S. Meroli et al.
http : // iopscience.iop.org/1748 - 0221/6/06/P06013
This paper discusses theoretical distribution and compares it to measurements in Si.

Figure 1: 100 MeV electrons in various thicknesses of Silicon:

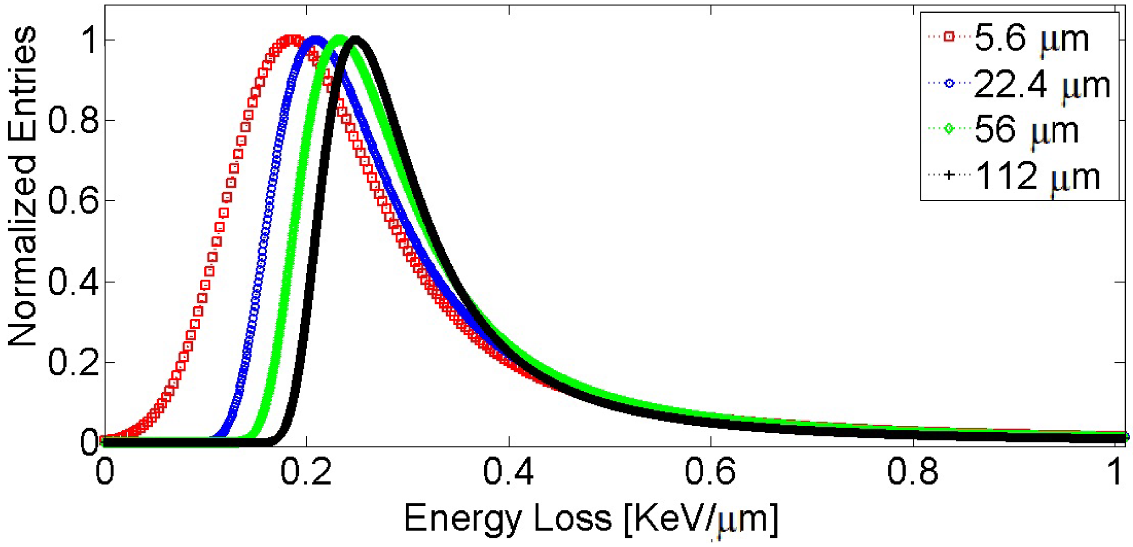
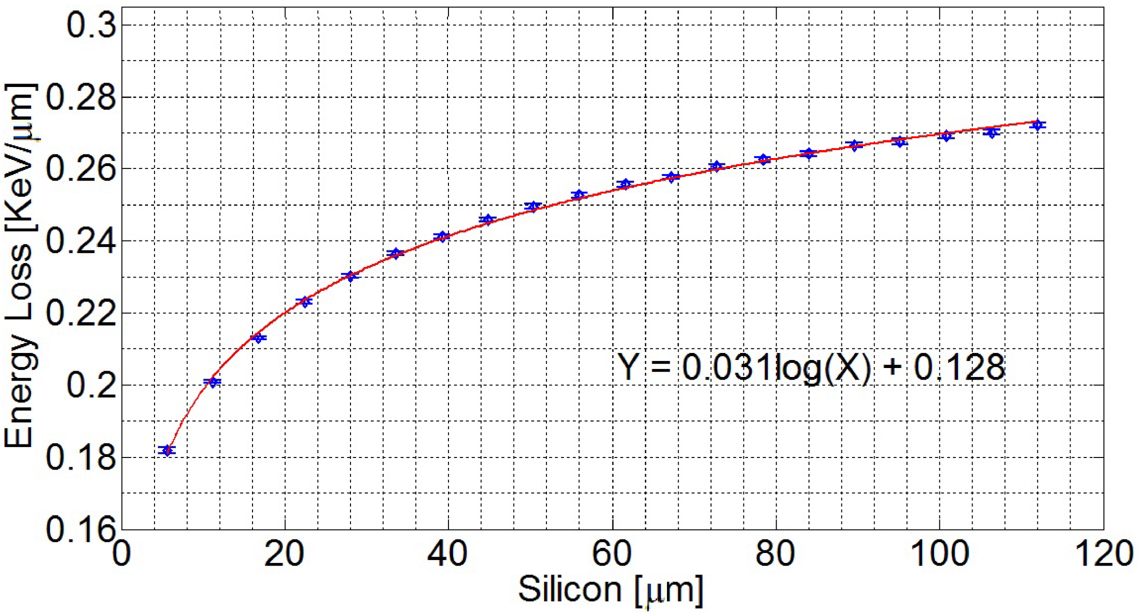
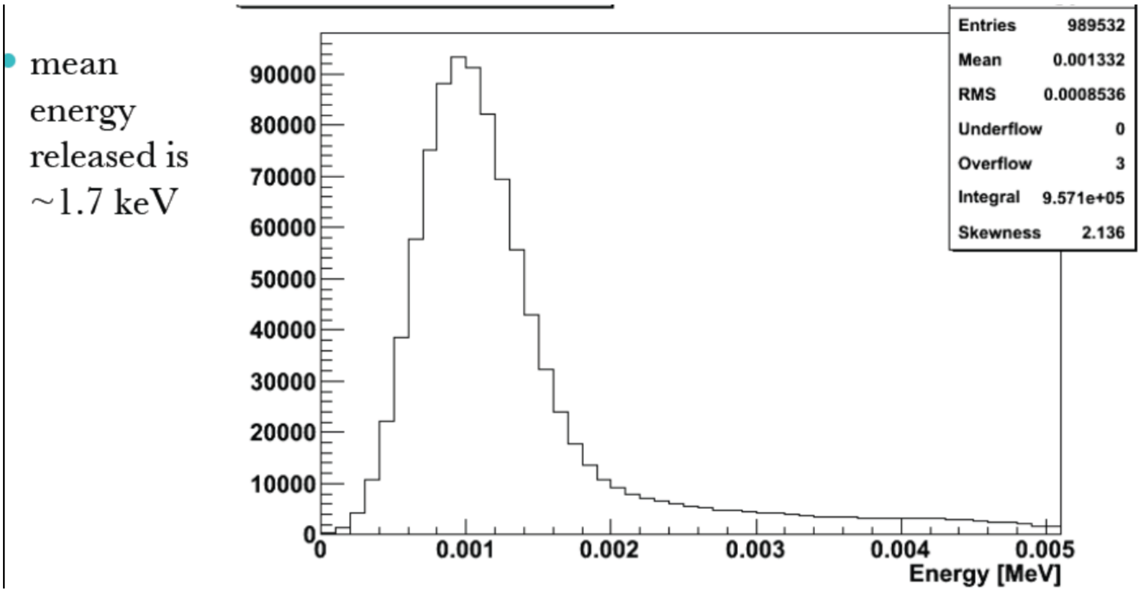


