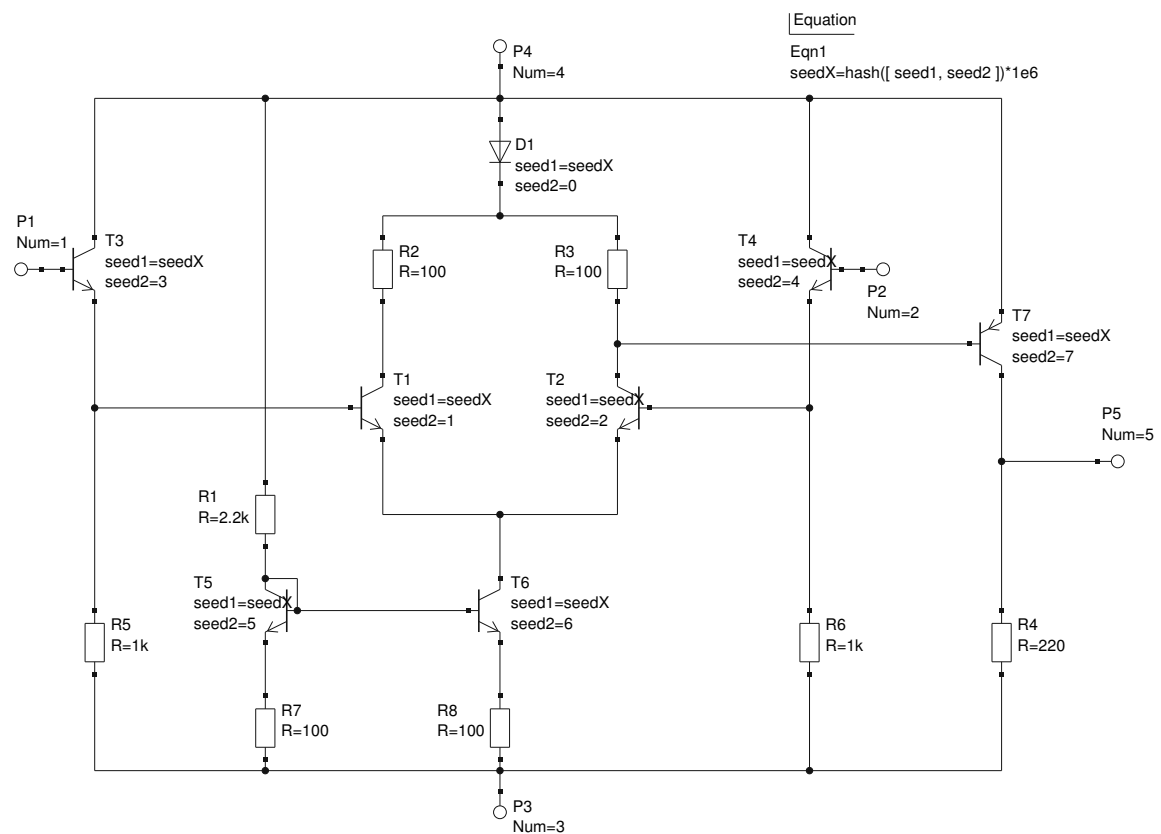


Seminararbeit für 360.019 CAE - Netzwerkanalyse: DC Simulation eines Komparators aus diskreten Halbleiterbauteilen unter Berücksichtigung der Exemplarstreuung der Bauteilparameter mit QUCS

Clifford Wolf
<http://www.clifford.at/>
 Matrikelnummer: e0828563

im November 2010



1 Die Schaltung und ihre Betriebsparameter

Bei der Schaltung handelt es sich um einen vergleichsweise einfachen Komparator aus Bipolartransistoren. Die Schaltung des Komparators ist auf dem Deckblatt abgedruckt. Die Ports haben folgende Belegung:

Port	Belegung
1	Invertierender Eingang ($-$)
2	Nicht-invertierender Eingang ($+$)
3	Negative Versorgung (VEE)
4	Positive Versorgung (VCC)
5	Ausgang

Als Transistoren sollen der BC547B (NPN) und der BC557B (PNP) eingesetzt werden. Die Diode soll eine 1N4148-Schaltdiode sein. Die Wahl fiel auf diese Bauteile weil ich sie griffbereit gehabt habe.

Die Schaltung ist für eine Versorgungsspannung von 9 V ausgelegt. Die Eingangspegel sollten im Betrieb nicht näher als 2 V an die VEE Spannung herangehen, da der Strom vom Eingang bis zum VEE-Potential durch zwei Basis-Emitter-Strecken und eine Kollektor-Emitter-Strecke fließen muss.

2 Entwurf der Schaltung und erste Simulation

Für den ersten Entwurf der Schaltung wurden die Standard-Transistor- und Dioden-Modelle von QUCS verwendet.

Ich habe die Schaltung als eigenes Modul gekapselt und habe ein Symbol für die Simulation gezeichnet.

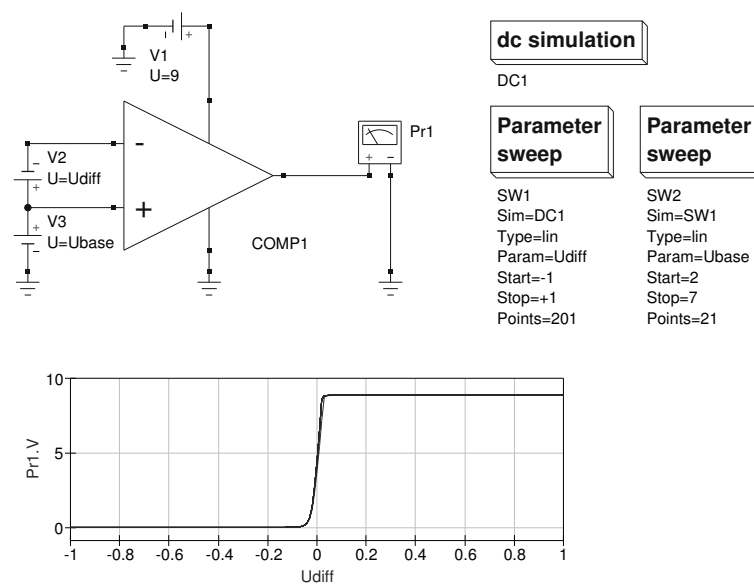
Die erste Schaltung hat die Diode D1 noch nicht beinhaltet, und statt des Stromspiegels aus T5 und T6 wurde ein einfacher Widerstand als Näherung für eine Stromquelle eingesetzt. Auch waren die Widerstandswerte verschiedenen von den Widerstandswerten der endgültigen Schaltung.

