Time response of an RCR circuit subject to a sinusoidal pulse input (Laplace)

This code successfully ran on Matlab **R2022b** (Linux)

Objectives: Model in state space and in transfer function form (plus initial conditions term) an electrical circuit. Obtain its time response to a single sinusoidal pulse input.

Presentations in video and latest version of materials:

http://personales.upv.es/asala/YT/V/sinpulLEN.html (Laplace transform of sine pulse)

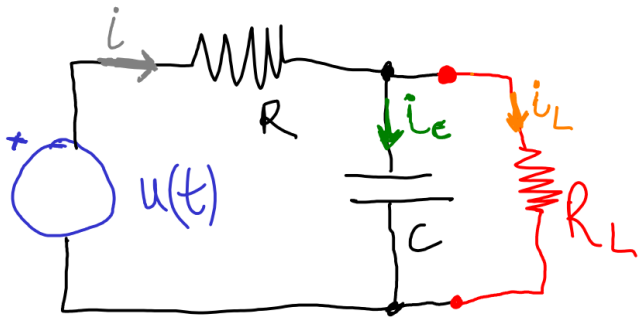
http://personales.upv.es/asala/YT/V/RCRmodEN.html (modelling)

http://personales.upv.es/asala/YT/V/sinpulRCREN.html (1.- superposition)

http://personales.upv.es/asala/YT/V/sinpulRCR2EN.html (2.- piecewise)

Table of Contents

System Modelling (state space and transfer function)



These are the equations governing the above circuit:

$$\frac{dV_c}{dt} = \frac{1}{C} i_c, \quad i_c = i - i_L, \quad i = \frac{1}{R}(u - V_c), \quad i_L = \frac{1}{R_L} V_c$$

Four equations and four unknowns: V_c, i, i_c, i_L so the model seems complete.

If we solve for i_c in the capacitor's equation, we get:

$$\frac{dV_c}{dt} = \frac{1}{C}\left(\cdot \frac{1}{R}(u - V_c) - \frac{1}{R_L} V_c \right)$$

so we may write normalised linear state and output equations as:

$$\frac{dV_c}{dt} = -\left(\frac{1}{RC} + \frac{1}{R_L C} \right) \cdot V_c + \frac{1}{RC} u$$

$$y = V_c$$

*This case study is inspired on a capacitive filter in a rectifier, but it is NOT the model of such rectifier, because rectifiers have diodes that force current to be non-negative, so we should change the third equation to $i = \max\left(0, \frac{1}{R}(u - V_c)\right)$, which is non-linear so, in principle, an straightforward linear/Laplace solution is not available...

Let us give the parameters some numerical values:

```
R=0.5;RL=20;C=2e-2;
A=-1/(R*C)-1/(RL*C)
```

```
A = -102.5000
```

```
B=1/(R*C); C=1; D=0; %"ss"
```

The Laplace domain model will be:

$$Y(s) = (C(sI - A)^{-1}B + D) \cdot U(s) + C(sI - A)^{-1} \cdot x(0)$$

```
syms s
FdT=simplifyFraction(C*1/(s-A)*B+D)
```

```
FdT =
```

$$\frac{200}{2\,s + 205}$$

```
TCI=simplifyFraction(C*1/(s-A)) %this multiplies initial capacitor voltage
V_c(0)
```

```
TCI =
```

$$\frac{2}{2\,s+205}$$

```
t_est_98=log(0.02)/A  %transient duration with "98%" criterion.
```

```
t_est_98 = 0.0382
```

Single sinusoidal pulse and its Laplace transform

Time-domain input pulse

The sinusoidal pulse (a single semiperiod π/ω) is:

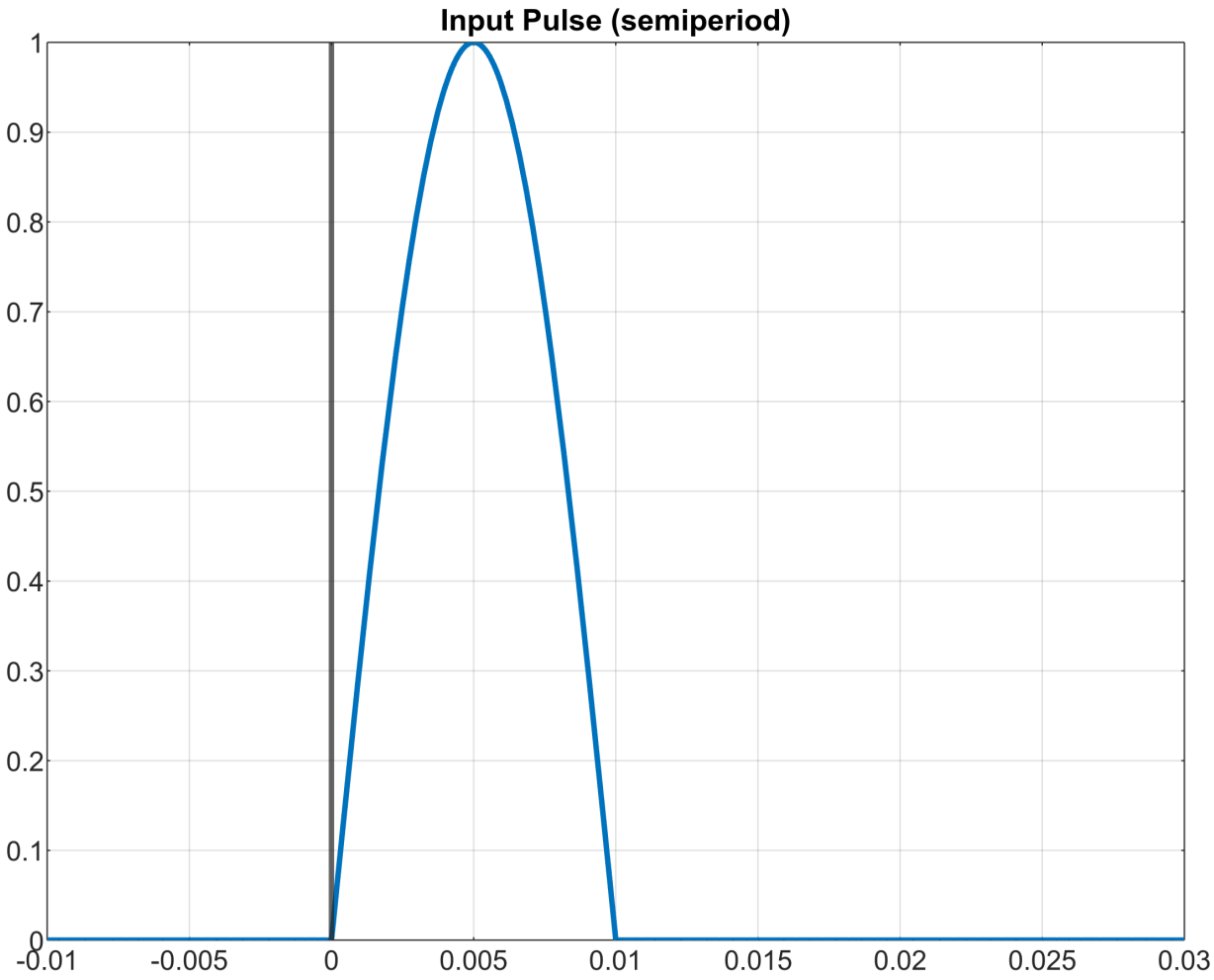
$$u(t)=\begin{cases} 0 & t<0 \\ \sin(\omega t) & 0\le t\le \dfrac{\pi}{\omega} \\ 0 & t>\dfrac{\pi}{\omega} \end{cases}$$

Numerically, we'll work with a single 50Hz semi-period pulse.

```
omega=100*pi;
SemiPeriod=pi/omega
```

```
SemiPeriod = 0.0100
```

```
syms t real
u(t)= heaviside(t)*heaviside(SemiPeriod-t)*sin(omega*t);
fplot(u,[-0.01 0.03],LineWidth=2), grid on, title("Input Pulse
(semiperiod)"),xline(0,LineWidth=2)
```



