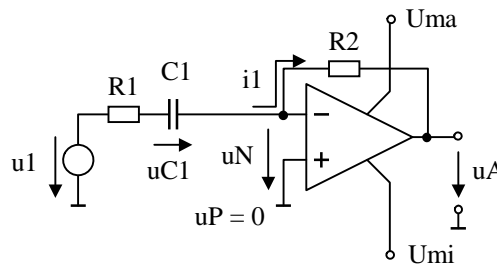


Differenzierer mit Realem Operationsverstärker, simuliert mit Tephys, Simulink, „reinem“ Matlab

Homepage:

<http://www.home.hs-karlsruhe.de/~kero0001>



Modell des „realen“ OP: Die Ausgangs-Spannung u_A ist das Integral der Eingangs-Spannungs-Differenz $u_P - u_N$, multipliziert mit der OP-eigenen Kreisfrequenz ωT . Dies Integral ist aber begrenzt auf die Maximalspannung U_{ma} und die Minimalspannung U_{mi} . Außerdem ist die „Geschwindigkeit“ $v = du_A/dt$ begrenzt auf die „Slew-rate“ v_{ma} (aufwärts) und v_{mi} (abwärts).

In **Tephys** kann man dies in einer **einzigen** Formel ausdrücken:

$u_A = \text{begr}(u_A + \text{begr}(\omega T \cdot (u_P - u_N), v_{ma}, v_{mi}) \cdot dt, U_{ma}, U_{mi})$, hier ist $u_P = 0$

Wie dies in Matlab und Simulink zu formulieren ist, wird weiter unten gezeigt.

Info über die Tephys- Begrenzerfunktion

$y = \text{begr}(x, \text{max}, \text{min})$ bedeutet (in Matlab-Syntax):

$y = x$; **if** $y > \text{max}$, $y = \text{max}$; **elseif** $y < \text{min}$, $y = \text{min}$; **end**

Bei Vernachlässigung des Eingangsstroms in den Minus-Eingang des Op ergibt sich aus dem Knotensatz für den Minus-Eingang $(u_1 - u_{C1} - u_N)/R_1 = (u_N - u_A)/R_2$ und daraus (nach Multiplikation mit $R_1 \cdot R_2$) die „algebraische“ Gleichung für u_N :

$$u_N = ((u_1 - u_{C1}) \cdot R_2 + u_A \cdot R_1) / (R_1 + R_2).$$

Der Strom i_1 durch den Kondensator ist $C_1 \cdot du_{C1}/dt = (u_N - u_A)/R_2$. Daraus folgt der **Algorithmus** zum Lösen dieser **DGL**: $u_{C1} = u_{C1} + (u_N - u_A) \cdot dt / (R_2 \cdot C_1)$. Dieser Algorithmus wird sowohl in Tephys als auch in „reinem“ Matlab verwendet (s. dort).

Simulation mit Tephys:

Man beachte die enorm platzsparende Formulierung von Tephys. „Reines Matlab“ benötigt sehr viel mehr Programmzeilen (s.u.) und außerdem ist die Rechenzeit von Matlab deutlich größer. Simulink ist viel einfacher zu formulieren als „reines Matlab“, s. dort.

----- D:\SI040306\KS_SIMUL\OPNEU\ORIDIF35.TXT -----

```

1| u1 = as*sin(2*pi*f*t)+ad*drei(f*t+del)+ar*sign(drei(f*t+del)) { Eingangsspannung}
2| uN = ((u1-uC1)*R2+uA*R1)/(R1+R2) { uN= Spannung des N-Eingangs gegen Erde}
3| uC1 = uC1+(uN-uA)*dt/(R2*C1) { uC1 = Spannung am Kondensator C1}
4| uA = begr(uA+begr(wT*(-uN),vma,vmi)*dt,Uma,Umi) { uA = Ausgangsspannung real OP}
5| uAid = begr(-R2*C1*(u1-u1alt)/dt,Uma,Umi) { uAid = Ausgangsspannung des idealen OP}
6| u1alt = u1 { u1al = alter Wert von u1, für Berechnung von uAid in Zeile 5}
7| Rap = 2*sqrt(R2/(wT*C1))-1/(wT*C1) { Rap = Wert für R1 beim aperiodischen Grenfall}
8| t = t+dt { dt nach Genauigkeits-Ansprüchen wählen}
    
```

