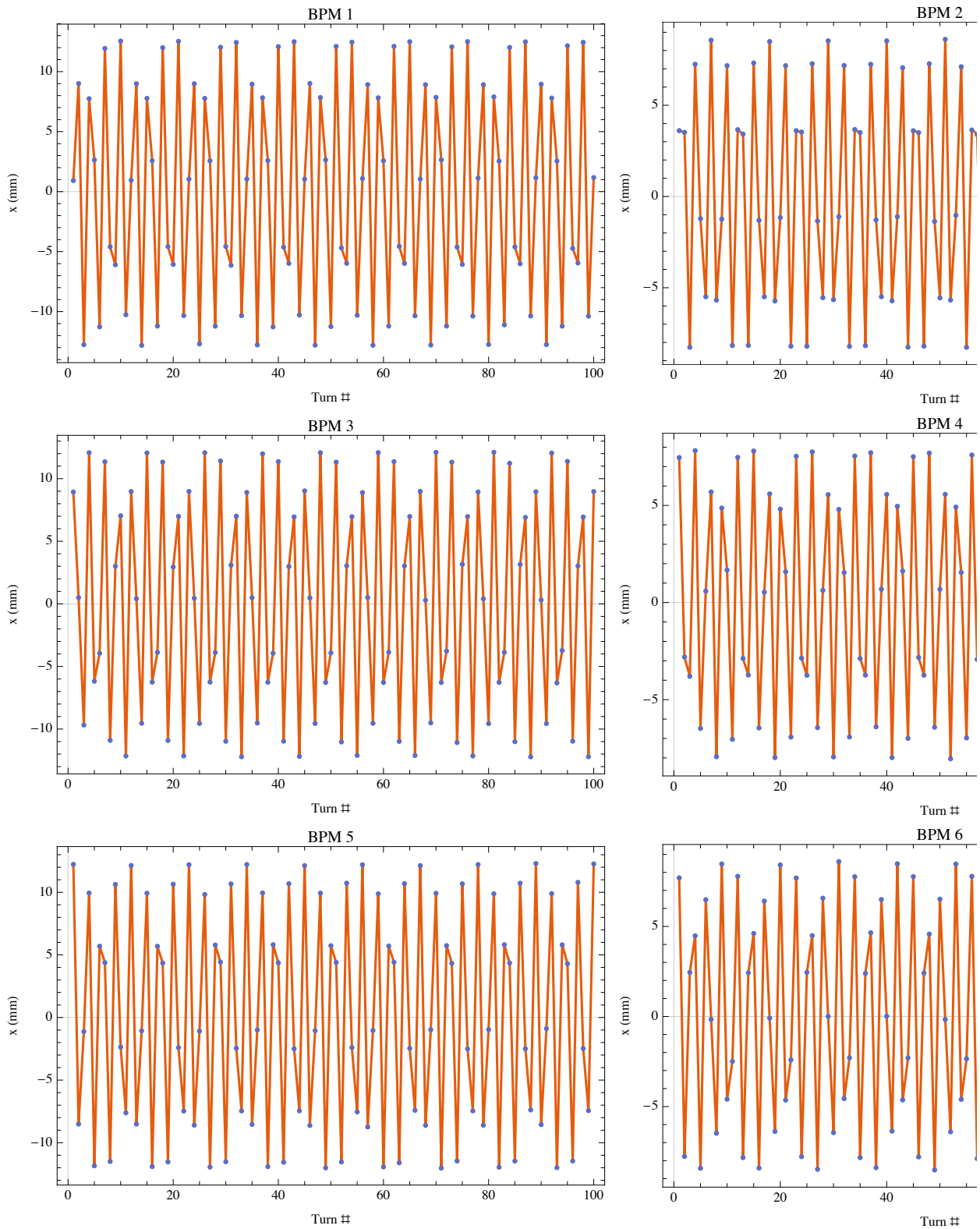
In[227]:=
```
For[n = 1, n ≤ Nturns - 1, n++,
  (*Print["n=", n];*)
  For[i = 1, i ≤ Nbpms, i++,
    (*prepare to wrap around if we are already at the last BPM *)
    If[i != Nbpms,
     j = i + 1; m = n,
     j = 1; m = n + 1
    ];

    (*prediction from BPM i to BPM i+1, at turn n*)
    xpred[[m]][[j]] = FF[i, xest[[n]][[i]]];
    Ppred[[m]][[j]] = AA[i, xest[[n]][[i]]].PPest.AA[i, xest[[n]][[i]]]ᵀ + QQ;

    (*estimate*)
    yobs = observations[[m]][[j]];
    ypred = h[xpred[[m]][[j]]];
    Hpred = H[xpred[[m]][[j]]];
    KK[[m]][[j]] = Ppred[[m]][[j]].Hpredᵀ.Inverse[(Hpred.Ppred[[m]][[j]].Hpredᵀ) + RR];
    xest[[m]][[j]] = xpred[[m]][[j]] + KK[[m]][[j]].(yobs - ypred);
    PPest = Pest[[m]][[j]] = (Inx - KK[[m]][[j]].Hpred).Ppred[[m]][[j]];

   ];
  (*xest[[1,3]]=xest[[-1]][[3]];*)
 ];
```

Plots

Fit to position

In[265]:=
```
gg1 =
 Grid[Table[{ListPlot[{Table[xest[[n]][[j]][[1, 1]] 10^3, {n, Nturns - 1}], Flatten[Table[
        observations[[n]][[j]] 10^3, {n, Nturns - 1}]]}, Joined → {True, False},
     PlotTheme → "Scientific", FrameLabel → {"Turn #", "x (mm)"},
     PlotLabel → "BPM " <> ToString[j], ImageSize → 400],
    ListPlot[{Table[xest[[n]][[j + 1]][[1, 1]] 10^3, {n, Nturns - 1}],
      Flatten[Table[observations[[n]][[j + 1]] 10^3, {n, Nturns - 1}]]},
     Joined → {True, False}, PlotTheme → "Scientific",
     FrameLabel → {"Turn #", "x (mm)"}, PlotLabel → "BPM " <> ToString[j + 1],
     PlotLegends → {"Kalman estimate", "Measured"},
     ImageSize → 400]}, {j, 1, Nbpms, 2}]]
```

