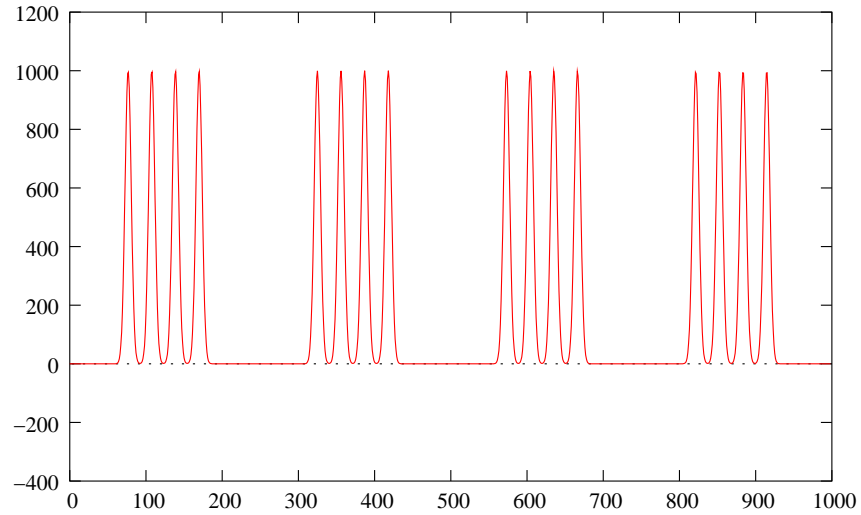
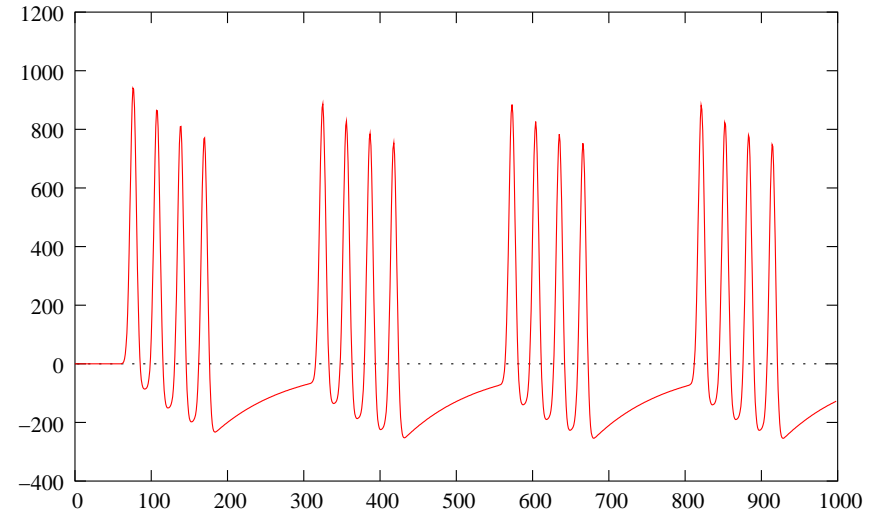




Properties of the signal

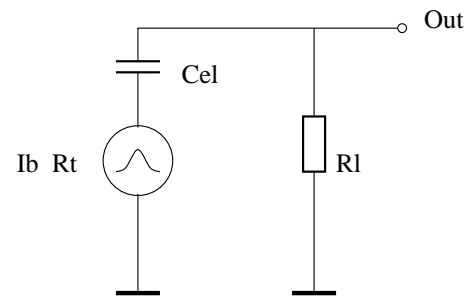


Instantaneous beam current



PU output signal (simulated)

