Physics 398 EMI May 10, 2001
Professor Steve Errede

Measuring the Effect of Heater Voltage on Cathode Current in a 12AX7 Vacuum Dual-Triode

Physics of Electronic Music Instruments Laboratory,
University of Illinois at Urbana-Champaign

TABLE OF CONTENTS

1. INTRODUCTION AND PROBLEM

In the history of electronic devices, the first half of the 20th century is remembered as the era of vacuum tubes. The first thermionic tube was used by Thomas Edison in 1883, in what became the world's first electronic circuit, to produce a voltage regulator. Vacuum tubes typically consist of at least an anode (plate) and a cathode, housed in an evacuated tube typically made of glass. The cathode is heated by a filament, causing electrons to be generated by boiling off. If the anode (plate) is made positive with respect to the cathode, electrons move to the anode, generating a current. In 1906, DeForest introduced a third electrode, called a control grid. The resulting tube, called a triode, was capable of regulating the relatively large plate current – and thus tube power – by the application of a small voltage and negligible power. This early voltage amplifier virtually made possible the electronics

industry. It is the fundamental form of other vacuum tubes such as the tetrode or pentode, which differ by an additional number of grids.

While the glory days of the tube came to an end with the invention of the transistor and later solid-state devices, the vacuum tube today is still used in audio applications such as guitar amplifiers. The relative ease of circuit modification (or construction) for vacuum-tube applications (compared to hard-circuited transistor designs), as well as the inimitable warmth of the vacuum-tube sound, continues to ensure its popularity among today's amp enthusiasts.

Modern-day measurements of vacuum-tube parameters are sparse and sources are difficult to obtain; most available data is nearly half a century old, utilizing rather vintage equipment that is no longer available today. Nevertheless, today's computer allows much more extensive and accurate determination of those parameters at a fraction of the time and effort required previously, and there is an interest to compare today's measurements with those in the past. One such apparatus available is the TrioPar triode parameter measurement system, constructed previously in the Univ. of Ill. Phys. of Electronic Music Instruments Lab., and which is configured specifically for 12A*7 dual-triode tubes, for example, the 12AX7. The 9-pin 12AX7 has an ideal amplification of 100 and is often used as a preamplifier to feed guitar signal to the amplifier power tubes. Using custom-built tube housing connected to a Lab-PC+ DAQ (direct acquisition) system, variations of cathode current with either grid or cathode currents can be recorded. Through such measurements, it is hoped that the determination of relevant parameters such as resistance, transconductance and amplification factor can be made possible.

It was said above that tube operation depends on the thermionic boiling of electrons from the cathode. The greater the heater filament voltage current, the hotter it will be and hence the greater the boiling off of electrons. This effect is described by the Richardson (1921)-Dushman (1930) Equation. The magnitude of the current density J is given by

$$J = \left|\vec{\mathbf{J}}\right| = AT^2 e^{-\frac{h_0}{T}}$$

where b_0 is about 11600 K and is the temperature equivalent of the (cathode) metal work function, which in tungsten is about 4.5 V. The question arises: how important is temperature dependence in vacuum-tube applications? Specifically, in today's applications, the filament or heater voltage depends directly on the external line voltage, which varies depending on location and time. For example, Dan Finkenstadt, the teaching assistant for this course, was complaining how the voltage in the Univ. of Ill. Physics Lab. suddenly increased to 130V after a recent fire at the power plant!

The purpose of this experiment, then, is to measure the effect of differing heater filament voltages on the cathode current at a fixed grid or plate voltage, specifically for the 12AX7 dual-triode. This would be a stepping-stone to future experiments on determining the optimal operating conditions for today's audio applications. This project was undertaken in part fulfillment for requirements of Physics 398 (Electronic Musical Instruments) at the Univ. of Ill. at Urbana-Champaign, under the supervision of Professor Steve Errede and guidance from teaching assistant Dan Finkenstadt, during the Spring 2001 semester.

2. INTERFACE AND PROGRAMMING

As previously mentioned, the system contains a custom-built tube housing connected to a Lab-PC+ DAQ (direct acquisition) system (see Pikenly). This is controlled by the TrioPar program (currently in version 4.4) which runs on the C-based National Instruments LabWindows/CVI platform. The strength of LabWindows/CVI lies in its adroit data-taking capability, made possible by the huge number of function libraries provided. When it comes to operating instrumentation, LabWindows/CVI also excels because of its flexibility – any programmer who knows the C language can write a program to do almost any type of measurement required.

However, its flexibility is also its limitation when it comes to data analysis, since LabWindows has no dedicated graphing/data analysis platform – if you want a graph, you have to program a function that passes your data as an array to one of LabWindow's plot

functions; if you want a different graph, or use a different dataset, you have to change your code or even write a whole new program at worst! The unwieldiness only increases when it becomes necessary to do data smoothing or curve fitting, transforms, derivatives, etc. Clearly then it becomes far more practical to delegate such analytical tasks to a specialized external program such as Microsoft Excel or Microcal Origin.

Between Excel and Origin, Origin offers a vastly superior platform for scientific analysis, having least-squares fitting, Fourier transforms, etc. as part of its standard functions. Its graphing capabilities are enormous yet fairly intuitive, and like Excel, Origin is able to read data from a large number of formats. The crowning advantage: Origin has its own script language, LabTalk, which can be used to automate almost any function in the program, making repetitious and time-consuming tasks as easy as a single click.

Previously (see Tellez and Leonard), the TrioPar program had been used to export data in tab-delimited ASCII format for graphing in Excel (the TrioPar program itself can plot graphs, but the colored background makes printing, especially on a black and white printer, rather undecipherable). While it was easy to use TrioPar in huge amounts of data-taking, it proved extremely laborious to use the exported data –

> "The data handling for this experiment was more extensive than originally expected and prevented us from doing some of the analysis that we had hoped to do... a lot of time was devoted to the organization of data" *(Tellez and Leonard)*

The problem lies in the way that data was saved in TrioPar (versions 1 to 4 – "old" TrioPar). Each TrioPar saved data set actually contains 24 separate sets of data, 12 from each of the Grid and Plate data families. For the Grid Family, it is Cathode 1 and Cathode 2 voltage versus Grid voltage for plate voltages of 50, 100, 150, 200, 250 and 300V (2 x 6 = 12). Similarly, for the Plate Family, the plate voltage is varied for fixed grid voltages, namely –3, -2.5, -2, -1.5, -1 and –0.5 V. All these points were printed in X-[tab]-Y format, 1 point per line, consecutively for all data sets. Because of the enormous number of points (and each point already averaged over 50 readings – behold the wonder of the

modern computer!), a typical data file would stretch about 9800-10000 lines and be fairly intractable. Comparison between data sets, data manipulation of selective portions, etc. was a Herculean task that only grad students can afford to do with no pay! Additionally, the prospect of having to do all the manipulation over again, should one decide to switch tubes, or even just had his disk eaten by his dog, is enough to prove that an alternative was required!

By now it's obvious the solution lies in LabTalk automation. Besides the original investigative purpose, the second purpose of this project is then to create a more friendly format for saving data in TrioPar; the third purpose is to demonstrate the usage of LabTalk automation in Origin as a superior and student-friendly data manipulation tool.

It wasn't easy. TrioPar 4 was a 4300-line C file! However, after lengthy examination of the code, I realized that there was a lot of repetitious programming, especially in the plot functions and save data functions. Recall that each grid family plot consists of 6 plots of different fixed plate voltages. Basically each was individually "hard-coded" – the only difference between each function call was just the color parameter and array column! I quickly replaced this with a "for"-loop, using an array of color parameters, whose corresponding element was passed for each function call, depending on the index of the iteration. This has the advantage that it's now easier to change a "global" parameter – for example if one decided to plot all dotted instead of lined graphs, he would just have to change a small bit of code once, not 12 times. Similarly I cut repetition in the Save_Data function. The total reduction was a staggering 1000 lines, making compilation time noticeably faster. Besides this, minor cosmetics were done to the plot function: where both cathode 1 and 2 plots were together in the same graph, I changed the latter to a dotted plot so one could differentiate them, and retained the same colors for each dataset (previously all cathode 1 plots were blue and all cathode 2 plots were red!).

Next I turned to changing the save format. I decided to let users be able to choose which format they wanted to plot. The main graphical user interface (GUI) of any LabWindows/CVI program can be easily modified by the LabWindows/CVI application,

and I added the new option – "Table Form" – to the Save Data menu. When the user clicks on that option, the function New_Save_Data, which I wrote, is called. The TrioPar4_4 program, which contains this, is found in the C:\CVI\P398EMI\Data\Wes directory.

