Experiment 20 - The Geiger-Müller Detector

In[1]:=
```
<< CurveFit`
```

```
CurveFit for Mathematica v7.x thru v11.x, Version 1.96, 4/4/2018

Caltech Sophomore Physics Labs, Pasadena, CA
```

In[∘]:=
```
(* We export the data from the lab notebook for analysis in CurveFit. *)
SetDirectory[NotebookDirectory[]];
v0vp = {{"S V0(V)", "Vp(V)"}, { 800, {2.96, 2.94, 2.98}},
   {790, {2.48, 2.42, 2.46}}, {780, {1.88, 1.88, 1.82}}, {770, {1.34, 1.38, 1.38}},
   {760, {0.92, 0.912, 0.928}}, {750, {0.480, 0.484, 0.480}},
   {740, {0.119, 0.120, 0.122}}, {738, {0.0736, 0.0704, 0.0712}},
   {800, {3.02, 2.96, 3.00}}, {825, {3.92, 3.96, 3.92}}, {850, {4.76, 4.72, 4.72}},
   {875, {5.68, 5.52, 5.68}}, {900, {6.64, 6.52, 6.64}}, {812, {3.46, 3.46, 3.40}},
   {837, {4.28, 4.24, 4.32}}, {862, {5.08, 5.16, 5.16}}, {887, {6.08, 6.08, 5.96}}};

(* Match first element of list to every element
 of corresponding sublist to split data into tuples. *)
match[list_] := If[StringTake[ToString[list[[1]]], 1] == "S", {list},
   Table[{list[[1]], list[[2]][[i]]}, {i, 1, Length[list[[2]]]}]];
v0vpDat = Flatten[Table[match[v0vp[[i]]], {i, 1, Length[v0vp]}], 1];
td = {{"S V0(V)", "td(us)"}, {780, 83},
   {800, 80}, {820, 78}, {840, 74}, {880, 70}, {920, 66}, {950, 68}};

Export["v0vp.dat", v0vpDat];
Export["td.dat", td];
```

