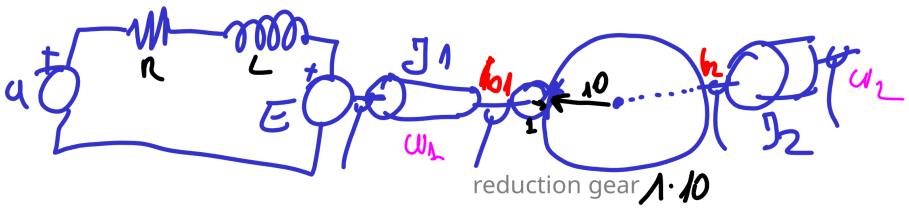
Analysis and simulation of DC motor system driving a mechanical load

© 2025, Antonio Sala Piqueras, Universitat Politècnica de València, Spain. All rights reserved.

Objective: understand the properties and the step response of a separately excited DC motor model driving a rotating load, as well as these from a reduced-order model L=0.



Presentations in video:

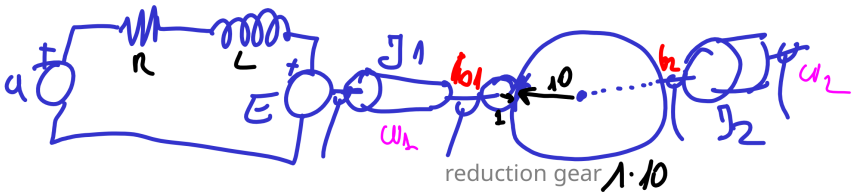
*use https://personales.upv.es/asala/YT/V/[videocode].html if links below do not work.

motstallEN	DC motor analysis: stalled dynamics (electric only)
motgeartfEN	DC motor with gearbox: transfer function representation
motssEN	DC motor with gears: steady state (equilibrium) and stationary DC efficiency analysis
motstepss	Respuesta escalón con exponencial de matriz: ejemplo motor CC [ESPAÑOL]
motstepssEN	Step response via matrix exponential: DC motor example
motcctfstepEN	DC motor with gears: step response, current limitation tweaks
motccredstepEN	DC motor, reduced model: comparison with full-order model, step response
motccbodap1	Aproximación sistemas en frecuencia: ejemplo motor eléctrico (1) [ESPAÑOL]
motccbodap2	Aproximación sistemas en frecuencia: simulaciones motor eléctrico (2) [ESPAÑOL]
motccbodap1EN	DC motor case study: frequency domain analysis and comparison with reduced-order model

Table of Contents

Electrical time constant and transient (stalled)....2
Complete electrical+mechanical model in Control Systems Toolbox....4
Transfer function (matrix) representation....5

Steady-state analysis, efficiency....6
Transient analysis: step response....7
1.- Using Control Systems Toolbox "step" command....7
2.- Step response via Transfer Function and Laplace method (Symbolic Math toolbox)...8
3.- Step response from state-space model (matrix exponential formula)....9
Reduced-order model (no electrical dynamics, 1st-order)....10
Step response comparison with full-order model....11
Frequency response comparison with full-order model....12
Start-up with current limit....17



***Modelling recap:**

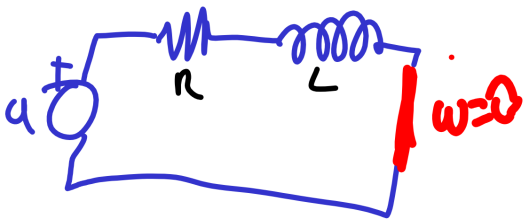
$$\frac{dI}{dt} = \frac{1}{L}\left(u - RI - k\omega_1\right)$$

$$\frac{d\omega_1}{dt} = \frac{1}{J_1 + J_2/N^2}\left(kI - (b_1 + \frac{b_2}{N^2})\omega_1\right)$$

In matrix form:

$$\frac{d}{dt}\begin{pmatrix} I \\ \omega_1 \end{pmatrix} = \begin{pmatrix} -R/L & -k/L \\ +k/J_{eq} & -b_{eq}/J_{eq} \end{pmatrix}\begin{pmatrix} I \\ \omega_1 \end{pmatrix} + \begin{pmatrix} 1/L \\ 0 \end{pmatrix} u$$

Electrical time constant and transient (stalled)



We'll define an operating DC voltage (it's an input, anyway):

```
OperatingVoltage=48; %for later simulations
```

Constant parameters of electrical part

```
R=.3; L=20e-3; %electric circuit parameters [INVENTED!]
StallCurrent=OperatingVoltage/R %this is important
```

```
StallCurrent =
160
```

For constant speed (constant E), the electrical-only part would be:

$$\frac{dI}{dt} = \frac{-R}{L} I + \frac{1}{L}(U - E)$$

StalledMotor=ss(-R/L,1/L,1,0);

This would be governed by $e^{-\frac{R}{L}\cdot t}$ under "stalled" rotor, then. Also (in incremental units), the same dynamics (with different final value) will appear if the mechanical part is driven by a third party at a constant rotational speed.

```
pp=-R/L %electric-only pole
```

```
pp =
-15
```

The time stall current will take to settle will be $e^{-\frac{R}{L}\cdot t_s} \approx 0.05$:

```
electrical_settling_time=fzero(@(t) exp(pp*t)-.05,0)
```

```
electrical_settling_time =
0.1997
```

```
stall_settling_time=log(0.05)/pp
```

```
stall_settling_time =
0.1997
```

```
log(0.05)
```