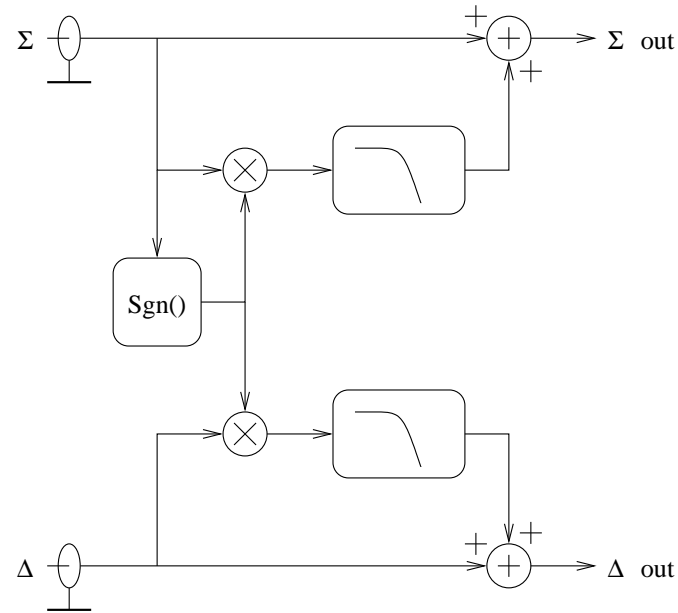
The PU, with its load resistance, yields a high-pass filtered version of the instantaneous beam current.





Principle of base line restitution

Wrong, alas



Block diagram of base line restorer

$$B_{\Sigma,n} = aB_{\Sigma,n-1} + (1-a)|\Sigma_n|$$

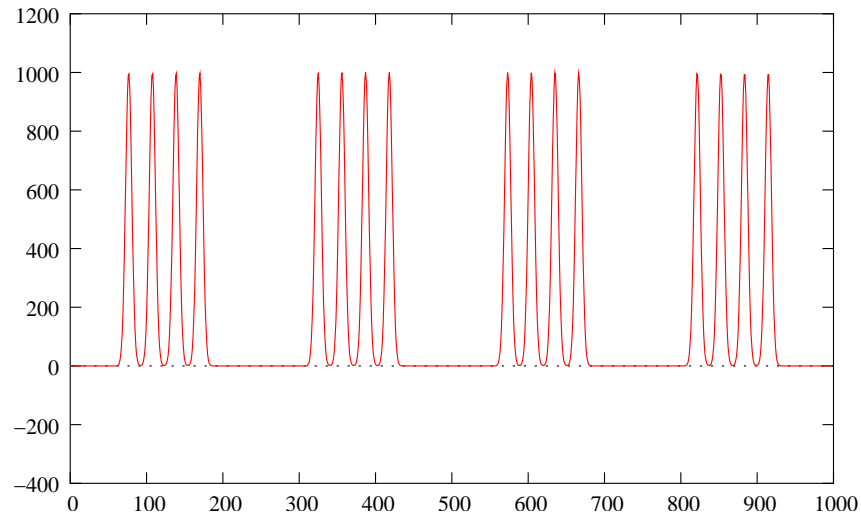
$$\Sigma_n = \Sigma_{raw} + B_{\Sigma,n}$$

$$B_{\Delta,n} = aB_{\Delta,n-1} + (1-a)\text{sgn}(\Sigma)\Delta_{raw}$$

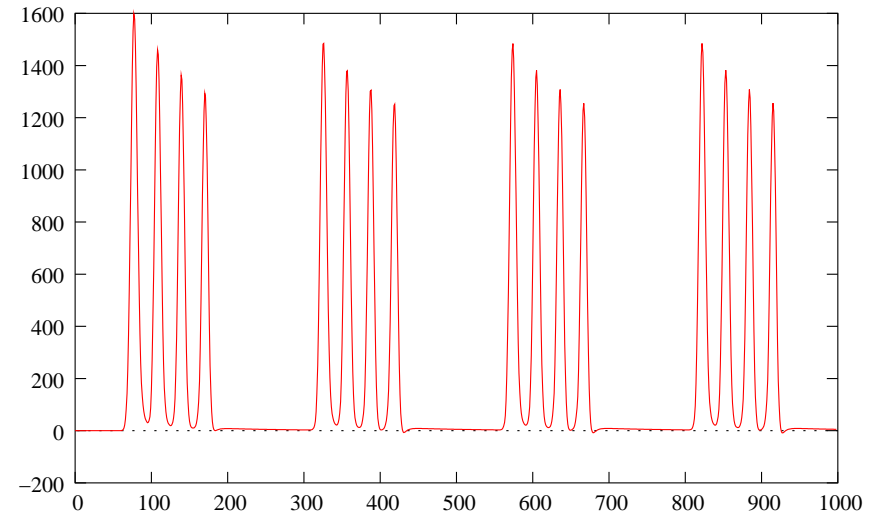
$$\Delta_n = \Delta_{raw} + B_{\Delta,n}$$



Effect of Base Line Restoration



It should be like this...

