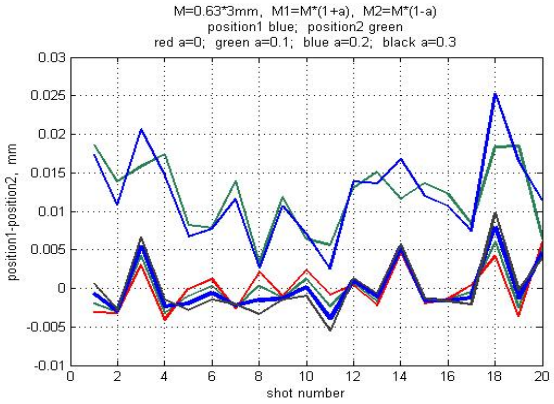


Fig. 1. This plot was shown 04 April. Some peaking =?

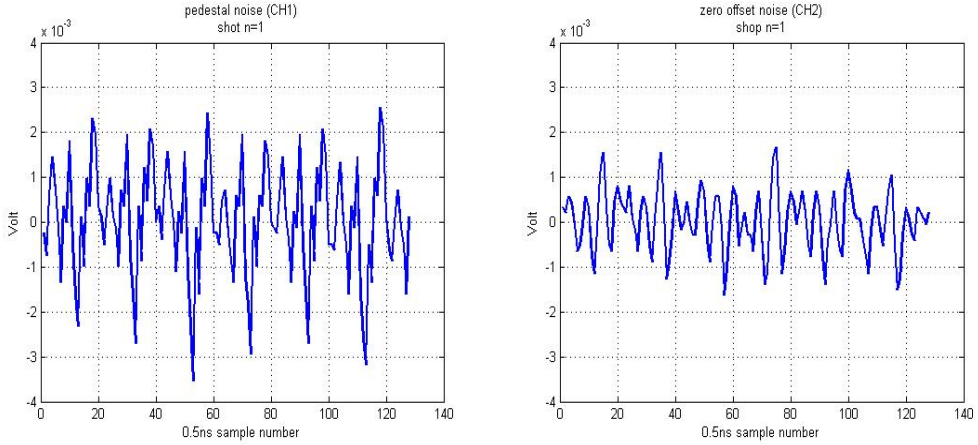


Fig. 2. Investigation of the raw signals. Pedestal noise (Sum1, CH1) and zero line noise (Dif1, CH2) are shown. Some regular oscillations =?

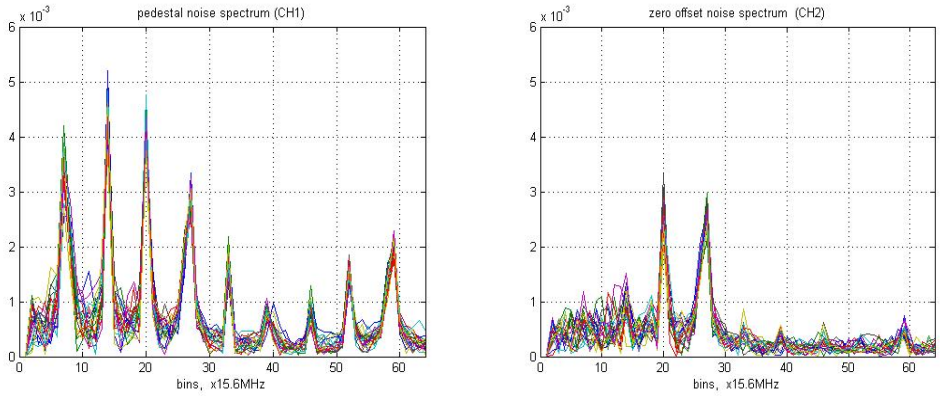


Fig. 3. The spectra of the pedestal noise and of the zero line noise. The peaks are at (about) 100MHz, 200MHZ, 300MHz, …
The GFT reference is 100MHz! Note the bigger the GFT input voltage, the more harmonics are there.