For every dataset family, the x-axis (either plate or grid famiiy voltages, depending on the family) has identical values across different families and identical spacing. This immediately lends support to saving the data in table form, where the different datasets of each family are listed as different columns against the same x-axis values in the leftmost column. Because both Cathode 1 and Cathode 2 readings shared the same x-axis values, therefore they were listed side by side – making a total of a 13 (1+6+6) columns by n rows (n is the number of x-axis values, typically about 480) table. Because grid and plate families had different x-axis values I decided to save the different families in different files, for ease of future manipulation. The savings in time was enormous. Where previously it had taken me 30 min to arrange data in plottable format under the old TrioPar, it took under a minute in Origin – after importing the data into a worksheet, just select the relevant columns and click "Plot"! Programming and testing "new" TrioPar (version 4.4) took numerous weeks, but the results speak for themselves.

However, I desired even more saving in time! The nature of my project involves opening, copying relevant columns of and plotting **4** different data files (one for each value of line voltage) for **each** plot family, each with **12** different datasets. The number of permutations of tasks was daunting! I needed to be quickly able to plot either grid or plot families, for either cathode, for either all the datasets (e.g. different plate voltages 50V... 300V) or for an individual set (e.g. 150 V). And that's only for **1** tube! Imagine having to do this for the whole 12A*7 family... or for more than 1 tube of the same kind for averaging purposes... or even to do other things like curve fitting or differentiation on the data.... clearly a challenge that would only be attempted by sleepless graduate students without the power of LabTalk automation.

Hence I set about learning the LabTalk language. While it has a structure similar to C and even has C++-like object extensions, it has been optimized by DOS-like syntax conventions. This allows much better manipulation of strings. LabTalk is much more flexible when it comes to variable substitution compared to C – you can substitute variables anywhere – even in the middle of a command. Also, formalities such as variable declarations are dispensed with – there are only number or string variables!

The well-written LabTalk manual helped to reduce the learning curve, as did my C programming experience. LabTalk scripts are in text format and are interpreted by Origin in situ – you can choose a portion with your mouse for Origin to execute, or even type in commands line by line and have Origin execute them instantly after each line (imagine what a boon this is to programmers during testing!).

A copy of my script, triopar_heater.txt is saved in the C:\Origin41 directory on the TrioPar computer (the code is also available in the appendix). The challenge was in producing something flexible enough for all the parameters I listed above! Firstly, the script asks the user to input the plot family type, and then the file locations of the appropriate plot family datasets (1 for each line voltage, 4 total). The locations are saved in an array, simply a string variable with fields separated by spaces, and then the files are opened. Then the user is asked to input the type of plot required. All inputs undergo rudimentary error checking. The corresponding columns of different data sets are then copied into a new worksheet (eg cathode 1 current vs grid voltage for plate voltage = 50 V... for line voltage of 100, 110, 120 and 130V) and then plotted. If the user wanted all the plots of each dataset eg 50 V to 300 V, this process is then repeated. The user will be finally presented with his desired plot, whose attributes (eg font size, color of plots) he can manually adjust later, without being bogged down in all the repetition required to produce the plots. Maximal flexibility was ensured by not dictating, for example, the names of graph or worksheet windows. So if the user were to run the script again, his previous data and plots would not be erased. This required a lot of string-substitution even in the middle of commands – but this is what LabTalk excels in.

Some of the plots that were produced by this script can be found in the appendix. Note that fatal illegal operation errors may be obtained, especially in manipulating the plate family curves. The cause of these glitches is unknown and seems to lie in the plate family files, since if grid family files were (wrongly) substituted, the program works fine! Also, another frustrating thing is that if the program was debugged by manually causing the main loop to run individually (setting r=1 and then running, and then r=2 and then running... etc) – the error disappears! Nevertheless, especially for the Grid Family files, the time saving over having to manipulate large chunks of data to produce the graphs from the old TrioPar saved data is simply enormous.

3. RESULTS AND ANALYSIS

Examples of some relevant plots are in the Appendix. Firstly the cathode current (for both cathodes) versus grid voltage is plotted with a line voltage of 110V. Next the cathode current (for both cathodes) versus plate voltage is plotted with a line voltage of 100V. These show that increasing the plate and grid voltages respectively will cause the current to increase and verifies the Richardson-Dushman equation.

The plots produced all show qualitatively that increasing the line voltage (hence the heater voltage) causes the cathode current to increase as expected. To investigate more quantitatively this increase, I plotted the proportional increase in cathode current versus the line voltage – actually the filament voltage. The proportional increase was calculated from first scaling all the initial values to 1, and then subtracting 1 from every reading. Unfortunately, this method could not be used for extensive elucidation, because there were only 4 available voltages. Nevertheless it's seen the especially for the bigger voltages, the increase is more linear compared to the smaller voltages, which seems to be more fluctuating, although the proportion of the increase is not as pronounced for bigger voltages.

4. CONCLUSION, FUTURE DIRECTION AND ACKNOWLEDGEMENTS

The Richardson-Dushman law has been qualitatively shown for both the Plate Families and Grid Families. Further experimentation is suggested to investigate the quantitative dependence of cathode current on filament/line voltage (temperature). Such an experiment would involve a greater number of intermediate voltages. Perhaps such an experiment would also examine the dependence at more plate or grid voltages as well.

The programming work was generally successful, resulting primarily in much reduced data turnover and manipulation time. The power of both Origin and LabTalk have been demonstrated. Future direction would include writing new scripts (or modifying the current one) to do more complex operations like differentiation or least squares fitting.

I would like to thank Professor Errede and the T.A. Dan Finkenstadt for their patient help and assistance throughout the course of this project. I have definitely learnt a lot through the experiment as well as during the semester as a whole.

5. References

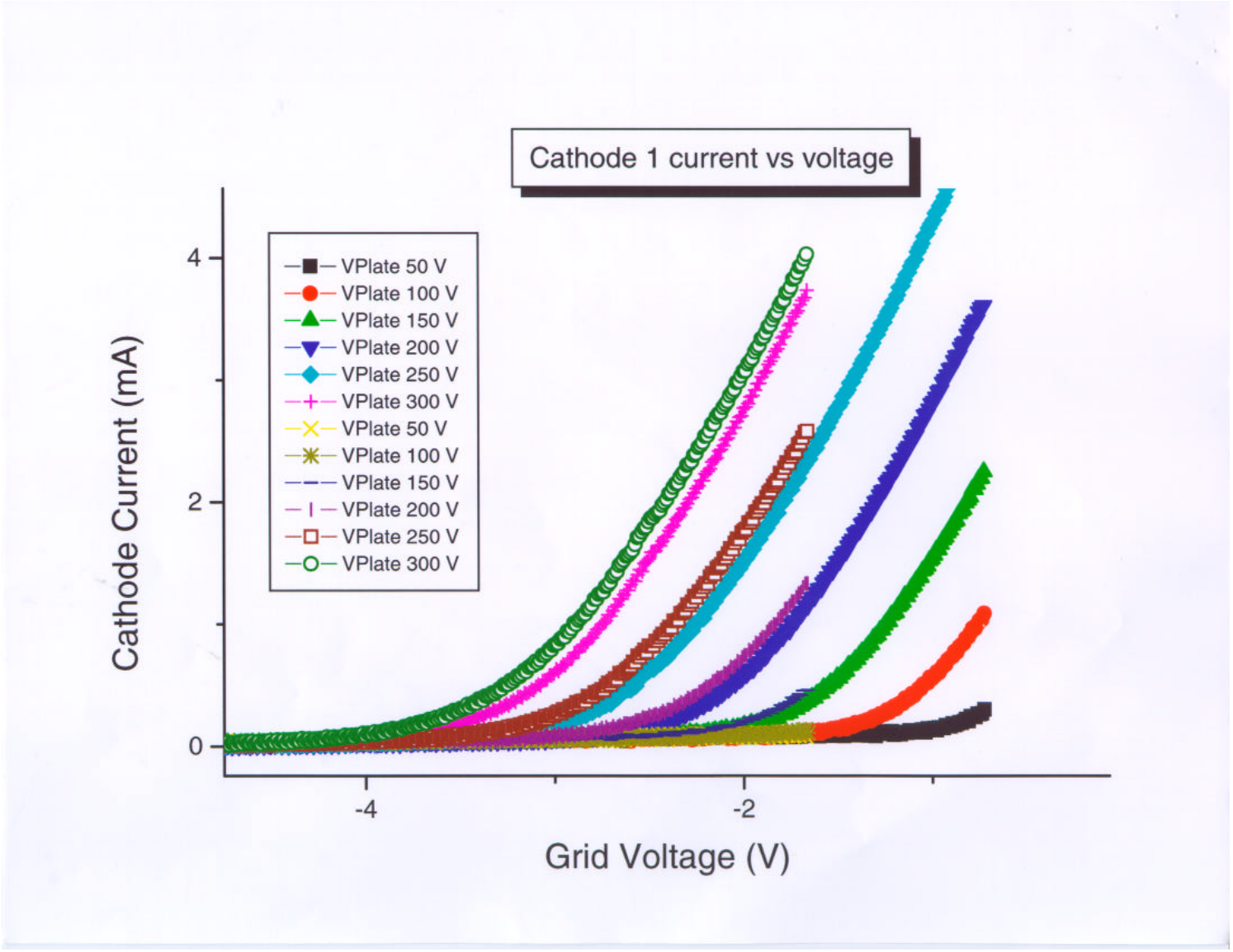
Some of the references that are pertinent to this project are listed as follows:

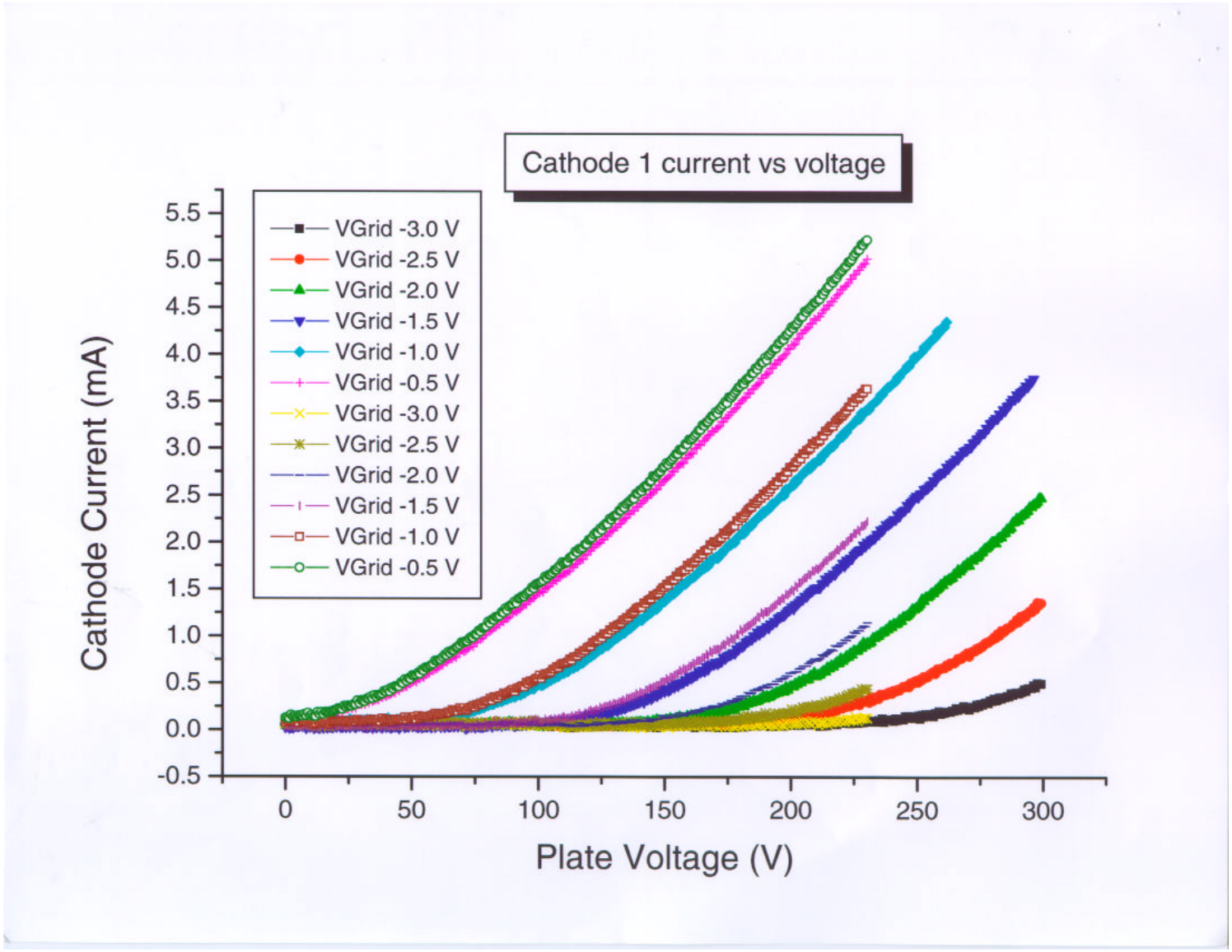
a. Millman, Jacob. "Vacuum-Tube and Semiconductor Electronics". NewYork: McGraw-Hill Book Company (1958)

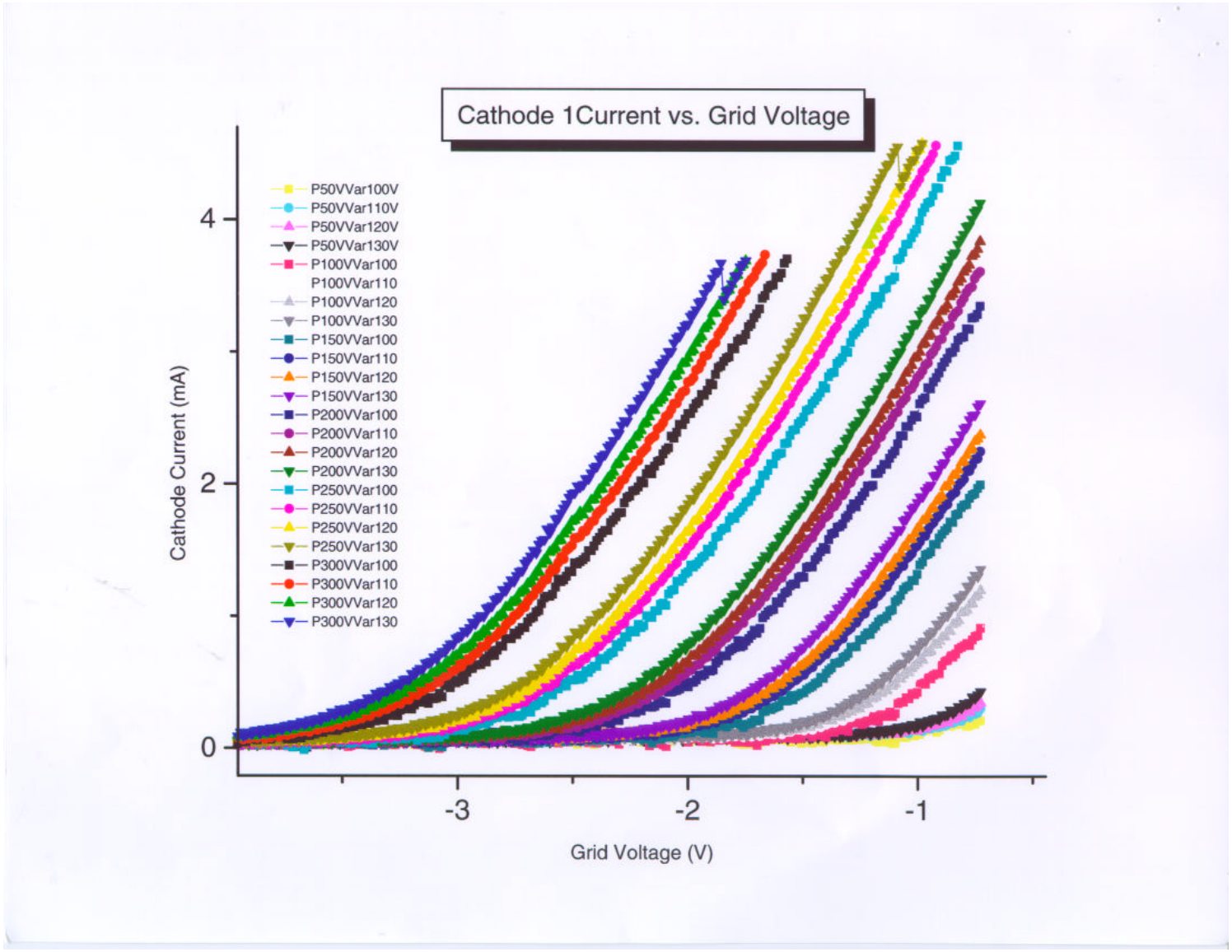
b. Pikelny, Noam. Summer research program report: "Triode Vacuum Tube Laboratory Development". Unpublished, Univ. of Ill. Physics Department, Illinois (2000)

