INGENIEURPROJEKT 2000/2001

PSPICE-SPEZIALMODELLE

ausgeführt an der HTL Saalfelden

Abteilung Elektrotechnik von

August Golser & Manuel Laggner

Schüler des V. Jahrganges ET

im Schuljahr 2000/2001

Projektbetreuer: Dipl.-Ing. Josef Stadler

Vorwort

Diese Dokumentation soll dazu dienen, nachfolgenden Schülern sowie den Professoren, die Erstellung von mathematischen Modellen und sogenannten Makromodellen in PSpice so einfach wie nur möglich zu gestalten. PSpice ist mittlerweile zum Defacto-Industrie-Standard in punkto Schaltungssimulation avanciert und stellt durch die Möglichkeit, Simulationen auch im Zeit- beziehungsweise Freuquenzbereich durchführen zu können, ein ausgereiftes Werkzeug zur Entwicklung jeglicher Form von Schaltungen dar.
Mit PSpice ist es auch möglich, Bauteile unter extremen Temperatur- und Worst-Case-Bedingungen zu testen. In diesem Zusammenhang ist in erster Linie die Vermeidung der Zerstörung teurer Bauelemente von Bedeutung.
Ein Teil unseres Ingenieurprojektes beschäftigte sich mit der Erstellung von regelungstechnischen Grundgliedern und der Simulation des Verhaltens im Zeit- bzw. Frequenzbereich. Diese Simulationen umfassen:

- Sprungantwort
- Bodediagramm
- Ortskurve

Der zweite Teil des Projektes bestand darin, die Erstellung solcher Bauteile auf drei verschiedene Arten zu dokumentieren. In dieser Dokumentation sind alle wichtigen Schritte zur Erstellung von (Makro-) Modellen genau beschrieben.
Um die folgenden Anleitungen verwenden zu können, sind grundlegende Kenntnisse im Programm PSpice notwendig!

Grundsätzlich ist diese Dokumentation in vier Teile gegliedert:

- Die regelungstechnischen Grundglieder
- Beschreibung zur Erstellung von mathematischen Modellen
- Beschreibung zur Erstellung von Subcircuits
- Erläuterung der Template Syntax

Inhaltsverzeichnis

Regelungstechnische Grundglieder

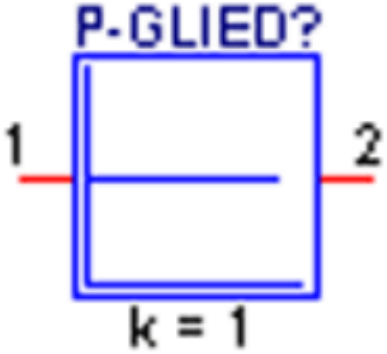
Modellerstellung

Subcircuits

Template Syntax

Regelungstechnische Grundglieder

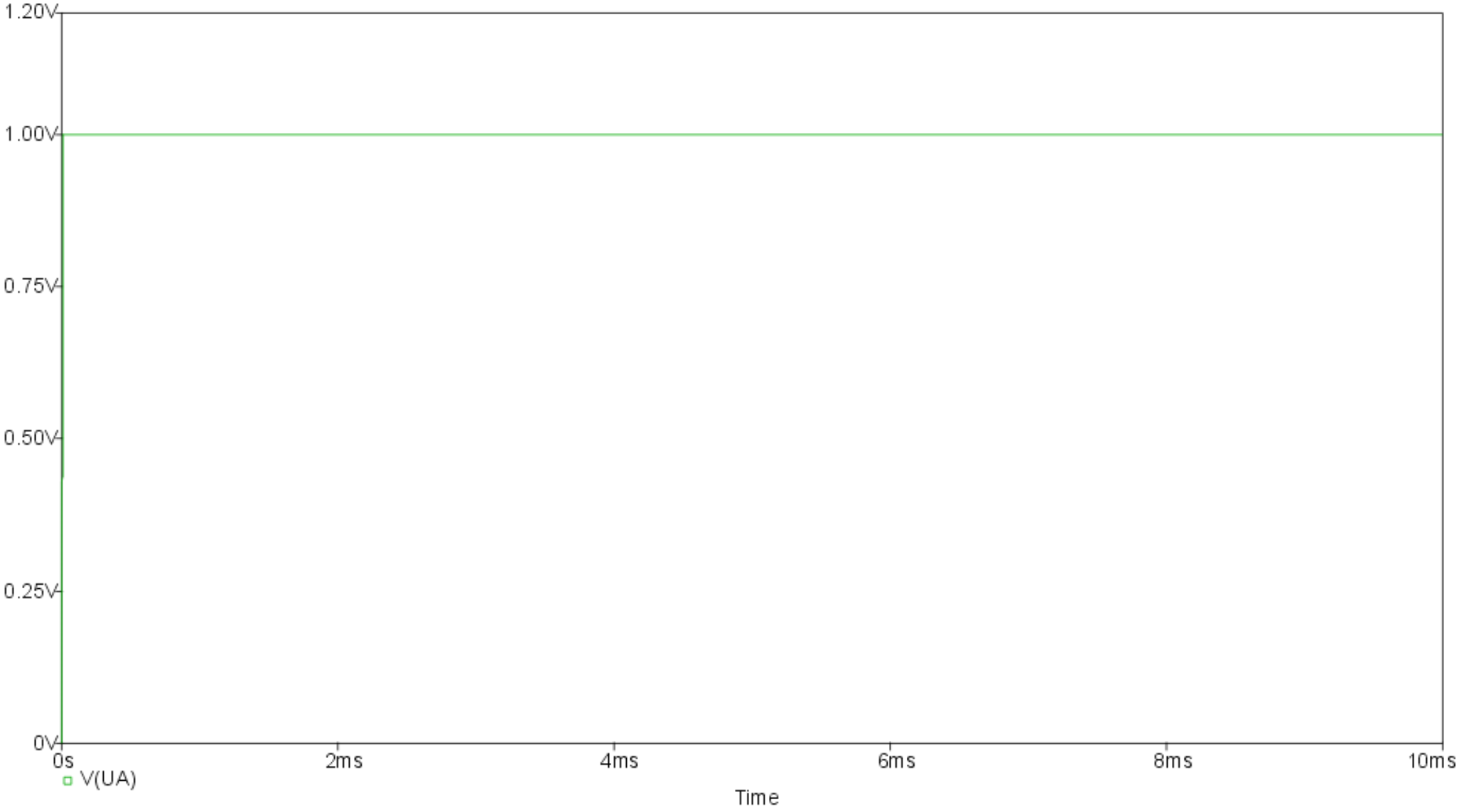
P - Element



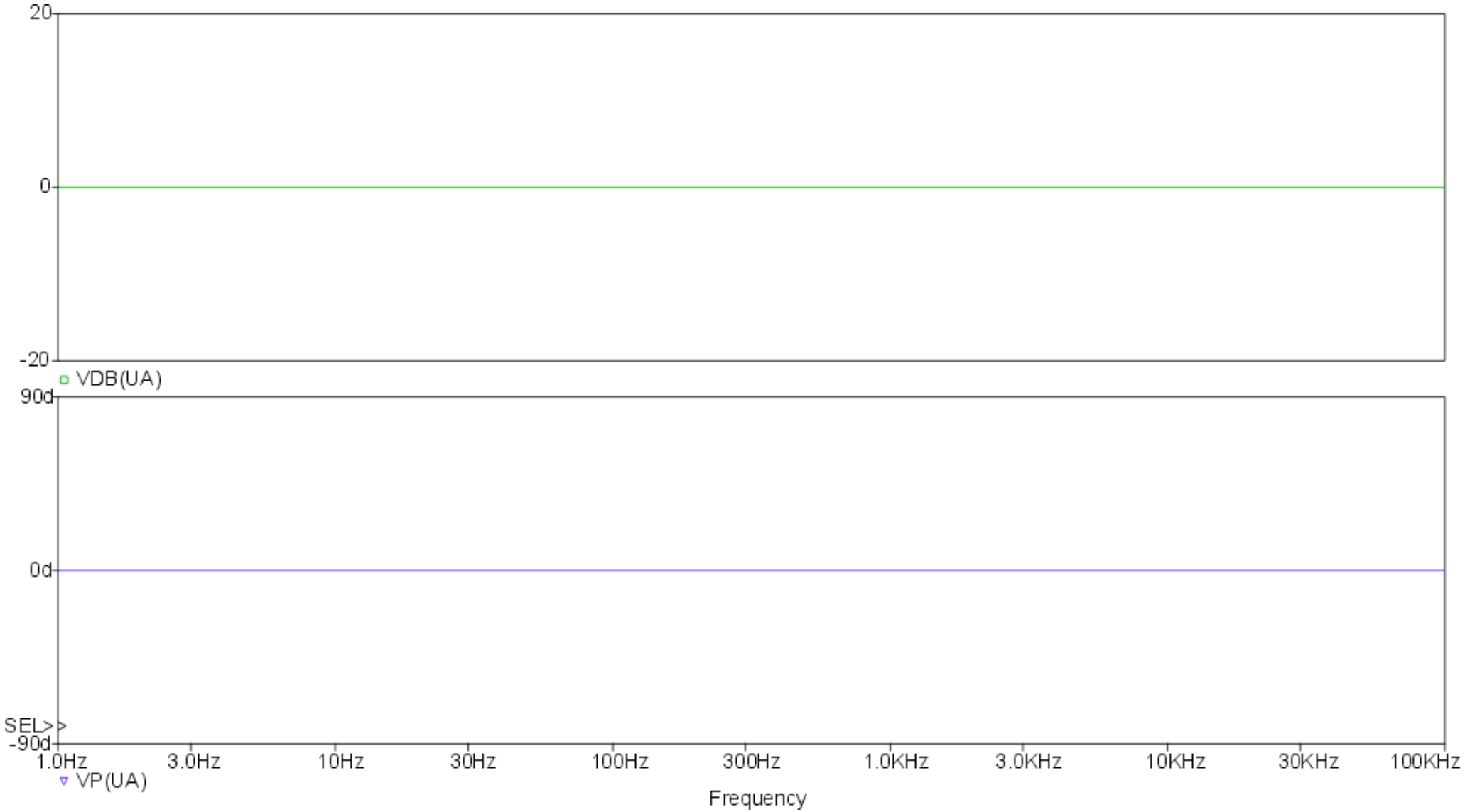
$$G(s) = k$$

Da unabhängig davon, ob Xe nach einer Sprung- oder Sinusfunktion verläuft, das Verhältnis Xa/Xe hier eine Konstante (k) ist, stimmen Übergangsfunktion, Übertragungs-funktion und Frequenzgang überein. Sie sind zeit- bzw. frequenz-unabhängig.

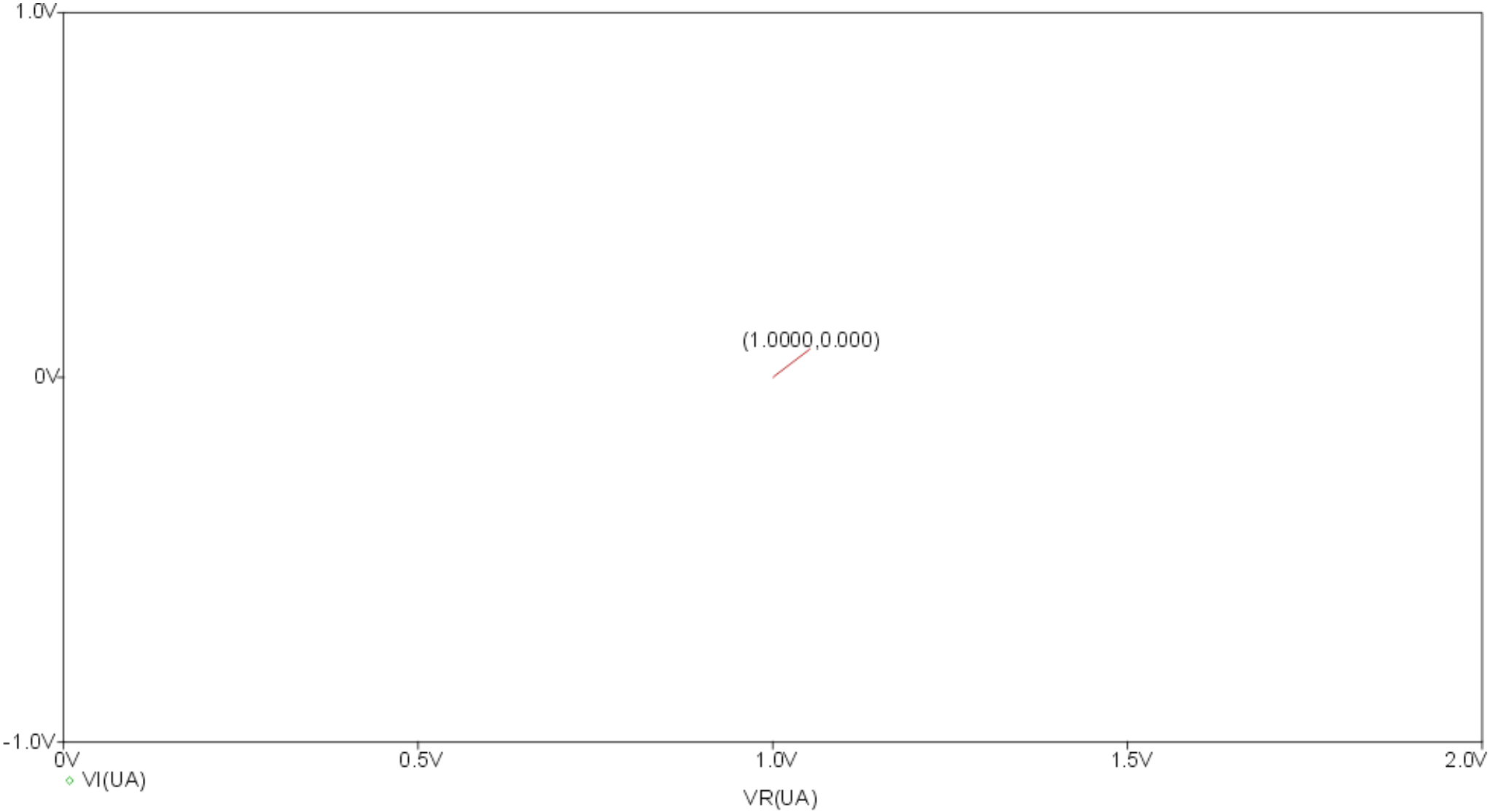
Sprungantwort



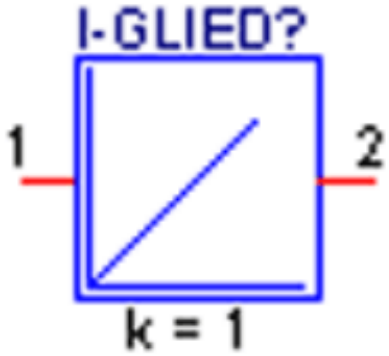
Bodediagramm



Ortskurve



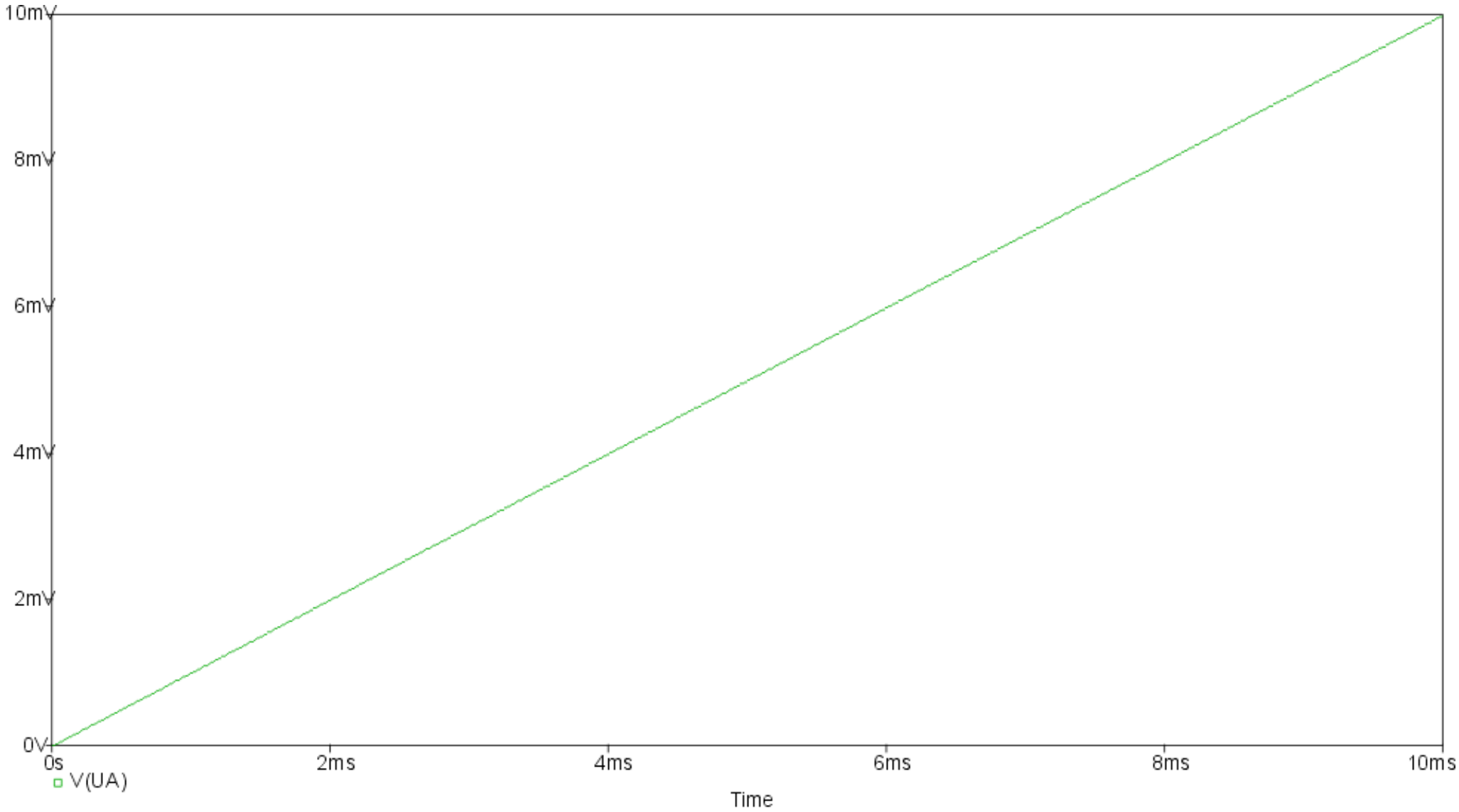
I - Element



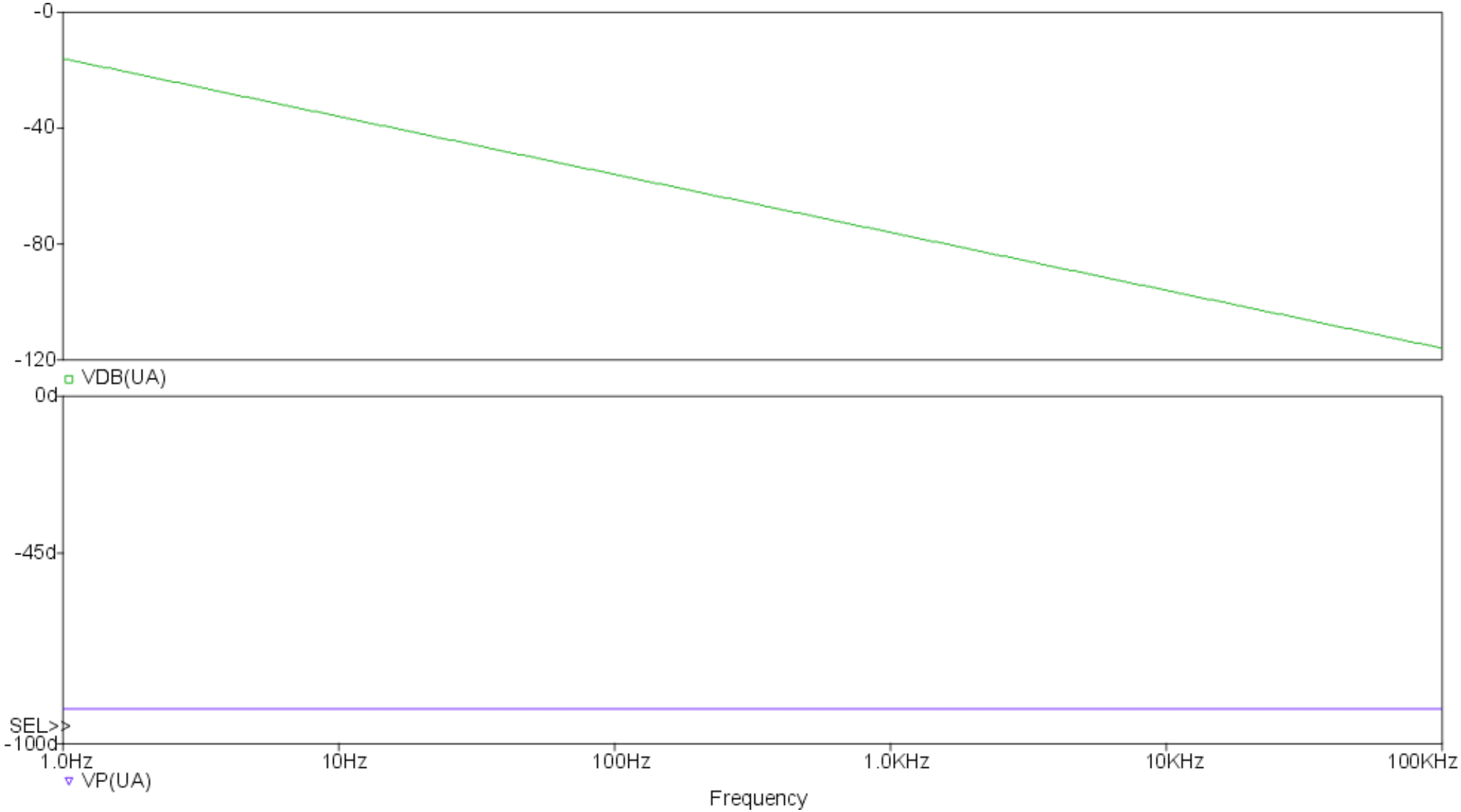
$$G(s) = \frac{k_I}{s}$$

Die Ausgangsgröße xa(t) ist proportional dem Zeitintegral der Eingangsgröße xe(t) bzw. die zeitliche Änderung der Ausgangsgröße ist proportional der Eingangsgröße.
I-Elemente sind Regelkreisglieder ohne Ausgleich, das heisst, sie streben keinem endlichen Endwert zu.

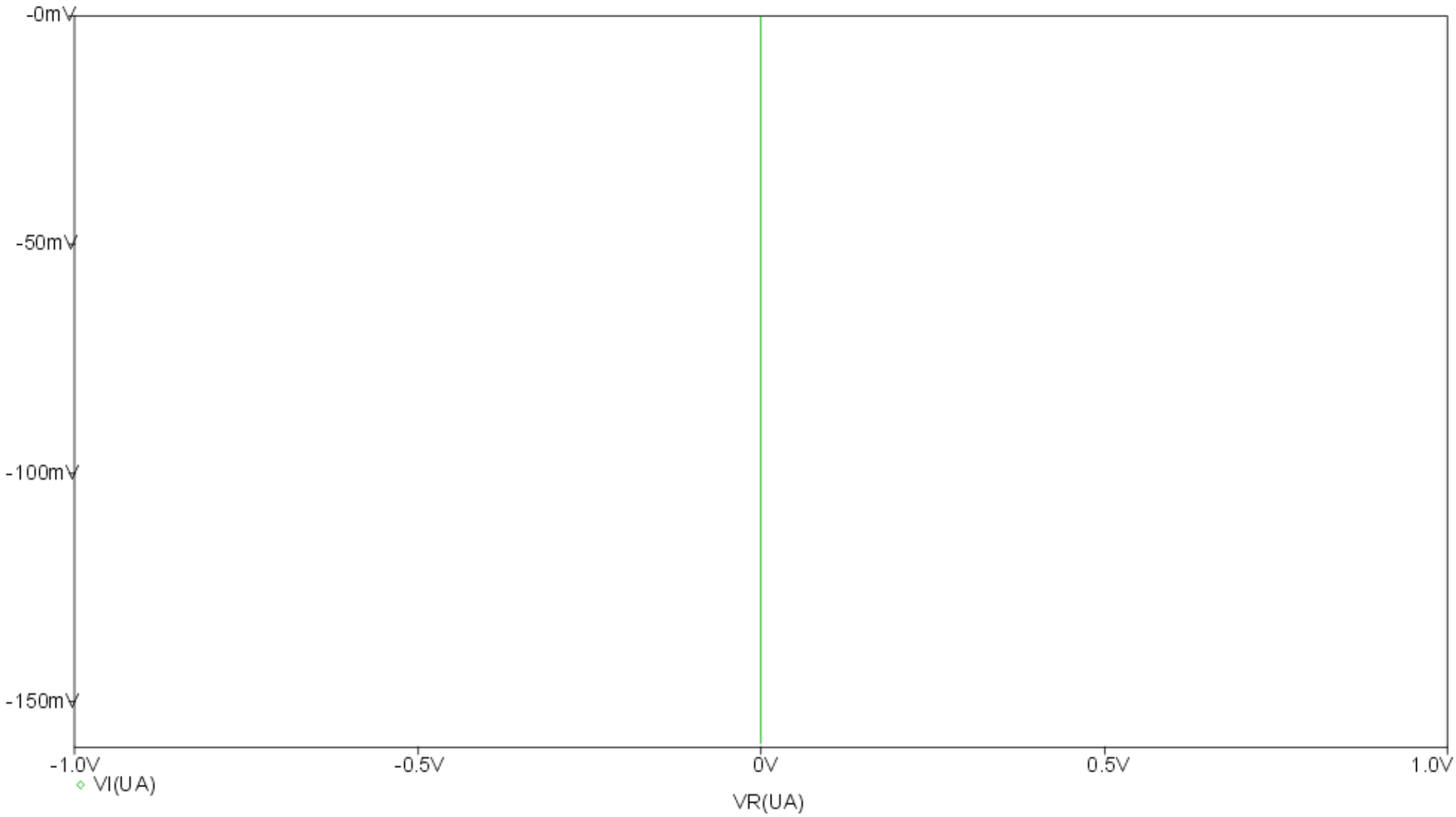
Sprungantwort



Bodediagramm



Ortskurve

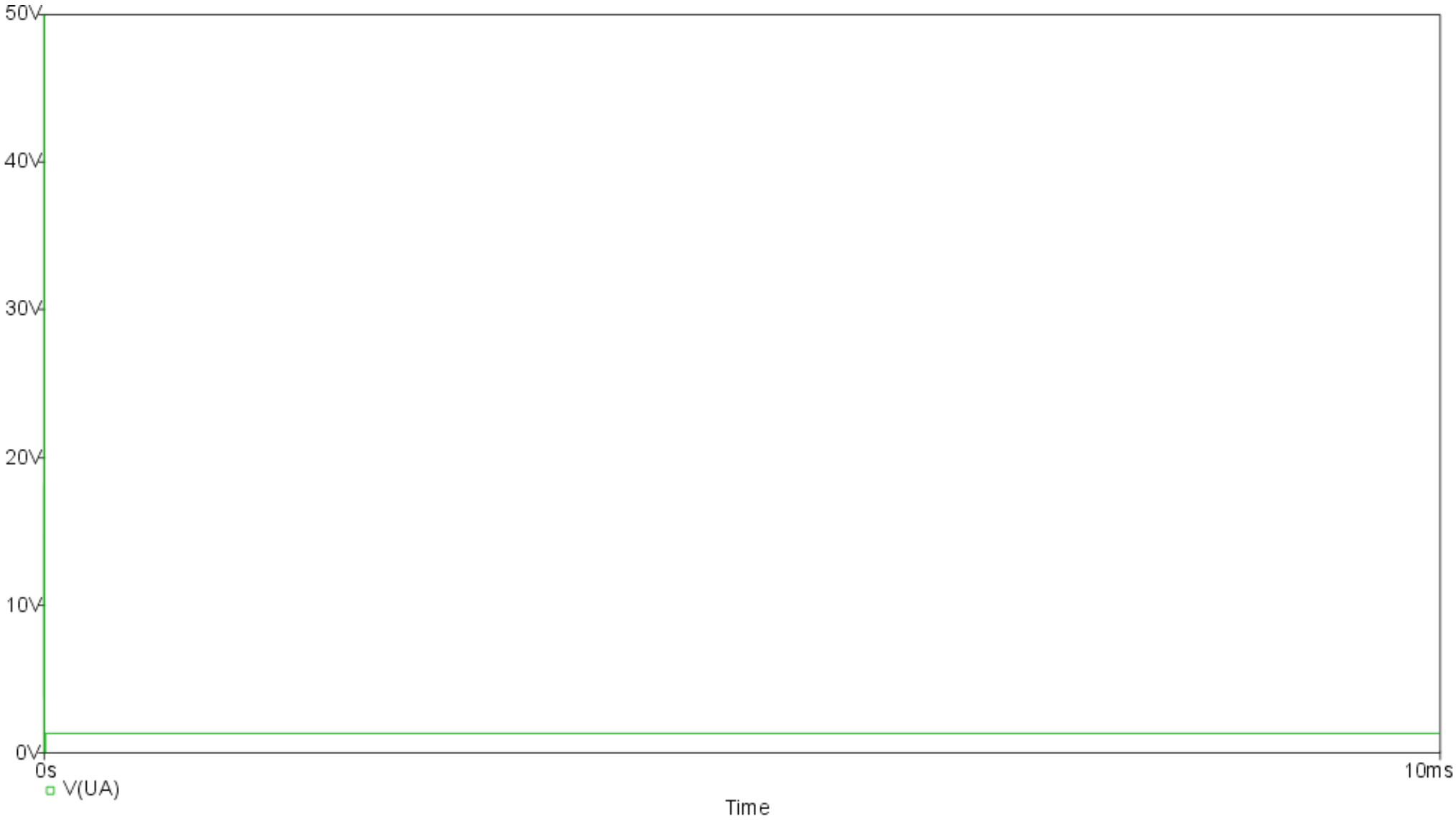


D - Element

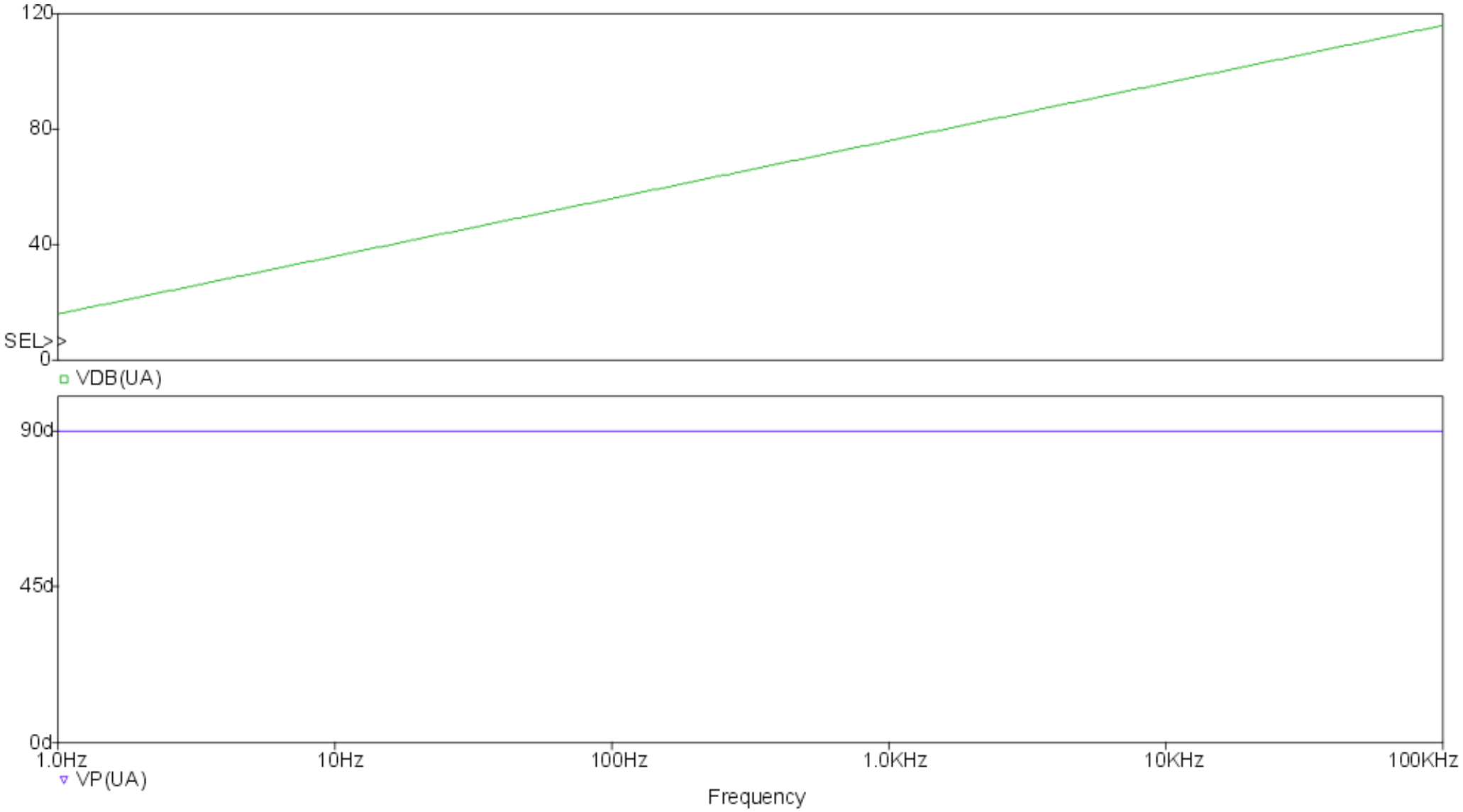
$$G(s) = k_D \cdot s$$

Reine D-Elemente sind in der Praxis, wie auch die Sprungantwort (ein Dirac-Impuls) zeigt, nicht exakt realisierbar. Physikalische Größen können keine unendlichen Werte annehmen. Zum Beispiel ist die abrupte Unterbrechung des Stromes in einem (mit einer Induktivität behafteten) Stromkreis nicht möglich - der die Unter-brechungsstelle überbrückende Lichtbogen verhindert einen unendlich hohen Spannungsanstieg. Einem auf einen Boden auffallenden Körper wiederfährt keine unendlich hohe Stoßkraft, da zufolge (plastischer oder elastischer) Deformation seine Beschleunigung endlich bleibt. D-Elemente stellen aber für die Berechnung bequeme und daher gebräuchliche Idealisierungen dar.

