Motor Physics, with Gears & Initial Conditions

Robert Atkinson, Maddy Nguyen, Samantha Hordyk, Cole Welch, and John Fraser
16 August 2018

An exploration of the physics of motors, in both symbolic and numeric form. A basic but sufficiently-complete parameterized model of a motor is presented. The differential equations governing that model are exhibited, and, from those, the (frequency domain) transfer functions of each of the model's variables is developed from first principles. Next, mathematical infrastructure for converting the response of general frequency domain functions input signals is developed, then applied to the transfer functions as a function of initial conditions, and the applied-voltage and external-applied-torque inputs. This yields time domain functions for each of the model's variables in terms of the (symbolic) value of the model's variables. With those time domain symbolic functions in hand, the steady-state response of the motor as a function of each of the model's parameters is explored.

1. Administrivia

We begin by loading some handy utilities.

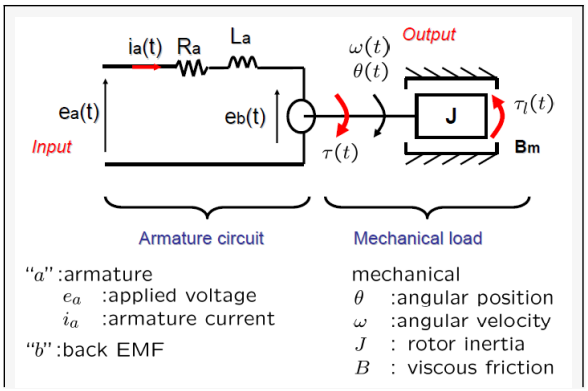
```
Get[NotebookDirectory[] <> "Utilities.m"]
```

```
outputDirectory = FileNameJoin[{NotebookDirectory[], "MotorPhysicsGearsInitialConditions.Output"}] <> $PathnameSeparator
CreateDirectory[outputDirectory] // Quiet;
```

```
C:\ftc\RobotPhysics\MotorPhysicsGearsInitialConditions.Output\
```

2. Motor Model

The following figure gives a conceptual overview of a DC motor (ref: http://www.egr.msu.edu/classes/me451/jchoi/2008/).



In addition to the the above, we also here incorporate into our model:

- an (optional) N:1 reducing gearbox with associated efficiency
- an autonomous time-dependent external torque

The state of the system can thus be described by the following variables.

- *vapp(t),* the applied voltage (illustrated as ea(t) above)
- *i(t),* the armature current
- *vg(t),* the back EMF ('g' for 'generator'; illustrated as eb(t) above)
- *θ(t)*, the angular position of the shaft, before the gearbox
- *ω(t)*, the angular velocity of the shaft, before the gearbox
- *α(t)*, the angular acceleration of the shaft, before the gearbox
- *τ(t)*, the torque driven by the armature current, before the gearbox

- $\tau app(t)$, an autonomous externally applied torque, after the gearbox. Often, this is constant, and caused by gravitational action.

The parameter constants that characterize the electrical aspects of the system are the following

- Ke, the electromotive force constant
- Kt, the motor torque constant
- L, electric inductance
- R, electric resistance

The following parameter constants characterize the mechanical aspects of the system

- J, the moment of inertia of the armature etc before the gearbox
- Jafter, the moment of inertia of any load or torque attached after the gearbox. May be (net) negative, depending on system; see τAfterExpr below.
- B, motor viscous drag constant
- Bafter, viscous drag of any load attached after the gearbox
- η, the efficiency of the motor's gear box (our current model does not support direction-dependent gearbox efficiencies).
- N (capital nu), the reduction ratio of the gear box

We collect together basic information about our parameters.

```
parameterNames = {J, Jafter, B, Bafter, Ke, Kt, R, L, η, N}
parameterAssumptions = {};
parameterAssumptions = Union[parameterAssumptions, # ≥ 0 & /@ {J, B, t}];
(*allow J & B to be zero so we can explore simpler motor models*)
parameterAssumptions = Union[parameterAssumptions, # > 0 & /@ {R, L, Ke, Kt, N, η}];
parameterAssumptions = Union[parameterAssumptions, # ∈ ℝ &/@{Jafter, Bafter, , vapp0, τapp0, ΔvappConst, ΔτappConst}];
parameterAssumptions = Union[parameterAssumptions, #[_] ∈ ℝ & /@ {Δvapp, i, vg, Θ, ω, α, τ, τafter, Δτapp}]
```

```
{J, Jafter, B, Bafter, Ke, Kt, R, L, η, N}
```

```
{Bafter ∈ ℝ, Jafter ∈ ℝ, Null ∈ ℝ, vapp0 ∈ ℝ, ΔvappConst ∈ ℝ, ΔτappConst ∈ ℝ, τapp0 ∈ ℝ, i[_] ∈ ℝ, vg[_] ∈ ℝ, α[_] ∈ ℝ, Δvapp[_] ∈ ℝ,
 Δτapp[_] ∈ ℝ, Θ[_] ∈ ℝ, τ[_] ∈ ℝ, τafter[_] ∈ ℝ, ω[_] ∈ ℝ, Ke > 0, Kt > 0, L > 0, R > 0, η > 0, N > 0, B ≥ 0, J ≥ 0, t ≥ 0}
```