Out[265]=



In[268]:=
```
Export[dir <> "kalmanfitPos.pdf", gg1]
```

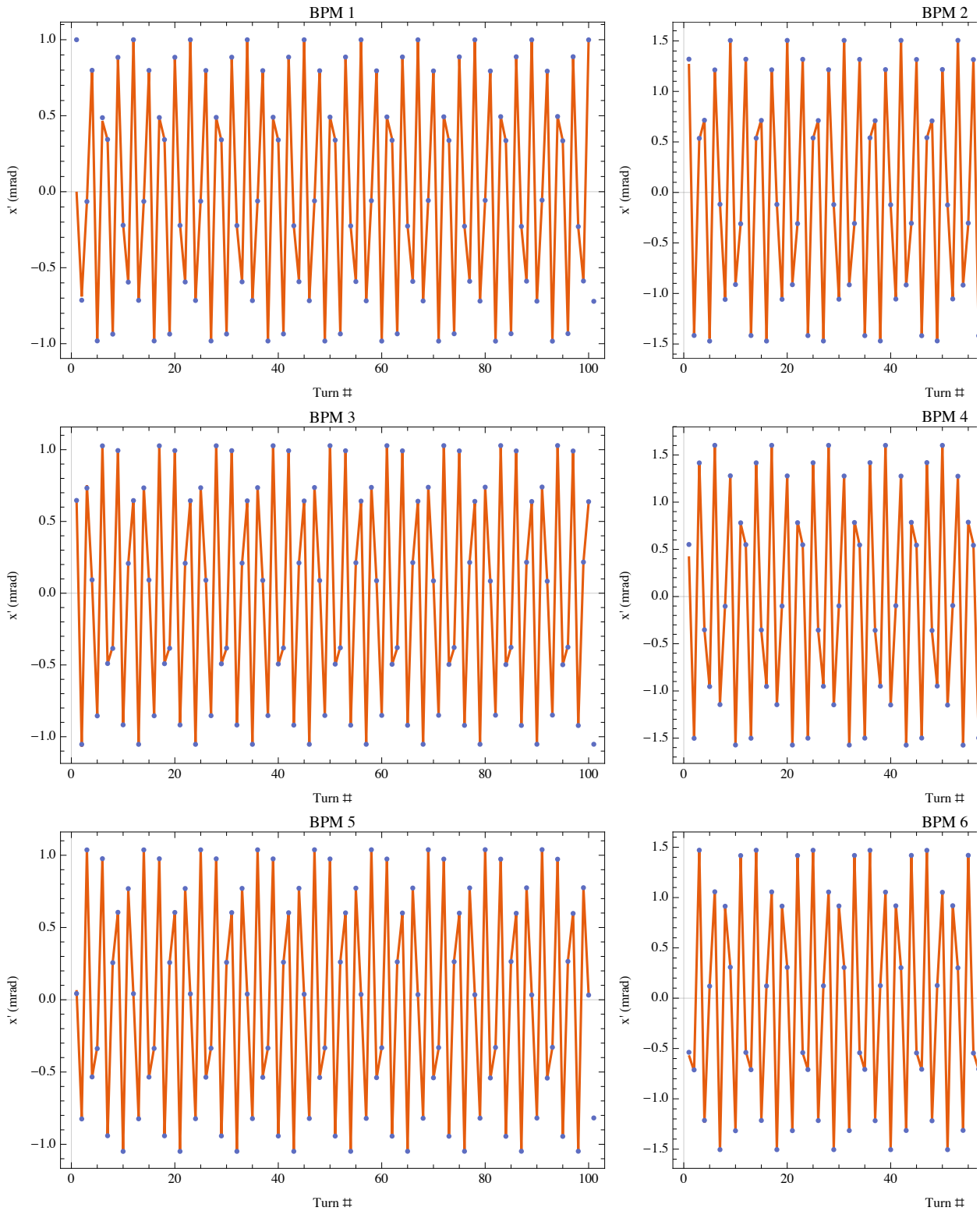
Out[268]= ~/expt/booster/lattice/math/kalmanfitPos.pdf

Fit to angle

In[229]:=
```
BPMerrList = {BPM1err, BPM2err, BPM3err, BPM4err, BPM5err, BPM6err};
```

In[267]:=
```
gg2 = Grid[Table[
    {ListPlot[{Table[xest[[n]][[j]][[2, 1]] 10^3, {n, Nturns - 1}], BPMerrList[[j]][[All, 2]] },
      Joined → {True, False}, PlotTheme → "Scientific",
      FrameLabel → {"Turn #", "x' (mrad)"}, PlotLabel → "BPM " <> ToString[j],
      ImageSize → 400], ListPlot[{Table[xest[[n]][[j + 1]][[2, 1]] 10^3, {n, Nturns - 1}],
       BPMerrList[[j + 1]][[All, 2]] }, Joined → {True, False}, PlotTheme → "Scientific",
      FrameLabel → {"Turn #", "x' (mrad)"}, PlotLabel → "BPM " <> ToString[j + 1],
      PlotLegends → {"Kalman estimate", "Measured"},
      ImageSize → 400]}, {j, 1, Nbpms, 2}]]
```

Out[267]=



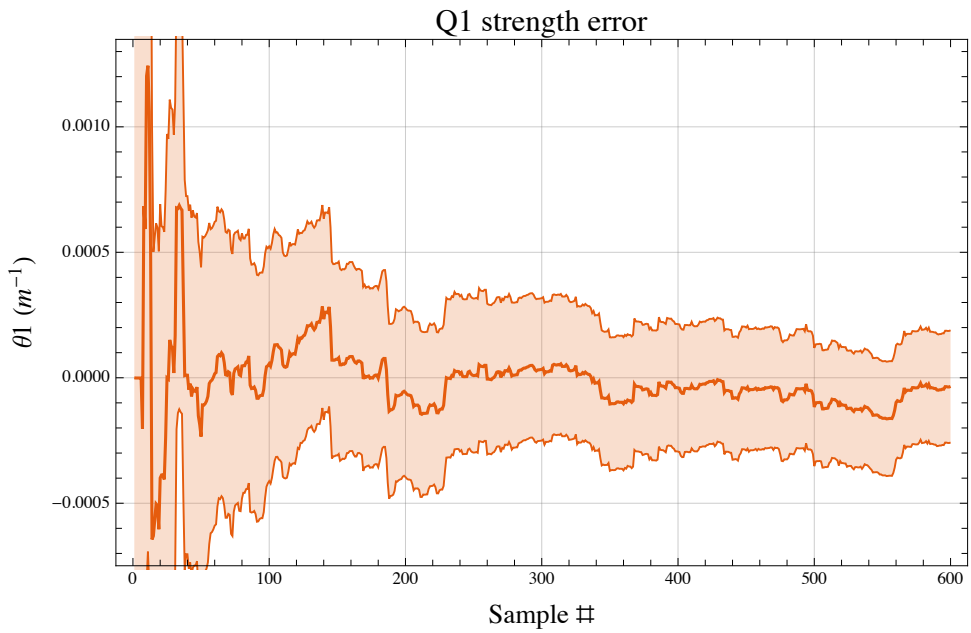
In[269]:=
```
Export[dir <> "kalmanfitAngle.pdf", gg2]
```