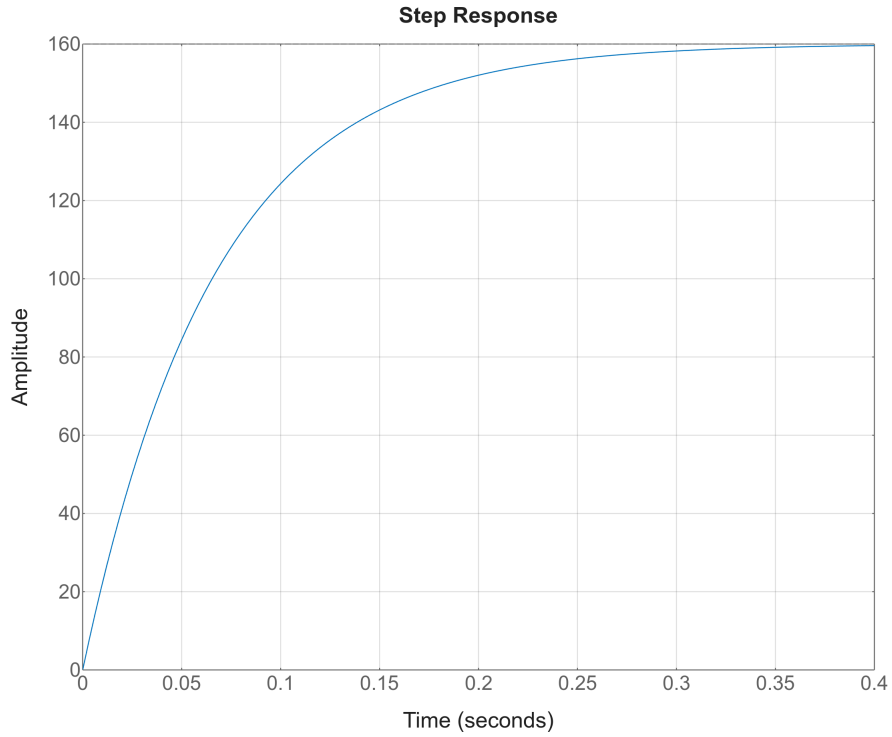
```
ans =
-2.9957
```

```
tau=1/abs(pp) %time constant L/R
```

```
tau =
0.0667
```

Duration of the electrical transient would be, then, around 3 time constants τ.

```
StalledMotor=ss(-R/L,1/L,1,0);
step(StalledMotor*OperatingVoltage), grid on
```



This can be somehow extrapolated to the case when "mechanical time constants are more than 10 times larger than those from the electrical part", so that they don't interfere (i.e., if motor is slowly moving large loads).

Complete electrical+mechanical model in Control Systems Toolbox

We need extra constant parameters for the model

```
k=0.36; %Motor constant
J1=.45; b1=0.13; %motor (fast) side mechanical parameters
N=10; %gear ratio
J2=5; b2=0.6; %slow side mechanical parameters
```

Equivalent inertia and friction coefficients on motor (fast) side:

```
Jeq=(J1+J2/N^2)
```

```
Jeq =
0.5000
```

```
beq=b1+b2/N^2
```

```
beq =
0.1360
```

State equations: $\dot{x} = Ax + Bu$

```
A=[-R/L -k/L;
    +k/Jeq -beq/Jeq];
```

```
B=[1/L; 0];
```

Output equations: $y = Cx + Du$; we wish to plot current and rotational speed, the latter in RPM, noting that 1 rad/s=60 rad/min= $60/(2\pi)$ revolutions per minute.

```
C=[1 0;
   0 60/2/pi];
D=[0;0];
```

Control systems Toolbox object for simulation:

```
sys=ss(A,B,C,D); %state-space object
sys.OutputName={'current (A)','ang speed (rpm)'};
sys.InputName={'Voltage input'};
```

Transfer function (matrix) representation

```
tf(sys)
```

```
ans =

  From input "Voltage input" to output...
                     50 s + 13.6
   current (A):  ---------------------
                 s^2 + 15.27 s + 17.04

                              343.8
   ang speed (rpm):  ---------------------
                     s^2 + 15.27 s + 17.04

Continuous-time transfer function.
Model Properties
```

```
zpk(sys) %factorized fractions (zeros, poles, gains)
```

```
ans =

  From input "Voltage input" to output...
                    50 (s+0.272)
   current (A):  -------------------
                 (s+14.06) (s+1.212)

                            343.77
   ang speed (rpm):  -------------------
                     (s+14.06) (s+1.212)

Continuous-time zero/pole/gain model.
Model Properties
```

In a pen-and-paper exam, you would manually carry out $s \cdot x(s) = Ax(s) + Bu(s)$, $(sI - A) \cdot x(s) = Bu(s)$, $x(s) = (sI - A)^{-1}B \cdot u(s)$, $y(s) = Cx(s) + Du(s)$, thus, using the Symbolic Math toolbox, we get:

```
syms s
TheTf=simplify(C*inv(s*eye(2)-A)*B+D)
```

```
TheTf =
```

$$\begin{pmatrix} \frac{50\ (125\,s+34)}{125\,s^2+1909\,s+2130} \\ \frac{6047748076523898375}{140737488355328\ (125\,s^2+1909\,s+2130)} \end{pmatrix}$$

```
[n,d]=numden(TheTf);
tmp=coeffs(d(1));
vpa((n(1)/tmp(3))/(d(1)/tmp(3)),4)
```

```
ans =
```

$$\frac{50.0\,s+13.6}{s^2+15.27\,s+17.04}$$

```
tmp=coeffs(d(2));
vpa((n(2)/tmp(3))/(d(2)/tmp(3)),4)
```

```
ans =
```

$$\frac{343.8}{s^2+15.27\,s+17.04}$$

Result is coincident with control system toolbox, as expected.

Steady-state analysis, efficiency

The final values per unit volt input (i.e., static or DC gain) are:

```
dcgainmatrix=dcgain(sys)
```

```
dcgainmatrix = 2×1
    0.7981
   20.1746
```

```
dcgainmatrix*48
```

```
ans = 2×1
   38.3099
  968.3794
```

Of course, in paper-and-pencil examinations, this is computed by setting all derivatives to zero, i.e. $0 = Ax_e + Bu_e$, thus $x_e = -A^{-1}Bu_e$, and as $y_e = Cx_e + Du_e$, we get the output equilibrium value by evaluating

```
dcgain2=C*inv(-A)*B+D %coincides with control system toolbox,
obviously
```

```
dcgain2 = 2×1
    0.7981
   20.1746
```

*In transfer function form, the dcgain is obtained by just setting "s=0" (as a corollary of final value theorem) in the transfer function, i.e., **dcgain(G(s))=G(0)**.