3. Differential Equations

3.1. Differential Equation Development

3.1.1. On Gearboxes and Torques

Our modelled system contains a gearbox with (reducing) gear ratio N and efficiency η. On either side of the gearbox we may have loads, drags, and driving torques, the latter being electrically generated on one side and externally applied on the other. In order to be able to write the torque differential equation for the system as a whole, we need to be able to take these two sets of torques as they are found in their respective *separate* places in the system and rewrite them both as the are *effectively* felt at some *one* point in the system. Commonly, in mechanical engineering this one point is chosen to be just before the gearbox, and the loads / torques after the gearbox are said to be "reflected" to create corresponding equivalent reflected loads and torques at the before-gearbox location. Here, we develop the mathematics of carrying out this reflection.

Just a couple of principles are involved:

- State related to angular position (position, velocity, acceleration, etc.) on the output side is 1 / Nth that of on the input side.
- Torque on the output side is N η times that of on the input side.

```
Clear[reflectAngle, reflectTorque, reflectAngleCoefficient, reflectDrag, reflectInertia]
reflectAngle[expr_, out_, in_, t_] := expr /. {out[t] → in[t] / N, Derivative[n_][out][t] :↦ Derivative[n][in][t] / N}
reflectAngle[expr_] := reflectAngle[expr, θafter, θ, t]
reflectTorque[τout_] := τout / (N η)
reflectAngleCoefficient[coeff_] := coeff / N
reflectDrag[b_] := b // reflectAngleCoefficient // reflectTorque
reflectInertia[j_] := j // reflectAngleCoefficient // reflectTorque

reflectTorque[ΔτappConst + Jafter θafter''[t] + Bafter θafter'[t]] // reflectAngle // Expand
{reflectDrag[Bafter], reflectInertia[Jafter]}
```

$$\frac{\Delta\tau\text{appConst}}{\eta\, N}+\frac{\text{Bafter}\,\theta'[t]}{\eta\, N^2}+\frac{\text{Jafter}\,\theta''[t]}{\eta\, N^2}$$

$$\left\{\frac{\text{Bafter}}{\eta\, N^2},\ \frac{\text{Jafter}}{\eta\, N^2}\right\}$$

3.1.2. Differential Equation System

With that understanding on how to reflect torques, we can now develop the differential equations that govern the system.

The motor torque is proportional to current.

```
ktEqn = τ[t] == Kt i[t]
```

$$\tau[t]==\text{Kt}\,i[t]$$

The back EMF is proportional to velocity.

```
backEmfEqn = vg[t] == Ke θ'[t]
```

$$\text{vg}[t]==\text{Ke}\,\theta'[t]$$

The after-gearbox angular position is a simple ratio of the before-gearbox position. This also applies to the velocity and acceleration.

```
θafterEqns = Derivative[#][θafter][t] == Derivative[#][θ][t] / N & /@ Range[0, 2]
```

$$\left\{\theta\text{after}[t]==\frac{\theta[t]}{N},\ \theta\text{after}'[t]==\frac{\theta'[t]}{N},\ \theta\text{after}''[t]==\frac{\theta''[t]}{N}\right\}$$

Before the gearbox, the electrically driven motor torque is reduced by viscous drag and the inertia of the motor.

```
τBeforeExpr = τ[t] + -B θ'[t] - J θ''[t]
```

$$\tau[t]-\text{B}\,\theta'[t]-\text{J}\,\theta''[t]$$

After the gearbox, we have an autonomous, time-dependant externally applied torque, a viscous drag, and an internal load and/or an acceleration-dependent external torque.

```
τAfterExpr = τapp[t] - Bafter θafter'[t] - Jafter θafter''[t]
```

$$\tau\text{app}[t]-\text{Bafter}\,\theta\text{after}'[t]-\text{Jafter}\,\theta\text{after}''[t]$$