Out[269]= ~/expt/booster/lattice/math/kalmanfitAngle.pdf

θ1 (Q1 should have no error)

In[252]:=
```
g1 = ListPlot[Flatten[Table[Around[xest[[n]][[j]][[3, 1]], Sqrt[Ppred[[n]][[j]][[3, 3]]]],
    {n, Nturns - 1}, {j, 1, 6}]], Joined → True, PlotRange → All,
  GridLines → Automatic, IntervalMarkers → "Bands", PlotTheme → "Scientific",
  FrameLabel → {Style["Sample #", 14], Style["θ1 (m^-1)", 14]},
  PlotLabel → Style["Q1 strength error", 16], ImageSize → 500]
```

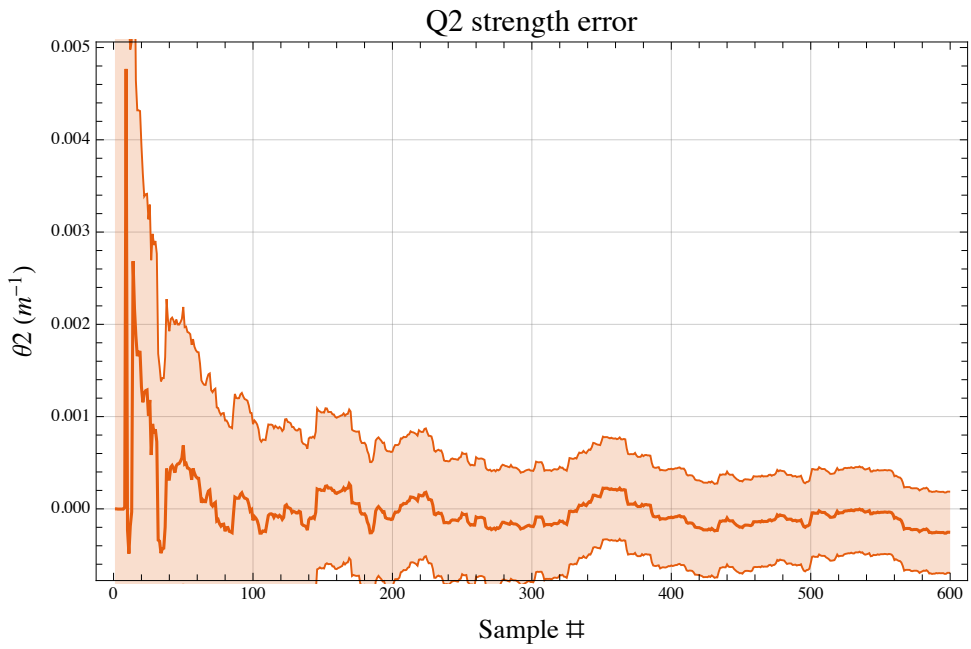
Out[252]=



θ2 (Q2 should have no error)

In[253]:=
```
g2 = ListPlot[Flatten[Table[Around[xest[[n]][[j]][[4, 1]], Sqrt[Ppred[[n]][[j]][[4, 4]]]],
    {n, Nturns - 1}, {j, 1, 6}]], Joined → True, PlotRange → All,
  GridLines → Automatic, IntervalMarkers → "Bands", PlotTheme → "Scientific",
  FrameLabel → {Style["Sample #", 14], Style["θ2 (m^-1)", 14]},
  PlotLabel → Style["Q2 strength error", 16], ImageSize → 500]
```

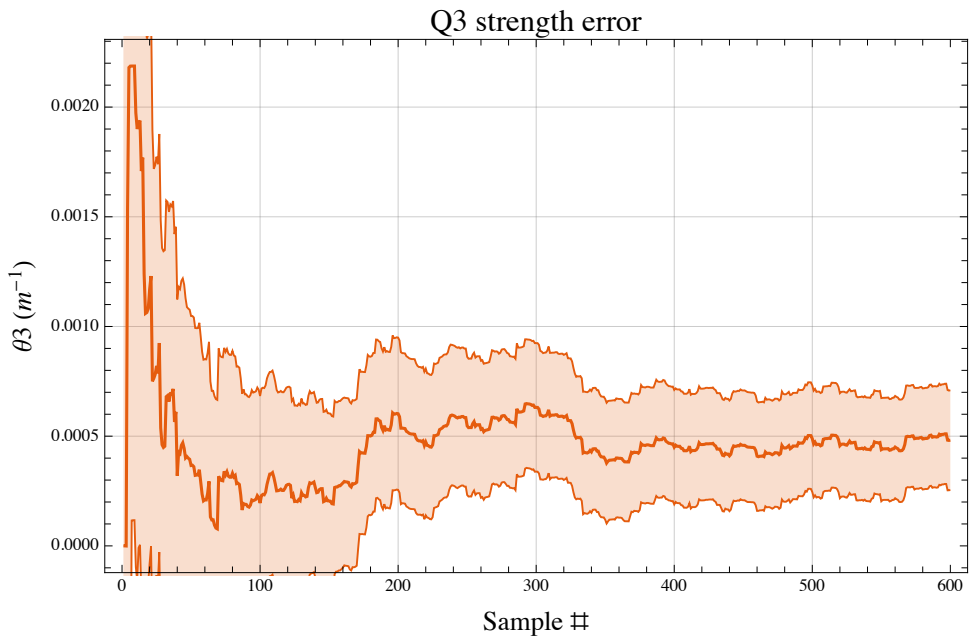
Out[253]=



θ3 (Q3 should have no error)

In[255]:=
```
g3 = ListPlot[Flatten[Table[Around[xest[[n]][[j]][[5, 1]], Sqrt[Ppred[[n]][[j]][[5, 5]]]],
    {n, Nturns - 1}, {j, 1, 6}]], Joined → True, PlotRange → All,
  GridLines → Automatic, IntervalMarkers → "Bands", PlotTheme → "Scientific",
  FrameLabel → {Style["Sample #", 14], Style["θ3 (m^-1)", 14]},
  PlotLabel → Style["Q3 strength error", 16], ImageSize → 500]
```