Die ersten Simulationen haben schnell gezeigt, dass die Schaltung mit einer Versorgungsspannung von 9 V und einem Widerstand als Stromquelle in einem viel zu kleinen Spannungsbereich ordentlich funktioniert. Daher wurde der Widerstand durch einen Stromspiegel (T5 und T6) ersetzt.

Außerdem wurde schnell klar, dass es sinnvoller ist, R2 und R3 relativ klein zu halten und mit einer Diode den Arbeitspunkt an der Basis von T4 nahe

an den aktiven Bereich zu bringen. (Der erste Ansatz hat darauf abgezielt R2 und R3 so zu dimensionieren, dass der Spannungsabfall an R3 allein genügt um T7 zu öffnen.)

In der ersten Simulation wurde der nicht-invertierende Eingang auf 21 verschiedene Spannungen im Intervall von 2 bis 7 Volt gelegt (also das Intervall mit einer Schrittweite von 250 mV durchlaufen) und für jede dieser Spannungen der Ausgangspegel in Abhängigkeit von der Differenzspannung zwischen nicht-invertierendem und invertierendem Eingang für den Bereich ± 1 V geplottet:



Der Eindruck einer einzigen Übertragungskennlinie in der Abbildung entsteht durch die gute Gleichtaktunterdrückung dieser idealisierten Schaltung. Hier wurden aber tatsächlich 21 Kennlinien für unterschiedliche Arbeitspunkte übereinander gezeichnet.

3 Erstellen eines einfachen 1N4148-Dioden-Modells

Für weitere Simulationen wollte ich zunächst Modelle für die verwendeten Bauteile erstellen. (Die fertigen Modelle aus der Bauteilbibliothek von QUCS zu verwenden wäre erstens unsportlich gewesen und zweitens wollte ich in späterer Folge Monte-Carlo-Simulationen für die Exemplarstreuungen durchführen, und die Bauteile aus der QUCS Bibliothek beinhalten die dazu notwendigen Informationen nicht.)

Die Diode wird in der zu untersuchenden Schaltung von 3 bis 4 mA durchflossen. Also habe ich im ersten Schritt im Datenblatt graphisch eine Sekante durch die (logarithmische) Diodenkennlinie in diesem Strombereich gelegt:

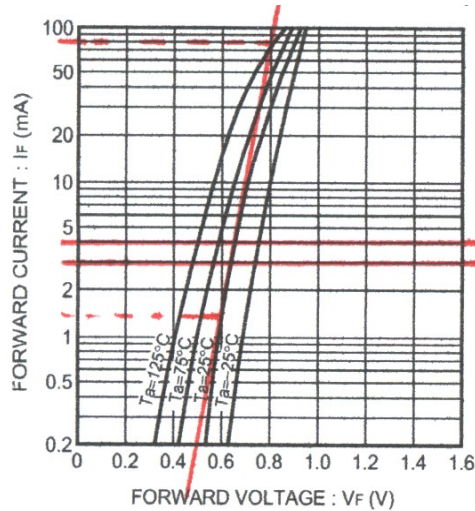


Fig. 1 Forward characteristics

Für diese logarithmisch linearisierte Kennlinie habe ich dann zwei Spannungs-Strom-Tupel abgelesen:

Spannung	Strom
0,6 V	1,2 mA
0,8 V	80 mA

QUCS verwendet im einfachsten Fall die Shockley-Gleichung zur Simulation einer Diode. Dazu sind ein Sättigungssperrstrom I_S und ein Emissionskoeffizient n als Modellparameter anzugeben. Aus den oben ausgelesenen Spannungs-Strom-Tupeln und der Shockley-Gleichung können diese Parameter leicht ermittelt werden:

$$I = I_S(e^{\frac{U}{nU_T}} - 1) \approx I_S \cdot e^{\frac{U}{nU_T}}$$

$$\begin{aligned} I_1 &= I_S \cdot e^{\frac{U_1}{nU_T}} \\ I_2 &= I_S \cdot e^{\frac{U_2}{nU_T}} \end{aligned} \quad \Rightarrow \quad n = \frac{U_2 - U_1}{\ln(I_2/I_1)U_T}, \quad I_S = I_1 / e^{\frac{U_1}{nU_T}}$$

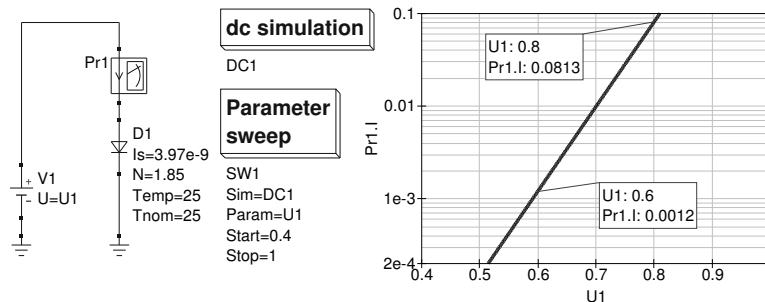
Wenn man für $U_T \approx 25,7 \text{ mV}$ annimmt¹ und die oben ermittelten Tupel einsetzt so erhält man die Modellparameter für die Diode:

$$n \approx \frac{0,8 \text{ V} - 0,6 \text{ V}}{\ln(80 \text{ mA}/1,2 \text{ mA}) \cdot 25,7 \text{ mV}} \approx 1,85$$

$$I_S = 1,2 \text{ mA} / e^{\frac{0,6 \text{ V}}{1,85 \cdot 25,7 \text{ mV}}} \approx 3,97 \cdot 10^{-9} \text{ A}$$

Ich bezweifle, dass diese Parameter physikalisch korrekt sind. Man sieht schon an der Rechtskrümmung der Kennlinie im Datenblatt, dass es einen Ohmschen Anteil geben muss der hier nicht richtig modelliert wurde. Mathematisch liefert dieses Modell aber gute Werte für jenen Bereich der Diodenkennlinie, der für diese Schaltung von Interesse ist.

