Monte Carlo PMT Simulation

This example follows notes posted online which you can find at the following url: http://superk.physics.sunysb.edu/~mcgrew/phy310/lectures/phy310-lecture-06-2007.pdf

Created using Maple 14.01

Jake Bobowski

```
> restart;
  with(stats):
  with(plots):
  with(Statistics):
  with(StringTools):
  FormatTime("%m−%d−%Y, %H:%M");
```

"03-24-2013, 23:39" **(1)**

In all Monte Carlo simulations it is necessary to generate random or pseudo-random numbers. The following statement will generate a random number drawn from a uniform distribution between 0 and 1.

```
> X := x→stats[random, uniform[0, 1]](1):
  X();
```

0.3957188605 **(2)**

First, suppose N photons hit the photocathode. Determine the number of photoelectrons that are generated. Assume that the incoming light is 400 nm and that the quantum efficiency of the photocathode is 0.23. (Following http://superk.physics.sunysb.edu/~mcgrew/phy310/lectures/phy310-lecture-06-2007.pdf)

```
> N := 2363:
  QE := 0.23:
  pe := 0:
  for i from 1 to N do:
  if X() < QE then:
  pe := pe + 1:
  end if:
  end do:
  pe;
```

537 **(3)**

When an electron with energy E hits a Dynode the average number of secondary electrons liberated is αE (i.e. number of secondary electrons is proportional to the energy of the incoming electron). Since we are counting electrons, the distribution of liberated electrons follows the Poisson distribution. As an example, below we generate random integers drawn from a Poisson parent distribtuion with mean 2000.

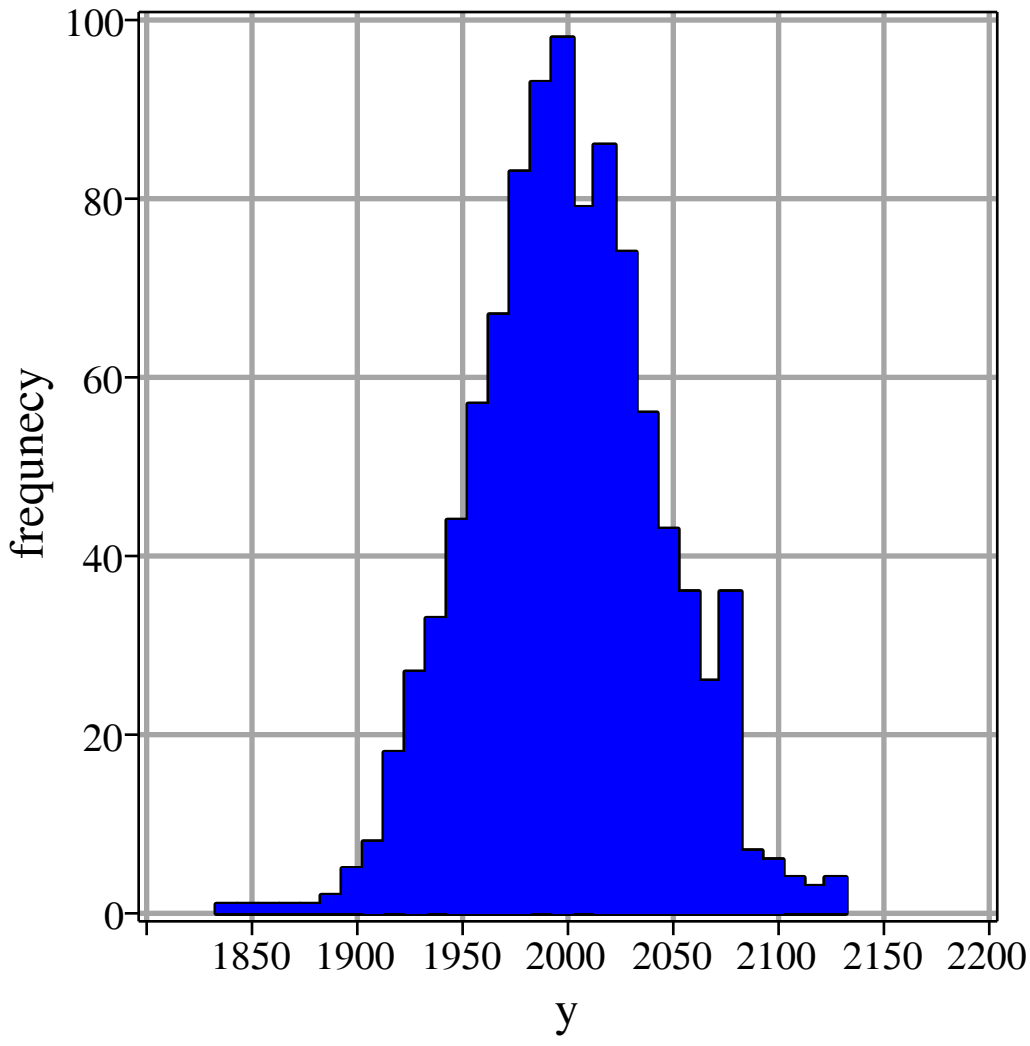
```
> Y := y→stats[random, poisson[μ]](1):
> μ := 2000:
  Y();
  Ydist := NULL:
```

```
for i from 1 to 1000 do:
Ydist := Ydist, Y( ) :
end do:
Ydist := [Ydist] :
Mean(Ydist);
(StandardDeviation(Ydist))^2;
Histogram(Ydist, frequencyscale = absolute, axes = boxed, view = [1800 .. 2200, 0 .. 100],
    labels = [typeset("y"), typeset("frequnecy")], labeldirections = ["horizontal", "vertical"],
    symbol = circle, symbolsize = 20, thickness = 2, tickmarks = [8, 8], colour = blue , axesfont
    = [Times, 12] , labelfont = [Times, 14], axis = [gridlines = [thickness = 2]]);
```

2060

1999.589000

1968.90898798799



A photomultiplier tube (PMT) consists of a photocathode followed by a series of dynodes maintained a different electric potentials and then finally an anode. When a single photon is incident on the photocathode it either produces a photoelectron or it doesn't. The probability that it produces a photoelectron is determined by the quantum efficiency *QE* of the photocathode. For this exercise we will assume that *QE* = 0.23. If a photoelectron is produced, it is accelerated towards the first dynode by means of a potential difference. We assume that all electrons, whether produced at the photocathode or

one of the dynodes, start with zero kinetic energy. Therefore, the energy an electron gains is simply its charge times the potential difference between its starting and final positions. When an electron collides with a dynode, secondary electrons are produced. The average number of secondary electrons produced is proportional to the energy of the incoming electron. In this problem, we assume that an electron accelerated through 20 V will, on average, produce one secondary electron upon colliding with the dynode. Because we are "counting" electrons, the distribution of secondary electrons generated will follow a Poisson distribution. Note that the red text are comments and are not part of the Maple inputs. Comments are started using #.

```
> QE := 0.23 : # Set the quantum effieicency of the photocathode #
  maxi := 10e3 : # Set the number of Monte Carlo iterations #
  dynode := [0, 150, 300, 450, 600, 750, 850] : # Set the number of dynodes and their voltages #
  dist := NULL :
      # The dist list will keep track of how many electrons where detected at the PMT anode for
      trials in which there was at least on photoelectron generated at the photocathode #
  FormatTime("%M:%S");
      # Print the time (minutes and seconds) that the simulation was started #
  for i from 1 to maxi do:
      # This loop runs the Monte Carlo simulation maxi times (10,000 in this case) #
    if X( ) < QE then:
      # Generate a rondom number between [0,1]. Compare X( ) to the photocathode quantum
      efficiency #
       electrons := 1 : # If true, then a photoelectron is produced. #
       for j from 1 to nops(dynode) − 1 do:
      # This loop will examine the secondary electrons produced at each of the dynodes #
          μ := electrons· (dynode[j + 1] - dynode[j]) / 20 :
      # Mean no. of electrons generated at a dynode determined from the no. of incoming
      electrons and the energy of the incoming electrons. One electron accelerated through 20 V
      will on average produce one electron #
          electrons := electrons + Y( );
      # The number of electrons leaving the dynode equal to the no. of incomming electrons plus
      the number generated determined by the Poisson distributed random number #
       end do: # end the dynode loop #
    dist := dist, electrons :
      # Add the total number of electrons detected at the anode for this trial to the dist list #
    end if: # end if #
  end do: # end the 1 to maxi loop #
  FormatTime("%M:%S");
      # Print the time (minutes and seconds) that the simulation completed #
  dist := [dist] :
      # build the final dist list -- just putting square brackets around all of the numbers #
  evalf(nops(dist) / maxi);
      # Determines what fraction of the maxi trials actually produced electrons at the PMT
      anode #
  Mean(dist); # Calculate the mean of the distribution #
  Histogram(dist, frequencyscale = absolute, axes = boxed, view = [0 .. 2e6, 0 .. 500], labels
      = [typeset("electrons"), typeset("distribution")], labeldirections = ["horizontal",
      "vertical"], symbol = circle, symbolsize = 20, thickness = 2, tickmarks = [8, 8], colour
      = blue , axesfont = [Times, 12] , labelfont = [Times, 14], axis = [gridlines = [thickness
      = 2]]);
```