As all the torque must be accounted for, the sum of the before torques and the reflected after torques must be zero.

```
τEqn = τBeforeExpr + (τAfterExpr // reflectTorque // reflectAngle) == 0;
τEqn = Collect[τEqn, {θ[t], θ'[t], θ''[t]}];
τEqns = {τEqn, reflectTorque[τafter[t]] == τ[t] }
```

$$\left\{\tau[t]+\frac{\tau\text{app}[t]}{\eta\, N}+\left(-\text{B}-\frac{\text{Bafter}}{\eta\, N^2}\right)\theta'[t]+\left(-\text{J}-\frac{\text{Jafter}}{\eta\, N^2}\right)\theta''[t]==0,\ \frac{\tau\text{after}[t]}{\eta\, N}==\tau[t]\right\}$$

The voltage drop across the resistance, induction and back EMF must equal the applied voltage (due to one of Kirchhoff's laws).

```
voltageEqn = vapp[t] == R i[t] + L i'[t] + vg[t]
```

```
vapp[t] == R i[t] + vg[t] + L i'[t]
```

We put it all together.

```
diffEqns = {ktEqn, backEmfEqn, voltageEqn} ~Join~ τEqns ~Join~ θafterEqns // FullSimplify;
diffEqns // prettyPrint
```

$$\begin{aligned}
&\text{Kt}\, i(t) = \tau(t)\\
&\text{vg}(t) = \text{Ke}\, \theta'(t)\\
&R\, i(t) + \text{vg}(t) + L\, i'(t) = \text{vapp}(t)\\
&\frac{\eta\, \tau(t)\, \text{N}^2 + \tau\text{app}(t)\, \text{N} - (B\, \eta\, \text{N}^2 + \text{Bafter})\, \theta'(t) - (J\, \eta\, \text{N}^2 + \text{Jafter})\, \theta''(t)}{\eta\, \text{N}} = 0\\
&\tau(t) = \frac{\tau\text{after}(t)}{\eta\, \text{N}}\\
&\theta\text{after}(t) = \frac{\theta(t)}{\text{N}}\\
&\theta\text{after}'(t) = \frac{\theta'(t)}{\text{N}}\\
&\theta\text{after}''(t) = \frac{\theta''(t)}{\text{N}}
\end{aligned}$$

3.2. Initial Conditions

3.2.1. Steady state before t==0

We will assume that initially, the motor is in steady state at some velocity because of the application of some particular applied voltage and external torque. In steady state, the derivative of current and velocity are zero.

```
someInitialConditions = {
  vapp[ss0] == vapp0,
  τapp[ss0] == τapp0,
  θ[ss0] == 0, (* arbitrary *)
  θ''[ss0] == 0, (* steady state condition *)
  i'[ss0] == 0 (* steady state condition *)
 }
```

```
{vapp[ss0] == vapp0, τapp[ss0] == τapp0, Θ[ss0] == 0, Θ''[ss0] == 0, i'[ss0] == 0}
```

From those, we can calculate the values of the remainder of the state.

```
diffEqFunctionsOfTime = Reap[Scan[If[MatchQ[#, _[t]], Sow[#]] &, diffEqns, Infinity]][[2, 1]] // Union
others = Complement[diffEqFunctionsOfTime /. t → ss0, someInitialConditions[[All, 1]]];
eqnsSubsSs0 = diffEqns /. t → ss0 /. (someInitialConditions /. Equal → Rule);
(solvedSs0 = (uniqueSolve[eqnsSubsSs0, others] /. Rule → Equal));
(initialConditions = solvedSs0 ~ Join ~ someInitialConditions) // prettyPrint
```

```
{i[t], vapp[t], vg[t], θ[t], θafter[t], τ[t], τafter[t], τapp[t], i'[t], θ'[t], θafter'[t], θ''[t], θafter''[t]}
```