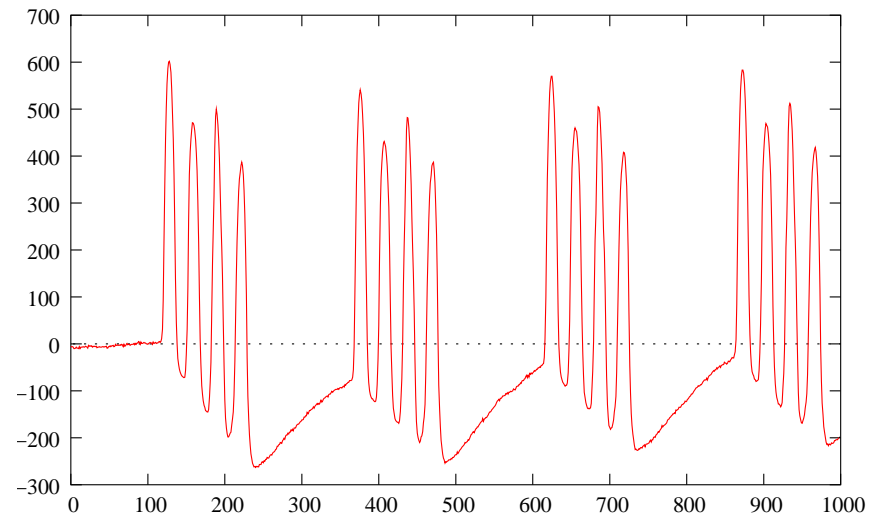


... but it looks like that

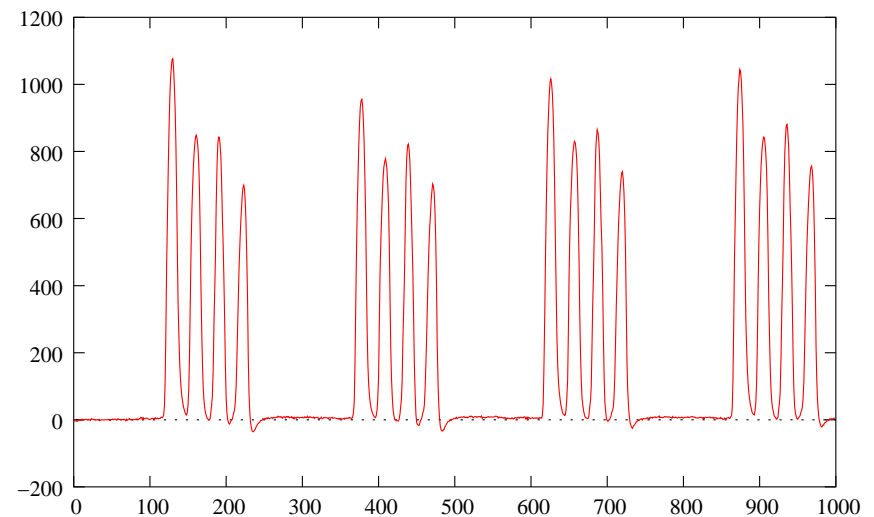
This BLR algorithm messes up the amplitude



Why didn't I notice?