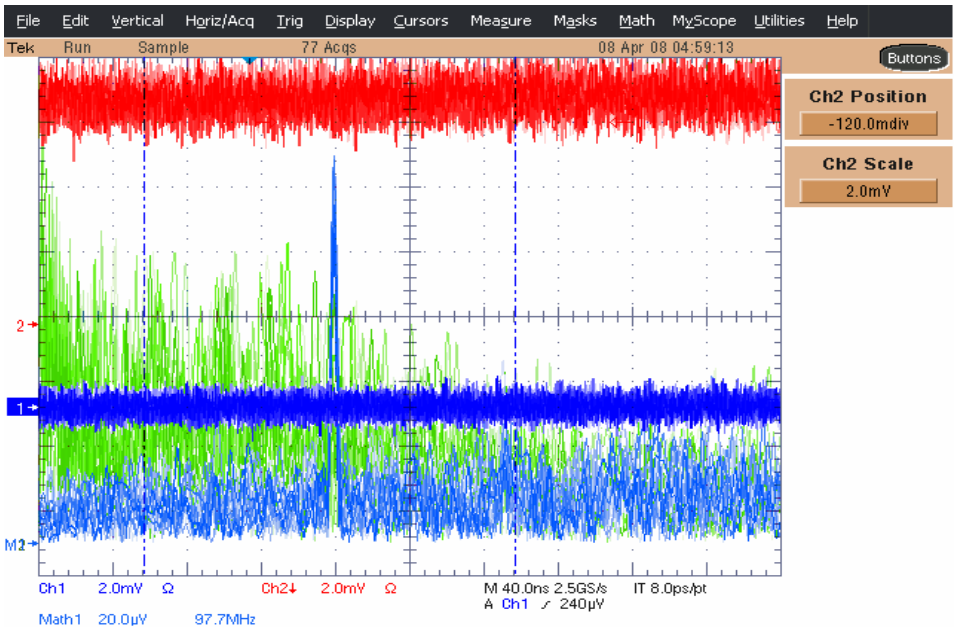


Fig. 4. The Dif1 output (red) as seen on oscilloscope. Its spectrum is given green. Another scope channel (blue): a sine 400MHz 100uV rms.
No oscillation at the ADC Driver output. So, GFT…

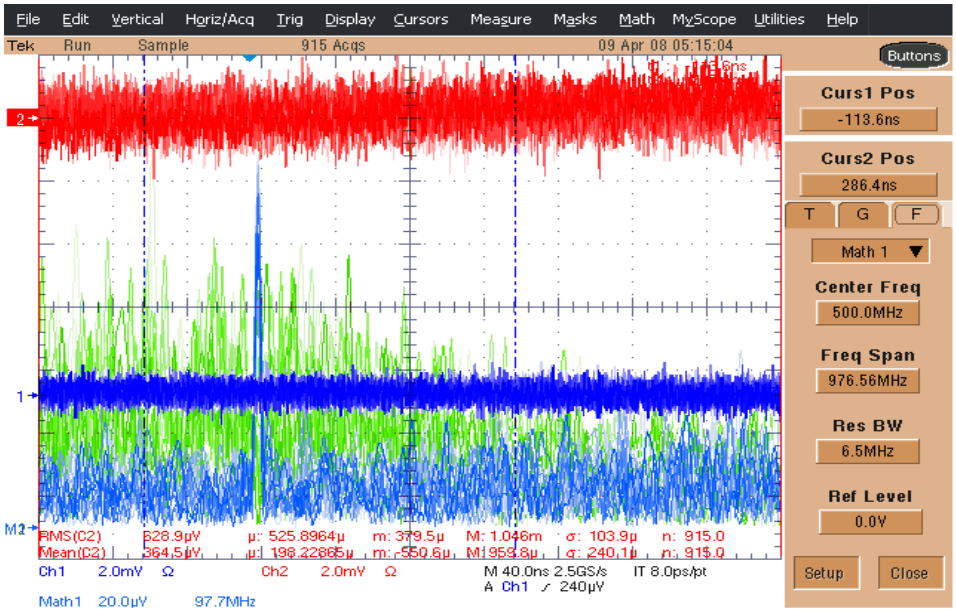


Fig. 5. An important observation. The Driver output noise is increasing to the low frequency end as it can be seen in Fig. 4. In time domain, the zero line as a whole is jumping shot-to-shot.
In a single bunch FF/FB BPM, a blocking capacitor should be used. The output spectrum with it is shown green. Blue: a sine 300MHz 100uV rms.

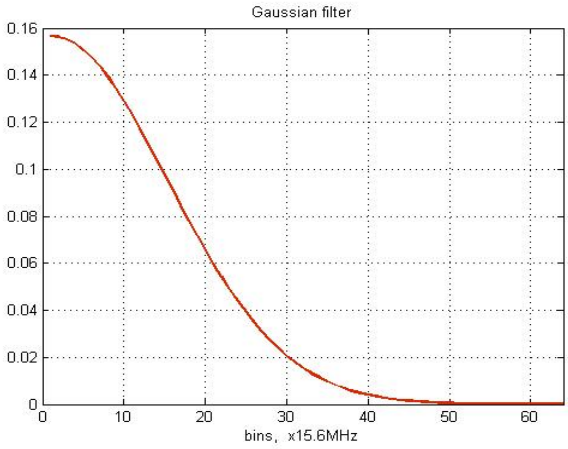


Fig. 6. How to reduce oscillation contribution to the BPM resolution calculated using these data?
A gaussian filter!