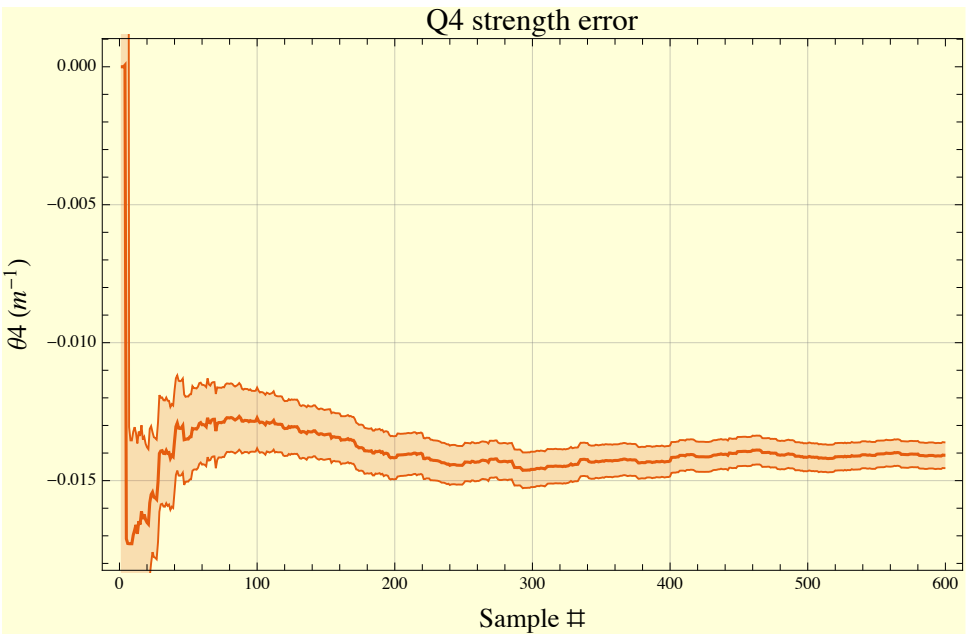
Out[255]=



θ4 (Q4 strength error)

In[260]:=
```
g4 = ListPlot[Flatten[Table[Around[xest[[n]][[j]][[6, 1]], Sqrt[Ppred[[n]][[j]][[6, 6]]]],
    {n, Nturns - 1}, {j, 1, 6}]], Joined → True, PlotRange → All,
  GridLines → Automatic, IntervalMarkers → "Bands", PlotTheme → "Scientific",
  FrameLabel → {Style["Sample #", 14], Style["θ4 (m^-1)", 14]},
  PlotLabel → Style["Q4 strength error", 16], Background → LightYellow, (*Prolog→
   {LightYellow,Rectangle[Scaled[{0,0}],Scaled[{1,1}]]},*)ImageSize → 500]
```

Out[260]=



θ5 (Q5 strength error)

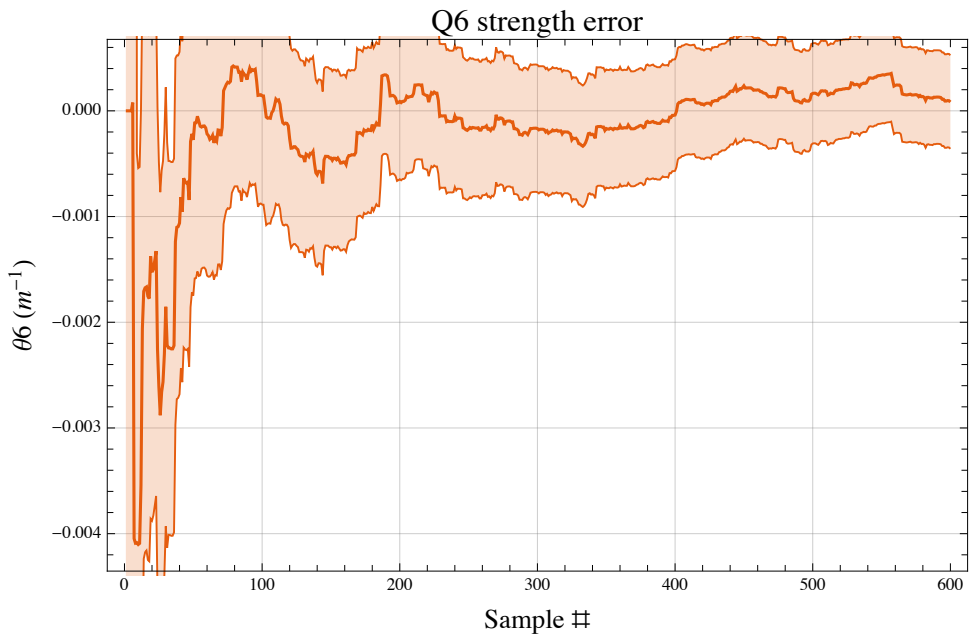
In[262]:=
```
g5 = ListPlot[Flatten[Table[Around[xest[[n]][[j]][[7, 1]], Sqrt[Ppred[[n]][[j]][[7, 7]]]],
    {n, Nturns - 1}, {j, 1, 6}]], Joined → True, PlotRange → All,
  GridLines → Automatic, IntervalMarkers → "Bands", PlotTheme → "Scientific",
  FrameLabel → {Style["Sample #", 14], Style["θ5 (m^-1)", 14]},
  PlotLabel → Style["Q5 strength error", 16], ImageSize → 500]
```