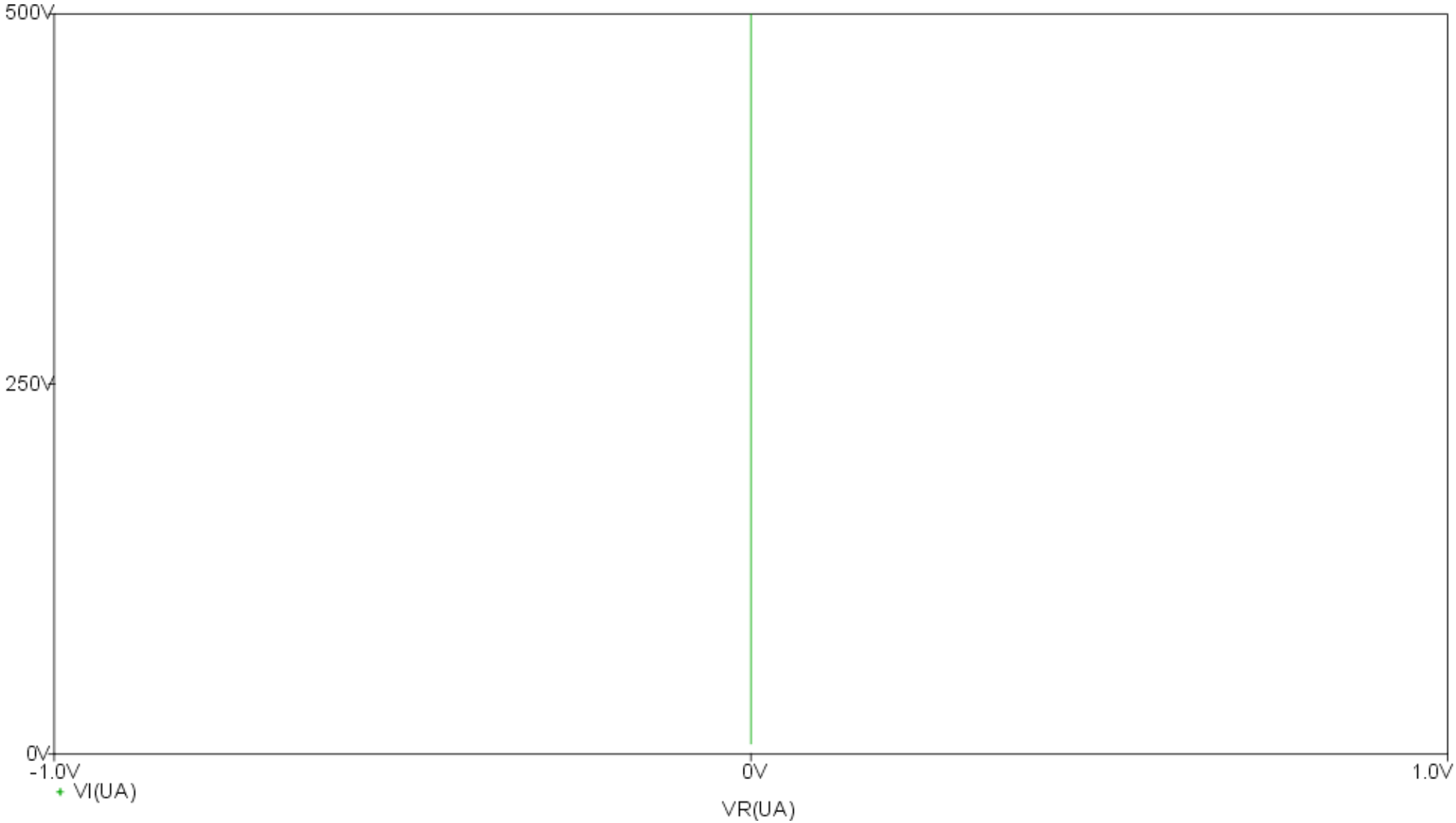
Sprungantwort



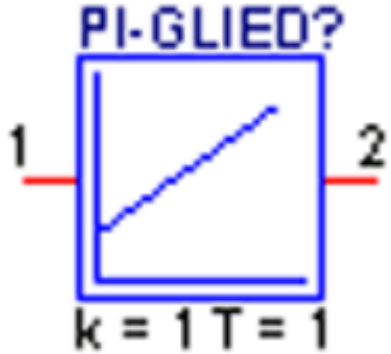
Bodediagramm



Ortskurve



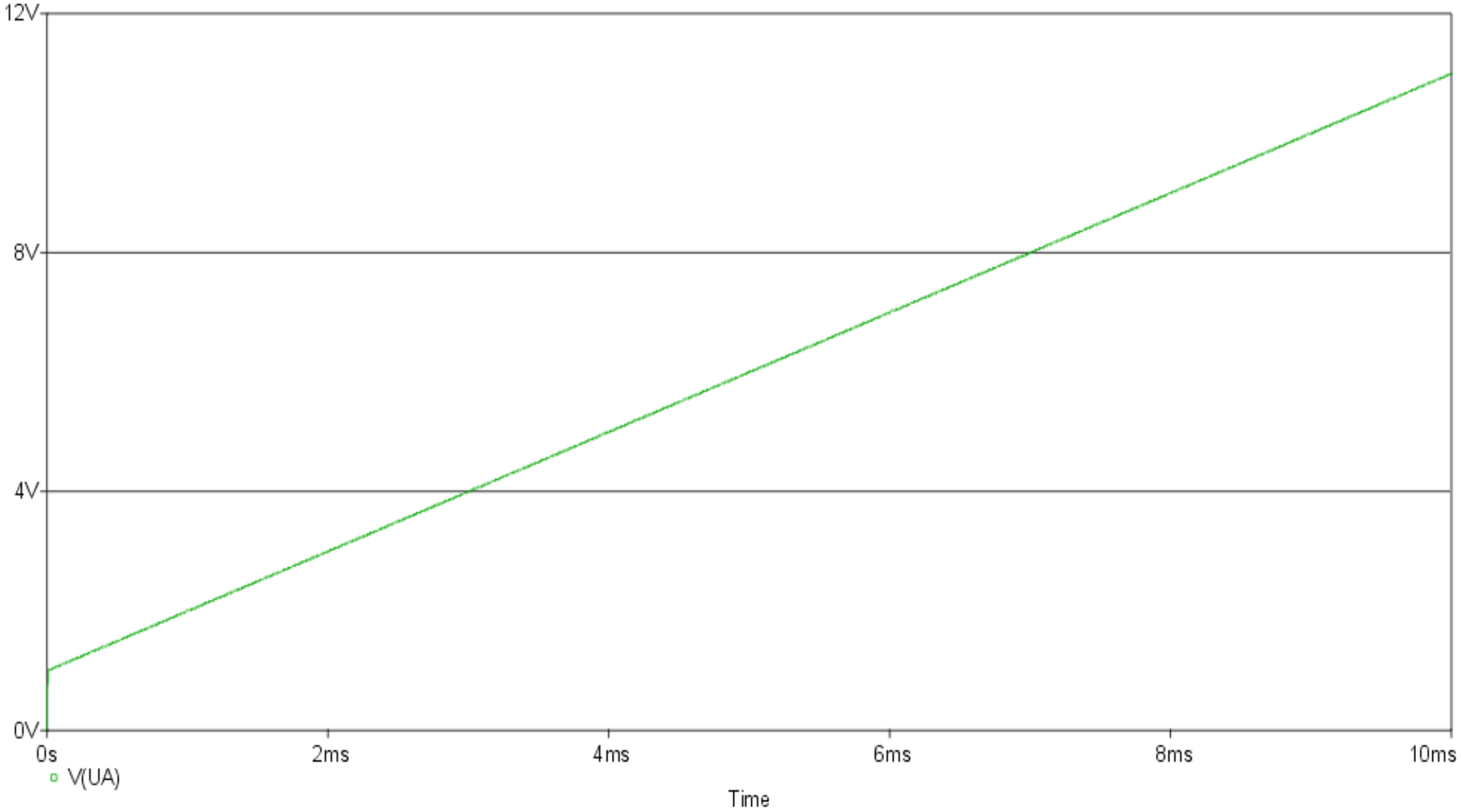
PI - Element



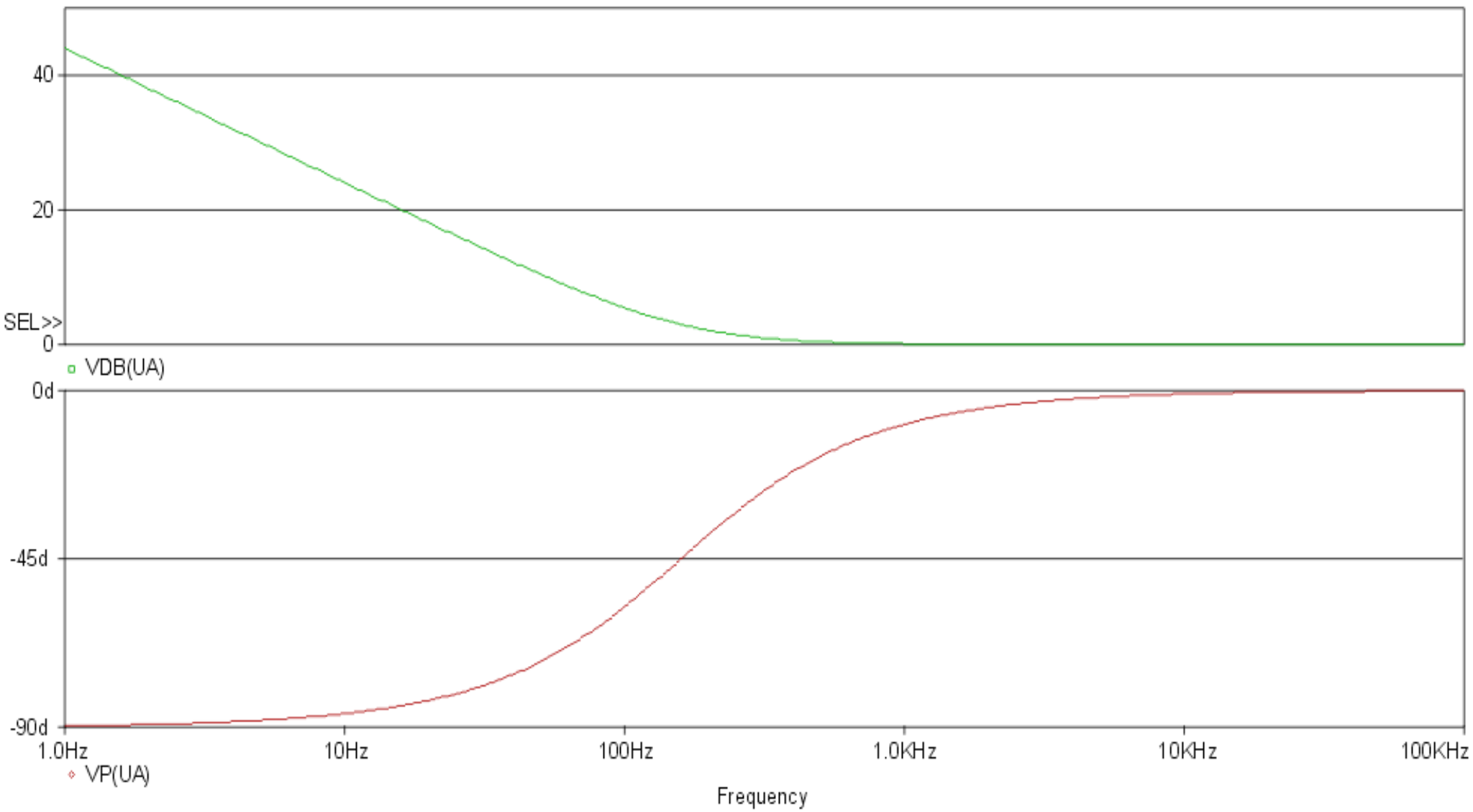
$$G(s) = k_R + \frac{1}{sT_0}$$

Ein PI-Element stellt ein I-Element mit dem Anfangswert k dar, welcher sich im Bodediagramm und in der Ortskurve bemerkbar macht. Durch den Anfangswert sinkt die Verstärkung nie unter einen bestimmten Wert.

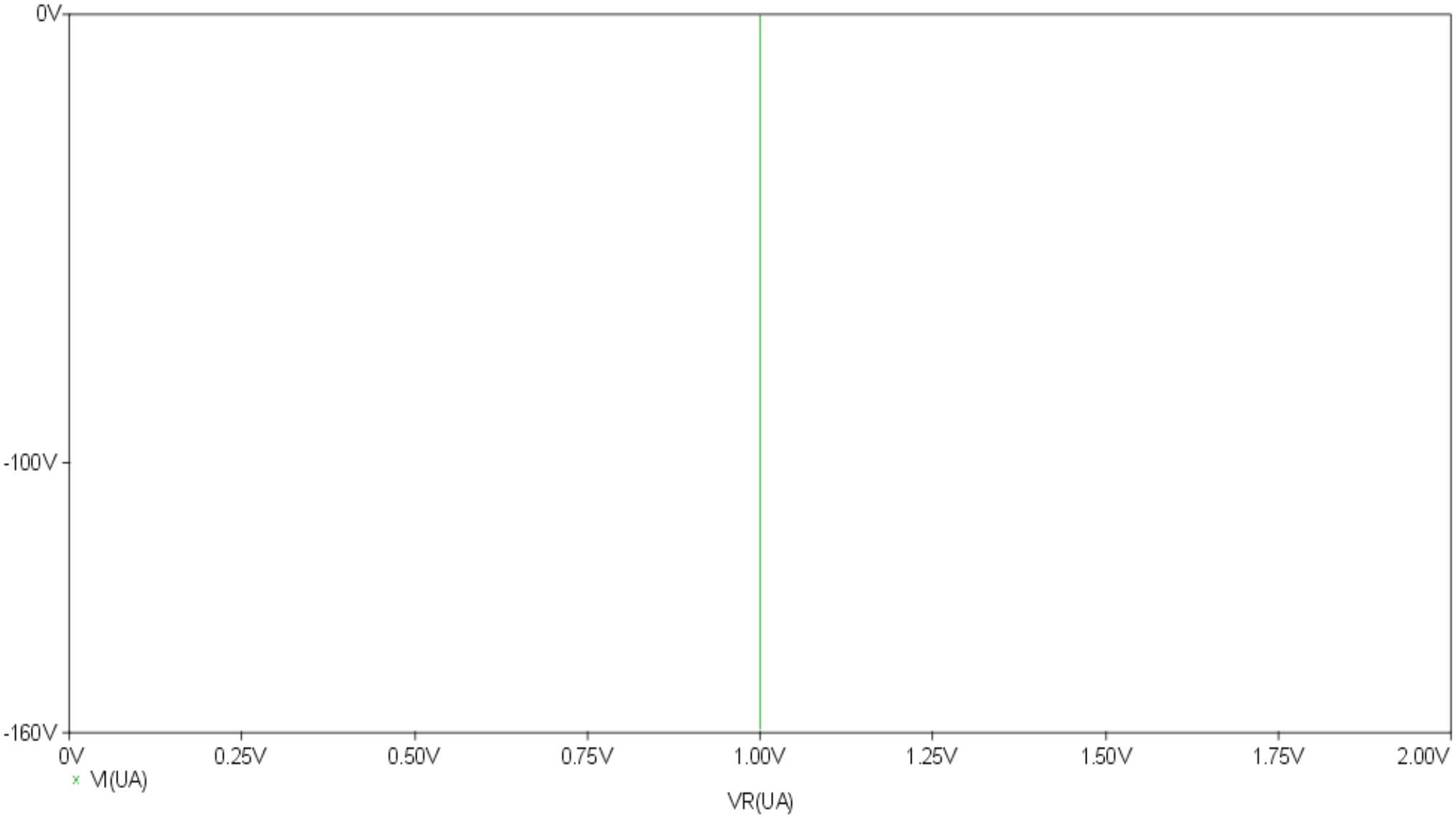
Sprungantwort



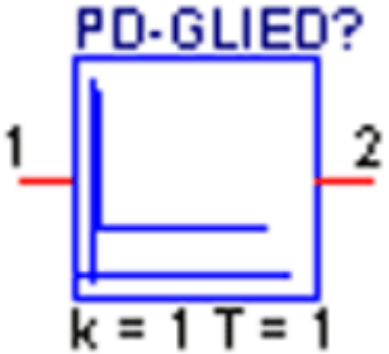
Bodediagramm



Ortskurve


0V
-100V
-160V
0V
0.25V
0.50V
0.75V
1.00V
1.25V
1.50V
1.75V
2.00V
VI(UA)
VR(UA)

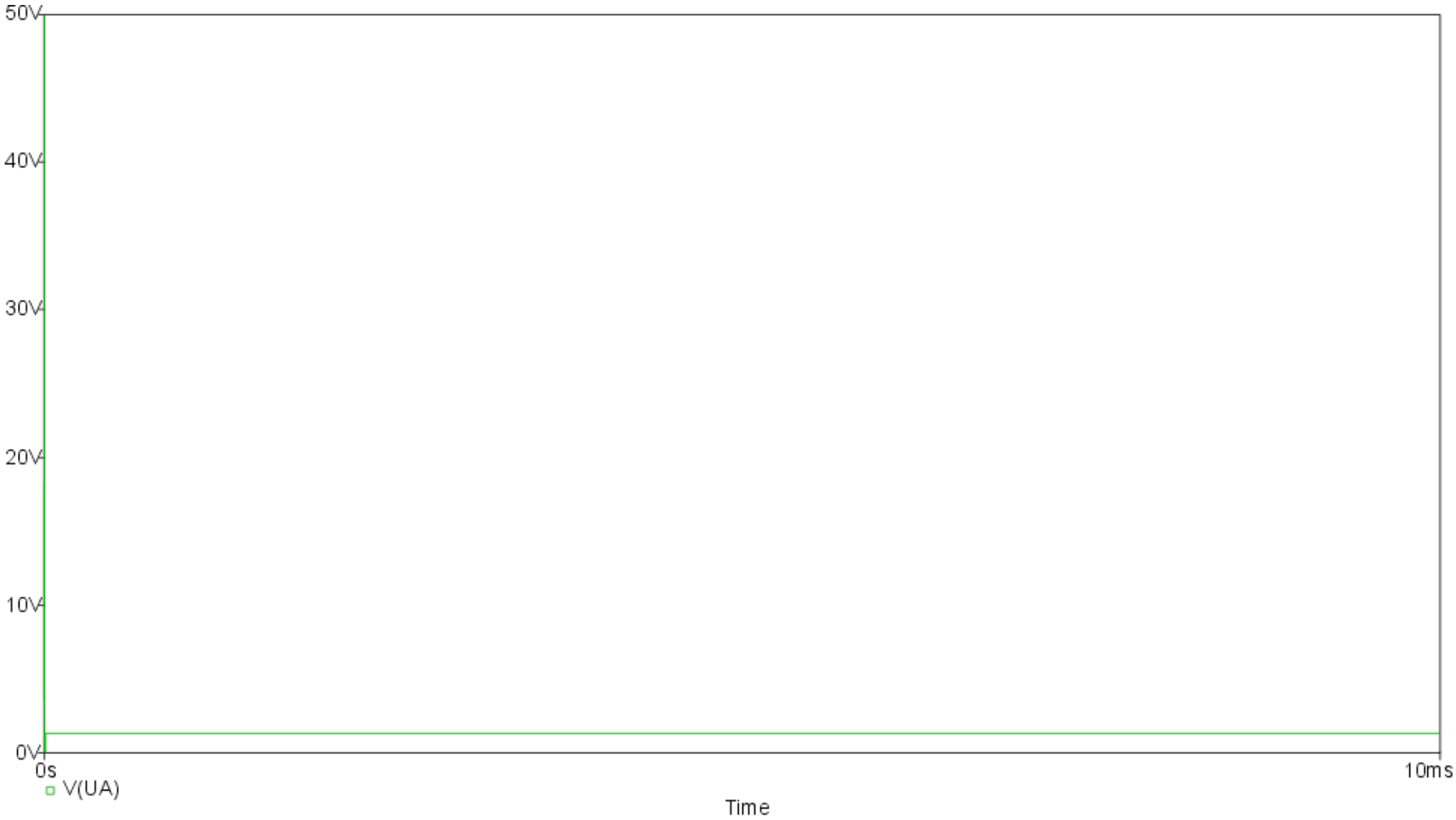
PD - Element



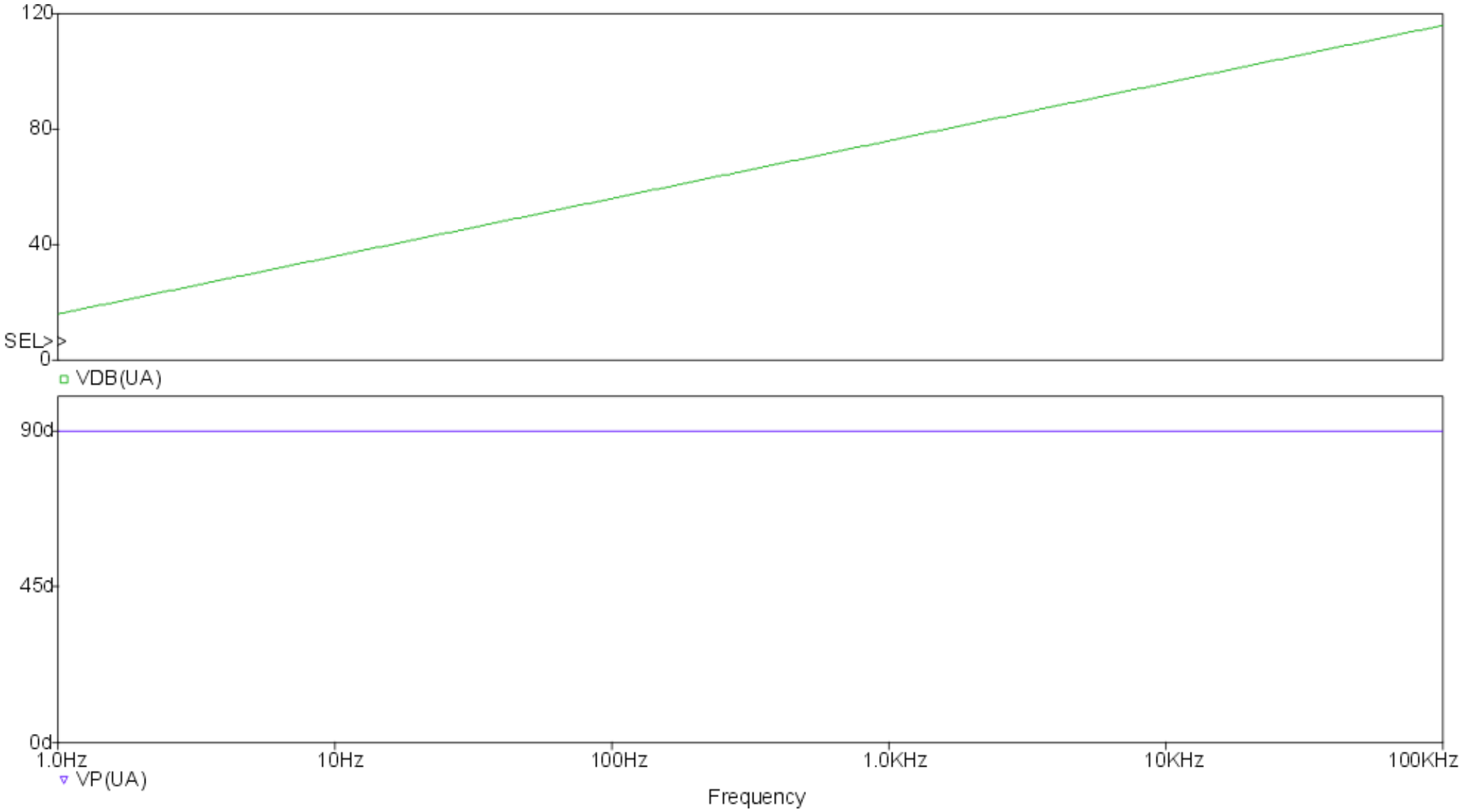
$$G(s) = k \cdot (1 + sT)$$

Ein PD-Element stellt grundsätzlich ein D-Element mit einer Stationärverstärkung dar, das heisst, dass sich nicht Ändernde Eingangssignale auch ein Ausgangssignal erzeugen können.