Pulse in the Laplace domain

Matlab's symbolic toolbox knows how to do it:

```
laplace(u)
```

```
ans =
```

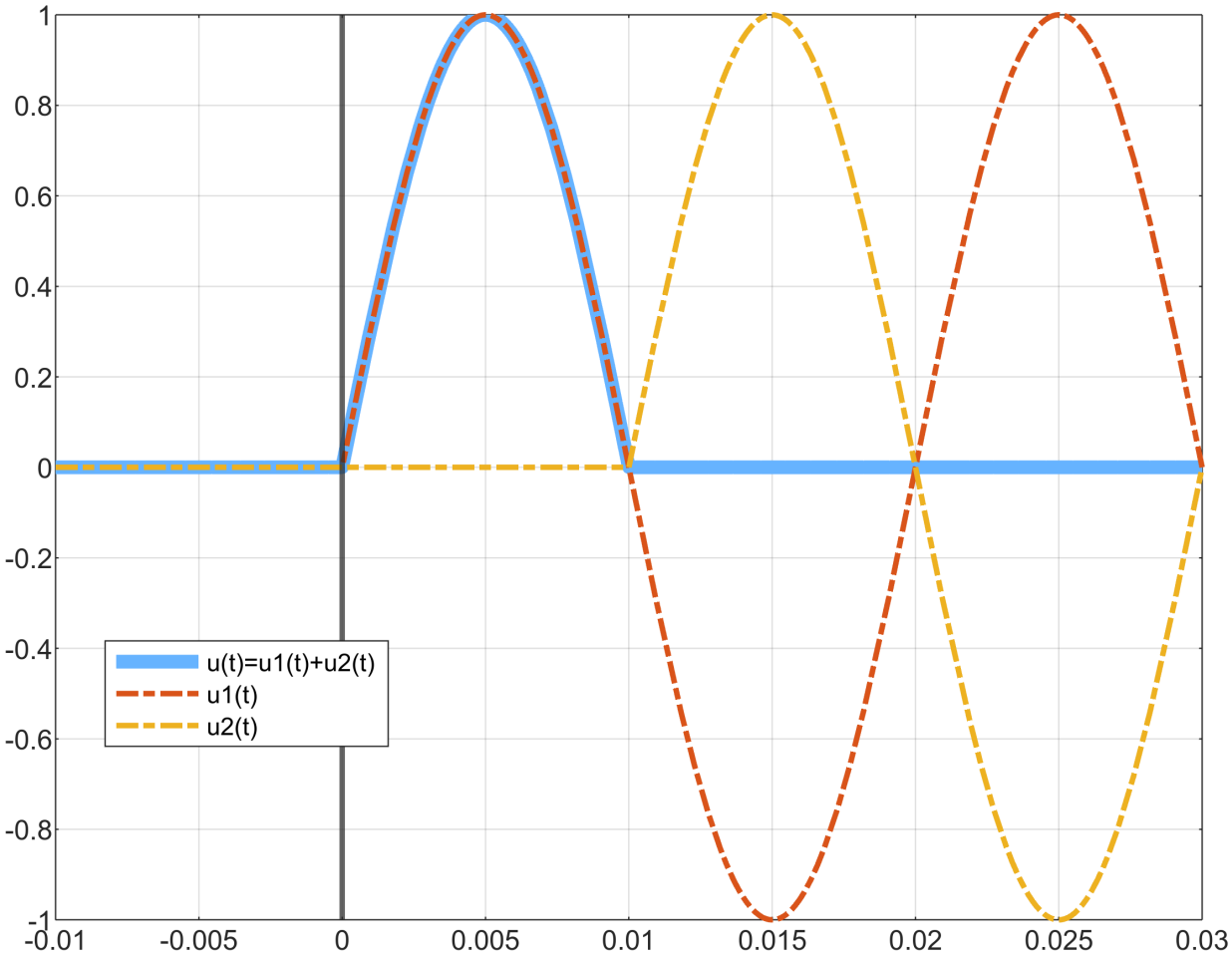
$$\frac{100\,\pi\left(\mathrm{e}^{-\frac{s}{100}}+1\right)}{s^2+10000\,\pi^2}$$

Laplace Transform via superposition and delay (recommended)

Input $u(t)$ is the sum of two waveforms:

```
u1(t)=heaviside(t)*sin(omega*t); %sine from t>=0
u2(t)=subs(u1(t),t,t-SemiPeriod); %delayed sine, from t>=0.01
fplot(u1+u2,[-0.01 0.03],LineWidth=5,Color=[.4 .7 1]), hold on
```

```
fplot([u1,u2],[-0.01 0.03],LineStyle='-.',LineWidth=2), grid on, hold off,
xline(0,LineWidth=2)
legend("u(t)=u1(t)+u2(t)","u1(t)","u2(t)",Location="best")
```




```
U1s=laplace(u1) %a sinusoidal wave starting at t=0   omega/(s^2+omega^2)
```

```
U1s =
```

$$\frac{100\,\pi}{s^2 + 10000\,\pi^2}$$

```
U2s=laplace(u2) %delayed wave, multiplying by exp(-d*s) the Laplace table
entry for sine.
```

```
U2s =
```

$$\frac{100\,\pi\,\mathrm{e}^{-\frac{s}{100}}}{s^2+10000\,\pi^2}$$

The sum of U1s and U2s yields the exact same result as Matlab's toolbox produced, of course.

NOTE: Although we have calculated $u(s)$ because it is a "typical" question in Laplace Transform exams, as we realised that the input is a superposition of simple (delayed) signals, it is NOT necessary to use the above Laplace Transform in the output calculation, since, due to linearity and time invariance, the output to this sinusoidal pulse can be computed from the output Y1 to a sinusoid "with infinite duration", by superimposing Y1 with its delayed version, as done with the input (details below).

Direct computation from Laplace transform's definition (not recommended)

$$U(s)=\int_0^{\infty}u(t)e^{-st}\,dt=\int_0^{0.01}\sin(100\pi\cdot t)\cdot e^{-st}\,dt$$

Integrating by parts:

$$U(s)=\int_0^{0.01}\sin(100\pi\cdot t)\cdot e^{-st}\,dt=e^{-st}\cdot(-\frac{\cos(100\pi t)}{100\pi})|_0^{0.01}-\int_0^{0.01}-\frac{\cos(100\pi\cdot t)}{100\pi}\cdot(-se^{-st})\,dt=\frac{1}{100\pi}\cdot(1$$

The second expression:

$$V(s)=\int_0^{0.01}\cos(100\pi\cdot t)\cdot e^{-st}\,dt=e^{-st}\frac{\sin(100\pi t)}{100\pi}|_0^{0.01}-\int_0^{0.01}\frac{\sin(100\pi\cdot t)}{100\pi}\cdot(-se^{-st})\,dt=0+\frac{s}{100\pi}U(s)$$

Thus,

$$U(s)=\frac{1}{100\pi}\cdot(1+e^{-0.01s})-\frac{s^2}{100^2\pi^2}U(s)$$

Hence, multiplying both sides by $(100\pi)^2$ we get:

$$(100\pi)^2 U(s) = 100\pi \cdot (1 + e^{-0.01s}) - s^2 U(s)$$

and, finally:

$$U(s) = \frac{100\pi \cdot (1 + e^{-0.01s})}{s^2 + (100\pi)^2}$$

We get the same result.