Out[262]=

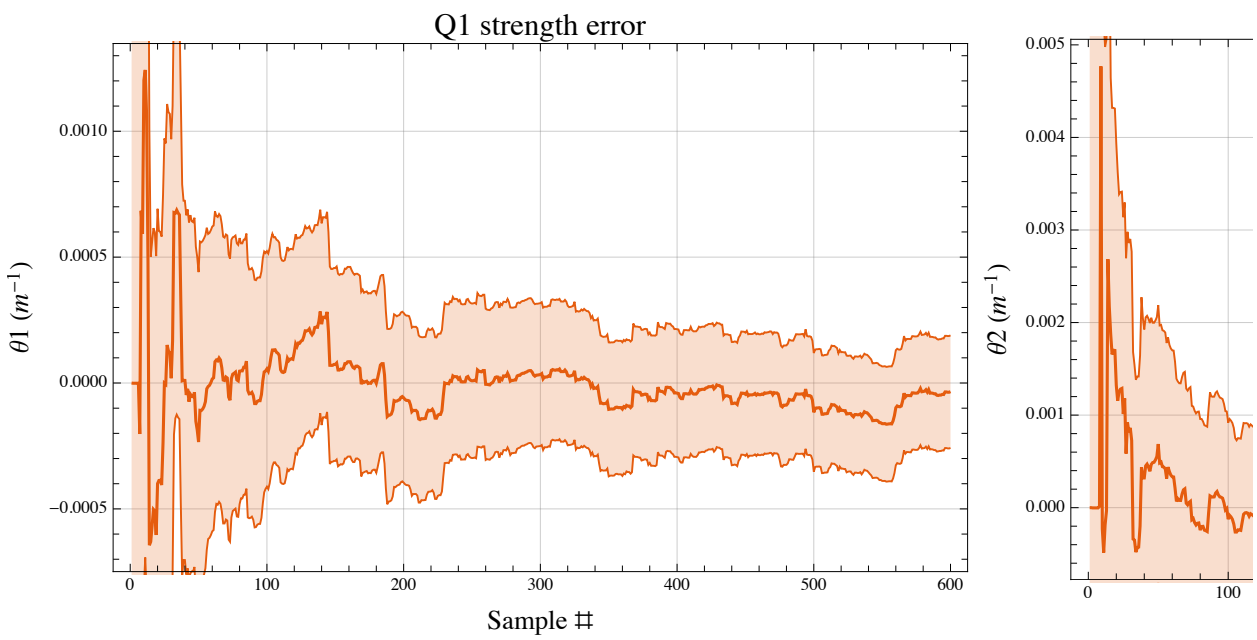
θ6 (Q6 strength error)

In[263]:=
```
g6 = ListPlot[Flatten[Table[Around[xest[[n]][[j]][[8, 1]], Sqrt[Ppred[[n]][[j]][[8, 8]]]],
    {n, Nturns - 1}, {j, 1, 6}]], Joined → True, PlotRange → All,
  GridLines → Automatic, IntervalMarkers → "Bands", PlotTheme → "Scientific",
  FrameLabel → {Style["Sample #", 14], Style["θ6 (m^-1)", 14]},
  PlotLabel → Style["Q6 strength error", 16], ImageSize → 500]
```

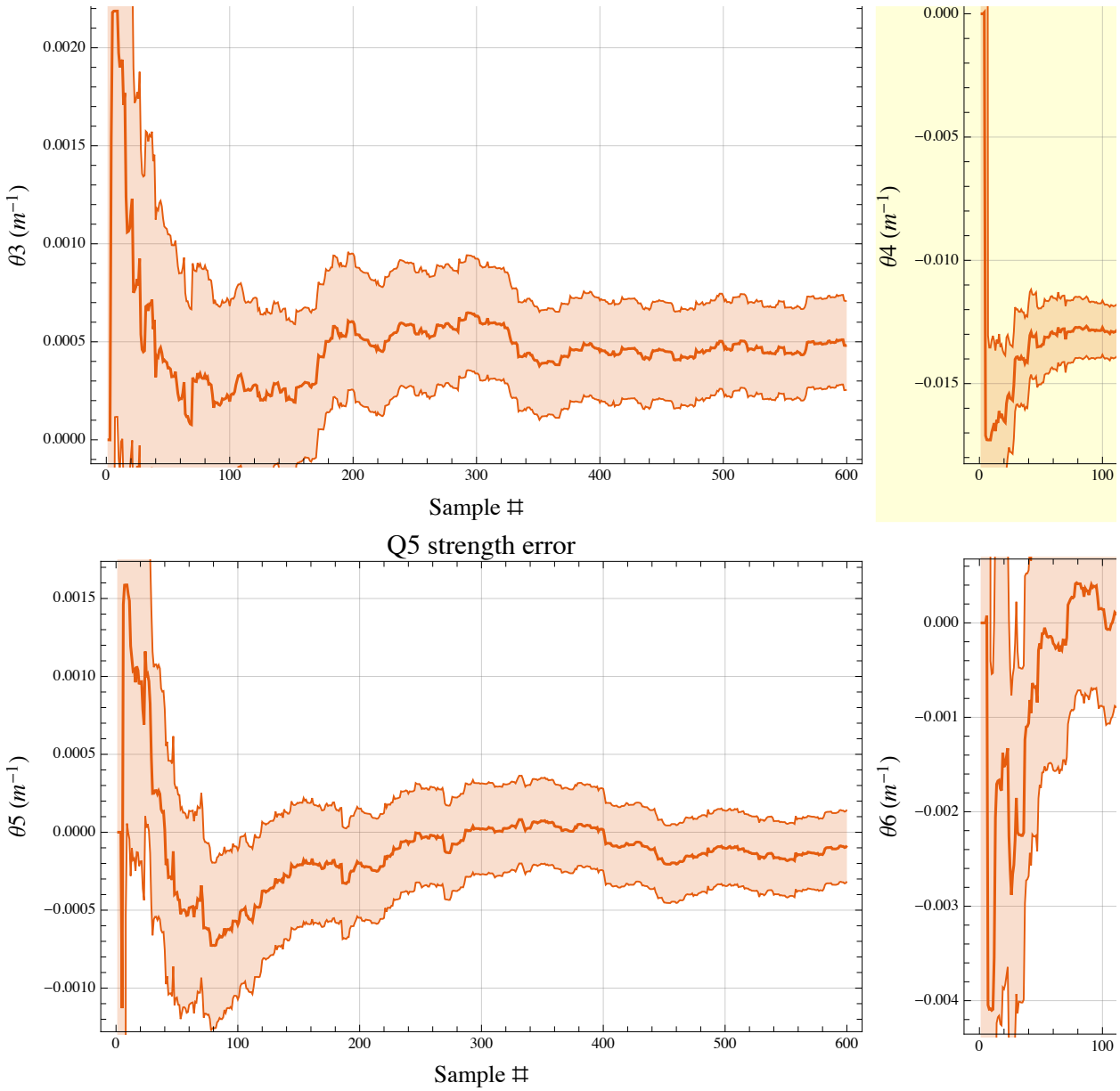
Out[263]=



In[270]:=
```
gg3 = Grid[{{g1, g2}, {g3, g4}, {g5, g6}}]
```



Out[270]=



In[271]:=
```
Export[dir <> "paramFit.pdf", gg3]
```

Out[271]= ~/expt/booster/lattice/math/paramFit.pdf

Focal length

Element 6 in xest[] is $\theta 4$. Note, because I have two fake lens with focal length -θ separated by lq, I have to use the 1/f = 1/u + 1/v formula to calculate the “actual” focal length error. The formula is:

$$\text{Solve}\left[\frac{1}{\text{ff}} == \frac{-1}{\text{ff} - \text{LD}} + \frac{1}{\text{v}},\ \text{v}\right] = \left\{\left\{\text{v} \to \frac{\text{ff}\,(\text{ff}-\text{LD})}{2\,\text{ff}-\text{LD}}\right\}\right\}$$