----- **Kommentar** zum File D:\SI040306\KS_SIMUL\OPNEU\ORIDIF35.TXT -----

Differenzierer mit Op, real und ideal, Heft R19 S.97 Apr 97, R55 11.11.04

Modell: $u_A = \text{begr}(u_A + \omega T \cdot (u_P - u_N) \cdot dt, U_{ma}, U_{mi})$

. Typ TL081: $\omega T = 20e6$ /sec, $v_{ma} = 8.7e6$ Volt/sec, $v_{mi} = -10$ Volt/sec

. Typ 741: $\omega T = 5e6$, $v_{ma} = 0.7e6$

Gleichzeitige Berechnung idealer OP (u_{Aid}) und realer Op (u_A)

. . fuer aperiod. Grenfall: $R_1 = 2 \cdot \sqrt{R_2 / (\omega T \cdot C_1)} - 1 / (\omega T \cdot C_1)$

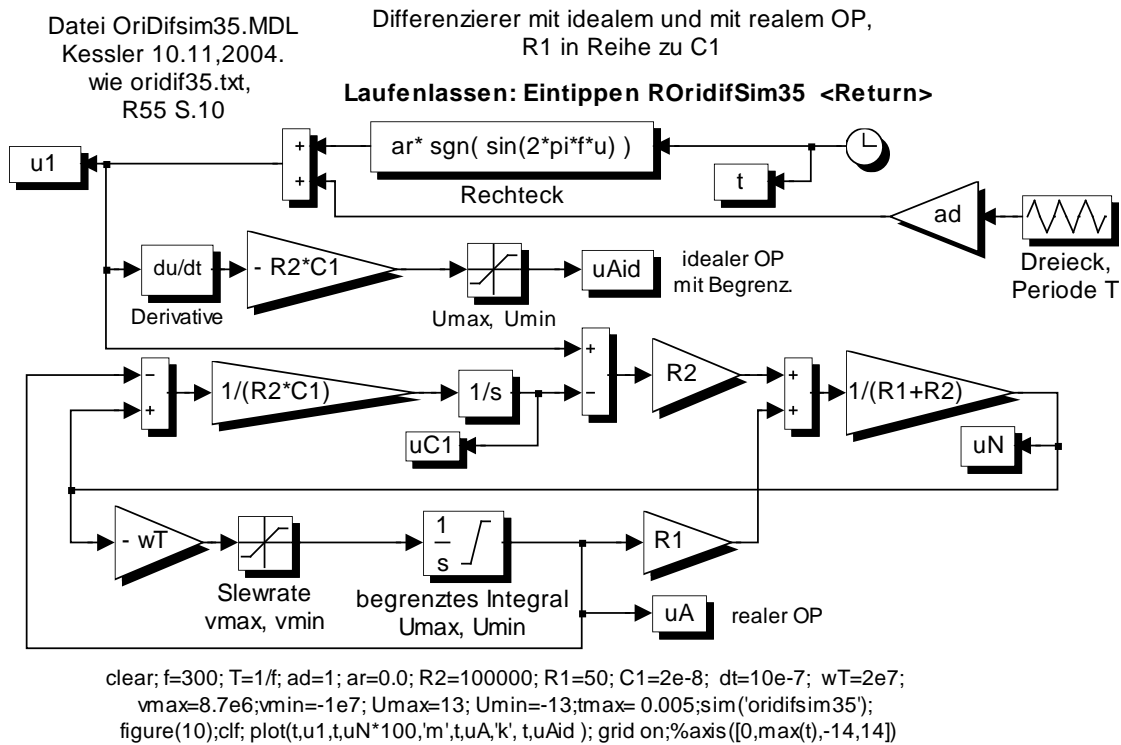
Für "optimales" Differenzieren R_1 anpassen bis "aperiodischer" Grenfall

für u_A bei $u_1 = \text{Dreieck}$.

Bei idealem OP "numerisches Differenzieren" -->

. $u_{Aid} = \text{begr}(-R_2 \cdot C_1 \cdot (u_1 - u_{1alt}) / dt, U_{ma}, U_{mi})$ mit u_{1alt} alter Wert von u_1

Simulation mit Simulink:



Zum Laufenlassen obiger Simulink-Datei: In die Kommandoebene eintippen **ROridifsim35 <Return>**
 Bei Bedarf die Parameter ändern, Datei abspeichern und wieder starten. Oder für jeden Parametersatz einen anderen Dateinamen wählen.