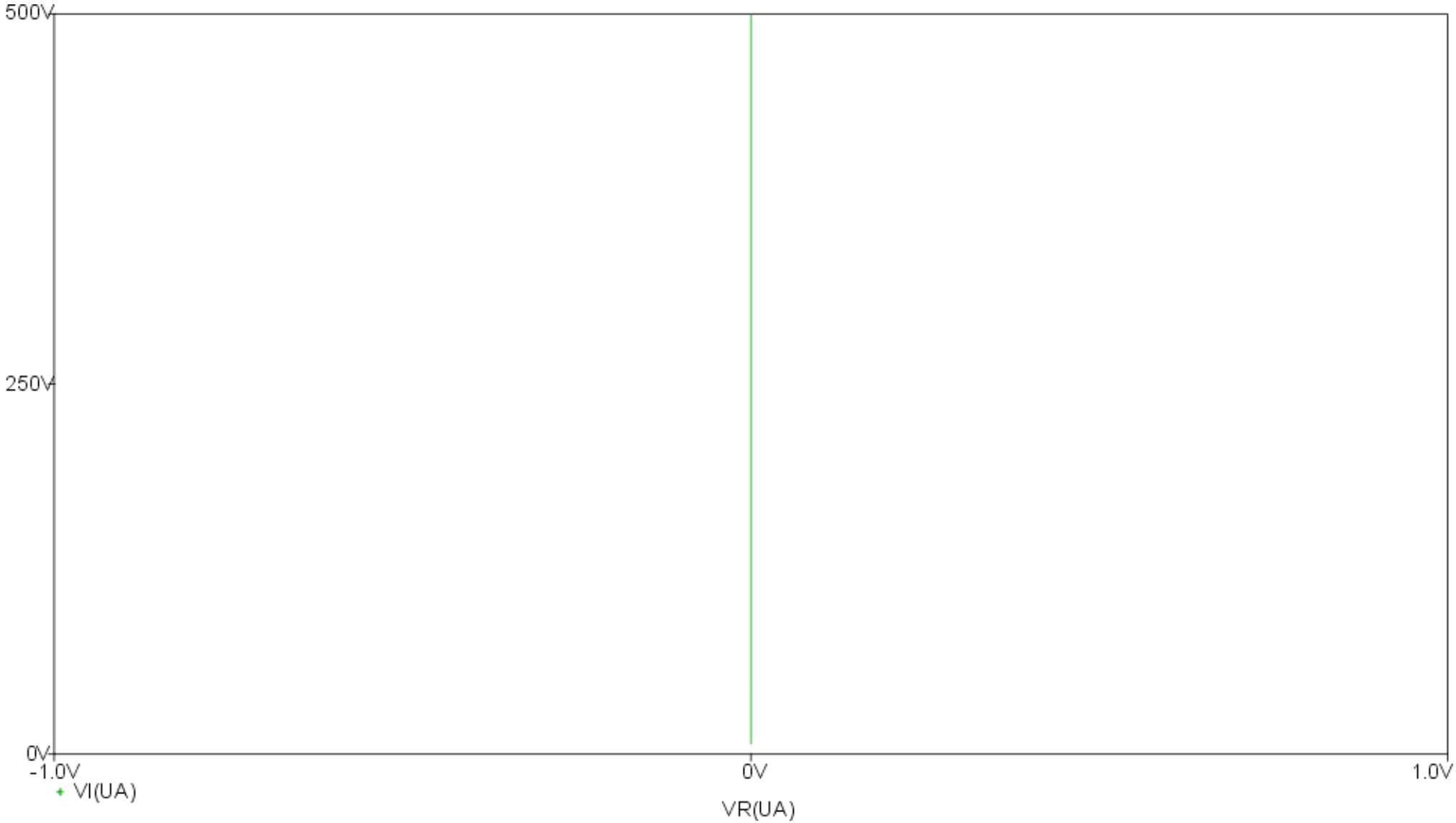
Sprungantwort



Bodediagramm



Ortskurve

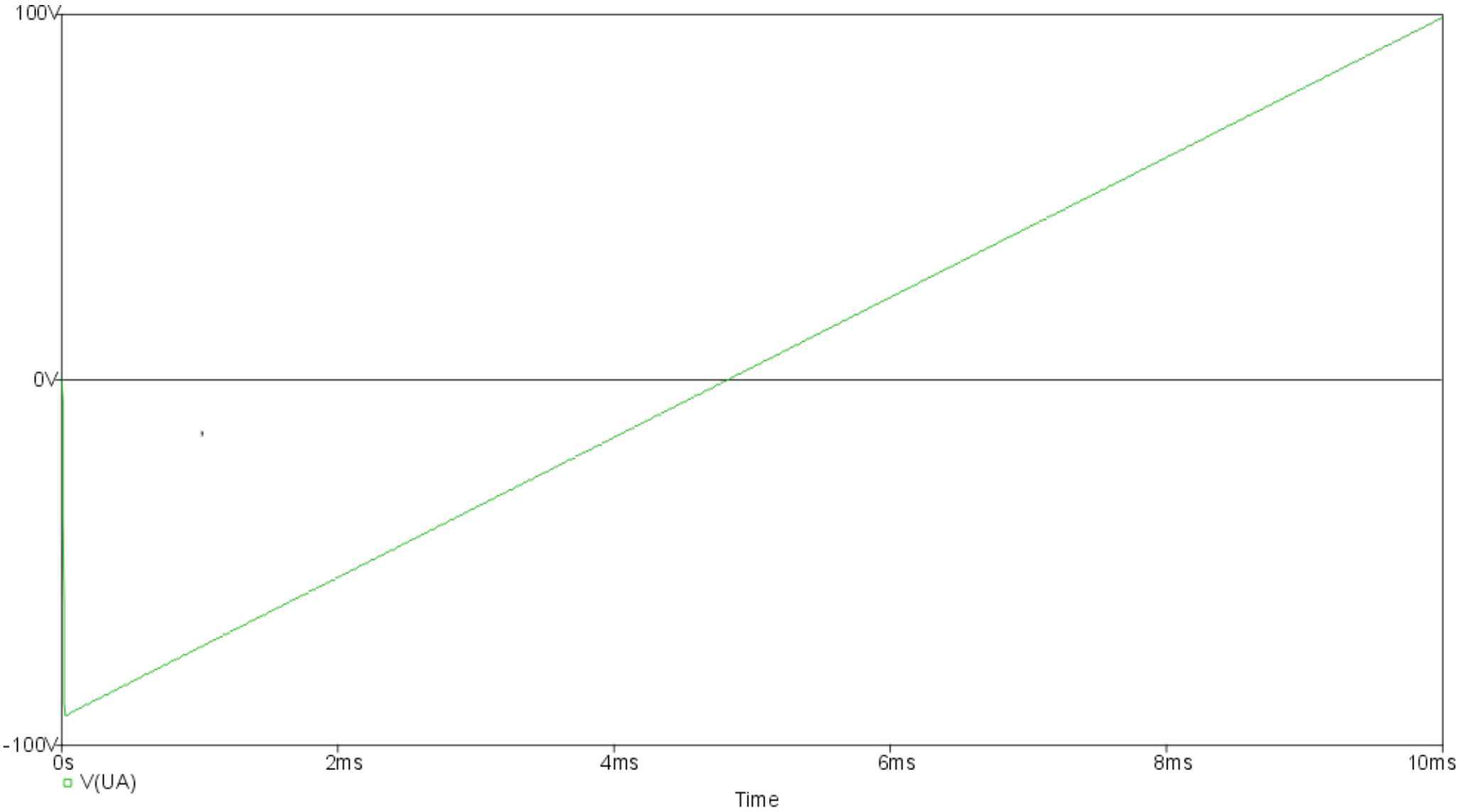


PID - Element

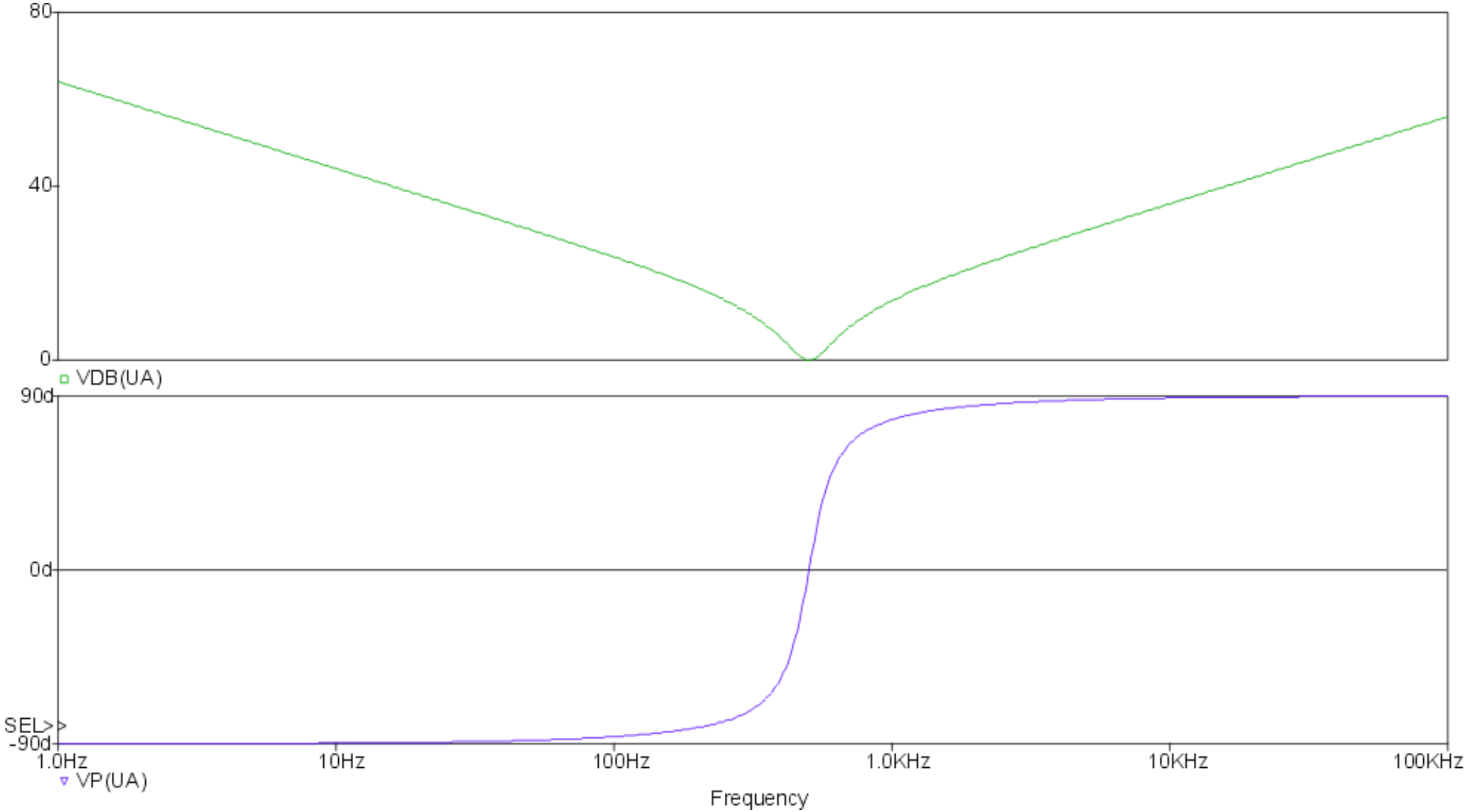
$$G(s) = k_R \cdot \left(1 + \frac{1}{sT_N} + sT_V \right)$$

Ein PID-Element besitzt 3 wichtige Eigenschaften: Es Integriert sich langsam ändernde Eingangssignale, differenziert sich schnell ändernde Eigangssignale und verstärkt alle Eingangssignale. Es kann auch als aktiver Bandfilter angesehen werden.

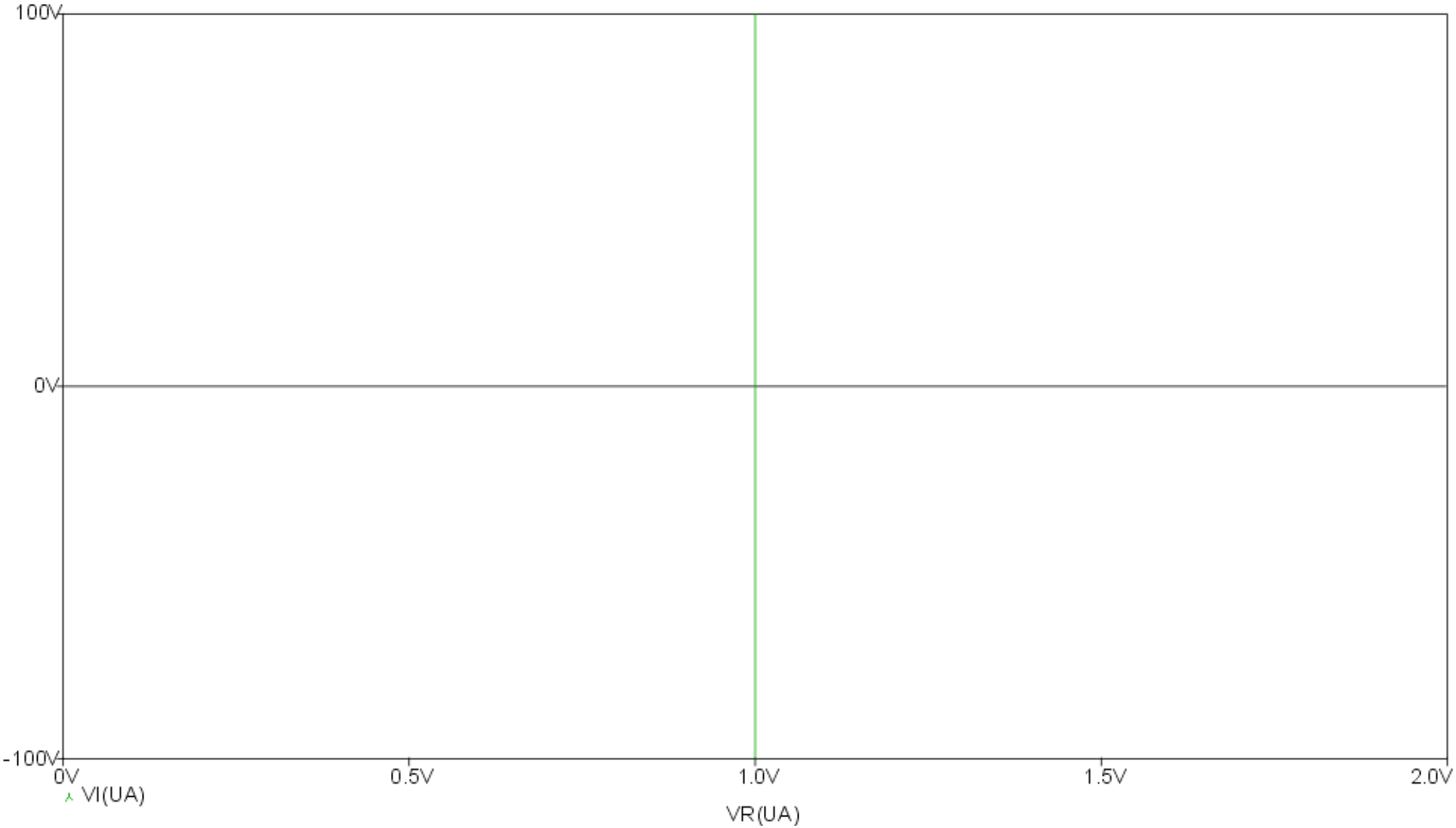
Sprungantwort



Bodediagramm



Ortskurve

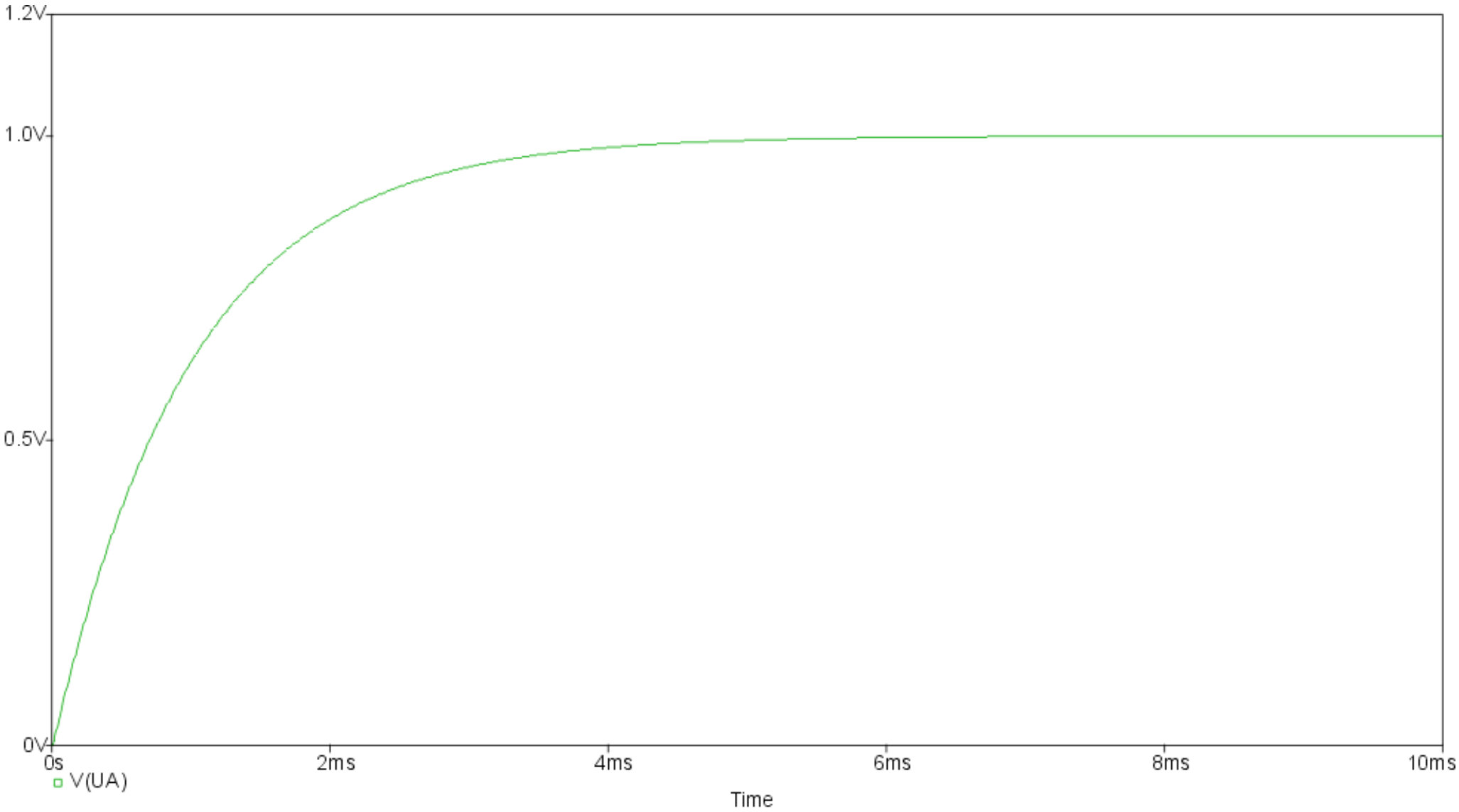


PT1 - Element

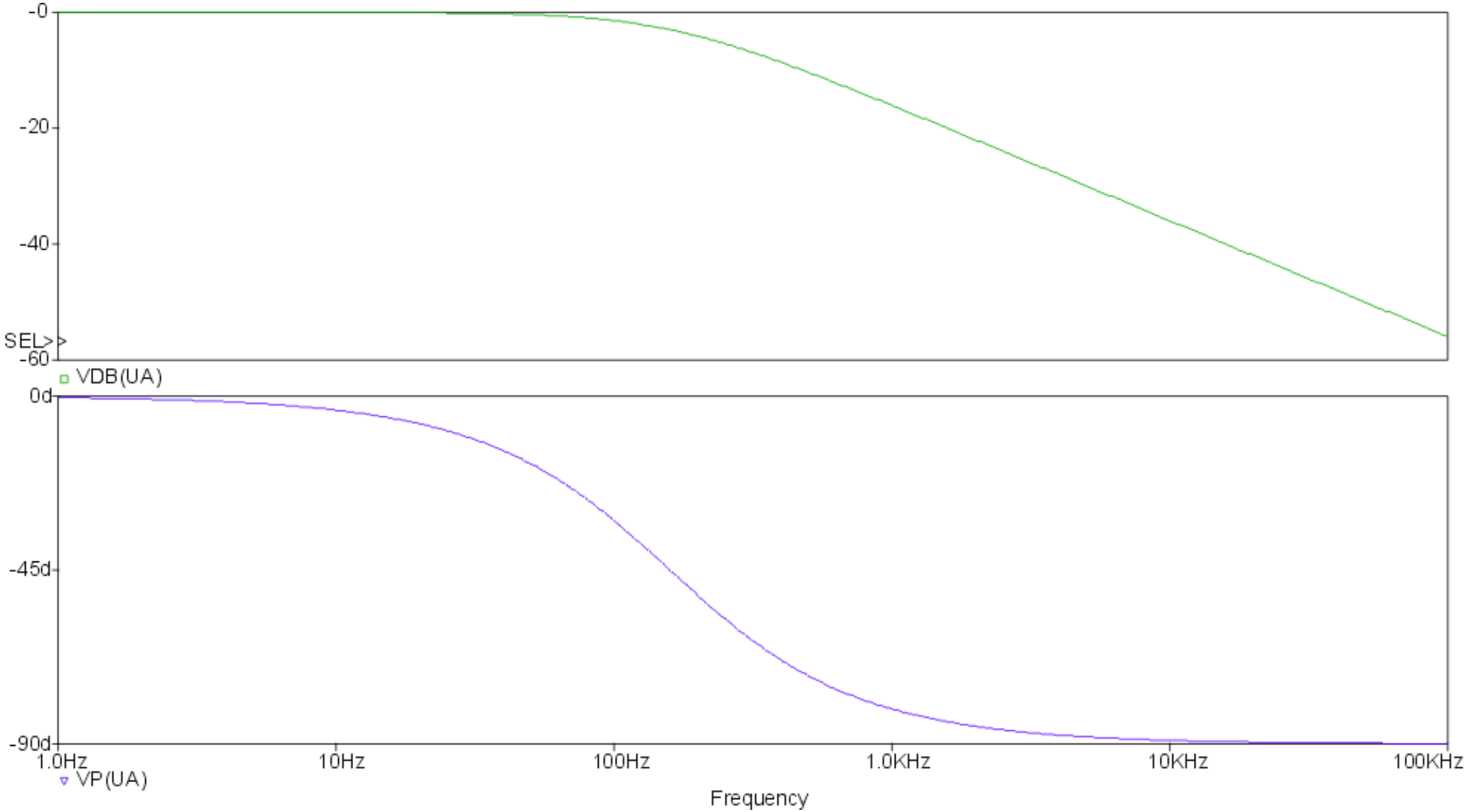
$$G(s) = \frac{k}{1 + sT}$$

Die meisten Regelkreisglieder reagieren mehr oder minder verzögert auf ein sprungförmiges Eingangssignal, was auf das Vorhandensein ein oder mehrerer Energiespeicher hinweist.

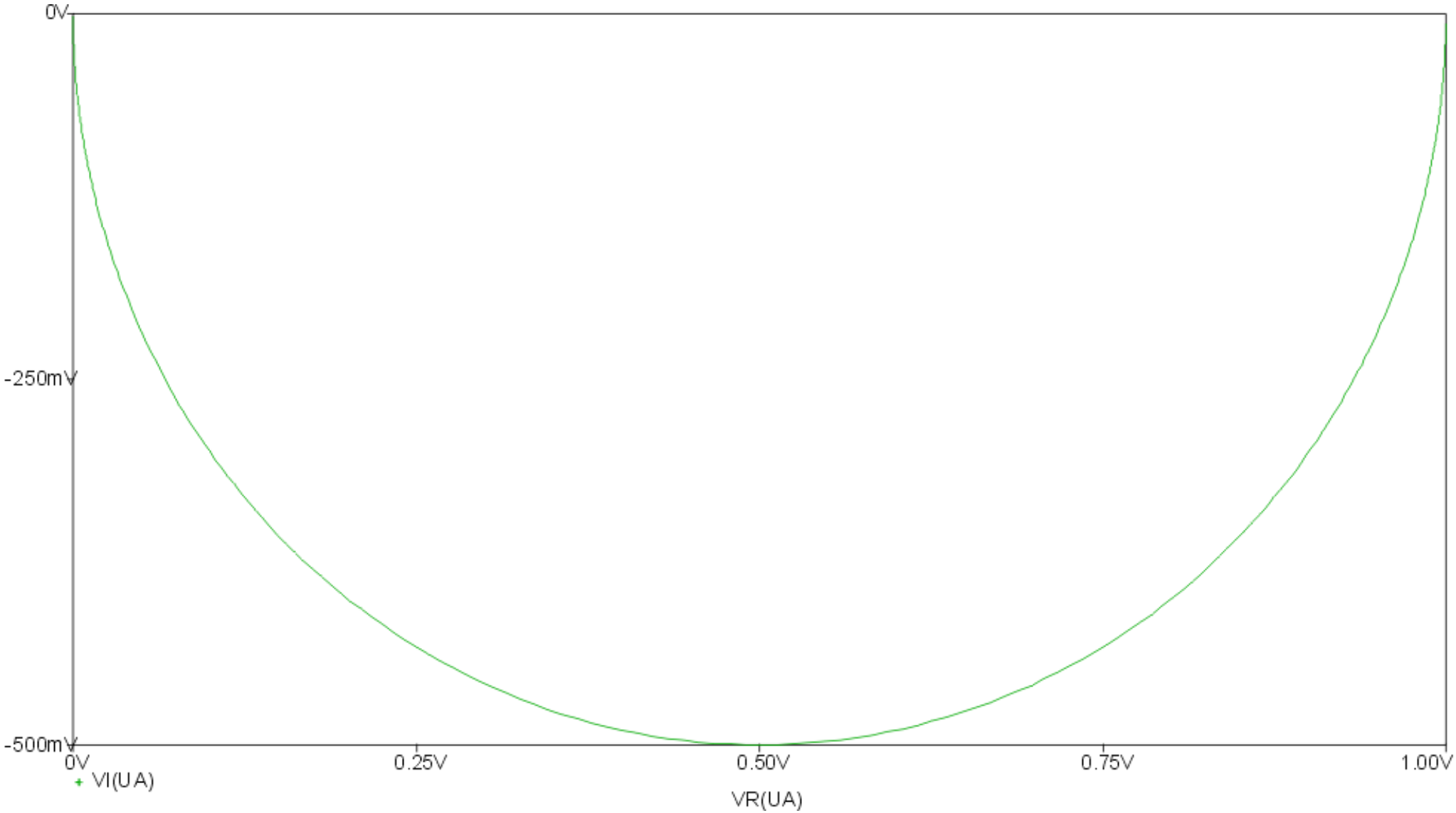
Sprungantwort



Bodediagramm



Ortskurve

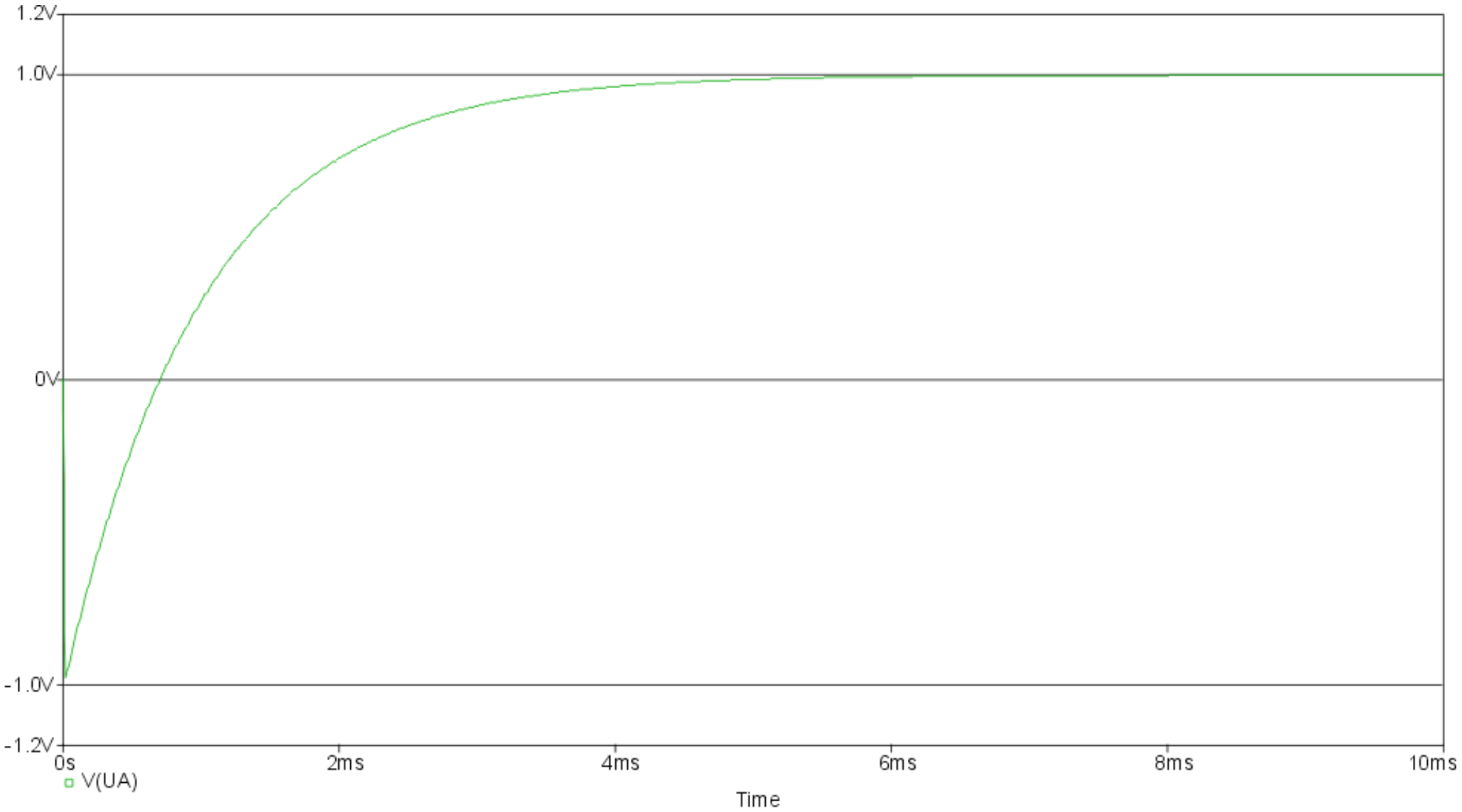


IT1 - Element

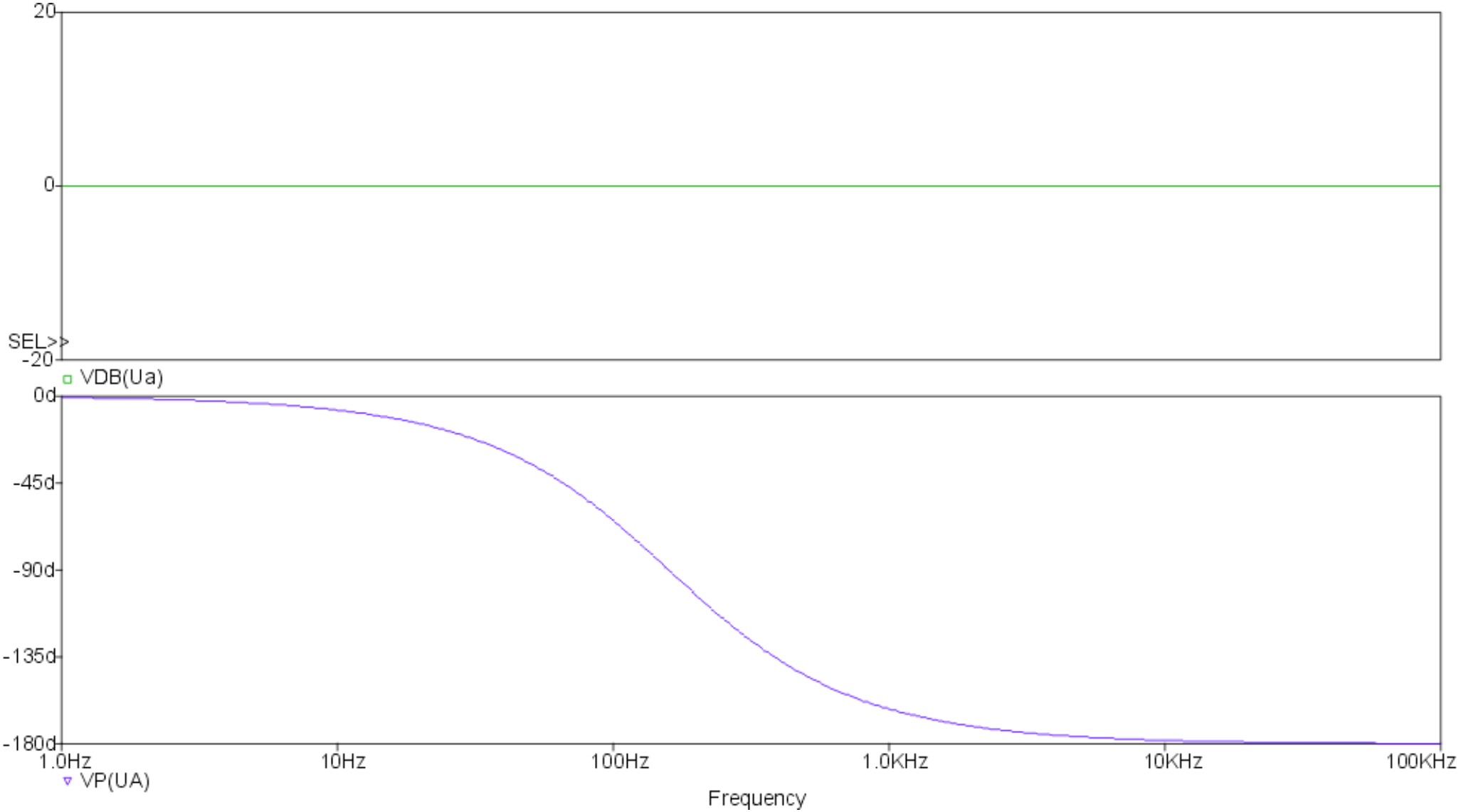
$$G(s) = \frac{1 - sT}{1 + sT}$$

Ein IT1-Element stellt eine I-Element mit einer Zeitverzögerung dar, welche bei Änderung der Eingangsgröße in Kraft tritt. Dadurch wird die spontane Änderung der Ausgangsgröße geschwächt.