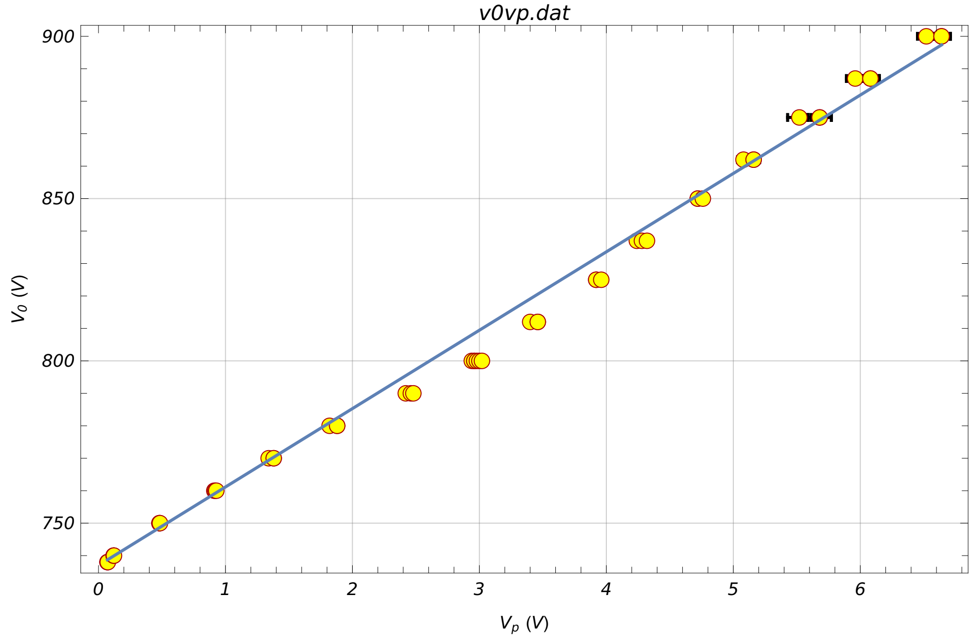
Determining V_{Th}

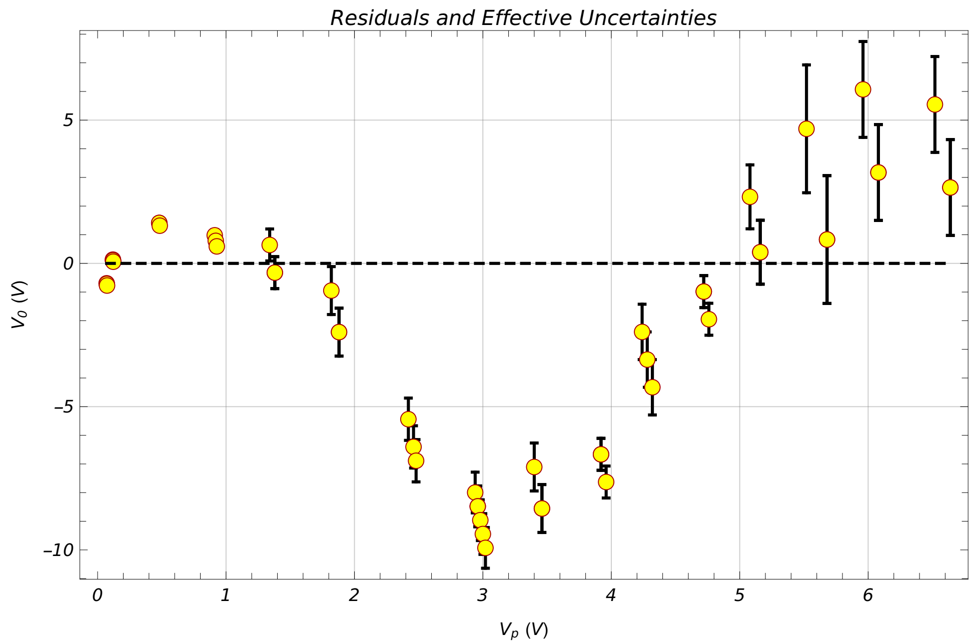
The threshold voltage V_{Th} is the smallest bias voltage for which Townsend cascades occur, i.e. the smallest V_0 for which pulses are detected. Therefore, our determination of V_0 will be carried out by performing a (linear) fit of V_0 against V_p and observing the y-intercept of the fit, essentially extrapolating to the point where the pulses just vanish.

```
In[•]:= With[ {name = SystemDialogInput["FileOpen",
            {DataFileName, {"data files" -> {"*.dat", "*.mca"}, "all files" -> {"*"}}}]},
         If[ name =!= $Canceled,
          LoadFile[name]]
        ]

In[•]:= CalculateYsigmas[]

In[•]:= SwitchXXandYY[]

In[•]:= LinearFit[]
```

In[∘]:=
```
LinearDifferencePlot[FrameLabel → {"Vp (V)", "V0 (V)"}]
```

Out[∘]=





y(x) = a + b x

a=	b=	
736.996	24.15	
σ_a=	σ_b=	$\chi^2/(n-2)$=
0.0157803	0.0336788	100.683

We observe a relatively bad linear fit indicated by the large $\tilde{\chi}^2$ and the clear trend in the residuals. As we calculated in the pre-lab, this indicates that some of our data points were taken with a V_0 for which the thin-sheath approximation does not hold, and therefore neither does our derived linear relation between V_p and V_0.

We have estimated the upper threshold for linear behavior to be around 800V, and this is indeed around where we observe a change in behavior. Therefore we remove the data points taken with $V_0 > 800\,V$ and perform a linear fit again. We also remove the data points we used for the point-estimate of V_{Th}, as those were taken at very low signal-to-noise ratios and are prone to error. The small V_0 measurements do however show some loss of linearity in V_p against V_0, possibly just due to measurement error for the reason mentioned, but this may also be due to a defect in our model at low bias voltages.

In[∘]:=
```
With[ {name = SystemDialogInput["FileOpen",
    {DataFileName, {"data files" -> {"*.dat", "*.mca"}, "all files" -> {"*"}}}]},
 If[ name =!= $Canceled,
  LoadFile[name]]
]
```

In[∘]:=
```
With[{x = SetXRange[ LinearDataPlot[], Log -> False,
    Label -> "Set the X values for the range you wish to keep." ]},
 Print[x];

 XRangeKeep[Sequence @@ x]
]
```

