ExtractModel: Manual and Examples V4.0

Derk Reefman

May 3, 2026

1 Introduction

The user interface to `ExtractModel`is, as a friend explained, quite Spartan. This document tries to explain the interface and modes of operation with a number of examples that hopefully clarify the operation of `ExtractModel`. In the below, examples are provided explaining running `ExtractModel` for most of the supported tube types and corresponding models.

This document describes `ExtractModel` version 4.x.y.

1.1 Installation

`ExtractModel` is a straightforward executable that needs to be run in a `cmd` window, or an `XTerm` under Linux. Hence, installation is easy and is simply copying the executable in a directory from which it can be run. It is advisory to have a PATH variable to that directory such that `ExtractModel` can be run from any directory. In the examples below, it will be assumed this is the case. New to `ExtractModel` 4.x is that there are other files to be installed as well. For Windows, this has been resolved through the creation of an `.msi` installation file. *During the installation, the suggested installation paths should NOT be changed!*

It is most convenient to run `ExtractModel` in the directory that also contains the .utd data files. This will cause the output files produced by `ExtractModel` to also end up in that same directory. New to `ExtractModel` 4.x is that there are other files to be installed as well. For Windows, the `.msi` installation file will install the following:

1. c:\Program Files\DerkR\ExtractModel\bin\EM4.exe
2. c:\Program Files\DerkR\ExtractModel\config\EM.cfg
3. c:\Program Files\DerkR\ExtractModel\bin\Old2NewIni.exe
4. c:\Program Files\DerkR\ExtractModel\bin\PinOut.exe
5. c:\Program Files\DerkR\ExtractModel\lib (folder)
6. c:\Program Files\DerkR\ExtractModel\config\PinOut.cfg
7. c:\Program Files\DerkR\ExtractModel\doc PDF files - manual (this document) and Theory

The `.msi` file will also set the path variable to c:\Program Files\DerkR\ExtractModel\bin. In the `lib` folder, both a current version of collected Spice models (`TubeLib.inc` will be installed, as well as a database `PinOut.db` with tube pinouts and high-level characteristics to allow quick and easy tube testing using the uTracer.

As `ExtractModel` uses `Gnuplot` as its plotting interface, it is important to have `Gnuplot` installed; it does not come with the installation of `ExtractModel`. Gnuplot can be downloaded from `www.gnuplot.info` for many platforms; see this website for installation of the package. `ExtractModel` needs to know how `Gnuplot` can be invoked as a command; for a Windows7/10/11 OS this is typically
`C:\Program Files (x86)\gnuplot\bin\wgnuplot.exe` or (more recently)
`C:\Program Files\gnuplot\bin\wgnuplot.exe`.
Under most Linux variants gnuplot is located in `/usr/bin/gnuplot`.

1.2 Changes from version 3

Apart from several bug fixes, several significant changes have been made to (the architecture of) `ExtractModel`, some of which are visible, and others which are not. The main changes made are:

- Compatibility with uTracer6. Due to the capability of uTracer6 to measure grid current, the `.utd` file format includes a separate column for this current, even if no positive grid voltages have been included;
- The code has been refactored to explicitly calculate the total space current. Whilst this was necessary to further implement the positive grid voltage model it also comes with the fact that other tube models such as pentodes and heptodes, no langer require separate runs (using tube model X or Y in `ExtractModel`3.x).
- A new model ('DerkG') has been added to model grid current for triodes and, in a somewhat approximative way, also pentodes; grid current modelling will not become available for beam tetrodes or other models.
- In the derivation of the DerkE models a mistake was made; for the P (pentode) and B (beam tetrode) models this has been corrected. The new Spice models are denoted by a suffix V4 to distinguish these from the earlier models of `ExtractModel`3 and before. As a result there is an improvement in modeling the knee in Vs=cst curves.
- As the minimization function for more complex models (notably the variable mu pentodes, and to a lesser extent also the the heptode models, often ran into a false / suboptimal minimum, a configuration file has been created. In this file certain parameter limits are provided that help keep the minimization away from these false minima. Some other parameters are included as well.
- The format of the `.ini` file has been changed; no longer is the file a fixed-format file. In order to make transitioning to `ExtractModel`4 easier, a program `Old2NewIni` is provided that converts old formatted `.ini` files into the new format.
- `ExtractModel`3 was compiled using a 32-bit microsoft compiler; `ExtractModel`4 is using a 64-bit GNU fortran compiler. As a result, the code executes faster.
- Several more general bug fixes have been included.

2 PinOut

In order to ease the (frequent) collection of tube data using a uTracer, a simple database application `PinOut` has been developed. It includes a text-oriented database that contains all tube data, and the definition of the descriptions of these tubes. Next to that it offers the option to also display a picture, which is intended to show the relevant digitized snap-shot of the Muiderkring tube description, as collected by Frank Philipse [Phi13] and (freely) available on `https://www.radiomuseum.org/` as low-resolution scans.

The use is either

```
Pinout Check
PinOut EL84
```

When `Check` is provided as argument, `PinOut` will check consistency of the database records with the rules that are given by the sections `[Descriptors]`, `[Pinning]`, and `[Electrodes]`. The second line shows the example where a tube name is provided; in that case `PinOut` will search for the relevant record in the database (following the `[Tubes]` section header), and list that part of the database.

The datafile is structured according to the following sections:

```
[Descriptors]
        Triode
        Pentode         Power straight variable_mu
        Hexode
        Heptode         straight variable_mu_g1 variable_mu_g3
        Oktode
        DoubleTriode    Power equal unequal
        TriodePentode   triode_power_pentode triode_straight_pentode triode_variable_mu_pentode
        TriodeDoubleDiodeDiode
        Indicator

[Pinning]
        1       2       3       4       5       6       7       8       9       10      T       S

[Electrodes]
        f       fm      k       a       g       g1      g2      g3      g4      g5
        i.c.    s       g2+4
        aI      kI      gI      aII     kII     gII     gT      aT      kT
        g1I     g2I     g1II    g2II    gT,g1I
        k,s     k,g3    k,g3,s  k,g5    gII,s
        aDI     aDII    aDIII   kDII    kT,kDI,kDIII
        aI,D1   t       aII,D2  D       a,D

[Tubes]
#
# Old Philips tube types
E463 Eq:RENS1384 Vf=4.0 Pentode Power
 1 2 3 4 5 6
 k,g3 f f g2 g1 a
```

```
 [P]
 Va Vg1 Vg2 Ia Ig2 S
 250 -22 250 36 3.2 2.7

# Modern tube types
AL2     Eq:4683,EL3N     Vf=4.0,6.3  Pentode Power
        1       2       3       4       5       6       7       8       T
        i.c.    f       f       k,g3    i.c.    i.c.    g2      a       g1
        [L]
        Va      Vg1     Vg2     Ia      Ig2     S
        250     -25     250     36      5       2.6
```

`PinOut` basically will read the section of the database that corresponds to the tube that is requested data for. On the line that contains the tube name (*e.g.*, AL2) a list of equivalent tubes is listed following the `Eq:` keyword. If, for example, we would have issued the command `PinOut 4683`, `PinOut` would have listed the contents of the AL2 record.

- `Descriptors`
  this section describes the allowed descriptions of the tube, including any futher descriptions.
- `Pinning`
  this section describes the allowed pin names; these could be numbers, but also letters such as 'T' for a top connection, or 'S' for a side connection
- `Electrodes`
  This describes the naming given to electrodes, such as `k,g3` which is the combination of the cathode and g3 electrode.

`PinOut` will also search for a configuration file `PinOut.cfg`, with contents similar to the below:

```
[Library]
# LibPath = Z:\Projects\uTracer\TubePinOut\lib\
 LibPath = c:\Program Files\DerkR\ExtractModel\lib\

[Images]
 ShowImage = 1
 Viewer = C:\Program Files (x86)\IrfanView\i_view32.exe

[Editor]
 DoEdit = 1
# Editor = c:\Windows\notepad.exe
 Editor = C:\Users\gebruiker\Desktop\PFE32.EXE
```

As usual, any line starting with a hash sign ('#') are considered comment lines. The meaning of these configuration items is below.

- `LibPath`
  The folder where the databases are stored. Upon installation, this folder will default `c:\Program Files\DerkR\ExtractModel\lib`. Note that a further subfolder `TubeGraphics` is included, where optional graphics are stored that will be displayed by `PinOut`. Note that the with the installation, `lib` folder is in the windows `Program Files` folder, which may be inconvenient as it usually will require special `write` permissions to change its contents. An alternative is to use a folder on a shared drive, that is accessible from multiple computers.

- `ShowImage`
  When zero, `PinOut` will not attempt to display a picture corresponding to the tube from the `Graphics` folder.

- `Viewer`
  The viewer to display the relvant tube graphic with. `PinOut` will search for tube graphics in the `Graphics` folder; for example, when `PinOut` is called with `EL84` as argument, `PinOut` will search for graphics files `EL84.xyz`, `el84.xyz` where the extension `xyz` can be any of `png, gif, jpg, bmp`. It will then launch `Viewer` with `EL84.xyz` as argument.

- `DoEdit`
  When non-zero, `PinOut` will launch an editor with the database name as argument when a certain tube is not found in the database.

- `Editor`
  Path to the editor used.