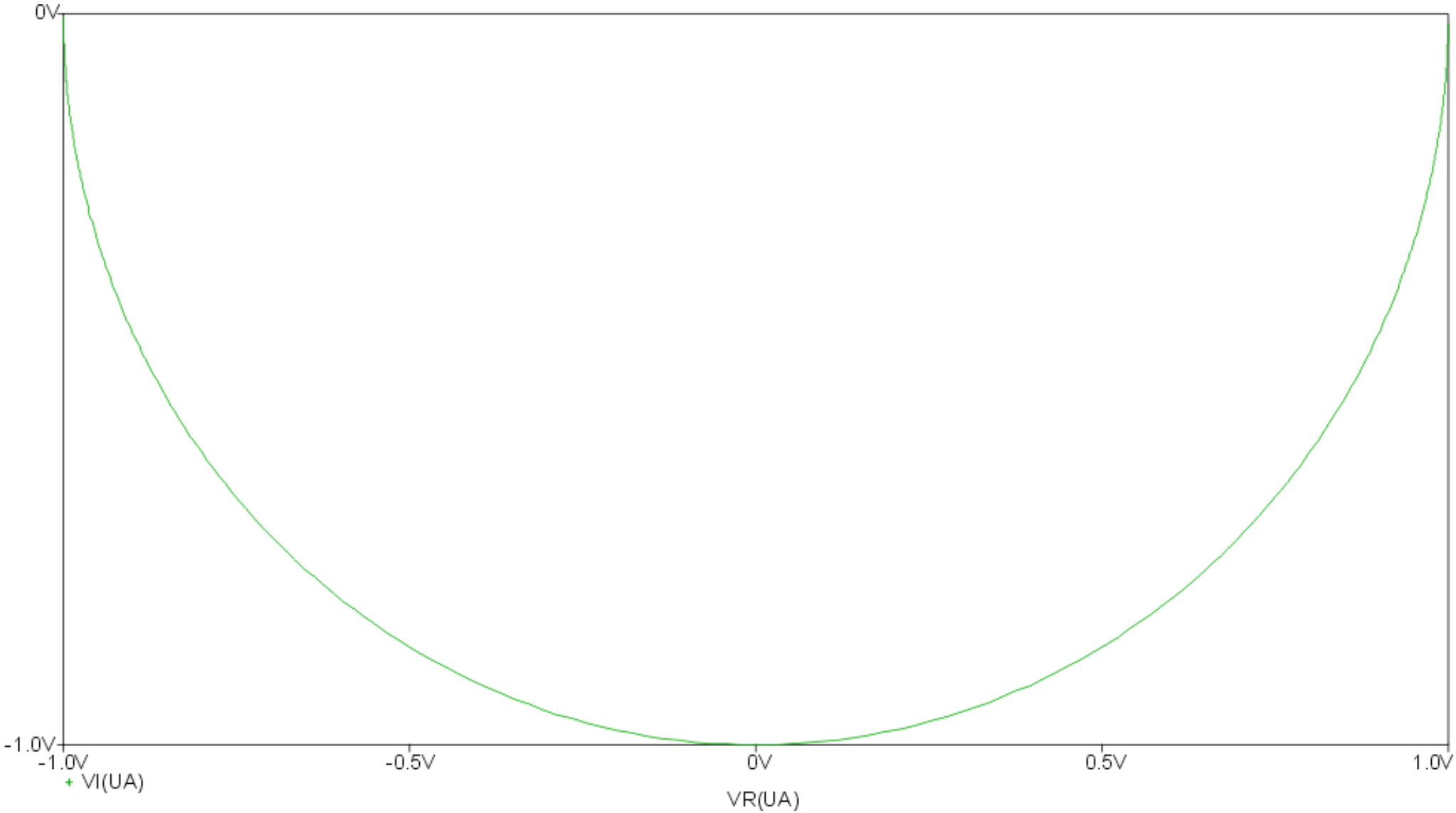
Sprungantwort



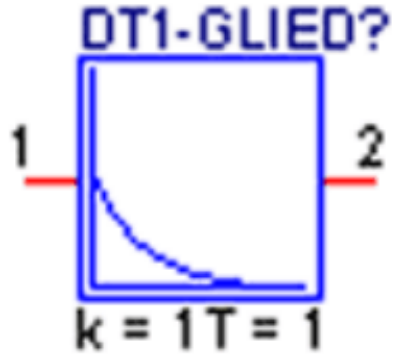
Bodediagramm



Ortskurve



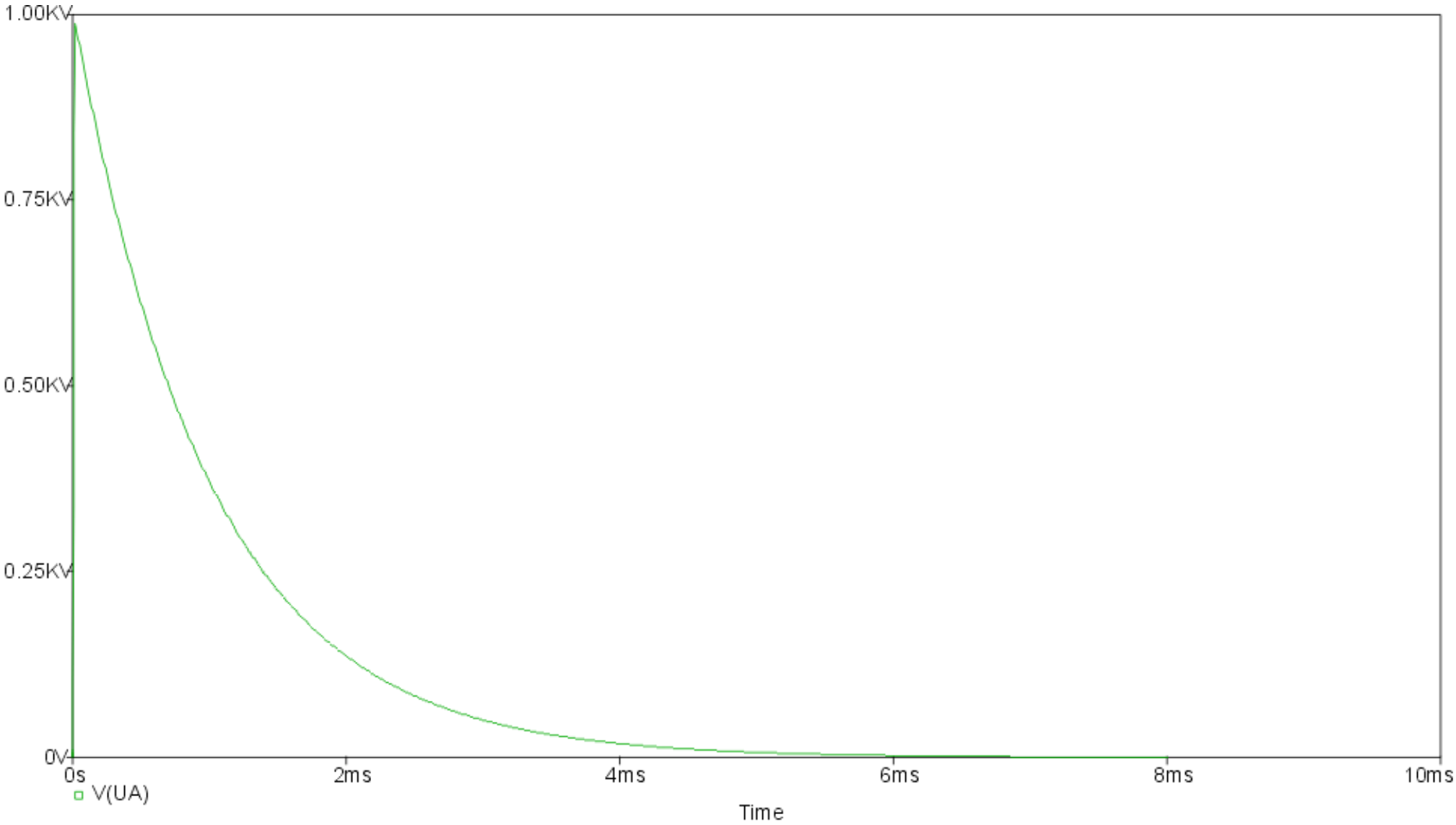
DT1 - Element



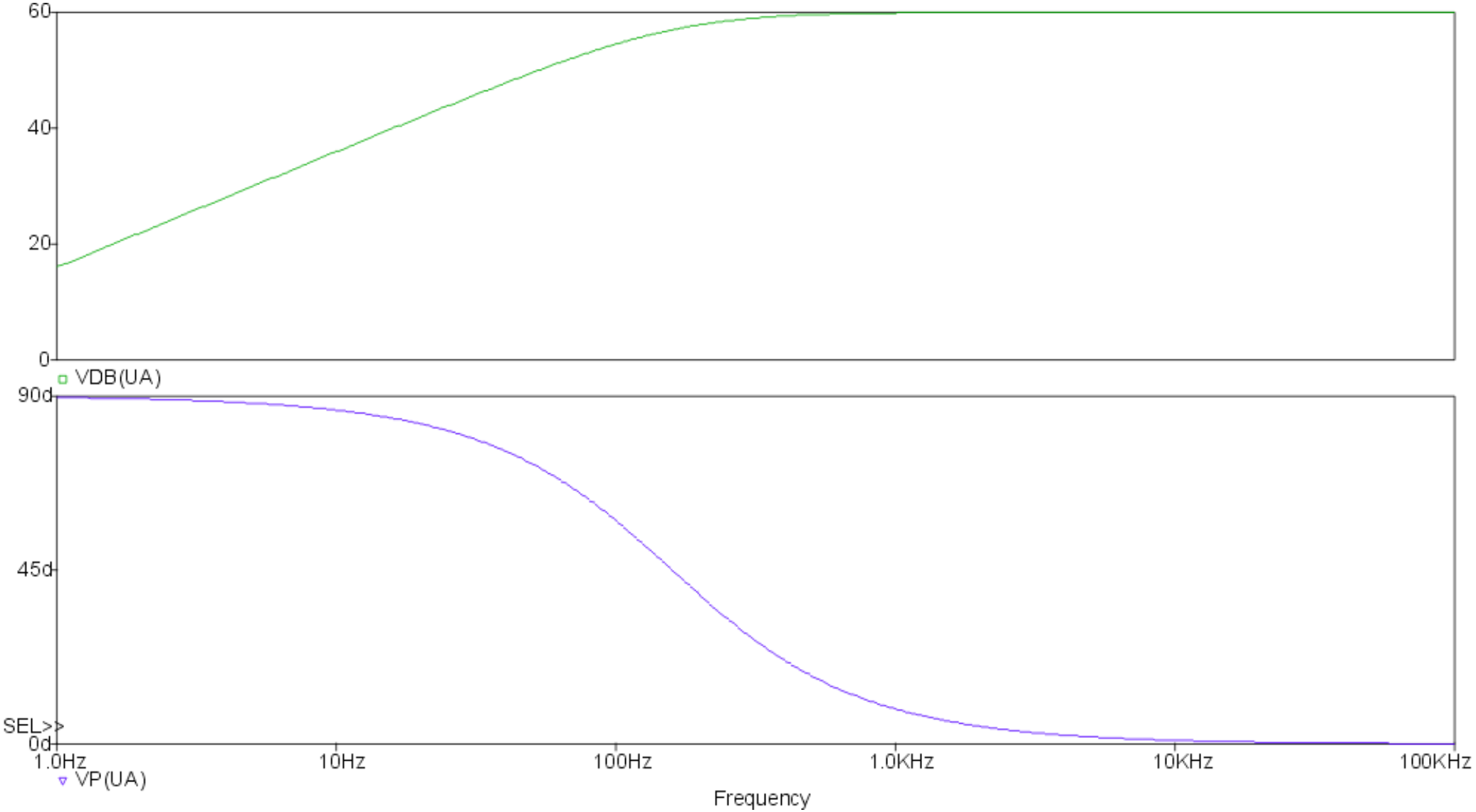
Ein DT1-Element stellt einen Differenzierer mit einer zusätzlichen Zeitverzögerung dar, die das differenzierende Verhalten "ab-schwächt". Die Sprungantwort ist kein Dirac-Impuls, sondern ein nadelförmiger Impuls mit endlicher Höhe. Reale Differenzierer sind meist DT1-Elemente.

$$G(s) = \frac{ks}{1 + sT}$$

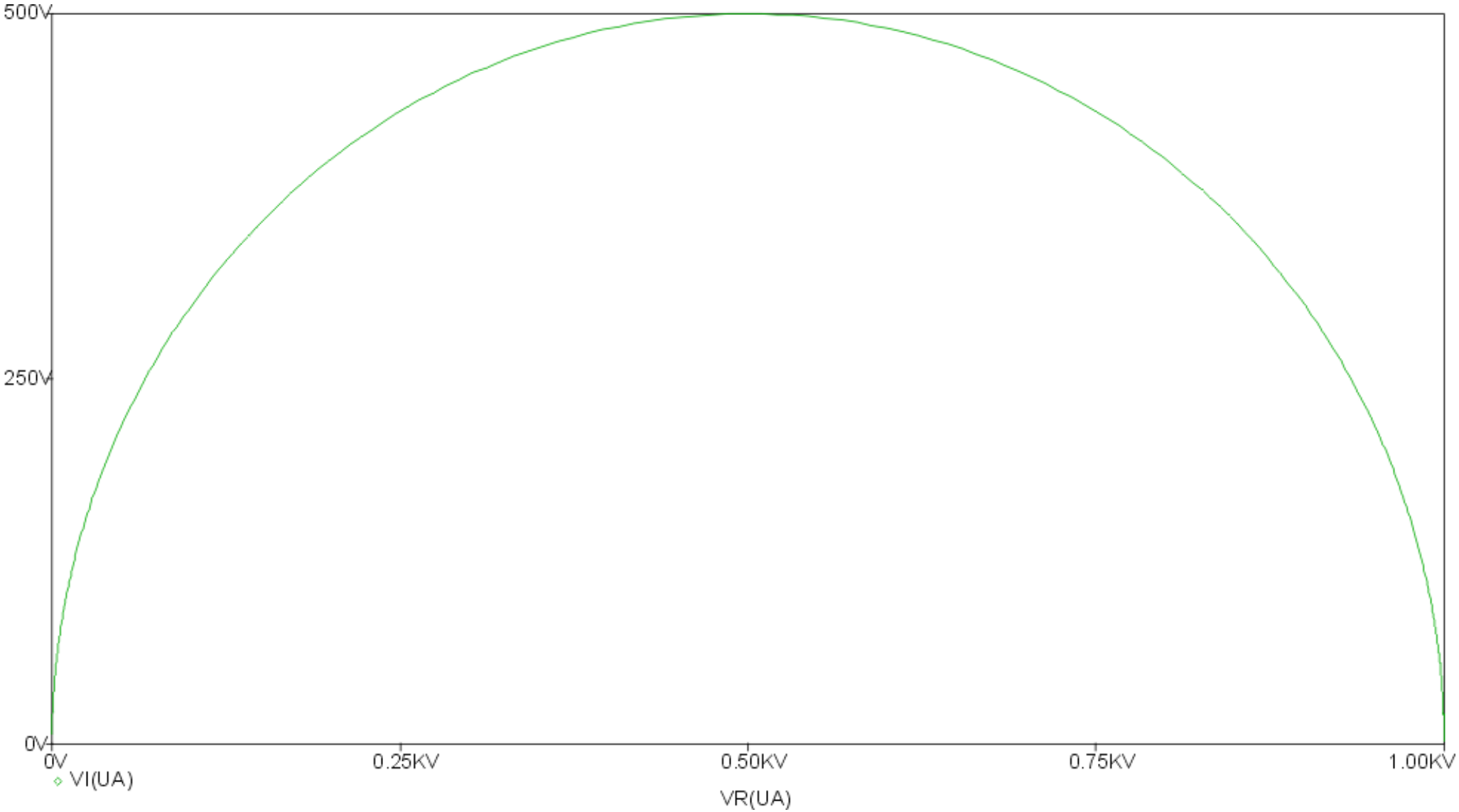
Sprungantwort



Bodediagramm



Ortskurve

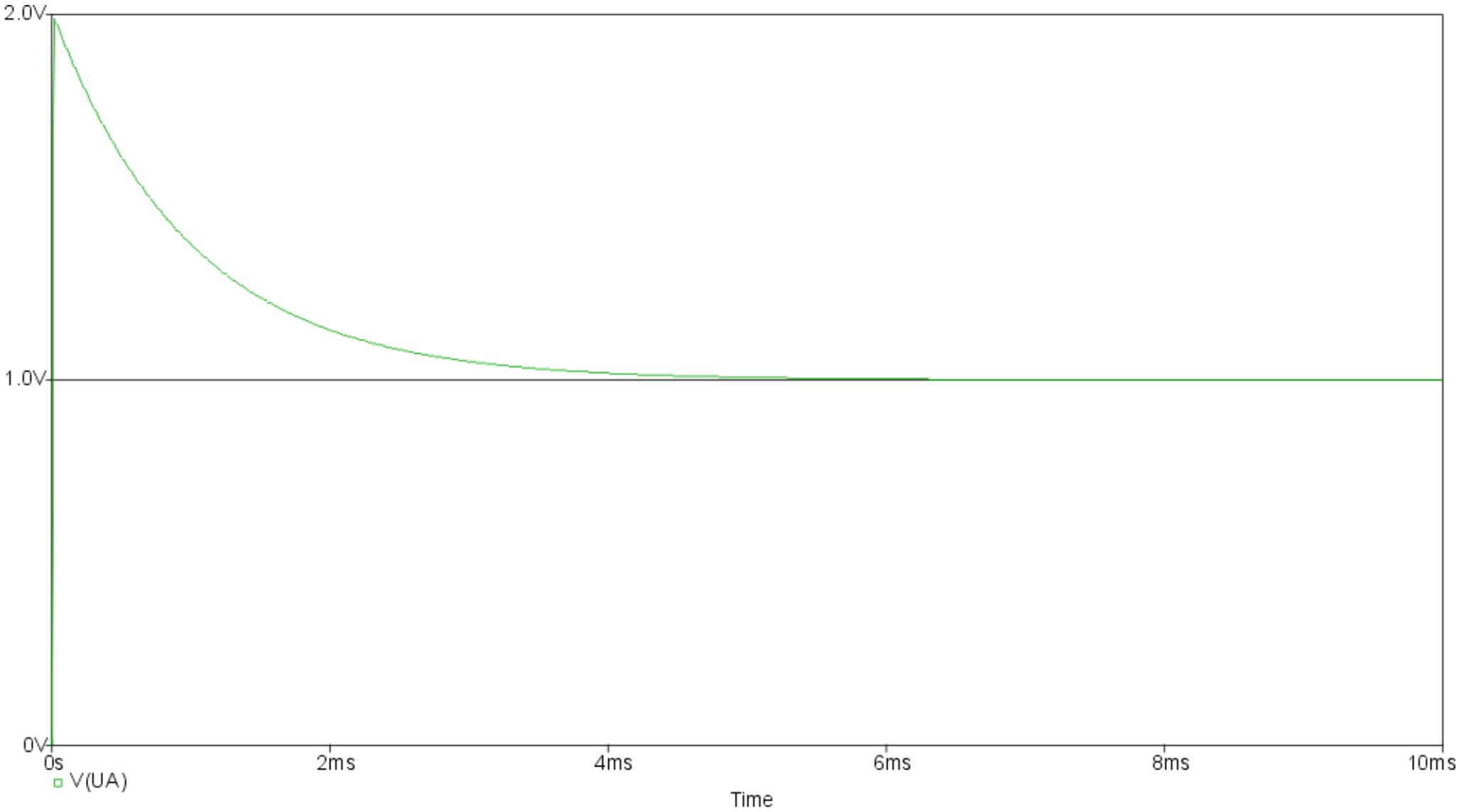


PDT1 - Element

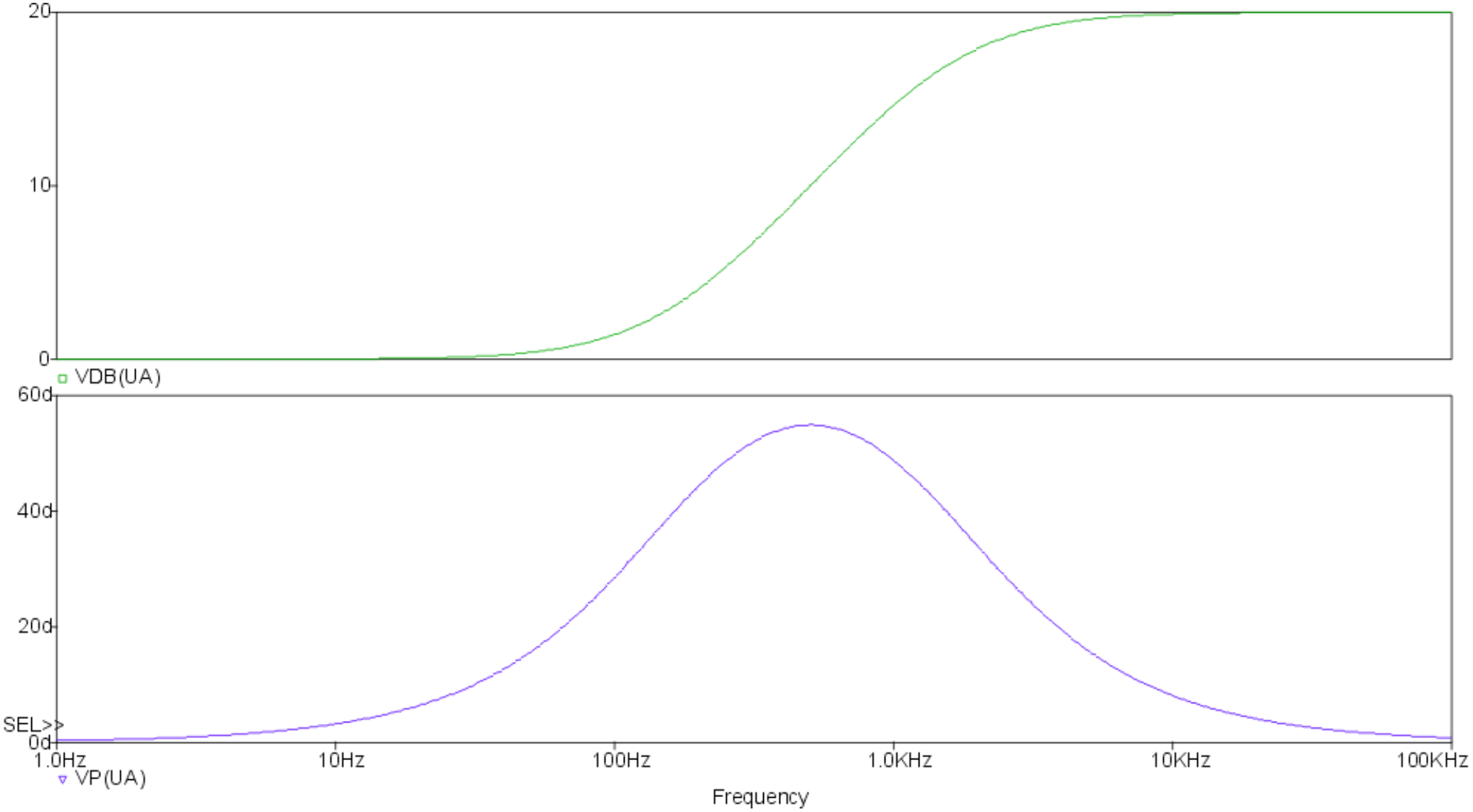
$$G(s) = k \cdot \frac{1 + sT_1}{1 + sT_2}$$

Ein PDT1-Element entsteht durch eine Serienschaltung eines PT1-Elementes und eines PD-Elementes. Je nach dem Verhältnis der beiden Zeitkonstanten T1 und T2 besitzt das PDT1-Element entweder vorwiegend differenz-ierendes oder verzögerndes Verhalten. Dementsprechend erhält man im Bereich der Knick-frequenzen eine Phasenanhebung oder Phasenabsenkung. Phasen-anhebende PDT1-Elemente finden Verwendung zur Verbesserung des Stabilitätsverhaltens von Regel-kreisen.

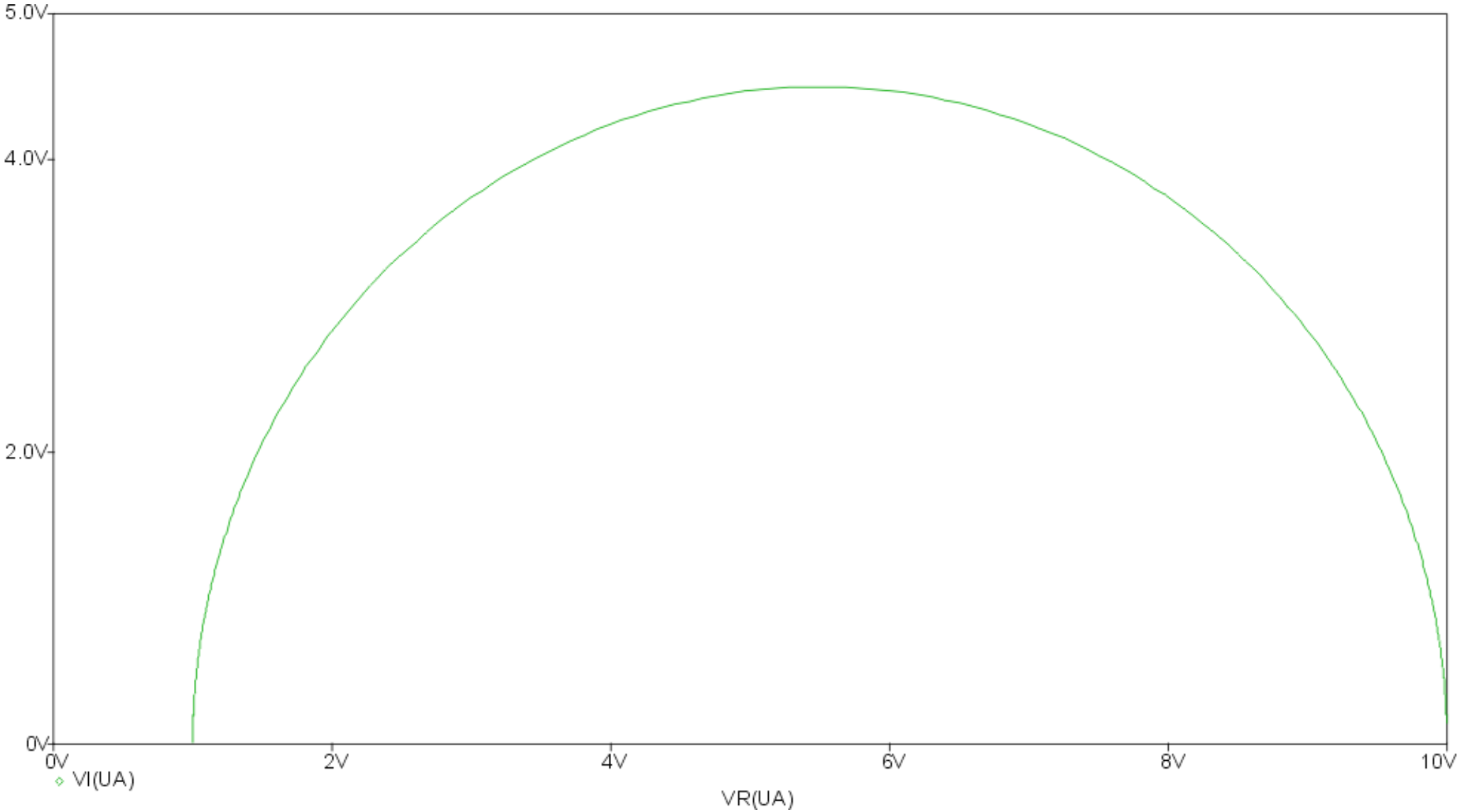
Sprungantwort (T1 > T2)



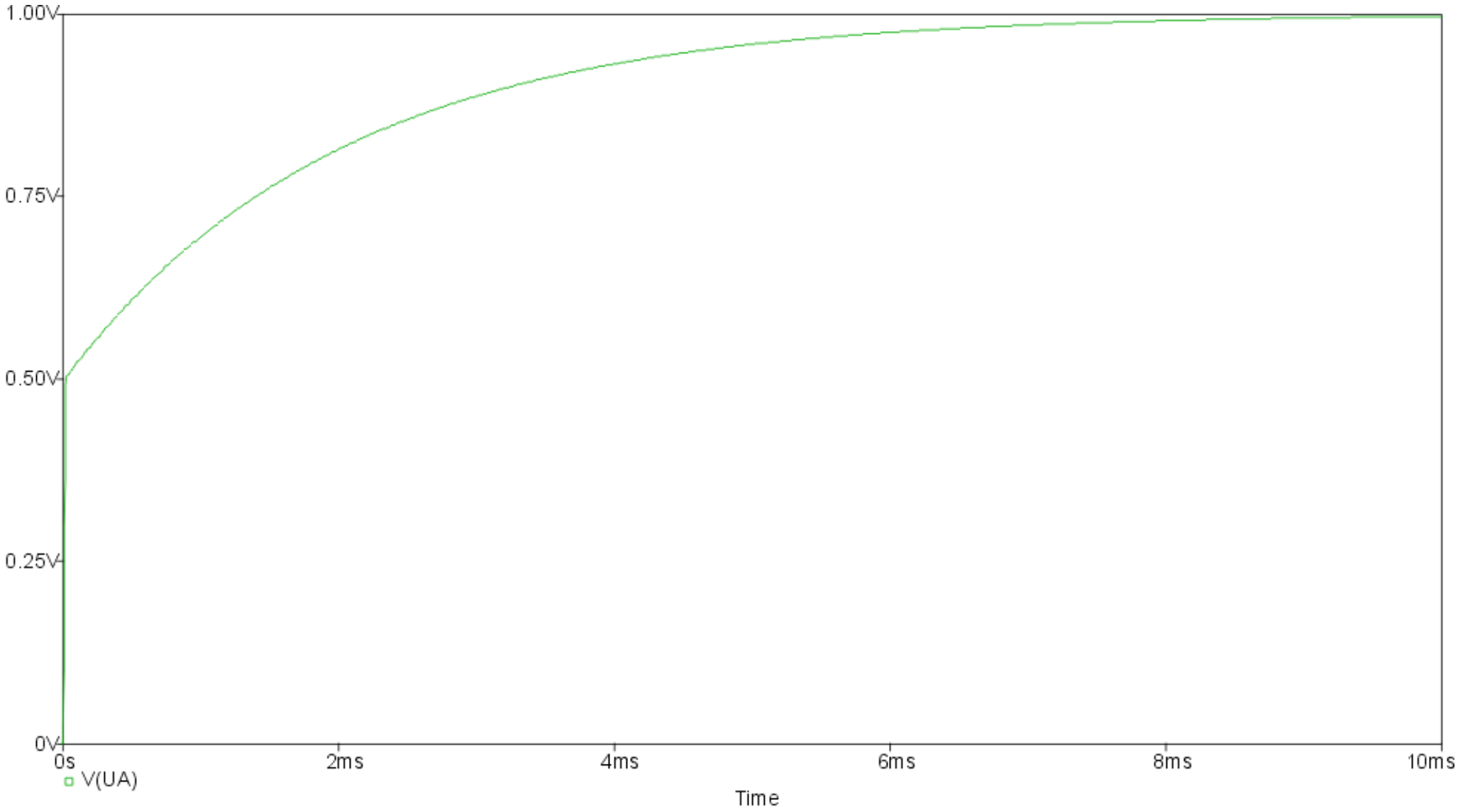
Bodediagramm (T1 > T2)



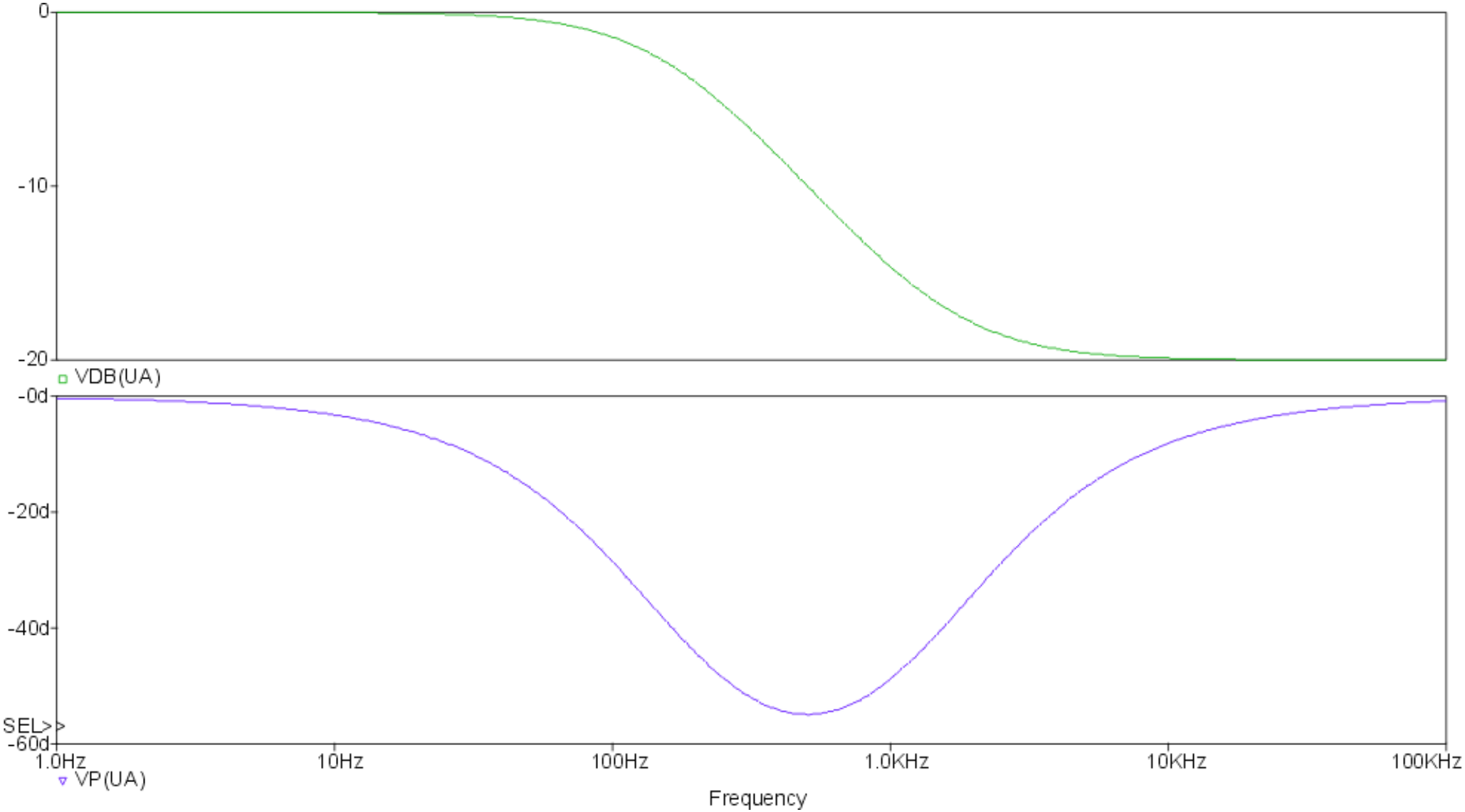
Ortskurve (T1 > T2)