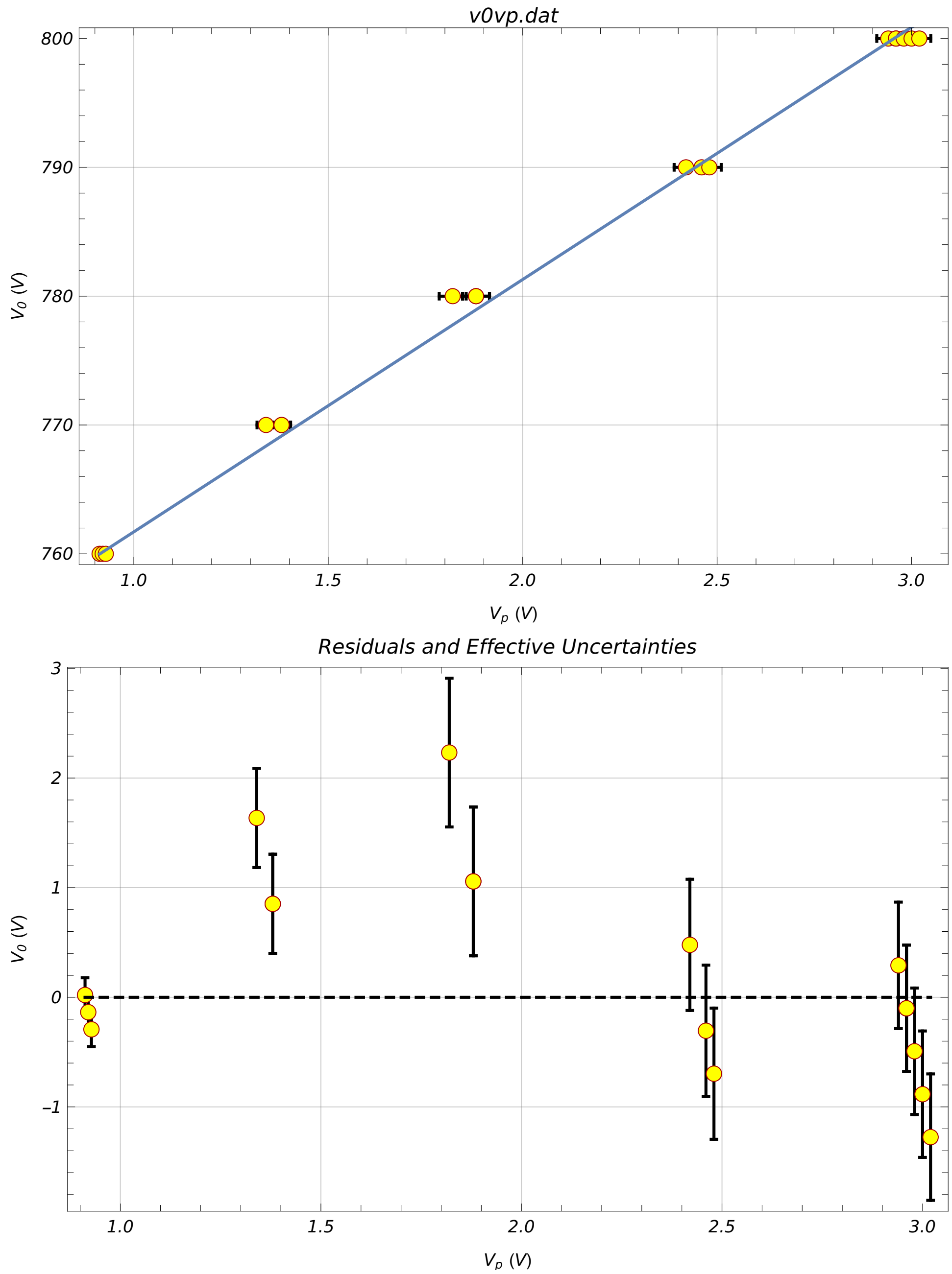
In[∘]:=
```
CalculateYsigmas[]
```

In[∘]:=
```
SwitchXXandYY[]
```

In[∘]:=
```
LinearFit[]
```

In[•]:= `LinearDifferencePlot[FrameLabel → {"V_p (V)", "V_0 (V)"}]`

Out[•]=




```
y(x) = a + b x
a=          b=
742.113     19.5905
σa=         σb=         χ²/(n-2)=
0.161309    0.111408    3.16602
```

As expected, we observe a much better fit as indicated by the relatively low $\tilde{\chi}^2$. The residuals do appear to show a (much less discernible) trend, again possibly showing that a linear model is not quite adequate, but they are mostly within error and therefore should be perfectly fine to use for the following analysis.

From the y-intercept of our linear plot, we can estimate V_{Th} = 742.1 ± 0.2 V. We observe that this is relatively close to our in-lab point estimate of 738 V.

Determining RC time constant in CI mode, τ

The long-term exponential decay of the Charge Integrated pulse is used to determine the RC time constant, τ, of the circuit in RC mode. We will then use τ to calculate C, the total capacitance of the circuit.

In[∘]:=
```
With[ {name = SystemDialogInput["FileOpen", {DataFileName,
      {"Tek waveform files" -> {"*.csv", "*.tsv"}, "all files" -> {"*"}}}]},
 If[ name =!= $Canceled,
  LoadTekFile[name]]
]
```

In[∘]:=
```
With[{x = SetXRange[ LinearDataPlot[], Log -> False,
     Label -> "Set the X values for the range you wish to keep." ]},
 Print[x];

 XRangeKeep[Sequence @@ x]
]
```