3 The program `ExtractModel`

A program called ExtractModel has been written to fit the models discussed in the previous sections to a dataset obtained by the uTracer of Ronald Dekker. The uTracer can generate a file with extension `.utd` that contains a table of all measured values, which serves as input to `ExtractModel`. The uTracer can perform different kinds of parameter sweeps. Note that two different versions of the uTracer exist; uTracer3 and uTracer6. These come with different GUI's; `ExtractModel` can work with the `.utd` output files of both. For diodes there are multiple sweeps that can be used; the primary requirement is that I_a is measured as function of V_a. To remain consistent with sweeps used for other tubes it is advised to stick to the 'I(Va,Vg) with Vs, Vh constant' sweep. It is sufficient to enter only one V_g value (the value of which is irrelevant as V_g is not used; as long as $V_g < 0$ to meet the `uTracer` requirements).

For triodes, the sweep that is used by `ExtractModel` is the 'I(Va,Vg) with Vs, Vh constant' or `Vs=cst` sweep for triodes or penthodes. This sweep performs a measurement of the anode current as a function of anode voltage, and does so for various grid voltages V_{g1}. While this sweep is sufficient to fit all parameters for a triode model, for a penthode model different values of V_s are required, too. This is done by repeating the 'I(Va,Vg) with Vs, Vh constant' sweep for various screen voltages V_s, and storing the data files separately. In addition the 'I(Va=Vs,Vg) with Vh constant' or `Vs=Va` sweep is used for penthodes.

The program is written in fortran; Gnu gfortran (mingw-w64 port) is used for compilation under Windows, and gfortran for compilation under Linux. The user interface is simple text based, aiding in easy portability over different platforms. No attempt is made for any algorithmic optimization for speed as focus has been on both robustness and ease of model development.

3.1 Initialization file description

An initialization file is required to provide all necessary inputs to `ExtractModel`. If no command line argument is provided to `ExtractModel`, `ExtractModel` will assume by default the existance of a file named '`Model.ini`'; different names can be used as command line arguments, for example: `C:\> EM4 EF86.ini`. The initialization file needs to include a number of required variables:

```
UTD = EH90_100_g3_0.0
UTD = EH90_150_g3_0.0
UTD = EH90_150_g3_5.0
UTD = EH90_200_g3_0.0
UTD = EH90_200_g3_5.0
UTD = EH90_100_g1_1.5
UTD = EH90_150_g1_1.5
UTD = EH90_150_g1_3.0
UTD = EH90_triode_g3_0.0
UTD = EH90_triode_g1_0.0
Tube = H
Model = Derk
```

Optional is an entry

```
PaMax = 1.
VgOffset = 0
IcMax = xxx
RdStart = <StartParfilename>
Verbose = 1
Debug=0
```

Following is a line-by-line description of the input required in the initialization file:

1. `UTD = <file>` The names of the data files. The maximum number of files supported by `ExtractModel` is 20. The extension `.utd` must be left out. If fitting a penthode / beam tetrode, the last file should be the file as measured in 'triode mode', *i.e.*, in a `Vs=Va` sweep. If a `Vs=Va` data file is listed, but *not* as the last file, this will stop `ExtractModel`. See also the examples for pentodes and heptodes. In addition, a data file `EH90_abc_g1_x.y` indicates a data file where V_{g2} has been kept constant at `abc` volts. It is not mandatory to include this in the filename, `ExtractModel` will automatically detect the screen voltage used. In contrast, `x.y` denotes the voltage V_{g1} at which g1 has been kept constant by an external source (while V_{g3} is controlled by the uTracer). There is no other way for `ExtractModel` to detect the value of this voltage, so including this in the file name is *mandatory*. Similarly, when V_{g1} is controlled by the uTracer, V_{g3} needs to be characterized in the filenaming.

2. `Tube`. The type of tube. Currently the choice of tubes is

   (a) D - for a diode

   (b) T - for a triode

   (c) P - for a penthode. For historical (programming) reasons, this tube designation is used if secundary emission is not playing any role. It can be used with both the Derk and DerkE models.

(d) V - for a variable-mu penthode. The model does not support secundary emission modelling. It can be used with both the Derk and DerkE models.

(e) B - for a beam tetrode. This is the tube designation that includes secondary emission effects. Even though it is called beam tetrode, it can equally well be used for pentodes that exhibit still effects due to secondary emission. It can be used with both the Derk and DerkE models.

(f) H - for a Hepthode, no scondary emission modelling.

(g) F - for a penthode with g3 modulation, no secondary emission modelling

3. `Model`. Model used for fitting. Known model names are 'Koren', 'Derk' (see Sec. **??**) ('DerkI' is also possible - see App. **??**), 'DerkE' (see Sec. **??**) and 'DerkG' (see Sec. **??**). For diodes and triodes, the only available model is 'Koren'; whenever a different name is given, it is still fitted according to the Koren model. For hepthodes, the only available model is 'Derk'.

Below follows the description of the optional input in the initialization file:

4. `PaMax` The maximum anode dissipation $P_{\max}$ used in fitting the data; only data for which $I_a V_a < P_{\max}$ is used to fit the data to. Default value (if no value provided) is 45W.

5. `VgOffset` The uTracer can only generate grid voltages between -49 and 0 volts. VgOffset is any constant voltage that is added to the uTracer grid voltage by an external source to create grid voltages below -49V. Thus `VgOffset` is expected to be negative.

6. `IcMax` is the maximum cathode current in the data file that is used for fitting. If this entry is left out, there is no limit to the cathode current used.

7. `StartParfilename` is the name of a file that contains initial parameter estimates that will be used starting the minimization - rather than the values that would be generated automatically by `ExtractModel`. The format of the file should be identical to the format that the `.par` output files have.

8. `Verbose` defines the level of verbosity of `ExtractModel`. Accepted levels are 0,1,2. Default value is 0.

9. `Debug` defines what debug information will be generated. Default value equals 0, and different values should only be used in case of unexpected behaviour of `ExtractModel`, in which case the output can be sent to the `ExtractModel` author.

3.2 Input file naming

The input filename conventions are not very strict for diodes and triodes, except for the fact that the filename extension should be `.utd`. A choice the author makes is to just call the file to the tube naming, *e.g.*, `ECC81.utd` or `PCF80C.utd`. For penthodes, more files are needed. Again, there are no strict rules to naming. A choice the author makes is to have a naming that indicates whether the tube has been measured in triode mode or with a fixed V_{g2} bias, *e.g.* `PF86_triode.utd`, which would be a PF86 measured in triode mode, and `PF86_200.utd` which would be a PF86 measured at $V_{g2} = 200V$. In all cases, the exact parameter values will be read from the data file itself.

For heptodes the naming is no longer free. Heptode data files need to be generated with both V_{g1} as a variable, and with V_{g3} as a variable, and the uTracer does not allow storing these as separate variables. The trick that needs to be played is to run several 'I(Va=Vs, Vg) with Vh constant' sweeps, where either V_{g1} or

V_{g3} is connected to the V_g terminal. Naming the datafile has therefore has to be according to the following convention:

filename = `abcdefgh_gn_xx.x.utd`

where n denotes the grid (either 1 or 3); and xx.x denotes the constant voltage that has been applied to that grid during the scan. An example is `ECH81_100_g3_10.0.utd`. This is a dataset where V_{g2} is kept at a constant voltage of 100V ('I(Va, Vg) with Vs, Vh constant' sweep), which will be reflected in the data it self. The first grid $g1$ is connected to the V_g terminal, and V_{g3} has been kept constant by an external voltage source to $-10.0V$. This voltage is not represented in the dataset, and the only way for `ExtractModel` to know this voltage is the filename. Another example is `ECH81_triode_g1_1.5.utd`. This represents a 'I(Va=Vs, Vg) with Vh constant' sweep, where an external source has been used to set $V_{g1} = -1.5V$. This implies that $g3$ of the heptode has been connected to the V_g terminal of the uTracer, and, hence, that the data tabulated in the file, has a variable V_{g3}.

3.3 Configuration file description

`ExtractModel4` utilizes a configuration file named `EM.cfg` to specify certain parameters that determine the operation of `ExtractModel`. `ExtractModel` will first look for this configuration file in the active directory; when no `EM.cfg` file is present it will look in the default location (where the file will be copied during the installation). These default locations are:

```
Linux   : $HOME/.config/EM.cfg
Windows: C:\Program Files\DerkR\ExtractModel\Config\EM.cfg
```

When also that file is absent `ExtractModel` will use built-in default values for all settings that are configured in `EM.cfg`, and assume that Gnuplot is located in

```
Linux   : /usr/bin/gnuplot
Windows: C:\Program Files\gnuplot\bin\wgnuplot.exe
```

Please refer to the sections on model explanation for the definition of the parameters. Below follows the typical appearance of the configuration file:

```
[Plot]
    GnuPlotPath = /usr/bin/gnuplot
#    C:\Program Files\gnuplot\bin\wgnuplot.exe

[UTD]
    DeltaVsVaMax = 20
    DeltaVsCstMax = 2

[Refine]
    kpmax = 2000
    kpmin = 5           !50
    MuMax = 70
    ExMin = 0.5
    HDmupMax = 250
    HDmupMin = 5
    HDmu3Max = 100
```

```
HDexpMin = 0.01
DBapMin = 0.1

FDmupMin = 0.1
FDexpMin = 0.001
FDmu3Max = 10000
FDV0Min = 0.1

WGrid = 1.
WScreen = 1.

MaxIterSpC = 100
MaxIterModel = 100
```

Following is a line-by-line description of the input required in the initialization file:

1. the `[Plot]` section describes parameters that are required for plotting.
   (a) `GnuPlotPath`: `ExtractModel` uses gnuplot [Gnu13] as the utility to create plots of the data and the curves that have been fitted to the data. Hence, ExtractModel needs to know where the executable of `Gnuplot` is located.
2. the `[UTD]` section describes parameters required for automatically determining the type of data file
   (a) `DeltaVsVaMax` describes the maximum average variation $< |V_a - V_s| >$ that may occur between the anode voltage V_a and screen voltage V_s, whilst the file is still considered to be of the $V_s = V_a$ type. It turns out that typically for a uTracer3, this value can be less than 10V; for a uTracer6 it is advisable to use 20V.
   (b) `DeltaVsCstMax` describes, similar to the above, the maximum average variation $< |V_s(i) - V_s(i+1)| >$ that may occur between the different succeeding measurements of the screen voltage V_s, whilst the file is still considered to be of the V_s=`Cst` type. It turns out that typically for both a uTracer3 and uTracer6, this value can be less than 2V.
3. the `[Refine]` section describes parameters required to limit the probability of `ExtractModel` descending into a false minimum by providing parameter boundaries that cannot be crossed during iterations.
   (a) `kpmax` the maxmimum value for k_p for both the triode model, as well as space current modeling
   (b) `kpmin` the minimum value for k_p for both the triode model, as well as space current modeling
   (c) `MuMax` the maxmimum value for μ for both the triode model, as well as space current modeling. In the above example `MuMax` is set to 70; the default value is 150. Lower values such as 70 turn out to be particularly useful when fitting variable-mu pentodes or, sometimes, heptode models.
   (d) `ExMin` the minimimum value for x for both the triode model, as well as space current modeling
   (e) `HDmupMax` Specific to the Heptode tube, and model Derk(E) this is the maximum value for μ'
   (f) `HDmupMin` Specific to the Heptode tube, and model Derk(E) this is the minimimum value for μ'
   (g) `HDmu3Max` Specific to the Heptode tube, and model Derk(E) this is the maximum value for μ_3
   (h) `HDexpMin` Specific to the Heptode tube, and model Derk(E) this is the minimimum value for x'

(i) `DBapMin` Specific to the Beam tetrode tube, and model Derk(E) this is the maximum value for a'

(j) `FDmupMin` Specific to the Pentode (w g_3 modulation) tube, and model Derk(E) this is the minimimum value for μ'

(k) `FDexpMin` Specific to the Pentode (w g_3 modulation) tube, and model Derk(E) this is the minimimum value for x'

(l) `FDmu3Max` Specific to the Pentode (w g_3 modulation) tube, and model Derk(E) this is the maximimum value for μ_3

(m) `FDV0Min` Specific to the Pentode (w g_3 modulation) tube, and model Derk(E) this is the minimimum value for V_0

(n) `WGrid` Weight given to the grid current contribution in the least-squares fit. Higher values will lead to more accurate fitting of the grid current, at the expense of lower accuracy for anode (and, if available, screen) currents.

(o) `WScreen` Weight given to the screen current contribution in the least-squares fit. Higher values will lead to more accurate fitting of the screen current, at the expense of lower accuracy for anode (and, if available, grid) currents.

(p) `MaxIterSpC` Maximum number of iterations for fitting space current.

(q) `MaxIterModel` Maximum number of iterations for fitting the tube model, once the space current has been modeled.

3.4 Output files

Output files of `ExtractModel` have not been changed since `ExtractModel` 1. The most important outputfile is 'data1.cir', which is a concatenation of the first data file name in the initialization file, and the '.cir' extension. This is the LTSpice file containing the model of the tube that has been fitted.

```
****************************************************
 .SUBCKT ECC85 1 2 3; A G C;
* ExtractModel V .995
* Model created: 29-Nov-13
X1 1 2 3 TriodeK MU= 94.0 EX=1.148 kG1=  59.9 KP= 230.9 KVB=3805. RGI=2000
+ CCG=0.0P  CGP=0.0P CCP=0.0P  ;
.ENDS

****************************************************
.SUBCKT TriodeK 1 2 3; A G C
E1 7 0 VALUE=
+{V(1,3)/KP*LOG(1+EXP(KP*(1/MU+V(2,3)/SQRT(KVB+V(1,3)*V(1,3)))))}
RE1 7 0 1G
G1 1 3 VALUE={0.5*(PWR(V(7),EX)+PWRS(V(7),EX))/kG1}
RCP 1 3 1G    ; TO AVOID FLOATING NODES IN MU-FOLLOWER
C1 2 3 {CCG}  ; CATHODE-GRID
C2 2 1 {CGP}  ; GRID-PLATE
C3 1 3 {CCP}  ; CATHODE-PLATE
D3 5 3 DX     ; FOR GRID CURRENT
```

```
R1 2 5 {RGI}  ; FOR GRID CURRENT
.MODEL DX D(IS=1N RS=1 CJO=10PF TT=1N)
.ENDS TriodeK
```

The `.cir` file contains the Spice definition of the model (DiodeK, TriodeK, PenthodeD(E), PentodeVD(E), BTetrodeD(E), HeptodeD, or Penthodeg3DE) and the tube specific description of the parameterset feeding the model. Also note that the signum function (see Eq. (**??**)) is implemented consistenly for all models. When a library of tubemodels is made, the Spice definitions of the models need to be copied only once, thereafter only the tube specific parameter description needs to be copied. The Spice code generated for a Triode is based on the code as proposed by Koren ([Kor01]). The code contains parametric descriptions of the inter-electrode capacitances, that are all set to zero in the output of ExtractModel. The user will have to add these based on datasheet reported values. Other files that are created are a `Model.par` file, which contains the refined parameters, and, if any kind of pentode or tetrode was fitted, a `Triode.par` file which is used to parametrize the space current. Further output files are the files that are fed to `Gnuplot` to do the plotting. The plot file for display of the anode currents is the `.plt` file; the plot file with extension `.spl` loads and displays the screen currents. Note that this extension is also used by Shockwave. These files can be loaded in Gnuplot any time later to review the fitting results, independent of `ExtractModel`.

3.5 Model fitting tips

Feedback from users working with `ExtractModel` has generated some practical tips for a good fit, and listed in the following. `ExtractModel`already incorporates, or will incorporate, some of the tips below and generate warnings when the data to be fitted is insufficient (sec. 3.5.1), when the tube displays saturation effects (sec. 3.5.2), or when the dynamic range is too large for a good fit (sec. 3.5.3). `ExtractModel` will leave it to the discretion of the user to continue - but results may not be correct.

3.5.1 Number of different values for V_g

For a triode fit, different values for V_g are required to get a good estimate of especially the parameter k_p. Depending on how accurate the measurements are, at least 5 different values for V_g are required, and at least a few values are needed that give an accurate description of the cut-off behaviour, which helps in good estimates of k_{VB}. This means that several V_g values should be present that have a region where the anode current is zero within the precision of the uTracer. Preferably (see also Sec. 3.5.3 one would have a range of V_g values which, at the maximum anode voltage and highest V_g, result in an anode current of about 150% of the maximum anode current, and at the maximum anode voltage and lowest V_g about 20% of that. Fig. **??** gives an idea for two different triodes. For regular penthodes, at least 3 different values of V_{g2} are required, and per V_{g2} value at least 3 different values for V_g. Variable-mu pentodes - on the contrary - require a large number of different values for v_{g1}, preferably evenly spread over the high-mu and low-mu characteristic of the pentode.

3.5.2 Saturation

Saturation of a tube is a phenomenon where the cathode is not capable of delivering the current that theoretically would flow, see also [Spa48] for a description of this phenomenon. For triodes, the anode current as function of anode voltage show a positive curvature (the curve becomes ever steeper with increasing anode voltage). Severe saturation will cause the anode current to show negative curvature (*i.e.*, the steepness of current increase as a function of voltage becomes less). It is important to remove curves that display this behaviour while fitting.

`ExtractModel` will try to fit the measurements (even though in very severe cases, it gives a warning); and as a result the fitted parameter values may not represent the true behaviour.

For penthodes, saturation may occur either due to excessive screen current at low anode voltage, or, similar as with a triode, due to excessive anode current at very high anode voltages. Saturation at low anode voltages can be detected by inspection of the screen current, which, at low anode voltages, should be a concave curve. When this curve becomes convex, saturation plays a role. Especially when secundary emission is included in the model fitting, `ExtractModel` may easily be fooled by the convex part in the screen current, and mistakenly interprets it as secundary emission (which also causes convex curvature of the screen current!).

As a rule of thumb, one should never have data with cathode currents that largely exceed the maximum values as published in the datasheets. For new tubes of reknowned brands, the cathode can typically deliver 150% of the tabulated maximum value. This is often not true for some russian NOS stubes, and old, used tubes that already spent a good deal of their life in heavy duty.

3.5.3 Dynamic range

The previous section Sec. 3.5.2 already implicitly gave an upper boundary for the current measurements. Curves reaching lower current values are required for covering the full space that is needed for a good parameter fit - however, if currents are very close to zero over the full anode voltage range, this adds no information (and actually detracts `ExtractModel`). Typically, a good dynamic range is between 5 and 10, meaning that the highest current reached for the V_g, V_{g2} combination that gives the lowest overall current, should be no less than about 1/10 to 1/5 of the maximum current over all parameters. Good examples for this are the curves presented throughout this document, where the dynamic range never exceeds 7.

For vaiable-mu pentodes, care should be taken that all curves cover the high-mu and low-mu part of the characteristic evenly. A dataset with many values for the low-mu part, and few for the high-mu part will give an untrustworthy result. Note that this is a mistake easily made, since an evenly distributed set of grid values V_{g1} will likely correspond to a few high-mu values, and many more low-mu values.