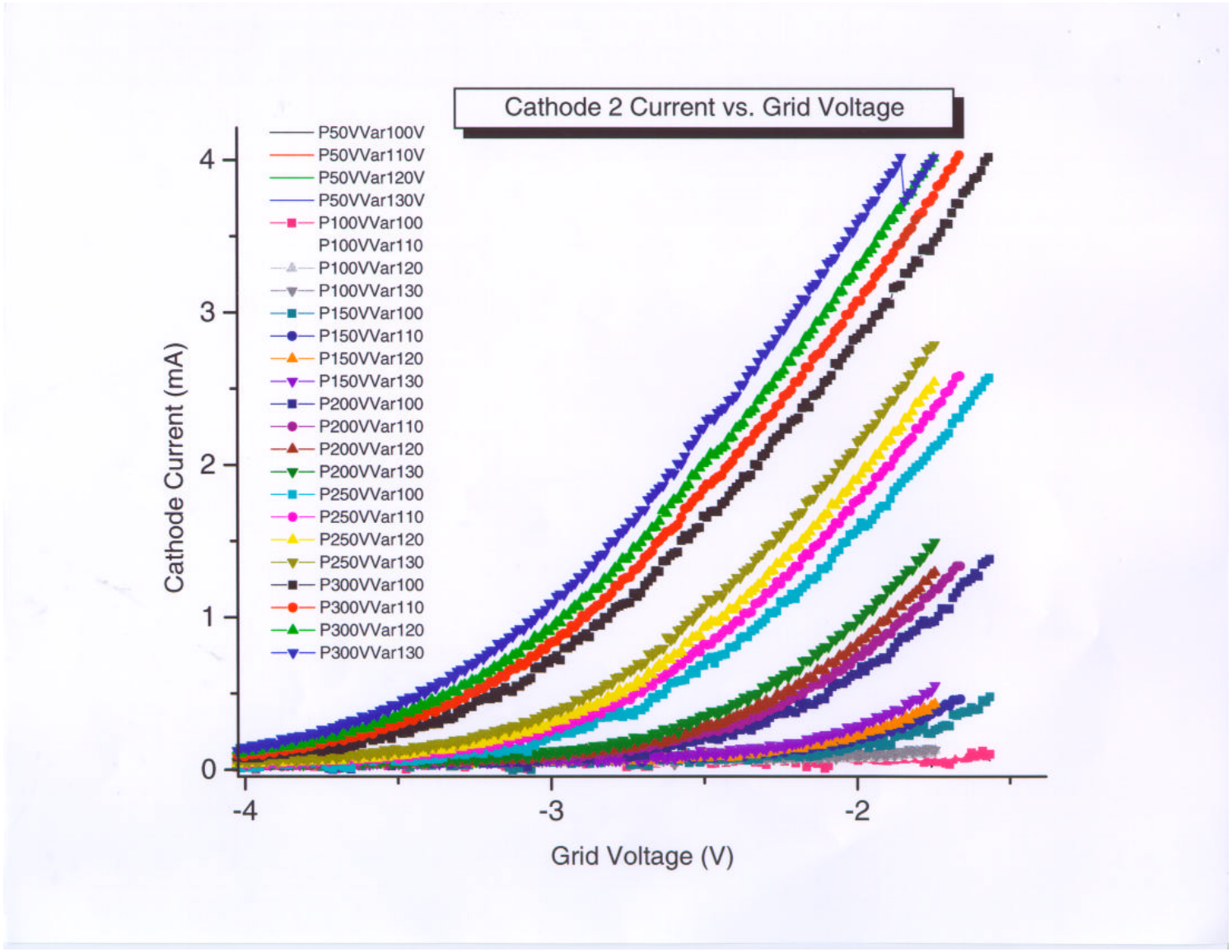
c. Tellez, Maricela and Leonard, Amanda. Physics 398 Final Report: "A Beginning comparison of 12AX7 Vacuum Tubes by Brand Name". Unpublished, Univ. of Ill. Physics Department, Illinois (2000)

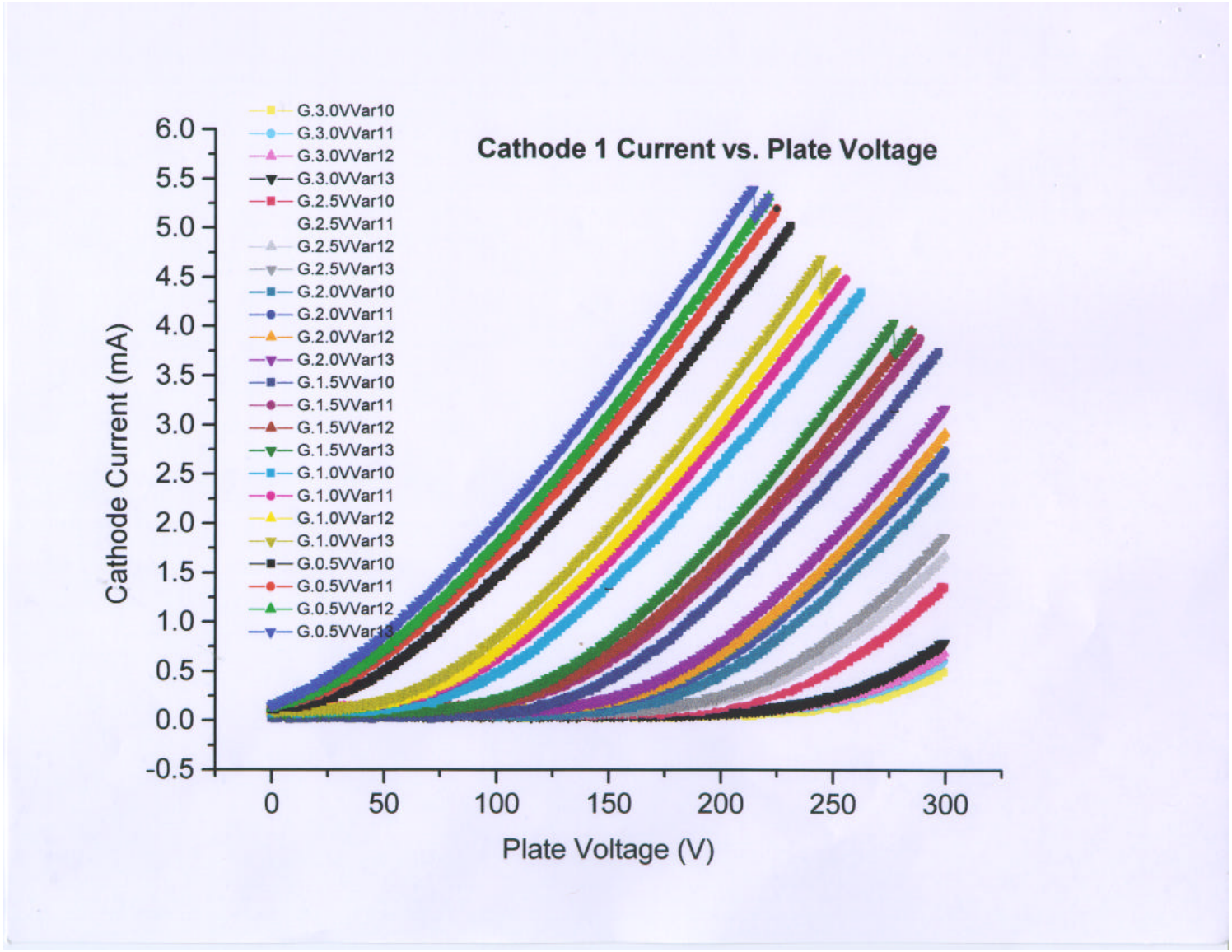
d. Errede, Steve. Physics 398 Lecture Notes: "Thermionic Emission of Electrons". Univ. of Ill. Physics Department, Illinois (2001)