Let us give names to each of the two gains:

```
dcgainCurrent=dcgainmatrix(1) %in A/V (sort of "conductance")
```

```
dcgainCurrent =
0.7981
```

```
dcgainOmega1=dcgainmatrix(2) %in rpm/V
```

```
dcgainOmega1 =
20.1746
```

Efficiency computation (per unit volt input) in steady state:

```
InputEqCurrent=dcgainCurrent*OperatingVoltage
```

```
InputEqCurrent =
38.3099
```

```
InputEqPower=InputEqCurrent*OperatingVoltage
```

```
InputEqPower =
1.8389e+03
```

```
EqPowerDissipatedInResistor=InputEqCurrent^2*R
```

```
EqPowerDissipatedInResistor =
440.2936
```

```
MechanicalOutputPower=InputEqPower-EqPowerDissipatedInResistor
```

```
MechanicalOutputPower =
1.3986e+03
```

```
SteadyStateEfficiency=MechanicalOutputPower/InputEqPower
```

```
SteadyStateEfficiency =
0.7606
```

All input power is expended in friction, of course. We need to revert to rad/s for unit consistency

```
omega1finalinradians=dcgainOmega1*OperatingVoltage*2*pi/60
```

```
omega1finalinradians =
101.4085
```

```
TotalFrictionPower=beq*omega1finalinradians^2
```

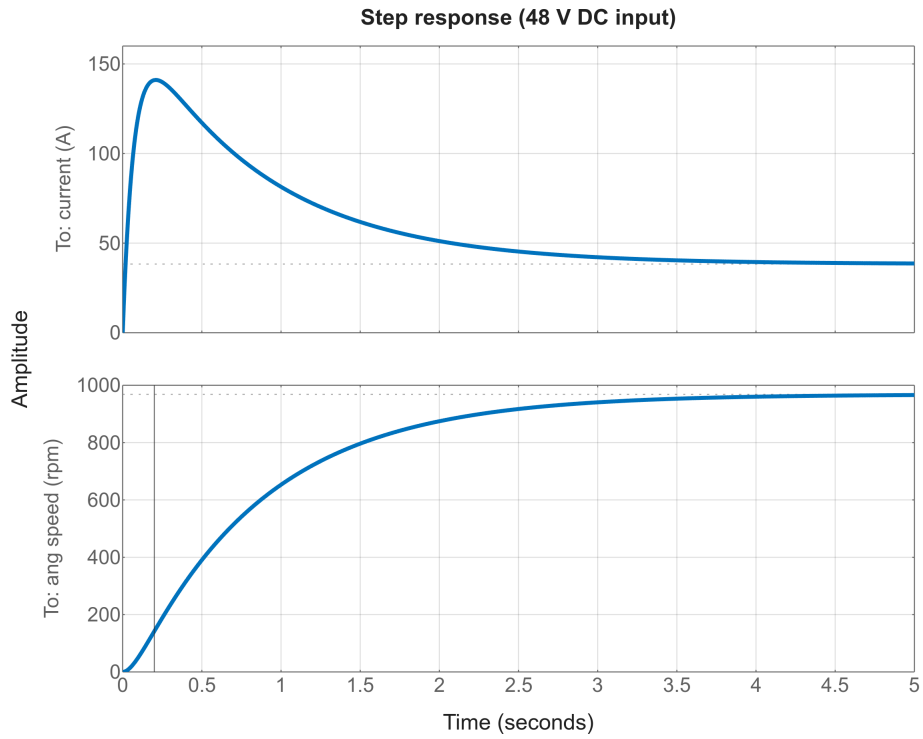
```
TotalFrictionPower =
1.3986e+03
```

Transient analysis: step response

1.- Using Control Systems Toolbox "step" command

```
h=stepplot(sys*OperatingVoltage); %step command would also work,
but we have a handle here
```

```
grid on, title("Step response (48 V DC input)")
h.Responses.LineWidth=2;
xline(electrical_settling_time)
```



2.- Step response via Transfer Function and Laplace method (Symbolic Math toolbox)

If we want the formula for the step response, we need Laplace transform manipulations:

```
Y=TheTf*48/s; %s was a Symbolic Math variable
vpa(Y,4)
```

```
ans =
```

$$\begin{pmatrix} \frac{2400.0\ (125.0\,s+34.0)}{s\ (125.0\,s^2+1909.0\,s+2130.0)} \\ \frac{2.063\text{e+}6}{s\ (125.0\,s^2+1909.0\,s+2130.0)} \end{pmatrix}$$

```
Ypartfrac=vpa((partfrac(Y,'FactorMode','real')),4)
```

```
Ypartfrac =
```

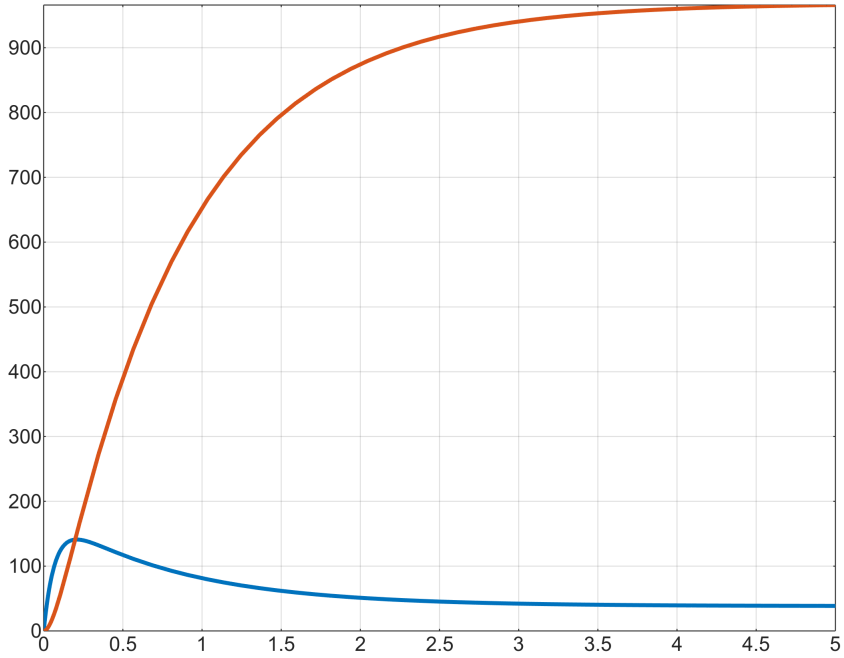
$$\begin{pmatrix} \frac{144.9}{s+1.212}-\frac{183.2}{s+14.06}+\frac{38.31}{s} \\ \frac{91.35}{s+14.06}-\frac{1060.0}{s+1.212}+\frac{968.4}{s} \end{pmatrix}$$

```
syms t real
Ysym(t)=ilaplace(Ypartfrac);
vpa(Ysym(t),4)
```

```
ans =
```

