Calculating current for ERPA

I = J * A

J = qn/cm^2s

```
%{
q = 1.6*10^-19                  % C - Coulombs of 1 electron
alpha = 10^9                    % 1/cm^2s - number of electrons in the cusp
from paper Laura sent
d = 1                           % inch - 1 inch diameter of the anode
d_cm = d * 2.54                 % cm - convert anode diameter from inches to
cm
area = (pi/4) * d_cm^2          % cm^2

j = q * alpha                   % qn/cm^2s - current density
current = j * area              % Amperes - q/s
```

Determining possible energy steps:

Finding minimum area for 1 nA of current

A = I/J

```
%{
current_min = 10^-9             % Amperes - minimum current for ERPA

area_min = current_min/j        % cm^2
d_min = 2 * sqrt(area_min/pi)   % cm - area to diameter
d_min_in = d_min / 2.54         % inches - diameter from cm to in
%}
```

With the current diameter of the ERPA's anode being 1 inch, it is slightly too small to collect 1 nA

Finding the energy bins

an almost linear relationship between (10 eV, 10^9) and (100 eV, 10^8). find the relationship

units would be: # / (eV * sr * s * cm^2)

can ignore sr for now as we are looking in one direction and all the electrons from that one direction. JUST BE CAREFUL THAT WE DON'T PUT per steradian, its per degree fov (wrt B)

```
coefficients = polyfit([10, 1000], [10^9, 10^7], 1);
a = coefficients (1)
b = coefficients (2)
```

now use this equation to figure out the densities of the each section lets try 10-20, 20-30, 40-50, 50-100, 100-150?

```
%energy levels would collect all the enrgies above it.
%do the normalizing HERE before the integration each delta E at a time
energy = 10:1:150;
plot((a*energy + b))
i = 1;
dense = zeros(1,length(energy));
for i = 1:length(energy);
    dense(i) = (a*energy(i) + b)/energy(i);
    i = i + 1;
end

plot(energy, dense)

fitob = fit(energy.', dense.', 'exp2')

plot(fitob, energy, dense)
af = 1.89e+08
bf = -0.1053
cf = 3.532e+07
df = -0.01327
fun = @(x) af*exp(bf*x) + cf*exp(df*x)

density = integral(fun,10,150)

%energy = 10:1:150
%normalize = 1/sum(energy)
%density = integ*normalize    %this number seems wrong

j_level = q * density                   % qn/cm^2s - current density
current_level = j_level * area

%and then to figure out if its enough, multiple by the cadence
time = 100 % time between measurements in ms
final = current_level * (time) / 1000


%now that we know it works lets make a loop
levels = [15,30,45,60,75,90,105,120]
charge = zeros(1,length(levels));
k = 1;
for k = 1:length(levels);
    level_density = integral(fun,levels(k),150);
    level_current = q * level_density * area;
    charge(k) = level_current * time / 1000;
    k = k + 1;
end
charge
%}
```

1 nC/s is the rate of the charges and the cumulation time is 1 ms

10^-12 C gathered as a minimum threshold for the ERPA

10 samples per second

each measurement 100 ms, 5 different energy steps

Update 2/7

I will be doing the previous calculations, except the numbers I originally went off of (10^9 and 10^8) are already normalized. So this round I take out the normalization I did in the previous script.

The points I was given are (10 eV, 10^9 # / (eV * sr * s * cm^2)) and (100 eV, 10^8 # / (eV * sr * s * cm^2)). This is given by the FAST paper/data that shows precipitating electrons have a uniform number of 10^10 #* eV / (eV * sr * s * cm^2). And due to the way the instrument measures, each point must be normalized by dividing 10^10 by the energy bin it is in. Therefore giving the 10^9 and 10^8 data points used.

Extrapolating this to 10^7 # / (eV * sr * s * cm^2) at 1000 eV is a decent approximation. But, this should be looked into further as there are other higher energy events that might contribute. Although, these are rare events, making this a decent approximation.

```
coefficients = polyfit([10, 1000], [10^9, 10^7], 1);
a = coefficients (1)
```

```
a = -1.0000e+06
```

```
b = coefficients (2)
```

```
b = 1.0100e+09
```

```

q = 1.6*10^-19                  % C - Coulombs of 1 electron
```

```
q = 1.6000e-19
```

```
alpha = 10^9                    % 1/cm^2s - number of electrons in the cusp
from paper Laura sent
```

```
alpha = 1.0000e+09
```

```
d = 1                           % inch - 1 inch diameter of the anode
```

```
d = 1
```