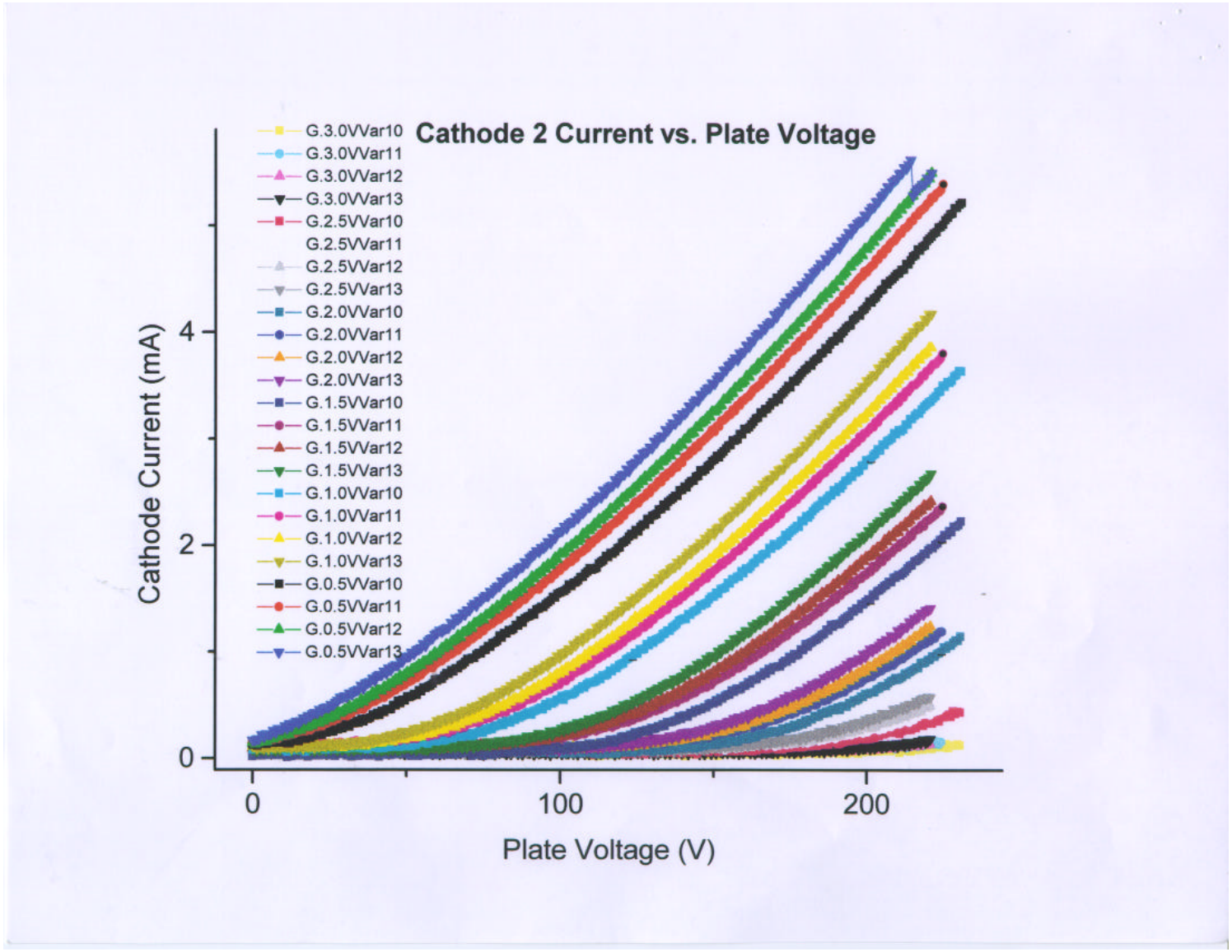

Cathode 1 current vs voltage
VPlate 50 V
VPlate 100 V
VPlate 150 V
VPlate 200 V
VPlate 250 V
VPlate 300 V
VPlate 50 V
VPlate 100 V
VPlate 150 V
VPlate 200 V
VPlate 250 V
VPlate 300 V
Cathode Current (mA)
4
2
0
-4
-2
Grid Voltage (V)

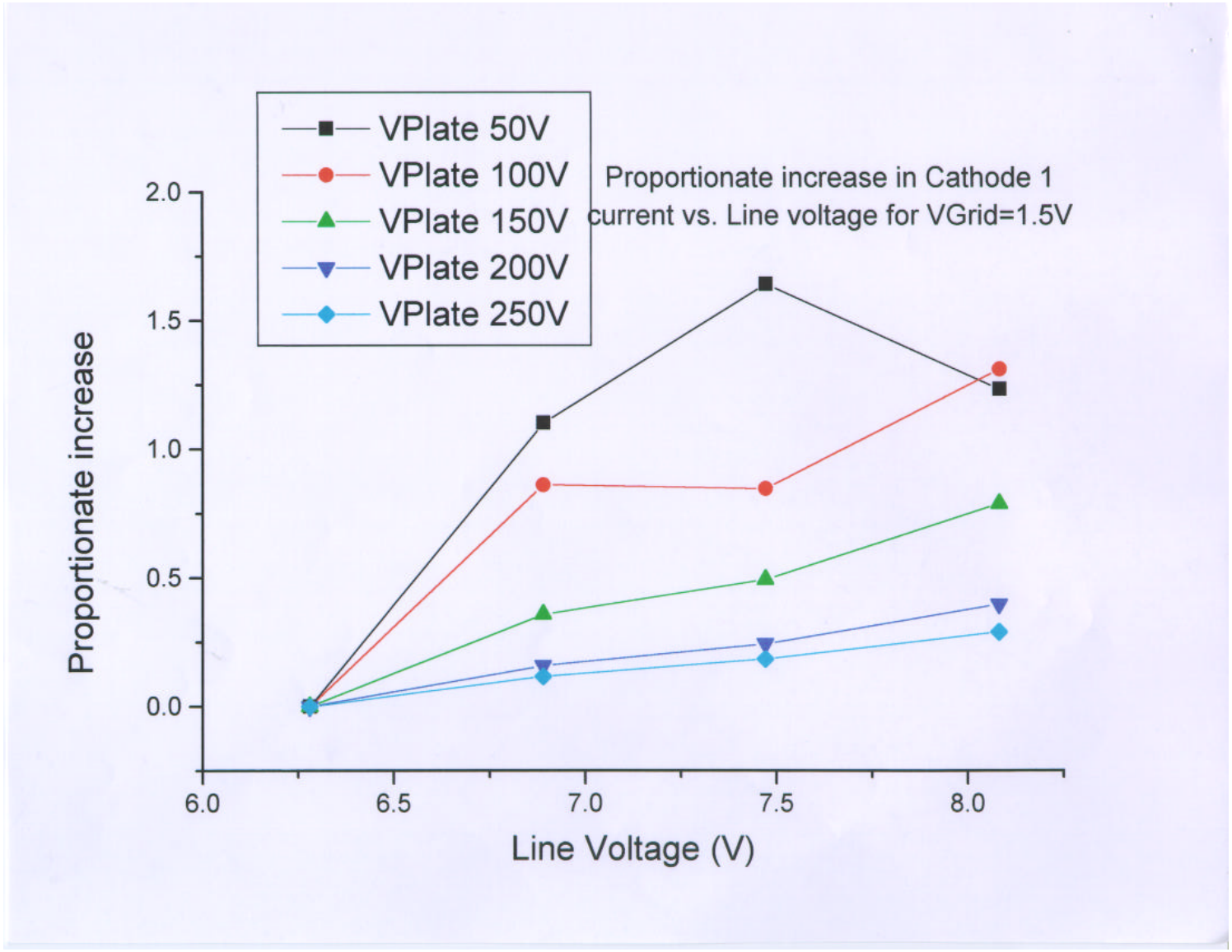

Cathode 1 current vs voltage
VGrid -3.0 V
VGrid -2.5 V
VGrid -2.0 V
VGrid -1.5 V
VGrid -1.0 V
VGrid -0.5 V
VGrid -3.0 V
VGrid -2.5 V
VGrid -2.0 V
VGrid -1.5 V
VGrid -1.0 V
VGrid -0.5 V
Cathode Current (mA)
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
0
50
100
150
200
250
300
Plate Voltage (V)