$$\begin{pmatrix} 144.9\,\mathrm{e}^{-1.212\,t}-183.2\,\mathrm{e}^{-14.06\,t}+38.31 \\ 91.35\,\mathrm{e}^{-14.06\,t}-1060.0\,\mathrm{e}^{-1.212\,t}+968.4 \end{pmatrix}$$

```
fplot(Ysym(t),[0 5], LineWidth=2), grid on %coincides with control
system toolbox, of course
```



3.- Step response from state-space model (matrix exponential formula)

Alternatively, constant input is $\dot{u} = 0$, so we can augment the dynamics incorporating that as a third state equation and obtain the free response of the augmented system $\dot{\xi} = \bar{A}\xi$, with $\xi = [x;\ u] = [I;\ \omega;\ u]$ and

$$\bar{A} = \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix}.$$

Such free response is the matrix exponential, i.e., $\xi(t) = e^{\bar{A}\cdot t}\xi(0)$. The output would require multiplying $\psi(t)$ by matrix C (and adding D times the constant input). Of course, this "matrix exponential" way of doing things is not valid for a generic input waveform.

```
syms t real
StepAmplitude=OperatingVoltage;
initpsi=[0;0;StepAmplitude]; %augmented initial conditions
BarA=[A B;[0 0] 0];
AugmentedStateStepResponse=expm(BarA*t)*initpsi;
vpa(AugmentedStateStepResponse,4)
```

```
ans =
```

$$\begin{pmatrix} 144.9\,e^{-1.212\,t} - 183.2\,e^{-14.06\,t} + 38.31 \\ 9.566\,e^{-14.06\,t} - 111.0\,e^{-1.212\,t} + 101.4 \\ 48.0 \end{pmatrix}$$

```
StateStepResponse=AugmentedStateStepResponse(1:2);
vpa(StateStepResponse,4)
```

ans =

$$\begin{pmatrix} 144.9\,e^{-1.212\,t} - 183.2\,e^{-14.06\,t} + 38.31 \\ 9.566\,e^{-14.06\,t} - 111.0\,e^{-1.212\,t} + 101.4 \end{pmatrix}$$

```
vpa([C D]*AugmentedStateStepResponse,4)
```

ans =

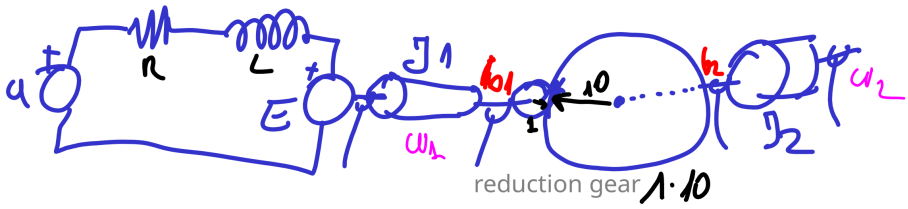
$$\begin{pmatrix} 144.9\,e^{-1.212\,t} - 183.2\,e^{-14.06\,t} + 38.31 \\ 91.35\,e^{-14.06\,t} - 1060.0\,e^{-1.212\,t} + 968.4 \end{pmatrix}$$

```
vpa(C*StateStepResponse+D*StepAmplitude,4)
```

ans =

$$\begin{pmatrix} 144.9\,e^{-1.212\,t} - 183.2\,e^{-14.06\,t} + 38.31 \\ 91.35\,e^{-14.06\,t} - 1060.0\,e^{-1.212\,t} + 968.4 \end{pmatrix}$$

Reduced-order model (no electrical dynamics, 1st-order)



If $L \to 0$, instead of using $\frac{dI}{dt} = \frac{1}{L} V_L$, we will use $L\frac{dI}{dt} = V_L$ to avoid division by zero. Then, the electrical equation ends in $V_L = 0$,

so $u = IR + E = IR + k\omega_1$, $I = \frac{1}{R} \cdot (u - k\omega_1)$.

From the normalised state equations,

$$\frac{d}{dt}\begin{pmatrix} I \\ \omega_1 \end{pmatrix} = \begin{pmatrix} -R/L & -k/L \\ +k/J_{eq} & -b_{eq}/J_{eq} \end{pmatrix}\begin{pmatrix} I \\ \omega_1 \end{pmatrix} + \begin{pmatrix} 1/L \\ 0 \end{pmatrix} u$$

multpliying the first one by L, we get $0 = -RI - k\omega_1 + u$, which ends up in the same value for I.