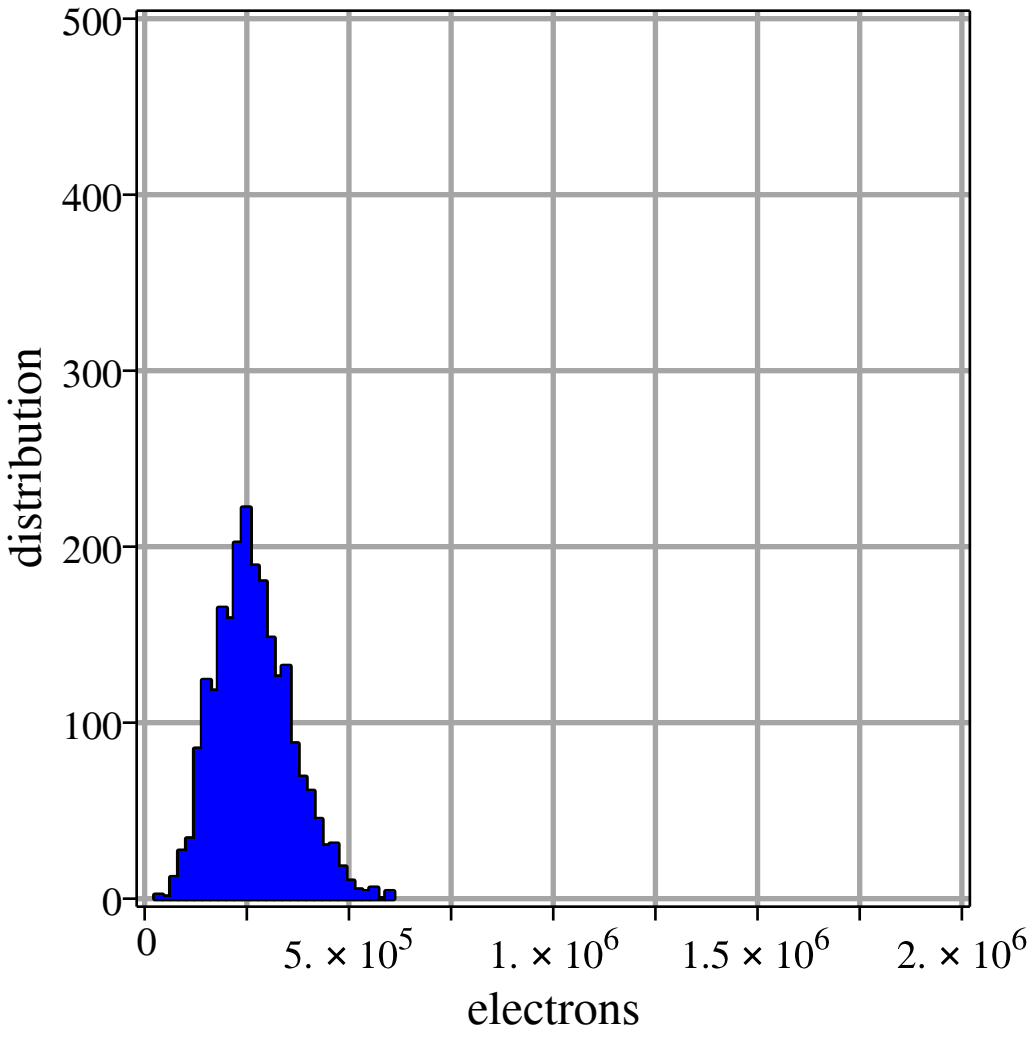
```
    # Plot a histogram of the distribution of the number of electrons collected at the anode #
binData := TallyInto(dist, [0..2e6], bins=100):
    # Use TalleyInto to build customized bins from the dist list #
 counts := [seq(rhs(binData[i]), i=1..nops(binData))]:
    # Extract the bin counts generated by TallyInto #
 centres := [seq(1e4+2e4·i, i=0..nops(binData)−1)]: # Make a list of the bin centres #
logplot(centres, counts, view=[0..2e6, 0.5..500], style=point, axes=boxed, labels
    =[typeset("electrons"), typeset("distribution")], labeldirections=["horizontal",
    "vertical"], symbol=circle, symbolsize=15, thickness=2, tickmarks=[8, 8], colour
    =blue, axesfont=[Times, 12], labelfont=[Times, 14], axis=[gridlines=[thickness
    =2]]);  # Make a semi-log plot of the distribution histogram #
```

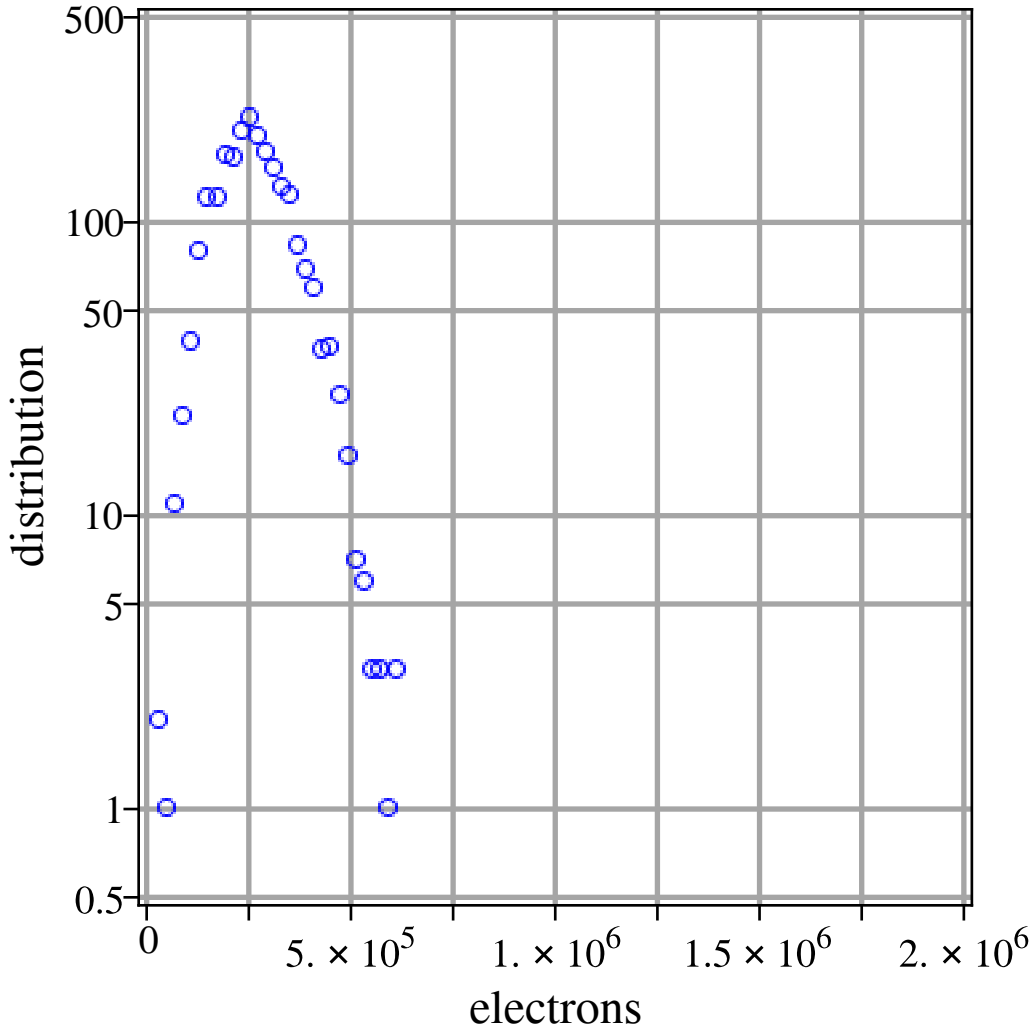
"39:49"

