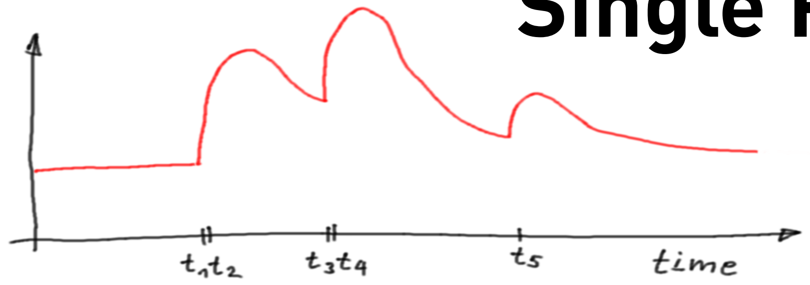
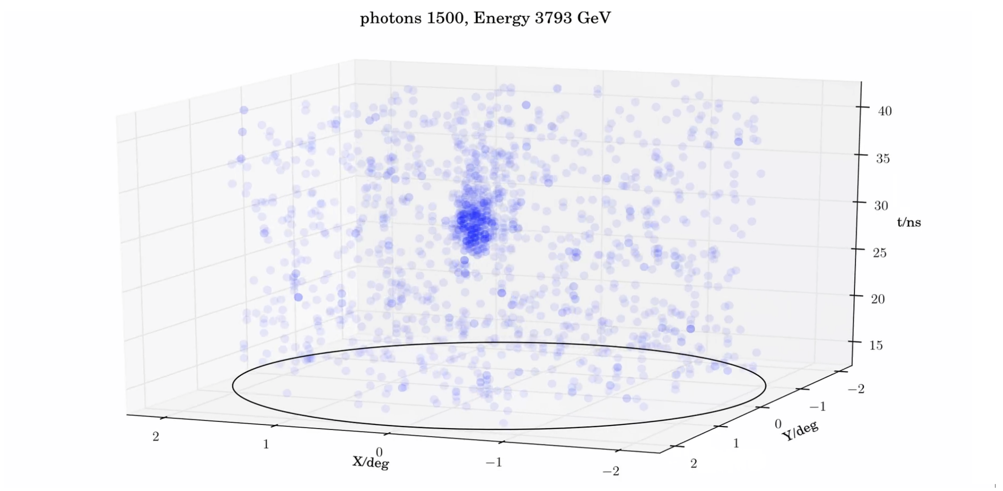
Single Pulse Extractor



[76ns, 45ns, 77ns, 46ns, 110ns]



and novel photon stream

Sebastian Achim Mueller
FACT General Meeting
Wuerzburg 2016

Motivation

FACT is recording full, uncompressed video sequences of 300 frames per event (300 float values / pixel)

Many other IACTs (VERITAS, HESS, WHIPPLE, CAT, HEGRA) only have time and charge to digital converters (2 float values / pixel)

Motivation

FACT is even storing the video sequences which makes its records **more than 2 orders of magnitude** larger. But so far the FACT analysis software only implements a time and charge converter.

A single pulse extractor, finding the arrival times of each pulse on a time line, might increase the analysis performance

It is time to get something in return for all the money and time we spent on storage and bandwidth!

Motivation

So far, FACT has a software time and charge converter which extracts the arrival time of the 'main pulse' and the photon equivalent of the 'main pulse'
(300 float values are reduced to 2 float values)

Challenge of the time and charge converter:
What is the main pulse?

For large pulses the main pulse is easy to find but for an IACT it is all about the small pulses

Lower primary particle energy reconstruction threshold
→reconstruct the smallest pulses

Motivation

Lower the primary particle energy reconstruction threshold by recognizing smaller pulses

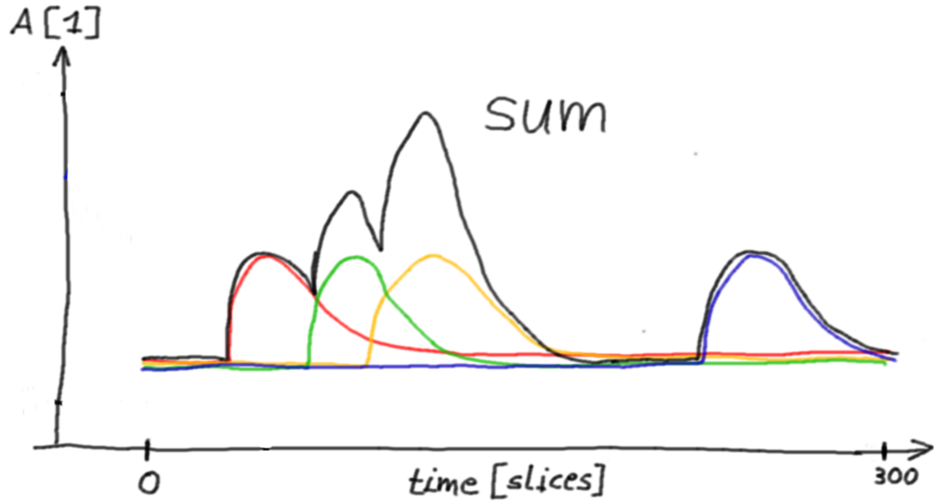
Increase gamma hadron separation power by new features based on the photon arrival time distribution (e.g. hadronic rain, Eckart Lorenz)

Calibrate the SIPM gain by extract the single p.e. pulses during usual observation in dark nights.

Method

Assumption
The amplitude distribution of a time line is only the sum of single pulses. No other effects contribute to the amplitudes.

We neglect electronic noise and assume that all read out artifacts (spikes, jumps,...) are already removed.



Method

```
->Initiate
  ->Template Pulse time lines
  ->Template Pulse integral
->Input time line
->do {
    ->convolve input time line
      and template pulse
    ->find max response in
      convolution time line
    ->store arrival time of
      max response
    ->subtract template pulse
      from input time line
    ->apply AC coupling to
      input time line
  } while(
      has max response in
      convolution time line
  )
->return arrival times
```

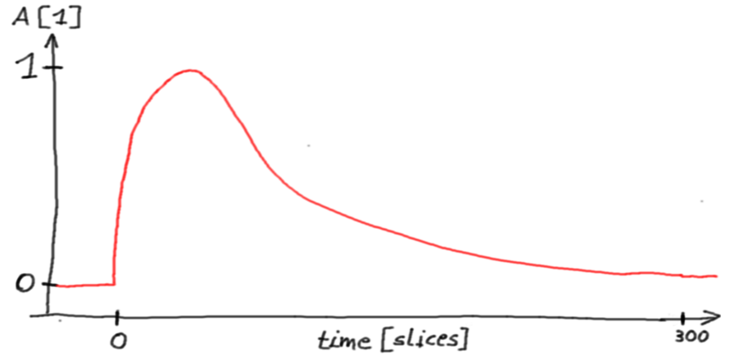
A robust pattern recognition to find the pulses on a time line using template convolution.
(by Sebastian)

An extractor to count the pulses iteratively by subtracting template pulses.
(by Adrian)

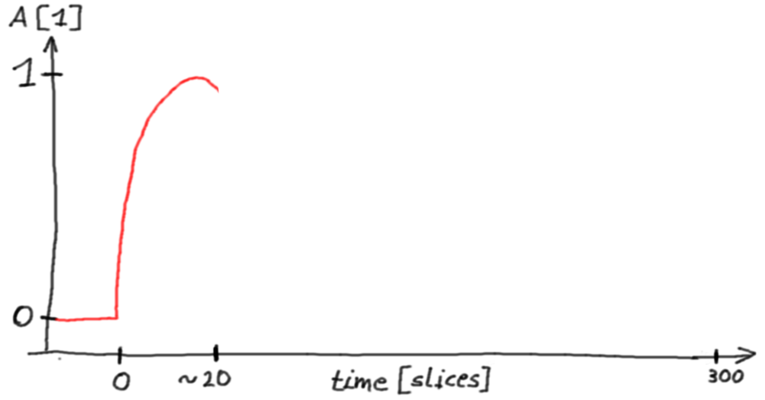
Algorithm

```
->**Initiate**
   ->**Template Pulse time lines**
   ->Template Pulse integral
->Input time line
->do {
    ->convolve input time line
       and template pulse
    ->find max response in
       convolution time line
    ->store arrival time of
       max response
    ->subtract template pulse
       from input time line
    ->apply AC coupling to
       input time line
 } while(
       has max response in
       convolution time line
 )
->return arrival times
```

Template Pulse
time line (full)



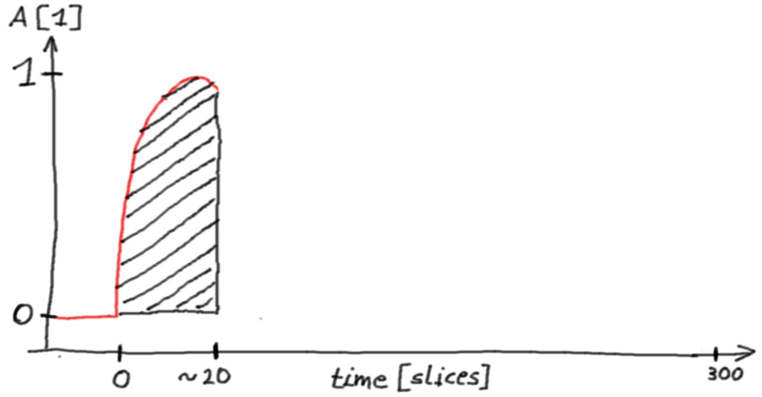
Template Pulse
time line (rising edge)