Of course, Matlab (Symbolic Toolbox) also knows how to integrate by parts:

```
syms s
int(sin(omega*t)*exp(-s*t),t,0,SemiPeriod)
```

```
ans =
```

$$\frac{100\,\pi\,\left(\mathrm{e}^{-\frac{s}{100}}+1\right)}{s^2+10000\,\pi^2}$$

Time response, method 1 (superposition)

Not using U(s), just with the output to a pure sine input $U_1(s)$, null initial conditions

```
Y1=FdT*U1s %response to sinusoidal t=0 to infinity, zero i.c.
```

```
Y1 =
```

$$\frac{20000\,\pi}{(2\,s+205)\,(s^2+10000\,\pi^2)}$$

```
partfrac(Y1,FactorMode='real')
```

```
ans =
```

$$\frac{\frac{164000\,\pi}{1600\,\pi^2+1681}-\frac{1600\,\pi\,s}{1600\,\pi^2+1681}}{s^2+10000\,\pi^2}+\frac{3200\,\pi}{(2\,s+205)\,(1600\,\pi^2+1681)}$$

```
vpa(partfrac(Y1,FactorMode='real'),5)
```

```
ans =
```

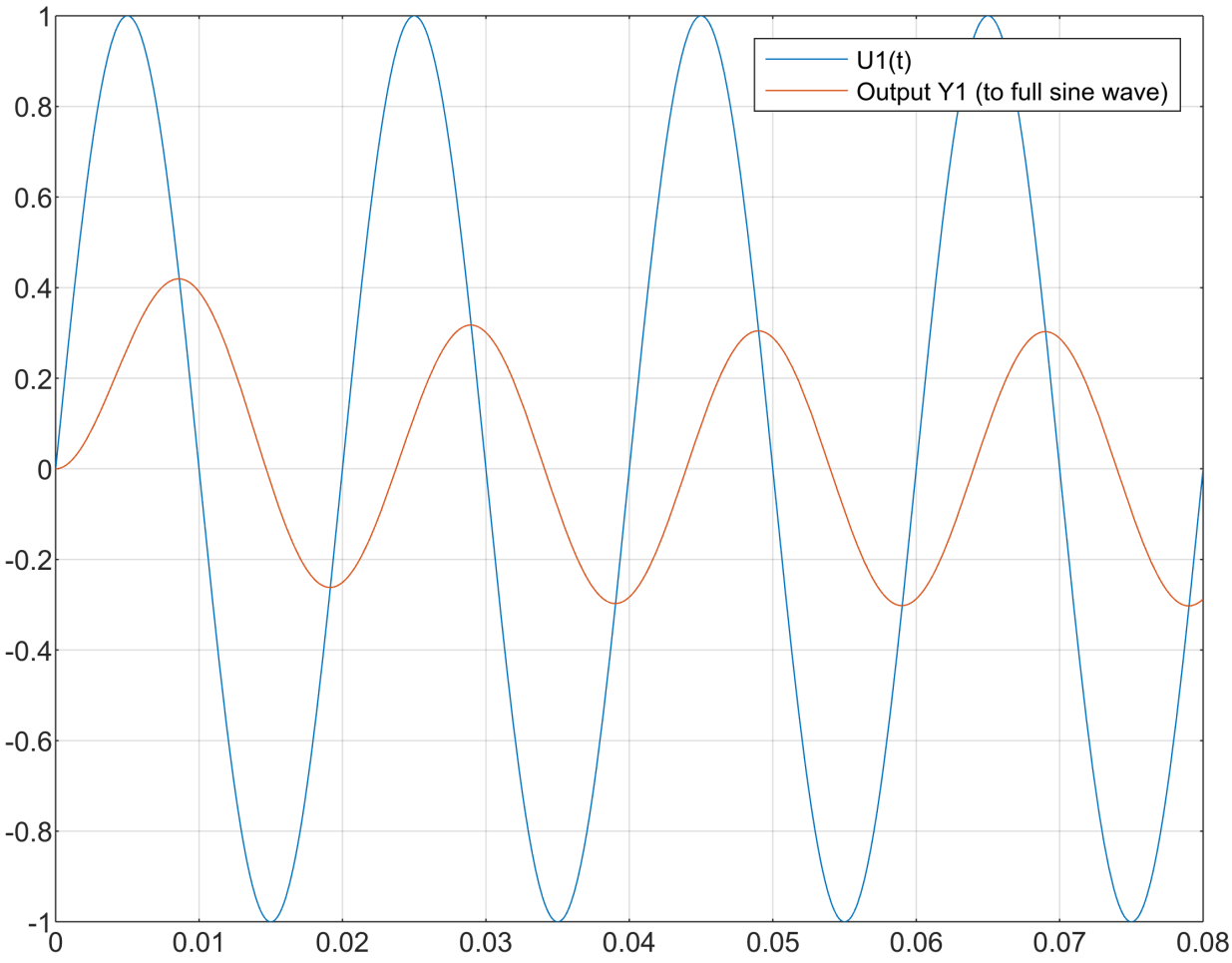
$$\frac{0.57537}{2.0\,s+205.0}-\frac{0.28769\,s-29.488}{s^2+98696.0}$$

```
Y1_t=vpa(ilaplace(Y1),5)
```

```
Y1_t =
```