3.5.4 Data range

In particular with heptodes, it is important to play a bit with the maximum anode dissipation and/or maximum cathode current of the data used for fitting. The heptode model can easily get caught in a false minimum if the maximum power/current used in fitting is too large. This leads to the interesting effect that the fit can sometimes (even for high currents) be better if the data used in the fit is restricted to values of 0.5W or even a bit below for the anode dissipation. Until initial parameter estimation is improved, this inconvenience has to be accepted.

4 Conversion utility for previous initialization files

As `ExtractModel` 4 uses a new format for its initialization file, a conversion utility `Old2NewIni` has been provided. Its use is very straightforward:

```
C:\Users\gebruiker>Old2NewIni
 Old2NewIni usage:
 Old2NewIni Old.ini
 Old2NewIni will generate a file NewOld.ini
```

and its results are demonstrated below:

```
C:\Projects\Spice\ExtractModelResults\EZ80>Old2NewIni Model.ini
 Initialization file is: Model.ini
 New ini file NewModel.ini has been created.
```

The conversion result is listed below, showing first the contents of `Model.ini` (which is in the `ExtractModel`3.x format), followed by the contents of the new file `NewModel.ini`:

```
C:\Projects\Spice\ExtractModelResults\EZ80>type Model.ini
C:\Program Files (x86)\gnuplot\bin\wgnuplot.exe
1
EZ80
D
Koren
18     !Pmax in fit
0       !Vg Offset


C:\Projects\Spice\ExtractModelResults\EZ80>type NewModel.ini
 UTD = EZ80
 Tube = D
 Model = Koren
 PaMax = 18
 VgOffset = 0

 Verbose = 0
 Debug = 0
```

In the above, it can be seen that the optional entries (see Sec. 3.1) `Verbose` and `Debug` have both been set to zero, which under usual conditions is the suggested value. Note, that this also represents their default value, and these parameters could have been left out with no difference. All other parameters in `NewModel.ini` have been retrieved from the original file `Model.ini`.

5 Examples for several tube types

In this section several examples will be shown of actual model creation.

5.1 Fitting a diode

Fitting a diode is theoretically the easiest application for `ExtractModel`. It requires a data file that has the measured Ia-Va curve of the diode; see Sec. 3 how to generate such a file. To fit the data in the file, an input file will have to be made as shown below:

```
 UTD = EZ80
 Tube = D
```

```
 Model = Koren
 PaMax = 18
 VgOffset = 0
```

To execute `ExtractModel`, open a command window (cmd under Windows, xterm under Linux) and execute `ExtractModel`. Depending on the level of verbosity, a number of messages will be generated by `ExtractModel`. A Gnuplot window pops up that shows the measured data points, and the curve based on the refined parameters. Under Windows, there is a second pop-up with a Pause button - press OK and it will disappear. The purpose of this pop-up is to show the fit results for any other data files (we will explore this later).

The directory will now contain the following files:

```
C:\Projects\Spice\ExtractModelResults\EZ80>dir
 Volume in drive C is Acer
 Volume Serial Number is 002D-65E2

 Directory of C:\Projects\Spice\ExtractModelResults\EZ80

25-12-2025  12:52    <DIR>          .
06-12-2025  14:25    <DIR>          ..
25-12-2025  12:05               548 EZ80.cir
25-12-2025  12:05               992 EZ80.dat
25-12-2025  12:05             1.526 EZ80.fit
17-08-2020  14:26               106 EZ80.log
25-12-2025  12:05               175 EZ80.plt
17-08-2020  14:26             3.264 EZ80.utd
17-08-2020  14:26               175 Model.ini
25-12-2025  12:05                75 Model.Par
25-12-2025  12:05             3.164 NewModel.ini
```

The most important file created is the `EZ80.cir` file, which is the Spice circuit description:

```
****************************************************
.SUBCKT EZ80 1 2; A C;
*       ExtractModel V4.060
* Model created: 25-Dec-2025
*
X1 1 2 DiodeK  EX=1.410 k =  975.3 eps=  -0.1
+ CCP=0.0P  ;
.ENDS

****************************************************
.SUBCKT DiodeK 1 2; A C
*
* NOTE: LOG(x) is base e LOG or natural logarithm.
* For some Spice versions, e.g. MicroCap, this has to be changed to LN(x).
*
```

```
G1   1 2  VALUE = {0.5/k*(PWR(V(1,2)+eps,EX)+PWRS(V(1,2)+eps,EX))}
C3   1 2  {CCP}   ; CATHODE-PLATE       A  - C
.ENDS DiodeK
```

This file contains the generic description of a diode (a lot of this description is copied from [Kor01]) as `.SUBCKT DiodeK`; and the model which is specific to the EZ80. The definitions of the capacitances is for the user; please edit these yourself based on values that are reported in the handbooks (see [Phi13] for an almost exhaustive collections of datasheets).

EZ80.plt is the `Gnuplot` command file that generates the plot. This file can always be loaded by Gnuplot, independent of `ExtractModel`, at a later moment. The two files `EZ80.dat` and `EZ80.fit` contain the original data (as presented in `EZ80.utd` and the model curve, respectively, that are used by the command file `EZ80.plt`. Finally, there is a `Model.par` file, which contains the values of the parameters.

Unfortunately, `uTracer` has not been designed for the parameter space most relevant to diodes: high current and low voltage. As a result, the data for a diode will be rather noisy. To make any fitting parameters less dependent on this noise, multiple measurements can be made (with identical settings in the `uTracer`) resulting in multiple `.utd` files. A typical `.ini` file corresponding to this would be:

```
UTD = EZ80_1
UTD = EZ80_2
UTD = EZ80_3
Tube = D
Model = Koren
PaMax = 18
VgOffset = 0
```

Here `EZ80_1, EZ80_2` *etc.* are the different files that have been collected. The remainder of the process is identical to fitting only one data file, and results in a file `EZ80_1.cir` with the Spice circuit describing the diode.

5.2 Fitting a triode

For fitting a triode, we first have to create the `.ini` file (named `ECLL800C.ini` in this example), which contains the input that `ExtractModel` will use. Below is an example input file, which is based on the assumption that in the directory where `ExtractModel` is called, a file ECLL800C.utd exists that contains Ia-Va curves for various values Vg of the grid bias:

```
UTD = ECLL800L
Tube = T
Model = Koren
PaMax = 18
VgOffset = 0
```

To execute `ExtractModel`, open a command window (cmd under Windows, xterm under Linux) and execute `ExtractModel`.

In the cmd window we see a lot of output: the version of `ExtractModel`, confirmation of the contents of the inputfile; the parameter values based on the first estimation; the sum of squares R_2 before refinement;

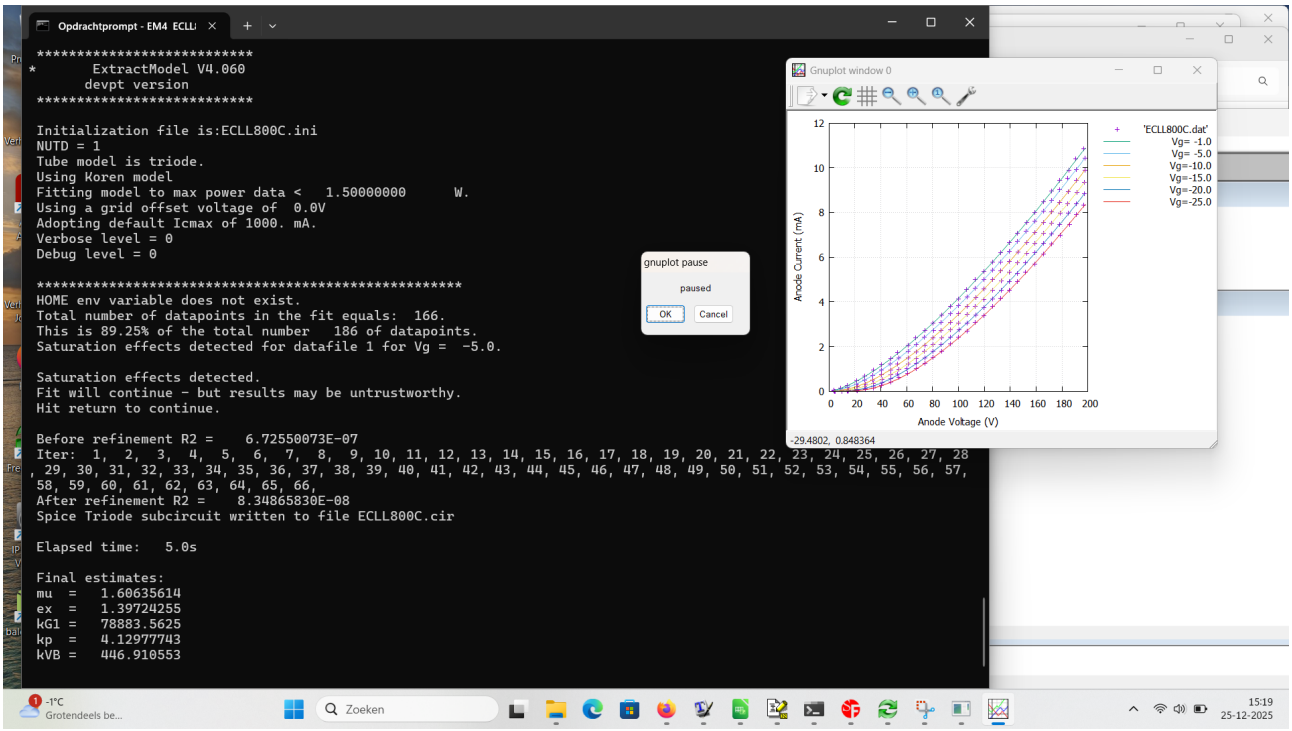

Figure 1: Output of `ExtractModel` for the ECLL800C example

the iteration number and final sum of squares. The last bit of information is the parameter values after the refinement, and the fact that a Spice triode subcircuit has been written to file `ECLL800C.cir`. This file name has been constructed from the input file name, with the addition of the `.cir` extension.