Replaced in the second state equation, we get a 1st-order model:

$$\frac{d\omega_1}{dt} = \frac{-1}{J_{eq}} \cdot \left(\frac{k^2}{R} + b_{eq}\right) \cdot \omega_1 + \frac{k}{RJ_{eq}} u$$

```
Ared=-1/Jeq*(k^2/R+beq);
```

```
Bred=k/R/Jeq;
Cred=[-k/R; 60/2/pi];
Dred=[1/R;0];
%we keep the same two outputs (current and speed).
sysReduced=ss(Ared,Bred,Cred,Dred);
sysReduced.InputName=sys.InputName;
sysReduced.OutputName=sys.OutputName;
zpk(sysReduced)
```

```
ans =

  From input "Voltage input" to output...
                3.3333 (s+0.272)
   current (A):  ----------------
                   (s+1.136)

                    22.918
   ang speed (rpm):  ---------
                     (s+1.136)

Continuous-time zero/pole/gain model.
Model Properties
```

Matlab can do the thing with "no physics reasoning", with:

```
reducedsys2 = xelim(sys,1,"MatchDC"); %This thing does the same:
sets to zero
% the derivatives of 1st state, without knowing the physical
interpretation
norm(reducedsys2-sysReduced,'inf') %they are the same
```

```
ans =
0
```

We don't need to repeat the "steady-state" analysis (it's the same):

```
dcgain(sys)
```

```
ans = 2×1
    0.7981
   20.1746
```

```
dcgain(sysReduced) %Reduction preserves DC gain
```

```
ans = 2×1
    0.7981
   20.1746
```

Step response comparison with full-order model

The two full-order system poles are:

```
eig(A)' %poles
```

```
ans = 1×2
  -14.0601   -1.2119
```

```
log(0.05)./ans %settling times of transient "modes" 95%
```

```
ans = 1×2
```

```
    0.2131    2.4718
```

The single pole of the reduced-order system is:

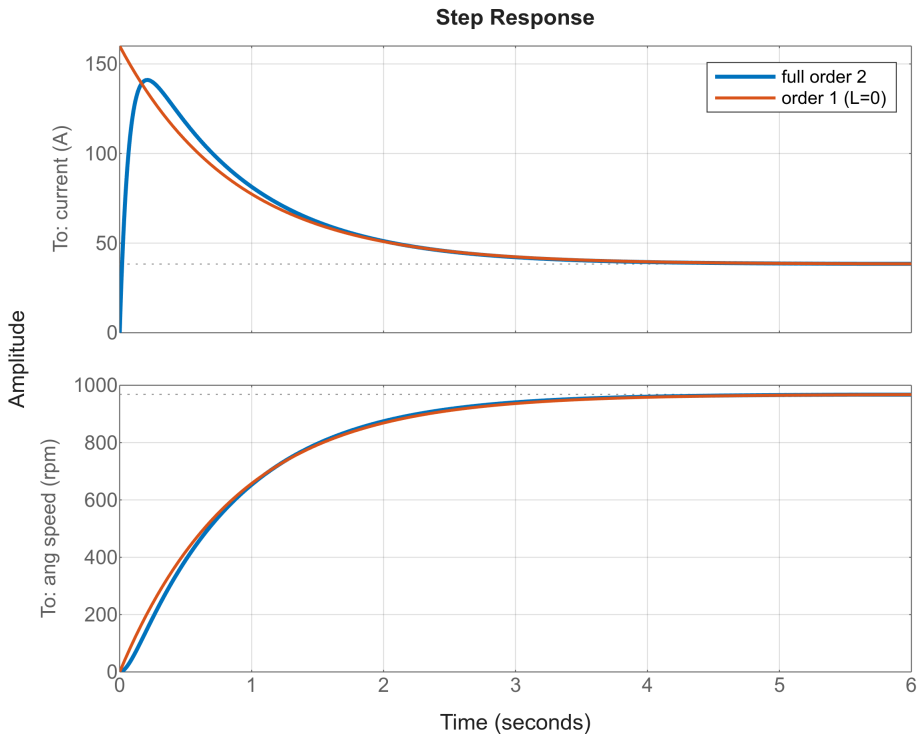
```
eig(Ared) %pole
```

```
ans =
-1.1360
```

```
log(0.05)./ans %settling time
```

```
ans =
2.6371
```

```
h=stepplot(sys*OperatingVoltage,sysReduced*OperatingVoltage);
grid on, legend("full order 2","order 1 (L=0)")
h.Responses(1).LineWidth=2;h.Responses(2).LineWidth=1.5;
```

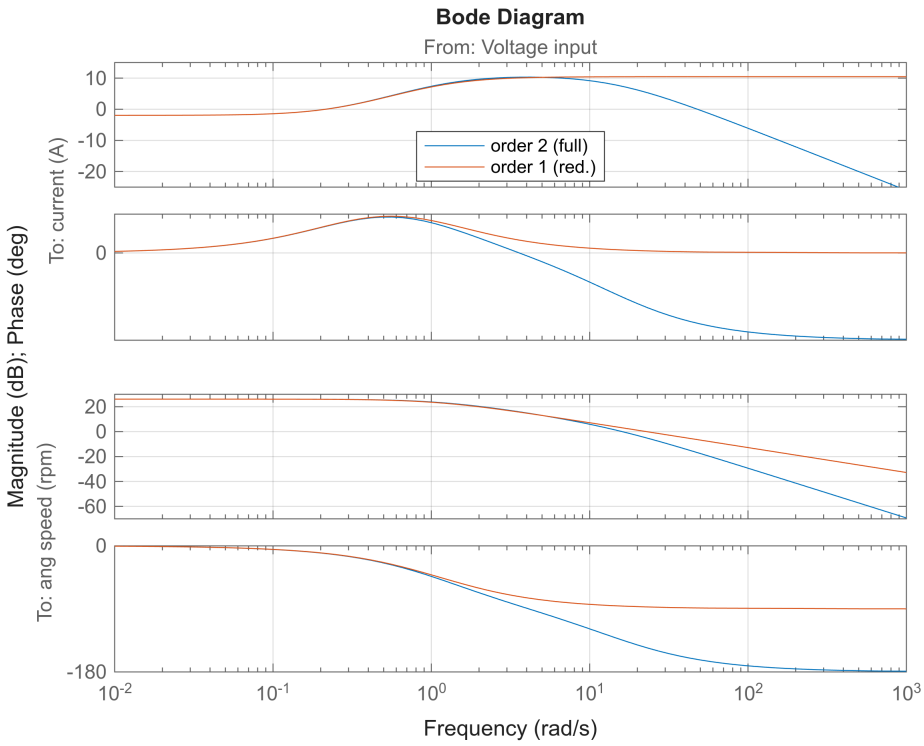


There is some diference at the start if we look at the "electrical output (I)", nothing in the long run and nothing of interest if we look at the mechanical output from a practical point of view. As we are on the "safe side" regarding peak current with the simpler model, we may perfectly work with it, probably, in most applications. Of course, the quality of the reduced-order model would depend on actual value of inductance and inertia... the ones here have been chosen so that 1st-order reduced model is a "good" approximation.