$$0.093862\sin(314.16\,t)-0.28769\cos(314.16\,t)+0.28769\,\mathrm{e}^{-102.5\,t}$$

```
fplot([u1;Y1_t],[0 0.08]), grid on, legend("U1(t)","Output Y1 (to full sine wave)")
```



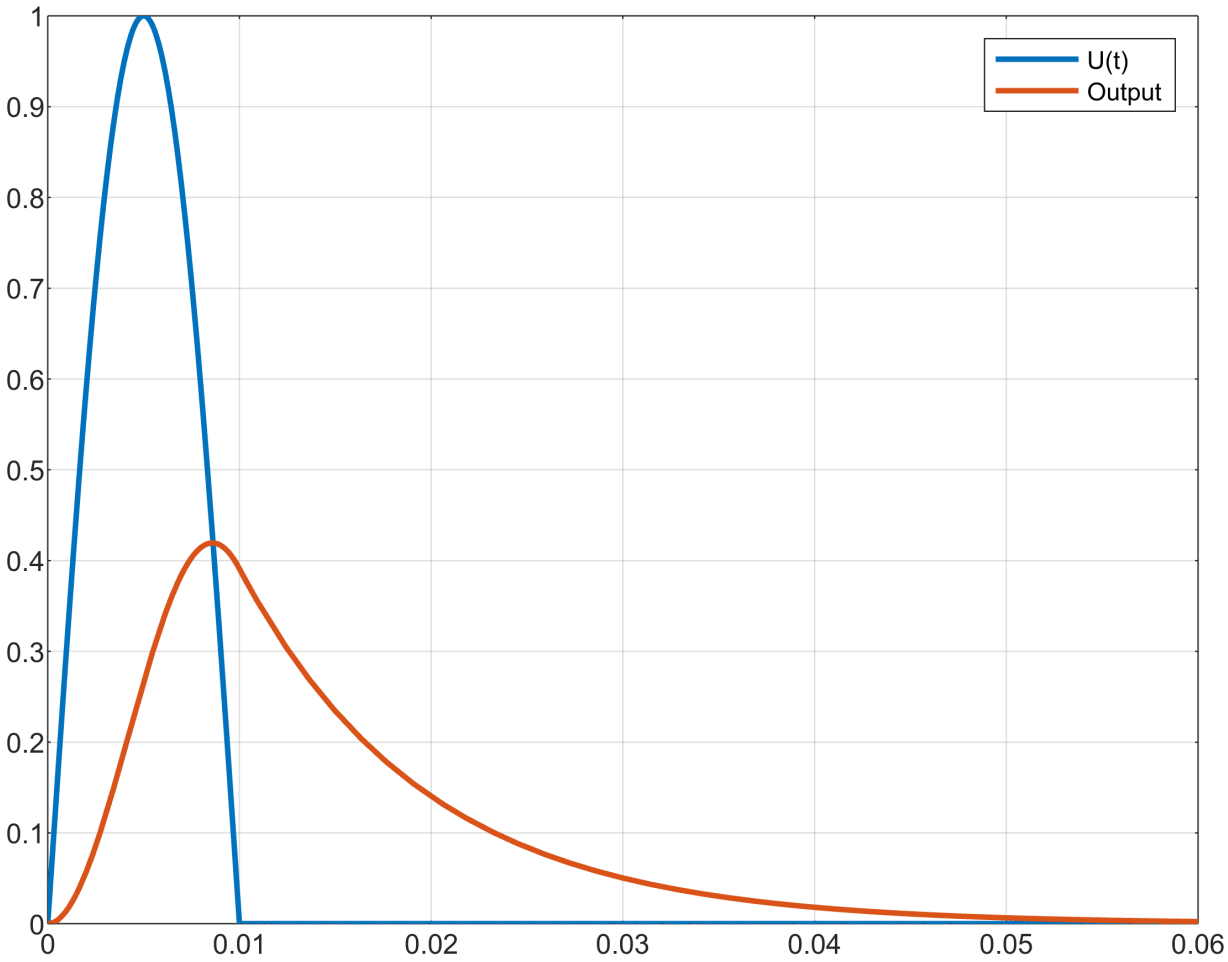
With y1, as the output to input u2 (which is, actually, a delayed u1) is the same y1, but delayed, the output to the sinusoidal pulse will be:

$$y(t) = y_1(t) + \begin{cases} 0 & t < 0.01 \\ y_1(t-0.01) & t \geq 0.01 \end{cases}$$

```
Y_t_pulse=Y1_t+subs(heaviside(t)*Y1_t,t,t-0.01);
vpa(Y_t_pulse,4)
```

ans = $0.09386\sin(314.2\,t) - 0.2877\cos(314.2\,t) + 0.2877\,e^{-102.5\,t} + \text{heaviside}(1.0\,t - 0.01)\,(0.2877\,e^{1.025-102.5\,t}$.

```
fplot([u;Y_t_pulse],[0 0.06],LineWidth=2), grid on,legend("U(t)","Output")
```