% Datei ROridifsim35.m

% zum Laufenlassen der Simulink-Datei Oridifsim35.mdl

```

format compact; % verhindert unnötige Leerzeilen
clear;
f=300; T=1/f; ad=1; ar=0; R2=100000;
R1=50;
% R1=800;
C1=2e-8;
dt=10e-7; wT=2e7; vmax=8.7e6; vmin=-1e7; % Werte für TL081
Umax=13; Umin=-13; tmax= 0.005; sim('oridifsim35');
Rlap=2*sqrt(R2/(wT*C1))-1/(wT*C1)
bild=11;
figure(bild); clf; plot(t,u1,t,uN*100,'m',t,uA,'k',t,uAid,'b'); grid on;
%axis([0,max(t),-14,14])

S1=['dt=',num2str(dt)]; S2=['tmax=',num2str(tmax)];
S4=['ad=',num2str(ad)];
S6=['freq=',num2str(f)]; S7=['R1=',num2str(R1)];
S8=['R2=',num2str(R2)]; S9=['C1=',num2str(C1)];
% Parameter aufs Bild bei Fadenkreuz:
gtext([S1,S2,S4,S6,S7,S8,S9]);
  
```

Simulation mit „reinem Matlab“:

Der Tephys-Algorithmus wird “wörtlich” in Matlab übersetzt.

Aber der Matlab-Text ist wesentlich umständlicher als der Tephys-Text, selbst wenn man alle Kommentare weglässt.