Algorithm

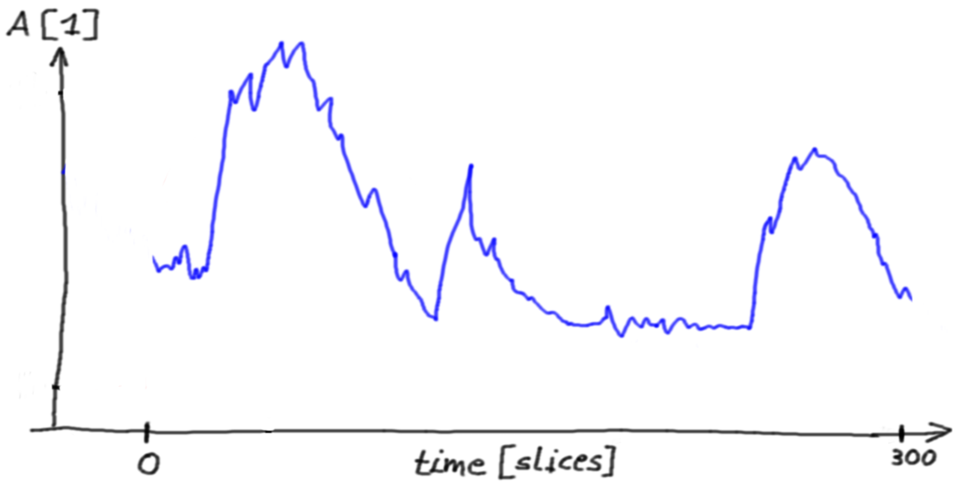
```
->**Initiate**
  ->Template Pulse time lines
  ->**Template Pulse integral**
->Input time line
->do {
   ->convolve input time line
     and template pulse
   ->find max response in
     convolution time line
   ->store arrival time of
     max response
   ->subtract template pulse
     from input time line
   ->apply AC coupling to
     input time line
 } while(
     has max response in
     convolution time line
 )
->return arrival times
```

Needed to normalize upcoming convolution



Algorithm

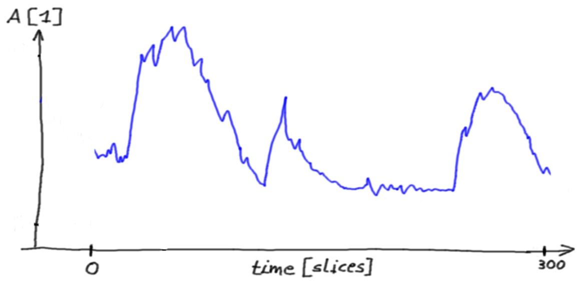
```
->Initiate
   ->Template Pulse time lines
   ->Template Pulse integral
->**Input time line**
->do {
    ->convolve input time line
       and template pulse
    ->find max response in
       convolution time line
    ->store arrival time of
       max response
    ->subtract template pulse
       from input time line
    ->apply AC coupling to
       input time line
  } while(
       has max response in
       convolution time line
  )
->return arrival times
```



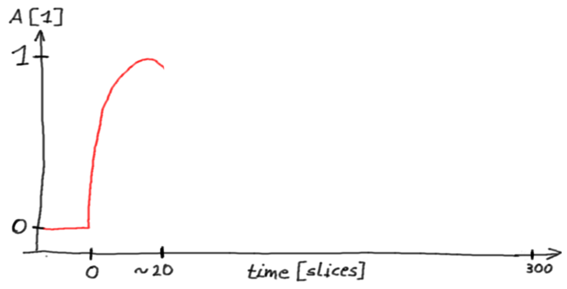
Amplitude is normalized to 1 photo equivalent pulse

Algorithm

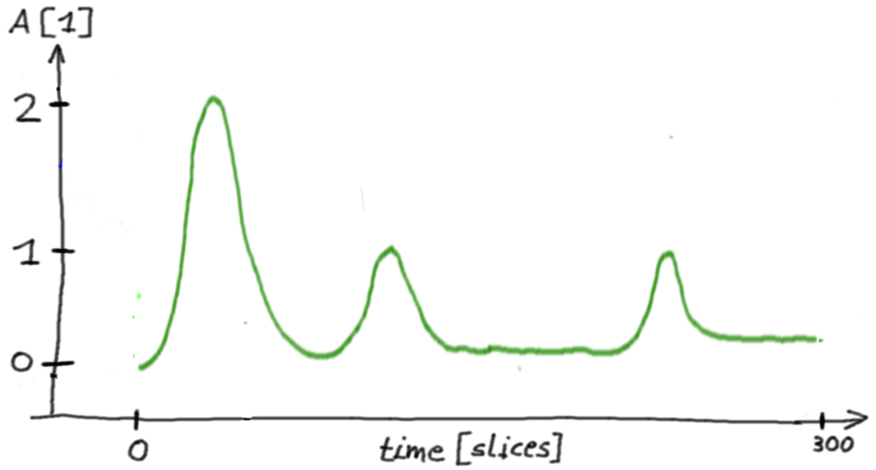
```
->Initiate
   ->Template Pulse time lines
   ->Template Pulse integral
->Input time line
->do {
    ->convolve input time line
      and template pulse
    ->find max response in
      convolution time line
    ->store arrival time of
      max response
    ->subtract template pulse
      from input time line
    ->apply AC coupling to
      input time line
 } while(
      has max response in
      convolution time line
 )
->return arrival times
```



*



Normalized using template pulse integral

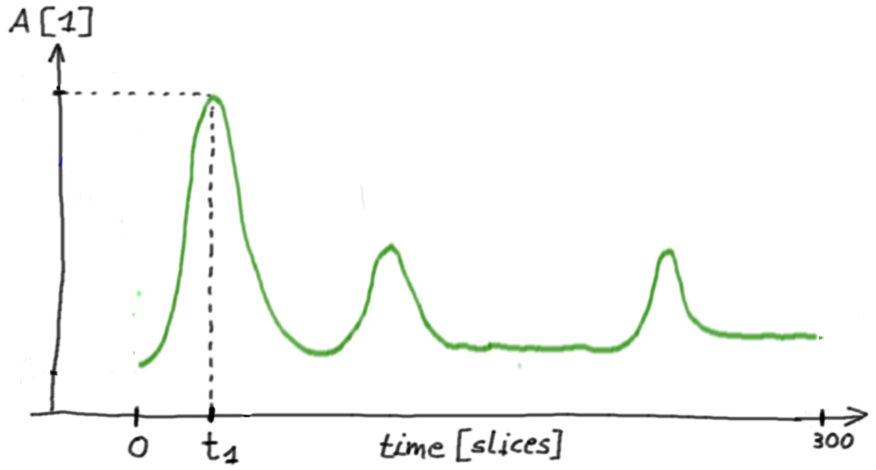


Very smooth, acts like a frequency filter optimized for the pulse template (Transmission spectrum of filter is the pulse spectrum)

Algorithm

```
->Initiate
   ->Template Pulse time lines
   ->Template Pulse integral
->Input time line
->**do {**
    ->convolve input time line
       and template pulse
    ->**find max response in**
       **convolution time line**
    ->store arrival time of
       max response
    ->subtract template pulse
       from input time line
    ->apply AC coupling to
       input time line
 **} while**(
       has max response in
       convolution time line
 )
->return arrival times
```

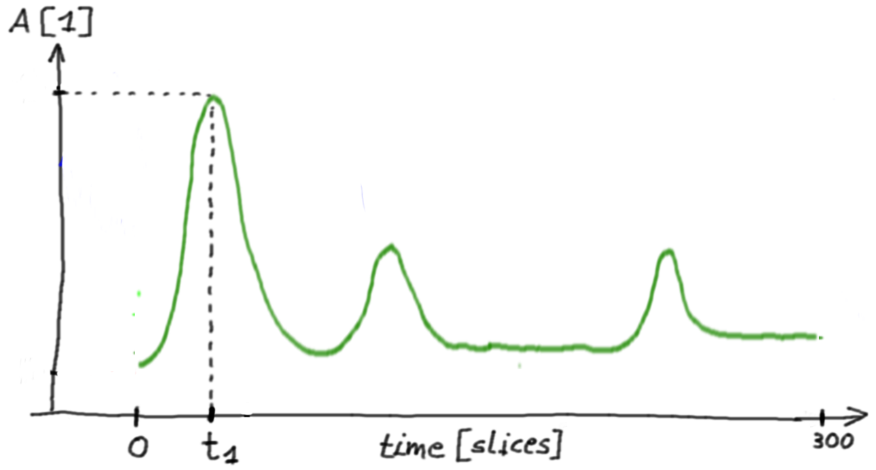
Easy to find the peaks, noise is suppressed



Algorithm

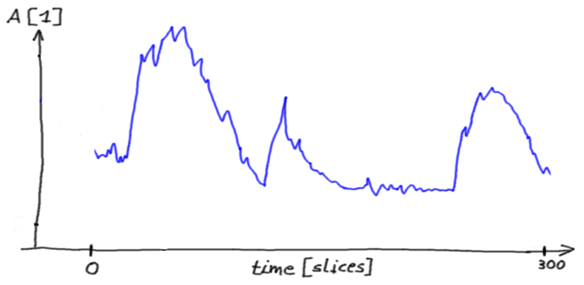
```
->Initiate
   ->Template Pulse time lines
   ->Template Pulse integral
->Input time line
->**do {**
    ->convolve input time line
       and template pulse
    ->find max response in
       convolution time line
    ->**store arrival time of
       max response**
    ->subtract template pulse
       from input time line
    ->apply AC coupling to
       input time line
 **} while**(
       has max response in
       convolution time line
 )
->return arrival times
```