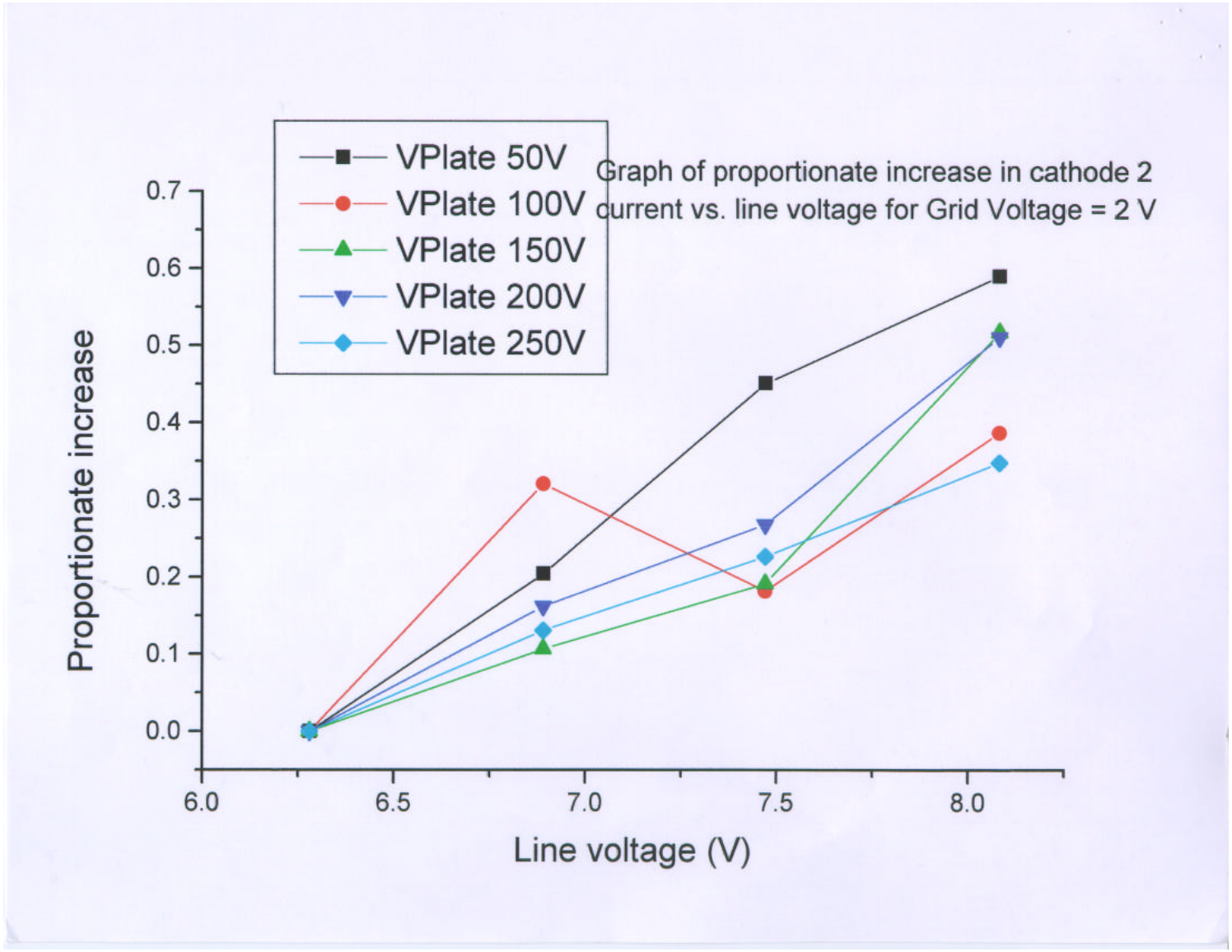

Cathode 1Current vs. Grid Voltage
P50VVar100V
P50VVar110V
P50VVar120V
P50VVar130V
P100VVar100
P100VVar110
P100VVar120
P100VVar130
P150VVar100
P150VVar110
P150VVar120
P150VVar130
P200VVar100
P200VVar110
P200VVar120
P200VVar130
P250VVar100
P250VVar110
P250VVar120
P250VVar130
P300VVar100
P300VVar110
P300VVar120
P300VVar130
Cathode Current (mA)
0
2
4
-3
-2
-1
Grid Voltage (V)

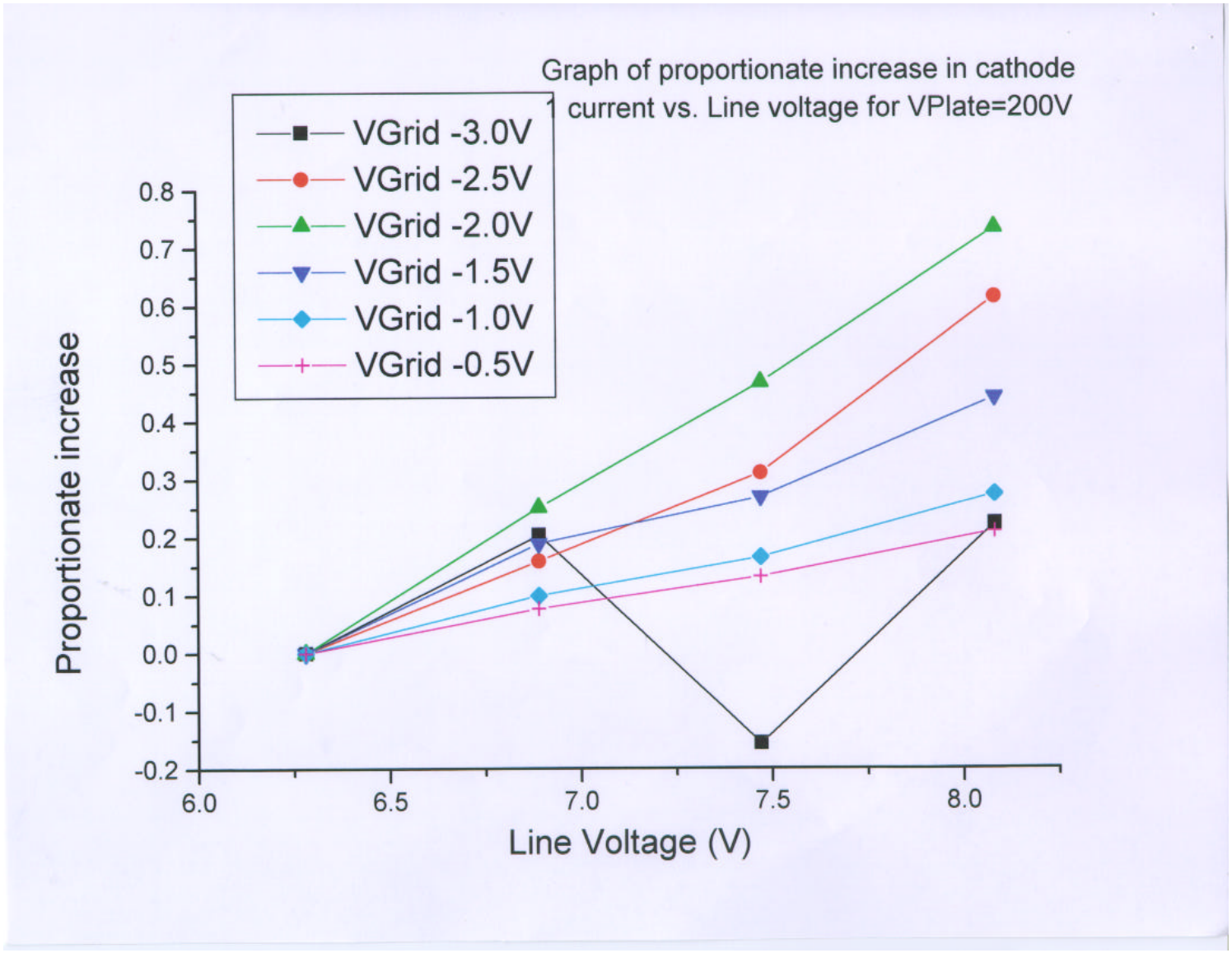

Cathode 2 Current vs. Grid Voltage
P50VVar100V
P50VVar110V
P50VVar120V
P50VVar130V
P100VVar100
P100VVar110
P100VVar120
P100VVar130
P150VVar100
P150VVar110
P150VVar120
P150VVar130
P200VVar100
P200VVar110
P200VVar120
P200VVar130
P250VVar100
P250VVar110
P250VVar120
P250VVar130
P300VVar100
P300VVar110
P300VVar120
P300VVar130
Cathode Current (mA)
0
1
2
3
4
-4
-3
-2
Grid Voltage (V)