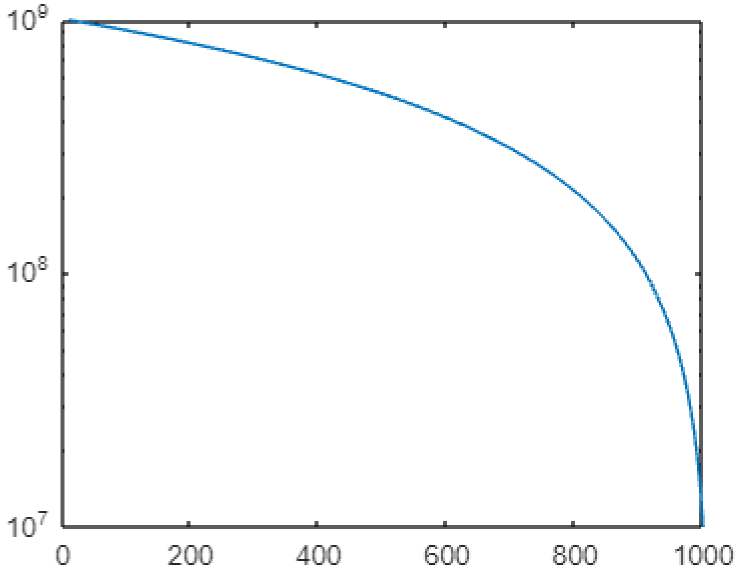
```
d_cm = d * 2.54                 % cm - convert anode diameter from inches to
cm
```

```
d_cm = 2.5400
```

```
area = (pi/4) * d_cm^2          % cm^2
```

```
area = 5.0671
```

```
%energy levels would collect all the enrgies above it.

energy = 10:1:1000;
dense = a*energy + b;
plot(energy, dense)
set(gca, 'YScale', 'log')
xlim([0,1000])
```



```
fun = @(x) a*x + b;

density = integral(fun,10,1000);

j_level = q * density;                    % qn/cm^2s - current density
current_level = j_level * area;

%and then to figure out if its enough, multiple by the cadence
time = 100 % time between measurements in ms
```

```
time = 100
```

```
final = current_level * (time) / 1000;


%now that we know it works lets make a loop!
levels = [3,10,15,20,25,30,40,45,50,60,75,80,90,100,105,120,125,140]
```

```
levels = 1×18
     3    10    15    20    25    30    40    45    50    60    75    80    90 ···
```

```
charge = zeros(1,length(levels));
k = 1;
for k = 1:length(levels);
```

```
    level_density = integral(fun,levels(k),1000);   % if you want to change summation limit, do here!
    level_current(k) = q * level_density * area;
    charge(k) = level_current(k) * time / 1000;
    k = k + 1;
end
charge
```

```
charge = 1×18
10^-7 ×
    0.4110    0.4053    0.4013    0.3973    0.3933    0.3893    0.3814    0.3774 ⋯
```

```
level_current
```

```
level_current = 1×18
10^-6 ×
    0.4110    0.4053    0.4013    0.3973    0.3933    0.3893    0.3814    0.3774 ⋯
```

Update 5/10

Concerns of these numbers being too high, so below I will normalize the data by using multiple points instead of two.

This was done previously, but with the wrong starting number (as it was the already normalized value).

```
%energy levels would collect all the enrgies above it.
%do the normalizing HERE before the integration each delta E at a time
energy_n = 3:1:1000;
i = 1;
dense_n = zeros(1,length(energy_n));
for i = 1:length(energy_n);
    dense_n(i) = (1e10)/energy_n(i);
    i = i + 1;
end

%plot(energy_n,dense_n)

fitob = fit(energy_n.', dense_n.', 'exp2')
```

```
fitob = 
     General model Exp2:
     fitob(x) = a*exp(b*x) + c*exp(d*x)
     Coefficients (with 95% confidence bounds):
       a =   5.118e+09  (4.988e+09, 5.249e+09)
       b =     -0.2195  (-0.2254, -0.2136)
       c =   4.733e+08  (4.541e+08, 4.924e+08)
       d =    -0.01388  (-0.01448, -0.01328)
```

```
af = 5.118e09
```

```
af = 5.1180e+09
```

```
bf = -0.2195
bf = -0.2195
```

```
cf = 4.733e08
```

```
cf = 473300000
```