Als einfachen Test für das Dioden-Modell habe ich den Simulator wieder die linearisierte Kennlinie errechnen lassen:



3.1 1N4148-Dioden-Modellparameter für die Monte-Carlo-Simulation

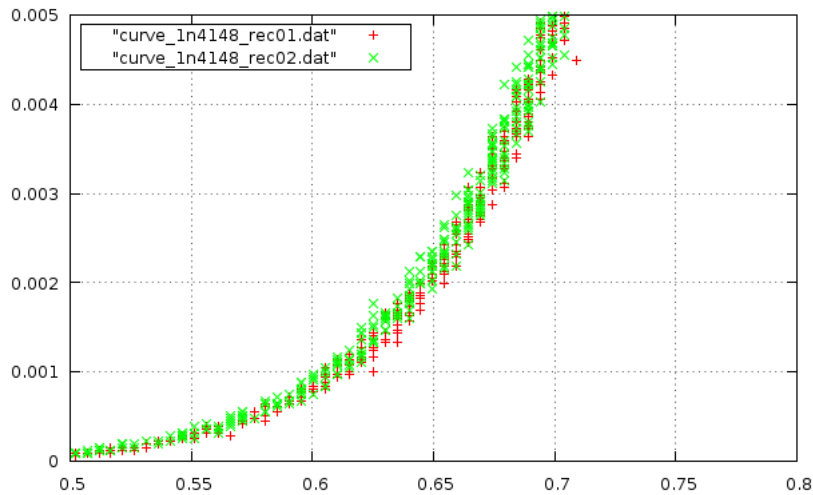
Leider beinhaltet das 1N4148-Datenblatt keine Informationen zu Exemplarstreuungen. Also habe ich einen einfachen Kennlinienschreiber auf Basis des Arduino-AVR-Development-Boards aufgebaut, der die ADCs des AVR-Mikrocontrollers nutzt um die Kennlinien von Dioden und Transistoren aufzuzeichnen.

Der Sourcecode zum Kennlinienschreiber (AVR-Firmware und die Linux-PC-Software) sowie die Schaltpläne für die Beschaltung des Messaufbaus stehen unter folgender URL zum Download bereit:

<http://svn.clifford.at/handicraft/2010/curvetracer/>

¹für 25 °C ergibt sich nach $U_T = \frac{kT}{e} \approx \frac{1,38 \cdot 10^{-23} \text{ J/K} \cdot (273,15 \text{ K} + 25 \text{ K})}{1,602 \cdot 10^{-19} \text{ As}} = 25,7 \text{ mV}$.

Mit Hilfe dieses Kennlinienschreibers konnte ich provisorisch die Kennlinien von zwei 1N4148-Dioden aufnehmen:



Man erkennt auf dem Plot gut das Quantisierungsrauschen des ADC auf der Spannungsachse. Dieses Quantisierungsrauschen ist auch für das große Rauschen auf der Stromachse verantwortlich. Darüber hinaus ist davon auszugehen, dass die absoluten Spannungs- und Stromangaben nicht völlig offsetbereinigt sind.

Es wäre sicherlich ein Leichtes mit einem geeigneten Messverstärker und einem sorgfältigen Abgleichprozess deutlich bessere Resultate zu erzielen. Da das Thema dieser Seminararbeit aber die Schaltungssimulation ist habe ich mich mit diesem einfachen Messaufbau zufriedengegeben.²

Man erkennt, dass das Rauschen der Messung deutlich größer ist als die systematischen Unterschiede zwischen den beiden Dioden. Also nehme ich für die weiteren Simulationen dieses Rauschen als die maximale Exemplarstreuung an.

Als Modellvorstellung für meine Monte-Carlo-Simulation verwende ich das Berechnungsverfahren aus dem vorhergehenden Abschnitt, verrausche aber den Strom an den Stützpunkten um $\pm 10\%$. Das entspricht der Schwankungsbreite von 1 mA bei 5 mA aus der oben beschriebenen Messung.

²Ebenso erlaubt eine Messung an zwei Dioden (wahrscheinlich des gleichen Herstellers und möglicherweise aus der gleichen Charge) natürlich noch keine Schlüsse über die tatsächlichen Exemplarstreuungen eines Bauteiltyps.

3.2 Monte-Carlo-Simulation der 1N4148-Diode

Um die Monte-Carlo-Simulation der Diode durchzuführen musste ich zunächst QUCS um eine Funktion erweitern. Zwar hat QUCS eine `random()` Funktion, diese ist aber ungeeignet zum Zeichnen von Kurvenscharen mit zufälligen Parametern je Kurve, denn die `random()` Funktion liefert bei jedem Aufruf – also für jeden Stützpunkt der Kurve – einen neuen zufälligen Wert zurück.

Desswegen habe ich QUCS um eine `hash()` Funktion erweitert. Diese `hash()` Funktion bekommt als Parameter einen Vektor beliebiger Länge und liefert eine reelle Semizufallszahl im Intervall $[0; 1]$ zurück, die mit Hilfe einer Hashfunktion über die ganzzahligen Anteile der Vektorkomponenten ermittelt wird.