Fig. 2: Most Probable Energy Loss:



So most probable enrgy loss in 60 micron Si is 0.25 keV*60=15 keV
take 3.68 eV per e-h pair. Then most probable is 4,100 e-h in 60 microns.
Below use an approximate form (Landau Distribution without Gaussian Convolution:
ie-the probability density for value x in a Landau distribution is proportional to
$\int_0^\infty \sin(2\,t)\exp(-t\,(x-\mu)/\sigma - 2/\pi\,t\log(t))\,d\,t)$.
Since we are going to treat a series of 1 micron layers to evaluate time jitter due to clustering it ’s useful to
refer to the modelling done by M. Fiorini in NA62 presentation below. Note extrapolation of above curve would give ~.15 KeV/micron in this case
so it is hard to reconcile with the following distribution:

Fig. 3: Fiorini ṡ distribution for 1 micron Si.



- mean energy released is ~1.7 keV

Nevertheless, for the moment we are just after estimating effect of this skewed distribution on time-of-arrival jitter. We first test available *Mathematica* distributions to find something approximating Fig. 1.

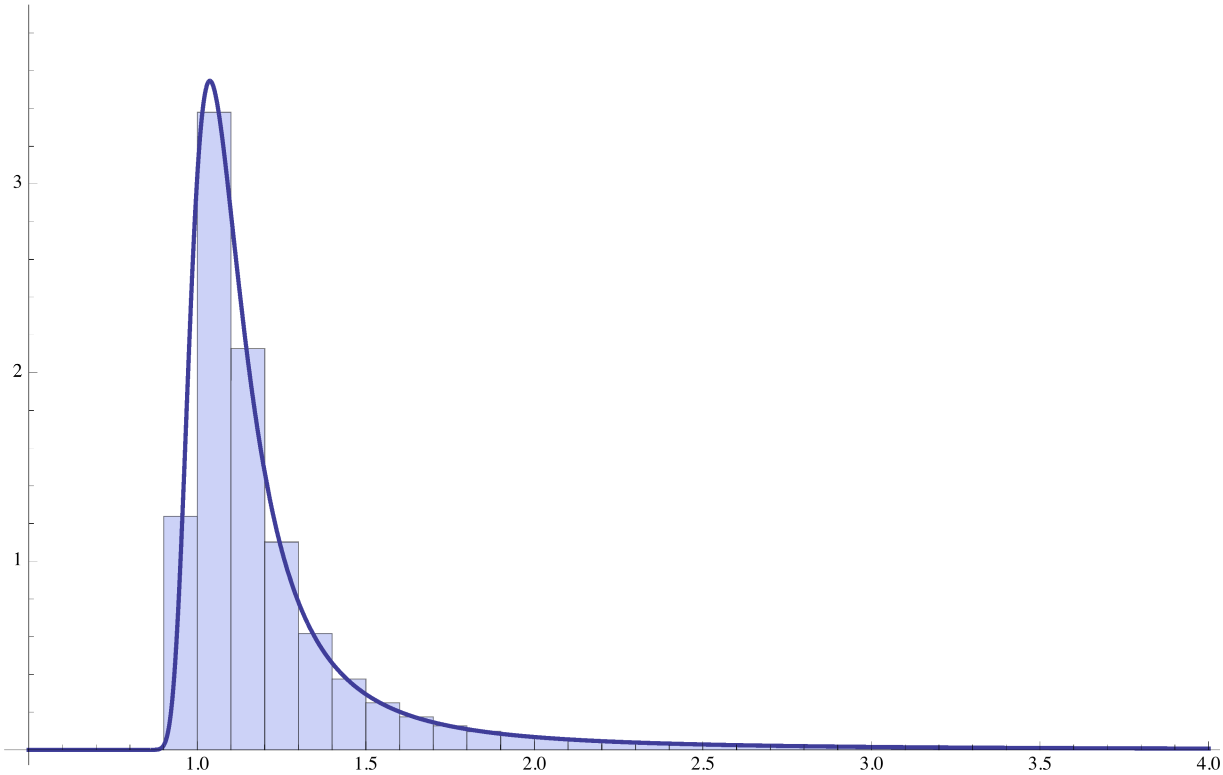
Generate Landau distributed Energy deposit:

```
In[340]:= σ = 0.08; μ = 1.2;
          data = RandomReal[LandauDistribution[μ, σ], 10^6];
          gL[Λ_] = PDF[LandauDistribution[μ, σ], Λ];
          NMaximize[gL[Λ], Λ];
          peak = Flatten[NMaximize[gL[Λ], Λ] /. Rule → List][[3]];
          ave = Mean[data];
```

```
Most Probable Energy Loss= 1.03702   and Mean Energy Loss= 1.76622
```

Compare its histogram to the PDF :

```
Show[
 Histogram[data, {0.5, 4, 0.1}, "PDF", AxesOrigin → {0.5, 0}],
 Plot[PDF[LandauDistribution[μ, σ], x],
  {x, .5, 4}, PlotRange → Full, PlotStyle → Thick, ImageSize → Large]]
```