Frequency response comparison with full-order model

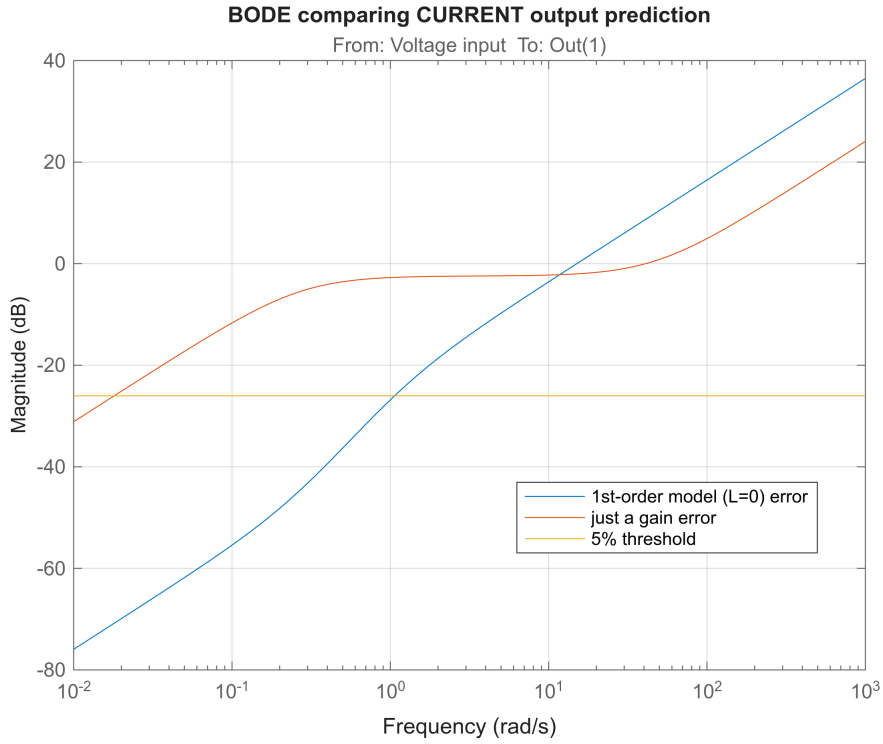
If signals were not so slow, maybe there would be differenes

```
bode(sys,sysReduced), grid on,
legend("order 2 (full)","order 1 (red.)",Location="best")
```

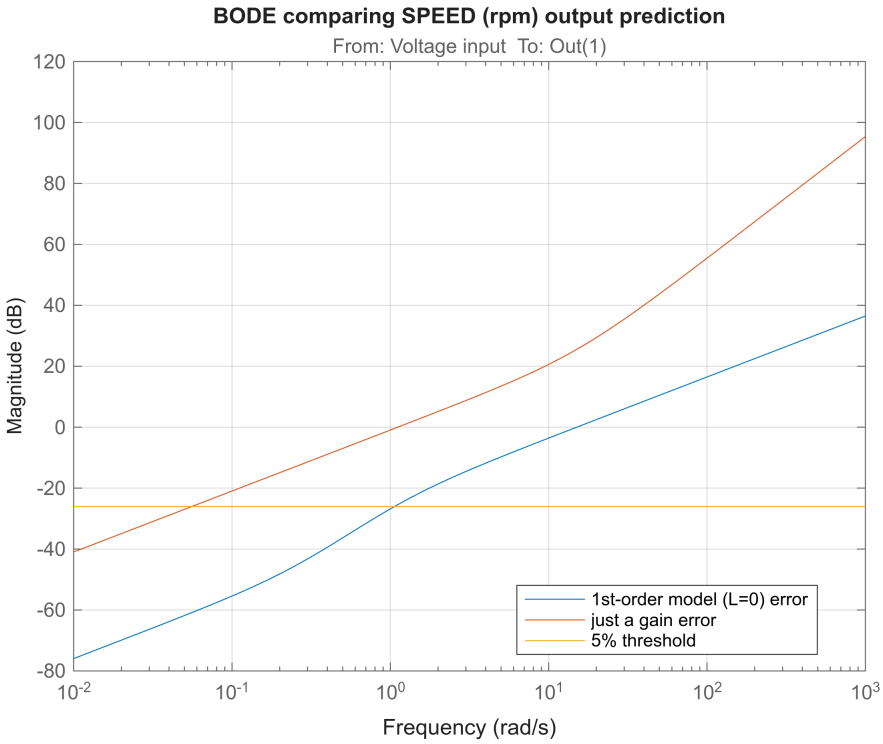


To be more precise, we can plot the relative error, and even the error with the "zeroth-order model", no dynamics, just the DC gain:

```
AbsoluteError=(sys-sysReduced);
AbsoluteError.OutputName={};
RelativeError=(inv([sys(1) 0;0 sys(2)])*AbsoluteError);
RelativeError.OutputName={};
AbsoluteError2=(sys-dcgain(sys));
RelativeError2=(inv([sys(1) 0;0 sys(2)])*AbsoluteError2);
RelativeError2.OutputName={};
%We'll also plot a 5 percent error bound as an horizontal line:
bodemag(RelativeError(1),RelativeError2(1),tf(0.05)), grid on
legend("1st-order model (L=0) error","just a gain error","5%
threshold",Location="best")
title("BODE comparing CURRENT output prediction")
```




```
bodemag(RelativeError(2),RelativeError2(2),tf(0.05)), grid on
legend("1st-order model (L=0) error","just a gain error","5%
threshold",Location="best")
title("BODE comparing SPEED (rpm) output prediction")
```



If the system isn't used to keep constant speed but to follow "band-limited" trajectories, with smooth band-limited inputs, the models would be similar (up to 5% error) under a maximum bandwidth of roughly 1 rad/s.