Die dazu notwendigen Änderungen am QUCS-Sourcecode als Patch im Unified-Diff-Format:

```
diff --git a/qucs-core/src/applications.h b/qucs-core/src/applications.h
index 503db9f..d88b019 100644
--- a/qucs-core/src/applications.h
+++ b/qucs-core/src/applications.h
@@ -973,6 +973,7 @@ struct application_t eqn::applications[] = {
    { "random", TAG_DOUBLE, evaluate::rand, 0, { TAG_UNKNOWN } },
    { "srandom", TAG_DOUBLE, evaluate::srand_d, 1, { TAG_DOUBLE } },
+   { "hash", TAG_DOUBLE, evaluate::hash_v, 1, { TAG_VECTOR } },

    { "vector", TAG_VECTOR, evaluate::vector_x, -1, { TAG_UNKNOWN } },
    { "matrix", TAG_MATRIX, evaluate::matrix_x, -1, { TAG_UNKNOWN } },
diff --git a/qucs-core/src/evaluate.cpp b/qucs-core/src/evaluate.cpp
index 0530d8f..d6660c3 100644
--- a/qucs-core/src/evaluate.cpp
+++ b/qucs-core/src/evaluate.cpp
@@ -27,6 +27,7 @@
@@ -27,6 +27,7 @@
# include <config.h>
#endif

+#include <openssl/sha.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
@@ -4339,6 +4340,25 @@ constant * evaluate::srand_d (constant * args) {
}
}

+constant * evaluate::hash_v (constant * args) {
+   _ARVO (v1);
+   _DEFD ();
+   int d = 0;
+   unsigned char data[SHA_DIGEST_LENGTH];
+   SHA_CTX c;
+   SHA1_Init(&c);
+   for (int i = 0; i < v1->getSize (); i++) {
+       int v = abs(v1->get(i));
+       data[0] = (v >> 24) & 0xff;
+       data[1] = (v >> 16) & 0xff;
+       data[2] = (v >> 8) & 0xff;
+       data[3] = (v >> 0) & 0xff;
+       SHA1_Update(&c, data, 4);
+   }
+   SHA1_Final(data, &c);
+   _RETD(((unsigned int*)data) / (double)0xffffffffU);
+}
+
// ***** immediate vectors *****
constant * evaluate::vector_x (constant * args) {
   _DETV ();
diff --git a/qucs-core/src/evaluate.h b/qucs-core/src/evaluate.h
```

```

index 925c5d2..073da8f 100644
--- a/qucs-core/src/evaluate.h
+++ b/qucs-core/src/evaluate.h
@@ -758,6 +758,7 @@ public:

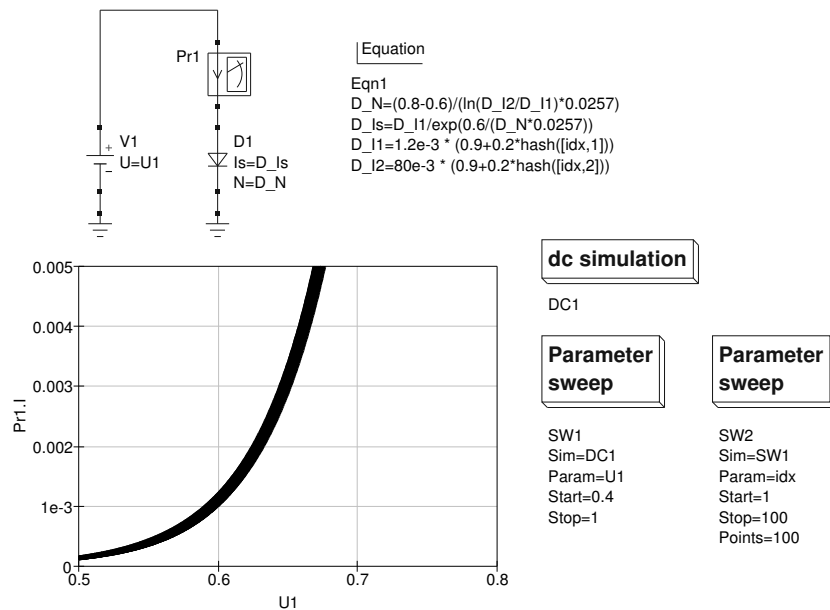
    static constant * rand (constant *);
    static constant * srand_d (constant *);
+   static constant * hash_v (constant *);

    static constant * vector_x (constant *);
    static constant * matrix_x (constant *);

```

Beim Übersetzen von QUCS mit diesem Patch muss QUCS mit der OpenSSL-Library gelinkt werden.

Mit Hilfe dieser `hash()` Funktion war es ein Leichtes die Monte-Carlo-Simulation der 1N4148-Diode umzusetzen:



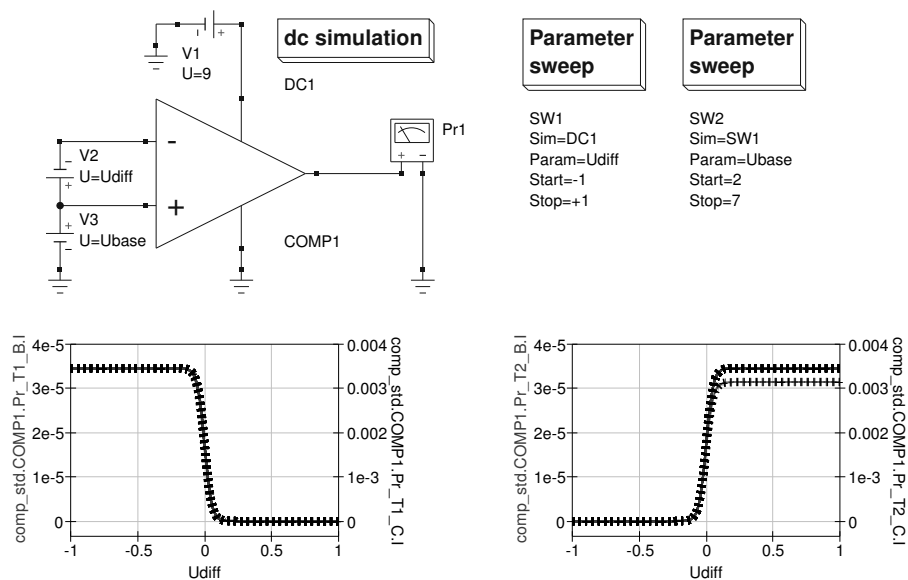
4 Ein einfaches BC547B (BC557B)-Modell

Die Modellierung der Transistoren in der Schaltung ist einfach, weil die Transistoren immer im aktiven Bereich arbeiten. Das heißt, der Kollektor kann immer genug Strom liefern, so dass das Verhältnis zwischen Basisstrom und Kollektorstrom der konstanten Stromverstärkung β des Transistors entspricht. Außerdem sind die Kollektorströme so klein, dass der Early-Effekt eine untergeordnete Rolle spielt und daher nicht im Transistor-Modell berücksichtigt werden muss.

Eine Ausnahme stellt natürlich der PNP-Transistor T7 dar: Er arbeitet zum

Teil im Schaltbetrieb. Aber das genaue Verhalten des Transistors spielt eine untergeordnete Rolle sobald er einmal durchgeschaltet ist. Der möglicherweise falsch simulierte Basisstrom an T7 beeinflusst zwar die internen Ströme und Spannungen im Differenzverstärker aus T1 und T2, hat aber keine wesentlichen Auswirkungen auf das von aussen beobachtbare Verhalten der Schaltung.