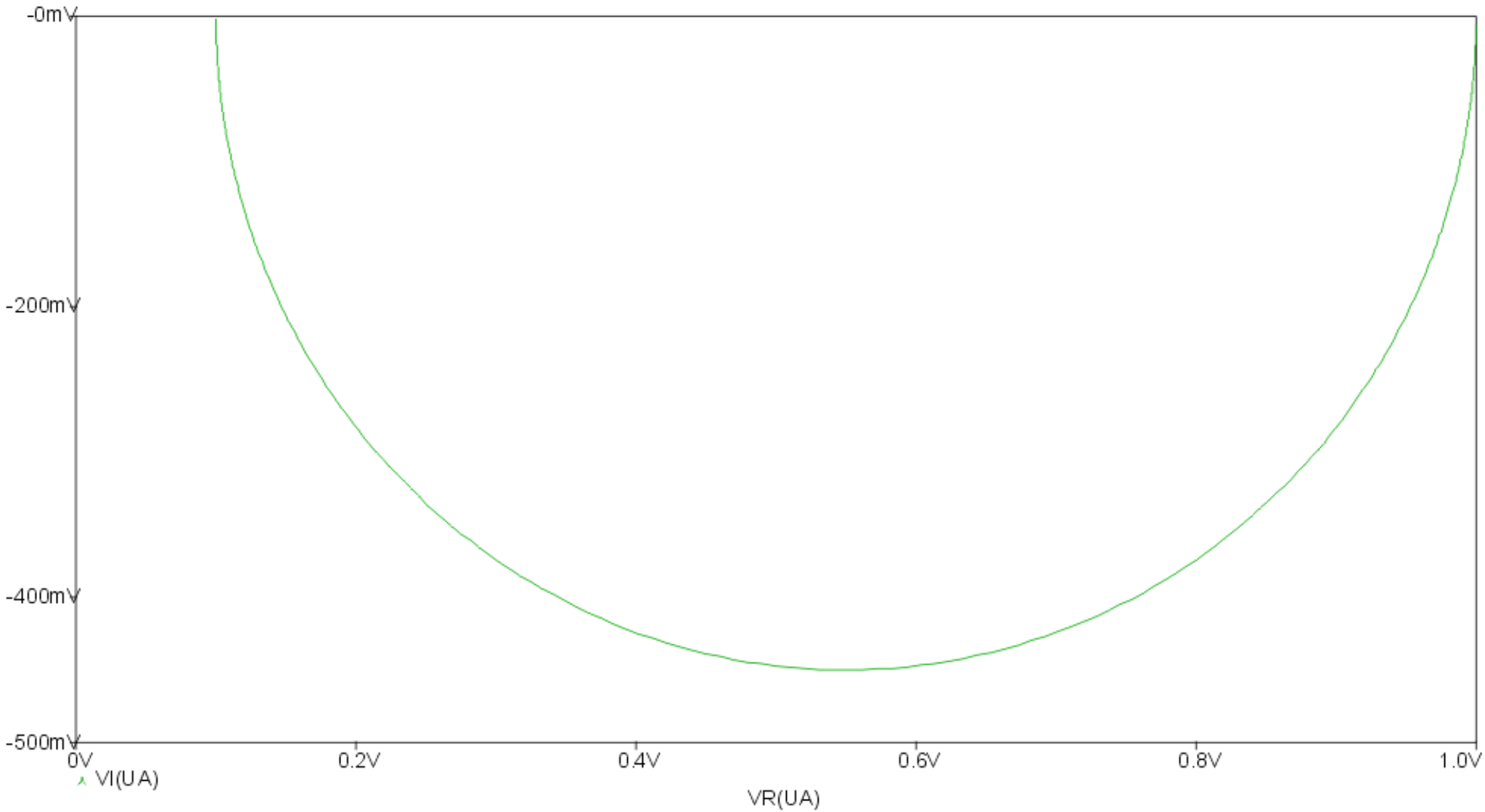
Sprungantwort (T1 < T2)



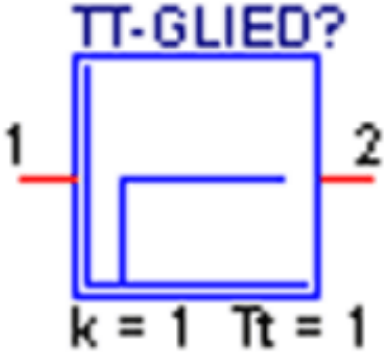
Bodediagramm (T1 < T2)



Ortskurve (T1 < T2)



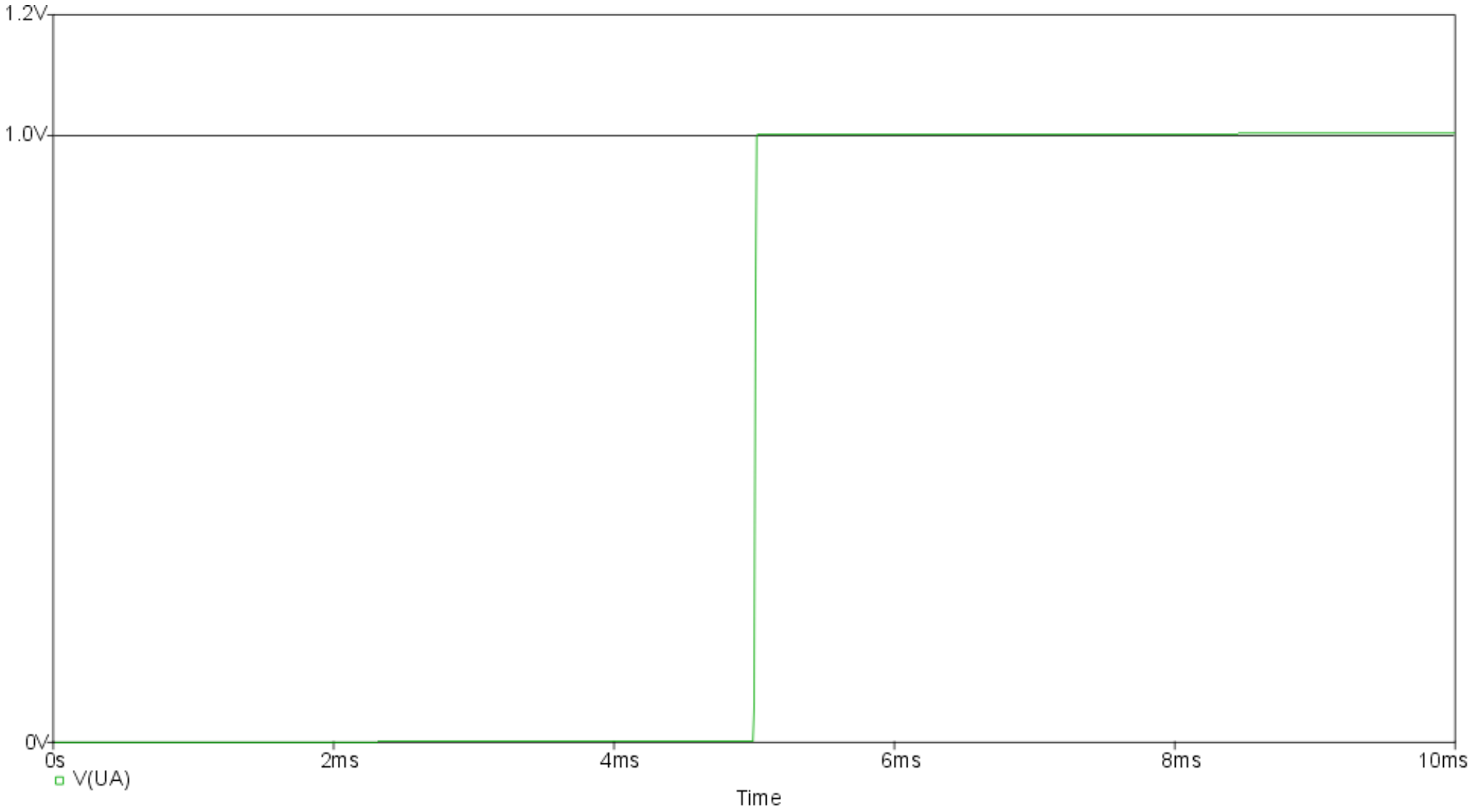
Totzeit Element



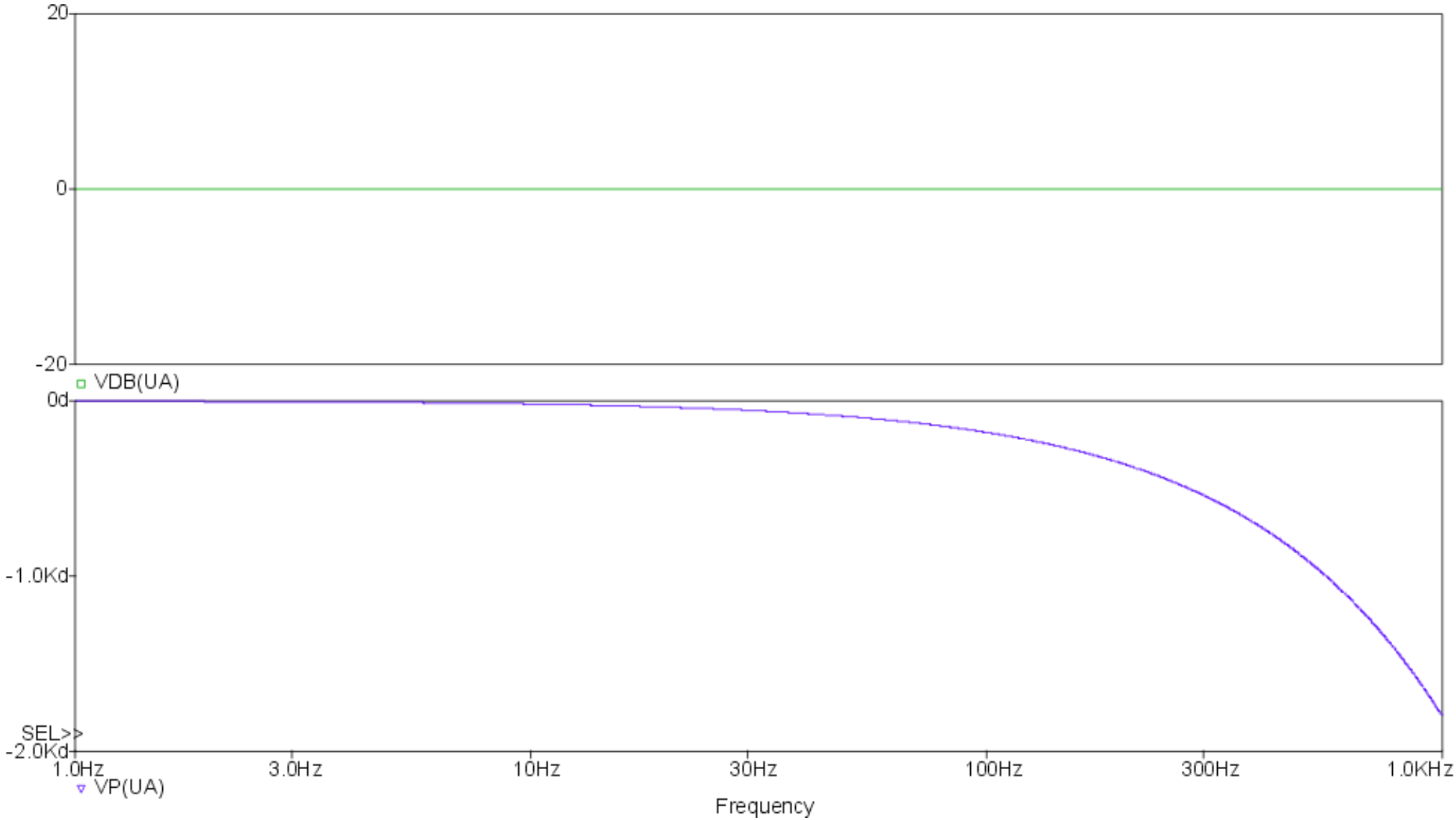
$$G(s) = e^{-sT_t}$$

Eine Totzeit in einem System bewirkt also keine Änderung des Betrages des Frequenzganges, sondern nur eine zusätzliche Phasendrehung. Die Ortskurve ist der Einheitskreis, der mit steigender Kreisfrequenz unendlich oft im Uhrzeigersinn durchlaufen wird.

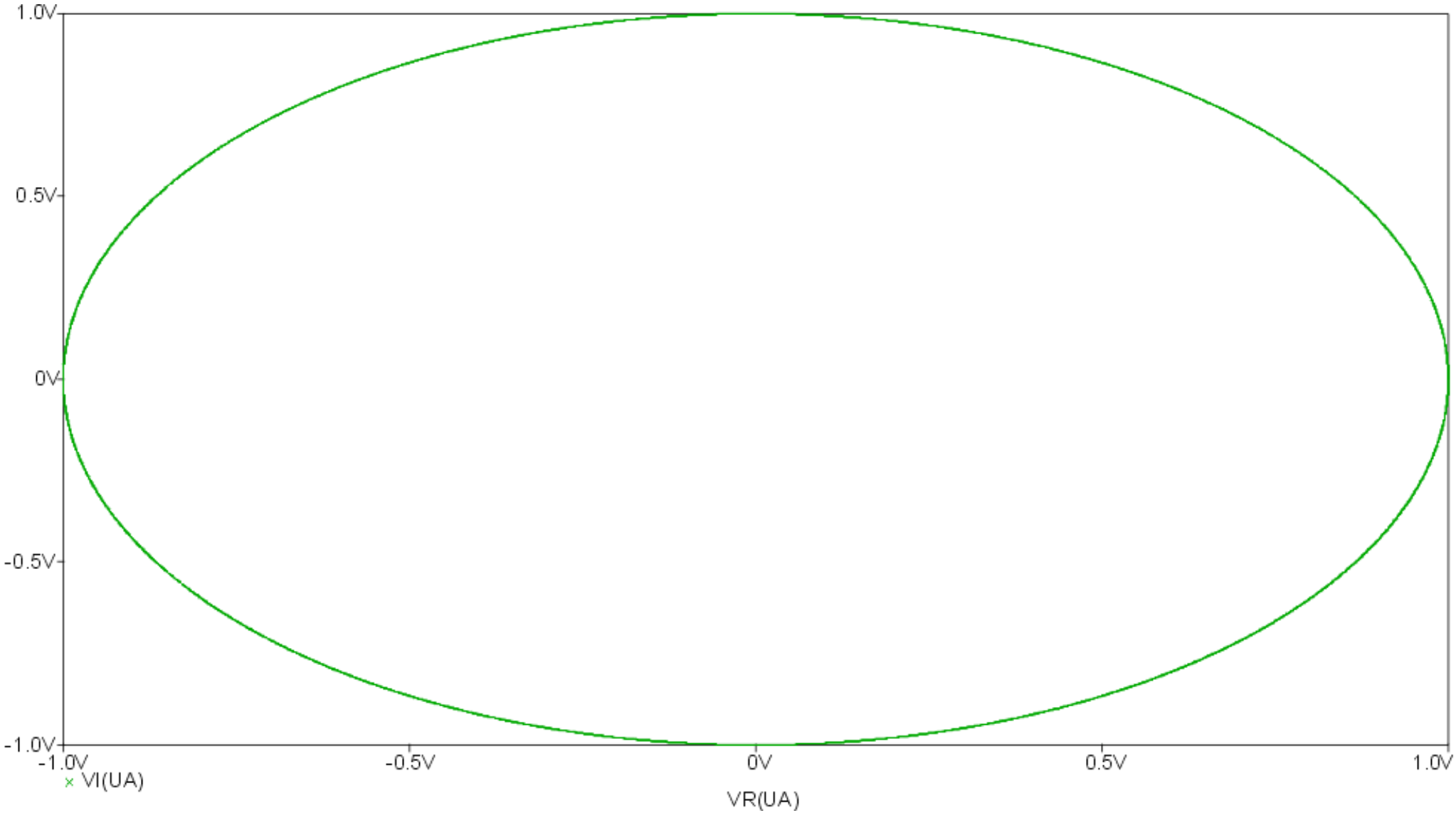
Sprungantwort



Bodediagramm



Ortskurve

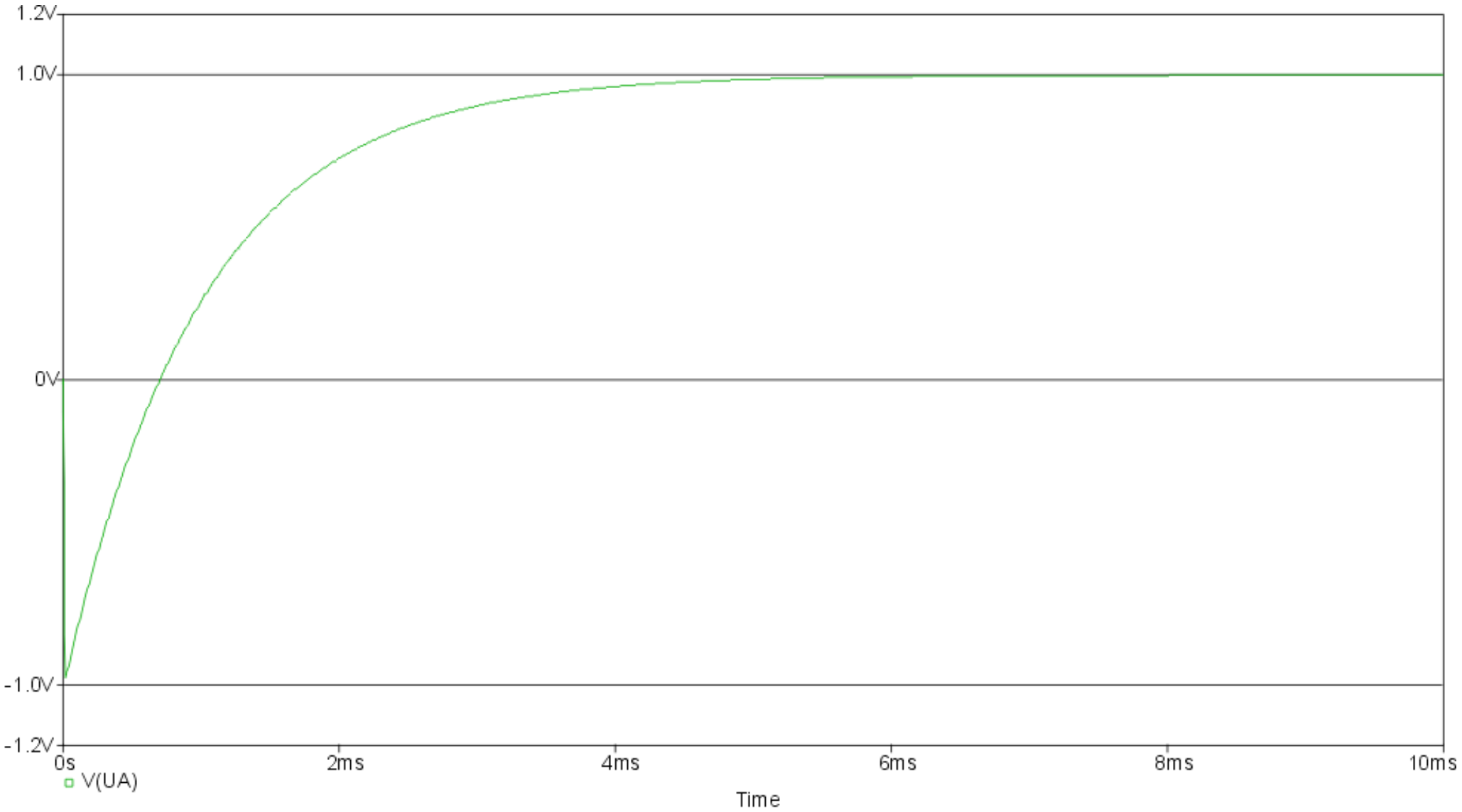


Allpass

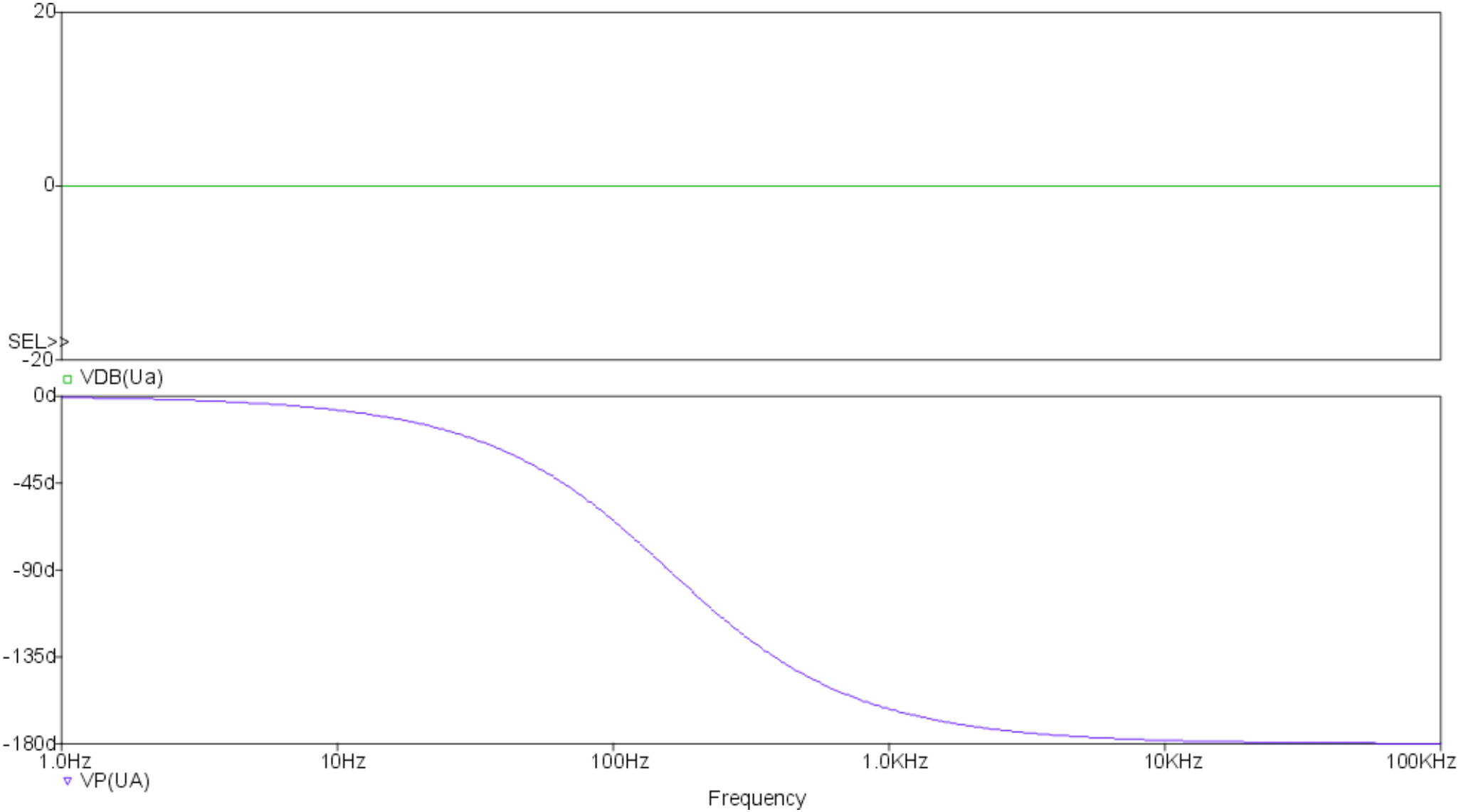
$$G(s) = \frac{1 - sT}{1 + sT}$$

Ein Allpass bewirkt in einem System ebenso wie das Totzeitglied keine Änderung des Betrages des Frequenzganges, sondern nur eine zusätzliche Phasendrehung, die aber im Gegensatz zur Totzeit zwischen 0 und -180° beschränkt bleibt. Ebenso wie Totzeiten wirken sich Allpässe aufgrund der Phasendrehung unangenehm auf das Stabilitätsverhalten von Regelkreisen aus. Ein Allpass hat außerdem die besondere Eigenschaft, dass seine Sprungantwort zunächst in die "verkehrte Richtung" ausschlägt.

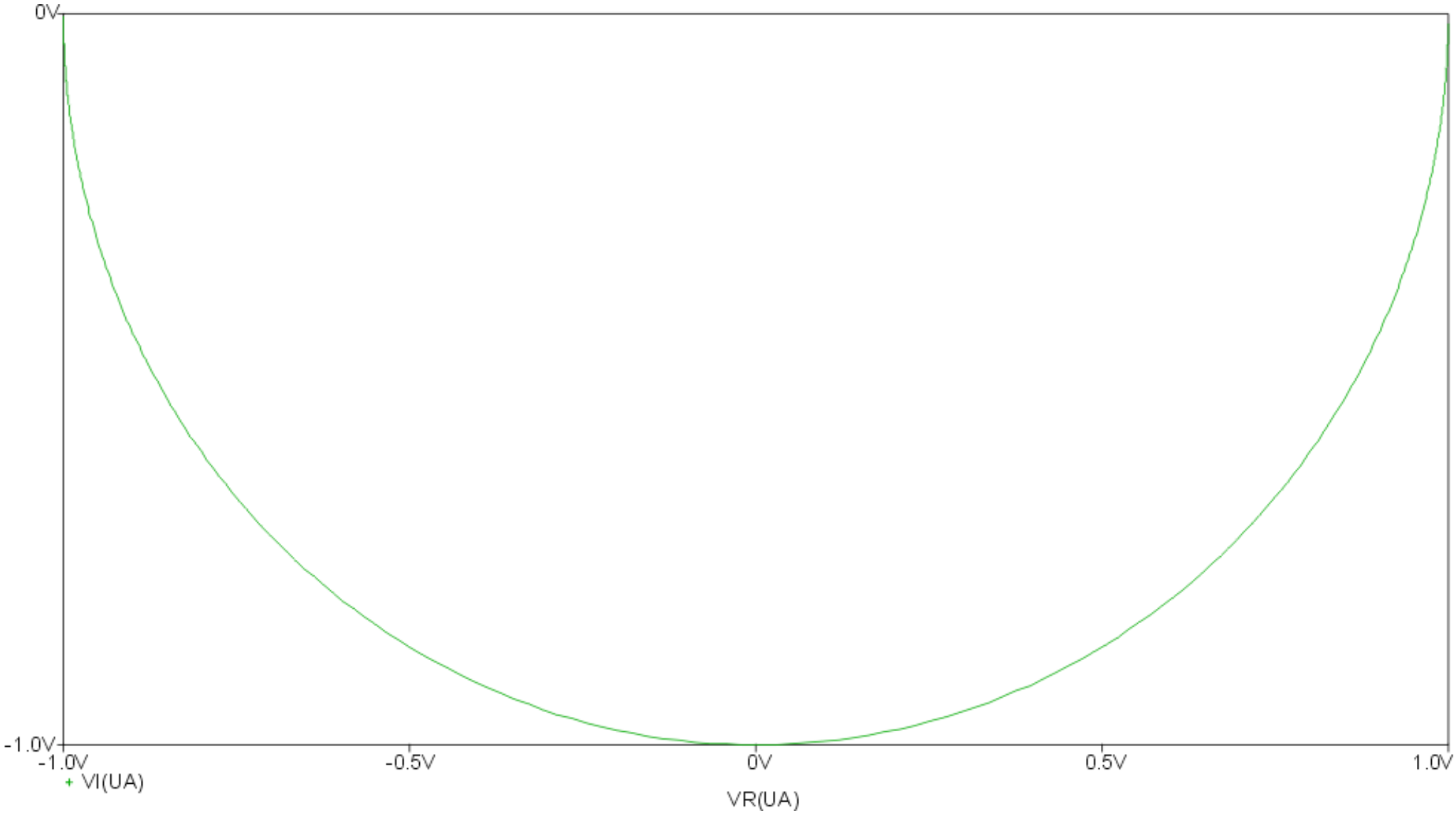
Sprungantwort



Bodediagramm



Ortskurve



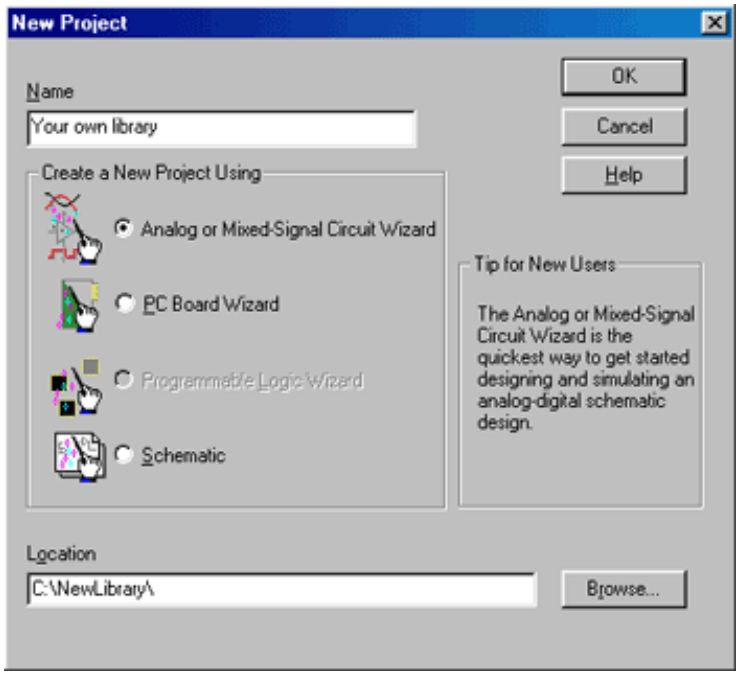
Modellerstellung

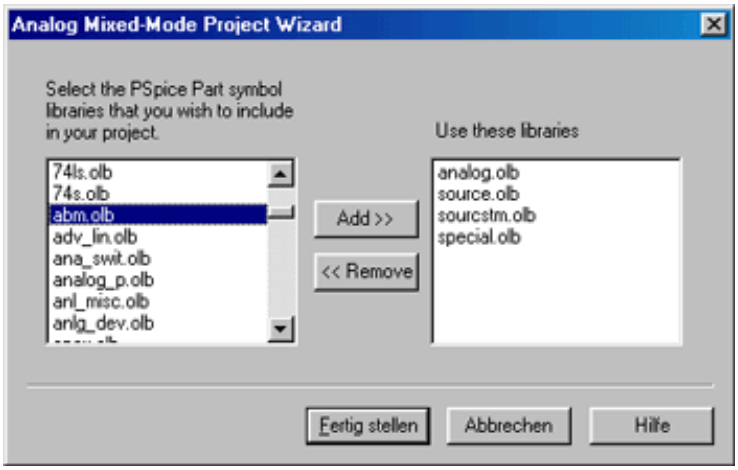
I. Ableiten eines Modells von der *ABM.OLB*

Das ist der schnellste und einfachste Weg eigene Modelle zu erstellen. Es sind nur wenige Schritte dazu nötig.

1. Erstellen eines neuen Projektes

Als erstes muss ein neues Projekt erstellt werden. Dies geschieht über die Menüpunkte ***File / New / Project***. Nach Angabe eines Projektnamens und des dazugehörigen Pfades muss noch sichergestellt werden, dass die Option ***Analog or Mixed-Signal Circuit Wizard*** gewählt ist. Anschießend werden die Angaben mit ***OK*** bestätigt.