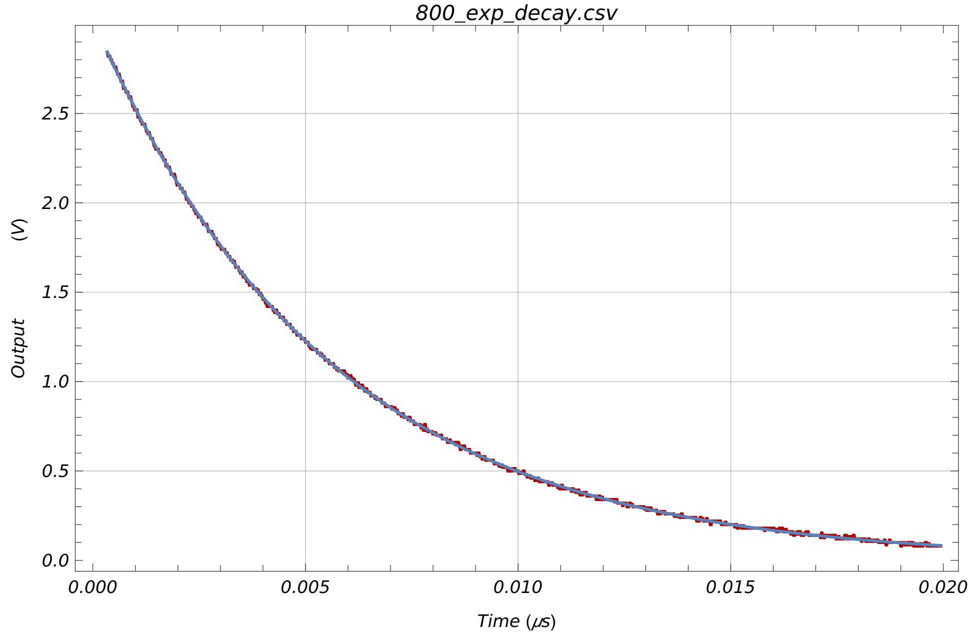
In[∘]:=
```
DecayingExponentialLFit::usage
```

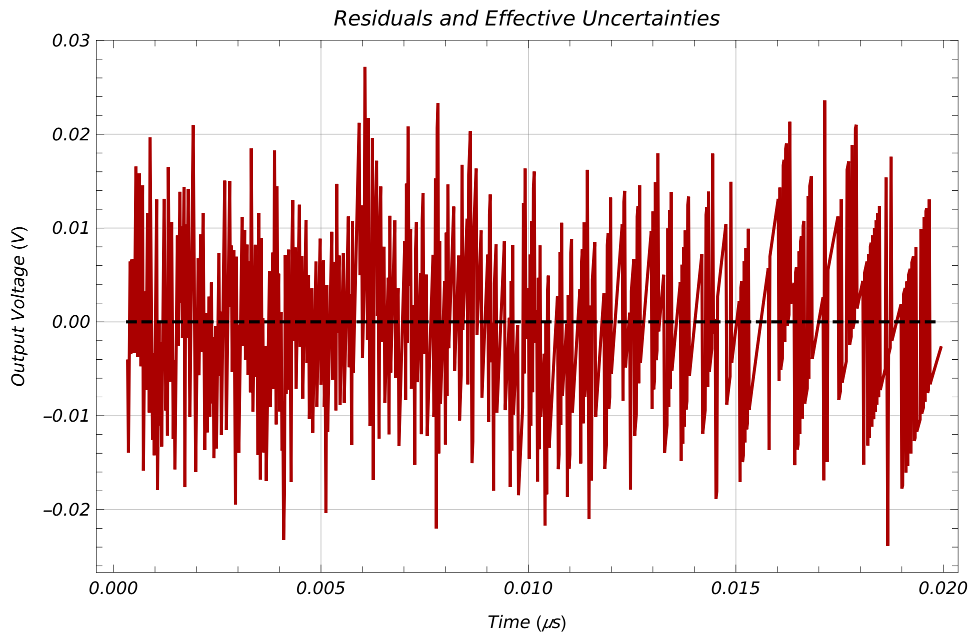
Out[∘]=
```
DecayingExponentialLFit[ ] fits data with:
y = a e^(b x) + c + d (x - xmin)
Important: b must be negative (i.e. decaying exponential).

Set the value of the variable:  region1
to the x-location where the linear background becomes dominant before
  calling this function.  It must be true that: xmin < region1 < xmax.

DecayingExponentialLFit[r1] sets region1 = r1 and then does the fit.
```

In[∘]:=
```
With[{x =
   SetX[ LogDataPlot[], Log -> False, Label -> "Set the X value for region1." ]},
 Print[x];

 DecayingExponentialLFit[x]
]
```

In[•]:=
```
LinearDifferencePlot[FrameLabel → {"Time (μs)", "Output Voltage (V)"}]
```

Out[•]=





y(x) = a Exp[b x] + c + d (x − x_{min})

a=	b=	c=	d=	x_{min}=
3.0368	-179.662	-0.0126833	0.560741	0.00034
σ_a=	σ_b=	σ_c=	σ_d=	
0.00814618	0.656363	0.00894566	0.425192	

Std. deviation=
0.00887417

In[70]:=
```
τ = -1/b {1, -Abs[sigb]/b} (* s *);
ScientificForm[τ, 3]
```

Out[71]//ScientificForm=

$\{5.57 \times 10^{-3},\ 2.03 \times 10^{-5}\}$

In[72]:=
```
cap = τ/(5 * 10^6) (* F *);
ScientificForm[cap, 4]
```

Out[73]//ScientificForm=

$\{1.113 \times 10^{-9},\ 4.067 \times 10^{-12}\}$

We obtain a reasonable fit as far as we can tell by the amplitude and distribution of the residuals (we have a single decay curve and therefore cannot perform a goodness-of-fit test). This gives us an estimate of $\tau = 5.56 \pm 0.02$ ms, from which we calculate $C = \frac{\tau}{R} = 1.113 \pm 0.004$ nF. Note that we have an underestimate on the uncertainty, as we do not know the tolerance of the 5MΩ resistance.