Während die Feststellung, dass T3 bis T6 im aktiven Bereich arbeiten, trivial ist, und T7 wie oben beschrieben eine Ausnahme darstellt, ist die Behauptung für T1 und T2 – obwohl wahr – nicht so trivial einsichtig. Deshalb habe ich zur Sicherheit eine Simulation mit Current-Probes an den Basen und Kollektoren von T1 und T2 durchgeführt:

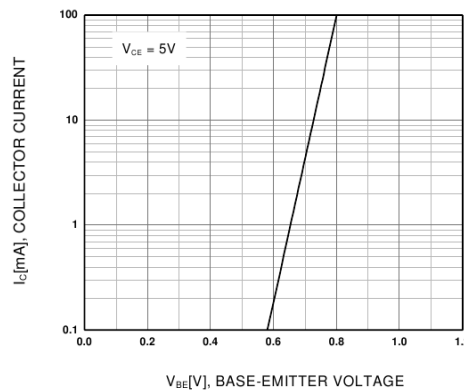


Man erkennt gut, dass der Kollektorstrom immer ein 100faches des Basisstroms ist – und das ist auch das β des Standard-Transistormodells von QUCS.

Der eine Ausreißer rechts stellt den Fall dar, dass der nicht-invertierende Eingang mit 2 V vorgespannt wird und der invertierende dann auf unter 2 V absinkt – also der Komparator eigentlich außerhalb seiner Spezifikation betrieben wird. Aber selbst hier stimmt das Verhältnis zwischen Basis- und Kollektorstrom noch – die Stromquelle aus T5 und T6 kann in diesem Fall aber nicht mehr den sonst üblichen Strom liefern.

Darüberhinaus wird bei allen Transistoren die Basis über die Spannung angesteuert. Durch diese Maßnahmen sollte sich die Stromverstärkung im wesentlichen nur auf den Eingangswiderstand der Schaltung auswirken.

Damit kann das Modell des BC547B genau so erstellt werden wie es bereits bei der Diode geschehen ist. Zunächst ziehe ich die Kennlinie aus dem Datenblatt heran und ermittle zwei Stützpunkte für das Lösen der Schockley-Gleichung. Da die logarithmische Kennlinie im Datenblatt nicht gekrümmt ist, war es nicht notwendig, eine linearisierte Kennlinie zu konstruieren:

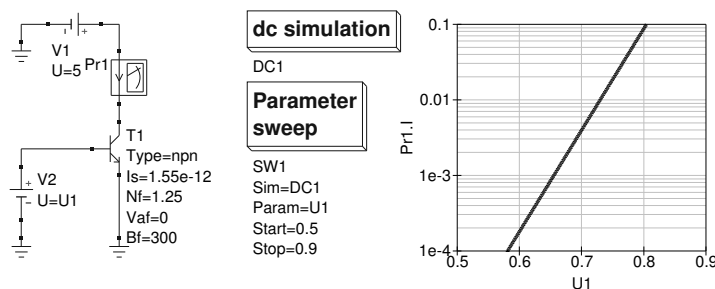


Spannung	Strom
0,6 V	0,2 mA
0,8 V	100 mA

$$n \approx 1,25$$

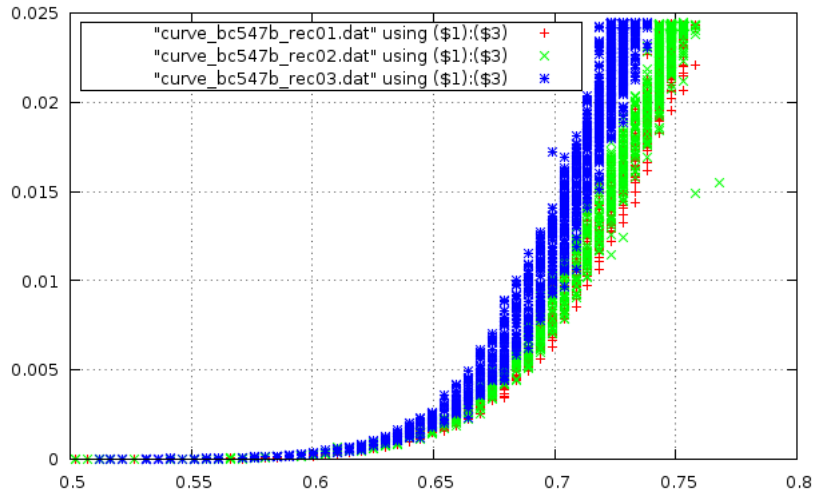
$$I_S \approx 1,55 \cdot 10^{-12}$$

Diese berechneten Werte habe ich dann zuerst mit einer einfachen Simulation nochmals überprüft:



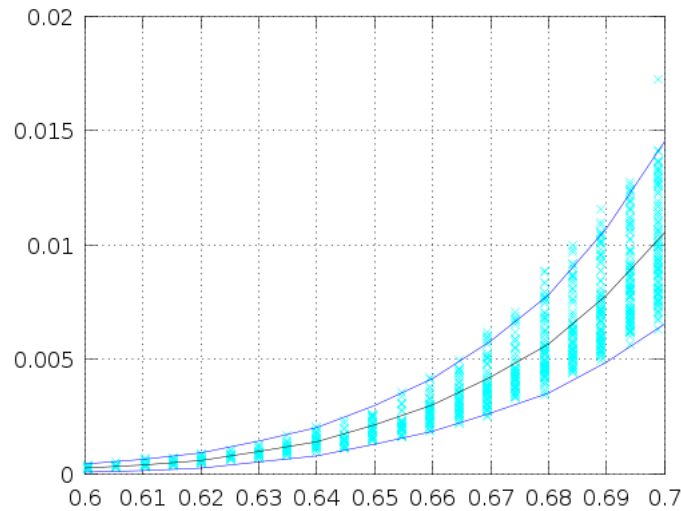
Anschließend habe ich mit meinem improvisierten Kennlinienschreiber die Kennlinien von drei BC547B-Transistoren aufgenommen. Dazu wurde der Kollektor mit 5 V vorgespannt. Am Emitter wurde ein $150\ \Omega$ -Widerstand eingesetzt, der den Transistor gegengekoppelt hat und auch zum Messen des Stroms verwendet worden ist. An die Basis ist dann eine Spannung von bis zu 4.5 V angelegt worden. Aus den ADC-Messwerten von Basis und Emitter wurden dann der Emitter-Strom sowie die Basis-Emitter-Spannung errechnet.

Die Daten aus diesen Messungen habe ich dann wieder geplottet. Hier ist zu beachten, dass die blaue Kennlinie zu einem Transistor eines anderen Herstellers gehört als die grüne und die rote Kennlinie. Während sich die grüne und die rote Kennlinie fast perfekt decken ist die blaue Kennlinie deutlich steiler als die anderen beiden.