Append arrival time t_1 of largest pulse to arrival time list

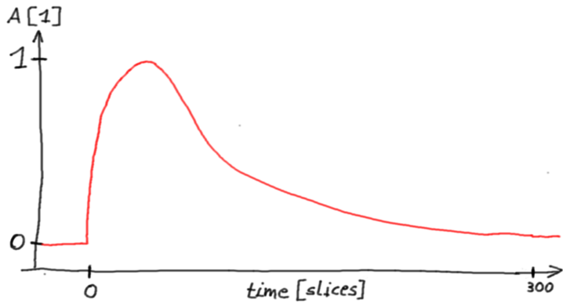


Algorithm

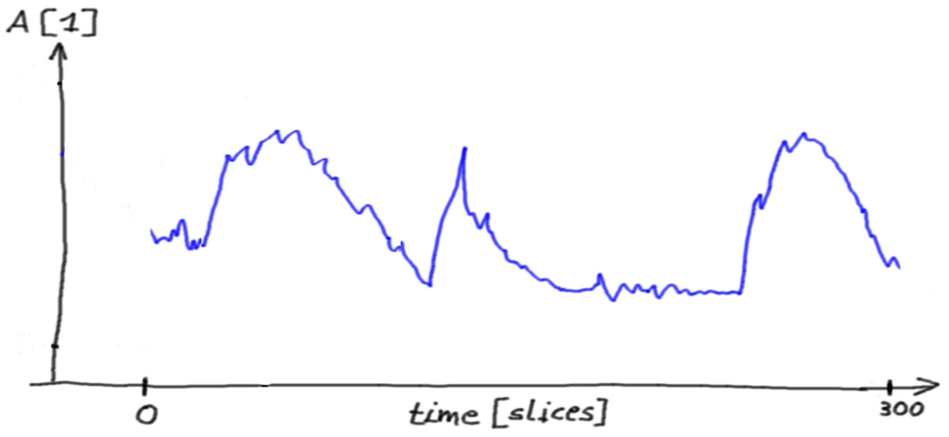
```
->Initiate
  ->Template Pulse time lines
  ->Template Pulse integral
->Input time line
->**do {**
   ->convolve input time line
     and template pulse
   ->find max response in
     convolution time line
   ->store arrival time of
     max response
   ->**subtract template pulse**
     **from input time line**
   ->apply AC coupling to
     input time line
 **} while**(
     has max response in
     convolution time line
 )
->return arrival times
```



-



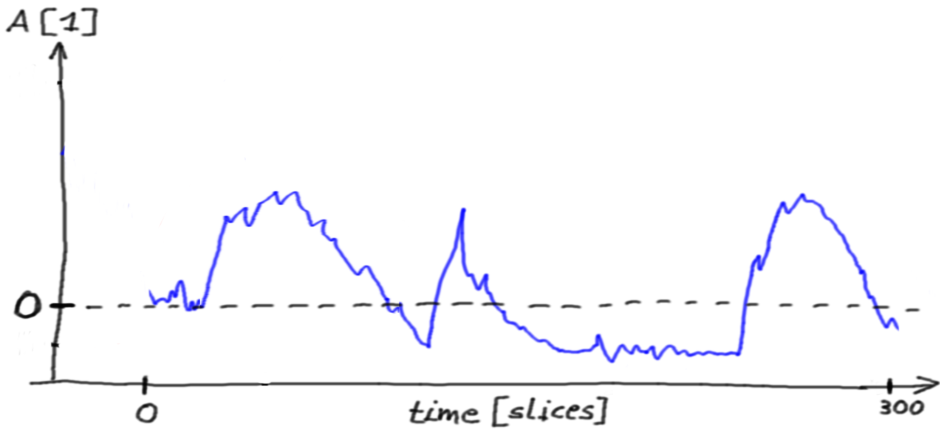
Subtract at time t_1



Algorithm

```
->Initiate
   ->Template Pulse time lines
   ->Template Pulse integral
->Input time line
->**do {**
    ->convolve input time line
       and template pulse
    ->find max response in
       convolution time line
    ->store arrival time of
       max response
    ->subtract template pulse
        from input time line
    ->**apply AC coupling to**
       **input time line**
  **} while**(
       has max response in
       convolution time line
  )
->return arrival times
```

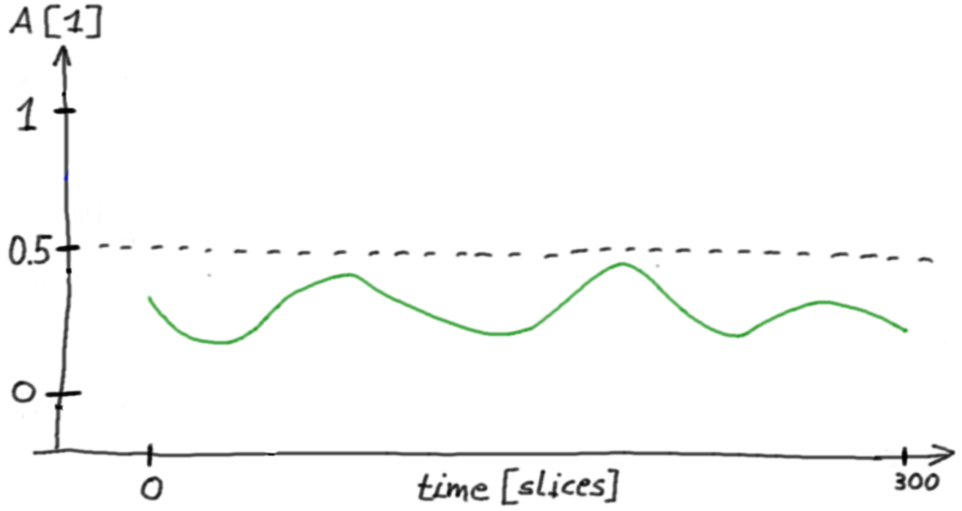
Applying the AC coupling of FACT's readout



Algorithm

```
->Initiate
   ->Template Pulse time lines
   ->Template Pulse integral
->Input time line
->do {
    ->convolve input time line
      and template pulse
    ->find max response in
      convolution time line
    ->store arrival time of
      max response
    ->subtract template pulse
      from input time line
    ->apply AC coupling to
      input time line
  } while(
      has max response in
      convolution time line
  )
->return arrival times
```

When the maximum in the convolution response is below a certain threshold (0.5 photon equivalents), the extraction is stopped

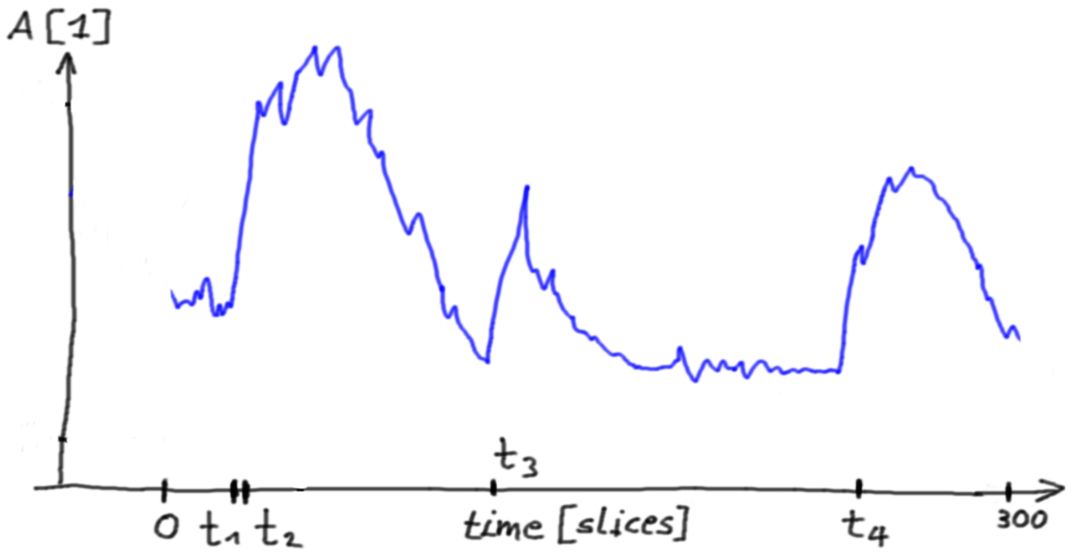


Algorithm

```
->Initiate
  ->Template Pulse time lines
  ->Template Pulse integral
->Input time line
->do {
   ->convolve input time line
     and template pulse
   ->find max response in
     convolution time line
   ->store arrival time of
     max response
   ->subtract template pulse
     from input time line
   ->apply AC coupling to
     input time line
 } while(
     has max response in
     convolution time line
 )
->return arrival times
```