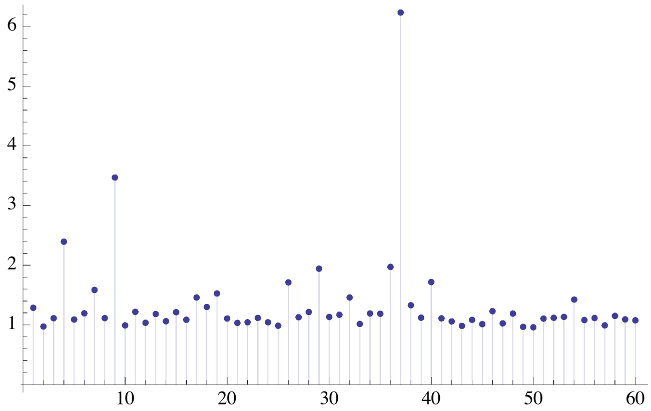


Although this LandauDistribution differs in detail from the curves in Fig. 1 (which are generated by the commonly used convolution of Landau with Gaussian) it gives roughly the dispersion between peak and mean that we are looking for.

Now make simple MonteCarlo to illustrate time jitter. The model is that dominant signal comes from amplification in the high field region of the APD, rather than Ramo' s Theorem treatment which is appropriate for the case of no Amplification.Then time structure is dominated from discontinuous arrival time of electrons into the amplification region moving at ~10^7 cm/sec -> ~1 micron/10 picosec. So, in the following, the multiplier for jitter, in terms of bins, is 10 picosec.
We treat the detector signal as coming from 60 bins arriving sequentially and so a typical even would look as follows:

```
edep = RandomReal[LandauDistribution[μ, σ], 60];
```

```
ListPlot[event, Filling → Axis, PlotRange → Full]
```



We define the mean time of arrival of electrons into the amplification region as :
< t >= ΣE (i)*t (i)/ΣE (i). So for uniform energy deposit, as in the case of UV pulse response the mean time of arrival is:

```
Sum[i, {i, 1, 60}] / 60
N[%]
```