Um einfach ein passendes Modell für die Exemplarstreuungen des Transistors zu finden habe ich die Messdaten in Octave (einen freien Matlab-Klon) importiert. Nach einigen Experimenten bin ich auf folgende Modellfunktion gekommen, bei der das Intervall, in dem der Kollektorstrom liegt, als Funktion der Basis-Emitter-Spannung dargestellt wird:

$$I_{C\max} - I_{C\min} = e^{U_{BE} \cdot 31} \cdot 3 \cdot 10^{-12}$$



In der Abbildung sind die Samplewerte als cyane Kreuze dargestellt, die schwarze Linie markiert den Median der Samplewerte und die blauen Linien das Intervall um den Median nach der oben erwähnten Formel.

Die Octave-Befehle zur Analyse der Sampledaten und der Erstellung dieser

Grafik waren:

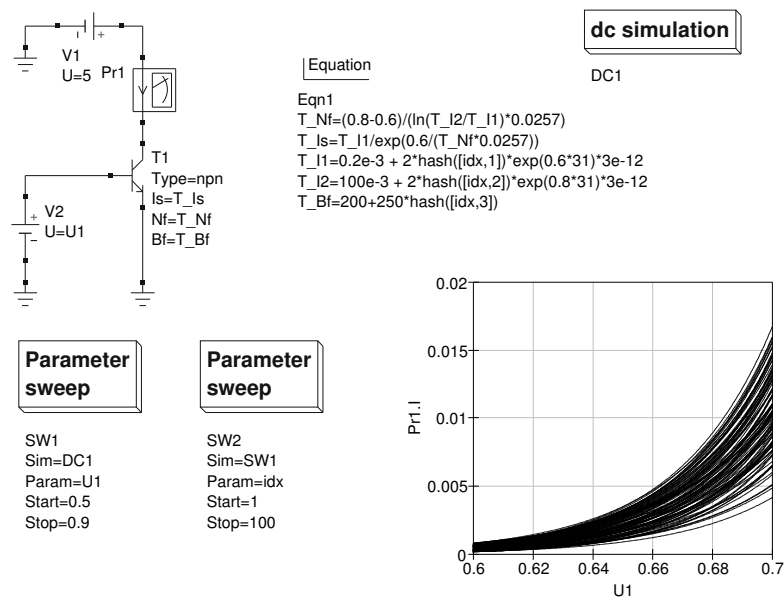
```
# Laden der Sampledaten
d1=load("curve_bc547b_rec01.dat");
d2=load("curve_bc547b_rec02.dat");
d3=load("curve_bc547b_rec03.dat");
d=[d1(:,[1 3]); d2(:,[1 3]); d3(:,[1 3])];

# Einfache Analyse
data=[];
for u=[min(d(:,1)):0.01:max(d(:,1))];
    idx=find((d(:,1) > u-0.01) & (d(:,1) < u+0.01));
    data(end+1,:)= [ u, max(d(idx,2))-min(d(idx,2)), median(d(idx,2)), size(idx) ];
endfor

# Plotten der Modellfunktion und der empirisch ermittelten Intervalle
plot(data(:,1), data(:,2), "b", [0:0.01:0.8], (exp([0:0.01 :0.8]*31)-1)*3e-12, "g");
axis([0 0.8 0 0.02])

# Plotten der Modellfunktion im Kontext der urspruenglichen Samples
plot(d(:,1), d(:,2), "xc", data(:,1), data(:,3), "k",
    data(:,1)*[1 1], data(:,3)*[1 1] + (exp(data(:,1)*31)-1)*3e-12*[-0.5 +0.5], "b");
axis([0.6 0.7 0 0.02])
```

Damit sind alle Informationen für eine Monte-Carlo-Simulation des BC547B beisammen. Leider zeigt der Vergleich zwischen den aufgenommenen Daten und den Kennlinien aus dem Datenblatt, dass die gemessenen Ströme durchwegs höher sind als die Ströme aus der Kennlinie im Datenblatt. Wenn man auf die Kennlinie im Datenblatt mein Modell für das Intervall, in dem der Kollektorstrom liegen kann, anwendet, so kommt man damit zum Teil sogar auf Emmissionskoeffizienten kleiner 1. Daher habe ich mich entschieden statt dessen die Ströme im Datenblatt als untere Grenze anzunehmen und das Intervall aus meinem Modell doppelt dazuzuaddieren. Auf diese Weise erhält man eine Kennlinienschar die die gemessenen Kennlinien gut umfasst:

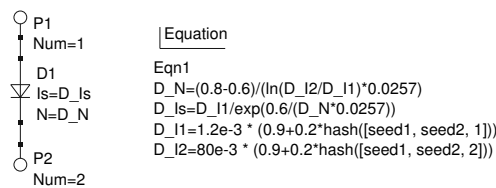


Als Intervall für die Stromverstärkung des Transistors habe ich das Intervall aus dem Datenblatt verwendet. Da in dieser Testschaltung die Basis direkt mit einer Spannungsquelle verbunden ist, dürfte die Stromverstärkung keinen Einfluss auf das resultierende Kennlinienbild haben.

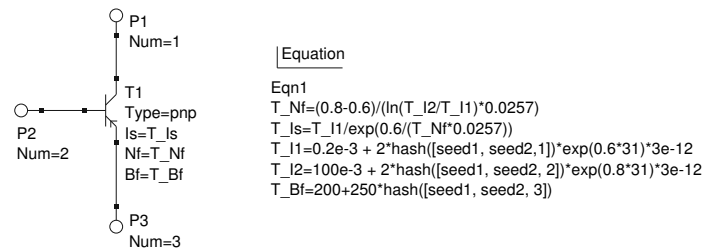
5 Module für die Monte-Carlo-Simulation der Bauteile

Um eine komplette Schaltung sinnvoll simulieren zu können müssen die Implementierungsdetails der Bauteile mit den Gleichungen für die Exemplarstreuungen in eigene Module gekapselt werden. Zu diesem Zweck habe ich die Module 1N4148_MC, BC547B_MC und BC557B_MC erstellt und passende Symbole gezeichnet.