Thus, 5% accuracy or better for "application bandwidth" of less than 1 rad/s with the 1st-order model. The gain-only model would be 95% accurate only to predict speeds up to 0.06 rad/second application bandwidth, and to predict currents up to 0.02 rad/second.

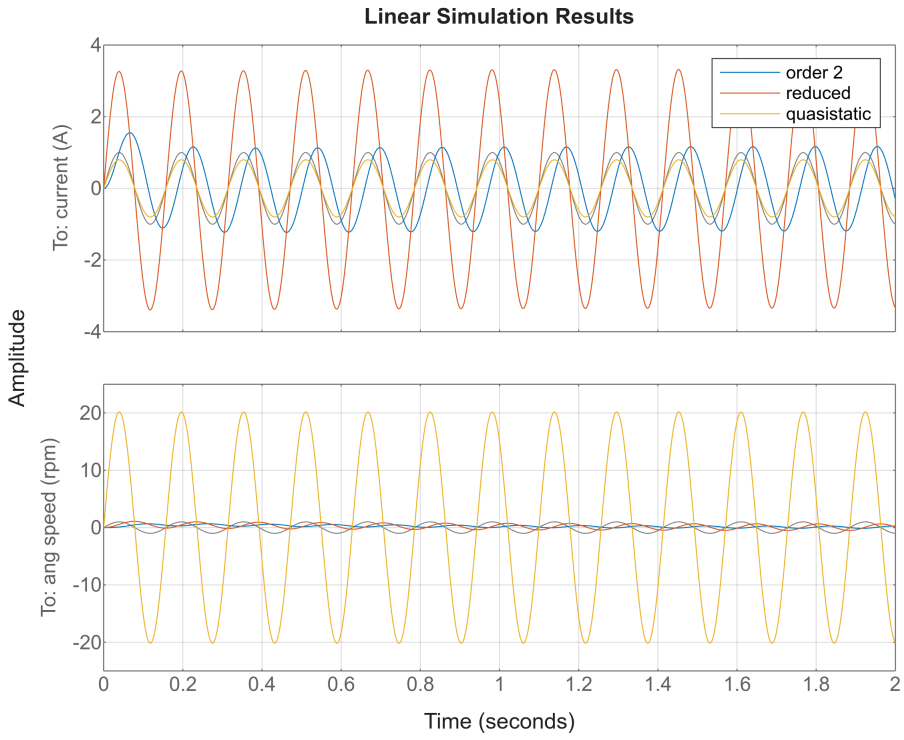
```
limit_bandwidth=1;
half_period=pi/limit_bandwidth
```

```
half_period =
3.1416
```

```
period=2*pi/limit_bandwidth
```

```
period =
6.2832
```

```
Trange=0:0.002:2;
actual_bw=40;
u=@(t) sin(actual_bw*Trange);
lsim(sys,u(Trange),Trange), grid on,
hold on
lsim(sysReduced,u(Trange),Trange), grid on,
lsim(tf(dcgain(sys)),u(Trange),Trange), grid on,
legend("order 2","reduced","quasistatic", Location="best")
hold off
```

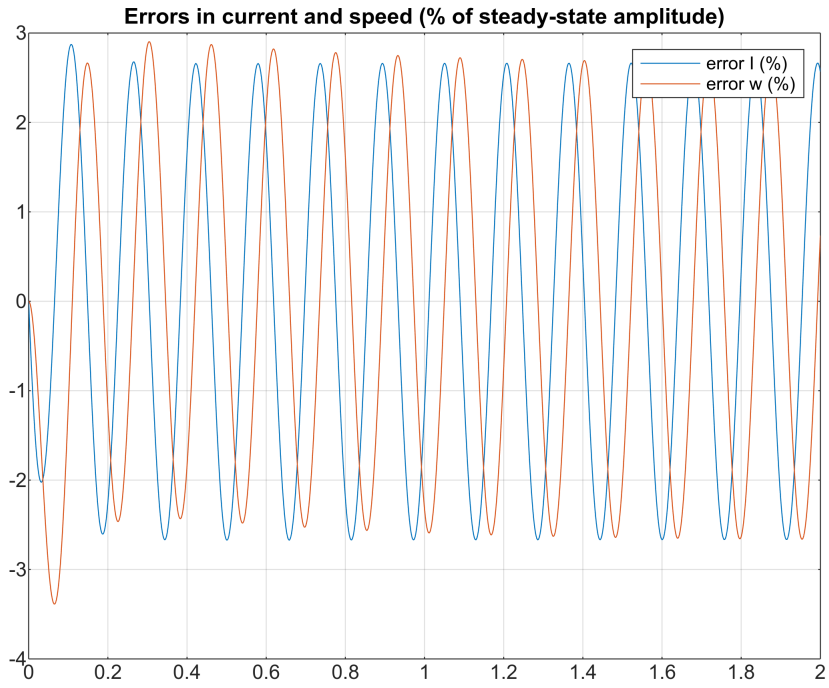



```
qq=abs(freqresp(sys,actual_bw)) %steady-state amplitudes
```

```
qq = 2×1
    1.1788
    0.2026
```