Man beachte besonders die Kommentare über die Initialisierung der Ausgabevektoren (hier die Zeile UN=t; UA=t; UAid=t;). Das ist enorm wichtig für die Rechenzeit

```
function oridif35(dt, tmax, Bild,as,ad,asaeg,freq,del, R1, R2, C1);
% function oridif35(dt, tmax, Bild,as,ad,asaeg,freq,del, R1, R2, C1);
% Aufruf z.B.so:      oridif35(1e-7,0.005, 1 , 0, 1, 0 ,300,-
0.25,50,1e5,2e-8);
% plot(t,u1,'k',t,UAid,'k', t,UA,'k', t,UN*fN,'k'); grid;
% Kessler 10.11.2004
%
% Realer OP als Differenzierer
% Schaltung: Spannungsquelle u1(Sinus,Dreieck,Rechteck,Sägezahn)
% über R1 und C1 mit Eingang N verbunden, von N mit R2 nach Ausgang
%
% OP-Modell(Tephys): uA = begr(uA+begr(wT*(uP-uN),vma,vmi)*dt,Uma,Umi);
% also begrenzte Geschwindigkeit (slew-rate) und begrenzter Ausgang
%
% Theorie ergibt für aperiodischen GrenzFall (s. R55 S.10)
% R1 für aperiodischenGrenzfall: R1(aper)= 2*sqrt(R2/(wT * C1)) -1/(wT*C1)

tic; % Stoppuhr startet
%----- Parameter -----%      wT,vma, vmi gelte für den OP TL081
ar = 0;
wT =      2.0000E+0007; % wT = wichtigste "Realität" des OP
vma =      8.7000E+0006; % vma,vmi =slew-rate, NICHT im Aufruf
vmi =      -1.0000E+0007;
Uma =      13.0000000000; % Uma, Umi Betriebsspannungen
Umi =      -13.0000000000;
%      wT=0.5e7; vma=0.1e6; vmi =-0.1e7; % Werte für OP 741
uP=0; f=freq;

t=0:dt:tmax; % Zeitvektor
drei=drei2(dt,tmax,0,f,del,1); % Holen des Vektors drei (Dreieckfunktion)