"48:04"

0.2297000000

$2.668587575\ 10^{5}$





This next block of code is very similar to the previous. This time, however, we imagine that there are 8 photons incident on the photocathode instead of one. Now there can be 0, 1, 2, ..., or 8 photoelectrons generated and then accelerated towards the first dynode. How does the distribution of electrons arriving at the anode change? We only require a relatively minor change to the code to study this problem. Plot the histograms using the same scale and the same binwidths to make comparisons with the previous results easy.

```
> QE := 0.23 : # Set the quantum effieicency of the photocathode #
  maxi := 10e3 : # Set the number of Monte Carlo iterations #
  dynode := [0, 150, 300, 450, 600, 750, 850] : # Set the number of dynodes and their voltages #
  dist := NULL :
      # The dist list will keep track of how many electrons where detected at the PMT anode for
      trials in which there was at least on photoelectron generated at the photocathode #
  numPhotons := 8 : # Set the number of photons incident on the photocathode #
  FormatTime("%M:%S");
      # Print the time (minutes and seconds) that the simulation was started #
  for i from 1 to maxi do:
      # This loop runs the Monte Carlo simulation maxi times (10,000 in this case) #
    pe := 0 :
      # Set the initial number of photoelectrons generated at the photocathode to be zero #
    for k from 1 to numPhotons do:
```

```
    # This loop individually steps through each of the photons incident on the photocathode #
      if X( ) < QE then:
    # Generate a rondom number between [0,1].  Compare X( ) to the photocathode quantum
    efficiency #
        pe := pe + 1 : # If true, then a photoelectron is produced.  Increment pe by 1 #
      end if : # end if #
    end do:  # end the incoming photon loop #
    if pe > 0 then:
    # If tbere are photoelectrons, then do something.  If there are no photoelectrons then do
    nothing #
      electrons := pe :
    # Set the initial number of electrons in the PMT to be equal to the number of photoelectrons
    generated at the photocathode #
      for j from 1 to nops(dynode) - 1 do:
    # This loop will examine the secondary electrons produced at each of the dynodes #
        μ := electrons· (dynode[j + 1] - dynode[j]) / 20 :
    # Mean no. of electrons generated at a dynode determined from the no. of incoming
    electrons and the energy of the incoming electrons.  One electron accelerated through 20 V
    will on average produce one electron #
        electrons := electrons + Y( );
    # The number of electrons leaving the dynode equal to the no. of incomming electrons plus
    the number generated determined by the Poisson distributed random number #
      end do:  # end the dynode loop #
      dist := dist, electrons :
    # Add the total number of electrons detected at the anode for this trial to the dist list #
    end if: # end if #
  end do:  # end the 1 to maxi loop #
FormatTime("%M:%S");
    # Print the time (minutes and seconds) that the simulation completed #
 dist := [dist] :
    # build the final dist list -- just putting square brackets around all of the numbers #
evalf(nops(dist) / maxi);
    # Determines what fraction of the maxi trials actually produced electrons at the PMT
    anode #
 Mean(dist);  # Calculate the mean of the distribution #
Histogram(dist, frequencyscale = absolute, binwidth = 2e4, axes = boxed, view = [0 ..2e6, 0
    ..500], labels = [typeset("electrons"), typeset("distribution")], labeldirections
    = ["horizontal", "vertical"], symbol = circle, symbolsize = 20, thickness = 2, tickmarks
    = [8, 8], colour = blue , axesfont = [Times, 12] , labelfont = [Times, 14], axis = [gridlines
    = [thickness = 2]]);
    # Plot a histogram of the distribution of the number of electrons collected at the anode #
binData := TallyInto(dist, [0 ..2e6], bins = 100) :
    # Use TalleyInto to build customized bins from the dist list #
 counts := [seq(rhs(binData[i]), i = 1 ..nops(binData))] :
    # Extract the bin counts generated by TallyInto #
 centres := [seq(1e4 + 2e4·i, i = 0 ..nops(binData) - 1)] : # Make a list of the bin centres #
logplot(centres, counts, view = [0 ..2e6, 0.5 ..500], style = point, axes = boxed, labels
    = [typeset("electrons"), typeset("distribution")], labeldirections = ["horizontal",
    "vertical"], symbol = circle, symbolsize = 15, thickness = 2, tickmarks = [8, 8], colour
```

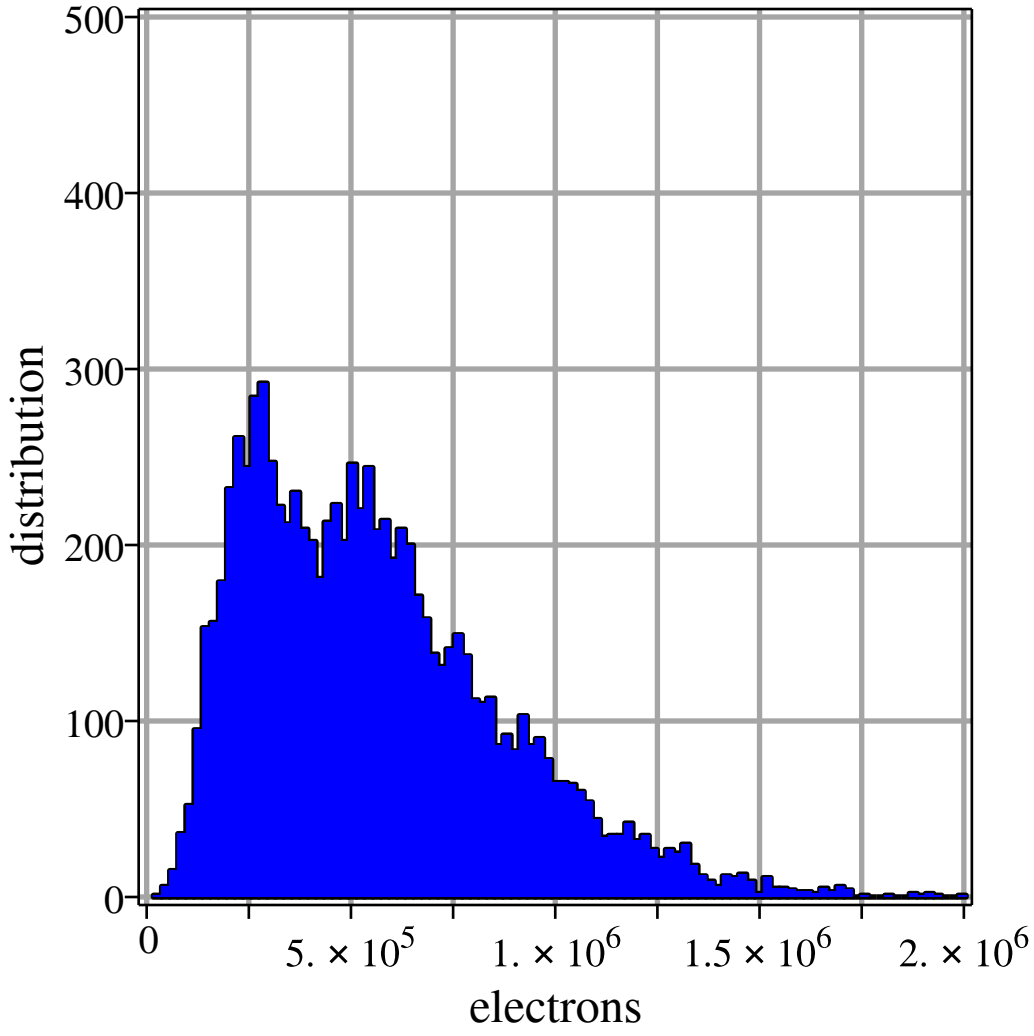
= *blue* , *axesfont* = [*Times*, 12] , *labelfont* = [*Times*, 14], *axis* = [*gridlines* = [*thickness* = 2]]); *# Make a semi-log plot of the distribution histogram #*