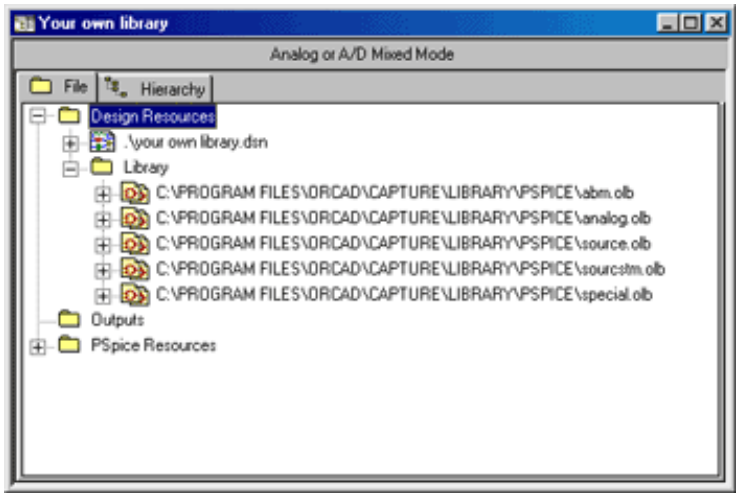


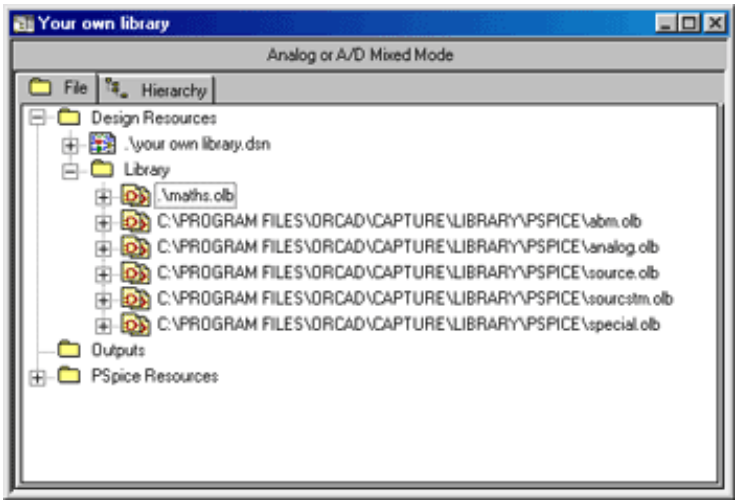
2. Die Bibliothek *ABM.OLB* zum Projekt hinzufügen

Nach dem Drücken von ***OK*** sollte sich das im Bild gezeigte Dialogfenster öffnen.
Es muss noch die Biobliothek *ABM.OLB* hinzugefügt werden. Dazu wählt man nach dem Markieren der *ABM.OLB* den Punkt ***Add***>>. Anschließend bestätigt man mit ***Fertig stellen***.

3. Erstellen einer eigenen Bibliothek

Alle dem Projekt hinzugefügten Bibliotheken werden im Projektmanager aufgelistet. Das sind unter anderem die Standardbibliotheken und die *ABM.OLB*. In unserem Fall ist die *ABM.OLB* die erste Bibliothek in der Liste. Durch Anklicken des Kreuzes wird der gesamte Bibliothekeninhalt angezeigt. Später folgt ein kleiner Überblick über die Bauteile der *ABM.OLB*. Als nächstes wird nun eine neue Bibliothek erstellt.





Dazu muss nur der Menüpunkt ***File / New / Library*** gewählt werden. Nun sollte eine neue Standardbibliothek erstellt worden sein. Diese ist nun auch im Projektmanager zu sehen. Um dieser Bibliothek einen Namen zu geben und sie an einem bestimmten Ort zu speichern, muss man nur einen Rechtsklick auf die Bibliothek im Projektmanager machen und ***Save as*** wählen. Nun sollte der Projektmanager ungefähr so aussehen (hier wurde die Bibliothek im Projektverzeichnis gespeichert) wie im Bild gezeigt.

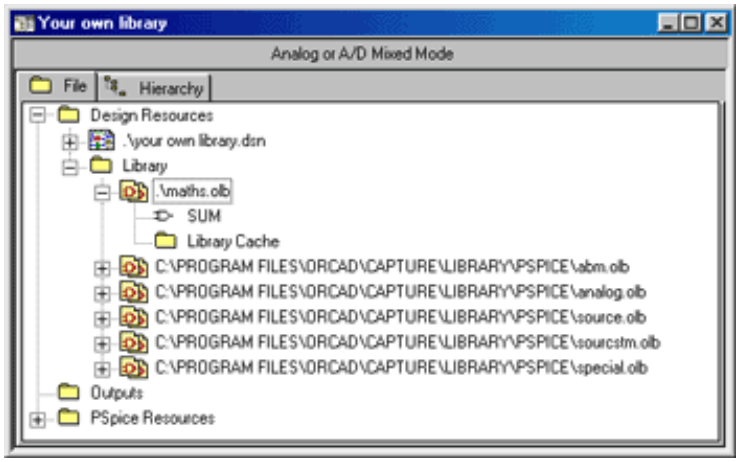
4. Bestimmung des als Vorlage dienenden Bauteils

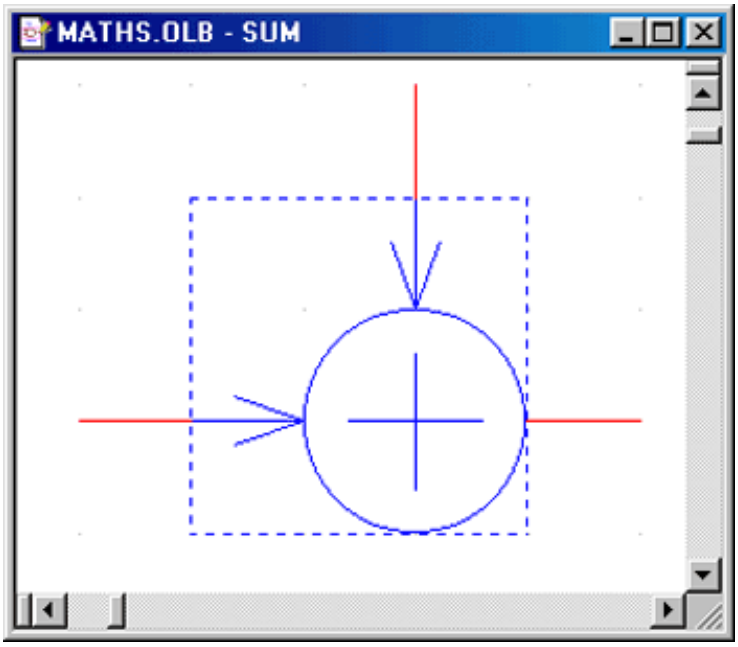
Category	Part	Description	Properties
Basic components	CONST	constant	VALUE
	SUM	adder	
	MULT	multiplier	
	GAIN	gain block	GAIN
	DIFF	subtracter	
Limiters	LIMIT	hard limiter	LO, HI
	GLIMIT	limiter with gain	LO, HI, GAIN
	SOFTLIM	soft (tanh) limiter	LO, HI, GAIN
Chebyshev filters	LOPASS	lowpass filter	FP, FS, RIPPLE, STOP
	HIPASS	highpass filter	FP, FS, RIPPLE, STOP
	BANDPASS	bandpass filter	F0, F1, F2, F3, RIPPLE, STOP
	BANDREJ	band reject (notch) filter	F0, F1, F2, F3, RIPPLE, STOP
Integrator and differentiator	INTEG	integrator	GAIN, IC
	DIFFER	differentiator	GAIN
Table look-ups	TABLE	lookup table	ROW1...ROW5
	FTABLE	frequency lookup table	ROW1...ROW5

Category	Part	Description	Properties
Laplace transform	LAPLACE	Laplace expression	NUM, DENOM
Math functions (where 'x' is the input)	ABS	$\|x\|$	
	SQRT	$x^{1/2}$	
	PWR	$\|x\|^{EXP}$	EXP
	PWRS	x^{EXP}	EXP
	LOG	$ln(x)$	
	LOG10	$log(x)$	
	EXP	e^x	
	SIN	$sin(x)$	
	COS	$cos(x)$	
	TAN	$tan(x)$	
	ATAN	$tan^{-1}(x)$	
	ARCTAN	$tan^{-1}(x)$	
Expression functions	ABM	no inputs, V out	EXP1...EXP4
	ABM1	1 input, V out	EXP1...EXP4
	ABM2	2 inputs, V out	EXP1...EXP4
	ABM3	3 inputs, V out	EXP1...EXP4
	ABM/I	no input, I out	EXP1...EXP4
	ABM1/I	1 input, I out	EXP1...EXP4
	ABM2/I	2 inputs, I out	EXP1...EXP4
	ABM3/I	3 inputs, I out	EXP1...EXP4

5. Kopieren und Editieren des Bauteils

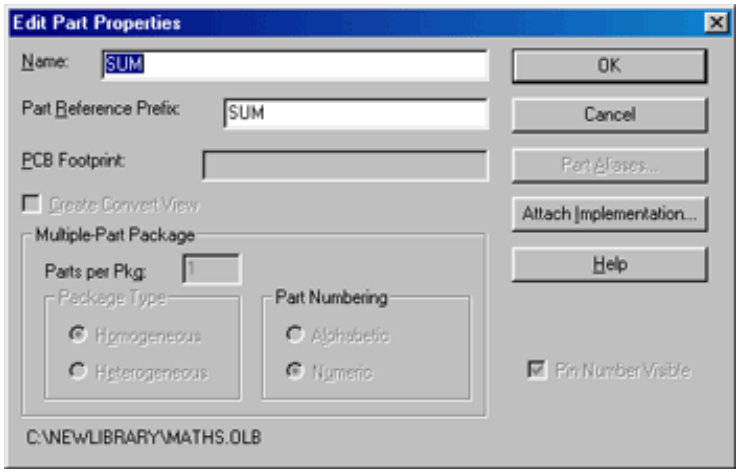
Um das Bauteil mit den gewünschten Eigenschaften in die Bibliothek zu kopieren, muss nur das gewünschte Bauteil der *ABM.OLB* angeklickt und ***STRG+C*** gedrückt werden. Anschließend wird das Bauteil durch Anklicken und Drücken von ***STRG+V*** eingefügt. Nach diesen Schritten sollte sich nun das gewünschte Bauteil in der Bibliothek befinden. Im Beispiel wurde das Bauteil *SUM* in die Bibliothek *maths* kopiert.

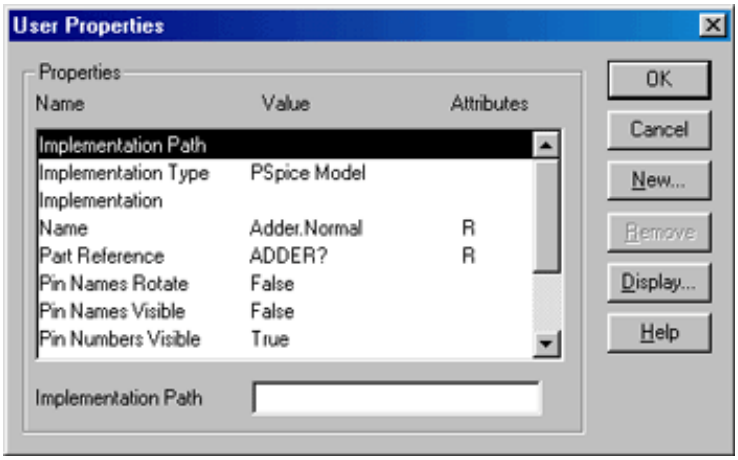




Nun kann dem eingefügten Bauteil noch ein anderer Name zugeordnet werden. Dies geschieht durch einen Doppeklick auf das Bauteil im Projektmanager. Das sich öffnende Fenster zeigt das Symbol des Bauteils. In unserem Fall einen Summierer.

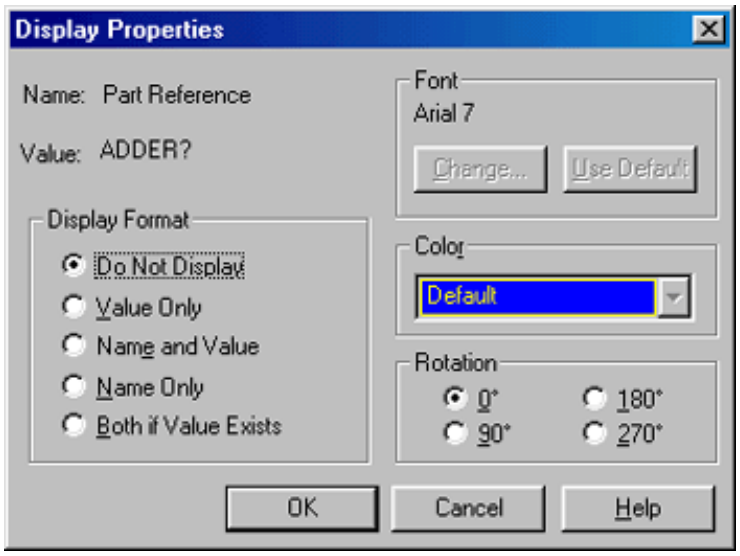
Um dem Bauteil nun einen neuen Namen zu geben, muss nur ***Options / Package Properties*** gewählt werden. Im Eingabefeld *Name* wird nun der Name des Bauteils, der im Projektmanager und bei der Bauteilauswahl erscheint, eingegeben. Im Beispiel wird das Bauteil *Adder* genannt. Im Eingabefeld *Part Reference Prefix* wird der Name des Bauteils, der bei der schematischen Darstellung der Schaltung angezeigt wird, eingegeben. Zum Beispiel *R* für Widerstände, *C* für Kondensatoren. **ACHTUNG:** Im *Part Reference Prefix* darf sich kein Leerzeichen befinden!

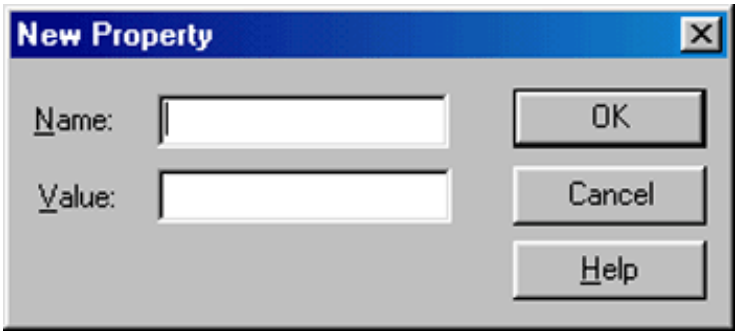




In unserem Fall wird auch hier der Bauteil mit *ADDER* benannt. Beim Klicken auf ***OK*** werden diese Namen aber nur temporär gespeichert. Jetzt sollten Nummern bei den PINS erschienen sein.
Durch einen Doppelklick auf das Symbol (nicht auf einer Linie doppelklicken!) erscheint ein neues Fenster, welches links dargestellt ist.

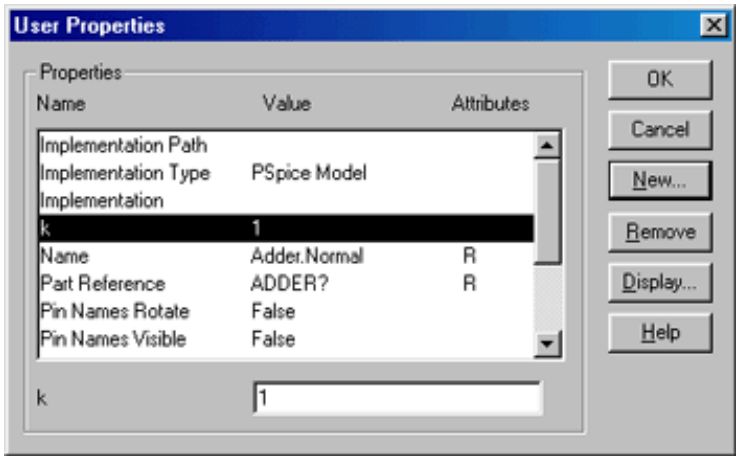
In diesem Fenster kann man alle Eigenschaften des Bauteils ändern. Die Eigenschaft *Part Reference* wurde schon im vorigen Fenster bearbeitet - sie dient zum Verändern des Namens des Bauteils in der schematischen Darstellung. Die Bearbeitung erfolgt durch einen Doppelklick auf die Eigenschaft oder einen Klick auf den Button ***Display***. In der sich öffnenden Dialogbox mit den Anzeigeeinstellungen sieht man die verschiedenen Einstellmöglichkeiten für Anzeige, Farbe, Schriftart und Größe. Um nur den Namen *Adder* anzuzeigen, muss man ***Value Only*** ankreuzen. Die Eigenschaft *Pin Numbers Visible* erlaubt es, die Pinnummern ein- bzw. auszuschalten. Mögliche Parameter *True/False*.





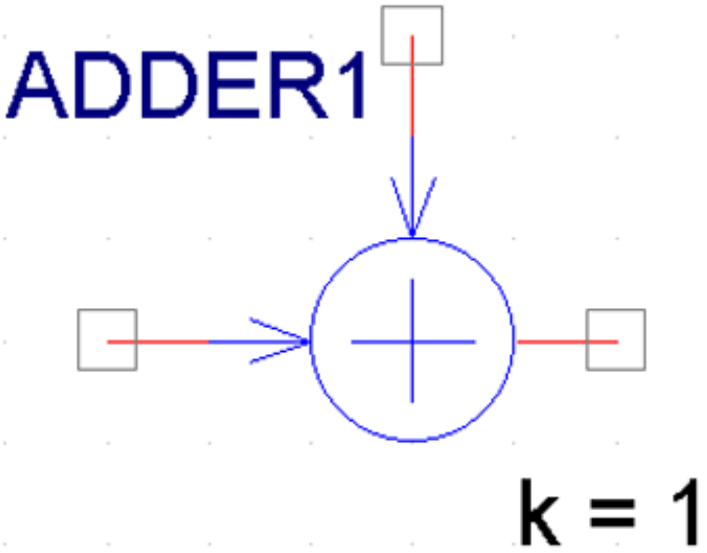
Die Eigenschaft *PSpice Template* ist das Herz eines Bauteils. Hier wird das elektrische Verhalten des Bauteils festgelegt. Bei der Modell-Erstellung über diesen Weg muss hier nichts geändert werden, da ein fertiger Bauteil mit festgelegtem Verhalten kopiert wurde.

Um zum Beispiel einen Eingang des Summierers zu verstärken, muss eine neue Variable für die Verstärkung eingefügt werden. Dieser Schritt kann durch eine simple Veränderung der Eigenschaften des Bauteils vorgenommen werden. Dazu muss nur ins Fenster ***User Properties*** gewechselt werden und über den Punkt ***New*** die Variable eingefügt werden.
In diesem Beispiel wurde eine Variable ***k*** mit dem Standardwert ***1*** eingefügt. Nachdem die Änderungen durchgeführt wurden, wird mit ***OK*** bestätigt. Die neue Eigenschaft ***k*** erscheint nun im Fenster *User Properties*. Um diese auch universell einsetzen zu können, ändert man die Anzeigeeinstellung auf ***Name and Value***.



Änderung des PSpice Templates:

Nun muss der letzte Schritt durchgeführt werden, um die Eingangsspannung zu verstärken. Dazu wird die Eigenschaft *PSpice template* geändert. Darunter findet man das vordefinierte Template des *SUM*-Bauteils aus der *ABM.OLB*. Um die Verstärkung ins Spiel zu bringen, muss der Eintrag editiert werden.
E^@REFDES %OUT 0 VALUE {V(%IN1)+V(%IN2)}
Erst muss entschieden werden, welche Eingansspannung verstärkt werden soll. Um herauszufinden welcher PIN welchen Input-Namen hat, muss man nur einen Rechtsklick auf einen PIN im Symbol durchführen und ***Edit Properties*** anklicken. Hier kann festgestellt werden welcher PIN welchen Input darstellt. Nun kann entschieden werden, welchen Input man verstärken möchte. In unserem Beispiel wird *IN1* verstärkt. Dazu muss nur ein ***@k**** vor *V(%IN1)* gegeben werden. Das @ wird benötigt, um den Wert der Variablen *k* zu nehmen. Ohne dem @ würde PSpice einen vordefinierten Wert *k* (sofern vorhanden) verwenden. Diese Modifikationen führen zu folgender Template Syntax:
E^@REFDES %OUT 0 VALUE {@k*V(%IN1)+V(%IN2)}

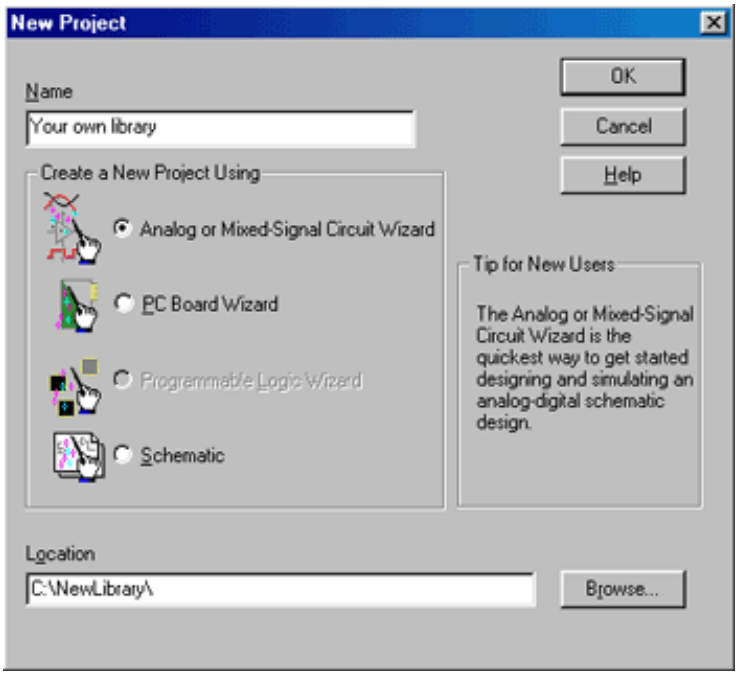


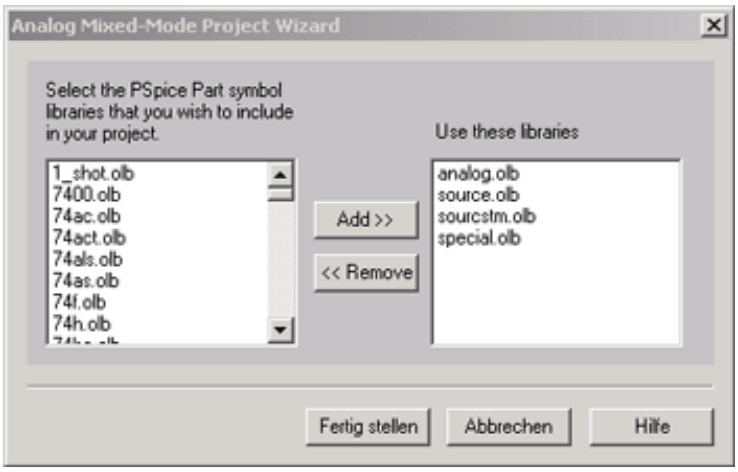
Sofern alle Schritte korrekt durchgeführt wurden, sollte der im Bild links dargestellte Summierer eine annähernde Ähnlichkeit mit dem erstellten Bauteil haben. Dieser Summierer kann dann bereits für Schaltungssimulationen verwendet werden.
Im Prinzip verläuft das Erstellen der anderen mathematischen Bauteile ident.

II. Erstellen einer neuen Bibliothek

1. Erstellen eines neuen Projektes

Als erstes muss ein neues Projekt erstellt werden. Dies geschieht über die Menüpunkte ***File / New / Project***. Nach Angabe eines Projektnamens und des dazugehörigen Pfades muss noch sichergestellt werden, dass die Option ***Analog or Mixed-Signal Circuit Wizard*** gewählt ist. Anschießend werden die Angaben mit ***OK*** bestätigt.



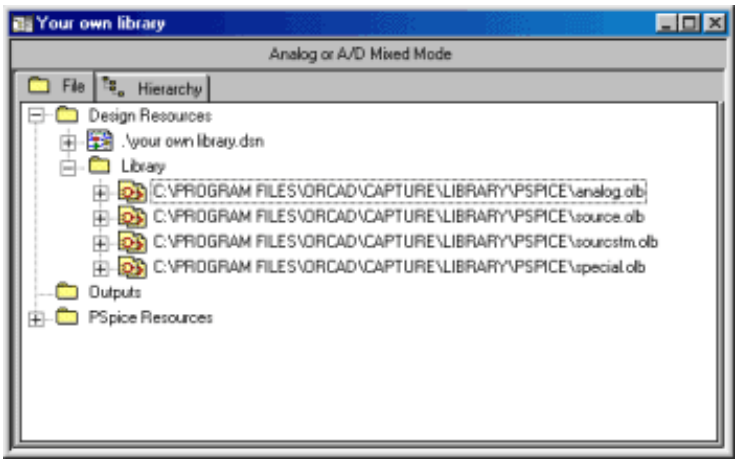


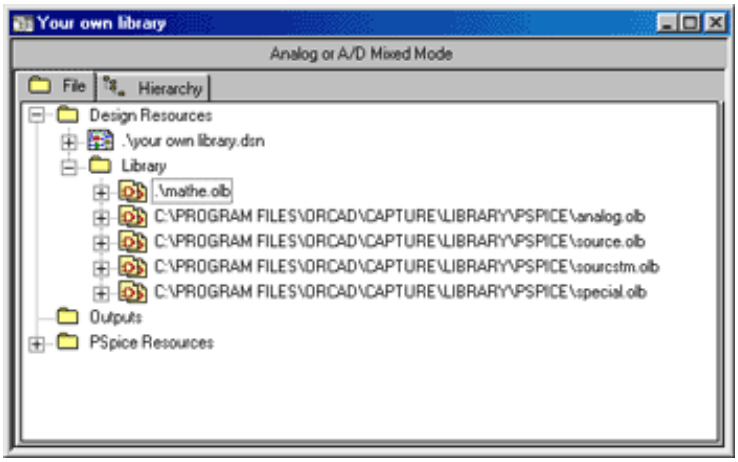
2. Bibliotheken hinzufügen (optional)

Im daraufhin erscheinenden Fenster können dem Projekt bestimmte Bibliotheken hinzugefügt werden. Für dieses Beispiel sind jedoch keine zusätzlichen Bibliotheken nötig. Nach dem Auswählen von ***Fertig stellen*** werden die für das Projekt nötigen Dateien vom Wizard erstellt.

3. Eine eigene Bibliothek erstellen

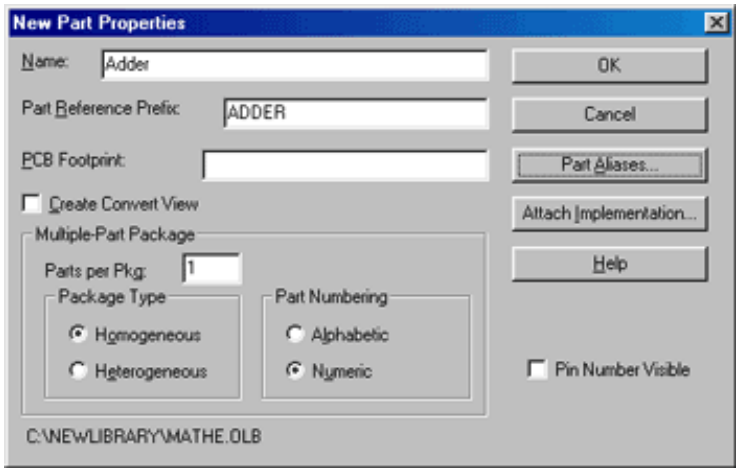
Im Projektmanager sieht man nun alle dem Projekt hinzugefügten Bibliotheken. Dies sind unter anderem die Standard-Bibliotheken *analog.olb, source.olb, special.olb und sourcstm.olb.* Ein Klick auf das Kreuz vor einer Bibliothek öffnet den gesamten Inhalt dieser Bibliothek.
Zunächst erstellen wir eine neue Bibliothek.

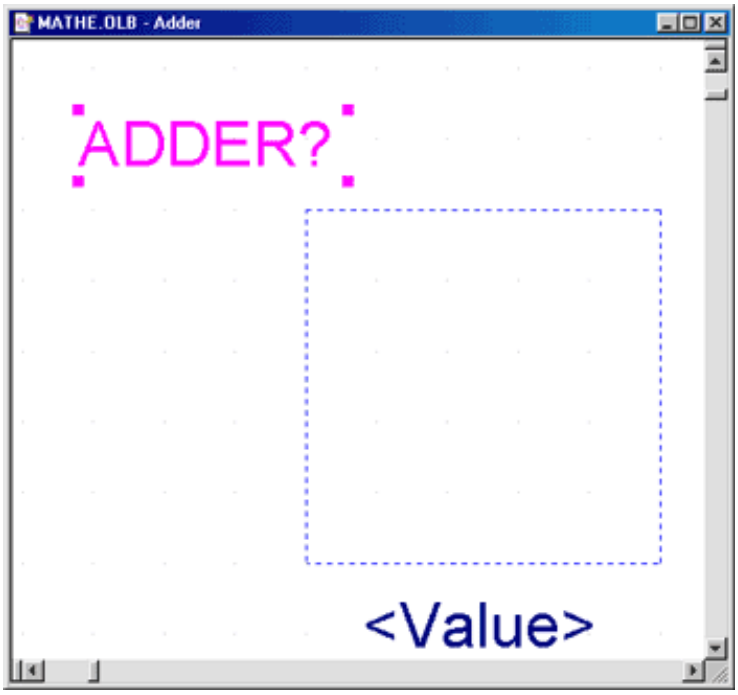




Hierzu wählt man die Menüpunkte ***File / New / Library***. Damit wird dem Projektmanager eine neue Bibliothek hinzugefügt. Mittels ***Rechtsklick*** auf die neue Bibliothek und ***Save as*** kann der gewünschte Zielort und Name der Bibliothek definiert werden. In diesem Beipiel wurde die Bibliothek *mathe.olb* genannt. Nun wird nach einem Rechtsklick auf die neue Bibliothek noch über ***New Part*** ein neuer Bauteil hinzufügt.

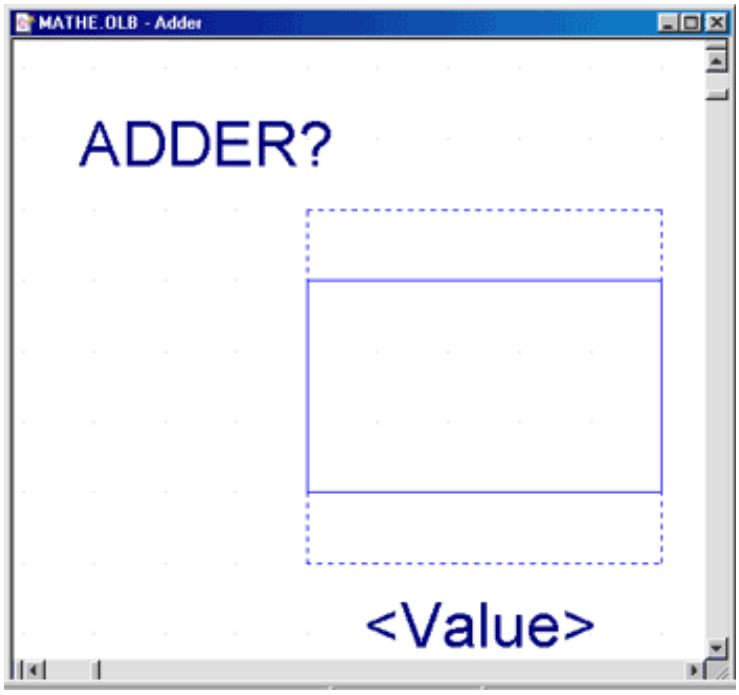
Anschließend öffnet sich ein neues Dialogfenster, in dem man die Grundeinstellungen des neuen Bauteils festlegen kann. Im Eingabefeld *Name* kann der Name des Bauteils, der im Projektmanager bzw. bei der Bauteilauswahl steht, festgelegt werden. Bei *Part Reference Prefix* wird der Name, der beim eingefügten Bauteil in einem Schaltplan angezeigt wird, festgelegt. Wichtig ist noch das *Part Numbering*. Hier wird die Zählweise der eingefügten Bauteile bestimmt: *Alphabetic (A,B,C,...)* oder *Numeric (1,2,3,...)*. Mittels Klick auf ***OK*** wird der neue Bauteil in der Bibliothek erstellt.

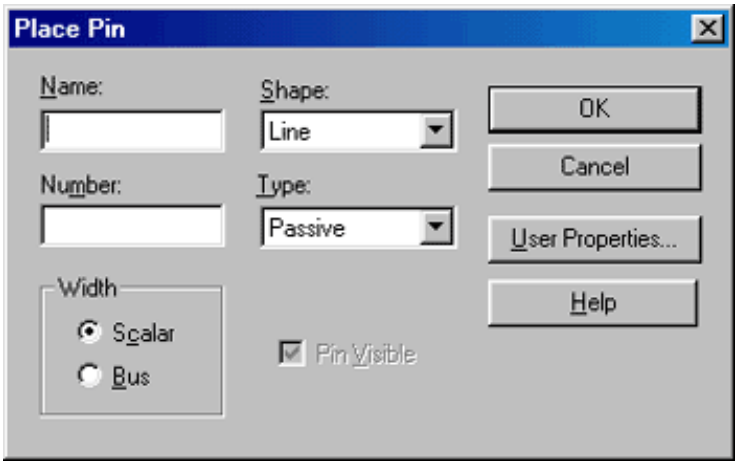




Dieser Bauteil wird dann via ***Rechtsklick*** und ***Edit Part*** bearbeitet. Wie man sieht, müssen bei dem Bauteil erst die dazugehörigen Anschlüsse erstellt und definiert werden. Dazu gibt es am rechten Rand des Fensters eine Zeichenpalette. Als erstes wird das "Begrenzungs"-Rechteck gezeichnet.