saegl=saeg(0,dt,tmax,f,asaeg); % Holen des Vektors saegl (Sägezahn)
u1 = as*sin(2*pi*f*t)+ad*drei+ar*sign(drei)+saegl; % Eingangsfunktion u1

% Initialisieren der Ausgabevektoren:
UN=t; UA=t; UAid=t;

% Hinweis: Wenn diese Initialisierung der Ausgabevektoren NICHT
% gemacht wird, gibt es NICHT etwa eine Fehlermeldung, sondern das
% Programm benötigt "endlos" lange Zeit.

% Wenn alle drei deklariert sind;
% tmax=      [0.001, 0.002, 0.003, 0.004 ] sec ergab (bei dt = 1e-7)
% Rechenzeit= [ 0.66, 1.26, 1.92, 2.58 ] sec; bei Matlab 5.3
% Also Rechenzeit proportional zu tmax (wie zu erwarten!)
% Zum Vergleich: Tephys braucht bei tmax=0.004 nur 0.87 sec

% Wenn UAid NICHT initialisiert ist (mittels % UAid)
% tmax=      [0.001, 0.002, 0.003, 0.004 ] sec ergab
% Rechenzeit= [ 1.65, 8.56, 23.8, 49.5 ] sec; bei Matlab 5.3
% Also nahezu quadratische Abhängigkeit von tmax

% Wennalle drei Ausgabevektoren NICHT deklariert sind:
% tmax=      [0.001, 0.002, 0.003, 0.004 ] sec ergab
```

```

% Rechenzeit= [ 5.88,    34,    85,   159 ] sec; bei Matlab 5.3
% Also etwa 3 mal so lange wie bei nur 1 nicht deklar. Ausgabevektor

% Startwerte:
uCl=0; uA=0; ulalt=u1(1);

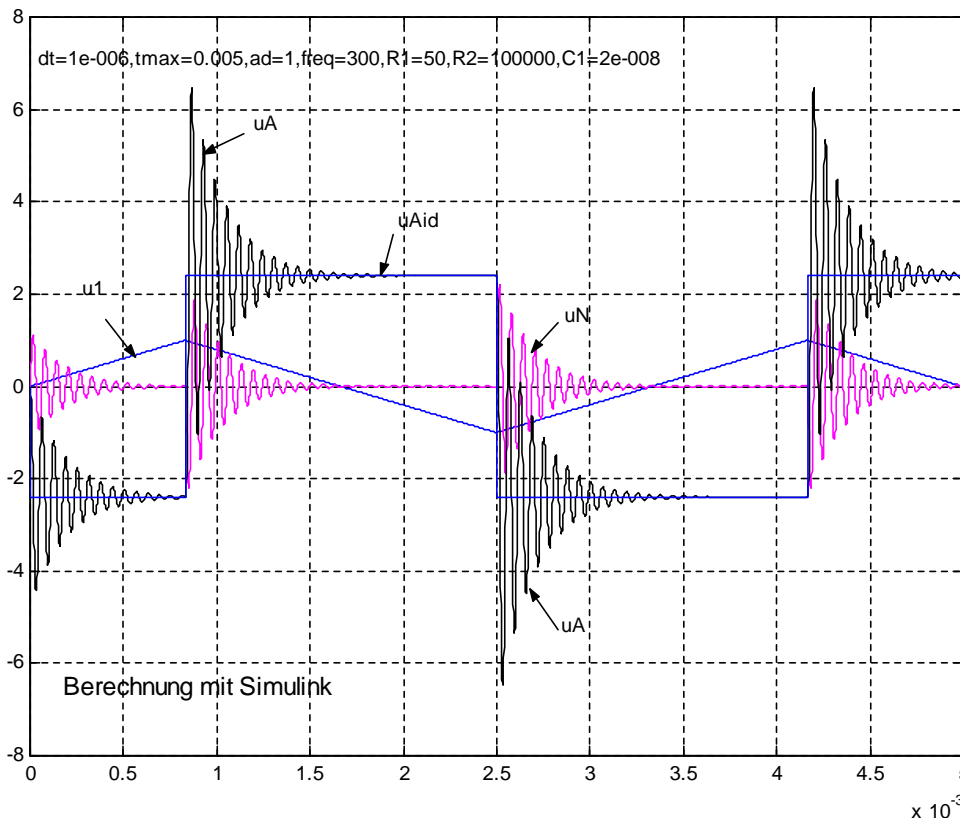
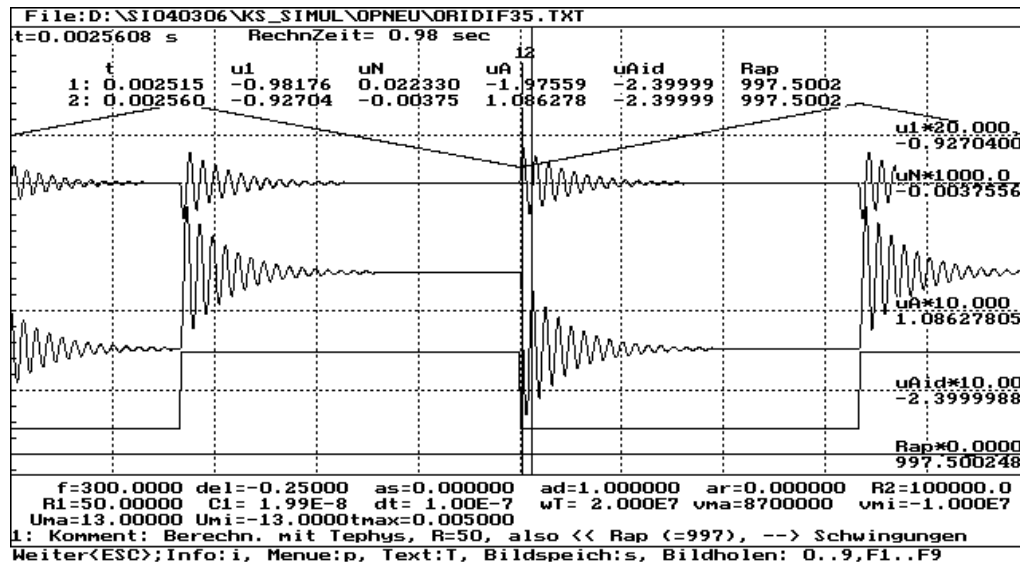
for n=1:length(t), % Lösungsalgorithmus "wörtlich" wie in Tephys
    uN = ((u1(n)-uCl)*R2+uA*R1)/(R1+R2);