Extracting relevant data from oscilloscope traces

Helper functions to process data

In[•]:=
```
(* We write a function to commit fit parameters to a list for each dataset. *)

(* Initialize list *)
fitParam = {};

(* We will create a list with the structure
 {V₀,{{a, b, siga,  sigb # below Vₑ},{a, b, siga, sigb # above Vₑ}}}.
  IMPORTANT: Need to start inputting values with parameters below Vₑ,
then above Vₑ. The reg argument can be either 'l' for below Vₑ,
or 'm' for above Vₑ. *)
addValues[v0_, reg_] := Module[
   {index},
   If[
    ToString[reg] == "l",
    fitParam = Join[fitParam, {{v0, {{a, b, siga, sigb}, {}}}}]
     ],
    If[
     ToString[reg] == "m",
     (index = Position[fitParam[[All, 1]], v0][[1]][[1]];
      fitParam[[index]][[2]][[2]] = {a, b, siga, sigb})
    ]
   ]
```

```
  ];

(* We also create a list for the Vp data. In order to find Vp,
we smooth the dataset to get rid of high-
 frequency electrical noise by performing a moving average,
then select the maximum value. *)

(* Initialize list *)
vpVal = {};

(* We create a list with structure {V0,Vp} *)
vpValues[filename_, v0_] := Module[
   {data, dataAvg, maxIndex, max},
   (
    LoadTekFile[filename];
    Pause[3];
    SaveFile[StringReplace[filename, "csv" → "dat"]];
    Pause[3];
    data = Import[StringReplace[filename, "csv" → "dat"]];
    dataAvg = MovingAverage[data[[All, 2]], 20];
    Export[StringReplace["avg_%",
      "%" → StringReplace[filename, "csv" → "dat"]], dataAvg];
    maxIndex = Ordering[dataAvg, -1][[1]];
    max = dataAvg[[maxIndex]];
    vpVal = Join[vpVal, {{v0, max}}];
   )
  ];
```

Processing data for analysis

In[∘]:=
```
(* We use our functions to store the data we pull from the fits
 of long and short timescale data for different bias voltages. *)

(* We start with the Ve data,
looping through the following commands for every dataset. *)
```

In[∘]:=
```
With[ {name = SystemDialogInput["FileOpen", {DataFileName,
      {"Tek waveform files" -> {"*.csv", "*.tsv"}, "all files" -> {"*"}}}]},
 If[ name =!= $Canceled,
  LoadTekFile[name]]
]
```

In[•]:=
```
With[{x = SetXRange[LinearDataPlot[], Log -> False,
    Label -> "Set the X values for the range you wish to keep."]},
 Print[x];

 XRangeKeep[Sequence @@ x]
]
```

In[•]:=
```
LinearFit[]
```

In[•]:=
```
addValues[780, m]
```

In[•]:=
```
fitParam
```

Out[•]=
```
{{760, {{0.14211, 87988.4, 0.000164461, 196.71},
    {0.197137, 61789.1, 0.00289608, 1169.96}}},
 {770, {{0.148238, 131204., 0.000222133, 362.729},
    {0.263943, 87402.8, 0.00241459, 959.95}}},
 {780, {{0.122872, 200817., 0.00040219, 532.662},
    {0.253494, 138604., 0.00536139, 2236.4}}}}
```

In[•]:=
```
SetDirectory["Geiger Muller"]
```

In[•]:=
```
(* For safekeeping *)
Export["fitParam.dat", fitParam];

(* Now for the Vp data *)
```

In[•]:=
```
files = {{760, "760_long.csv"}, {770, "770_long.csv"}, {780, "780_long.csv"}};
vpValues[#[[2]], #[[1]]] & /@ files
```