Next to that, a Gnuplot window pops up that shows the measured data points, and the curve based on the refined parameters. Under Windows, there is a second pop-up with a Pause button - press OK and it will disappear. The purpose of this pop-up is to show the fit results for any other data files (we will explore this later).

The directory will now contain the following files:

```
 Volume in drive C is Acer
 Volume Serial Number is 002D-65E2

 Directory of C:\Projects\Spice\ExtractModelResults\ECLL800

25-12-2025  15:18    <DIR>          .
25-12-2025  13:50    <DIR>          ..
25-12-2025  14:13                 0 Debug.log
25-12-2025  15:18               902 ECLL800C.cir
25-12-2025  15:18             5.952 ECLL800C.dat
25-12-2025  15:18             5.632 ECLL800C.fit
```

```
25-12-2025  15:17               603 ECLL800C.ini
25-12-2025  14:43               604 ECLL800C.ini.$$$
25-12-2025  15:18               412 ECLL800C.plt
21-12-2025  13:01            18.888 ECLL800C.utd
21-12-2025  13:20            20.706 ECLL800L_150.utd
21-12-2025  13:25            20.706 ECLL800L_200.utd
21-12-2025  13:23            20.706 ECLL800L_250.utd
21-12-2025  13:15            19.898 ECLL800L_triode.utd
25-12-2025  14:13               509 Eigen.dat
25-12-2025  14:13               382 Hinv.dat
25-12-2025  15:18               125 Model.par
              15 File(s)        116.025 bytes
               2 Dir(s)  373.565.554.688 bytes free
```

where the files `ECLL800L_abc` are pentode measurements that will be analyzed in a later section.

The most important file created is the `ECLL800C.cir` file, which is the Spice circuit description:

```
****************************************************
.SUBCKT ECLL800L 1 2 3; A G C;
*       ExtractModel V4.060
* Model created: 25-Dec-2025
*
X1 1 2 3 TriodeK MU=  1.61 EX=1.397 KG1=78884. KP=   4.1 KVB=  447. RGI=2000
+ CCG=0.0P  CGP=0.0P CCP=0.0P  ;
.ENDS

****************************************************
.SUBCKT TriodeK 1 2 3; A G C
*
* NOTE: LOG(x) is base e LOG or natural logarithm.
* For some Spice versions, e.g. MicroCap, this has to be changed to LN(x).
*
E1 7 0 VALUE=
+{V(1,3)/KP*LOG(1+EXP(KP*(1/MU+V(2,3)/SQRT(KVB+V(1,3)*V(1,3)))))}
RE1 7 0 1G
G1 1 3 VALUE={0.5*(PWR(V(7),EX)+PWRS(V(7),EX))/KG1}
RCP 1 3 1G    ; TO AVOID FLOATING NODES IN MU-FOLLOWER
C1 2 3 {CCG}  ; CATHODE-GRID
C2 2 1 {CGP}  ; GRID-PLATE
C3 1 3 {CCP}  ; CATHODE-PLATE
D3 5 3 DX     ; FOR GRID CURRENT
R1 2 5 {RGI}  ; FOR GRID CURRENT
.MODEL DX D(IS=1N RS=1 CJO=10PF TT=1N)
.ENDS TriodeK
```

This file contains the generic description of a triode (a lot of this description is copied from [Kor01]) as `.SUBCKT Triode`; and the model which is specific to the EC(LL)800. The definitions of the capacitances is for

the user; please edit these yourself based on values that are reported in the handbooks (see [Phi13] for an almost exhaustive collections of datasheets).

ECLL800C.plt is the `Gnuplot` command file that generates the plot. This file can always be loaded by Gnuplot, independent of `ExtractModel`, at a later moment. The two files `ECLL800C.dat` and `ECLL800C.fit` contain the original data (as presented in `ECLL800C.utd` and the model curve, respectively, that are used by the command file `ECLL800C.plt`.

Finally, there is a `Model.par` file, which contain the values of the parameters.

5.3 Fitting a triode with grid current

For fitting a triode with grid current, we largely follow the sequence of steps outlined in the prvious section. As example, we wil model a 6AQ5 tube, strapped as triode.

```
 UTD = 6AQ5Triode
 Tube = T
 Model = DerkG
 PaMax = 18
 VgOffset = 0
uT62= 1
```

To execute `ExtractModel`, open a command window (cmd under Windows, xterm under Linux) and execute `ExtractModel`.

In the cmd window we see a lot of output: the version of `ExtractModel`, confirmation of the contents of the inputfile; the parameter values based on the first estimation; the sum of squares R_2 before refinement; the iteration number and final sum of squares. The last bit of information is the parameter values after the refinement, and the fact that a Spice triode subcircuit has been written to file `ECLL800C.cir`. This file name has been constructed from the input file name, with the addition of the `.cir` extension.

Next to that, a Gnuplot window pops up that shows the measured data points, and the curve based on the refined parameters. Under Windows, there is a second pop-up with a Pause button - press OK and the fit of the grid current will appear.

The most important file created is the `6AQ5Triode.cir` file, which is the Spice circuit description:

```
****************************************************
.SUBCKT 6AQ5triode 1 2 3; A G C;
*       ExtractModel V4.100
* Model created:  5-Feb-2026
*
X1 1 2 3 TriodeDG MU= 10.51 EX=1.330 KG1=  483. KP=  33.1 KVB= 2022. kG=  3626. gp= 2.060 alg= 2.643 beg
+ CCG=0.0P  CGP=0.0P CCP=0.0P  ;
.ENDS

****************************************************
.SUBCKT TriodeDG 1 2 3; A G C
*
* NOTE: LOG(x) is base e LOG or natural logarithm.
* For some Spice versions, e.g. MicroCap, this has to be changed to LN(x).
*
```

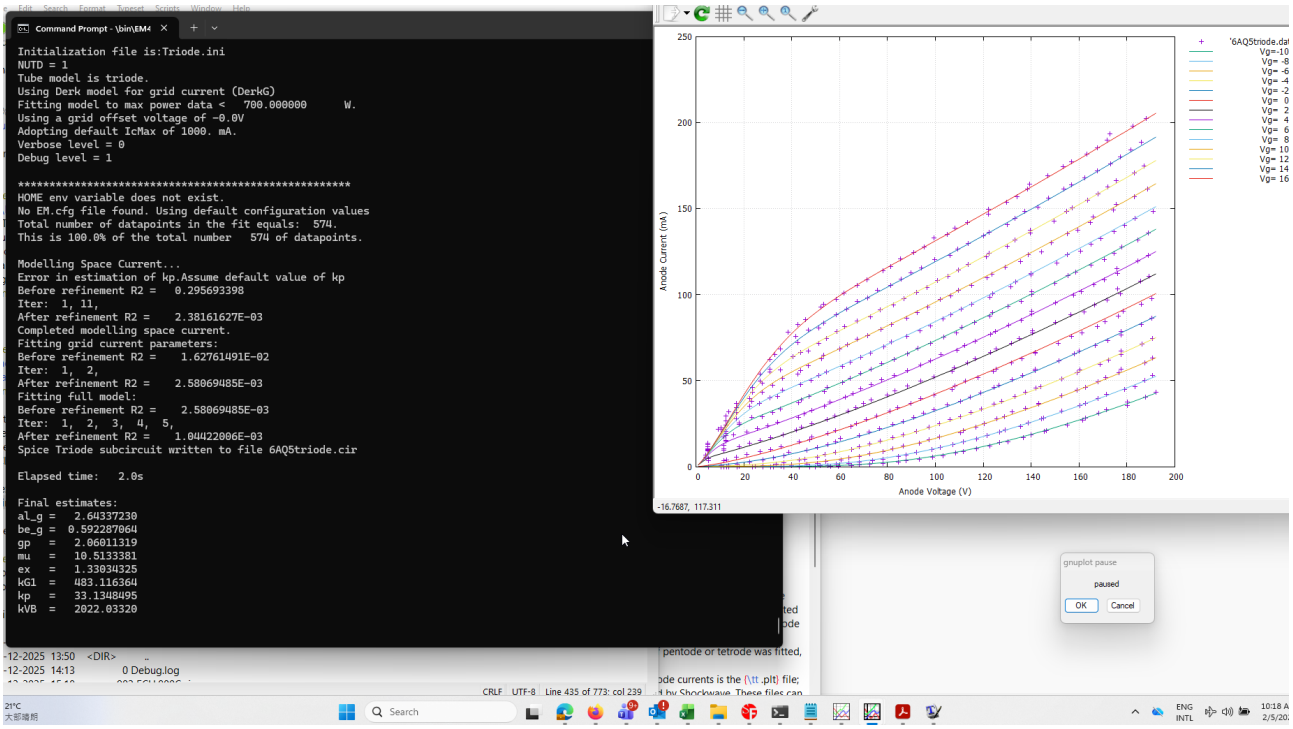

Figure 2: Output of `ExtractModel` for the 6AQ5 (wrapped as triode) example

```
E1 7 0 VALUE=
+{V(1,3)/KP*LOG(1+EXP(KP*(1/MU+V(2,3)/SQRT(KVB+V(1,3)*V(1,3)))))}
RE1 7 0 1G
E2 8 0 VALUE = { PWR( (V(1,3)/MU + V(2,3)), EX) }
E3 9 0 VALUE={IF (V(2,3)>0, V(8)/KG1, 0.5*(PWR(V(7),EX)+PWRS(V(7),EX))/KG1)}
* grid:
E4 10 0 VALUE = {IF (V(2,3) > 0, PWR(V(2,3),EX)/kG * (1 + alg*Exp(-beg*V(1,3)/V(2,3)*SQRT(beg*V(1,3)/V(2
G1 2 3 VALUE = {V(10)}
G2 1 3 VALUE = {V(9) - gp * V(10) }
RCP 1 3 1G    ; TO AVOID FLOATING NODES IN MU-FOLLOWER
C1 2 3 {CCG}  ; CATHODE-GRID
C2 2 1 {CGP}  ; GRID-PLATE
C3 1 3 {CCP}  ; CATHODE-PLATE
.ENDS TriodeDG
```