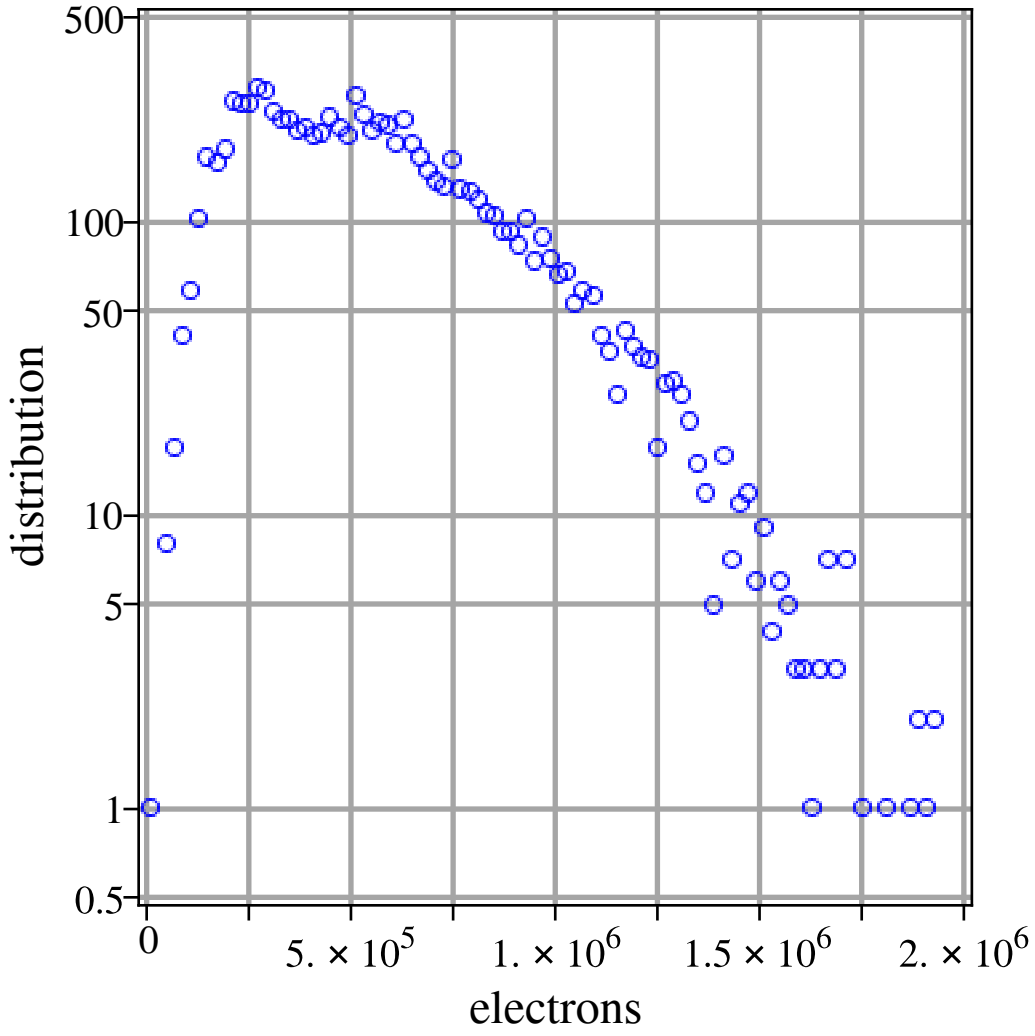
"48:04"

"35:27"

0.8706000000

$5.573354118\ 10^{5}$





Finally, this last block of code runs the same PMT simulation above four times, each time using a different number of incoming photons [1, 2, 4, 8]. I put a restart at the beginning of the block only to emphasize that this chunk of code doesn't rely on anything that came before it. These 44 lines of code will simulate the expected distributions of anode electrons for 4 difference sets of incoming photons. There real virtue of the Monte Carlo simulation is that we can now vary properties of the PMT with trivial modifications to the code below and systematically study the effects. For example, what if there were mode dynodes? All we have to do is modify the line dynode:=[0,150,300,450,600,750,800]. Alternatively, we could keep the number of dynodes fixed and modify the potential applied to the dynodes. Of course, after making this relatively simple simulation work, we could make modifications to make it more sophisticated. What if 8 photons are directed towards the photocathode, but they arrive at slightly different times. What does the current pulse at the anode look like? That's not a problem that we'll tackle here, but it does demonstrate the versitility of the Monte Carlo method. Warning: This block of code took just under two hours to complete on my laptop.

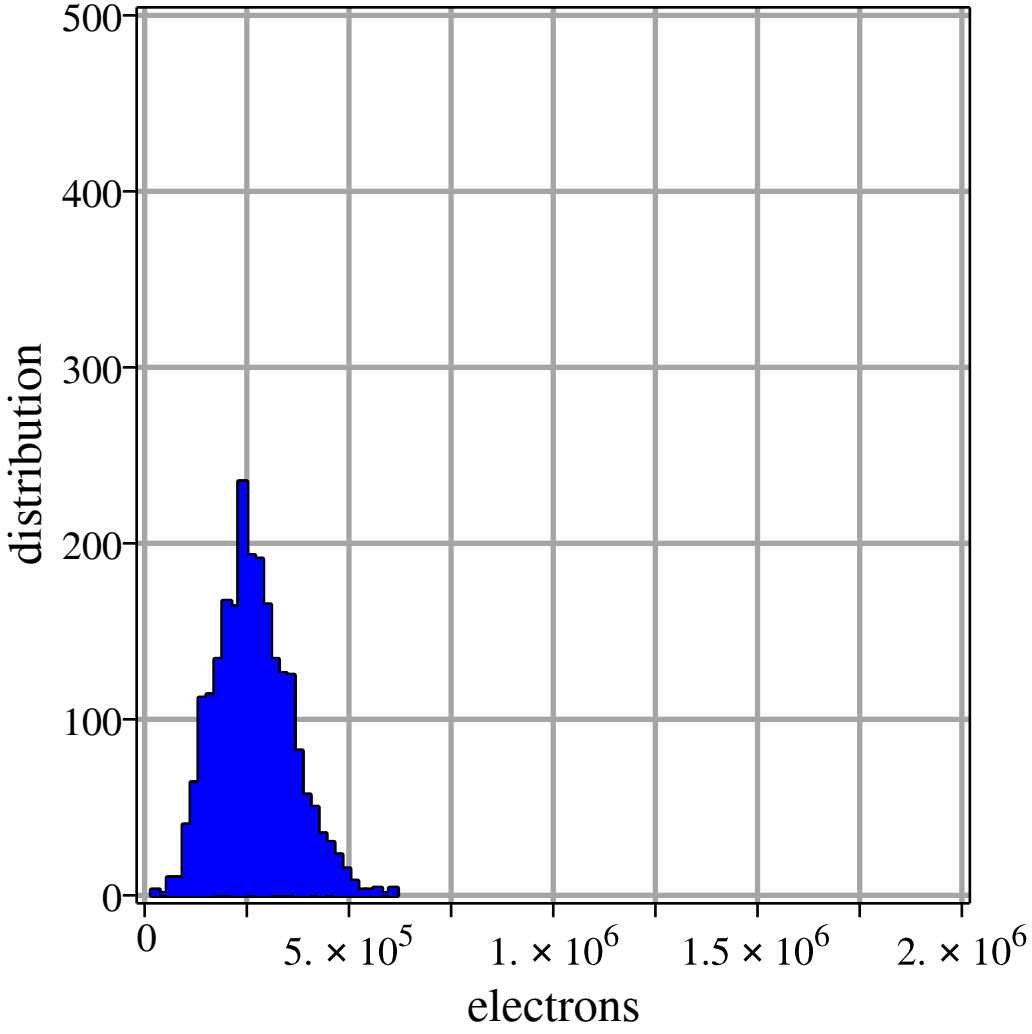
```
> restart;
  with(stats) :
  with(plots) :
  with(Statistics) :
  with(StringTools) :
  X := x→stats[random, uniform[0, 1]](1) :
      # Use X( ) to generate random numbers uniformly distributed between [0, 1] #
```