Response to nonzero initial conditions

Initial capacitor voltage:

```
V0=0.95;
```

Using superposition, we'll just add the free response and that's all:

```
resp_libre_Laplace=TCI*V0
```

```
resp_libre_Laplace =
```

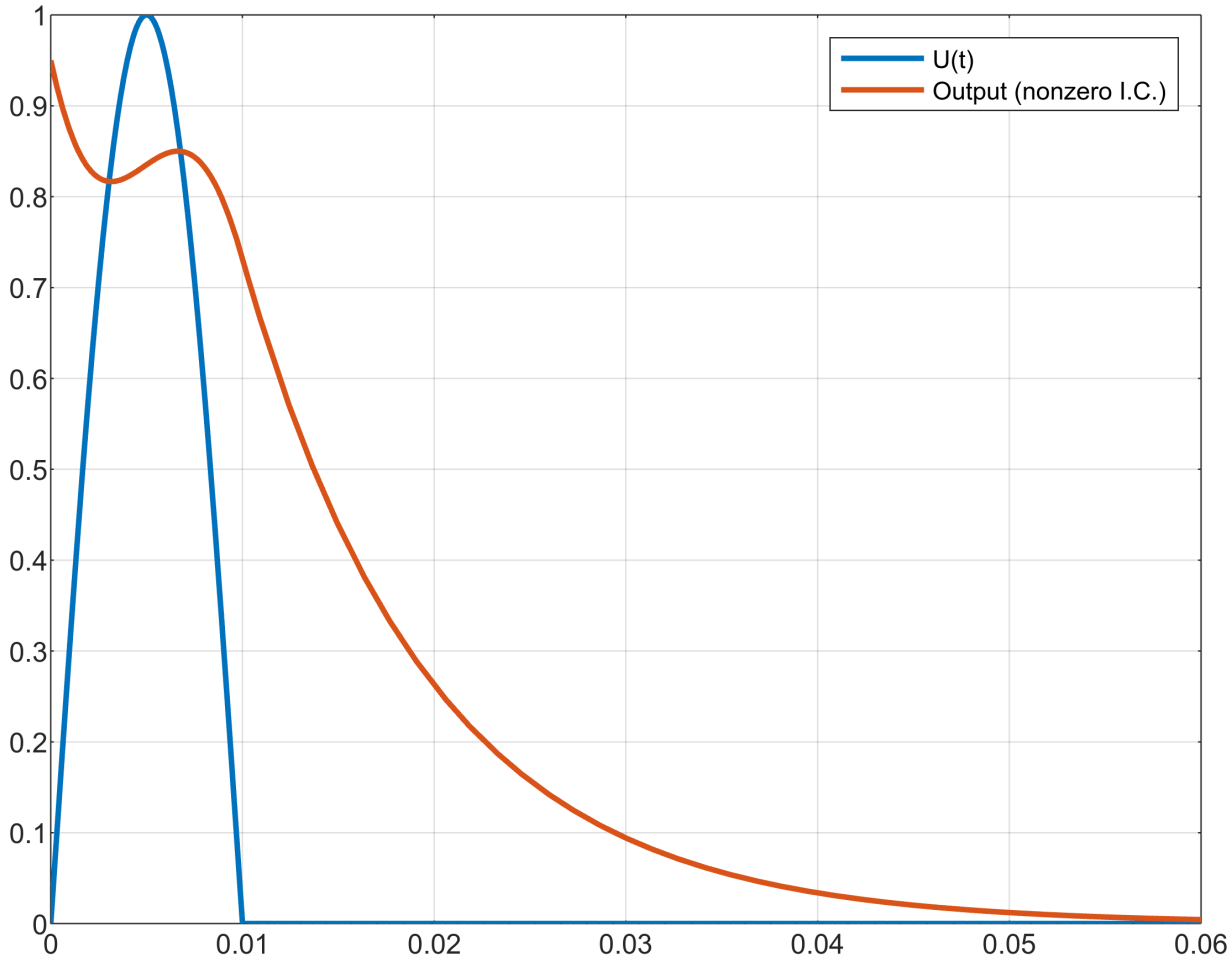
$$\frac{19}{10\ (2\,s+205)}$$

```
resp_libre_t=ilaplace(resp_libre_Laplace)
```

```
resp_libre_t =
```

$$\frac{19\,e^{-\frac{205\,t}{2}}}{20}$$

```
Y_t_conCI=Y_t_pulse+resp_libre_t;
fplot([u;Y_t_conCI],[0 0.06],LineWidth=2), grid on,legend("U(t)","Output
(nonzero I.C.)")
```