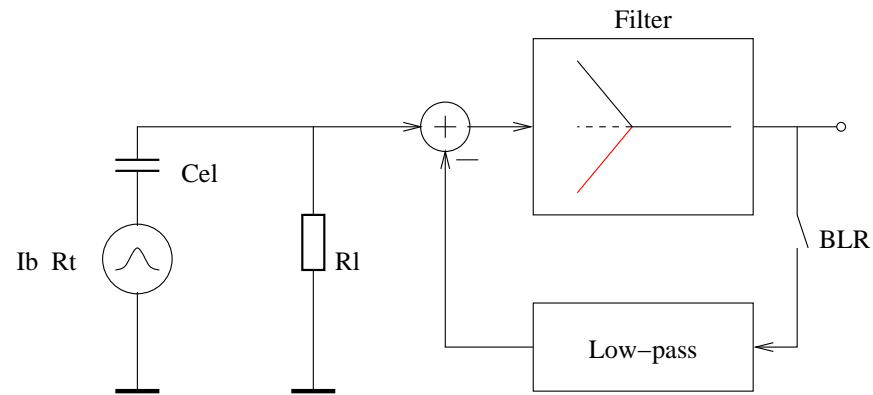
A real PU signal from an AD type beam



Base line restored



Another try at restoring the base line

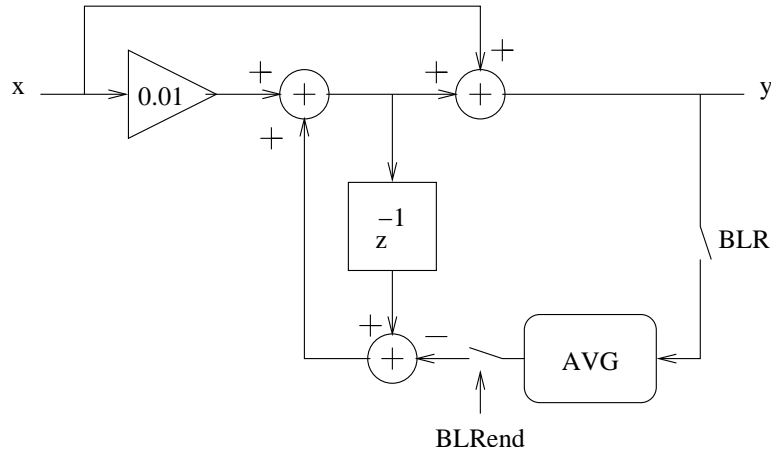


- Amplify the low frequencies strongly
- Fix base line wander by clamping "at the right instants"
- Timing reference required



Base line restoration

Maxim's algorithm (MatLab)



Principle of Maxim's BLR

Problems:

- Length of BLR is hard-wired
- Restoration is brutal

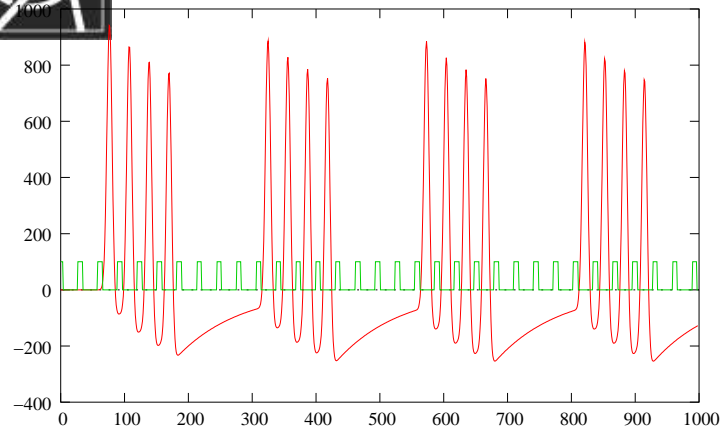
```

%Baseline restoration
%calculate a baseline approximation
%works ONLY if the pll is LOCKED and
%the gate is SET properly!
if C(p1,tableB)
    m=0;    m1=0;
    count1=0;
elseif count1<4 % 4 points used for baseline correction
    m=m+s;    m1=m1+s1; % add 4 samples
    count1=count1+1;
if count1==4
    b = b-m/4; %overflow protection and vertical shift
    b1= b1-m1/4; %the same for Delta
end
end
%inversion of lowpass
b = b+0.01*Data;
s = Data+b; %restored Sigma
b1 = b1+0.01*Data2;
s1 = Data2+b1; %restored Delta
signalB(k) = s; %accumulate restored Sigma
signalB2(k) = s1; %accumulate restored Delta
%End of baseline restoration

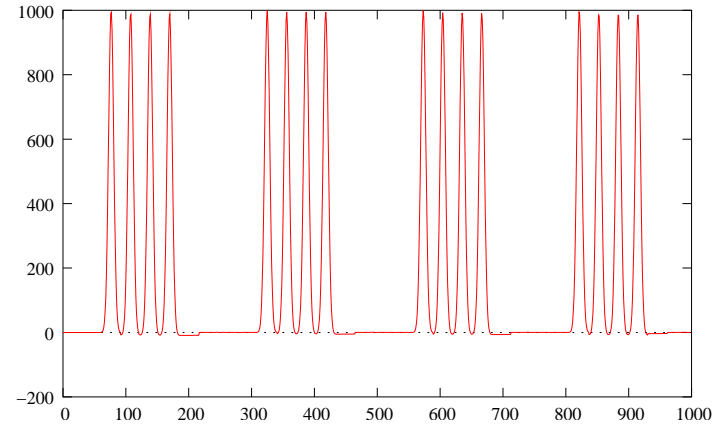
```