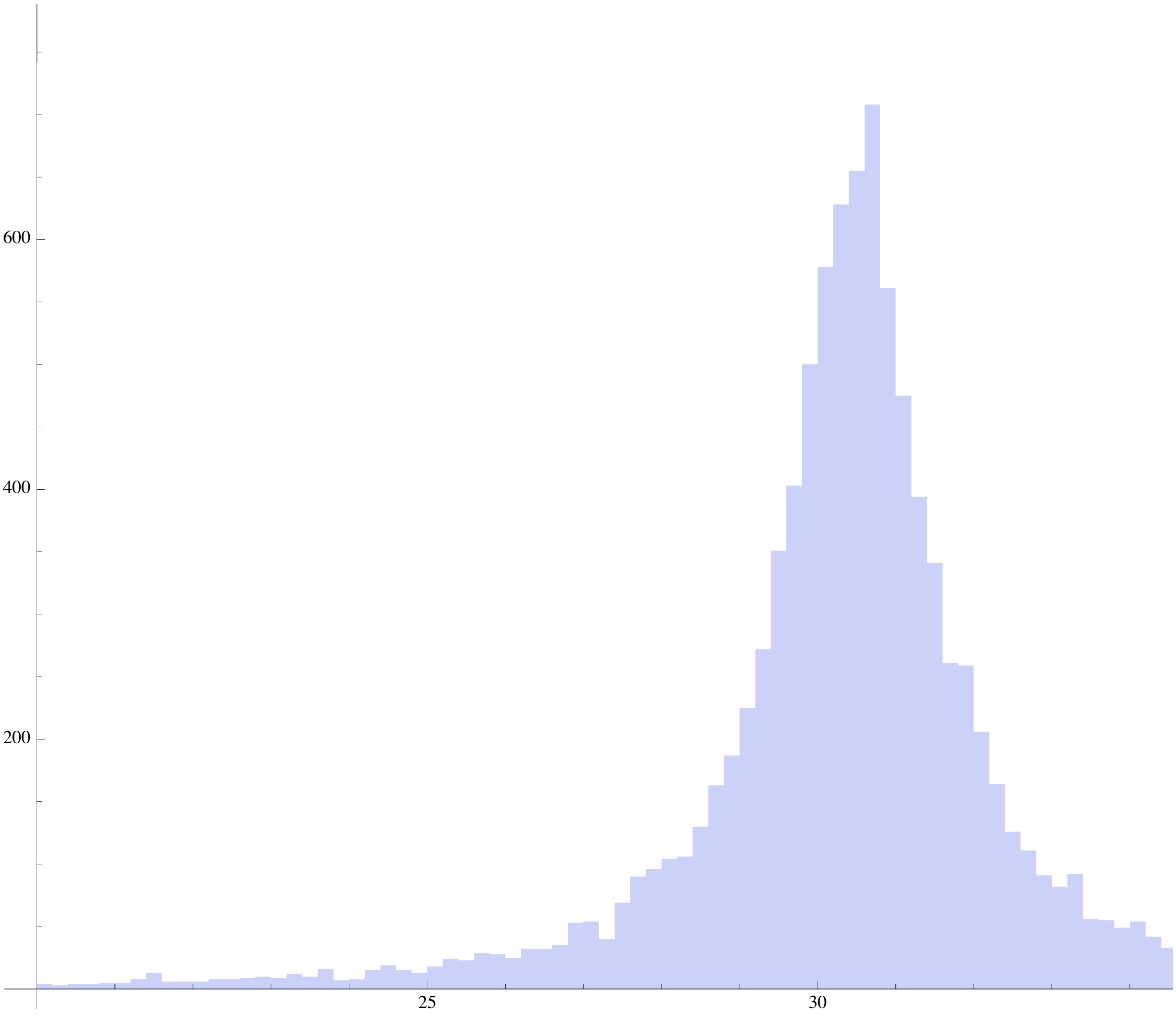
$$\frac{61}{2}$$

```
30.5
```

ie the average drift time = 1/2 the total, as expected.

Now plot the mean arrival time distribution for several randomly generated events.

```
time = Range[60]; meantime = ConstantArray[0, 10 000];
Do[
  edep = RandomReal[LandauDistribution[μ, σ], 60];
  meantime[[ievent]] = Sum[i * edep[[i]], {i, 1, 60}] / Sum[edep[[i]], {i, 1, 60}];
  , {ievent, 10 000}];
Histogram[meantime, {20, 40, .2}]
mt = Mean[meantime]
rmt = RootMeanSquare[meantime - mt]
```



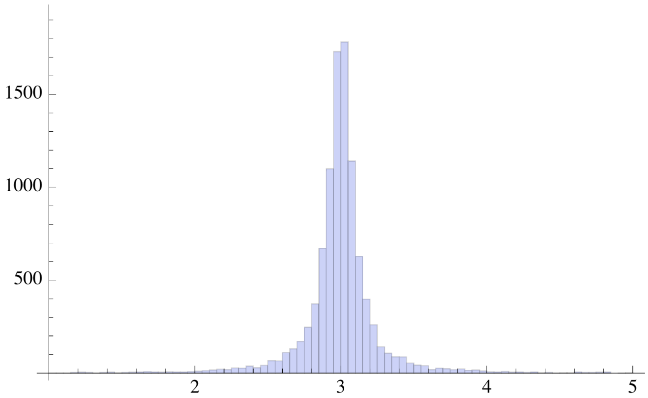
```
30.4901

3.34078
```

So rms of distribution for a 60 micron detector is 3 time units (30 picosec for a saturated electron velocity of 10 ps/micron). Note, however, that the FWHM is also about 3 units. So the RMS is really dominated by non - Gaussian tails.

Now repeat for a 5 micron thick depletion region (a la Hamamatsu).

```
time = Range[5]; meantime = ConstantArray[0, 10000];
Do[
  edep = RandomReal[LandauDistribution[μ, σ], 5];
  meantime[[ievent]] = Sum[i * edep[[i]], {i, 1, 5}] / Sum[edep[[i]], {i, 1, 5}];
  , {ievent, 10000}];
Histogram[meantime]
mt = Mean[meantime]
rmt = RootMeanSquare[meantime - mt]
```



```
2.99771

0.280478
```

So, sure enough, a thinner depletion region gives better timing - so long as we don' t add signal - to - noise issues in the optimization.