Time response, method 2: piecewise computations

The time response to the whole sinusoid (infinite duration) and given initial conditions will be identical to the pulse response we are seeking, as causal systems cannot depend on "future" input waveform features:

```
Y1=FdT*U1s+TCI*V0;
%time response to sinusoid with nonzero initial conditions V0.
vpa(partfrac(Y1,FactorMode='real'),5)
```

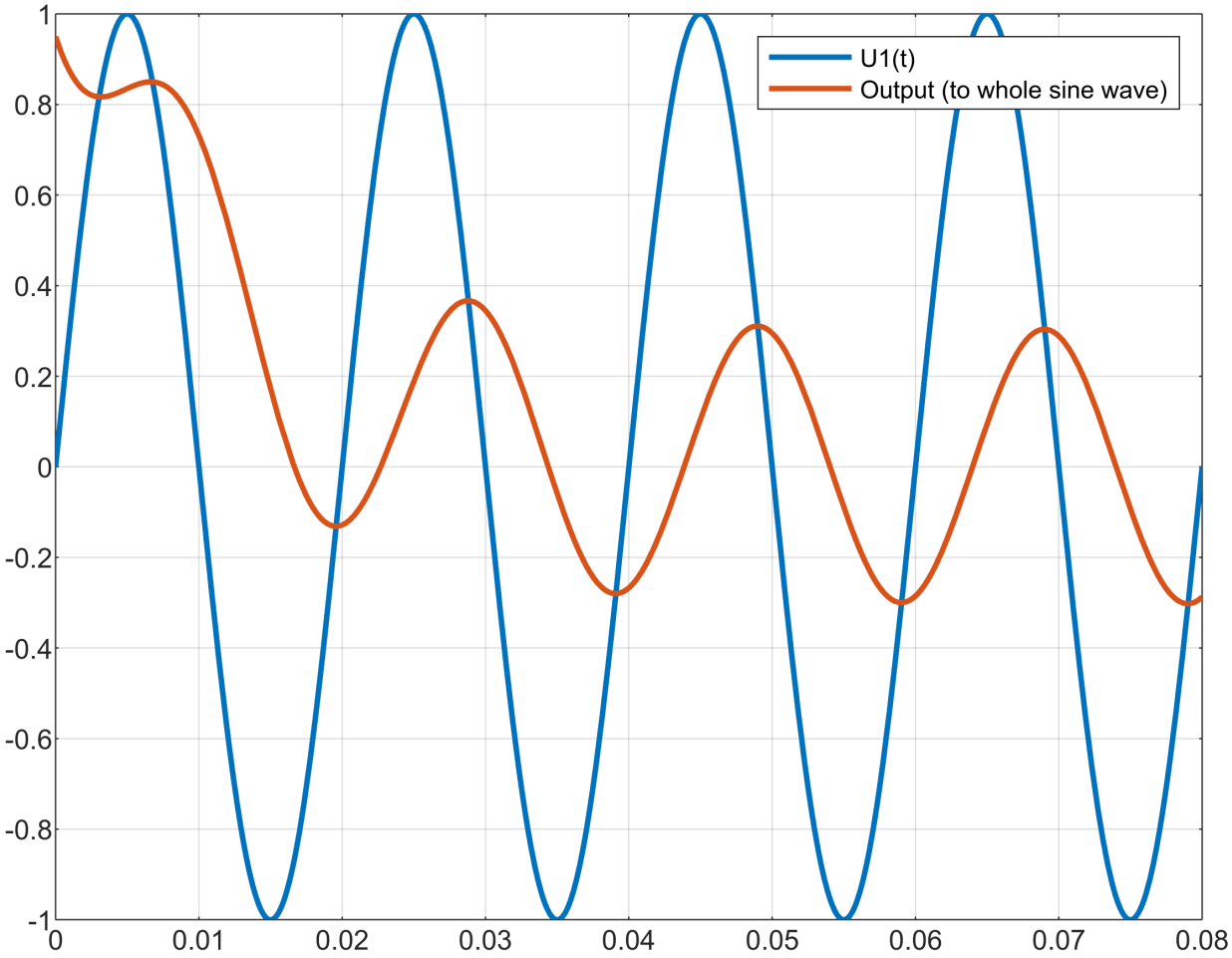
```
ans =
```

$$\frac{2.4754}{2.0\,s + 205.0} - \frac{0.28769\,s - 29.488}{s^2 + 98696.0}$$

```
Y1_t(t)=ilaplace(Y1); vpa(Y1_t,4)
```

ans(t) = $0.09386\sin(314.2\,t) - 0.2877\cos(314.2\,t) + 1.238\,e^{-102.5\,t}$

```
fplot([u1;Y1_t],[0 0.08],LineWidth=2), grid on, legend("U1(t)","Output (to
whole sine wave)")
```



As we have a change in the input "formula" at $t = 0.01$, switching to "zero", the above solution is only valid in the first 0.01 seconds, being meaningless from that time onwards. For $t \geq 0.01$ we will have a "free" response (zero input) from the initial conditions that the first semi-period have left the system at:

```
CondIniFragment2=vpa(Y1_t(0.01),6)
```

CondIniFragment2 = 0.731763

Obviously, if we were simulating a system with order >1, we would need the output and a certain number of its derivatives at $t = 0.01$, or the whole state at that instant (not just the output). In this case, for simplicity, we don't need that because it's fisrt order stuff.