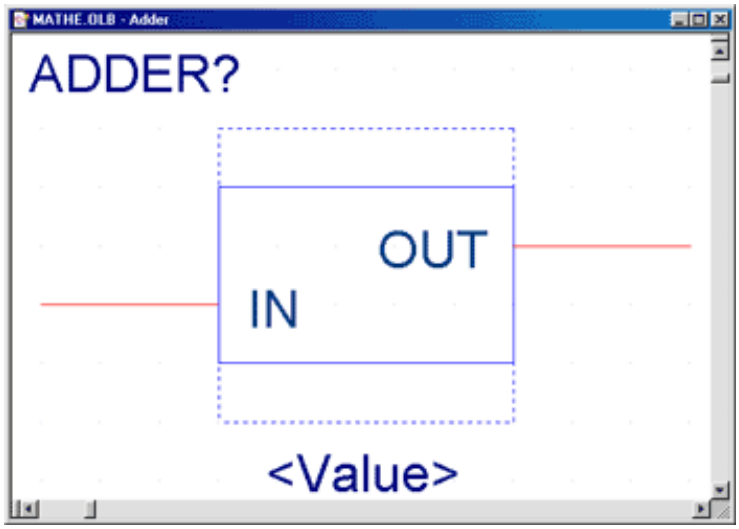
Das sollte dann ungefähr so aussehen. Um diesen Bauteil noch mit der restlichen Schaltung verbinden zu können, müssen sogenannte *PINS* eingefügt werden. Dazu dient der 3. Button der Zeichenpalette ***ADD PIN***, oder man kann auch über das Menü ***Place / Pin*** einen Anschluss einfügen.

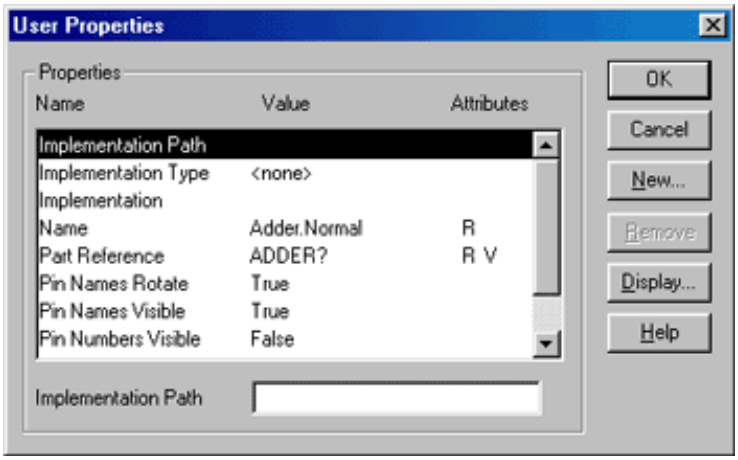




Bei dem Eingabefeld *Name* wird der Name des Pins eingegeben, im Beispiel ***IN*** für einen Eingang. Bei *Shape* kann die Form des Pins bestimmt werden, wobei ***Line*** meist am passendsten ist. Bei *Number* wird die Nummer des Pins eingegeben. Diese kann auch später dann eingeblendet werden. Unter *Type* wählt man aus, welche Art von Anschluss vorliegt (*Passiv, Aktiv,...*). Bei *Width* kann die Anzeigebreite des Pins geändert werden, um Bus-Pins von normalen zu unterscheiden.

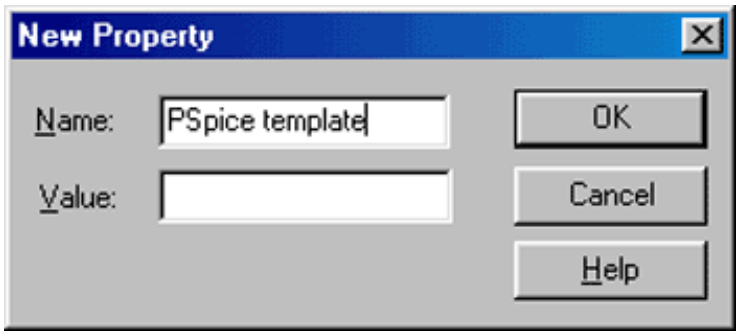
Nun wurden 2 Anschlüsse für das neue Bauteil erstellt. Das optische Layout des neuen Bauteils wäre somit fertig, nun muss man die eletrischen Eigenschaften editieren. Dazu genügt ein Doppelklick in das Fenster des Bauteils oder man gelangt über das Menü ***Options / Part Properties*** zu den Bauteileigenschaften.





Hier sieht man fast alle vordefinierten Eigenschaften des neu erstellten Bauteils. Die wichtigsten Eigenschaften sind: *Pin Names Visible* - legt fest, ob die vorher vergebenen Pin Namen angezeigt werden. Mögliche Parameter *True/False*. *Pin Numbers Visible* - legt fest, ob die vorher vergebenen Pin Nummern angezeigt werden. Mögliche Parameter *True/False*.

Nun wird über den Punkt ***New*** die Eigenschaft PSpice Template eingefügt um die elektrischen Eigenschaften des Bauteils zu beschreiben. Anschließend öffnet sich das Dialogfenster *New Property*. Man gibt bei *Name* ***PSpice template*** ein und lässt *Value* vorerst noch leer. Die Eigenschaft wird mit ***OK*** gespeichert und kann dann mit dem gewünschten Template ergänzt werden.



Nun zu einer kleinen Erklärung der "Template Syntax" für mathematische Funktionen:

Die *PSpice Template Syntax* wird in folgende Teile aufgeteilt:

<Name> <Verbundene Knoten> <Funktionsname> <Funktionsparameter>

In unserem Fall lautet der Name des Template

E^@REFDES

Bei verbundenen Knoten wird es in den meisten Fällen der Ausgang sein. Dazu verwendet man folgende Syntax:

%OUT 0

Bei Funktionsname kommen nun alle von PSpice unterstützten Funktionen in Frage - in unserem Fall wird dafür die Funktion *VALUE* ausreichen, da man damit fast alle Funktionen bewerkstelligen kann. Daraus ergibt sich nun die in diesem Zusammenhang immer gebrauchte Syntax:

E^@REFDES %OUT 0 VALUE {}

In die geschwungenen Klammern können nun alle gewünschten mathematischen Funktionen eingefügt werden, das sind:

V(%IN1)+V(%IN2) + ...		Addition von 2 oder mehreren Eingängen
V(%IN1)-V(%IN2) - ...		Subtraktion von 2 oder mehreren Eingängen
V(%IN1)*V(%IN2) * ...		Multiplikation von 2 oder mehreren Eingängen
V(%IN1)/V(%IN2) / ...		Division von 2 oder mehreren Eingängen
ABS(V(%IN))		Absolutwert
SQRT(V(%IN1))		Wurzel
PWR(V(%IN),@EXP)		%IN^EXP (*EXP* muss definiert werden!)

PWRS(V(%IN),@EXP)		|%IN|^EXP (*EXP* muss definiert werden!)
LOG(V(%IN))		natürlicher Logarithmus (LN)
LOG10(V(%IN))		10er Logarithmus (LOG)
EXP(V(%IN))		e^%IN
SIN(V(%IN))		sin(%IN)
COS(V(%IN))		cos(%IN)
TAN(V(%IN))		tan(%IN)
ATAN(V(%IN))		atan(%IN)

Für Laplace Operatoren muss das Template ein bisschen abgeändert werden:

E^@REFDES %OUT 0 LAPLACE {V(%IN)} {(@NUM)/(@DENOM)}

Laplace-Gleichungen müssen in PSpice immer auf einen gemeinsamen Nenner gebracht werden!

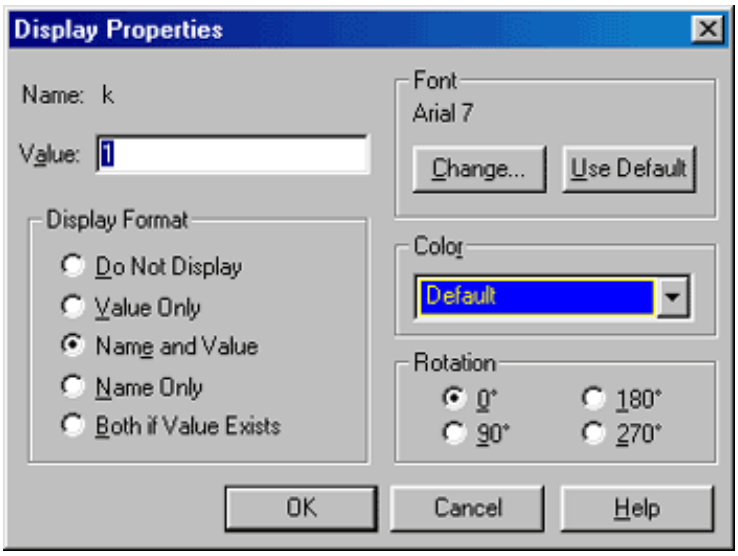
@NUM		Zähler der Gleichung (muss wieder eingefügt werden!)
@DENOM		Nenner der Gleichung (muss wieder eingefügt werden!)

Als Laplace Operator wird ***s*** verwendet.

Nun kann man die komplette Syntax für einen Addierer, der einen konstanten Wert addiert, erstellen:

E^@REFDES %OUT 0 VALUE {V(%IN)+5V}

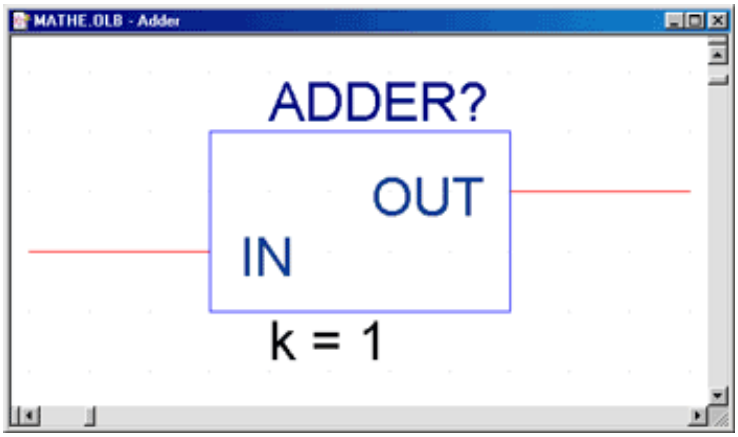
Man kann natürlich diesen konstanten Wert von aussen frei veränderbar machen. Dazu muss man nur eine neue Eigenschaft einfügen. In unserem Fall wurde sie ***k*** gennant und ein Anfangswert von ***1*** zugeordnet.



Nun kann man diesen Wert auch in der schematischen Darstellung des Netzwerkes anzeigen. Dazu klickt man die gewünschte Eigenschaft an und wählt daraufhin ***Display***. Bei ***Name and Value*** wird der Name der Eigenschaft als auch der Wert angezeigt. Mittels dieser Anzeige kann der Wert in der schematischen Darstellung auch einfach geändert werden. Nun muss dem Template noch mitgeteilt werden, dass es diesen Wert dazuaddieren soll. Dazu wird das Template folgendermaßen modifiziert:

E^@REFDES %OUT 0 VALUE {V(%IN)+@k}

Nun sollte das fertig gestellte Bauteil wie im Bild gezeigt aussehen.
Auf diese Weise lassen sich die oben angeführten mathematischen Funktionen einfach erstellen.



Pin - Formen

In PSpice können unterschiedliche Formen des Pins gewählt werden, aber keiner der Formen hat Auswirkungen auf die Funktion als Pin. Sie sind lediglich zum Layout des Bauteils einstellbar.

Shape		Description
Dot		An inversion bubble.
Clock		A clock symbol.
Dot-Clock		A clock symbol with an inversion bubble.
Zero		A normal pin with a lead zero grid units in length.
Short		A normal pin with a lead one grid unit in length.
Line		A normal pin with a lead three grid units in length.

Pin - Typen

Hier nun eine kleine Übersicht über die verschiedenen Typen von Pins, die man beid er Erstellung von eigenen Modellen auswählen kann.

Pin type	Description
3-state	A 3-state pin has three possible states: low, high, and high impedance. In its high impedance state, a 3-state pin looks like an open circuit. For example, the 74LS373 latch has 3-state pins.
Bidirectional	A bidirectional pin acts as both input and output. For example, pin 2 on the 74LS245 bus transceiver is a bidirectional pin. The value at pin 1 (an input) determines the activity of pin 2, as well as others.
Input	An input pin is one to which you apply a signal. For example, pins 1 and 2 on the 74LS00 NAND gate are input pins.
Open collector	An open collector gate omits the collector pull-up. Use an open collector to make "wired-OR" connections between the collectors of several gates and to connect with a single pull-up resistor. For example, pin 1 on the 74LS01 NAND gate is an open collector gate.
Open emitter	An open emitter gate omits the emitter pull-down. The proper resistance is added externally. ECL logic uses an open emitter gate and is analogous to an open collector gate. For example, the MC10100 has an open emitter gate.
Output	An output pin is one to which the part applies a signal. For example, pin 3 on the 74LS00 NAND gate is an output pin.
Passive	A passive pin is typically connected to a passive device. A passive device does not have a source of energy. For example, a resistor lead is a passive pin.
Power	A power pin expects either supply voltage or ground. For example, on the 74LS00 NAND gate, pin 14 is VCC and pin 7 is GND.

Üblicherweise benötigt man nicht alle diese Typen. Die meistens benötigten Typen wären:

- Input (als Eingang des Bauteils)
- Output (als Ausgang)

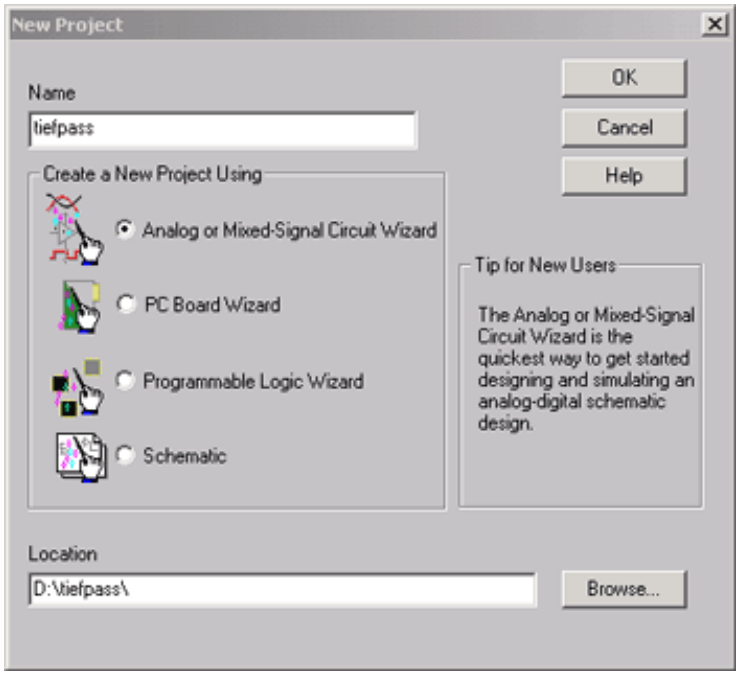
Subcircuits

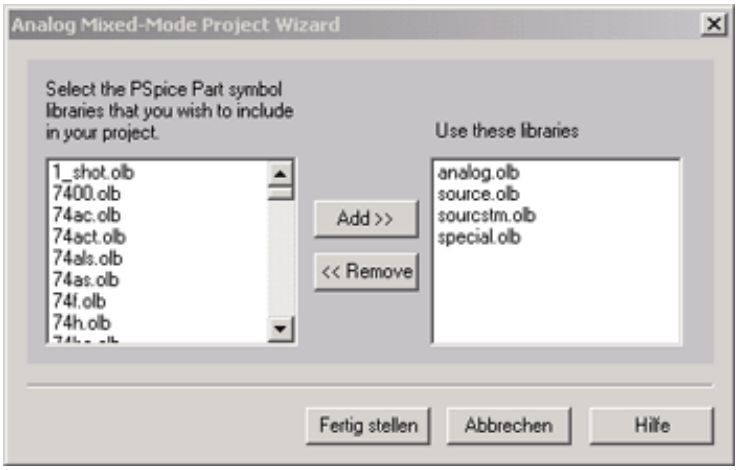
I. Erstellen der Schaltung und der Netzliste

Grundlagen in punkto Zeichnen von Schaltplänen und Umgang mit "Capture" werden vorausgesetzt.

1. Erstellen eines neuen Projektes

Als erstes muss ein neues Projekt erstellt werden. Dies geschieht über die Menüpunkte ***File / New / Project***. Nach Angabe eines Projektnamens und des dazugehörigen Pfades muss noch sichergestellt werden, dass die Option ***Analog or Mixed-Signal Circuit Wizard*** gewählt ist. Anschießend werden die Angaben mit ***OK*** bestätigt.



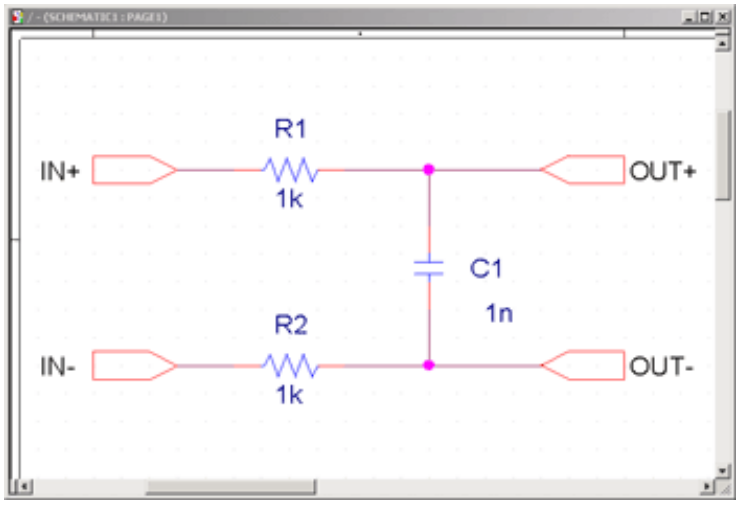


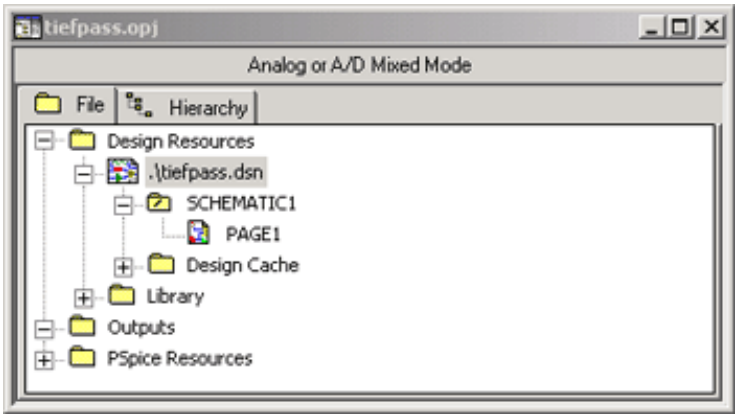
2. Bibliotheken hinzufügen (optional)

Im daraufhin erscheinenden Fenster können dem Projekt bestimmte Bibliotheken hinzugefügt werden. Für dieses Beispiel sind jedoch keine zusätzlichen Bibliotheken nötig. Nach dem Auswählen von ***Fertig stellen*** werden die für das Projekt nötigen Dateien vom Wizard erstellt.

3. Erstellen der Schaltung

Als nächstes muss die Schaltung, für die ein Unternetzwerk erstellt werden soll, im Schematic-Fenster erstellt werden. Das ist in unserem Fall ein als Vierpol ausgeführter RC-Tiefpass. Für Ein- bzw. Ausgänge werden ***Hierarchical Ports*** verwendet - für Eingänge *PORTRIGHT-R* und für Ausgänge *PORTLEFT-L*. In dem Beispiel tragen die Eingänge die Bezeichnung *IN+* und *IN-* und die Ausgänge *OUT+* und *OUT-*. *R2* verhindert die direkte Verbindung zwischen *IN-* und *OUT-*.

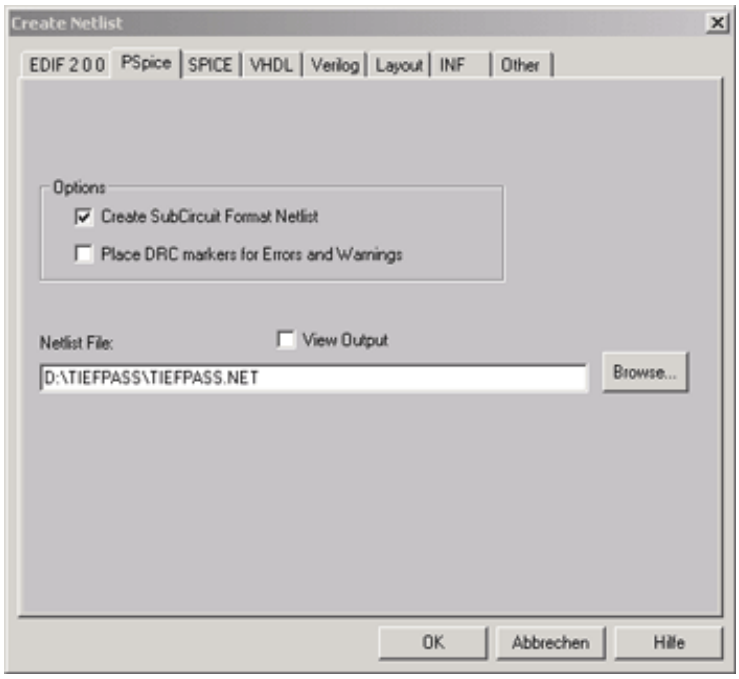


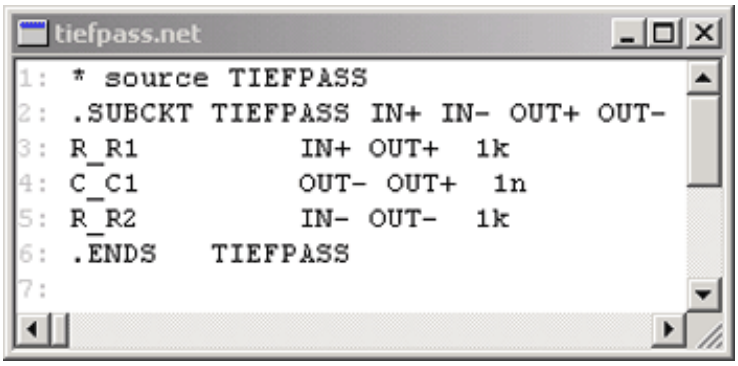


4. Erstellen der Netzliste

Die Netzliste wird benötigt, um für die entworfene Schaltung eine Library-Datei (Endung .LIB) erstellen zu können - außerdem muss diese im Subcircuit-Format vorliegen. Dazu markiert man zunächst die Design-Datei (Endung .DSN), für die das Unternetzwerk erstellt werden soll. Dann wählt man vom Menü ***Tools / Create Netlist***.

Im sich öffnenden Fenster wechselt man dann auf die Registerkarte ***PSpice*** und wählt dort die Option ***Create SubCircuit Format Netlist***. Da ansonsten keine weiteren Angaben nötig sind, bestätigt man die Einstellung anschließend mit ***OK***. Es wird unter Umständen eine Warnung angezeigt, die jedoch ignoriert werden kann.





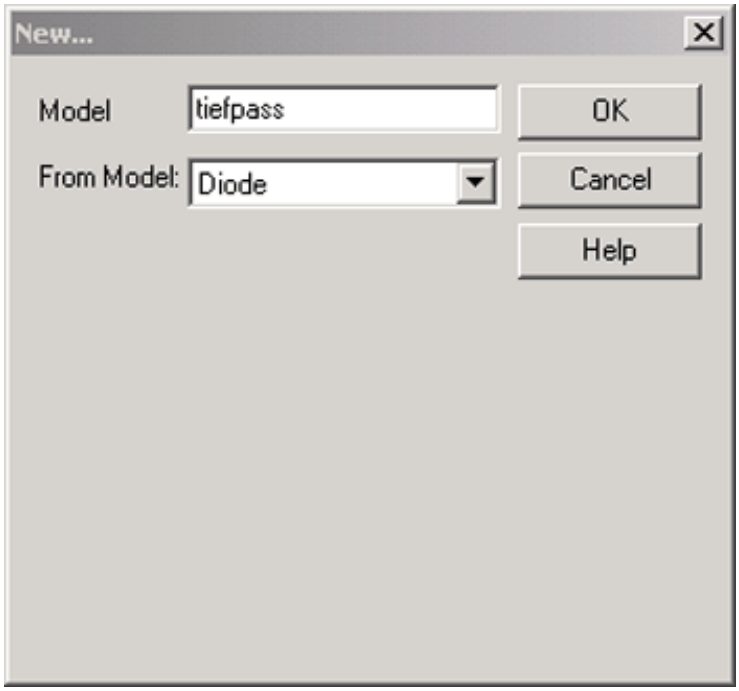
Daraufhin wird im Projektmanager unter *Outputs* die Datei ***tiefpass.net*** erzeugt. Diese Datei kann mit einem Doppelklick geöffnet werden und sollte den im Bild links gezeigten Inhalt haben. Die Syntax dieser Dateien ist unter Subcircuits - Allgemein näher beschrieben.

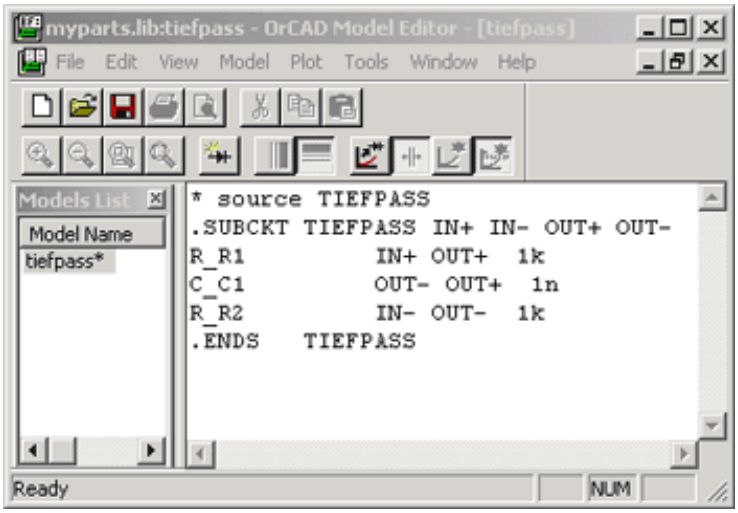
II. Erstellung und Einbindung der Library-Datei

In den folgenden Schritten wird eine LIB-Datei für das Unternetzwerk erstellt.

1. Starten des Model Editors

Als erstes muss der Model Editor gestartet und über ***File / Save As*** eine neue Bibliothek erstellt werden. Im sich öffnenden Fenster wählt man einen Pfad und einen Namen für die zu erstellende .LIB-Datei (im Beispiel *D:\tiefpass\myparts.lib*). Dann wird über den Punkt ***Model / New*** ein neues Model hinzugefügt. Unter *Model* wird dann der gewünschte Name eingegeben (im Beispiel *tiefpass*) und unter *From Model* ***Diode*** gewählt.





2. Anpassen der .LIB-Datei

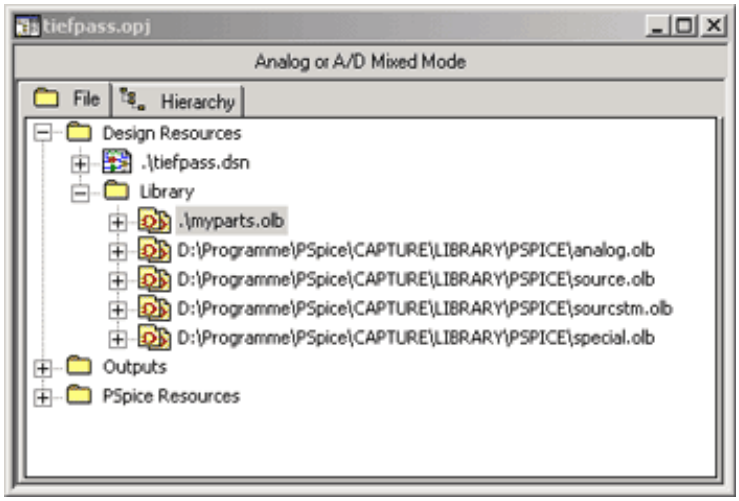
Nun muss die vom Wizard generierte .LIB-Datei modifiziert werden. Dazu wird die Option ***View / Model Text*** gewählt und daraufhin die gesamte Definition entfernt. Als nächstes wird die von Pspice generierte Subcircuit-Definition (tiefpass.net) an diese Stelle kopiert. Falls keine weiteren Bauteile mehr definiert werden müssen, kann der Model Editor nach dem Speichern (***File / Save***) verlassen werden.

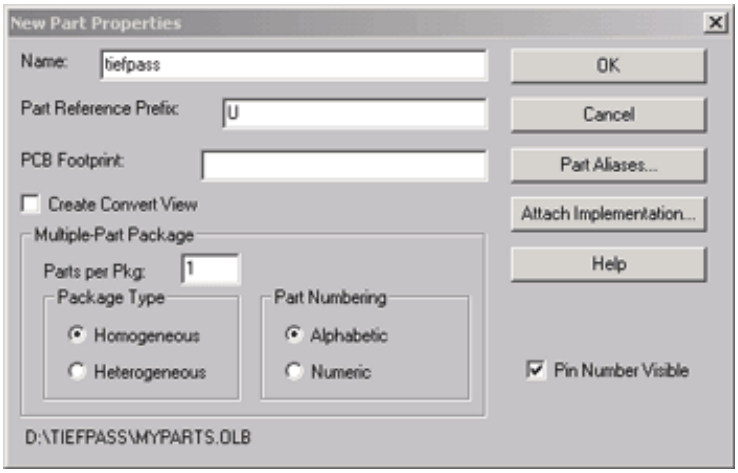
III. Erstellung des Bauteils in Capture

In den folgenden Schritten wird ein Bauteil für die erstellte .LIB-Datei erzeugt.

1. Erstellen einer neuen Bibliothek

Man wählt ***File / New / Library***, klickt die als *library1.olb* erzeugte Datei an und wählt ***File / Save As***. Danach wird ein Pfad und Name für die Datei gewählt und mit ***Speichern*** bestätigt.



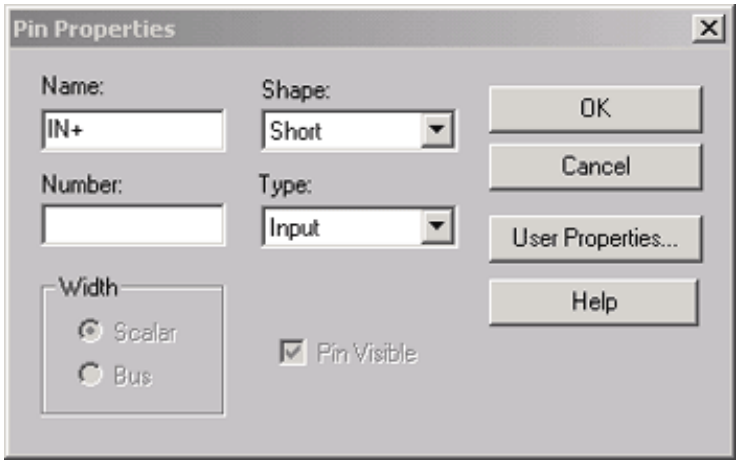


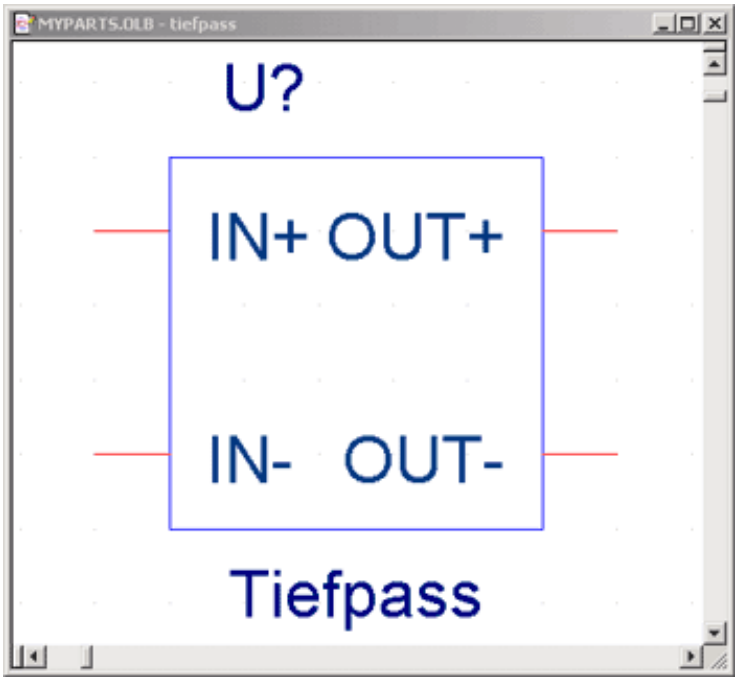
2. Hinzufügen des Bauteils

Nach dem Vergewissern, dass im Projektmanager die soeben erstellte Bibliothek markiert ist, wählt man ***Design / New Part***. Als *Name* wird *Tiefpass* gewählt, danach wird mit ***OK*** bestätigt.