This file contains the generic description of a triode (a lot of this description is copied from [Kor01]) as `.SUBCKT Triode`; and the model which is specific to the 6AQ5. The definitions of the capacitances is for the user; please edit these yourself based on values that are reported in the handbooks (see [Phi13] for an almost exhaustive collections of datasheets).

All other output files are similar to those described in the previous section.

5.4 Fitting a pentode

In order to fit a pentode, several files need to be collected:

1. A data file created with the $V_s = V_a$ sweep
2. Several data files created with the $V_s = cst$ sweep, for various values of V_s.

Starting, our directory contents are listed below:

```
C:\Projects\Spice\ExtractModelResults\PF86>dir
 Volume in drive C is Acer
 Volume Serial Number is 002D-65E2

 Directory of C:\Projects\Spice\ExtractModelResults\PF86

25-12-2025  15:47    <DIR>          .
25-12-2025  13:50    <DIR>          ..
17-08-2020  14:27               278 PF86.ini
17-08-2020  14:27            18.888 pf86_200.utd
17-08-2020  14:27            18.888 pf86_250.utd
17-08-2020  14:27            18.888 pf86_300.utd
17-08-2020  14:27            22.019 pf86_triode.utd
               5 File(s)         78.961 bytes
               2 Dir(s)  373.568.495.616 bytes free
```

where the contents of the initialization file `PF86.ini` are as follows:

```
 UTD = pf86_200
 UTD = pf86_250
 UTD = pf86_300
 UTD = PF86_triode
 Tube = P
 Model = Derk
 PaMax = 2.
 VgOffset = 0
 Debug=0

 Verbose = 0
```

Here the files `pf86_xxx.utd` are the files with a $Va = cst$ sweep and `pf86_triode` the file with a $V_s = V_a$ sweep. Note the order: the $V_s = V_a$ sweep must be last! We now run `ExtractModel`, with `PF86.ini` as argument (we could have called the initialization file anything - if we would have called it `Model.ini` `ExtractModel` could have been run without argument).

In the output we can identify that `ExtractModel` first does a refinement of the space current parameters. `ExtractModel` also reports that two spice files are generated: a file `pf86_200.cir` which contains the spice model for a pentode, and a file `PF86_triode.cir` which contains the spice circuit for the PF86 strapped as a triode (*i.e.*, the screen is connected to the anode). There is also the `Gnuplot` pop-up - pressing OK will now

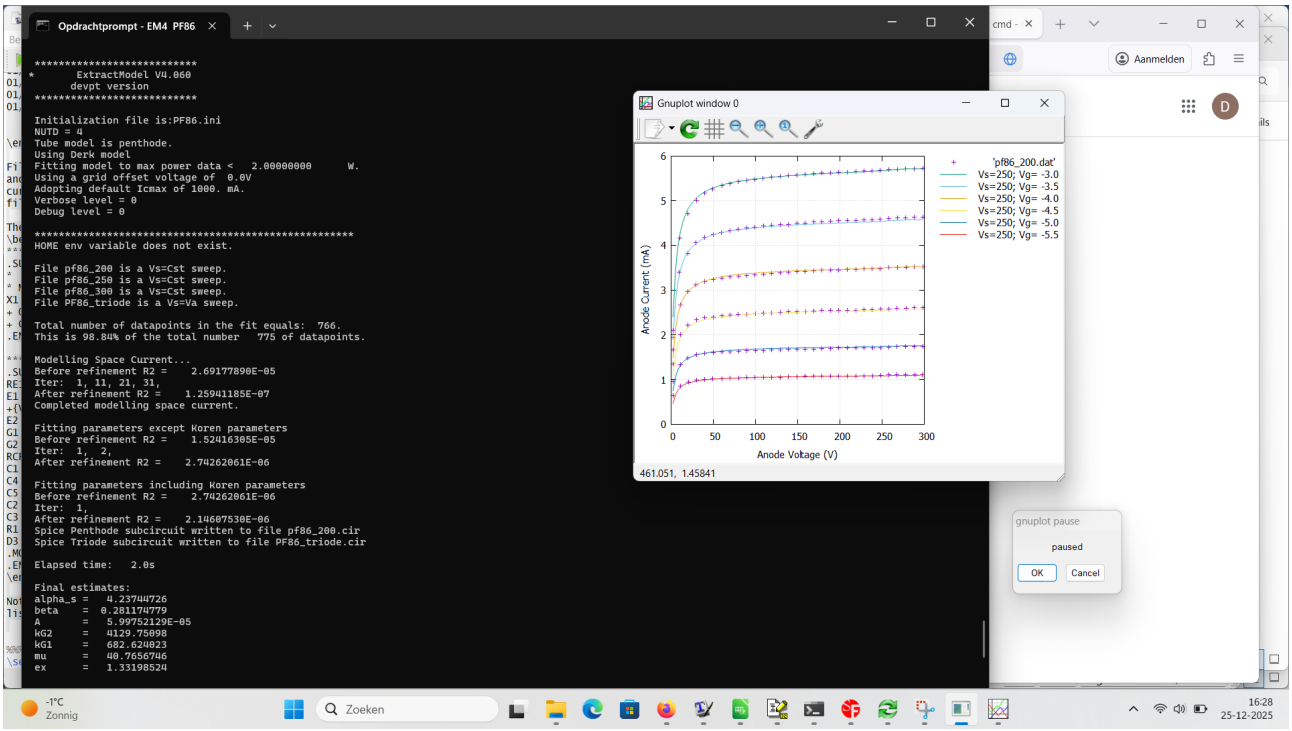

Figure 3: Output of `ExtractModel` for the PF86 example with the PF86.ini initialization file.

display the screen currents and fit for the first dataset. Repeatedly pressing 'OK' will display the anode and screen currents of next dataset *etc.*, until we have the last dataset, which is the $V_s = V_a$ dataset.

The directory now contains the following output files:

```
C:\Projects\Spice\ExtractModelResults\PF86>dir
 Volume in drive C is Acer
 Volume Serial Number is 002D-65E2

 Directory of C:\Projects\Spice\ExtractModelResults\PF86

25-12-2025  16:23    <DIR>          .
25-12-2025  13:50    <DIR>          ..
25-12-2025  16:23               261 Model.par
25-12-2025  15:52               656 PF86.ini
25-12-2025  15:50               656 PF86.ini.$$$
25-12-2025  16:23             1.396 pf86_200.cir
25-12-2025  16:23             5.952 pf86_200.dat
25-12-2025  16:23             5.680 pf86_200.fit
25-12-2025  16:23               412 pf86_200.plt
25-12-2025  16:23             5.680 pf86_200.sft
```

```
25-12-2025  16:23               430 pf86_200.spl
17-08-2020  14:27            18.888 pf86_200.utd
25-12-2025  16:23             5.952 pf86_250.dat
25-12-2025  16:23             5.680 pf86_250.fit
25-12-2025  16:23               412 pf86_250.plt
25-12-2025  16:23             5.680 pf86_250.sft
25-12-2025  16:23               430 pf86_250.spl
17-08-2020  14:27            18.888 pf86_250.utd
25-12-2025  16:23             5.952 pf86_300.dat
25-12-2025  16:23             5.680 pf86_300.fit
25-12-2025  16:23               412 pf86_300.plt
25-12-2025  16:23             5.680 pf86_300.sft
25-12-2025  16:23               430 pf86_300.spl
17-08-2020  14:27            18.888 pf86_300.utd
25-12-2025  16:23               905 PF86_triode.cir
25-12-2025  16:23             6.944 PF86_triode.dat
25-12-2025  16:23             6.507 PF86_triode.fit
25-12-2025  16:23               485 PF86_triode.plt
25-12-2025  16:23             6.507 PF86_triode.sft
25-12-2025  16:23               505 PF86_triode.spl
17-08-2020  14:27            22.019 pf86_triode.utd
25-12-2025  16:23               125 Triode.par
              30 File(s)        158.092 bytes
               2 Dir(s)  373.561.999.360 bytes free
```

Files with the `.plt` extension are the plot files that can be load in `Gnuplot` and will show the anode measured and fitted current. Files with the `.spl` extension will do the same for the screen current. Files with the `.dat` and `.fit` extension are the data files used by the `.plt` files, and files with the `.sft` extension are used by the `.spl` files.