    % Tephys: uCl = uCl+(u1(n)-uCl-uN)*dt/(R1*C1);
    uCl = uCl+(uN-uA)*dt/(R2*C1);
    % Tephys: uA = begr(uA+begr(wT*(uP-uN),vma,vmi)*dt,Uma,Umi);
    v = wT*(uP-uN); if v>vma,v=vma; elseif v<vmi, v=vmi;end;
    uA=uA+v*dt; if uA>Uma,uA=Uma; elseif uA<Umi, uA=Umi;end;
    % Tephys: uAid = begr(-R2*C1*(u1-ulalt)/dt,Uma,Umi);
    uAid = -R2*C1*(u1(n)-ulalt)/dt;
    if uAid >Uma, uAid=Uma; elseif uAid < Umi, uAid = Umi; end;
    ulalt=u1(n);
    % Ausgabevektoren füllen: Man beachte die Groß- und Kleinschreibung!
    UAid(n)=uAid; UA(n)=uA; UN(n)=uN;
end; % for n=...
figure(Bild); clf;
if asaeg ~0 fN=10; else fN=100; end;
plot(t,u1,t,UN*fN,'m',t,UA,'k',t,UAid,'b'); grid on;

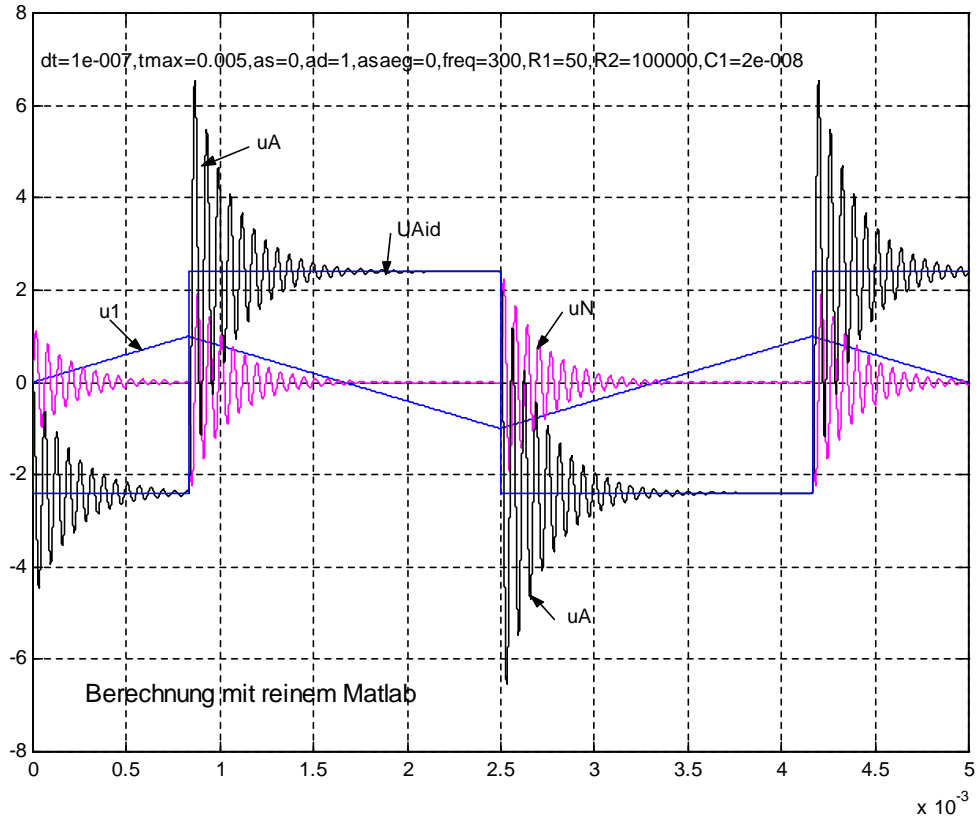
% Parameter zur Ausgabe aufs Bild in Stringvektor schreiben:
S1=['dt=',num2str(dt)];
S2=['tmax=',num2str(tmax) ];
S3=['as=',num2str(as)];
S4=['ad=',num2str(ad)];
S5=['asaeg=',num2str(asaeg)];
S6=['freq=',num2str(freq)];
S7=['R1=',num2str(R1)];
S8=['R2=',num2str(R2)];
S9=['C1=',num2str(C1)];
toc, % elapsed time in sec
% Parameter aufs Bild bei Fadenkreuz:
gtext([S1,S2,S3,S4,S5,S6,S7,S8,S9]);
disp(''),% Zeilenvorschub
disp(['R1(aperiodisch)=2*sqrt(R2/wT*C1)-1/(wT*C1) = ',...
    num2str(2*sqrt(R2/(wT * C1)) -1/(wT*C1) ),' Ohm' ]);
%Die folgende Zeile zur Erleichterung des Funktions-Aufrufs
disp('oridif35(dt, tmax, Bild, as, ad,asaeg, freq,del, R1, R2, C1); ' ),