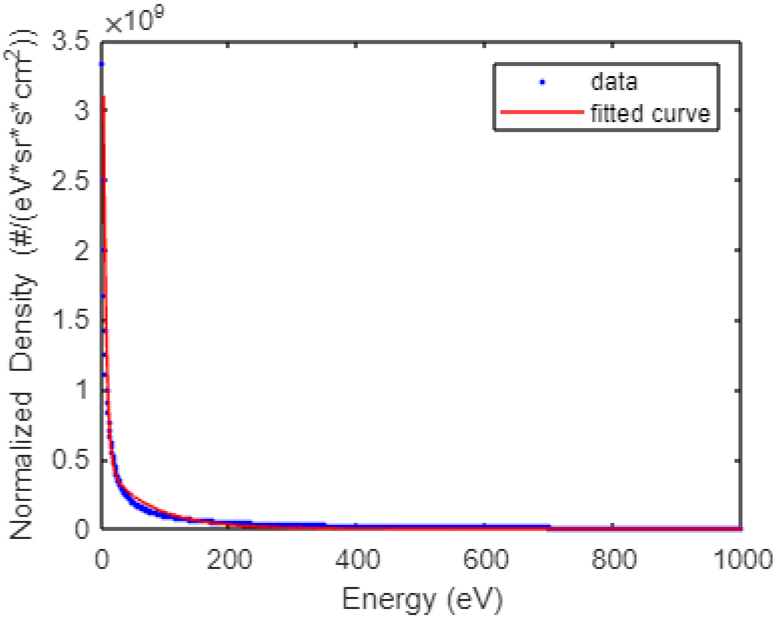
```
df = -0.01388
```

```
df = -0.0139
```

```
fun = @(x) af*exp(bf*x) + cf*exp(df*x)
```

```
fun = function_handle with value:
    @(x)af*exp(bf*x)+cf*exp(df*x)
```

```
plot(fitob, energy_n, dense_n)
ylabel('Normalized Density (#/(eV*sr*s*cm^2))')
xlabel('Energy (eV)')
```



```
density_n = integral(fun,10,1000)
```

```
density_n = 3.2277e+10
```

```
levels_n = [3,10,15,20,25,30,40,45,50,60,75,80,90,100,105,120,125,140]
```

```
levels_n = 1×18
     3    10    15    20    25    30    40    45    50    60    75    80    90···
```

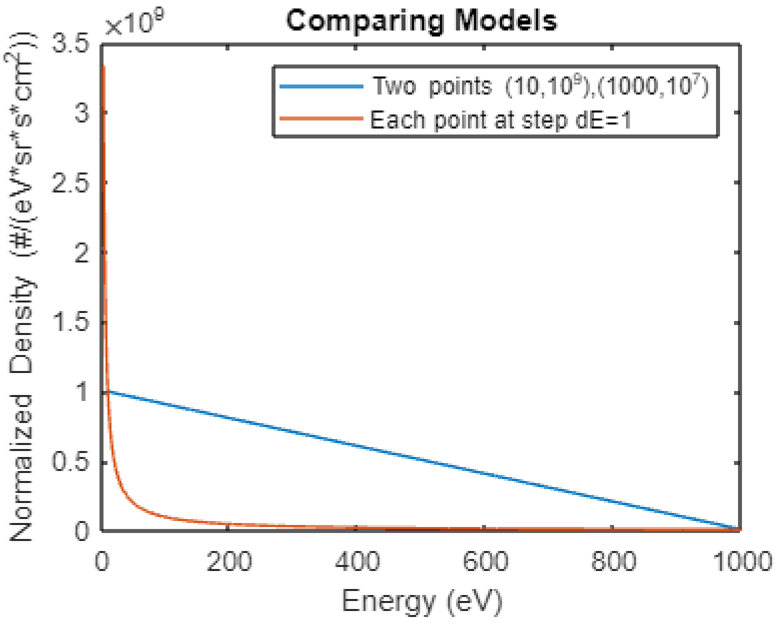
```
charge_n = zeros(1,length(levels_n));
l = 1;
for l = 1:length(levels_n);
    level_density_n = integral(fun,levels_n(l),1000);   % if you want to
change summation limit, do here!
    level_current_n(l) = q * level_density_n * area;
    charge_n(l) = level_current_n(l) * time / 1000;
    l = l + 1;
end
charge_n
```

```
charge_n = 1×18
10^-8 ×
    0.3630    0.2617    0.2315    0.2118    0.1962    0.1826    0.1587    0.1480 ···
```