Returns arrival time list

$$[\ t_1,\ t_2,\ t_3,\ t_4\ ]$$



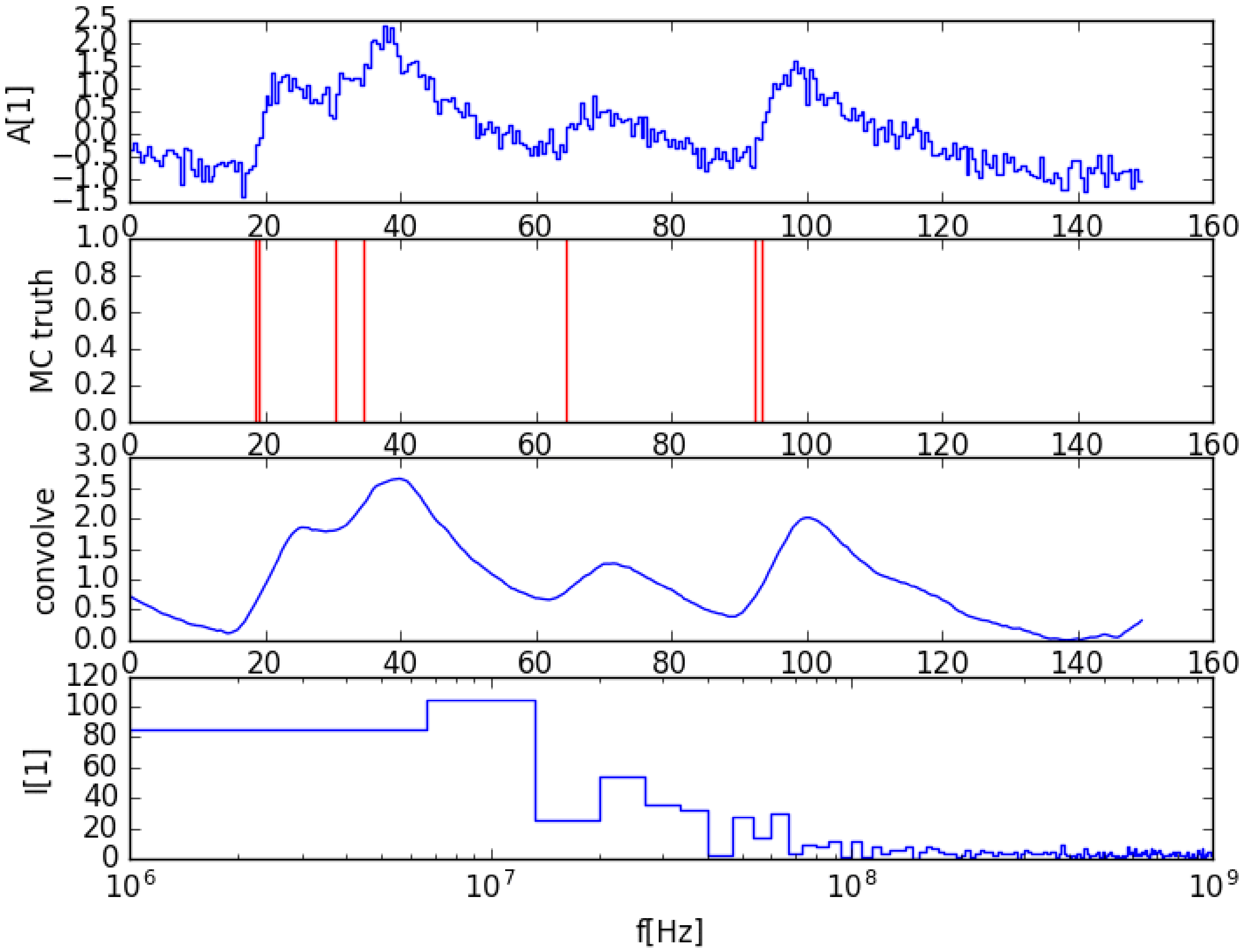
Implementation

Development Sandbox
https://github.com/fact-project/single_pulse_extractor
pure toy simulation with full simulation truth
(python)

Production Code in FACT-Tools
https://github.com/fact-project/fact-tools
no toy simulation, only FACT observations or FACT Monte Carlos
in Master branch, ready to use
(java)

Example

Found at:



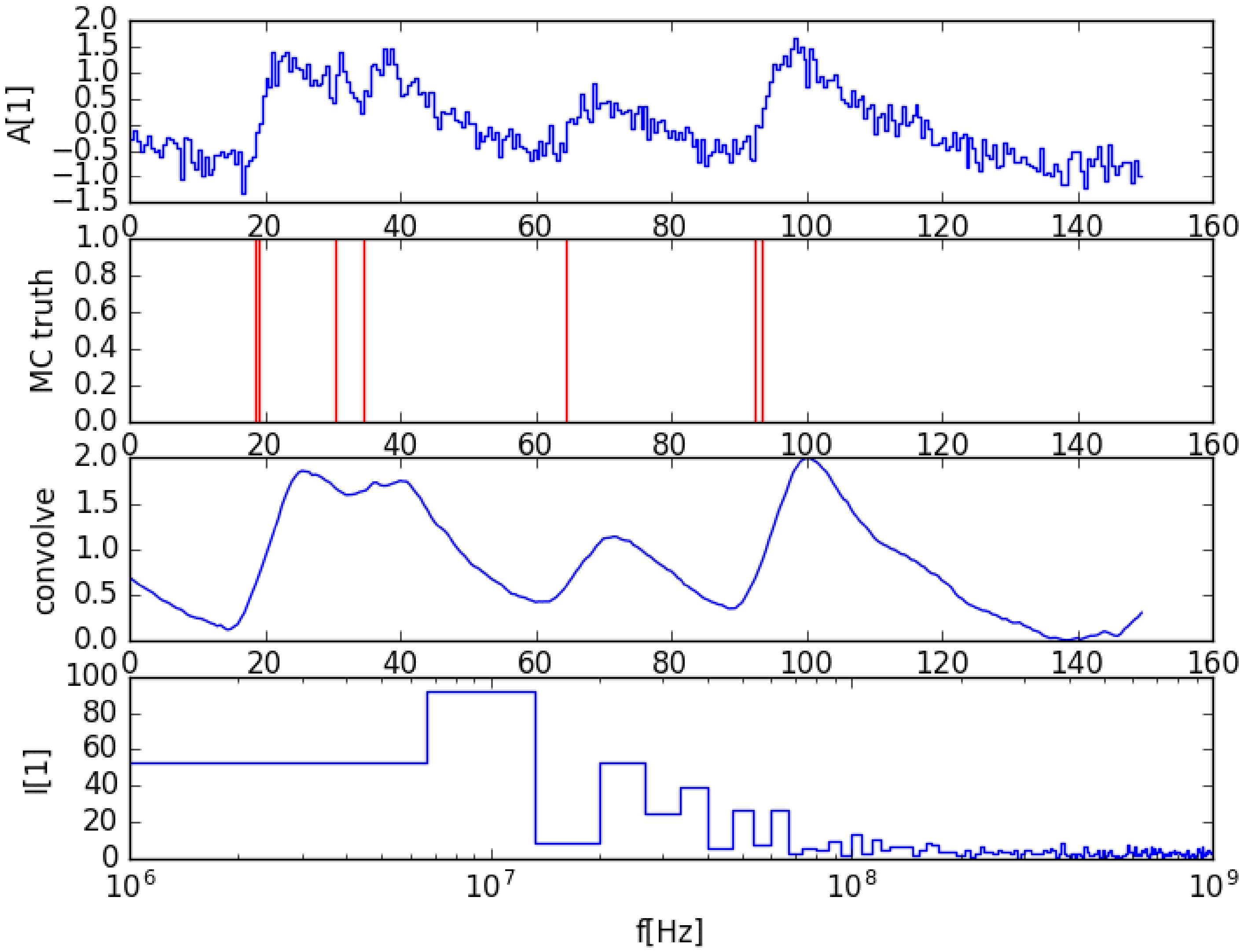
photons found: 0

photons injected: 7

Example

Found at:

31.5ns



photons found: 1

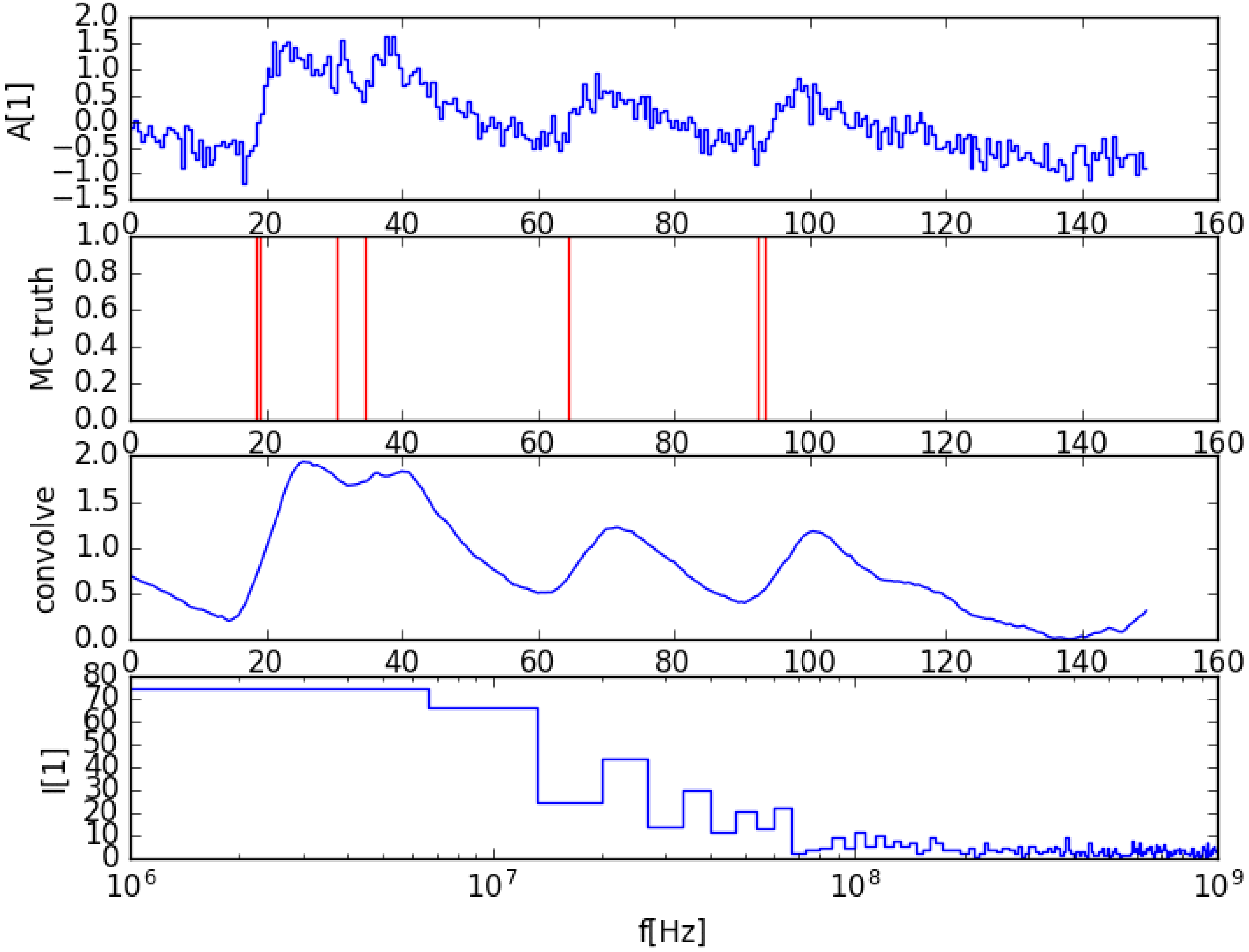
photons injected: 7

Example

```
Found at:

31.5ns

92.0ns
```



```
photons found: 2

photons injected: 7
```

Example

Found at:

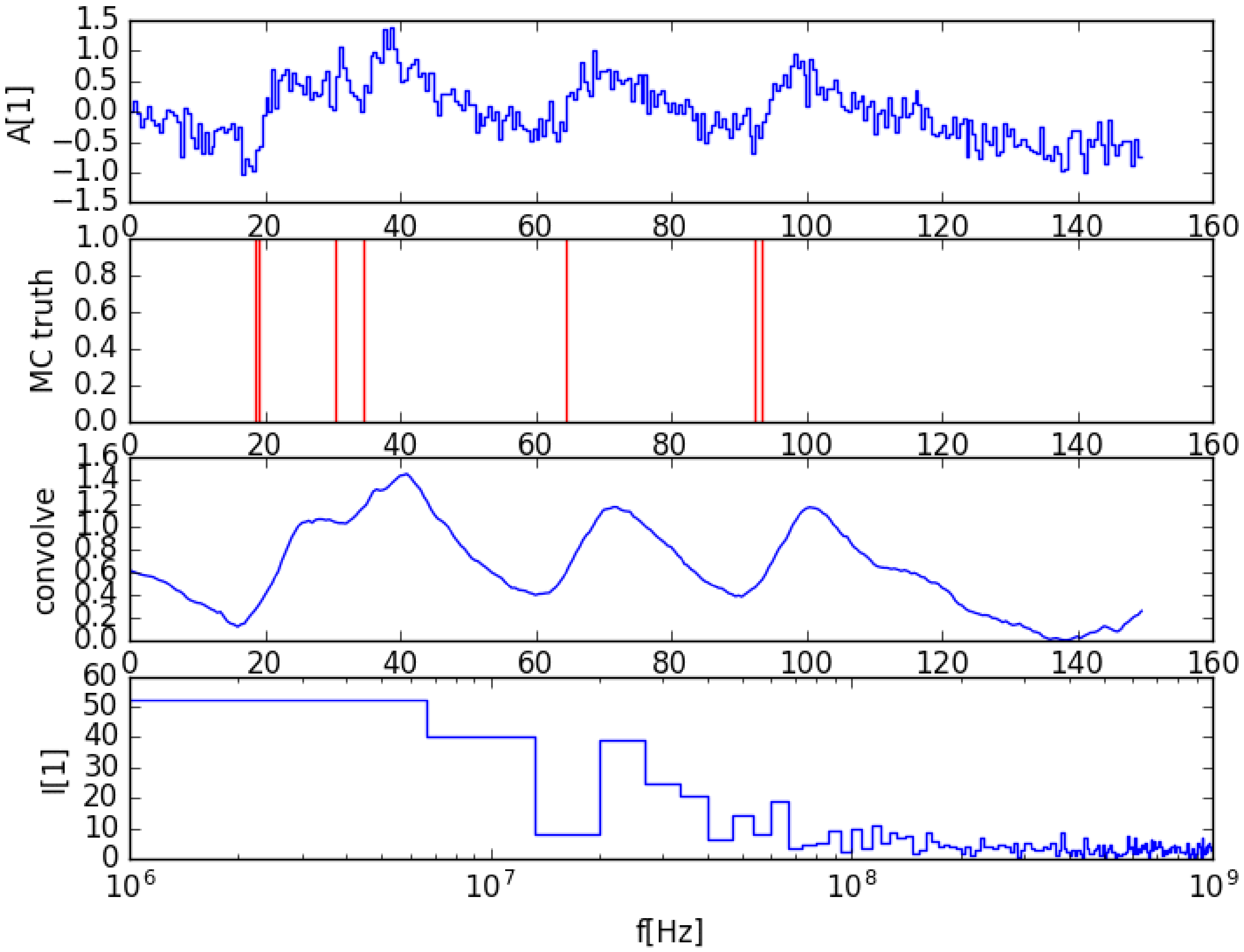
31.5ns

92.0ns

17.0ns

photons found: 3

photons injected: 7



Example

Found at:

31.5ns

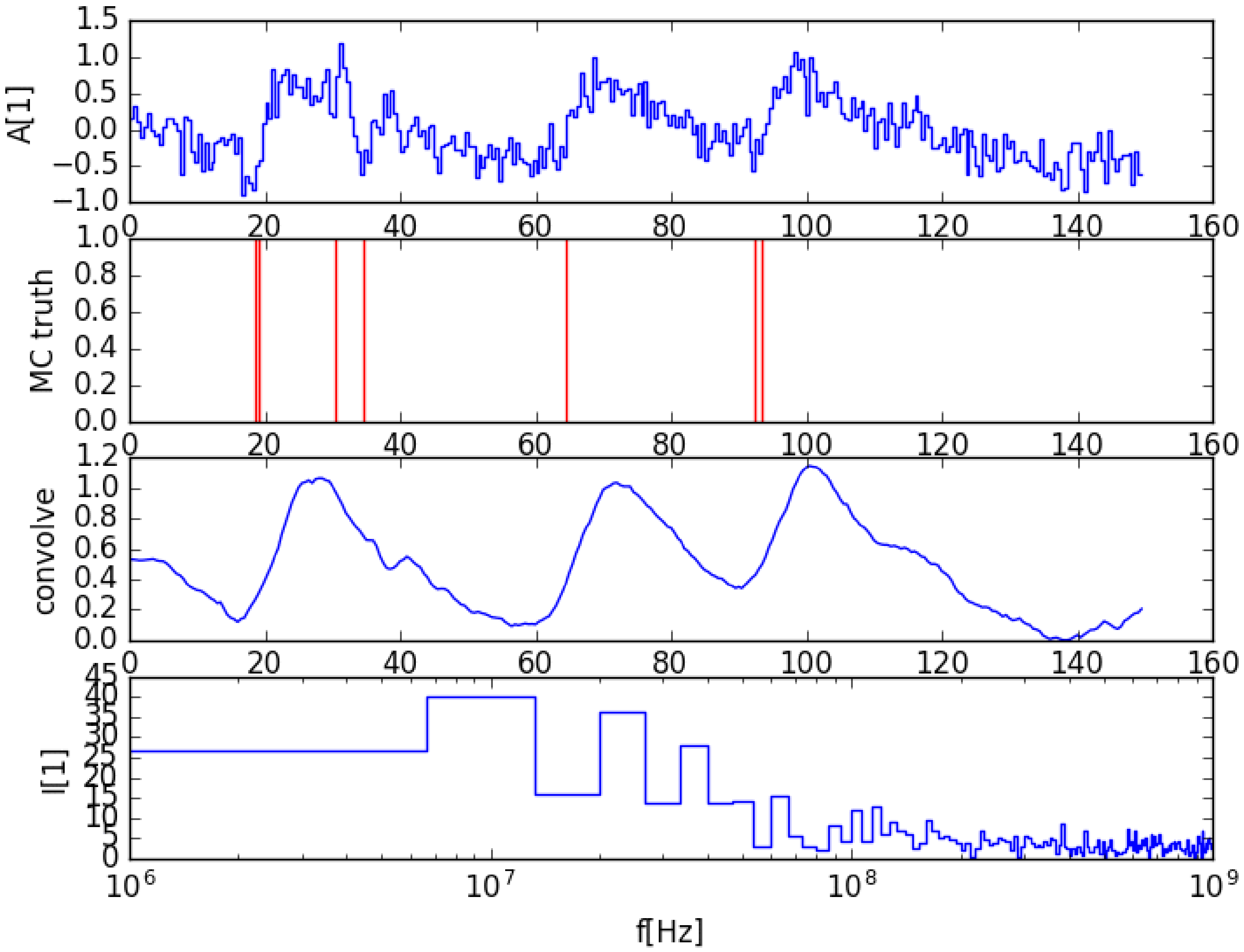
92.0ns

17.0ns

32.5ns

photons found: 4

photons injected: 7



Example

Found at:

31.5ns

92.0ns

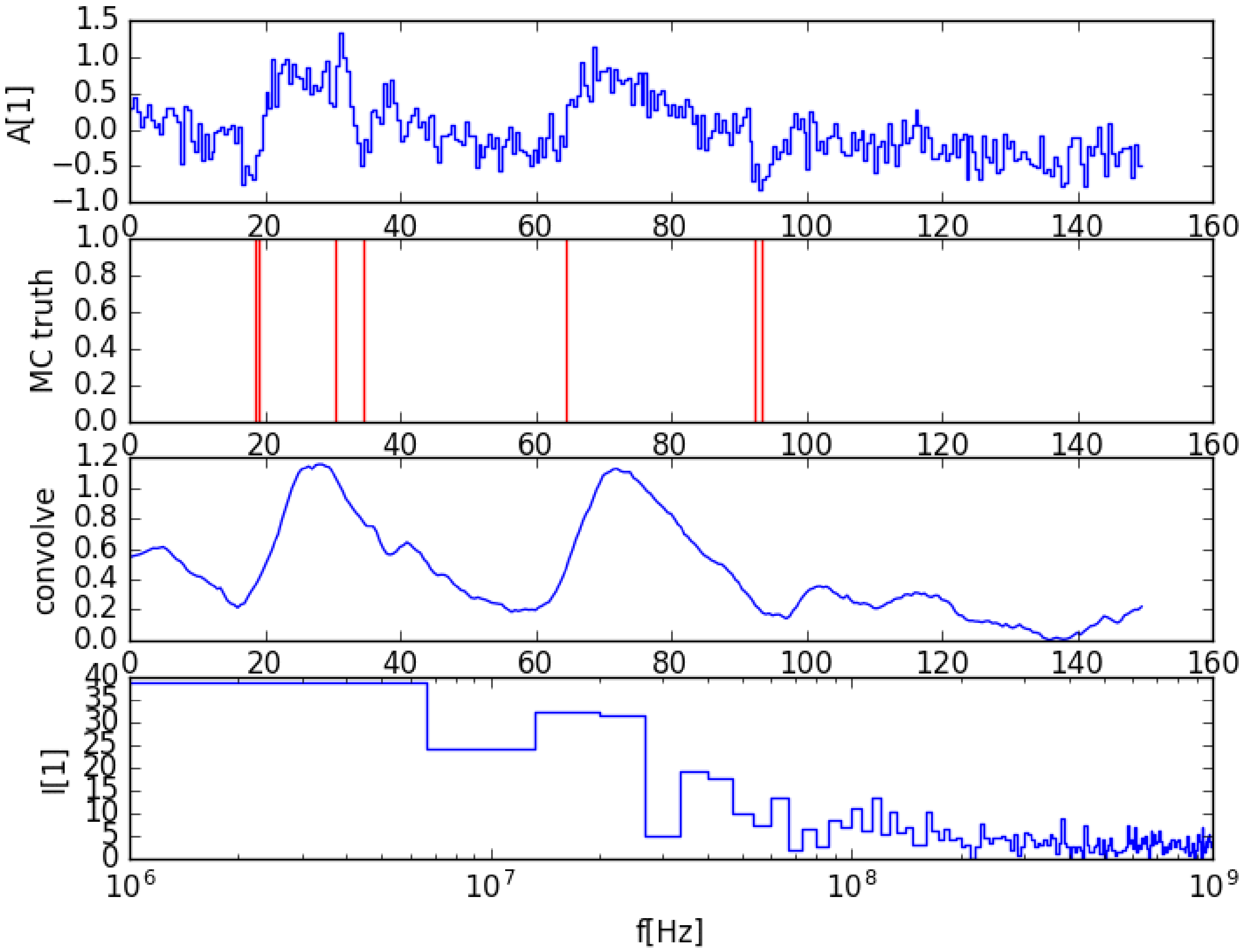
17.0ns

32.5ns

92.0ns

photons found: 5

photons injected: 7



Example

Found at:

31.5ns

92.0ns

17.0ns

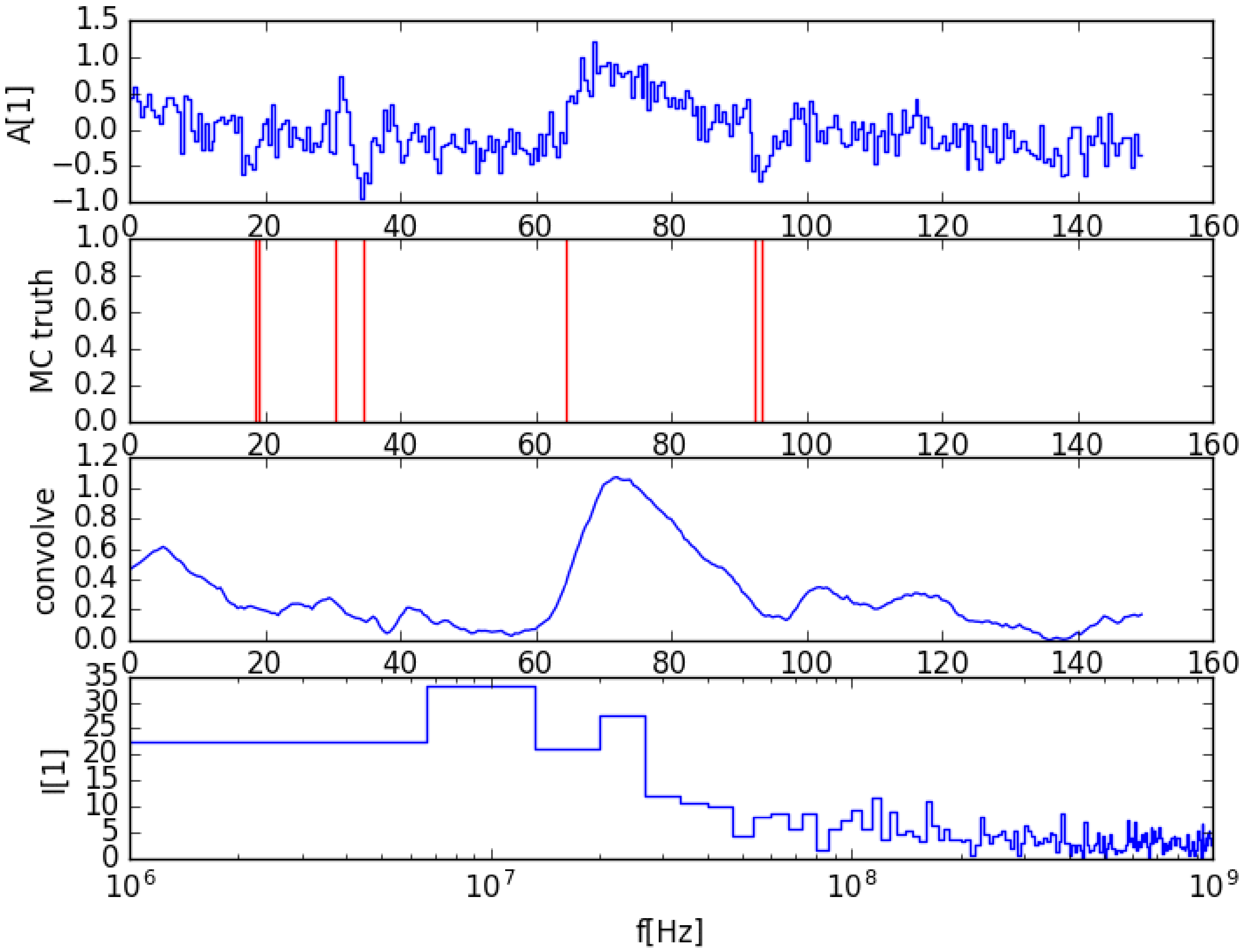
32.5ns

92.0ns

19.5ns

photons found: 6

photons injected: 7



Example

Found at:

31.5ns

92.0ns

17.0ns

32.5ns

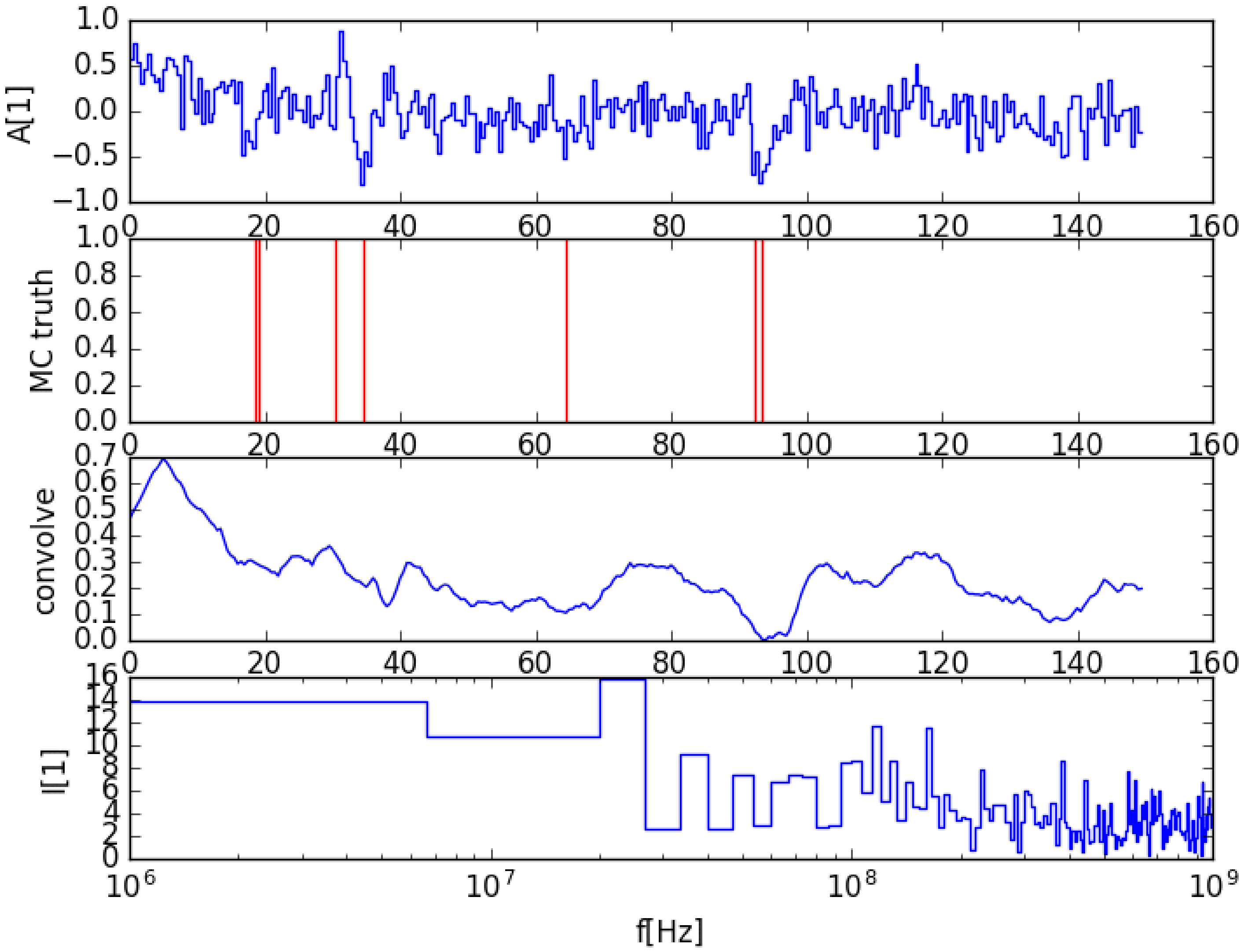
92.0ns

19.5ns

63.5ns

photons found: 7

photons injected: 7

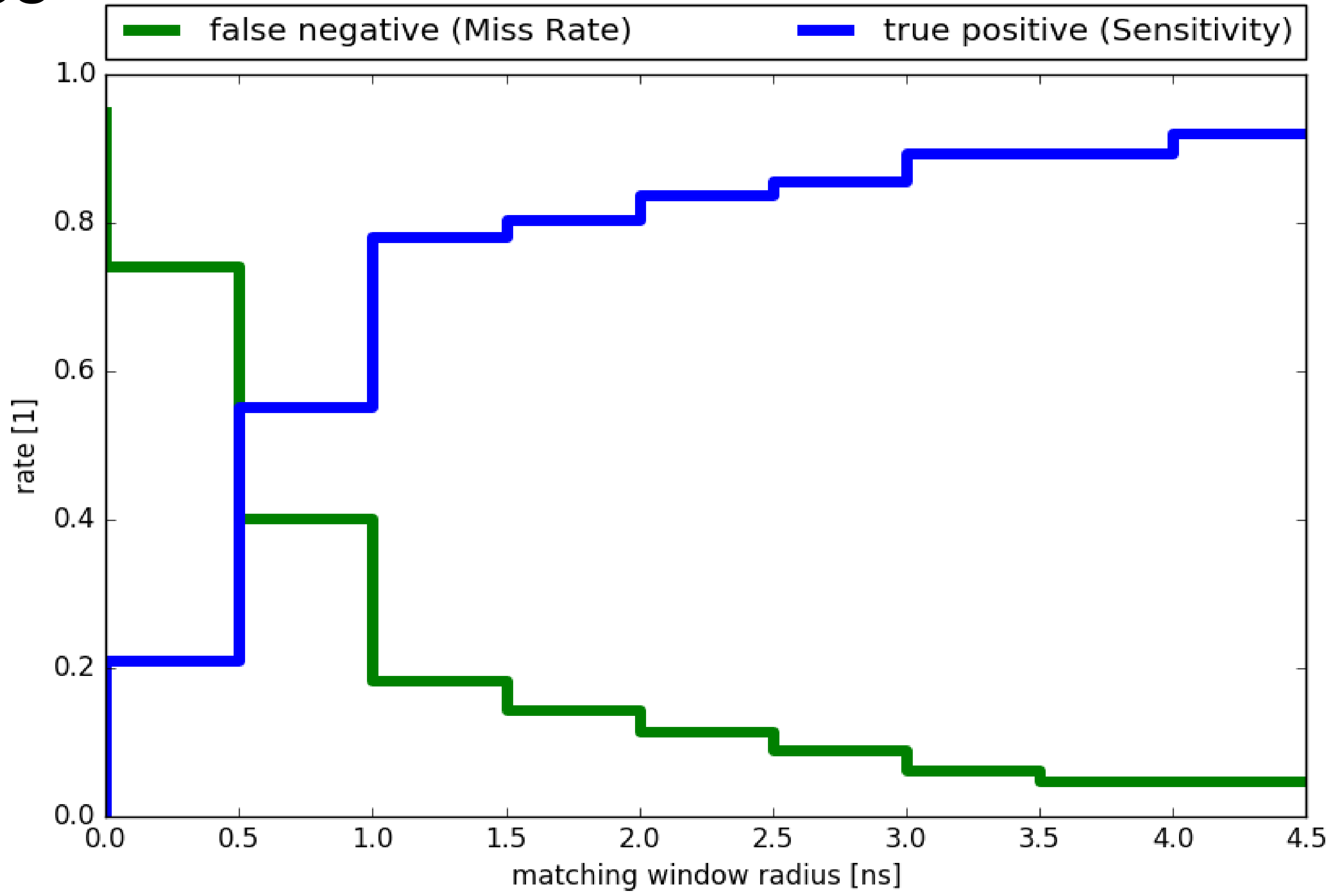


Performance

Sandbox toy MC
based on MC truth
(python)

Dark night
50MHz NSB
no Cherenkov
single photons

(1000 time lines/bin)



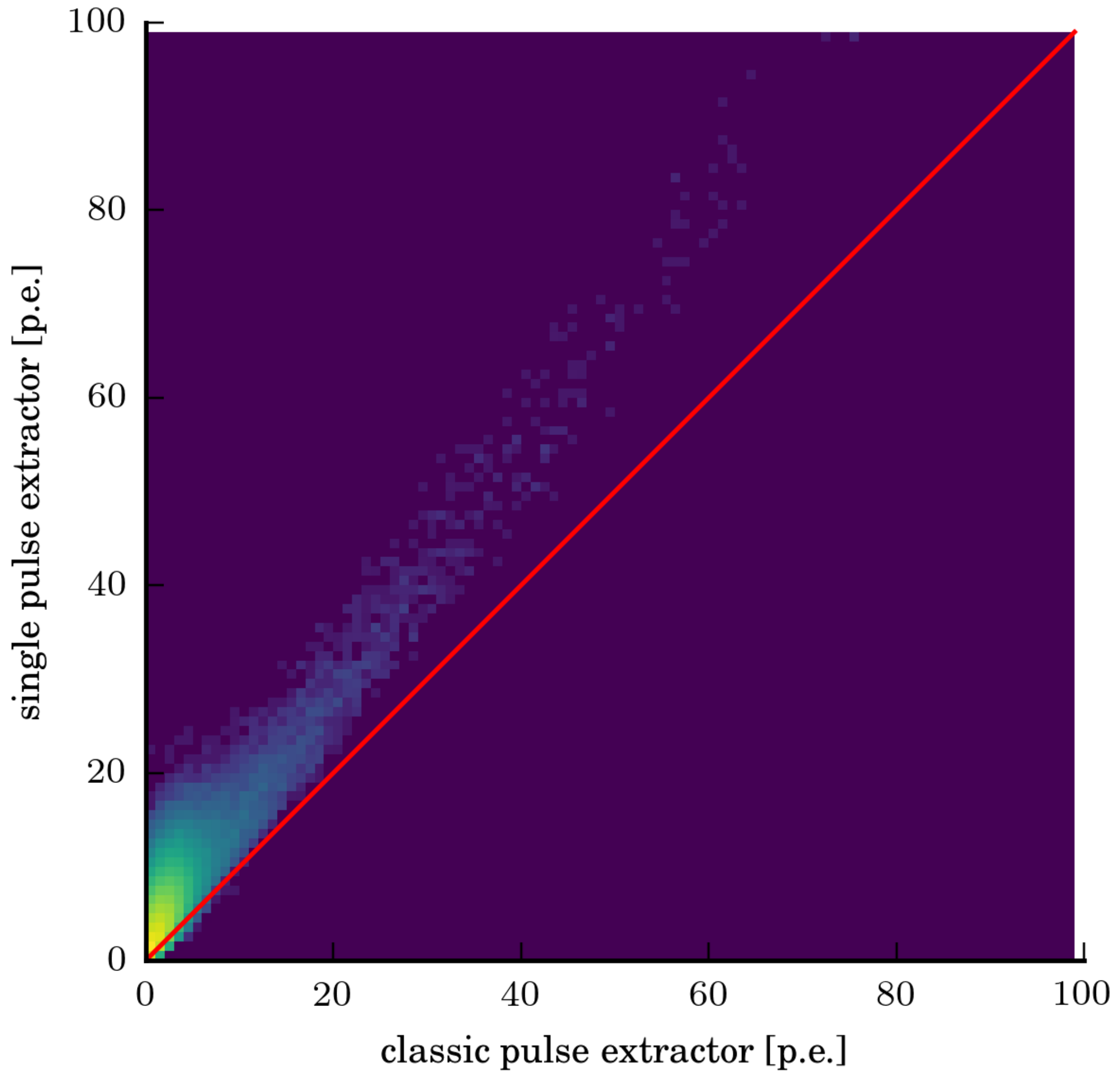
Compare

Single Pulse Extractor
fact-tools
production code
(java)