```

Anschließend drei Bilder mit gleicher Physik, zunächst mit Tephys, dann mit Simulink, dann mit reinem Matlab berechnet. Man erkennt, dass die Bilder das Gleiche liefern.

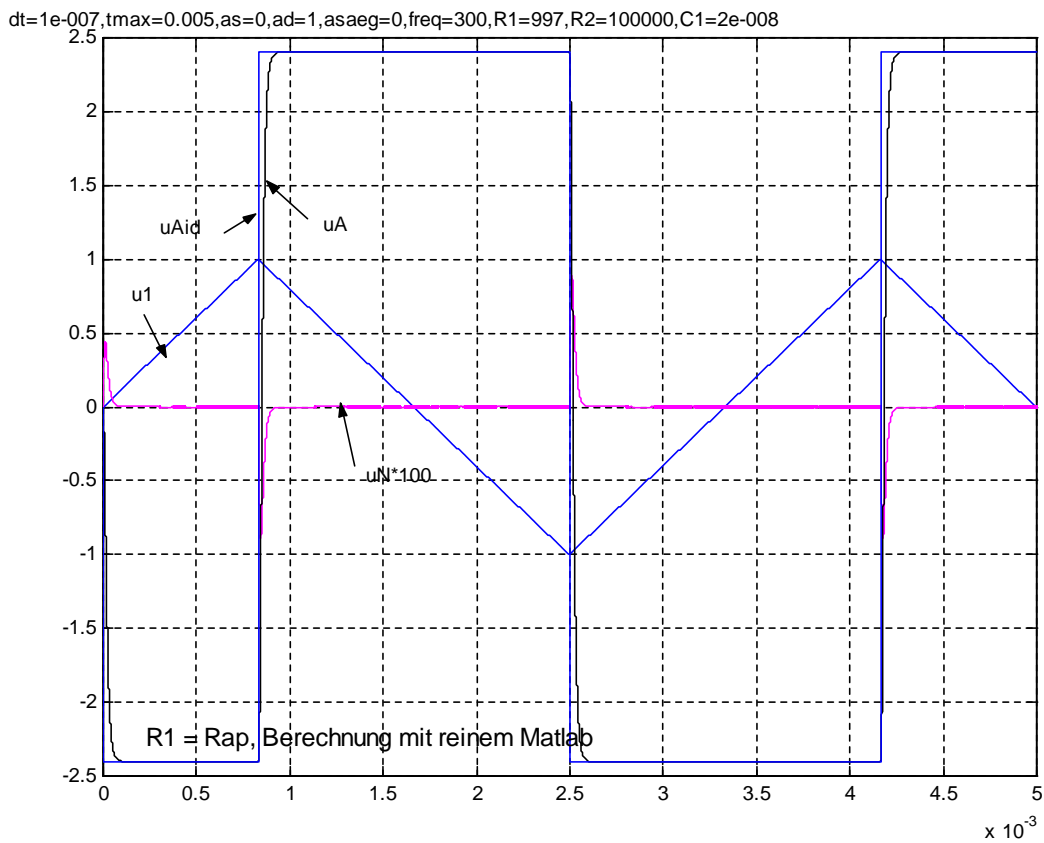
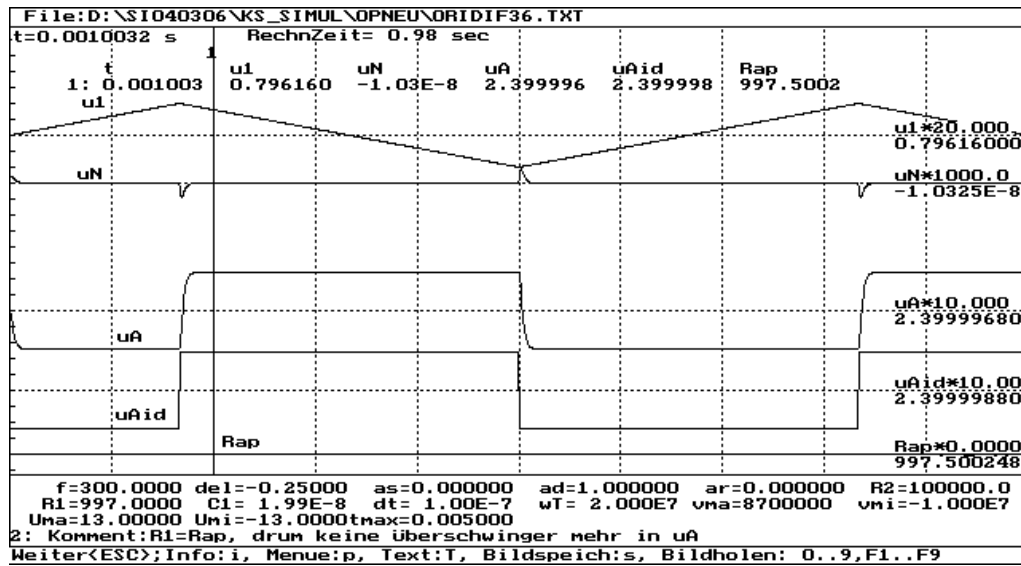
Der Widerstand $R1 = 50 \text{ Ohm}$ (wie üblicherweise bei Verwendung von Funktionsgeneratoren mit deren Ausgangswiderstand 50 Ohm). Im vorliegenden Fall (ωT , $C1$, $R2$) würde sich für den „aperiodischen“ Widerstand ergeben $R_{ap} = 997.5 \text{ Ohm}$ (vgl. die Ergebnisse). Da also hier $R1$ klein gegen R_{ap} ist, resultieren sich sehr deutliche Überschwinger, sowohl in μA also auch in μN . Mit $R1 = 50 \text{ Ohm}$ ist diese Schaltung als „Differenzierer“ also falsch dimensioniert.