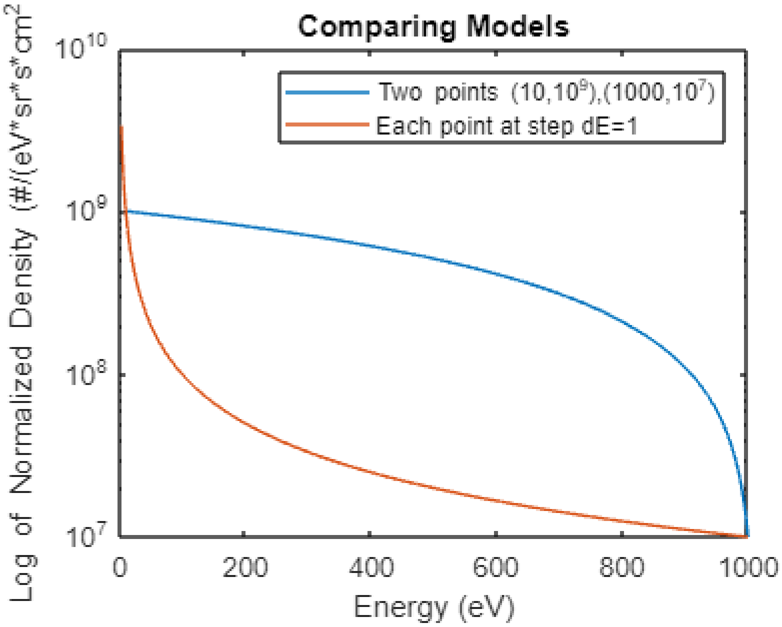
```
level_current_n
```

```
level_current_n = 1×18
10^-7 ×
    0.3630    0.2617    0.2315    0.2118    0.1962    0.1826    0.1587    0.1480 ···
```

```

%I want to plot next a comparison of the previous density used and this
%new version

plot(energy,dense,energy_n,dense_n)
title('Comparing Models')
ylabel('Normalized Density (#/(eV*sr*s*cm^2))')
xlabel('Energy (eV)')
legend({'Two points (10,10^9),(1000,10^7)','Each point at step
dE=1'},'Location','northeast')
```



```

%plot y axis logarithmically
semilogy(energy,dense,energy_n,dense_n)
title('Comparing Models')
ylabel('Log of Normalized Density (#/(eV*sr*s*cm^2))')
xlabel('Energy (eV)')
legend({'Two points (10,10^9),(1000,10^7)','Each point at step
dE=1'},'Location','northeast')
```



6/7 UPDATE:

Suggested to use CREX-2 EPLAS data that Marc gave me as it is in Hz over energy and time. This multiplied by charge gives a current over multiple energies.

The approximate value that Marc picked out from the data is ~5 (log) MHz for energies (~0.025 keV - 0.2 to 1 keV)

```
%inputting the assumptions discussed above and converting to current
hz = exp(5) * 10^6     %units of Hz (MHz to Hz)
```

```
hz = 1.4841e+08
```

```
current_c = hz * q     %current, units of charge/s
```

```
current_c = 2.3746e-11
```

```

%then we want to relate it to the threshold of 10^-12 C, so integrate over
%100 ms
crex_q = current_c * time/1000  %using time in ms, then converting to s
```

```
crex_q = 2.3746e-12
```

```

%this oddly has a response that matches the pre-normalized data.
%so I will use this and sum over the previous energy levels but only
%starting at 25 eV
fun_c = @(x) x*0 + current_c
```

```
fun_c = function_handle with value:
    @(x)x*0+current_c
```

```
levels_c = [25,30,40,45,50,60,75,80,90,100,105,120,125,140]
levels_c = 1×14
    25    30    40    45    50    60    75    80    90   100   105   120   125 ···
```

```
charge_c = zeros(1,length(levels_c));
l = 1;
for l = 1:length(levels_c);
    level_current_c(l) = integral(fun_c,levels_c(l),1000);   % if you want
to change summation limit, do here!
    charge_c(l) = level_current_c(l) * time / 1000;
    l = l + 1;
end
level_current_c
```

```
level_current_c = 1×14
10^-7 ×
    0.2315    0.2303    0.2280    0.2268    0.2256    0.2232    0.2197    0.2185 ···
```

```
charge_c
```

```
charge_c = 1×14
10^-8 ×
    0.2315    0.2303    0.2280    0.2268    0.2256    0.2232    0.2197    0.2185 ···
```

8/2 UPDATE:

We should convert the Hz data to the same number density as previously worked with. This allows us to compare the EPLAS and the ERPA. To do this divide by the geometric factor of the CREX-2 EPLAS. Ian Cohen's thesis gives geo factor of each anode, g = 1.12x10^-4 sr * cm^2 * (ev/ev). So we should multiply by the number of anodes to get full geo factor.

Difference in radii between the inner and outer electrodes is 0.196 cm (cohen thesis)

Also look into, do we need to normalize this like Laura previously had me do?