VS

Classic Photon Charge
fact-tools
production code
(java)

(316 MC events x 1440 time lines)



Integration into analysis

At the moment, all our image/video analysis is based on 2 float values / pixel (arrival time and photon equivalent)

```
arrival_time, photon_equvalent = timeAndCharge(pixel_time_line)
```

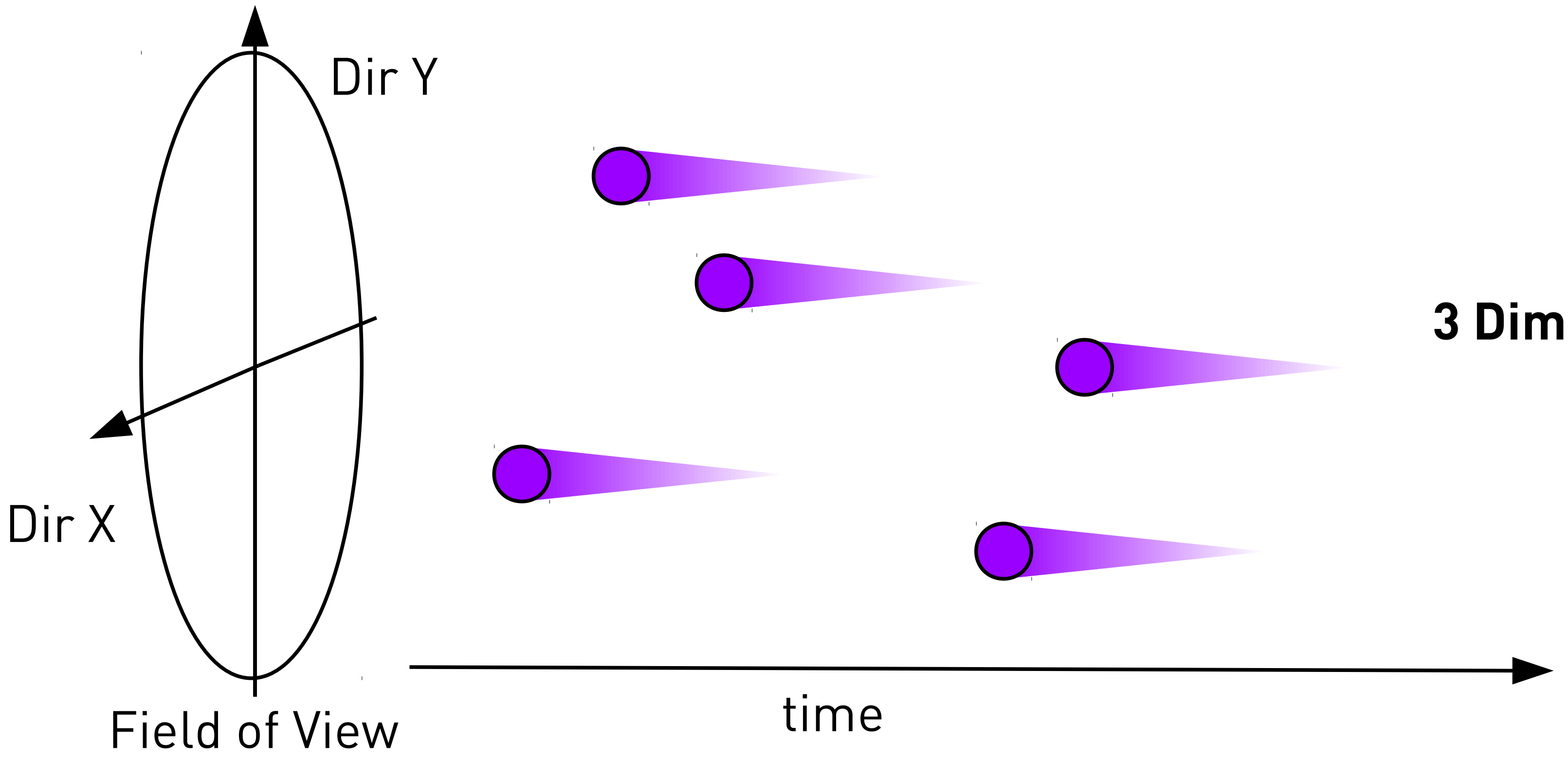
But the novel single pulse extractor returns a variable length float array of photon arrival times

```
arrival_times[] = singlePulseExtractor(pixel_time_line)
```

The concept is very different. Integration is not straight forward.

Photon Stream

We call the photon arrival time lists of all pixels the **photon stream**



3 Dimensions

Photon Stream

for each pixel:

Variable length array of the arrival slices for each photon

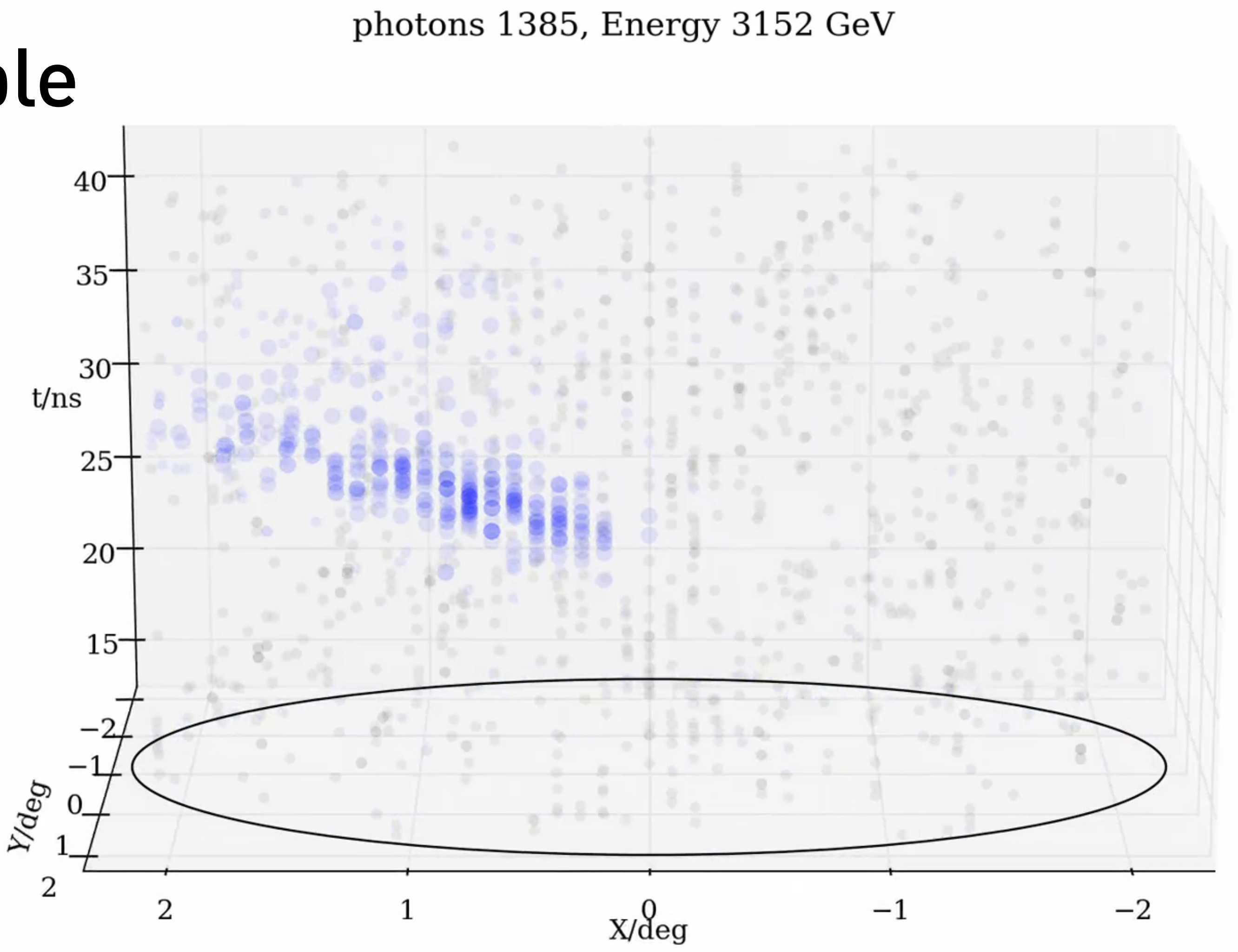
```
Pixel 0: [],
Pixel 1: [111, 88, 114, 88],
Pixel 2: [],
Pixel 3: [95, 223, 98],
Pixel 4: [217],
Pixel 5: [40, 40, 41, 41, 42, 43],
Pixel 6: [...
```

$\downarrow$

Only ~ 3.5kByte/event (typical)

Only photons. No noise.

Example



photons 1385, Energy 3152 GeV
t/ns
40
35
30
25
20
15
Y/deg
−2
−1
0
1
2
X/deg
2
1
0
−1
−2