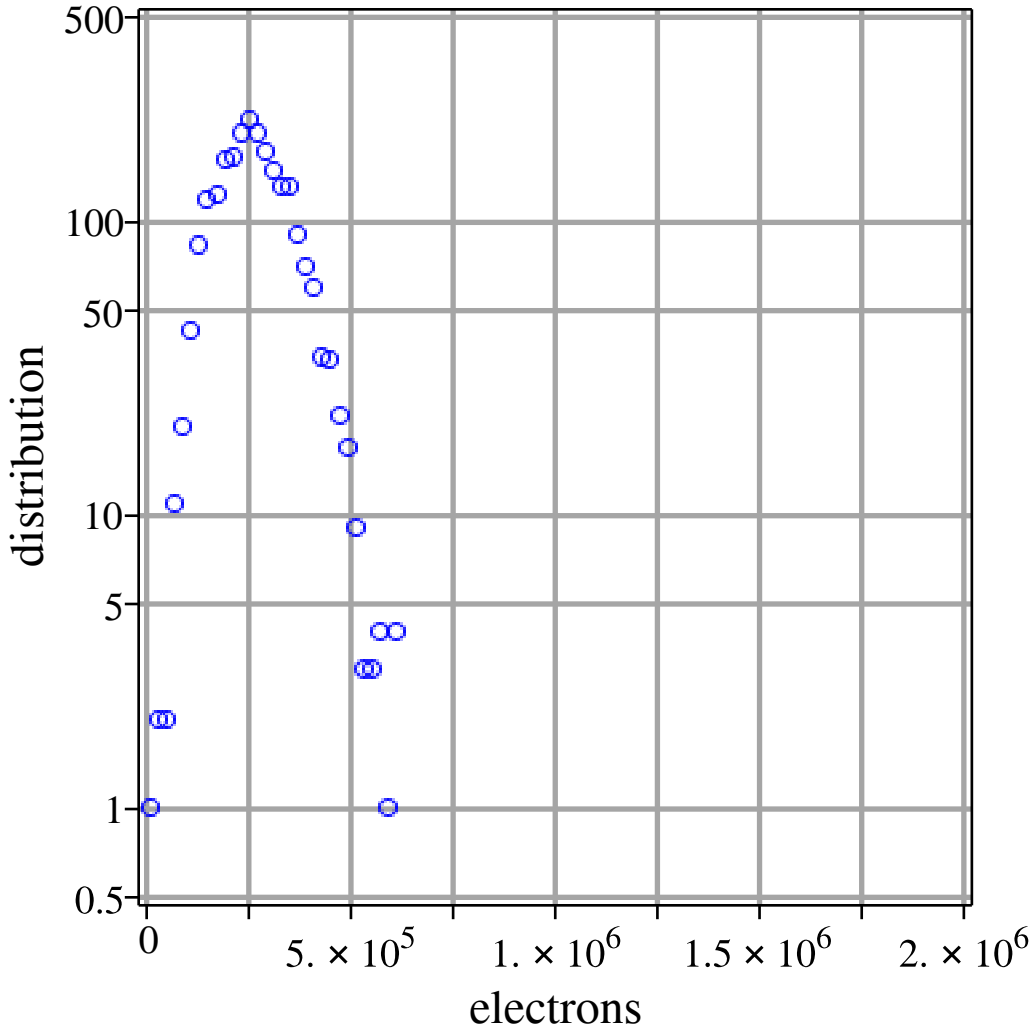
```
μ := 'μ': # Clear any previous use of μ #
Y := y → stats[random, poisson[μ]](1) :
    # Will use Poisson distributed nos. to determine the number of secondary electrons
    produced at a dynode #
QE := 0.23 : # Set the quantum effieicency of the photocathode #
maxi := 10e3 : # Set the number of Monte Carlo iterations #
dynode := [0, 150, 300, 450, 600, 750, 850] :
    # Set the number of dynodes and their voltages #
eventsList := NULL :
    # enventsList will keep track of how many of the maxi photon trials produced an electron
    shower at the anode of the PMT #
numPhotons := [1, 2, 4, 8] :
    # Will run the Monte Carlo simulation maxi times for 1 incoming photon, then 2 incoming
    photons, ... #
FormatTime("%H:%M");
    # Print the time (hours and minutes) that the simulation was started #` `
for m from 1 to nops(numPhotons) do:
    # This loop runs the simulation first with 1 incoming photon, then with 2 incoming photons,
    ... #
  dist := NULL :
    # The dist list will keep track of how many electrons where detected at the PMT anode for
    trials in which there was at least on photoelectron generated at the photocathode #
  for i from 1 to maxi do:
    # This loop runs the Monte Carlo simulation maxi times (10,000 in this case) #
     pe := 0 : # Set the number of photoelectrons to be zero #
     for k from 1 to numPhotons[m] do:
    # This loop individually steps through each of the photons incident on the photocathode #
       if X( ) < QE then:
    # Generate a rondom number between [0,1]. Compare X( ) to the photocathode quantum
    efficiency #
         pe := pe + 1 : # If true, then a photoelectron is produced. Increment pe by 1 #
       end if : # end if #
     end do: # end the incoming photon loop #
     if pe > 0 then:
    # If tbere are photoelectrons, then do something. If there are no photoelectrons then do
    nothing #
       electrons := pe :
    # Set the initial number of electrons in the PMT to be equal to the number of photoelectrons
    generated at the photocathode #
       for j from 1 to nops(dynode) − 1 do:
    # This loop will examine the secondary electrons produced at each of the dynodes #
         μ := electrons· (dynode[j + 1] - dynode[j]) / 20 :
    # Mean no. of electrons generated at a dynode determined from the no. of incoming
    electrons and the energy of the incoming electrons. One electron accelerated through 20 V
    will on average produce one electron #
         electrons := electrons + Y( );
    # The number of electrons leaving the dynode equal to the no. of incomming electrons plus
    the number generated determined by the Poisson distributed random number #
       end do: # end the dynode loop #
       dist := dist, electrons :
```

```
    # Add the total number of electrons detected at the anode for this trial to the dist list #
      end if: # end if #
    end do: # end the 1 to maxi loop, i.e. the main Monte Carlo loop #
    dist := [dist]:
    # build the final dist list -- just putting square brackets around all of the numbers #
    print(numPhotons[m], evalf(nops(dist)/maxi), Mean(dist), FormatTime("%H:%M"));
    # This sequency of code can take a long time to complete.  Generate some intermediate
    output to keep track of progress and the amount of time elapsed #
    eventsList := eventsList, [numPhotons[m], evalf(nops(dist)/maxi)]:
    # Build eventsList.  Determines what fraction of the maxi trials actually produced electrons
    at the PMT anode #
    histPlot := Histogram(dist, frequencyscale = absolute, binwidth = 2e4, axes = boxed, view
      = [0 ..2e6, 0 ..500], labels = [typeset("electrons"), typeset("distribution")], labeldirections
      = ["horizontal", "vertical"], symbol = circle, symbolsize = 20, thickness = 2, tickmarks
      = [8, 8], colour = blue , axesfont = [Times, 12] , labelfont = [Times, 14], axis = [gridlines
      = [thickness = 2]]) :
    # Plot a histogram of the distribution of the number of electrons collected at the anode #
    print(display(histPlot)); # Display the histogram as an intermediate output #
    binData := TallyInto(dist, [0 ..2e6], bins = 100) :
    # Use TalleyInto to build customized bins from the dist list #
    counts := [seq(rhs(binData[i]), i = 1 ..nops(binData))]:
    # Extract the bin counts generated by TallyInto #
    centres := [seq(1e4 + 2e4·i, i = 0 ..nops(binData) − 1)]: # Make a list of the bin centres #
    logPlot := logplot(centres, counts, view = [0 ..2e6, 0.5 ..500], style = point, axes = boxed,
      labels = [typeset("electrons"), typeset("distribution")], labeldirections = ["horizontal",
      "vertical"], symbol = circle, symbolsize = 15, thickness = 2, tickmarks = [8, 8], colour
      = blue , axesfont = [Times, 12] , labelfont = [Times, 14], axis = [gridlines = [thickness
      = 2]]) :
    # Make a semi-log plot of the distribution histogram.  The log scale will highlight the
    differences between the distributions for different numbers of incoming photons #
    print(display(logPlot)); # Display the semi-log plot as an intermediate output #
end do: # end the loop over the number of incoming photons #
eventsList := [eventsList]; # display the enventsList data #
```