Die Innenschaltung des 1N4148_MC-Moduls:



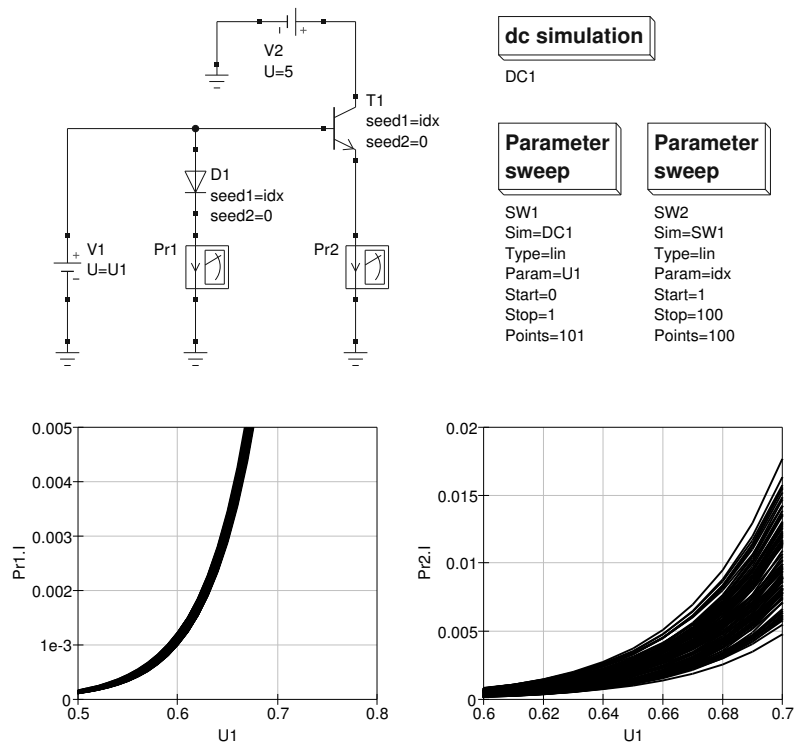
Die Innenschaltung des BC547B_MC-Moduls:



Die Innenschaltung des BC557B_MC-Moduls ist mit der des BC547B_MC-Moduls bis auf die Verwendung eines inneren PNP-Transistormodells ident.

Alle diese Module besitzen die zwei Parameter **seed1** und **seed2**. Beide Parameter werden in der internen Hash-Funktion verwendet. Durch die Verwendung von zwei solchen Parametern kann z.B. ein Parameter dazu verwendet werden, die Bauteile durchnummerieren, und der andere, um verschiedene Variationen in einem Parameter-Sweep durchzuprobieren.

Abschließend wurden die vorhergehenden Tests nochmals mit den neuen gekapselten Modulen wiederholt:



6 Monte-Carlo-Simulation des Komparators

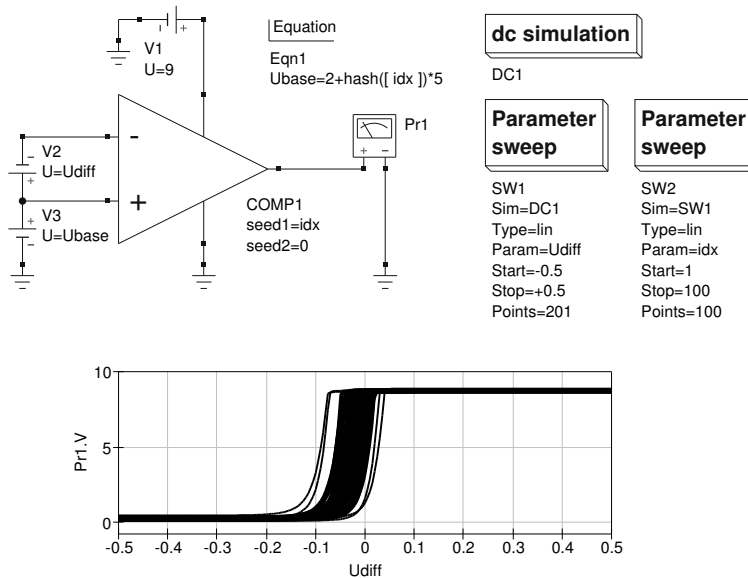
Letztendlich habe ich dann die Transistoren und Dioden in der Komparator-Schaltung durch meine eigenen Symbole mit den **seed**-Parametern ersetzt. (Schaltplan siehe Deckblatt)

Die Komparator-Schaltung selbst hat ebenfalls die zwei Modulparameter **seed1** und **seed2** erhalten. Diese beiden Parameter werden in der Schaltung mit meiner Hash-Funktion zu einer **seedX**-Variablen kombiniert, die allen Transistoren und der Diode als **seed1** Wert übergeben wird. Als **seed2**-Werte wurden einfach fortlaufende Nummern für die Bauteile vergeben.

Nach diesem Schema könnte man jetzt z.B. mehrere der Komparatoren auch wieder zu einer größeren Schaltung zusammenfügen und eine Monte-Carlo-Simulation dieser größeren Schaltung durchführen u.s.w.

Zum Schluss habe ich eine Testschaltung für die Monte-Carlo-Simulation

der Komparator-Schaltung erstellt:



Man kann gut erkennen, dass der Komparator zwar auch mit nicht-identischen Transistoren noch gut arbeitet, aber vor allem die Offsetspannung nun bei zwei solchen Komparatoren und bei zwei verschiedenen Arbeitspunkten um bis zu 150 mV verschieden sein kann.

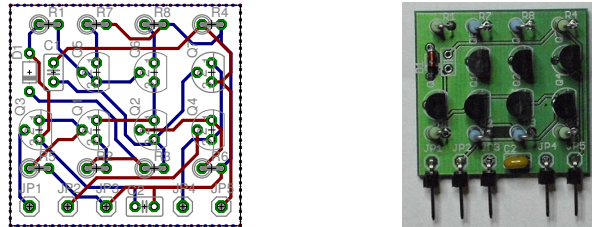
Die Qualität solcher Aussagen hängt natürlich stark davon ab, wie gut die Exemplarstreuungen der Bauteilwerte modelliert werden – und nicht zuletzt aufgrund der schlechten messtechnischen Möglichkeiten (schnell zusammen-improvisierter Kennlinienschreiber) sowie der viel zu kleinen Stichprobe an Bauteilen ist die hier besprochene Simulation wahrscheinlich wenig aussagekräftig.

Aber die verwendeten Methoden können genau so wie hier dargestellt verwendet werden, um Exemplarstreuungen in der Simulation zu berücksichtigen, sofern ausreichend gute Modelle der Exemplarstreuungen verfügbar sind.