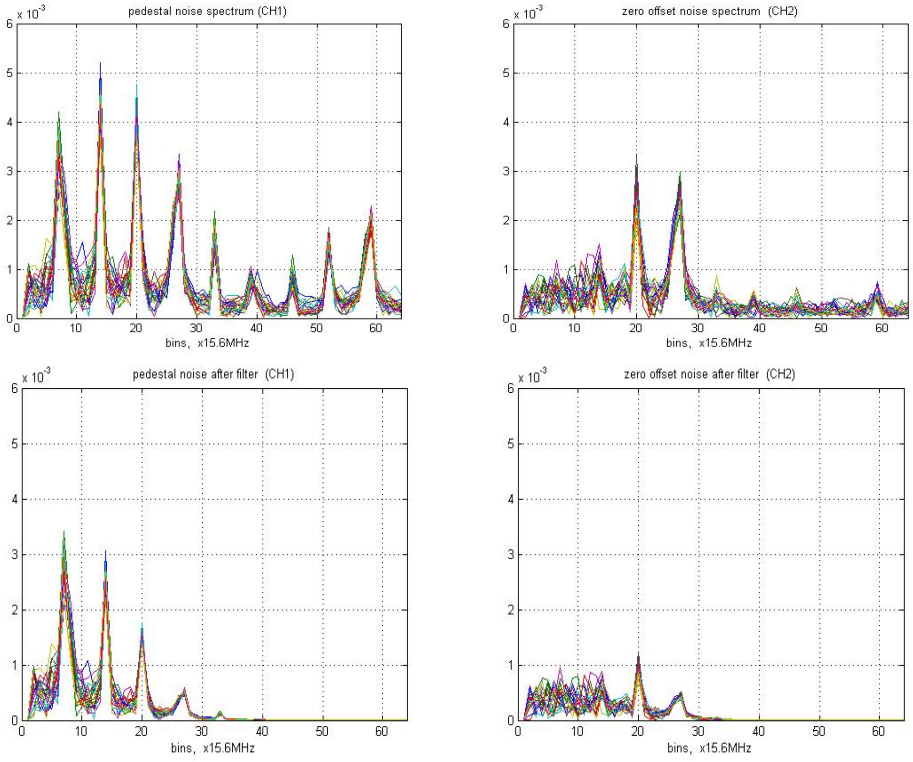


Fig. 7. Top: the spectra of the pedestal noise and of the zero line noise again.
Bottom: the spectra after filter.
In both the Sum and Dif, the BPM noise in the GFT bandwidth is conserved.

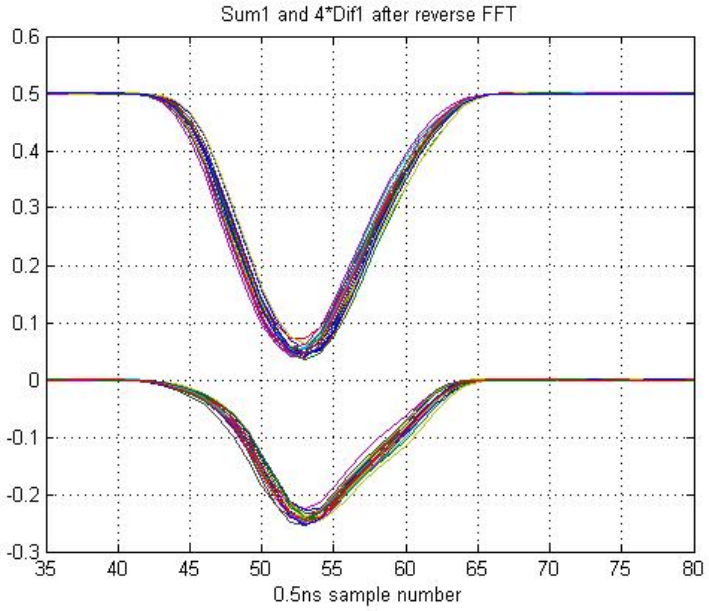


Fig. 8. The Sum1 and Dif1 signals after the filters.