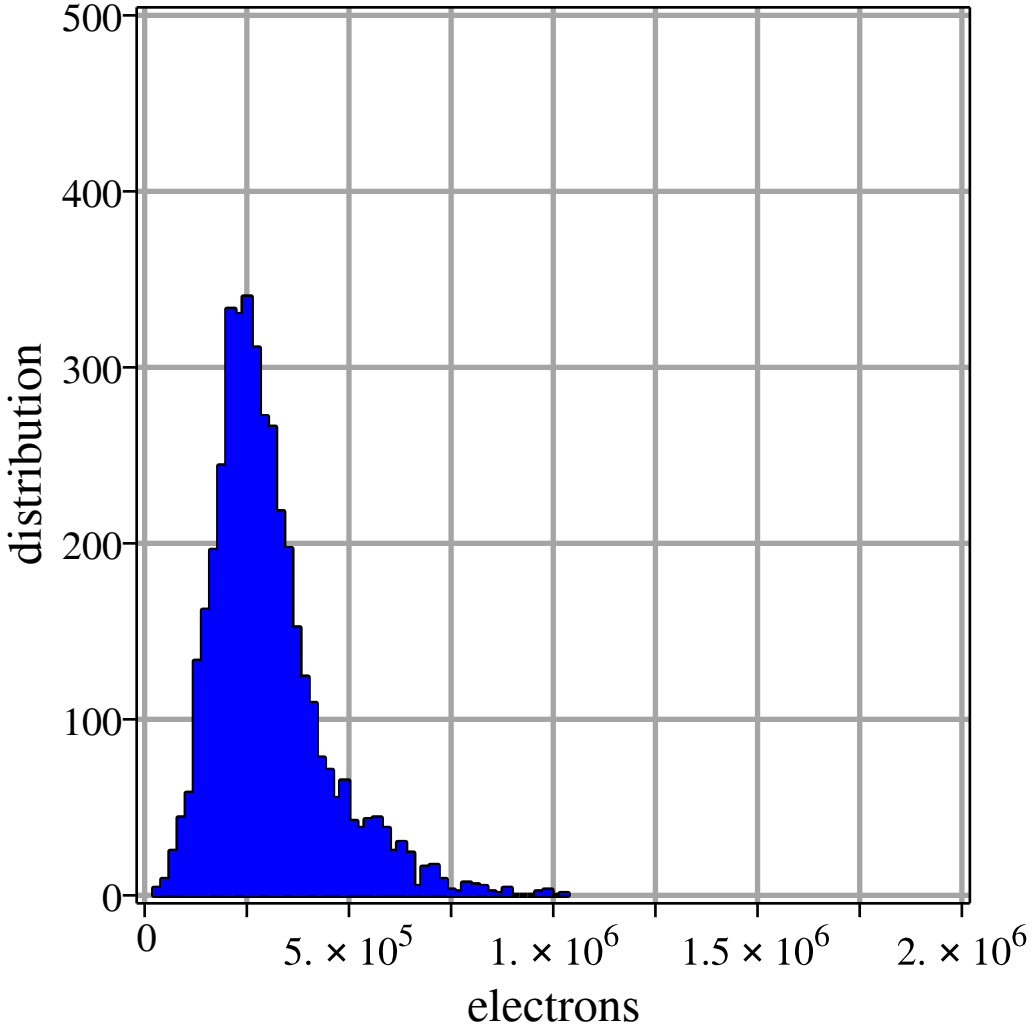
"00:35"

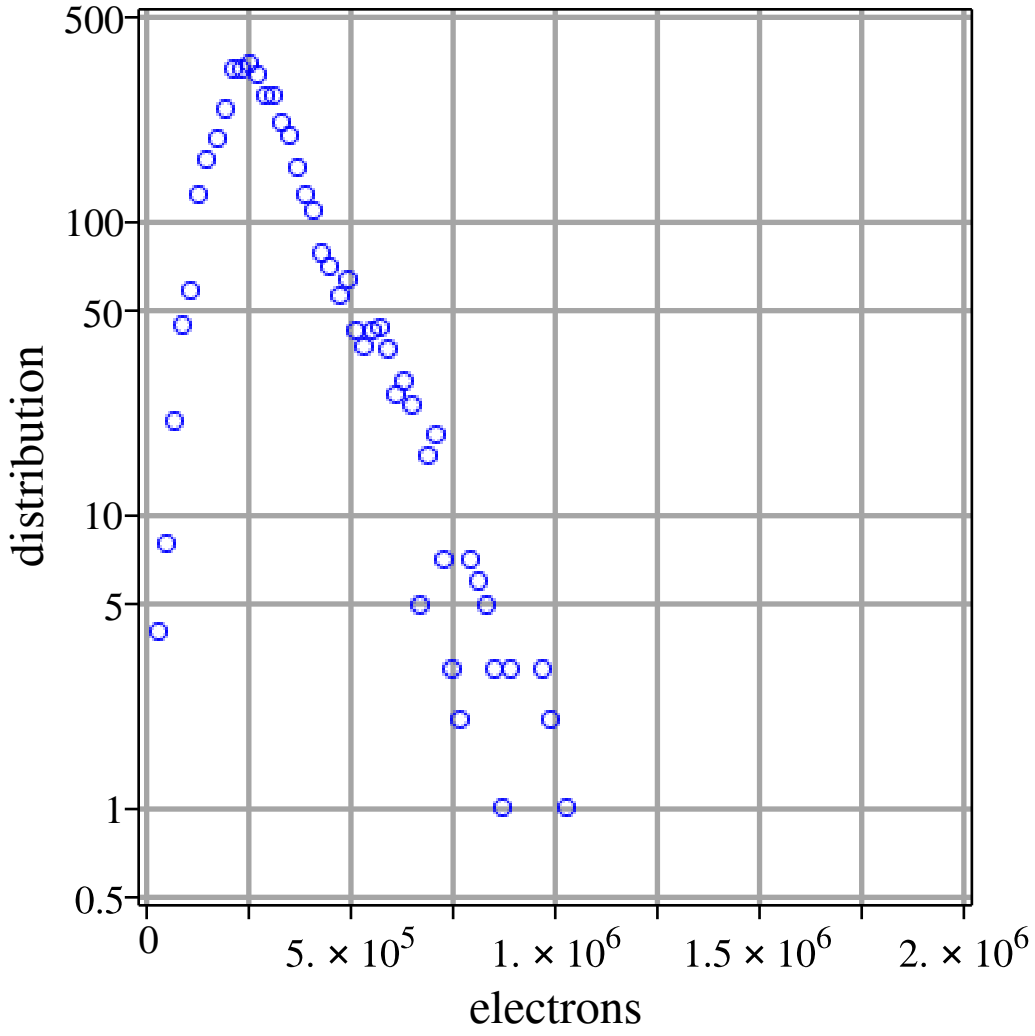
1, 0.2303000000, $2.668248289\ 10^5$, "00:43"