3. Zeichnen des Bauteils

Im sich öffnenden Fenster wird werden nun die Ports für den Bauteil festgelegt. Man wählt ***Place / Pin*** und trägt die vorher beim Zeichnen der Schaltung definierten Pins ein.
ACHTUNG: Die Pin-Namen müssen die gleiche Bezeichnung wie in der .LIB-Datei haben! Weiters muss zwischen *Input* und *Output* unterschieden werden!

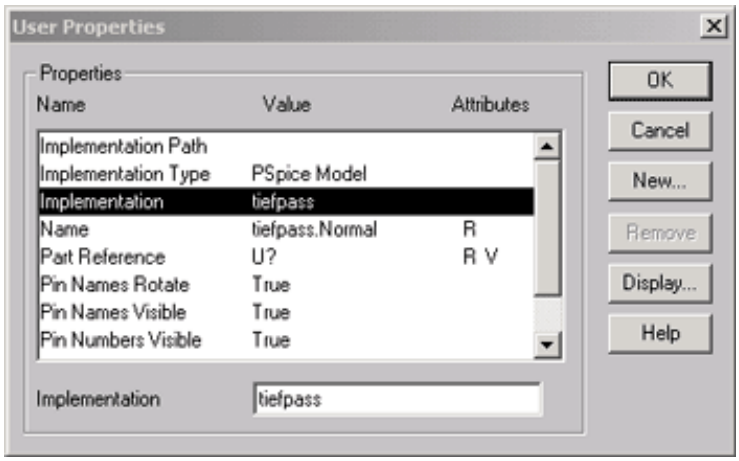




Nach dem Zeichnen des "Begrenzungs-Rechtecks" und Ersetzen des vordefinierten Wertes *Value* durch *Tiefpass* sollte der Teil wie im Bild links gezeigt aussehen. Als nächstes wird das Projekt über ***File / Save*** gespeichert.

4. Editieren der Eigenschaften

Nun müssen die Eigenschaften des Bauteils über ***Options / Part Properties*** angepasst werden. Als *Implementation Type* wird ***PSpice Model*** und als *Implementation* ***tiefpass*** gewählt.



Nun wird über ***New*** ein neuer Eintrag mit dem Namen ***PSpiceTemplate*** mit dem Wert

X^@REFDES %IN+ %IN- %OUT+ %OUT- @MODEL

erstellt.
Näheres zu dieser Syntax unter Template Syntax.

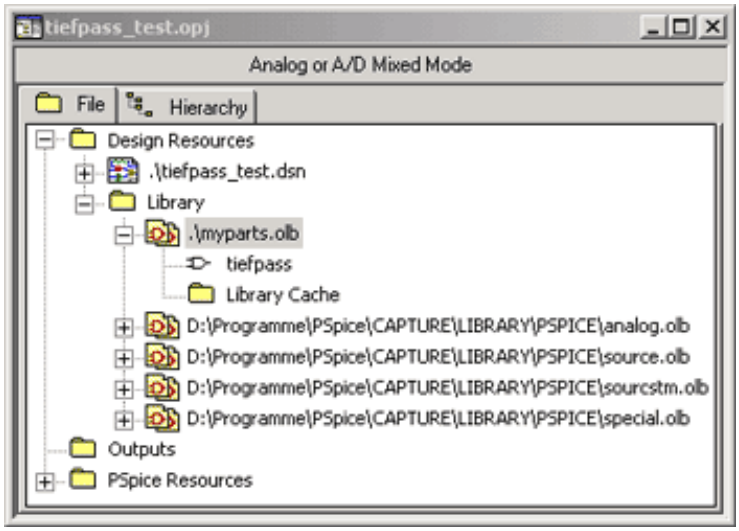
IV. Testen des Bauteils

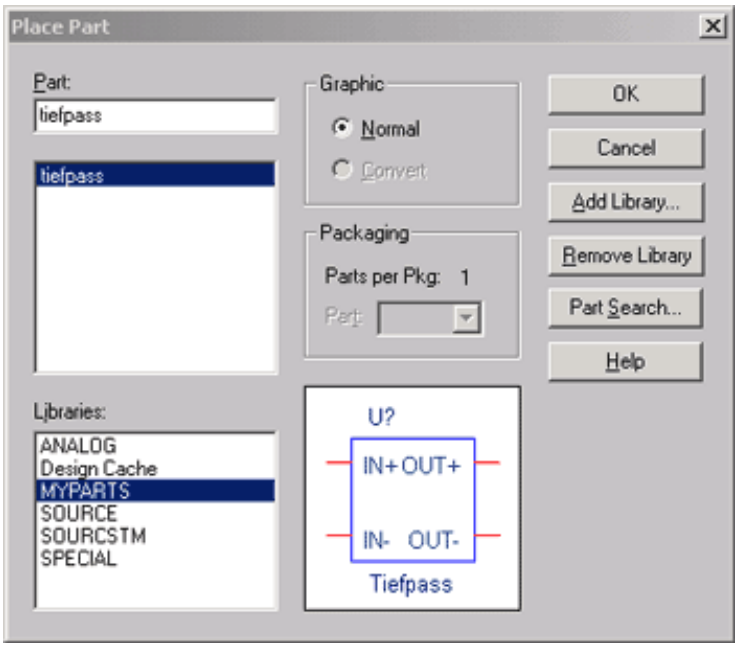
Grundlagen in punkto Programmbedienung und Durchführung von Simulationen werden vorausgesetzt.

1. Erstellen eines neuen Projektes

Nachdem das alte Projekt inklusive dem Bauteil abgespeichert wurde, wird zum Testen des Bauteils ein neues Projekt erstellt. (im Beispiel *tiefpass_test*)

Anschließend wird die vorhin erstellte Library über einen Rechtsklick auf *Library* und dann ***Add File*** hinzugefügt.

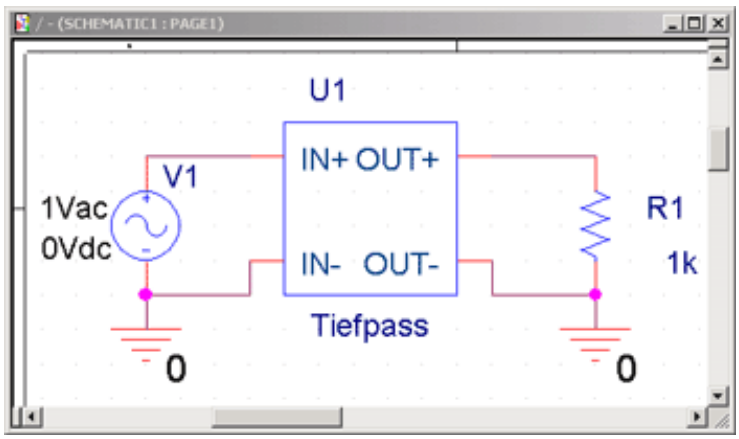


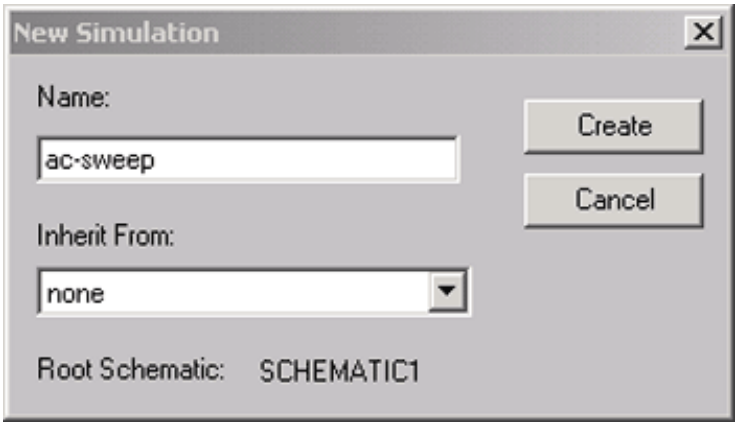


Der Part wird dann über ***Place / Part*** in die Schematic eingefügt. Dazu wird unter *Libraries* die Library ***MYPARTS.OLB*** angeklickt und der bis dahin einzige Bauteil darin mit dem Namen ***tiefpass*** ausgewählt. Die Aktion wird abschließend mit ***OK*** bestätigt.

2. Aufbau einer kleinen Testschaltung

Zum Testen wird nun die rechts gezeigte Schaltung aufgebaut. Als Quelle wurde im Beispiel ***VAC*** verwendet.

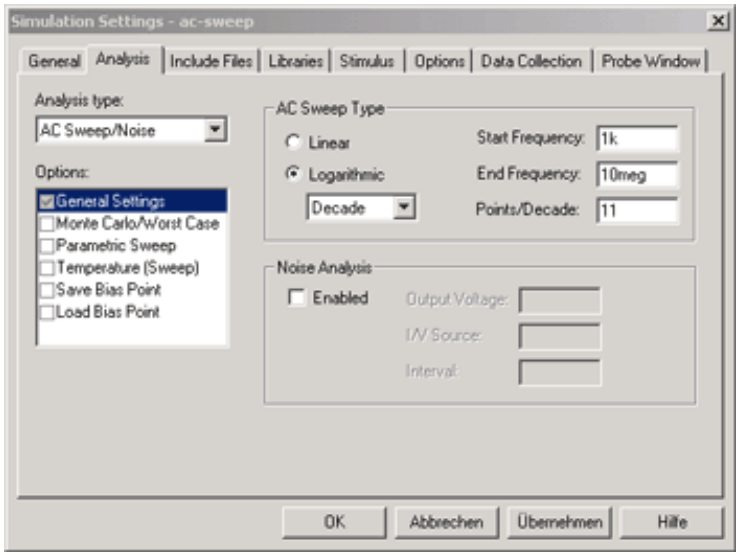


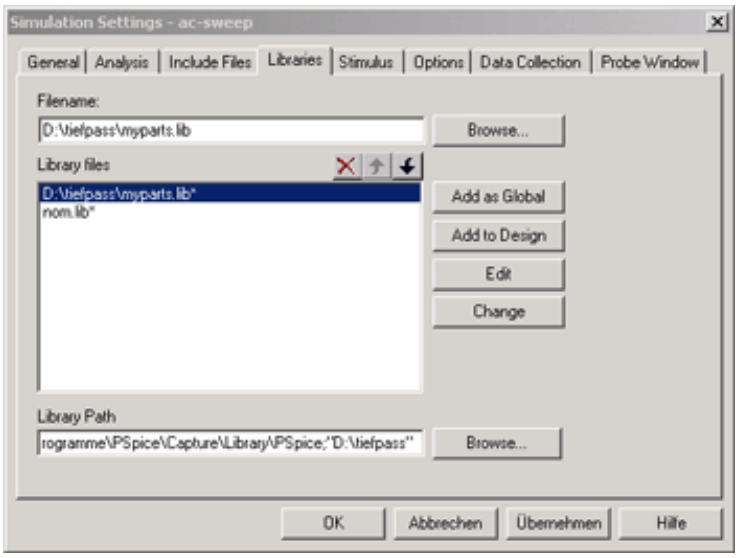


3. Starten der Simulation

Nun wird über ***PSpice / New Simulation Profile*** eine Simulation erstellt. Als *Name* wird ***ac-sweep*** angegeben. Anschließend wird das Profil durch Klicken von ***Create*** erstellt.

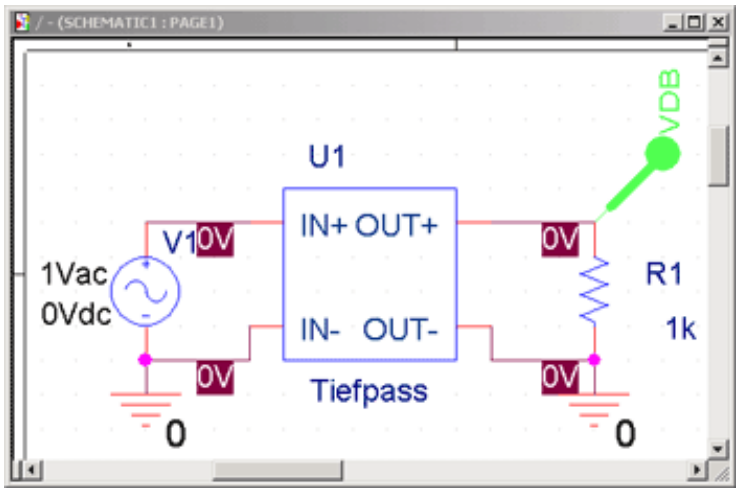
Im sich öffnenden Fenster wurden die im Bild gezeigten Einstellungen verwendet. Wichtig ist nur, dass unter der Registerkarte ***Analysis*** beim Punkt *Analysis Type* ***AC Sweep/Noise*** gewählt wird.

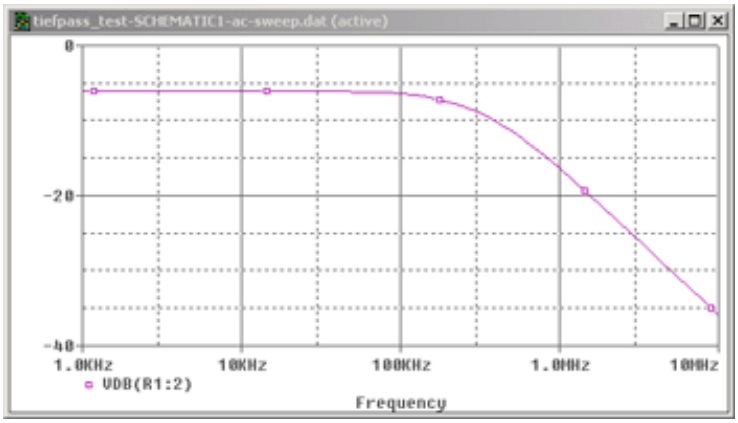




Weiters muss sichergestellt werden, dass unter der Registerkarte ***Libraries*** der Pfad (*Library Path*) zur erstellten Bibliothek (im Beispiel *D:\tiefpass*) und die dazugehörige Datei (im Beispiel *D:\tiefpass\myparts.lib*) über ***Add as Global*** zum Simulationsprofil hinzugefügt wird. Die Einstellungen werden mit ***OK*** bestätigt.

Nun wird die Simulation über ***Pspice / Run*** aufgerufen und anschließend der Schaltung über ***PSpice / Markers / Advanced / dB Magnitude of Voltage*** noch ein Marker am Ausgang hinzugefügt (diese Marker sind nur bei AC-Simulationen zugänglich!).





Das Probe-Fenster sollte nun den im Bild gezeigten, typischen Amplitudengang eines Tiefpasses darstellen.

V. Übergabe von Parametern

In den folgenden Schritten wird anhand des erstellten Beispieles erläutert, wie Parameter an die Innenschaltung des Unternetzwerkes (in unserem Fall für R1, R2, und C1) übergeben werden können.

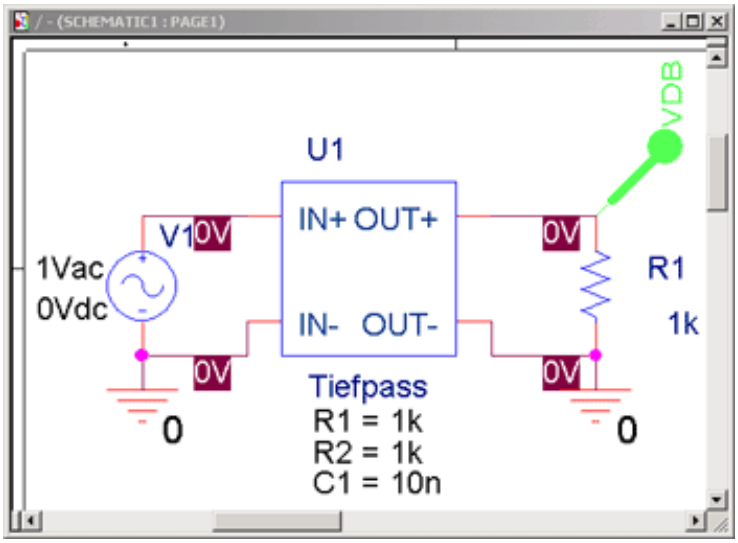
1. Modifizierung der .LIB-Datei

Als erstes muss die vorhin generierte Netzliste im Subcircuit-Format modifiziert werden. Dazu wird bei der angefertigten Schaltung das Bauteil angeklickt und der Punkt ***Edit / PSpice Model*** gewählt. Daraufhin wird der Model Editor aufgerufen und die Unternetzwerk-Definition angezeigt. Das Bild rechts zeigt die nötigen Änderungen. Auf die Syntax dieser Dateien wird unter [Allgemein] näher eingegangen.

```
* source TIEFPASS
.SUBCKT TIEFPASS IN+ IN- OUT+ OUT-

+ PARAMS: R1=1k C1=1n R2=1k

R_R1          IN+ OUT+    {R1}
C_C1          OUT- OUT+   {C1}
R_R2          IN- OUT-    {R2}
.ENDS    TIEFPASS
```



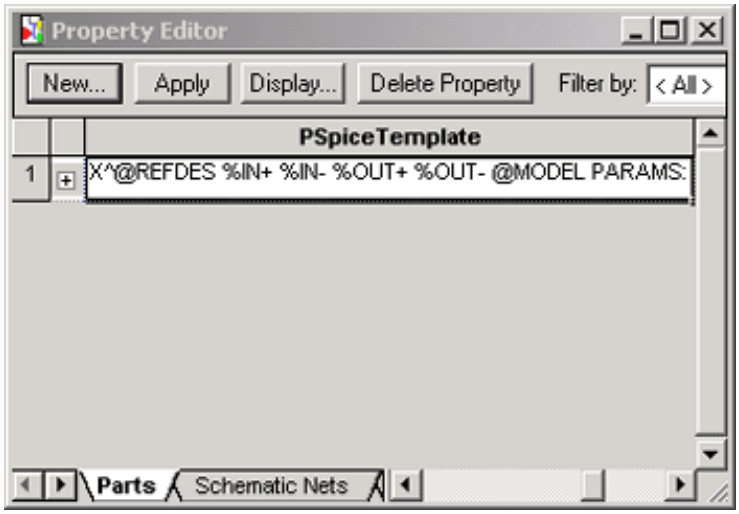
2. Anpassen der Eigenschaften

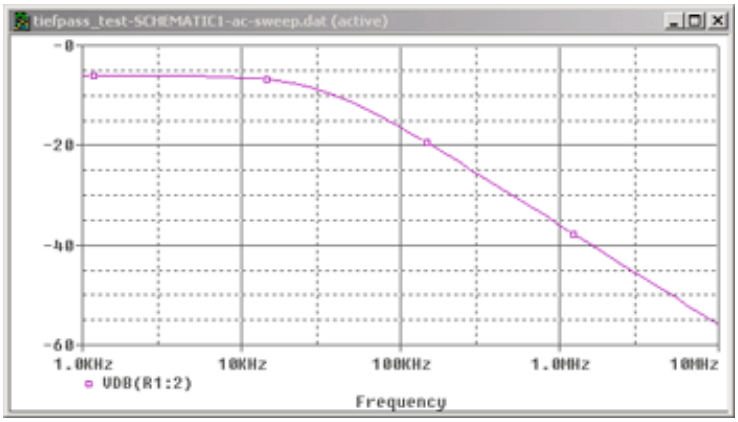
Nach dem Speichern der .LIB-Datei müssen noch die Eigenschaften des Bauteils über ***Edit / Part*** und dann Doppelklick auf das Bauteil geändert werden. Anschließend wird über ***New*** für jedes zu übergebende Element ein Eintrag mit einem bestimmten Wert erstellt. Weiters wird für jeden neu erstellten Eintrag unter ***Display*** die Option ***Name and Value*** gewählt und mit ***OK*** bestätigt.

3. Ändern des PSpice Templates

Zunächst werden die Eigenschaften durch einen Doppelklick auf das Bauteil geöffnet. Der Eintrag *PSpice Template* muss nun wie folgt modifiziert werden:

X^@REFDES %IN+ %IN- %OUT+ %OUT- @MODEL PARAMS:\n+ R1=@R1 C1=@C1 R2=@R2

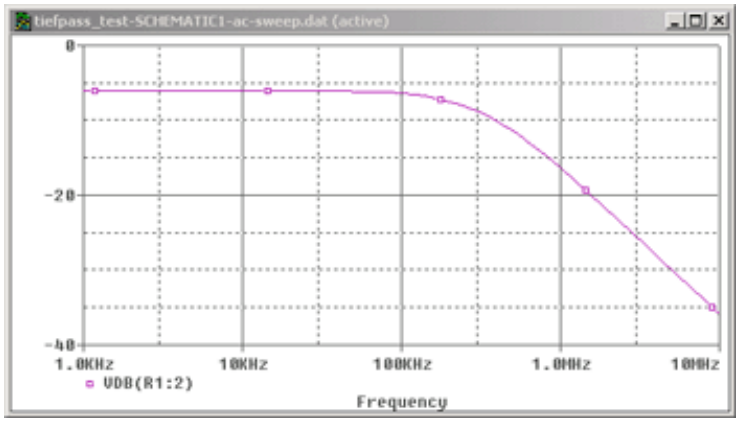


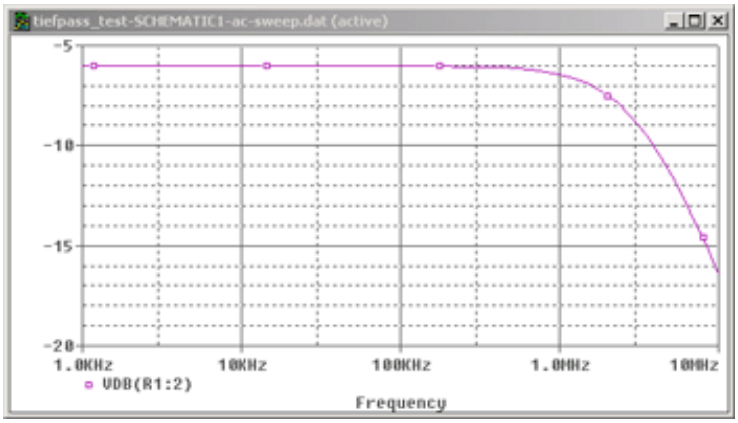


4. Testen der Änderungen

Nun wird die Simulation mit den gleichen Simulationseinstellungen wie vorhin gestartet (***PSpice / Run***). Man erkennt, dass der Knick nun um eine Dekade nach links verschoben ist, da ***C1=10n***.

Wird der Wert von C1 durch einen Doppelklick auf *C1* geändert (***C1=1n***), so erhält man die Darstellung wie vorhin.





Setzt man ***C1=0.1n***, so verschiebt sich der Knick wieder eine Dekade weiter nach rechts.

I. Allgemein

Damit PSpice Ströme und Potenziale in einer Schaltung (el. Netzwerk) überhaupt berechnen kann, muss das elektrische Verhalten von Bauelementen mathematisch implementiert sein. Genau dies leisten die sogenannten ***Modelle***.
Man unterscheidet zwischen elementaren, in PSpice bereits eingebauten (built-in) Modellen und den komplexeren ***Subcircuits***, in denen neben Instanzen von built-in-Modellen auch Quellen und Bauelemente über eine Netzliste zu einer Gruppe zusammengefasst sind (z.B. beim OPV). Subcircuits sind frei definierbar (quasi als "Unterschaltplan"), während bei built-in-Modellen nur noch Parameter verändert werden können.

II. Built-in (eingebaute) Modelle

Syntax:

.MODEL	<model name> [AKO: <reference model name>]
+	<model type>
+	[(<parameter name> = <value> [tolerance specification] ...)]

mit:

+	Zeilenfortsetzungszeichen
[...]	keine "MUSS"-Angabe
<model name>	(Fast) frei zu vergebender Name
<reference model name>	Name des Referenzmodells, auf das durch AKO (A Kind Of) Bezug genommen wird. Das abgeleitete neue Modell erbt die Eigenschaften des Referenzmodells.
<model type>	Bauelementtypus (built-in-Model oder Algorithmus)
<parameter name> = <value>	Angabe der Parameter (aus der Parameterliste) mit Wertzuweisung
[tolerance specification]	Angabe von Toleranzen

III. Subcircuits (Unternetzwerke)

Syntax:

.SUBCKT	<subcircuit name> <nodes>
+	[PARAMS: <Param1> = <Value_Param1> <Param2> = <Value_Param2> ...]
*	[<Kommentar>]
Interne Netzliste zur Beschreibung der Struktur und Funktion des Bauteils	
.ENDS	

mit:

+	Zeilenfortsetzungszeichen
[...]	keine "MUSS"-Angabe
<subcircuit name>	Frei zu vergebender Name
<nodes>	Anschlüsse (Knotennamen) des Unternetzwerkes nach außen hin
[PARAMS: <Param1> = <Value_Param1> ...]	Parameterübergabe an Parameter in der Netzliste
*	Einleiten von Kommentaren

IV. Verwalten von Modellen

Modellbibliotheken

Modelldefinitionen (MODEL's bzw. SUBCKT's) werden in Dateien, den Modellbibliotheken (model libraries), gespeichert. Diese sind reine Text-Files und beinhalten eine oder mehrere Modell-Definition(en).
Typischerweise haben die *Modellbibliotheken* die ***.lib****-Extension*. Die meisten Bibliotheken fassen Modelle ähnlichen Typs zusammen.

Pfade zu Modellbibliotheken

Für die Simulation sucht PSpice die Bibliotheken nach den einzelnen Modelldefinitionen ab. Daher müssen die Pfade zu den Modellbibliotheken spezifiziert und die Bibliotheken im einzelnen genannt werden. Dabei wird auch die Suchreihenfolge festgelegt.

Globale und Design-Modelle / Globale und Design-Bibliotheken

Modelle bzw. Modellbibliotheken sind entweder einem spezifischen Design oder global zugänglich.

Design Models sind nur **einem Design** zugeordnet. Der Schematic Editor erzeugt automatisch Design Models, wenn die Modelldefinition einer ***Part***-Instanz modifiziert wird.

Global Models sind **allen Designs**, die sie erzeugen, zugänglich. Der Part Editor erzeugt immer dann automatisch Global Models, wenn ein Part mit einer Modelldefinition erzeugt wird.

Schachtelung von Modellbibliotheken

Neben Modelldefinitionen können Modellbibliotheken auch ***Verweise*** (*Syntax:* ***.LIB***) auf andere PSpice-Modellbibliotheken enthalten.

Beispiel:
Zwei Bibliotheken, *mydiodes.lib* und *myopamps.lib*, sollen immer bei einer PSpice-Simulation gesucht werden.
Dann kann eine weitere, globale Bibliothek mit dem Namen *mymodels.lib* mit folgendem Inhalt erzeugt werden: **.LIB** mydiodes.lib **.LIB** myopamps.lib
Modellbibliotheken, die man erstmalig mit den OrCAD-Programmen installiert hat, sind in einer speziellen Bibliothek, der ***nom.lib*** gelistet.

Template Syntax

I. Allgemein

Das Attribut ***PSpice Template*** legt fest, mit welcher Syntax ein Bauteil in die Simulations-Netzliste (*.NET) eingetragen wird. Dadurch wird eine Übersetzungsvorschrift vom Schaltplan zur Simulations-Netzliste definiert. Jedes Element, das simuliert werden soll, muss über dieses Attribut verfügen. Beim Generieren der PSpice-Netzliste wird das Attribut ***PSpice Template*** abgearbeitet und in die Netzliste übertragen. Folgende Voraussetzungen müssen erfüllt sein:

- Die Pin-Namen in der Template-Zeile müssen mit den Pin-Namen des Bauteiles übereinstimmen

- Die Anzahl und die Reihenfolge der Pins in der Template-Zeile muss mit der durch die Modell-Definition (über **.MODEL** oder **.SUBCKT**) geforderten Anzahl und Reihenfolge übereinstimmen.

- Der erste Buchstabe in der Template-Zeile muss der Kennbuchstabe für das aufzurufende Modell sein (z.B. Q für einen Bipolartransistor).

Eine Template-Zeile enthält normalen Text, der einfach in die Netzliste übertragen wird und die Namen von anderen Bauteil-Attributen. Für die Behandlung der Bauteil-Attribute stehen die Steuerzeichen *@, &, ?, ~* und *#* zur Verfügung. Diese führen zu folgenden Ausgaben in der Netzliste:

Syntax:

@<Attribut>		Wert von <Attribut>, aber Fehler, falls <Attribut> nicht definiert/ohne Wert
&<Attribut>		Wert von <Attribut>, falls <Attribut> definiert
?<Attribut>t...t		Text zwischen t...t, falls <Attribut> definiert
?<Attribut>t1...t1 t2...t2		Text zwischen t1...t1, falls <Attribut> definiert, sonst Text zwischen t2...t2
~<Attribut>t...t		Text zwischen t...t, falls <Attribut> nicht definiert
~<Attribut>t1...t1 t2...t2		Text zwischen t1...t1, falls <Attribut> nicht definiert, sonst Text zwischen t2...t2
#<Attribut>t...t		Text zwischen t...t, falls <Attribut> definiert, der Rest der Template-Zeile wird gelöscht, falls <Attribut> nicht definiert
^-Zeichen		vollständiger hierarchischer Pfad zum Element, das in der Netzliste erscheint
\n		Erzwingt in der Netzliste einen Zeilenumbruch
% (vor Pin-Namen)		Netzname des an den Pin angeschlossenen Netzes

Als Trennzeichen (t, t1, t2) können Kommas (,), Strickpunkte (;), Schrägstriche (/) oder senkrechte Striche (|) benutzt werden. Soll in der Netzliste das Zeichen % erscheinen, so muss dies in der Template-Zeile doppelt (%%) eingetragen werden.

II. Beispiele

Voraussetzung:

In der Schaltung befindet sich ein Widerstand mit der Bezeichnung *R12* und einem Wert von *1kOhm*, der an die Netze *IN* und *OUT* angeschlossen ist.

PSpice Template:

```
R^@REFDES %1 %2 @Value
```

Netzlisten-Eintrag:

```
R_R12 IN OUT 1k
```

Voraussetzung:

In der Schaltung befindet sich eine Spannungsquelle mit der Bezeichnung *V6*, die an die Netze *NETZ1* und *NETZ2* angeschlossen ist, die über ein Attribut *DC* mit dem Wert *5V* verfügt und bei der das Attribut *AC* mit keinem Wert belegt ist.

PSpice Template:

```
V^@REFDES %+ %- ?DC|DC=@DC| ?AC|AC=@AC|
```

Netzlisten-Eintrag:

```
V_V6 NETZ1 NETZ2 DC=5V
```

Voraussetzung:

In der Schaltung befindet sich ein zweipoliges Element *U4*, das über ein Unternetzwerk mit dem Namen *BEISPIEL* definiert ist und mit den Netzen *IN1* und *IN2* verbunden ist. Im Unternetzwerk ist ein Parameter *G* enthalten, dessen Wert 1000 sein soll, wenn nichts anderes vorgegeben wird. Beim Bauteil ist für das Attribut *G* der Wert 1024 definiert.

PSpice Template:

```
X^@REFDES %a %b @MODEL PARAMS: ?G|G=@G| |G=1000|
```

Netzlisten-Eintrag:

```
X_U4 IN1 IN2 BEISPIEL PARAMS: G=1024
```