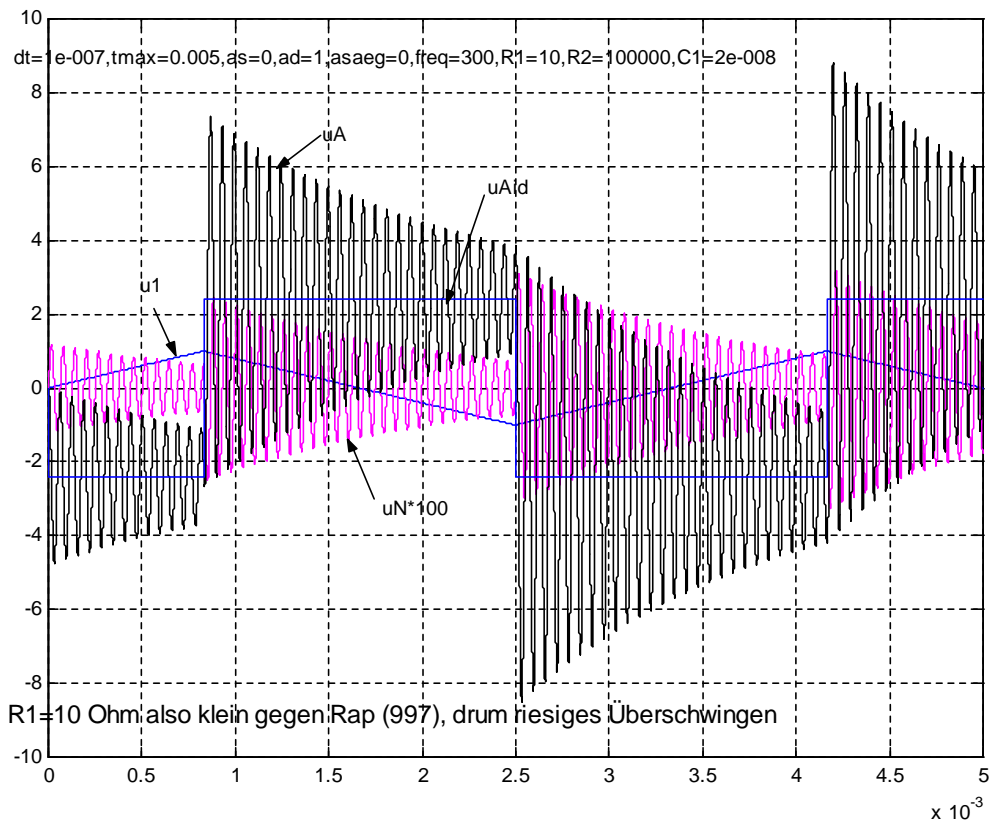
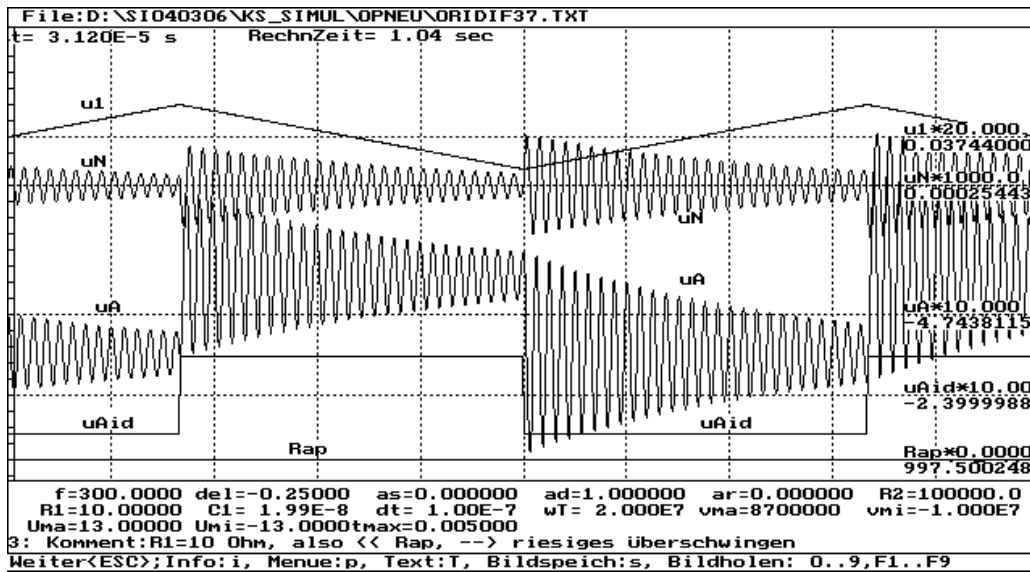
Jetzt R1 = Rap, zunächst Tephys, dann reines Matlab

Man erkennt, dass jetzt die Ausgangsspannung tatsächlich etwa dem „idealen“ Differenzierer entspricht, allerdings ist uA (notwendigerweise!) etwas „verrundet“, statt „eckig“



Jetzt $R1 = 10 \text{ Ohm}$, also $R1 \ll Rap$, zunächst Tephys, dann reines Matlab:

Da hier $R1 \ll Rap$ ist, ergeben sich riesige Überschwinger. Man kann kaum noch erkennen, dass die Schaltung eigentlich ein Differenzierer sein sollte.



Jetzt $R1 = 2 * Rap = 1995 \text{ Ohm}$, zunächst Tephys, dann reines Matlab

Erwartungsgemäß ist bei $R1 > Rap$ der „Kriechfall“ vorhanden, die Ausgangsspannung uA ist also unnötig stark schleichend.