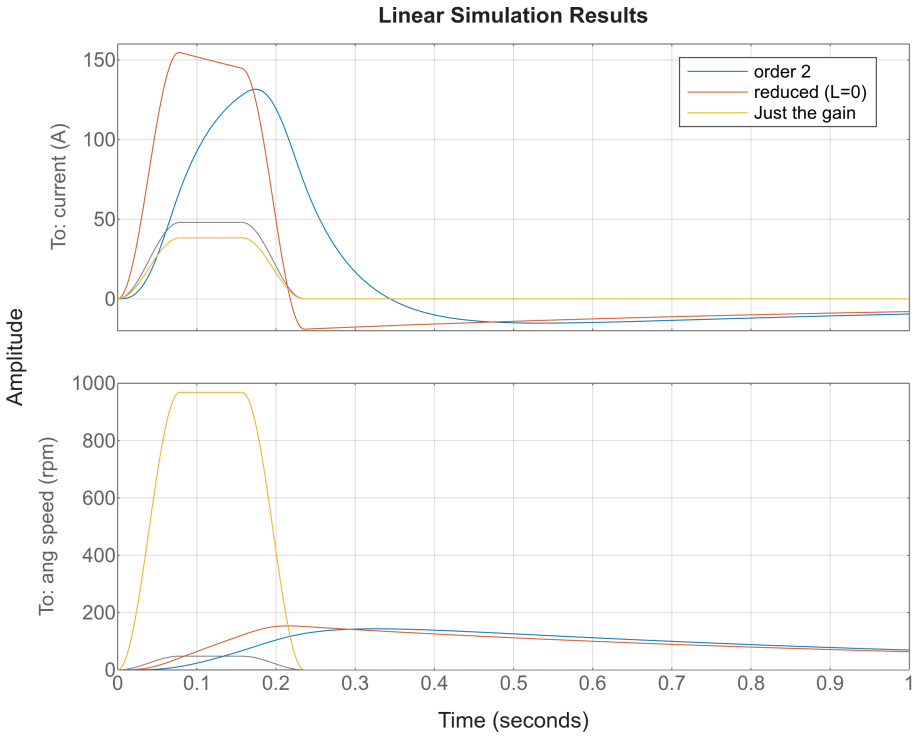
```
[errsim,tsim]=lsim(sys-sysReduced,u(Trange),Trange);
```

```
plot(tsim,errsim*inv(diag(qq))), grid on, legend("error I (%)",
"error w (%)")
title("Errors in current and speed (% of steady-state amplitude)")
```



Let's simulate a sort of "smooth step transition": smooth transitions are nice to avoid "peak currents", high-frequency vibrations, etc... as an example of a "bandwitdh limited" application (sort of).

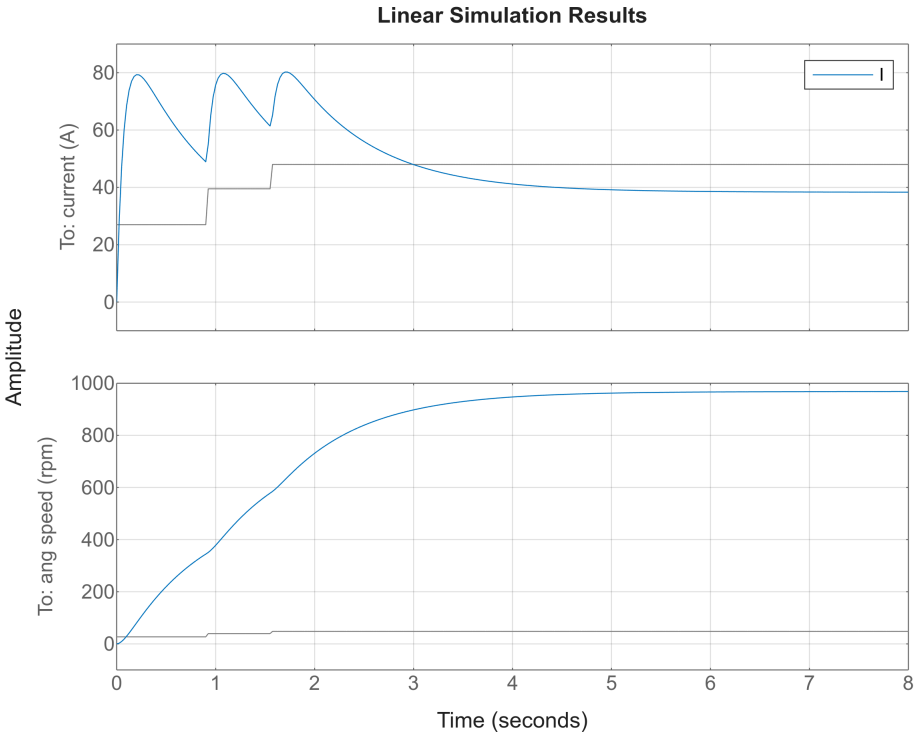
```
actual_bw=40;
half_period=pi/actual_bw;
Trange=0:0.002:max(1,4*half_period);
u=@(t) 0.5*(1-cos(actual_bw*Trange)).*(t<=half_period) ...
    +1*(t>half_period).*(t<2*half_period) ...
    +0.5*(cos(actual_bw*Trange)
+1).*(t>2*half_period).*(t<3*half_period);
lsim(sys,u(Trange)*OperatingVoltage,Trange), grid on,
hold on
lsim(sysReduced,u(Trange)*OperatingVoltage,Trange), grid on,
lsim(tf(dcgain(sysReduced)),u(Trange)*OperatingVoltage,Trange),
grid on,
legend("order 2","reduced (L=0)","Just the gain",Location="best")
hold off
```



Start-up with current limit

If we don't wish to have more than (approx) 80A during the transient, we need to start with a lower voltage. We may have an input profile consisting of two steps, a ramp... Details out of the scope of this material, obviously.

```
Trange=0:0.025:8;
t_switch1=0.9;t_switch2=1.55;
u=@(t) 27+12.5*(t>t_switch1)+8.5*(t>t_switch2);
%u=@(t) min(48,23.85+16.4*t);
lsim(sys,u(Trange),Trange), grid on, legend('I')
```


Linear Simulation Results
Amplitude
To: current (A)
I
80
60
40
20
0
To: ang speed (rpm)
1000
800
600
400
200
0
0
1
2
3
4
5
6
7
8
Time (seconds)