In[•]:=
```
vpVal
```

Out[•]=
```
{{760, 0.9252}, {770, 1.3836}, {780, 1.902}}
```

In[•]:=
```
(* For safekeeping *)
Export["vpVal.dat", vpVal];
```

Extracting V_e , V_p and $\frac{dV_{out}}{dt}$ data from processed traces

In[∘]:=
```
(* We create a list with structure {V0,Ve,σVe} by calutating Ve from our
 fit data and propagating uncertainties. We Flatten fitParam to make the
 uncertainty propagation more readable by avoiding messy indexing. *)
veInt = Partition[Flatten[fitParam], 9];
veVal = Module[
  {ve, a1, b1, siga1, sigb1, a2, b2, siga2, sigb2},
  (

   a1 = 2; b1 = 3; siga1 = 4; sigb1 = 5; a2 = 6; b2 = 7; siga2 = 8; sigb2 = 9;
   Table[
    {
     veInt[[i]][[1]],
     ve = (veInt[[i]][[a1]] * veInt[[i]][[b2]] - veInt[[i]][[a2]] * veInt[[i]][[b1]]) /
       (veInt[[i]][[b2]] - veInt[[i]][[b1]]),
     ve * (Sqrt[(Sqrt[(veInt[[i]][[a1]] * veInt[[i]][[b2]]
                (Sqrt[(veInt[[i]][[siga1]] / veInt[[i]][[a1]])^2 + (veInt[[i]][[sigb2]] / veInt[[i]][[b2]])^2]))^2 +
              (veInt[[i]][[a2]] * veInt[[i]][[b1]] (Sqrt[(veInt[[i]][[siga2]] / veInt[[i]][[a2]])^2 +
                  (veInt[[i]][[sigb1]] / veInt[[i]][[b1]])^2]))^2] / (veInt[[i]][[a1]] *
                veInt[[i]][[b2]] - veInt[[i]][[a2]] * veInt[[i]][[b1]]))^2 +
          (Sqrt[(veInt[[i]][[sigb2]] / veInt[[i]][[b2]])^2 + (veInt[[i]][[sigb1]] / veInt[[i]][[b1]])^2])^2])
    },
    {i, 1, Length[veInt]}
   ]
  )
 ]
```

Out[∘]=
```
{{760, 0.326914, 0.0132689}, {770, 0.494825, 0.00996239}, {780, 0.544507, 0.0200946}}
```

In[•]:=
```
(* We create a list with structure {V0,dVout/dt,σdVout/dt}. *)
dvOut = Module[
  {b2, sigb2},
  (
   b2 = 7; sigb2 = 9;
   Table[
    {veInt[[i]][[1]], veInt[[i]][[b2]], veInt[[i]][[sigb2]]},
    {i, 1, Length[veInt]}
   ]
  )
 ]
```

Out[•]= {{760, 61789.1, 1169.96}, {770, 87402.8, 959.95}, {780, 138604., 2236.4}}

```
(* We already have a list with structure {V0,Vp}. We only have one sweep
 for each bias voltage and we did not obtain the peak value using a fit,
therefore we do not have accurate, independent estimates for the uncertainty
 in Vp. To attempt a better error propagation in the following analysis,
we simply observe the standard deviation of the fit for the exponential
 decay trace to be ~0.0089 V and use this as a somewhat arbitrary
 estimate of the uncertainty in our values for Vp. As mentioned *)
```

In[•]:=
```
vpValUnc = Table[Flatten[{vpVal[[i]], 0.0089}], {i, 1, Length[vpVal]}]
```

Out[•]= {{760, 0.9252, 0.0089}, {770, 1.3836, 0.0089}, {780, 1.902, 0.0089}}

Determining r_{s0}

We calculate r_{s0} from our data and propagate uncertainties using $r_{s0} = a(b/a)^{V_e/V_p}$.

In[•]:=
```
(* We define a function to calculate rs0
 for each V0 from the processed datasets above. *)
rs0[veUnc_, vpUnc_] :=
 Module[
  {aDim, bDim, rs0List},
  (
   aDim = 0.3135 * 10^-3 (* m *); bDim = 7.62 * 10^-3 (* m *);
   rs0List = aDim (bDim / aDim)^(veUnc[[2]]/vpUnc[[2]])
     {1, Log[bDim / aDim] (veUnc[[2]]/vpUnc[[2]]) Sqrt[(veUnc[[3]]/veUnc[[2]])^2 + (vpUnc[[3]]/vpUnc[[2]])^2]};
   {vpUnc[[1]], rs0List[[1]], rs0List[[2]]}
  )
 ]
```

In[•]:=
```
rs0Val = Table[rs0[veVal[[i]], vpValUnc[[i]]], {i, 1, Length[vpVal]}]
```

Out[•]=
```
{{760, 0.000967994, 0.000045523},
 {770, 0.000981341, 0.0000236684}, {780, 0.000781524, 0.0000265562}}
```

In[•]:=
```
rs0Table =
  Join[{{"V0(V)", "rs0  (10^-4 m)"}}, Table[{rs0Val[[i]][[1]], StringReplace["%1 ± %2",
      {"%1" → ToString[NumberForm[rs0Val[[i]][[2]] * 10^4, 3]], "%2" → ToString[
         NumberForm[rs0Val[[i]][[3]] * 10^4, 2]]}]}, {i, 1, Length[rs0Val]}]];
Grid[rs0Table, Alignment → Left, Spacings → {2, 1}, Frame → All,
 ItemStyle → "Text", Background → {{Gray, None}, {LightGray, None}}]
```