Changes from 6/7 involve dividing by geometric factor g per anode, multiply number of anodes, then multiplying by the area of EPLAS to get the current.

```
%g is the geometric factor for the CREX-2 EPLAS (geo factor per anode times
%number of anodes)
g = 4.032 * 10^-3          %units of sr*cm^2*(ev/ev)
```

```
g = 0.0040
```

```
%r2 =                %radius of center electrode
%r1 =                %radius of electrode
%a_c = pi * (r2-r1)^2
%inputting the assumptions discussed above and converting to current
%hz2 = exp(5) * 10^6 / g
hz2 = exp(5) * 10^6     %units of Hz (MHz to Hz)
```

```
hz2 = 1.4841e+08
```

```
current_c2 = hz * q     %current, units of charge/s
```

```
current_c2 = 2.3746e-11
```

```
%then we want to relate it to the threshold of 10^-12 C, so integrate over
%100 ms
crex_q2 = current_c2 * time/1000  %using time in ms, then converting to s
```

```
crex_q2 = 2.3746e-12
```

```
%this oddly has a response that matches the pre-normalized data.
%so I will use this and sum over the previous energy levels but only
%starting at 25 eV
fun_c2 = @(x) x*0 + current_c2
```

```
fun_c2 = function_handle with value:
    @(x)x*0+current_c2
```

```
levels_c2 = [25,30,40,45,50,60,75,80,90,100,105,120,125,140]
```

```
levels_c2 = 1×14
    25     30     40     45     50     60     75     80     90    100    105    120    125 ···
```

```
charge_c2 = zeros(1,length(levels_c2));
l = 1;
for l = 1:length(levels_c2);
    level_current_c2(l) = integral(fun_c2,levels_c2(l),1000);   % if you
want to change summation limit, do here!
    charge_c2(l) = level_current_c2(l) * time / 1000;
    l = l + 1;
end
level_current_c2
```

```
level_current_c2 = 1×14
10^-7 ×
    0.2315    0.2303    0.2280    0.2268    0.2256    0.2232    0.2197    0.2185 ···
```

```
charge_c2
```

```
charge_c2 = 1×14
10^-8 ×
    0.2315    0.2303    0.2280    0.2268    0.2256    0.2232    0.2197    0.2185 ···
```

8/25 Update - RENU2 ERPA Current:

Marc gave me the RENU2 ERPA values in nA, which would contain total current from all energies above 3 eV. Approximate peak values from electron precipitation events were seen as 0.8 nA, 1.5 nA, 1.0 nA, 0.7 nA, 1.4 nA, 0.5 nA. The background current was approximately 0.2 nA. As a note, the RENU2 ERPA has a cadence of 1 ms.

Can assume here as well that the flux distribution is approximately flat over energies.

```
levels_re = [3,25,30,40,45,50,60,75,80,90,100,105,120,125,140]
levels_re = 1×15
     3    25    30    40    45    50    60    75    80    90   100   105   120 ···
```

```
current_re = [0.2,0.5,0.7,0.8,1.0,1.4,1.5] * 10^-9      %current values
converted from nA to A
```

```
current_re = 1×7
10^-8 ×
    0.0200    0.0500    0.0700    0.0800    0.1000    0.1400    0.1500
```

```

%assuming that the charge is distributed approximately equally amongst the
%energy levels, we can take the values measured by the ERPA and multiply by
%the fraction of total energy from that energy level to 150 eV.

%i will demonstrate as a proof of concept using the maximum current (1.5nA)
%measured by the ERPA from the energy level of 3 eV. That current is summed
%from 3 eV to 150 eV, so there are 147 eV to distribute current in.

ex_current = current_re(7) * (150 - levels_re(2))/147
```

```
ex_current = 1.2755e-09
```

11/8/22 Update - RENU2 ERPA/EPLAS:

I was given plots for the RENU2 ERPA that shows e- temp, skin current, payload potential, and current of e- above 3 eV. This will compare with the RENU2 EPLAS plot that is included in David Kenward's thesis (fig 28) to see how the flux was broken down above 3 eV.

RENU2 EPLAS has energy range of 5 eV - 14.6 keV.

The point of this is to determine what the pointing requirement should be, so I picked an event at approximately 7:44:30 as it is lower in flux and the breakdown in eV/(eV*s*sr*cm^2) appears linear in the logarithmic colorbar. So a possibly more average event to inform the requirement.

```
%finding the relationship between the
%x = 1:10
%y = zeros(1,length(x))
%slope = (10^10 - 2*10^9) / (100 - 5)
%for m = 1:length(x);
%    y[m] = (1e(slope*x[m]));
%    m = m + 1;
%end

```