Effect on simulated data

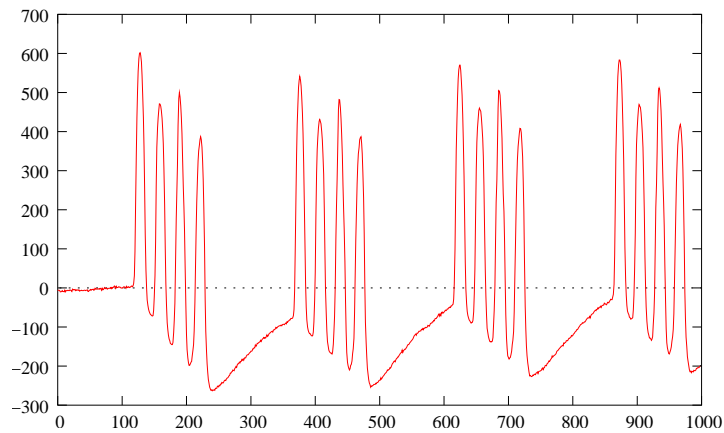


BLR timing reference required

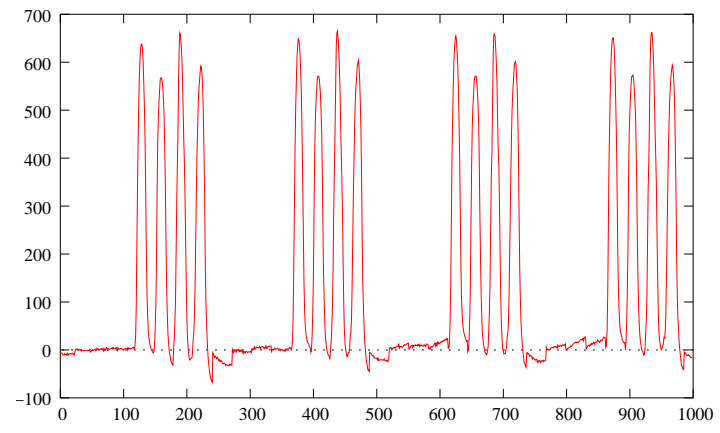


Result on simulated AD data

Effect on real data



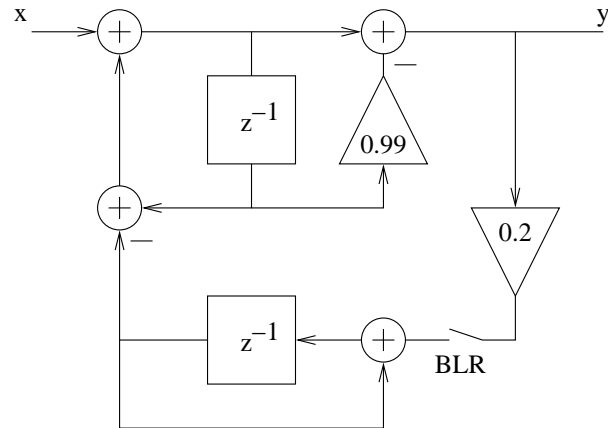
Real acquired AD-type signal



Base line restored



A different implementation



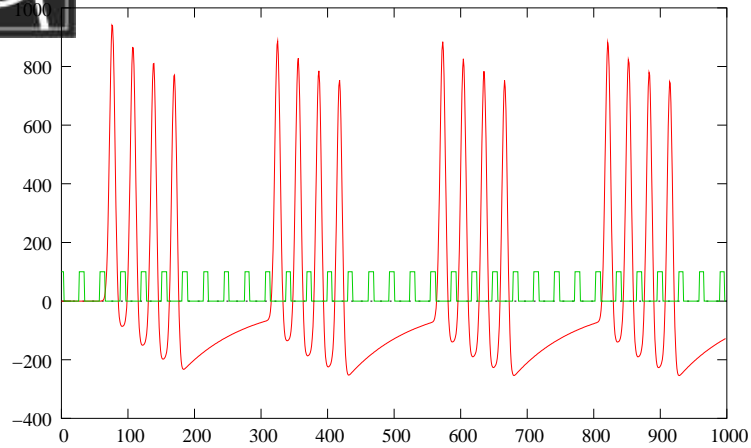
```
y = (x-bl) + 0.01 * e;  
e += (x-bl);  
if (blr != 0) {  
    bl += 0.2 * y;  