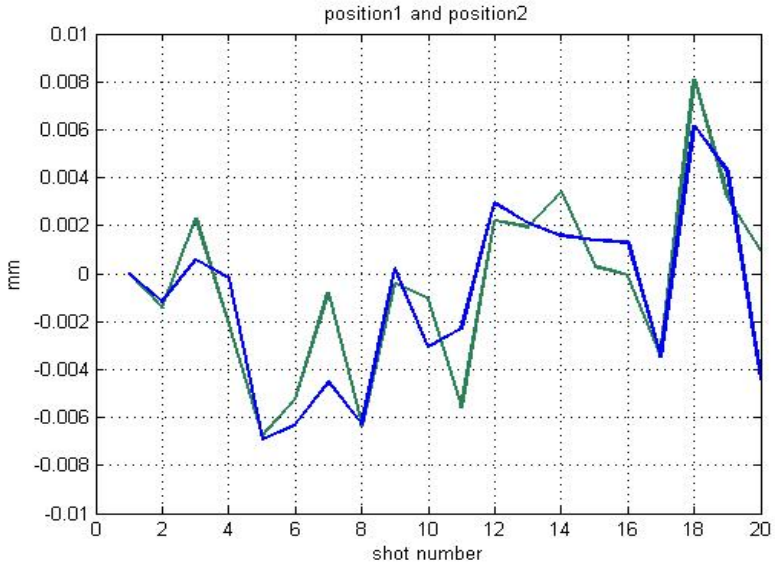
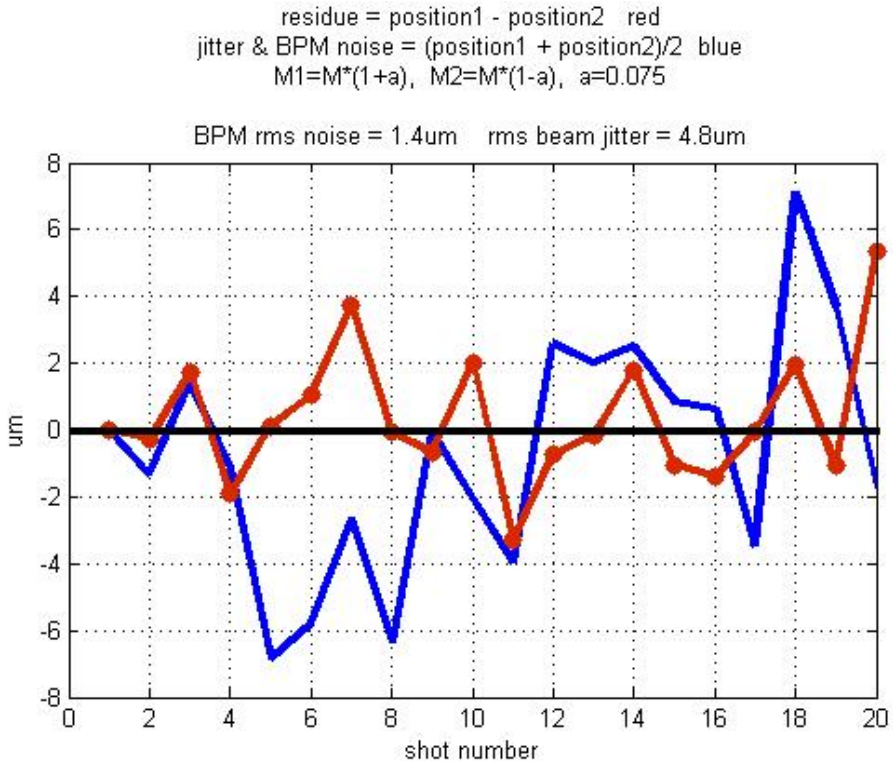


Fig. 9. Position1 and Position2. Each of them is sum of shot-to-shot beam jitter and shot-to-shot BPM noise.



Fog. 10. A residue (red) as (position1-position2) and a jitter + BPM noise (blue) as (position1+position2)/2. The array 22-28-16 was used.

```
M=0.63*3;          %[mm]
a=+0.075;
M1=M*(1+a);
M2=M*(1-a);

for n=1:N
   pos1(n)=M1*Dd2(n)/Sum1(n);
   pos2(n)=M2*Dd4(n)/Sum3(n);

   pp1=mean(pos1);
   pp2=mean(pos2);
   Pos1(n)=pos1(n)-pp1;
   Pos2(n)=pos2(n)-pp2;

   residue12_um(n)=1000*(Pos1(n)-Pos2(n));
   BPM_rms_noise_um=std(residue12_um)/sqrt(2);

   jitterANDnoise_um(n)=1000*(Pos1(n)+Pos2(n))/2;
   rms_jitter_um=(sqrt((std(2*jitterANDnoise_um))^2-2*(BPM_rms_noise_um)^2))/sqrt(2);
end
```

The BPM rms noise is 1.4um. The shot-to-shot rms jitter is 4.8um.