500
400
300
200
100
0
distribution
0
5. × 10^5
1. × 10^6
1.5 × 10^6
2. × 10^6
electrons

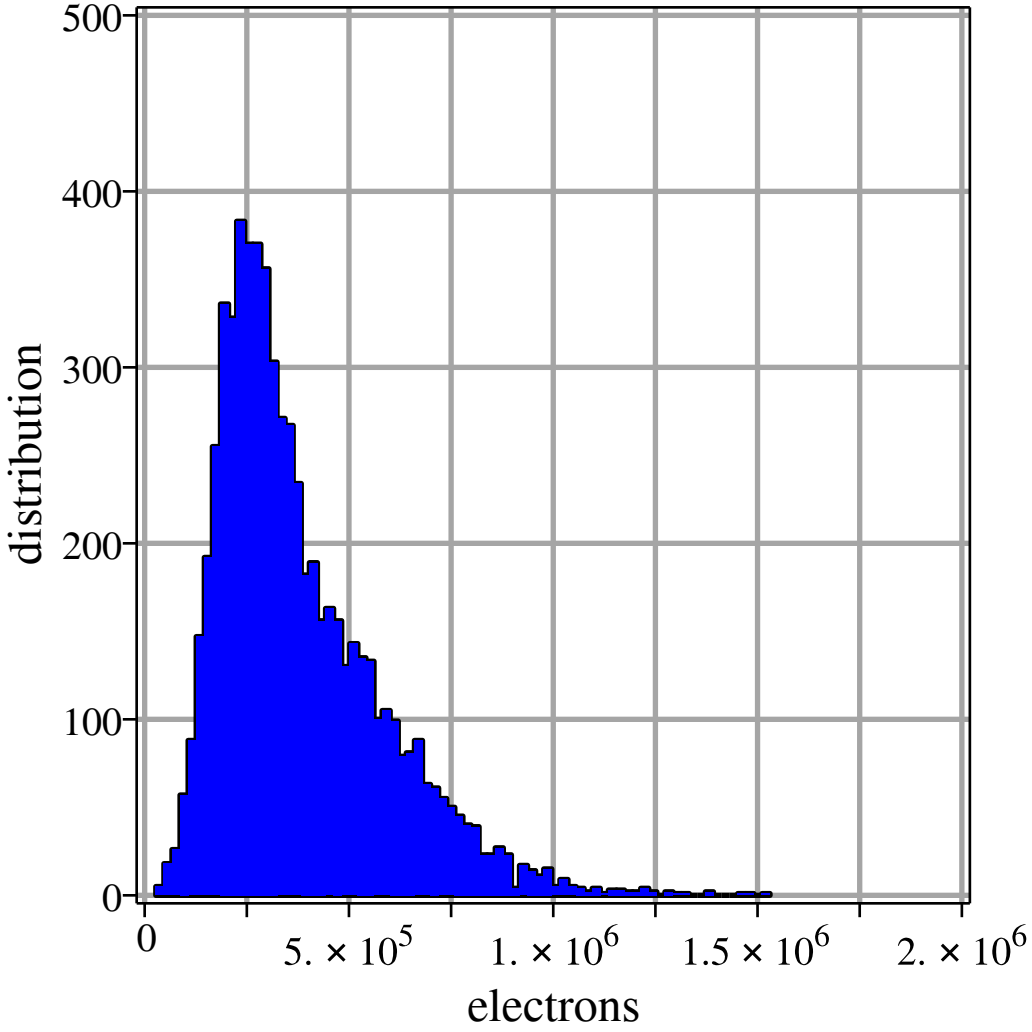


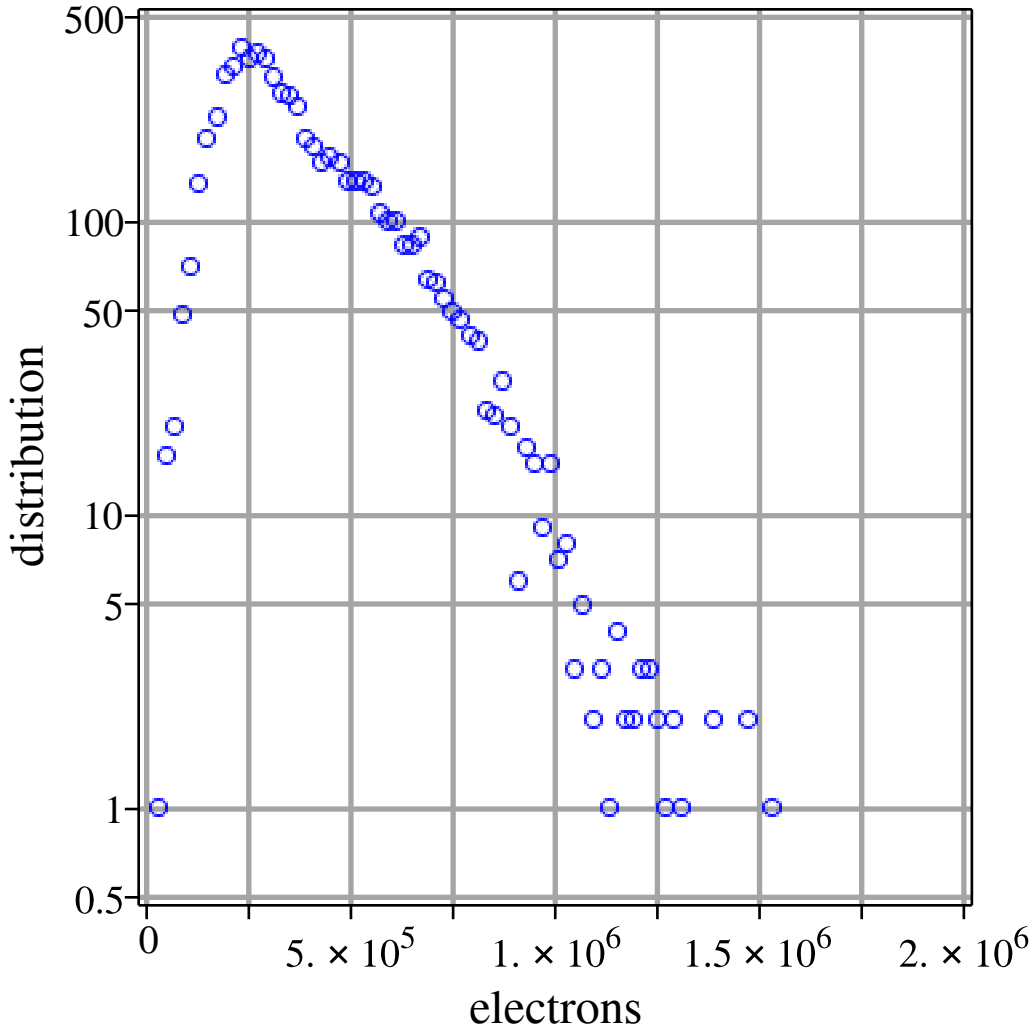
2, 0.4163000000, $3.008111307\ 10^5$, "00:59"