The v that is found is the focal length of the system w.r.t. the 2nd downstream lens. So to bring it back to the middle of the quad, I have to add lq/2 to this value, i.e.

f = v + lq/2

In[284]:= $\theta v4 = xest[[Nturns - 1]][[6]][[6, 1]]$

Out[284]= -0.0140796

In[279]:= $fv4 = \frac{-1}{xest[[Nturns - 1]][[6]][[6, 1]]}$

Out[279]= 71.0249

In[277]:= $\sigma v4 = Sqrt[Ppred[[n]][[j]][[6, 6]]]$

Out[277]= 0.000465789

In[294]:= $\Delta f = \frac{ff\,(ff - LD)}{2\,ff - LD} + lq/2\ /.\ \{ff \to fv4,\ LD \to lq/2\}$

Out[294]= 35.6998

In[285]:= $dF = D\left[\frac{-1/\theta\,(-1/\theta - LD)}{2\,(-1/\theta) - LD},\ \theta\right]\Delta\theta\ //\ Simplify$

Out[285]= $\frac{\Delta\theta\,(2 + 2\,LD\,\theta + LD^2\,\theta^2)}{\theta^2\,(2 + LD\,\theta)^2}$

In[286]:= $dF\ /.\ \{\theta \to \theta v4,\ LD \to lq/2,\ \Delta\theta \to \sigma v4\}$

Out[286]= 1.17485

In[234]:= $kq = 0.54102;$
$lq = 0.5;$

In[236]:= $flen = \frac{1}{kq\,lq}\ //\ N$

Out[236]= 3.69672

In[237]:= $kq\,0.1$

Out[237]= 0.054102

In[238]:= $1/(kq\,lq\,0.1)$

Out[238]= 36.9672

β function

I am going to use BPM6 and last turn to set $\theta 1$ to $\theta 6$. But first I need the Twiss matrices

Twiss matrices

$$\begin{pmatrix}\beta\\ \alpha\\ \gamma\end{pmatrix}_{s_2} = \begin{pmatrix} C^2 & -2SC & S^2 \\ -CC' & SC'+S'C & -SS' \\ C'^2 & -2S'C' & S'^2 \end{pmatrix} * \begin{pmatrix}\beta\\ \alpha\\ \gamma\end{pmatrix}_{s_1}.$$

In[239]:= $\begin{pmatrix} \text{Cqf} & \text{Sqf} \\ \text{Cpqf} & \text{Spqf} \end{pmatrix}$ = MQFerr[K, l, θ]

Out[239]= $\left\{\left\{\text{Cos}\left[\sqrt{K}\ l\right] - \frac{\theta\ \text{Sin}\left[\sqrt{K}\ l\right]}{\sqrt{K}}, \frac{\text{Sin}\left[\sqrt{K}\ l\right]}{\sqrt{K}}\right\}, \left\{-\theta\ \text{Cos}\left[\sqrt{K}\ l\right] - \sqrt{K}\ \text{Sin}\left[\sqrt{K}\ l\right], \text{Cos}\left[\sqrt{K}\ l\right]\right\}\right\}$

In[240]:= TQFerr[KK_, ll_, θθ_] :=

$$\begin{pmatrix} \text{Cqf}^2 & -2\ \text{Sqf Cqf} & \text{Sqf}^2 \\ -\text{Cqf Cpqf} & \text{Sqf Cpqf} + \text{Spqf Cqf} & -\text{Sqf Spqf} \\ \text{Cpqf}^2 & -2\ \text{Spqf Cpqf} & \text{Spqf}^2 \end{pmatrix} \text{ /. } \{K \to KK,\ l \to ll,\ \theta \to \theta\theta\}$$

In[241]:= $\begin{pmatrix} \text{Cqd} & \text{Sqd} \\ \text{Cpqd} & \text{Spqd} \end{pmatrix}$ = MQDerr[K, l, θ]

Out[241]= $\left\{\left\{\text{Cosh}\left[\sqrt{K}\ l\right] - \frac{\theta\ \text{Sinh}\left[\sqrt{K}\ l\right]}{\sqrt{K}}, \frac{\text{Sinh}\left[\sqrt{K}\ l\right]}{\sqrt{K}}\right\},\right.$
$\left.\left\{-\theta\ \text{Cosh}\left[\sqrt{K}\ l\right] + \sqrt{K}\ \text{Sinh}\left[\sqrt{K}\ l\right], \text{Cosh}\left[\sqrt{K}\ l\right]\right\}\right\}$

In[242]:= TQDerr[KK_, ll_, θθ_] :=

$$\begin{pmatrix} \text{Cqd}^2 & -2\ \text{Sqd Cqd} & \text{Sqd}^2 \\ -\text{Cqd Cpqd} & \text{Sqd Cpqd} + \text{Spqd Cqd} & -\text{Sqd Spqd} \\ \text{Cpqd}^2 & -2\ \text{Spqd Cpqd} & \text{Spqd}^2 \end{pmatrix} \text{ /. } \{K \to KK,\ l \to ll,\ \theta \to \theta\theta\}$$

In[243]:= $\begin{pmatrix} \text{Cl} & \text{Sl} \\ \text{Cpl} & \text{Spl} \end{pmatrix}$ = Mdrift[l]

Out[243]= {{1, l}, {0, 1}}

In[244]:= Tdrift[ll_] := $\begin{pmatrix} \text{Cl}^2 & -2\ \text{Sl Cl} & \text{Sl}^2 \\ -\text{Cl Cpl} & \text{Sl Cpl} + \text{Spl Cl} & -\text{Sl Spl} \\ \text{Cpl}^2 & -2\ \text{Spl Cpl} & \text{Spl}^2 \end{pmatrix}$ /. l → ll