7 Prototyp

Nachdem ich einiges an Zeit in die Simulation der Schaltung gesteckt hatte wollte ich natürlich wissen ob die tatsächliche Schaltung in ihrem Verhalten innerhalb der Simulationsergebnisse liegt. Also habe ich ein kleines PCB für die Schaltung mit Eagle entworfen und 6 Prototypen hergestellt.

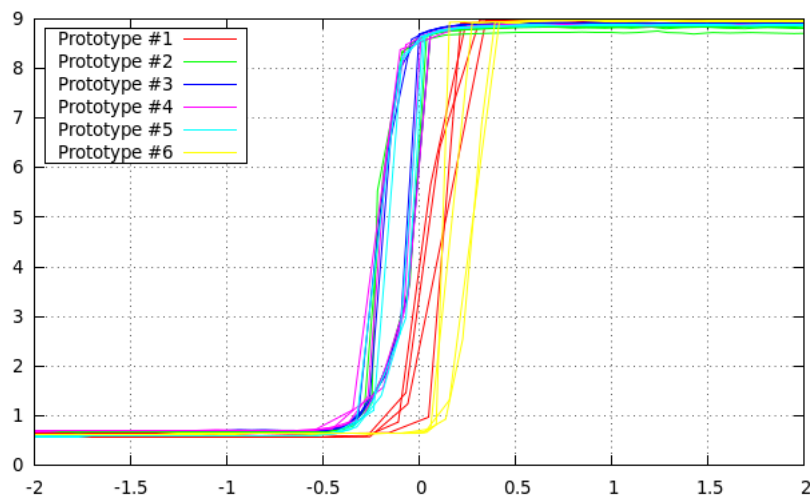
Die folgenden Abbildungen zeigen das PCB in Originalgrösse und ein Photo eines fertigen Prototyps:



Die Schaltung wurde für den Prototyp noch um die beiden Kondensatoren C1 und C2 erweitert. C1 verbindet Basis und Kollektor von T7 und blieb in den Tests unbestückt. Er wurde vorgesehen, um Resonanzen zu unterdrücken falls die Schaltung einmal als Operationsverstärker in einer Beschaltung mit Gegenkopplung betrieben werden soll. C2 ist einfach ein Stützkondensator für die Versorgungsspannung.

Leider musste ich zur Bestückung der Prototypen weitere Transistoren einkaufen, so dass die Transistoren auf den Prototypen wieder von einem anderen Hersteller waren als die Transistoren, die ich für die Erstellung des Transistormodells für die Simulation herangezogen habe.

Ich habe mit dem improvisierten Kennlinienschreiber die Übertragungskennlinie der Prototypen gemessen:



Auf der X-Achse ist die Spannung zwischen dem invertierenden und dem nicht-invertierenden Eingang, und auf der Y-Achse die Ausgangsspannung in Volt aufgetragen.

Auffällig ist zunächst, dass die Ausgangsspannung nie unter 0,6 Volt absinkt. Ich bin dem nicht weiter nachgegangen, da ich annehme, dass es sich dabei um einen Messfehler (Ende des gültigen Input-Ranges des ADC) handelt und nicht um ein Problem bei der Ansteuerung von T7.

Des Weiteren fällt auf, dass die Offsetspannung sogar in einem Intervall von 0,5 V abhängig von Prototyp und Lage des Arbeitspunktes schwankt, also deutlich stärker als die Simulation hätte vermuten lassen.

8 Conclusio

Für eine wirklich aussagekräftige Simulation wären deutlich bessere Modelle notwendig als die, die ich in diesem Experiment erstellt habe.

Wahrscheinlich wäre es auch sinnvoll, nicht ein Modell für die Exemplarstreuung zu erstellen, sondern statt dessen halbwegs hochauflösend die Kennlinien einer grossen Stichprobe aufzunehmen und dann für jedes Bauteil aus der Stichprobe die Modellparameter zu ermitteln. In der Monte-Carlo-Simulation könnte dann einfach zufällig einer der gemessenen Modellparametersätze verwendet werden. Auf diese Weise würden auch die statistische Streuung sowie die Abhängigkeit zwischen den Modellparametern richtig modelliert werden. Dazu wäre eine weitere Erweiterung von QUCS zum Einlesen von tabellarischen Daten sinnvoll.

9 Dateien

Dieses PDF-Dokument enthält angehängte Dateien. Die meisten PDF-Reader erlauben es, diese Dateien zu extrahieren. Um das zu tun muss man mit der rechten (zweiten) Maustaste auf den rot gesetzten Dateinamen klicken und den entsprechenden Eintrag im Kontextmenü auswählen.

Die jeweils aktuelle Version dieser Seminararbeit kann als PDF-Datei unter folgender URL heruntergeladen werden:

http://www.clifford.at/papers/2010/sem_cae_netana/

Patch zu QUCS für die hash() Funktion:

qucs_hash.patch

QUCS Schematic Files:

<code>1n4148_mc.sch</code>	<code>1n4148_test1.sch</code>	<code>1n4148_test2.sch</code>
<code>bc547b_mc.sch</code>	<code>bc547b_test1.sch</code>	<code>bc547b_test2.sch</code>
<code>bc557b_mc.sch</code>	<code>compdemo1.sch</code>	<code>compdemo_std1.sch</code>
<code>compdemo_std2.sch</code>	<code>comp.sch</code>	<code>comp_std.sch</code>
<code>mctest.sch</code>		

Die mit dem Kennlinienschreiber aufgezeichneten Datenfiles:

<code>curve_1n4148_rec01.dat</code>	<code>curve_1n4148_rec02.dat</code>
<code>curve_bc547b_rec01.dat</code>	<code>curve_bc547b_rec02.dat</code>
<code>curve_bc547b_rec03.dat</code>	<code>curve_compdata.dat</code>

EAGLE-Design für den Schaltungsprototyp:

`eagle_comp.brd` `eagle_comp.sch`

Der L^AT_EX-Quelltext zu diesem Dokument:

`report.tex`

10 Links zur verwendeten Software

Der Kennlinienschreiber:

<http://svn.clifford.at/handicraft/2010/curvetracer/>

QUCS:

<http://qucs.sourceforge.net/>

GNU Octave:

<http://www.gnu.org/software/octave/>

Gnuplot:

<http://www.gnuplot.info/>