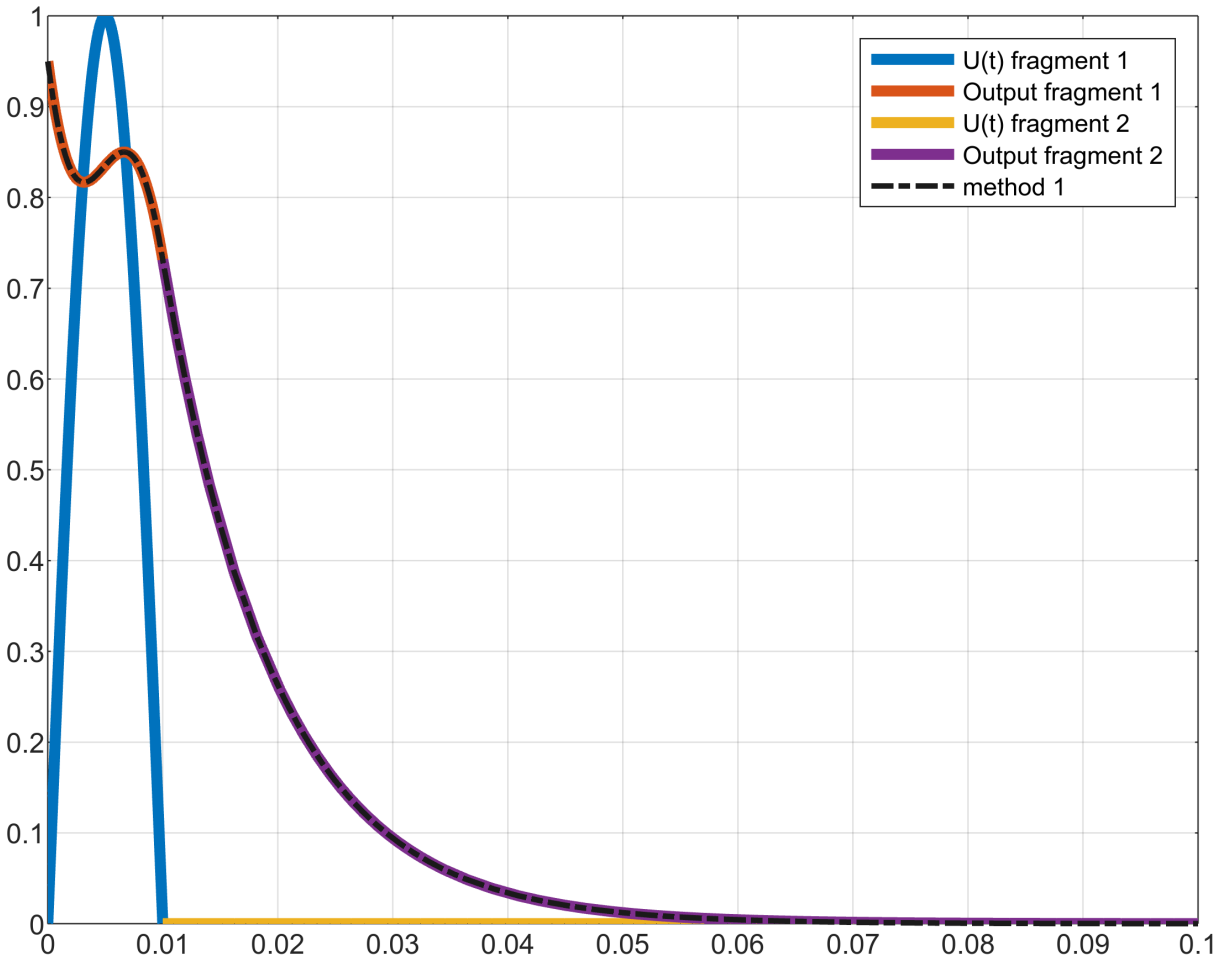
The free response with "initial" conditions equal to the "final" ones of the previous fragment is:

```
syms t_new %new time variable
Y_t_Fragment2(t_new)=ilaplace(TCI*CondIniFragment2,t_new);
vpa(Y_t_Fragment2,5)
```

ans(t_new) = $0.73176\,e^{-102.5\,t_{new}}$

But, beware, actually "zero" of this new clock is 0.01 of the "official" timing; so we need to be careful when writing/plotting what we just obtained:

```
fplot([u1;Y1_t],[0 0.01],LineWidth=4), hold on
fplot([0;Y_t_Fragment2(t-0.01)],[0.01 0.1],LineWidth=4),
fplot(Y_t_conCI,[0 0.1], Color=[.1 .1 .1],LineStyle="-.",LineWidth=2)
%solution method 1
hold off, grid on
legend("U(t) fragment 1","Output fragment 1","U(t) fragment 2","Output
fragment 2","method 1")
```



Of course, both methods give the same solution, as expected.