}  
printf("%lf\n",y);
```

Maxim's algorithm, reviewed by JMB

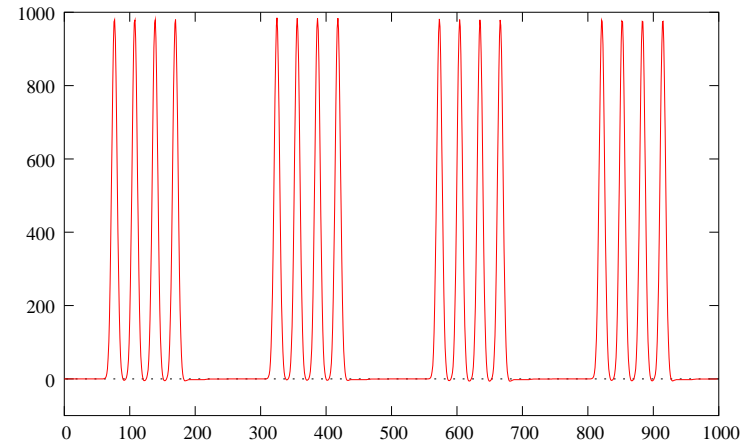
A feedback loop works to get the base line to zero on average.



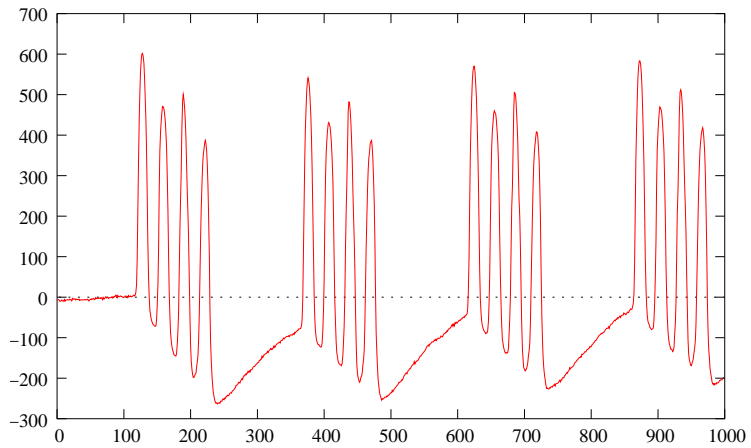
Maxim's algorithm, reviewed by JMB



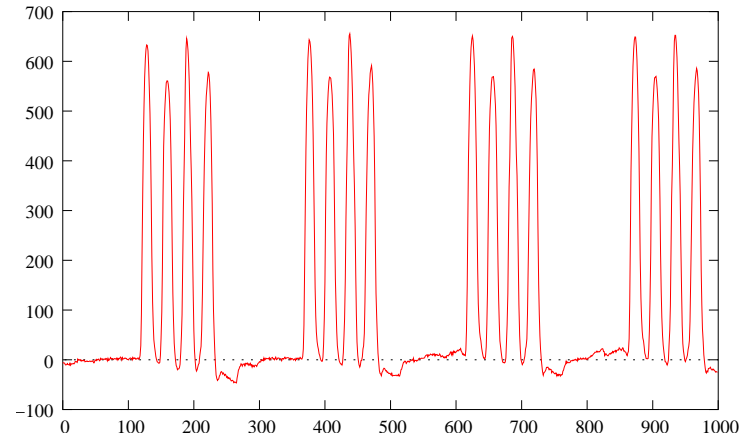
Algorithm requires use of BLR timing reference signal



Result of new BLR algorithm (simulated data)



Real acquired AD-type signal



Base line restored