Twiss elements

In[245]:=
```
Telement = {
    {TQFerr, k, lq/2, θ1}, {Tdrift, ld}, {TQDerr, k, lq/2, θ2},
    {TQDerr, k, lq/2, θ2}, {Tdrift, ld}, {TQFerr, k, lq/2, θ3},


    {TQFerr, k, lq/2, θ3}, {Tdrift, ld}, {TQDerr, k, lq/2, θ4},
    {TQDerr, k, lq/2, θ4}, {Tdrift, ld}, {TQFerr, k, lq/2, θ5},

    {TQFerr, k, lq/2, θ5}, {Tdrift, ld}, {TQDerr, k, lq/2, θ6},
    {TQDerr, k, lq/2, θ6}, {Tdrift, ld}, {TQFerr, k, lq/2, θ1}
   };
```

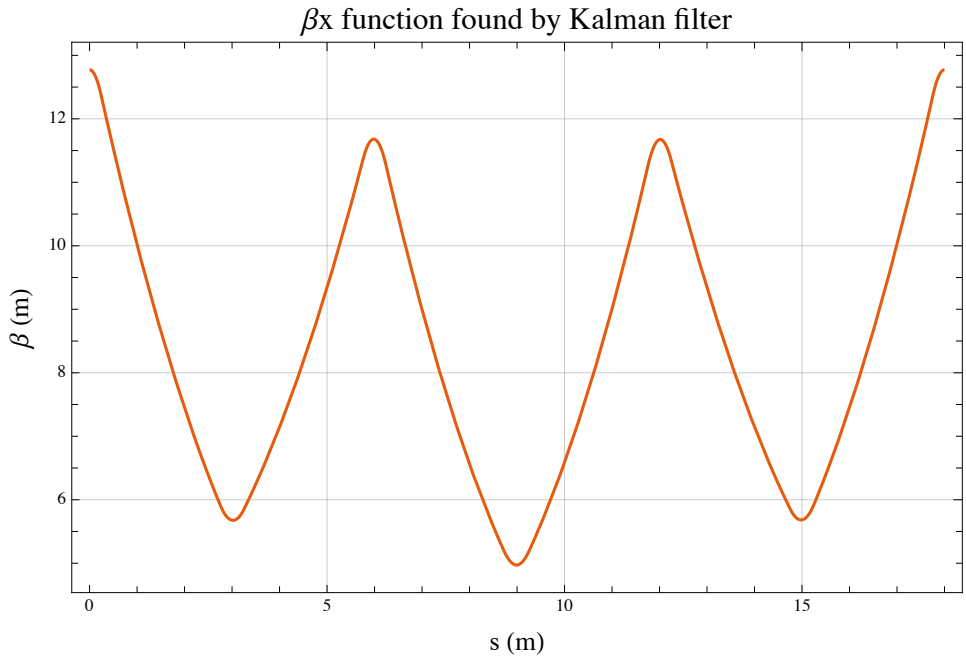
Calculate β's

In[246]:=
```
βx[s_] :=
  Module[{Δk1, Δk2, Δk3, Δk4, Δk5, Δk6lattice, ϕ, βs, αs, γs, xs, d0, d1, Tw, i},
   (*get the strengths of the errors from BPM6 at the last turn*)
   {Δk1, Δk2, Δk3, Δk4, Δk5, Δk6} = { xest[[Nturns - 1]][[6]][[3, 1]],
     xest[[Nturns - 1]][[6]][[4, 1]], xest[[Nturns - 1]][[6]][[5, 1]], xest[[Nturns - 1]][[6]][[6, 1]],
     xest[[Nturns - 1]][[6]][[7, 1]], xest[[Nturns - 1]][[6]][[8, 1]]};
   (* Make the lattice to find xs*)

   lattice = FF6to1.FF5to6.FF4to5.FF3to4.FF2to3.FF1to2 /.
     {θ1 → Δk1, θ2 → Δk2, θ3 → Δk3, θ4 → Δk4, θ5 → Δk5, θ6 → Δk6};

   (*got the lattice, so solve for the initial Twiss parameters, xs*)
   ϕ = ArcCos[1/2 Tr[lattice]];
   βs = lattice[[1, 2]]/Sin[ϕ] // N;
   αs = Chop[(lattice[[1, 1]] - Cos[ϕ])/Sin[ϕ]] // N;
   γs = -lattice[[2, 1]]/Sin[ϕ] // N;

   xs = (βs, αs, γs);

   (*Finally, I can do the transport*)
```

```
d0 = d1 = 0;
Tw = IdentityMatrix[3];

For[i = 1, i ≤ Length[Telement], i++,
 (*Is it a quad?*)
 If[(Telement⟦i, 1⟧ == TQFerr) || (Telement⟦i, 1⟧ == TQDerr),
  d1 += Telement⟦i, 3⟧; (*get length*)
  If[d0 ≤ s ≤ d1,
   Return[Telement⟦i, 1⟧[Telement⟦i, 2⟧, s - d0, Telement⟦i, 4⟧].Tw.xs /.
      {θ1 → Δk1, θ2 → Δk2, θ3 → Δk3, θ4 → Δk4, θ5 → Δk5, θ6 → Δk6}];
  ];
  (*not in the quad, so transport through*)
  Tw = Telement⟦i, 1⟧[Telement⟦i, 2⟧, Telement⟦i, 3⟧, Telement⟦i, 4⟧].Tw /.
    {θ1 → Δk1, θ2 → Δk2, θ3 → Δk3, θ4 → Δk4, θ5 → Δk5, θ6 → Δk6};
 ];

 (*Is it a drift?*)
 If[Telement⟦i, 1⟧ == Tdrift,
  d1 += Telement⟦i, 2⟧; (*get length*)
  If[d0 ≤ s ≤ d1,
   Return[Telement⟦i, 1⟧[s - d0].Tw.xs];
  ];
  (*not in the drift space, so transport through*)
  Tw = Telement⟦i, 1⟧[Telement⟦i, 2⟧].Tw;
 ];

 d0 = d1;
]; (*for*)
];
```

In[247]:=
```
Plot[βx[s]⟦1, 1⟧ , {s, 0, 3 * (lq / 2 + ld + lq + ld + lq / 2)}, PlotTheme → "Scientific",
 GridLines → Automatic, FrameLabel → {Style["s (m)", 14], Style["β (m)", 14]},
 PlotLabel → Style["βx function found by Kalman filter", 16], ImageSize → 500]
```