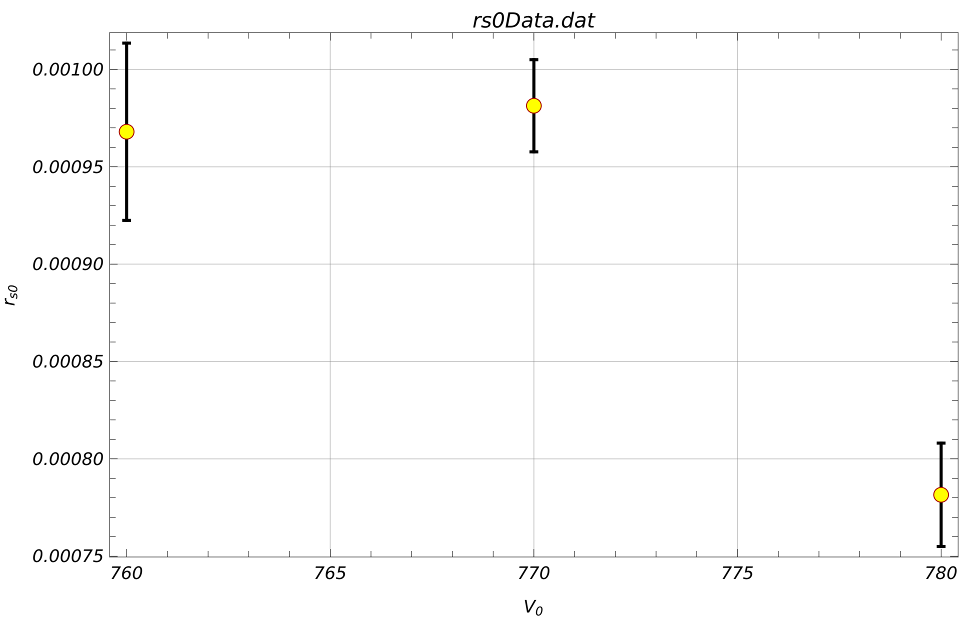
Out[•]=

V_0(V)	r_{s0} (10^{-4} m)
760	9.68 ± 0.46
770	9.81 ± 0.24
780	7.82 ± 0.27

In[•]:=
```
Export["rs0Data.dat", rs0Val];
```

In[•]:=
```
LinearDataPlot[FrameLabel → {"V0", "rs0"}]
```

Out[•]=



Because of the scarcity of the data, and the fact that we only have rough estimates on the uncertainties (since we only have a single trace for each data point), it is difficult to infer any sort of relation between V_0 and r_{s0}. Even disregarding those issues, we do not observe a clear trend in the table above or by plotting the three data points.

Determining mobility of Ne^+

In[140]:=
```
(* We use a package I wrote to help with propagation of uncertainties. *)
<< SimpleErrorPropagation`
```

In[142]:=
$$\text{errorFunction} = \text{PropagateUncertainties}\left[\frac{(\text{rs0 Log}[b/a])^2}{\text{vP}\left(\text{v0} - \frac{\text{cp}}{\text{cd}}\left(\frac{1}{2}\text{vP} - \text{vE}\right)\right)}\text{dVOut}, \{\text{rs0}, \text{vP}, \text{cp}, \text{vE}, \text{dVOut}\}\right] \text{ // Simplify}$$

Out[142]=
$$\sqrt{\left(\left(1.50903\times10^{25}\,\text{cp}^2\,\text{dVOut}^2\,\text{rs0}^4\,\text{sigmaVE}^2\,\text{vP}^2 + 1.50903\times10^{25}\,\text{dVOut}^2\,\text{rs0}^4\,\text{sigmaCp}^2\left(\text{vE}-\frac{\text{vP}}{2}\right)^2\text{vP}^2 + 102.011\,\text{rs0}^4\,\text{sigmaDVOut}^2\,\text{vP}^2\left(3.84615\times10^{11}\,\text{cp vE}+\text{v0}-1.92308\times10^{11}\,\text{cp vP}\right)^2 + 408.042\,\text{dVOut}^2\,\text{rs0}^2\,\text{sigmaRsO}^2\,\text{vP}^2\left(3.84615\times10^{11}\,\text{cp vE}+\text{v0}-1.92308\times10^{11}\,\text{cp vP}\right)^2 + 1.50903\times10^{25}\,\text{dVOut}^2\,\text{rs0}^4\,\text{sigmaVP}^2\left(1.\,\text{cp vE}+2.6\times10^{-12}\,\text{v0}-1.\,\text{cp vP}\right)^2\right)\Big/\left(\text{vP}^4\left(3.84615\times10^{11}\,\text{cp vE}+\text{v0}-1.92308\times10^{11}\,\text{cp vP}\right)^4\right)\right)}$$