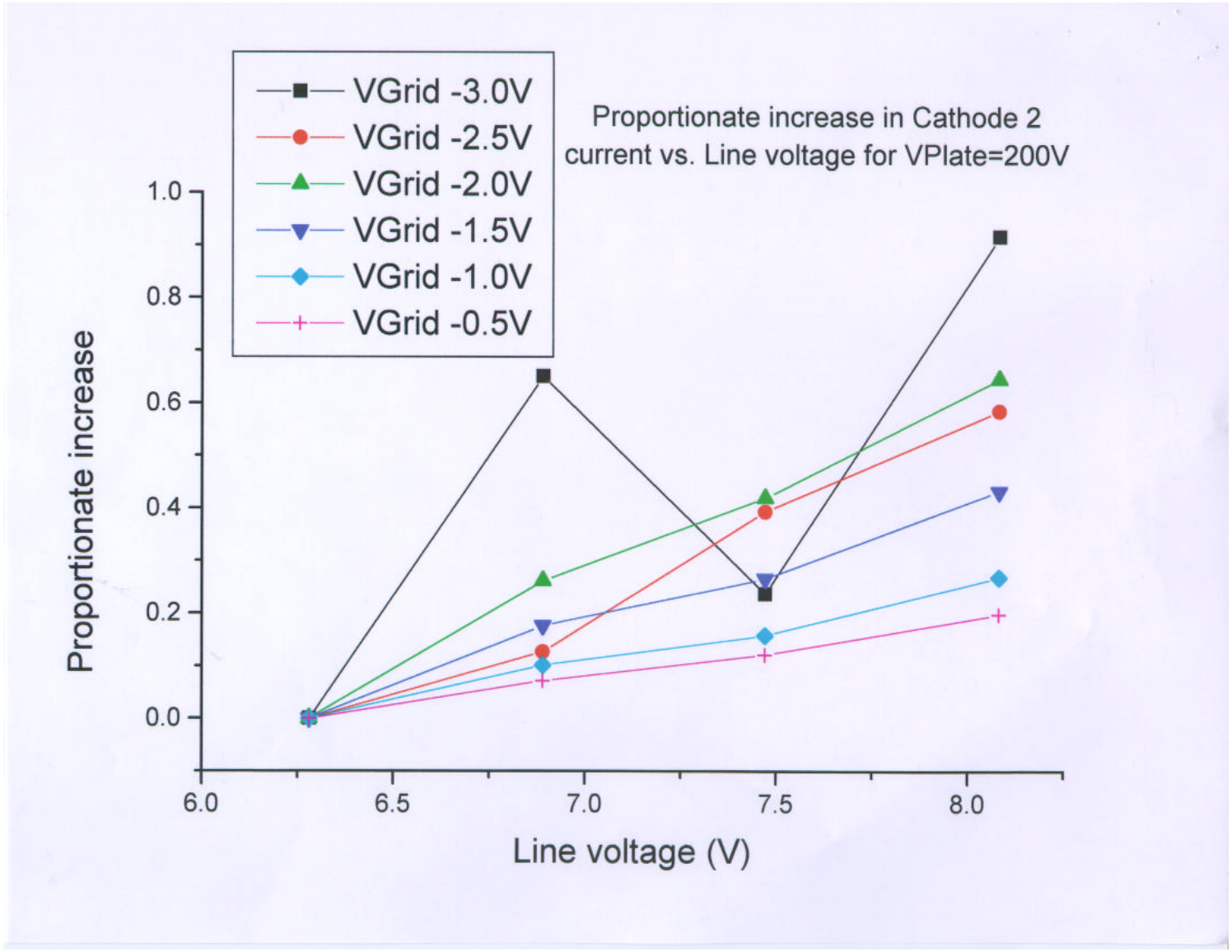

Cathode 1 Current vs. Plate Voltage
Cathode Current (mA)
Plate Voltage (V)
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
0
50
100
150
200
250
300
G.3.0VVar10
G.3.0VVar11
G.3.0VVar12
G.3.0VVar13
G.2.5VVar10
G.2.5VVar11
G.2.5VVar12
G.2.5VVar13
G.2.0VVar10
G.2.0VVar11
G.2.0VVar12
G.2.0VVar13
G.1.5VVar10
G.1.5VVar11
G.1.5VVar12
G.1.5VVar13
G.1.0VVar10
G.1.0VVar11
G.1.0VVar12
G.1.0VVar13
G.0.5VVar10
G.0.5VVar11
G.0.5VVar12
G.0.5VVar13


Cathode 2 Current vs. Plate Voltage
G.3.0VVar10
G.3.0VVar11
G.3.0VVar12
G.3.0VVar13
G.2.5VVar10
G.2.5VVar11
G.2.5VVar12
G.2.5VVar13
G.2.0VVar10
G.2.0VVar11
G.2.0VVar12
G.2.0VVar13
G.1.5VVar10
G.1.5VVar11
G.1.5VVar12
G.1.5VVar13
G.1.0VVar10
G.1.0VVar11
G.1.0VVar12
G.1.0VVar13
G.0.5VVar10
G.0.5VVar11
G.0.5VVar12
G.0.5VVar13
Cathode Current (mA)
0
2
4
0
100
200
Plate Voltage (V)


VPlate 50V
VPlate 100V
VPlate 150V
VPlate 200V
VPlate 250V
Proportionate increase in Cathode 1 current vs. Line voltage for VGrid=1.5V
Proportionate increase
2.0
1.5
1.0
0.5
0.0
6.0
6.5
7.0
7.5
8.0
Line Voltage (V)


Graph of proportionate increase in cathode 2 current vs. line voltage for Grid Voltage = 2 V
VPlate 50V
VPlate 100V
VPlate 150V
VPlate 200V
VPlate 250V
Proportionate increase
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
6.0
6.5
7.0
7.5
8.0
Line voltage (V)


Graph of proportionate increase in cathode current vs. Line voltage for VPlate=200V
VGrid -3.0V
VGrid -2.5V
VGrid -2.0V
VGrid -1.5V
VGrid -1.0V
VGrid -0.5V
Proportionate increase
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
-0.1
-0.2
6.0
6.5
7.0
7.5
8.0
Line Voltage (V)


VGrid -3.0V
VGrid -2.5V
VGrid -2.0V
VGrid -1.5V
VGrid -1.0V
VGrid -0.5V
Proportionate increase in Cathode 2 current vs. Line voltage for VPlate=200V
Proportionate increase
1.0
0.8
0.6
0.4
0.2
0.0
6.0
6.5
7.0
7.5
8.0
Line voltage (V)

APPENDIX 6.11 New_Save_Data FUNCTION CODE

```
void New_Save_Data()
{

	/* This function will save data in Origin-friendly form. */
	int ii, jj, kk, qq;
	int save_cancel;
	int the_file;

	/* These control the column labels on the saved data files */
	int PlateVoltageArray[6]={50,100,150,200,250,300};
	float GridVoltageArray[6]={-3.0, -2.5, -2.0, -1.5, -1.0, -0.5};
	char buffer[50];

	char my_path  [263];

	status = MessagePopup ("Save Data",
                                                    "Your data for the Grid/Plate Families will be saved in 2
separate files.");
	status = MessagePopup ("Grid Family",
                                                    "First, save the GRID FAMILY data.");


	save_cancel = FileSelectPopup ("c:\\cvi\\p398\\data", "*.dat", "*.*",
                                        "Save the GRID FAMILY Data Where?", VAL_SAVE_BUTTON, 0, 0, 1, 1,
my_path);

	if (save_cancel != 0)
		{
		the_file = OpenFile (my_path, 2, 0, 1);

		FmtFile   (the_file, "%s\n",my_path);

		WriteFile (the_file, "\nDAQ PROGRAM: TRIOPAR4.prj \n", 28);

		WriteFile (the_file, "\n\nDate: \n", 9);
		WriteFile (the_file, *curr_date,    10);

		WriteFile (the_file,   "\nTime: \n", 8);
		WriteFile (the_file, *curr_time,     8);

		WriteFile (the_file,   "\n\nNumber of Samples/Data Point: \n", 33);
		FmtFile   (the_file, "%s<%i", npts);

		WriteFile (the_file,     "\nTotal Number of Grid Family Data Points:  \n", 44);
		FmtFile   (the_file, "%s<%i", Gfjmax);

		WriteFile (the_file,     "\nTotal Number of Plate Family Data Points:  \n", 45);
		FmtFile   (the_file, "%s<%i", Pfjmax);

		WriteFile (the_file,     "\nPlate Step Size (Volts):       \n", 33);
		FmtFile   (the_file, "%s<%f", VPlateStep);

		WriteFile (the_file,     "\nGrid Step Size (Volts):       \n", 32);
		FmtFile   (the_file, "%s<%f", VGridStep);

		WriteFile (the_file,     "\nMaximum Plate Power Dissipation (Watts): \n", 43);
		FmtFile   (the_file, "%s<%f", MaxPowerD);

		FmtFile (the_file, "\n\n \t Cathode 1: \t \t \t \t \t \tCathode 2: ");

		FmtFile (the_file, "%s<\n Vgrid (V) \t");

		/* One set of identifiers for each cathode.*/
		for (ii=0; ii<=5; ii++)
			{
			FmtFile (the_file, "VPlate %i V \t",  PlateVoltageArray[ii]);
			}
		for (ii=0; ii<=5; ii++)
```

```
                    {
                    FmtFile (the_file, "VPlate %i V \t",  PlateVoltageArray[ii]);
                    }

        /* First write the VGrid, which is the x-axis. Then the cathode current values */
                for (jj=0; jj<=GfCountHolder[0]; jj++)
                    {
                    FmtFile (the_file, "\n");
                    FmtFile (the_file, "%s<%f[w15]\t", VGrid [0][jj]);

                    for (kk=0; kk<=5; kk++)
                        {
                        if (GfIK0 [kk][jj] != 0)
                            {
                            FmtFile (the_file, "%s<%f[w15]\t", GfIK0 [kk][jj]);
                            }
                        else
                            {
                            WriteFile (the_file, "\t",2);
                            }
                        }
                    for (kk=0; kk<=5; kk++)
                        {
                        if (GfIK1 [kk][jj] != 0)
                            {
                            FmtFile (the_file, "%s<%f[w15]\t", GfIK1 [kk][jj]);
                            }
                        else
                            {
                            WriteFile (the_file, "\t", 2);
                            }
                        }
                    }
                CloseFile (the_file);
                }


        status = MessagePopup ("Plate Family",
                                                "Next, save the PLATE FAMILY data.");


        save_cancel = FileSelectPopup ("c:\\cvi\\p398\\data", "*.dat", "*.*",
                                "Save the PLATE FAMILY Data Where?", VAL_SAVE_BUTTON, 0, 0, 1, 1,
my_path);


  /* Repeat above, but now for the Plate family. */
        if (save_cancel != 0) {
                the_file = OpenFile (my_path, 2, 0, 1);

                FmtFile   (the_file, "%s\n",my_path);

                WriteFile (the_file, "\nDAQ PROGRAM: TRIOPAR4.prj \n", 28);

                WriteFile (the_file, "\n\nDate: \n", 9);
                WriteFile (the_file, *curr_date,    10);

                WriteFile (the_file,   "\nTime: \n", 8);
                WriteFile (the_file, *curr_time,     8);

                WriteFile (the_file,   "\n\nNumber of Samples/Data Point: \n", 33);
                FmtFile   (the_file, "%s<%i", npts);

                WriteFile (the_file,     "\nTotal Number of Grid Family Data Points:  \n", 44);
                FmtFile   (the_file, "%s<%i", Gfjmax);

                WriteFile (the_file,     "\nTotal Number of Plate Family Data Points:  \n", 45);
                FmtFile   (the_file, "%s<%i", Pfjmax);

                WriteFile (the_file,     "\nPlate Step Size (Volts):       \n", 33);
```

```
                FmtFile   (the_file, "%s<%f", VPlateStep);

                WriteFile (the_file,     "\nGrid Step Size (Volts):       \n", 32);
                FmtFile   (the_file, "%s<%f", VGridStep);

                WriteFile (the_file,     "\nMaximum Plate Power Dissipation (Watts): \n", 43);
                FmtFile   (the_file, "%s<%f", MaxPowerD);

                FmtFile (the_file, "\n\n \t Cathode 1: \t \t \t \t \tCathode 2: ");

                FmtFile (the_file, "%s<\n VPlate (V) \t");

                /* One set of identifiers for each cathode.*/
                for (ii=0; ii<=5; ii++)
                        {
                        FmtFile (the_file, "VGrid %f[p1] V \t",  GridVoltageArray[ii]);
                        }
                for (ii=0; ii<=5; ii++)
                        {
                        FmtFile (the_file, "VGrid %f[p1] V \t",  GridVoltageArray[ii]);
                        }


                for (jj=0; jj<=PfCountHolder[0]; jj++)
                        {
                        FmtFile (the_file, "\n");
                        FmtFile (the_file, "%s<%f[w15]\t", VPlate [0][jj]);

                        for (kk=0; kk<=5; kk++)
                                {
                                if (PflK0 [kk][jj] != 0 )
                                        {
                                        FmtFile (the_file, "%s<%f[w15]\t", PflK0 [kk][jj]);
                                        }
                                else
                                        {
                                        WriteFile (the_file, "\t", 2);
                                        }
                                }
                        for (kk=0; kk<=5; kk++)
                                {
                                if (PflK1 [kk][jj] != 0)
                                        {
                                        FmtFile (the_file, "%s<%f[w15]\t", PflK1 [kk][jj]);
                                        }
                                else
                                        {
                                        WriteFile (the_file, "\t", 2);
                                        }
                                }
                        }
                                CloseFile (the_file);

                Sound(550,0.1);
                Sound(440,0.1);
                Sound(550,0.1);

        status = MessagePopup (" SAVE RAW DATA ",
                        " Data successfully written to specified file in specified area. Please back up/copy/save your
files to floppy disk on the A: drive, then please DELETE your files in the c:\\cvi\\p398\\data area when no longer needed!!!
Thanks!!! ");
                }
}
```