500
400
300
200
100
0
distribution
0
5. × 10^5
1. × 10^6
1.5 × 10^6
2. × 10^6
electrons

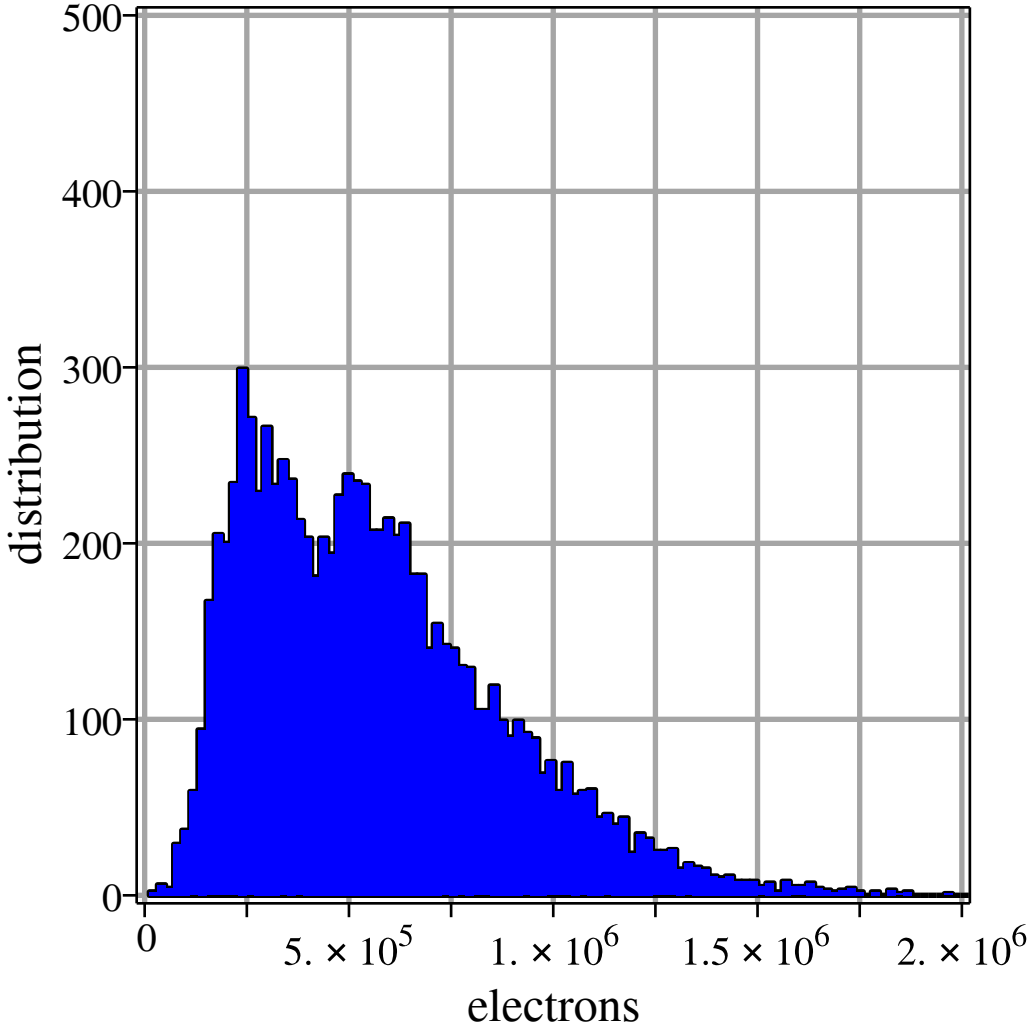


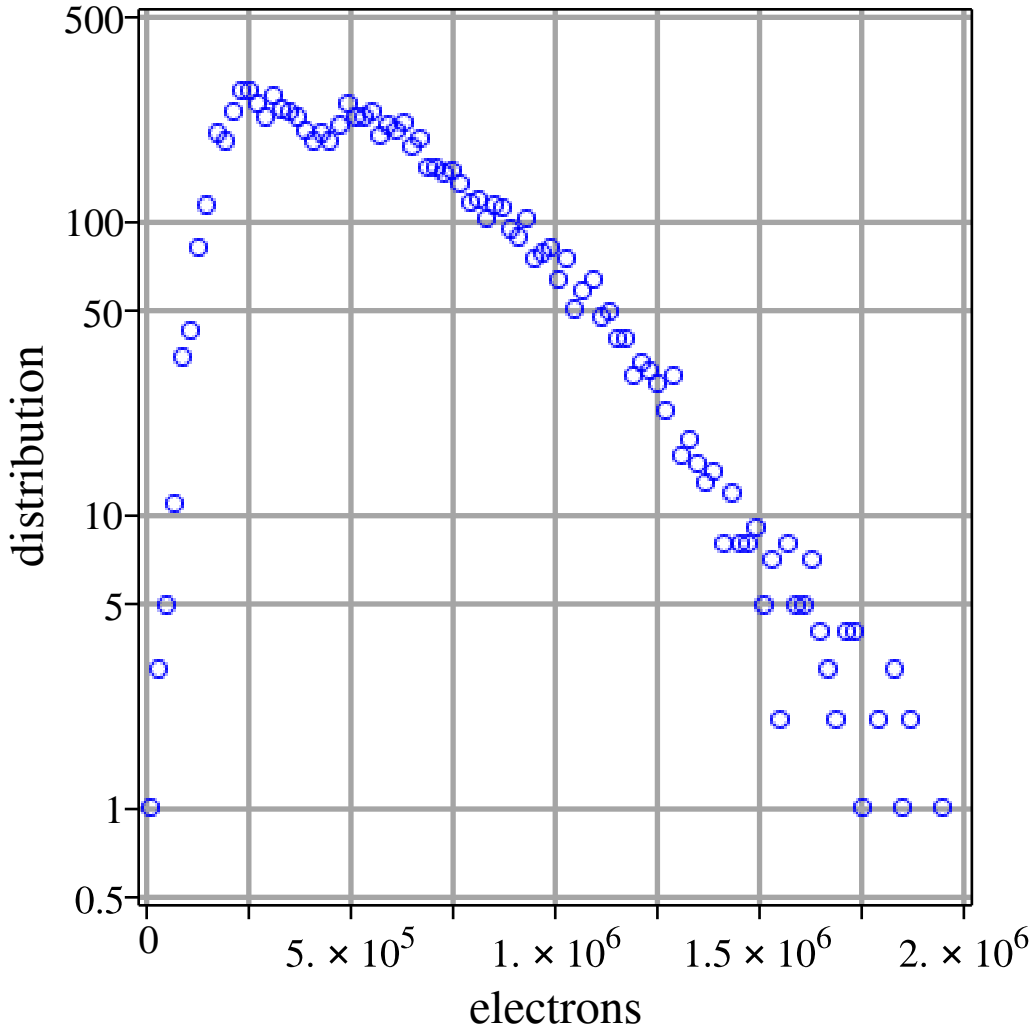
4, 0.6510000000, 3.779752266 10^5, "01:27"


500
400
300
200
100
0
distribution
0
5. × 10^5
1. × 10^6
1.5 × 10^6
2. × 10^6
electrons



8, 0.8771000000, $5.640644114\ 10^5$, "04:20"


500
400
300
200
100
0
distribution
0
5. × 10^5
1. × 10^6
1.5 × 10^6
2. × 10^6
electrons



eventsList := [[1, 0.2303000000], [2, 0.4163000000], [4, 0.6510000000], [8, 0.8771000000]] **(4)**

>