The `pf86_200.cir` file is the file with the Spice model describing the pentode:

```
****************************************************
.SUBCKT pf86_200 1 2 3 4 ; A G2 G1 C;
*       ExtractModel V4.060
* Model created: 25-Dec-2025
*
X1 1 2 3 4 pentodeD  MU= 40.8 EX=1.332 kG1=  683. KP= 353.0 kVB =  1953.9 kG2= 4129.8
+ 0okg1m0okG2=.12E-02 Aokg1=.88E-07 alkg1palskg2=.12E-02 be=  0.28 als=  4.24 RGI=2000
+ CCG1=0.0P  CCG2 = 0.0p CPG1 = 0.0p CG1G2 = 0.0p CCP=0.0P  ;
.ENDS

****************************************************
.SUBCKT pentodeD 1 2 3 4; A G2 G1 C
*
* NOTE: LOG(x) is base e LOG or natural logarithm.
* For some Spice versions, e.g. MicroCap, this has to be changed to LN(x).
*
```

```
RE1  7 0  1MEG    ; DUMMY SO NODE 7 HAS 2 CONNECTIONS
E1 7 0 VALUE=
+{V(2,4)/KP*LOG(1+EXP(KP*(1/MU+V(3,4)/SQRT(KVB+V(2,4)*V(2,4)))))}
E2   8 0 VALUE = {Ookg1mOokG2 + Aokg1*V(1,4) - alkg1palskg2/(1 + be*V(1,4))}
G1   1 4  VALUE = {0.5*(PWR(V(7),EX)+PWRS(V(7),EX))*V(8)}
G2   2 4 VALUE = {0.5*(PWR(V(7),EX)+PWRS(V(7),EX))/KG2 * (1+ als/(1+be*V(1,4)))}
RCP  1 4  1G      ; FOR CONVERGENCE     A  - C
C1   3 4  {CCG1}   ; CATHODE-GRID 1     C  - G1
C4   2 4  {CCG2}   ; CATHODE-GRID 2     C  - G2
C5   2 3  {CG1G2}   ; GRID 1 -GRID 2    G1  - G2
C2   1 3  {CPG1}  ; GRID 1-PLATE        G1 - A
C3   1 4  {CCP}   ; CATHODE-PLATE       A  - C
R1   3 5  {RGI}   ; FOR GRID CURRENT    G1 - 5
D3   5 4  DX      ; FOR GRID CURRENT    5  - C
.MODEL DX D(IS=1N RS=1 CJO=10PF TT=1N)
.ENDS pentodeD
```

Note that the circuit is called pf86_200, which is the name of the first data file appearing in the list of datasets.

5.5 Fitting a beam tetrode or pentode with secundary emission

In this section, a model will be fitted to the EL500 tube (which is identical to the PL500/504). Fitting a penthode with secundary emission is similar to fitting a regular penthode model;

```
UTD = EL500_300
UTD = EL500_250
UTD = EL500_200
UTD = EL500_triode
Tube = B
Model = DerkE
PaMax = 700.
VgOffset = -25
iDebug=0

Verbose = 0
```

Note the fact that now the entry for VgOffset equals -25 - the measurements have been done while adding this offset to the first grid in order to keep anode currents below 200mA. For PaMax a value of 700 is used, basically saying that no limitation for anode dissipation is used.

Note the use of the 'B' designator for using a penthode model with secundary emission, and the fact that the used model is 'DerkE'. Alternatively, the 'Derk' model could have been used, with a worse fit.

The output is very similar to the output for the penthode model, the main difference being that there are more parameters that are displayed. `Gnuplot` will also show the same Pause pop-up; pressing 'OK' will have `Gnuplot` show the screen current or the next dataset. The output file types are identical as for a penthode.

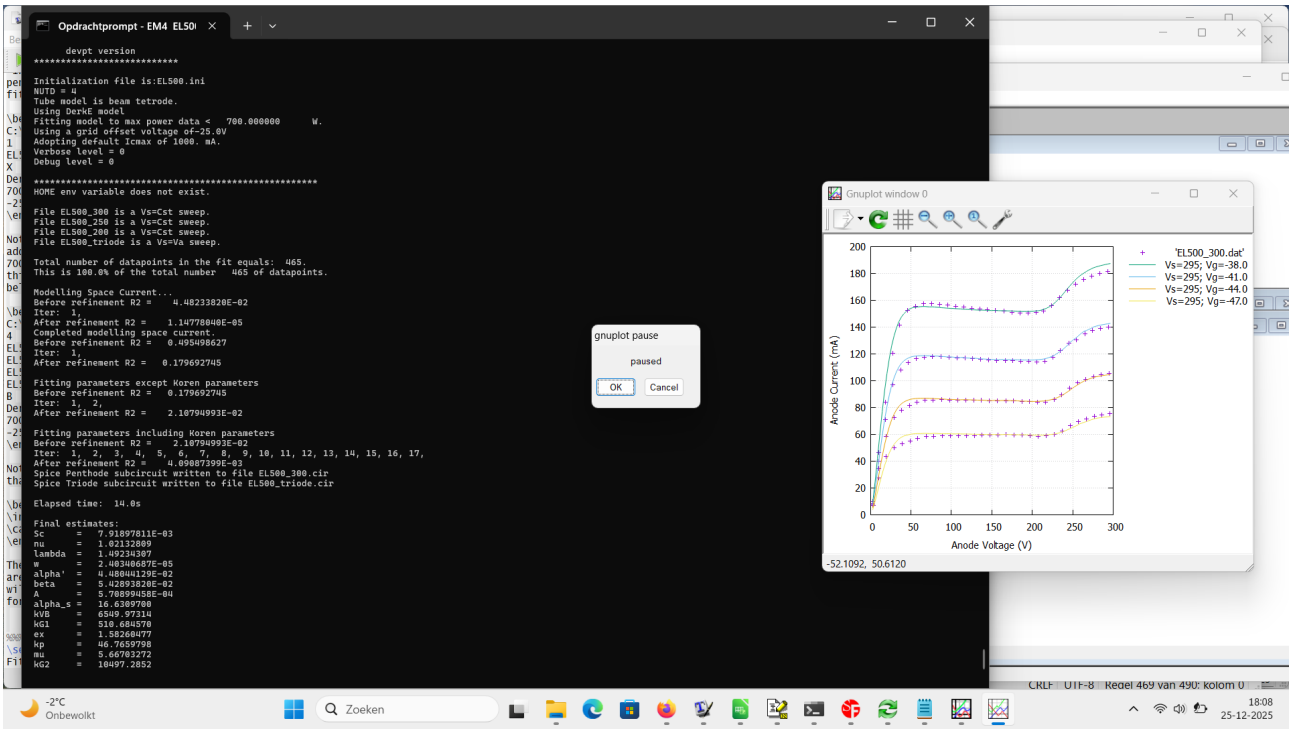

Figure 4: Output of `ExtractModel` for the EL500 example with the Pentode.ini initialization file.

5.6 Fitting a variable mu pentode

Fitting a variable mu pentode requires a rather different approach compared to what is outlined for a standard pentode and or a pentode with secondary emission. The sequence of fitting a triode strapped tube, followed by the pentode fit, is similar, however. We first describe the triode-strapped fit. Care should be taken in the dataset that it covers both the high and low-mu regions - *i.e.*, one should have a large range of V_{g1} voltages. A rather typical `.ini` file is shown below

```
UTD = EF89_125
UTD = EF89_150
UTD = EF89_rc_125
UTD = EF89_rc_150
UTD = EF89_200
UTD = EF89_triode
UTD = EF89_triode_rc
Tube = V
Model = Derk
PaMax = 4.       !Pmax in fit
VgOffset = 0     !Vg Offset
Verbose = 1
```

```
Debug=0
IcMax = 25
```

Note that some files carry the `_rc` suffix, which indicates these files are in the remote-cut-off area, with highly negative grid voltage. The use of this suffix is not required;it is required however that the last two files are $V_s = V_a$ files, of which the first is in the regular operating regime, and the second in the remote cut-off regime. Executing `ExtractModel` with this `.ini` file leads to the result depicted below:

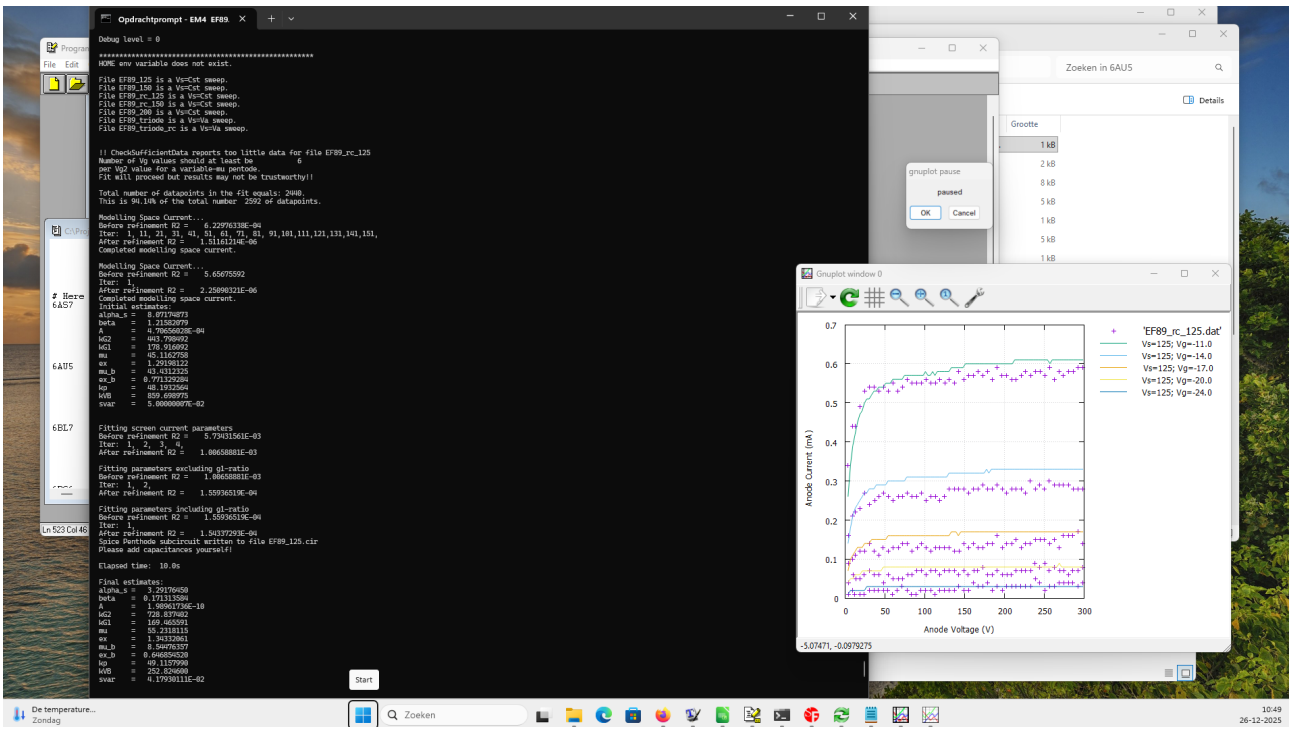

Figure 5: Output of `ExtractModel` for an EF89.

Fitting a variable mu-pentode can be tricky and heavily depends on the choice of data files; it is critical to include sufficient data in the remote cut-off area. In this particular case, the fit is excellent as demonstrated by the quality of the fit in the remote cut-off area. Experience has shown that the value for `MuMax` in the `EM.cfg` file can also have an important influence: choosing `MuMax` too high often results in a false minimum. In the current example, using the default value `MuMax = 150` gives satisfactory results.

5.7 Fitting a Heptode

For a heptode, a number of datafiles need to be available for various V_{g1} and V_{g3} voltages. In this example, we will fit a heptode. A sample input file (EK90.ini in the example set) is listed below. Note that only the `Derk` model is supported for heptodes.

```
UTD = EK90_100_g3_0.0
```

```
UTD = EK90_150_g3_0.0
UTD = EK90_150_g3_5.0
UTD = EK90_200_g3_0.0
UTD = EK90_200_g3_5.0
UTD = EK90_100_g1_2.5
UTD = EK90_150_g1_2.5
UTD = EK90_150_g1_5.0
UTD = EK90_triode_g3_0.0
UTD = EK90_triode_g1_0.0
Tube = H
Model = Derk
PaMax = 4.       !Pmax in fit
VgOffset = 0     !Vg Offset
Verbose = 1
Debug=0
```

The only straight heptode known by the author is the EH90. Most other well-known heptodes have at least one grid that displays a remote cut-off characteristic. For the EK90, g3 has a remote cut-off characteristic; for many other heptodes such as the ECH81 it is g1 that is used to control amplification. `ExtractModel` will ignore this remote cut-off behaviour! Hence there are also no `.utd` files that are recorded in the remote cut-off regime (in contrast to the model for variable mu pentodes). Note in the above that the last two files are $V_s = V_a$ files, where the order is such that in the first of these V_{g1} is varied (while V_{g3} is kept constant at zero), and the reverse for the second data file. Also note the use of the filename to indicate the voltage of the grid that is not controlled by the uTracer, but is generated by an external voltage source. See Sec. 3.1 for more details.

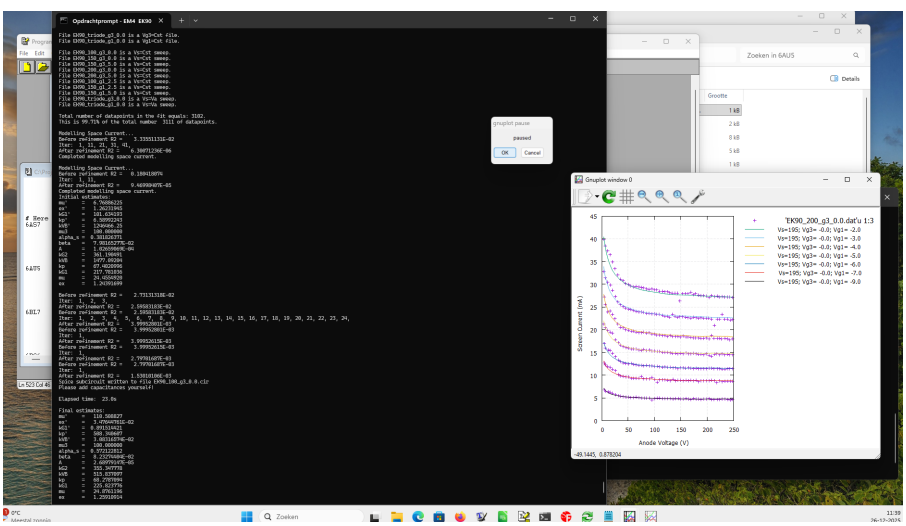

Figure 6: Output of `ExtractModel` for the EK90 example.

In Fig. 6, the output is shown. It is observed that the parameter `mu3 = 100` runs against its limit set in the default configuration file `EM.cfg`, where it is set through `HDmu3Max = 100`. Another observation is the

high value for `mu' = 110.5`; this value is caused by trying to optimize the fit for some of the remote cut-off characteristics of g3, and no physical meaning should be attributed to this value.

5.8 Fitting a Pentode with g_3 modulation

Like with a heptode, a number of datafiles needs to be available for various V_{g1} and V_{g3} voltages. In this example, we will fit a pentode with g_3 modulation.

A sample `.ini` (in this example called `pg3.ini`) file is

```
UTD = EF80_200_g3_0.0
UTD = EF80_250_g3_0.0
UTD = EF80_150_g3_0.0
UTD = EF80_250_g3_20.0
UTD = EF80_250_g3_40.0
UTD = EF80_150_g1_1.0
UTD = EF80_200_g1_1.0
UTD = EF80_250_g1_1.0
UTD = EF80_triode_g3_0.0
UTD = EF80_triode_g1_1.0
UTD = EF80_triode_g3_20.0
Tube = F
Model = DerkE
PaMax = 4
Icmax = 30
Verbose =1
Debug=0
```

In the above, file naming is not free, as in heptode fitting. A file ending on `_g1_3.0` indicates that this is a file where g_1 has been constant at $-3V$, and g_3 has been connected to the uTracer. Note, that here the optional instruction `Icmax = 30` is given. This instructs `ExtractModel` to ignore data with a cathode current larger than 30 mA in the fit. In addition, a local `EM.cfg` file has been created to limit `MuMax = 70`. In the default situation of `MuMax = 150`, `ExtractModel` would straddle down in a local minimum with `mu = 150` that is incorrect (we know that `mu` should be around 50 for the EF80).

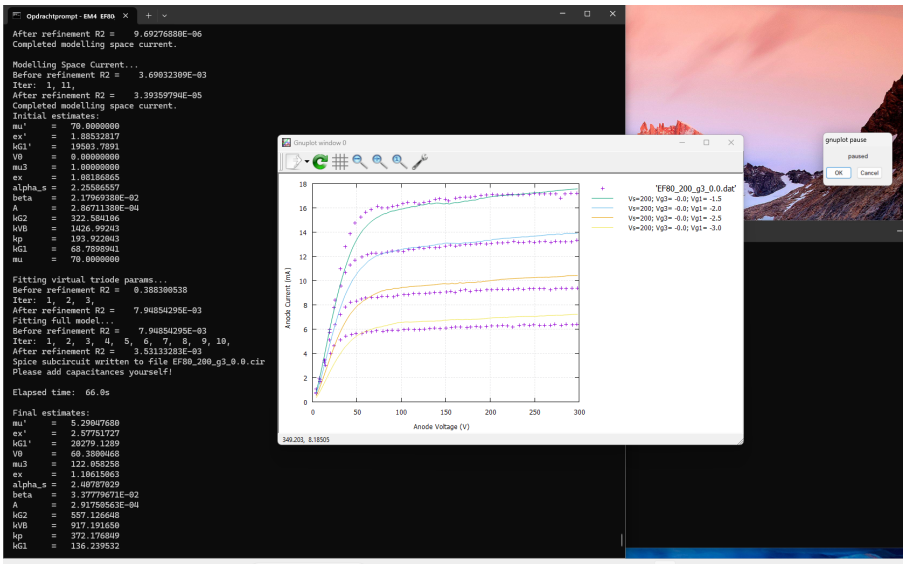

Figure 7: Output of `ExtractModel` for the EF80 with g_3 modulation example with the `EF80g3.ini` initialization file.

References

[Gnu13] www.gnuplot.info, 2013. Release used in preparing this document is: 09.10.2013: Release gnuplot 4.6.4.

[Kor01] Norman Koren. http://www.normankoren.com/Audio/Tubemodspice_article.html, 2001.

[Phi13] Frank's electron tube pages. http://www.tubedata.org/, 2013.

[Spa48] Karl Spangenberg. *Vacuum tubes*. McGraw-Hill, 1948.