Out[247]=

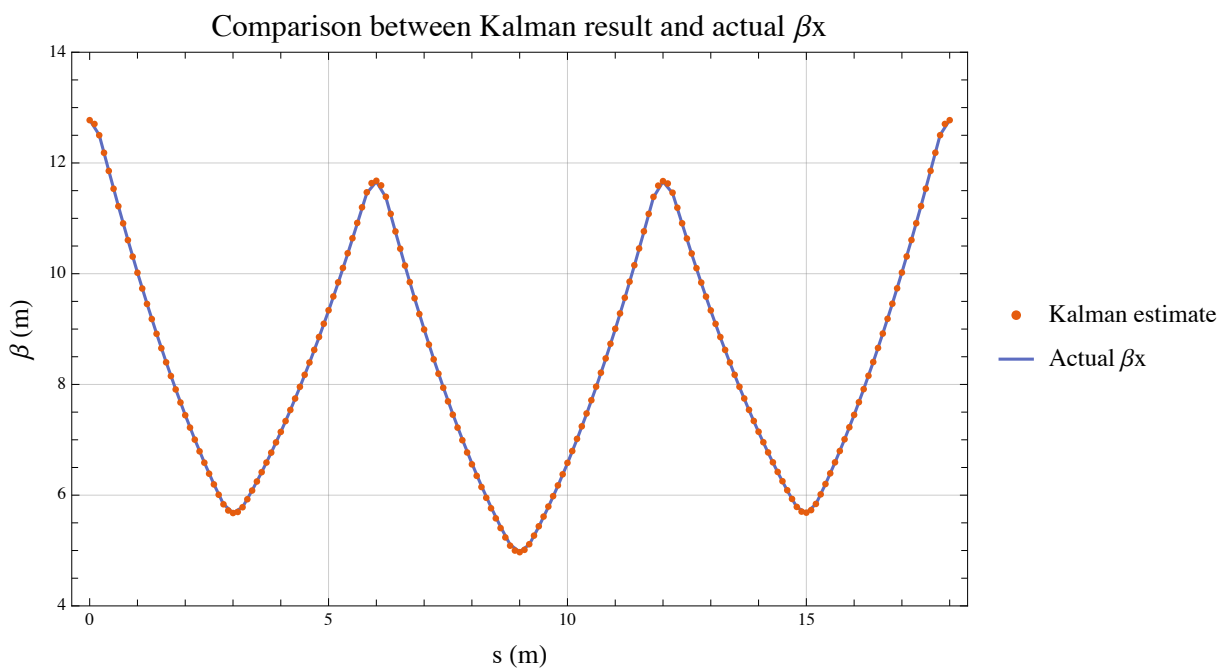


Get the β function that generated the error. The data came from fodo_error.nb

In[248]:=
```
Lβactual = ReadList[dir <> "betax_err.dat", {Number, Number}];
```

In[287]:=
```
gg4 = ListPlot[{Table[{s, βx[s][[1, 1]] }, {s, 0, 3 * (lq / 2 + ld + lq + ld + lq / 2), 0.1}],
    Lβactual[[ ;; ;; 2]]}, Joined → {False, True},
  PlotRange → {4, 14}, PlotTheme → "Scientific", GridLines → Automatic,
  FrameLabel → {Style["s (m)", 14], Style["β (m)", 14]},
  PlotLabel → Style["Comparison between Kalman result and actual βx", 16],
  PlotLegends → { "Kalman estimate", "Actual βx"}, ImageSize → 500]
```

Out[287]=

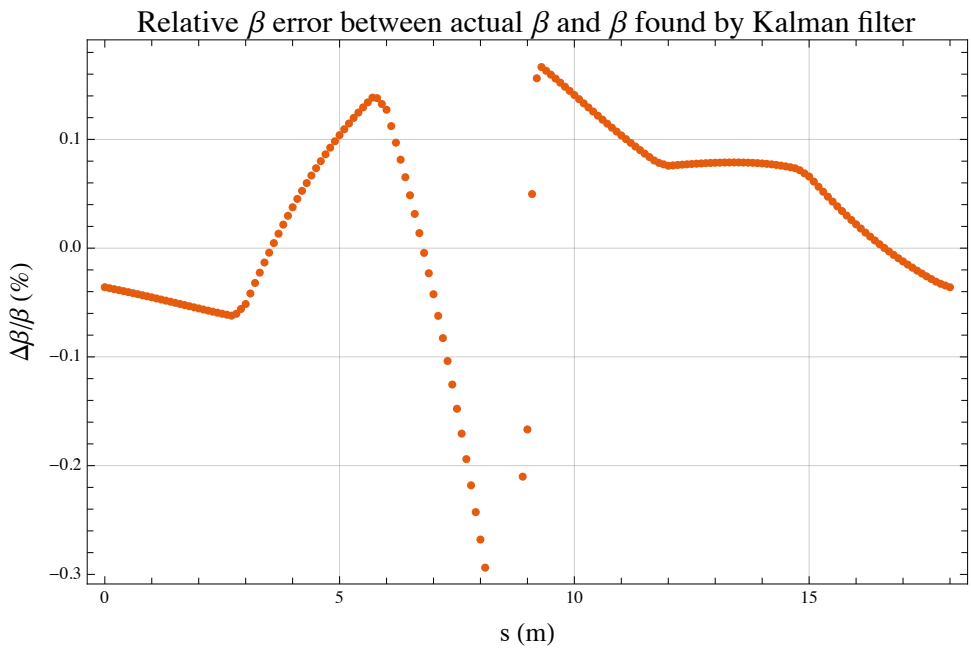


In[288]:=
```
Export[dir <> "beta.pdf", gg4]
```

Out[288]=
```
~/expt/booster/lattice/math/beta.pdf
```

In[289]:=
```
gg5 = ListPlot[Table[{Lβactual[[i, 1]], βx[Lβactual[[i, 1]][[1, 1]] - Lβactual[[i, 2]]] / Lβactual[[i, 2]] 100},
    {i, Length[Lβactual]}], PlotTheme → "Scientific", GridLines → Automatic,
  FrameLabel → {Style["s (m)", 14], Style["Δβ/β (%)", 14]}, PlotLabel →
   Style["Relative β error between actual β and β found by Kalman filter", 16],
  ImageSize → 500]
```

Out[289]=



In[290]:=
```
Export[dir <> "betaErr.pdf", gg5]
```

Out[290]=
```
~/expt/booster/lattice/math/betaErr.pdf
```