In[143]:=
$$\mu[\text{v0\_}, \text{vP\_}, \text{vE\_}, \text{rs0\_}, \text{cp\_}, \text{dVOut\_}, \text{sigmaVP\_}, \text{sigmaVE\_}, \text{sigmaRsO\_}, \text{sigmaCp\_}, \text{sigmaDVOut\_}] := \left\{\frac{(\text{rs0 Log}[b/a])^2}{\text{vP}\left(\text{v0}-\frac{\text{cp}}{\text{cd}}\left(\frac{1}{2}\text{vP}-\text{vE}\right)\right)}\text{dVOut},\right.$$
$$\left.\sqrt{\left(\left(1.5090314822808658\text{`*\^{}25}\,\text{cp}^2\,\text{dVOut}^2\,\text{rs0}^4\,\text{sigmaVE}^2\,\text{vP}^2 + 1.5090314822808658\text{`*\^{}25}\,\text{dVOut}^2\,\text{rs0}^4\,\text{sigmaCp}^2\left(\text{vE}-\frac{\text{vP}}{2}\right)^2\text{vP}^2 + 102.01052820218655\text{`}\,\text{rs0}^4\,\text{sigmaDVOut}^2\,\text{vP}^2\left(3.846153846153846\text{`*\^{}11 cp vE}+\text{v0}-1.923076923076923\text{`*\^{}11 cp vP}\right)^2 + 408.0421128087462\text{`}\,\text{dVOut}^2\,\text{rs0}^2\,\text{sigmaRsO}^2\,\text{vP}^2\left(3.846153846153846\text{`*\^{}11 cp vE}+\text{v0}-1.923076923076923\text{`*\^{}11 cp vP}\right)^2 + 1.5090314822808658\text{`*\^{}25}\,\text{dVOut}^2\,\text{rs0}^4\,\text{sigmaVP}^2\left(1.\text{` cp vE}+2.6000000000000002\text{`*\^{}-12 v0}-1.\text{` cp vP}\right)^2\right)\Big/\left(\text{vP}^4\left(3.846153846153846\text{`*\^{}11 cp vE}+\text{v0}-1.923076923076923\text{`*\^{}11 cp vP}\right)^4\right)\right)}\right\}$$

In[146]:=
```
mobility =
 Partition[Flatten[Table[{vpValUnc[[i]][[1]], μn[vpValUnc[[i]][[1]] (* v0 *),
     vpValUnc[[i]][[2]] (* vp *), veVal[[i]][[2]] (* ve *), rs0Val[[i]][[2]]
     (* rs0 *), cap[[1]] (* c *), dvOut[[i]][[2]] (* dvOut *),
     vpValUnc[[i]][[3]] (* sigmavp *), veVal[[i]][[3]] (* sigmave *),
     rs0Val[[i]][[3]] (* sigmars0 *), cap[[2]] (* sigmacp *),
     dvOut[[i]][[3]] (* sigmadvOut *)]}, {i, 1, Length[vpValUnc]}]], 3]
```

Out[146]=
```
{{760, 0.000900461, 0.0000869232},
 {770, 0.000896119, 0.0000448031}, {780, 0.000741873, 0.0000528969}}
```

In[147]:=
```
mobilityTable = Join[{{"V₀(V)", "μd (10^-4 m^2V^-1s^-1)"}},
   Table[{mobility[[i]][[1]], StringReplace["%1 ± %2",
      {"%1" → ToString[NumberForm[mobility[[i]][[2]] * 10^4, 3]], "%2" → ToString[
         NumberForm[mobility[[i]][[3]] * 10^4, 2]]}]}, {i, 1, Length[mobility]}]];
Grid[mobilityTable, Alignment → Left, Spacings → {2, 1}, Frame → All,
 ItemStyle → "Text", Background → {{Gray, None}, {LightGray, None}}]
```

Out[148]=

V_0(V)	μ_d (10^{-4} $m^2V^{-1}s^{-1}$)
760	9. ± 0.87
770	8.96 ± 0.45
780	7.42 ± 0.53

We now convert our mobility values to mobilities at standard pressure using the conversion factor determined in the pre-lab.

In[149]:=
```
stdMobility = Table[{mobility[[i]][[1]], mobility[[i]][[2]]/760 * 425,
   mobility[[i]][[3]]/760 * 425}, {i, 1, Length[mobility]}]
```

Out[149]=
```
{{760, 0.000503547, 0.0000486084},
 {770, 0.000501119, 0.0000250544}, {780, 0.000414863, 0.0000295805}}
```

In[152]:=
```
stdMobilityTable = Join[{{"V₀(V)", "μd (10^-4 m^2V^-1s^-1)"}},
   Table[{stdMobility[[i]][[1]], StringReplace["%1 ± %2",
      {"%1" → ToString[NumberForm[stdMobility[[i]][[2]] * 10^4, 3]],
       "%2" → ToString[NumberForm[stdMobility[[i]][[3]] * 10^4, 2]]}]},
    {i, 1, Length[stdMobility]}]];
Grid[stdMobilityTable, Alignment → Left, Spacings → {2, 1}, Frame → All,
 ItemStyle → "Text", Background → {{Gray, None}, {LightGray, None}}]
```

Out[153]=

V_0(V)	μ_d (10^{-4} $m^2V^{-1}s^{-1}$)
760	5.04 ± 0.49
770	5.01 ± 0.25
780	4.15 ± 0.3

We observe very similar mobilities for the bias voltages of 760V and 770V, both being higher than what we would expect for Ne^+ and lower than what we would expect for Br^+. This is somewhat expected for a mixture of ions with different mobilities, although we would need more information to better qualify this statement. The mobility for a bias voltage of 780V, however, appears to be very close to the expected mobility of Ne^+. Again, it is difficult to make any conclusive statements given the scarcity of the data.