APPENDIX 6.12: ORIGIN SCRIPT

```
/*****************************************************/
/* Physics 398 - Electronic Musical Instruments Lab */
/*               Professor Steve Errede              */
/* TrioPar4 Origin Companion Script                  */
/* Variation of Cathode current with heater voltage */
/*                                                   */
/* First written by Wesley Cheong May 7, 2001        */
/* Last modified May 10 2001 1000h                   */
/*                                                   */
/* Note: To run this program you require 4 of EACH  */
/* of Plate OR Grid Family data saved in TrioPar4    */
/* under the "Table form" option.                    */
/*****************************************************/

/* Variable initialization */
/* For system variables eg %A, %B, %H etc. see Labtalk manual p31.*/
Cathode=0;          #To choose which cathode to plot.
k=0;                #Use to determine whether to plot individually or all plots.
l=0;                #Use to reference the right voltage column to plot.
%K="1 2 3 4 5 6";       #Options for the voltage column.
%L="";                  #Stores the name of the 4 different saved data
worksheets.
%M="-3.0 -2.5 -2.0 -1.5 -1.0 -0.5"; #Grid Voltages for Plate family
%N="50 100 150 200 250 300"; #Plate Voltages for Grid family
%O="100 110 120 130";   #Output voltages from variac
%P="";                  #Stores names of all the cross-variac voltage data.
%Q="";                  #Stores the name of the graph window.
%R="";                  #Stores the title of the plot.

/** Begin **/

type -b Welcome to Origin Script program!
This script was written by Wesley Cheong.;

/** Selecting the type of Plot Family **/
for (i=1; i>=1; i++)
   {getString (Select the type of plot Family: 1=Grid, 2=Plate) (1) Enter Plot
Type;
   if (%B==1||%B==2) break;
   };
PType=%B; #Saves the plot type.

/** Selecting the files - storing the names into %L **/
type -b Choose the 4 corresponding Family Files RESPECTIVELY;
getfilename -m *.*;
for (n=1; n<=4; n++)
{
win -T Data;
getfilename -g $(n);
open -w %A;
%L=%L %H;
};

/** Getting the type of user plot required. Simple error checking **/
for (i=1; i>=1; i++)
```

```
   {getnumber
    (Cathode) Cathode
    (Individual Plots) k:2s
    (Voltage Column) l:K
    (Choose Values to Plot);

   if (Cathode==1||Cathode==2)
      {if (k==0) break;
       else if (l!=0) break;
      };
   };

/** Main loop. If individual plot, will break at end of 1st iteration. **/
for (r=1; r<=6; r++)
   {
   if (k==0) l=r; #Chooses the right column if multiple plots requested.
   win -T Data;
   %P=%P %H; #Saves name of cross-variac data worksheet.
   copy %(%[%L,#l],@Data,1) %(%[%P,#r],@Data,1); #Copies the common X values.
   worksheet -a 3; #Adds columns.

   m=(Cathode-1)*6; #adjusts to get the correct cathode user has indicated.

     /** This loop copies the corresponding cross-variac Y-data. **/
     for (n=1;n<=4;n++)
     {
     win -a %[%L,#n];
     copy %(%H,@Data,1+m+1) %(%[%P,#r],@Data,n+1);
     };

   /** Selects the active cross-variac worksheet and names the columns. **/
   /** According to the right plot family the user has chosen. **/
   if (k==0) win -a %[%P,#l];
   else if (k!=0) win -a %P;

   if (PType==1)
     {worksheet -n 1 VGrid (V)};
   else if (PType==2)
     {worksheet -n 1 VPlate (V)};

   for (n=1;n<=4;n++)
     {
     if (k==0)
          {if (PType==1)
              {worksheet -n (n+1) P %[%N,#l] V Var %[%O,#n]V;}
           else if (PType==2)
              {worksheet -n (n+1) G%[%M,#l]V%[%O,#n]V;}
          }
     else if (k!=0)
           {worksheet -n (n+1) Variac %[%O,#n]V;}
      };

   /** Now plot our data. 1st time round requires a special command. **/
   /** 202 for line and scatter plots. See Labtalk manual p. 128. **/
   worksheet -s 0 0;
   if (k==0)
      {if (r==1)
```

```
            {worksheet -p 202;
             %Q=%H;
            }
         else
            {win -b %Q;
             layer -w %E 0 0 0 0 202;
            }
      }
   else if (k!=0)
      {worksheet -p 202;
       break;
      }
   };

legend;

/** Finally, adding the title and axes to the graph. **/
if (PType==1)
   {label -xb Grid Voltage (V);
    %R=Cathode $(Cathode) Current vs. Grid Voltage;
    if (k!=0) %R=%R\n VPlate = %[%N,#1] V;
   }
else if (PType==2)
   {label -xb Plate Voltage (V);
    %R=Cathode $(Cathode) Current vs. Plate Voltage;
    if (k!=0) %R=%R\n VGrid = %[%M,#1] V;
   }

label -q 4 -s -sa -t %R;
label -s -sa -yl Cathode Current (mA);
```