$$i(\text{ss0}) = -\frac{-B\,\text{vapp0}\,\eta\,\text{N}^2+\text{Ke}\,\tau\text{app0}\,\text{N}-\text{Bafter}\,\text{vapp0}}{\text{Ke}\,\text{Kt}\,\eta\,\text{N}^2+B\,R\,\eta\,\text{N}^2+\text{Bafter}\,R}$$
$$\text{vg}(\text{ss0}) = -\frac{-\text{Ke}\,\text{Kt}\,\text{vapp0}\,\eta\,\text{N}^2-\text{Ke}\,R\,\tau\text{app0}\,\text{N}}{\text{Ke}\,\text{Kt}\,\eta\,\text{N}^2+B\,R\,\eta\,\text{N}^2+\text{Bafter}\,R}$$
$$\theta\text{after}(\text{ss0}) = 0$$
$$\tau(\text{ss0}) = \frac{\text{Kt}\,(B\,\text{vapp0}\,\eta\,\text{N}^2-\text{Ke}\,\tau\text{app0}\,\text{N}+\text{Bafter}\,\text{vapp0})}{\text{Ke}\,\text{Kt}\,\eta\,\text{N}^2+B\,R\,\eta\,\text{N}^2+\text{Bafter}\,R}$$
$$\tau\text{after}(\text{ss0}) = \frac{\text{Kt}\,\eta\,\text{N}\,(B\,\text{vapp0}\,\eta\,\text{N}^2-\text{Ke}\,\tau\text{app0}\,\text{N}+\text{Bafter}\,\text{vapp0})}{\text{Ke}\,\text{Kt}\,\eta\,\text{N}^2+B\,R\,\eta\,\text{N}^2+\text{Bafter}\,R}$$
$$\theta'(\text{ss0}) = -\frac{-\text{Kt}\,\text{vapp0}\,\eta\,\text{N}^2-R\,\tau\text{app0}\,\text{N}}{\text{Ke}\,\text{Kt}\,\eta\,\text{N}^2+B\,R\,\eta\,\text{N}^2+\text{Bafter}\,R}$$
$$\theta\text{after}'(\text{ss0}) = -\frac{-\text{Kt}\,\text{vapp0}\,\eta\,\text{N}-R\,\tau\text{app0}}{\text{Ke}\,\text{Kt}\,\eta\,\text{N}^2+B\,R\,\eta\,\text{N}^2+\text{Bafter}\,R}$$
$$\theta\text{after}''(\text{ss0}) = 0$$
$$\text{vapp}(\text{ss0}) = \text{vapp0}$$
$$\tau\text{app}(\text{ss0}) = \tau\text{app0}$$
$$\theta(\text{ss0}) = 0$$
$$\theta'(\text{ss0}) = 0$$
$$i'(\text{ss0}) = 0$$

3.2.2. Initial conditions at t==0

Recall our differential equations.

```
diffEqns // prettyPrint
```

$$\text{Kt}\,i(t) = \tau(t)$$
$$\text{vg}(t) = \text{Ke}\,\theta'(t)$$
$$R\,i(t)+\text{vg}(t)+L\,i'(t) = \text{vapp}(t)$$
$$\frac{\eta\,\tau(t)\,\text{N}^2+\tau\text{app}(t)\,\text{N}-(B\,\eta\,\text{N}^2+\text{Bafter})\,\theta'(t)-(J\,\eta\,\text{N}^2+\text{Jafter})\,\theta''(t)}{\eta\,\text{N}} = 0$$
$$\tau(t) = \frac{\tau\text{after}(t)}{\eta\,\text{N}}$$
$$\theta\text{after}(t) = \frac{\theta(t)}{\text{N}}$$
$$\theta\text{after}'(t) = \frac{\theta'(t)}{\text{N}}$$
$$\theta\text{after}''(t) = \frac{\theta''(t)}{\text{N}}$$

In order to be able to use Laplace transforms, we need to adjust these so that the input values are zero at t==0. To do so, we substitute in a bias for both inputs.

```
(ΔdiffEqns = diffEqns /. {vapp[t] → vapp0 + Δvapp[t], τapp[t] → τapp0 + Δτapp[t]}) // prettyPrint
```

$$\text{Kt}\,i(t) = \tau(t)$$
$$\text{vg}(t) = \text{Ke}\,\theta'(t)$$
$$R\,i(t)+\text{vg}(t)+L\,i'(t) = \text{vapp0}+\Delta\text{vapp}(t)$$
$$\frac{\eta\,\tau(t)\,\text{N}^2+(\tau\text{app0}+\Delta\tau\text{app}(t))\,\text{N}-(B\,\eta\,\text{N}^2+\text{Bafter})\,\theta'(t)-(J\,\eta\,\text{N}^2+\text{Jafter})\,\theta''(t)}{\eta\,\text{N}} = 0$$
$$\tau(t) = \frac{\tau\text{after}(t)}{\eta\,\text{N}}$$
$$\theta\text{after}(t) = \frac{\theta(t)}{\text{N}}$$
$$\theta\text{after}'(t) = \frac{\theta'(t)}{\text{N}}$$
$$\theta\text{after}''(t) = \frac{\theta''(t)}{\text{N}}$$

We now set up the initial conditions applicable for t >= 0.

```
initialConditionsRules = initialConditions /. Equal → Rule;
ΔsomeInitialConditions = {
  Δvapp[0] == ΔvappConst,
  Δτapp[0] == ΔτappConst,
  θ[0] == 0,
  i[0] == i[ss0] /. initialConditionsRules /. ss0 → 0,
  θ'[0] == θ'[ss0] /. initialConditionsRules /. ss0 → 0
 }
```

$$\left\{\Delta\text{vapp}[0] == \Delta\text{vappConst},\ \Delta\tau\text{app}[0] == \Delta\tau\text{appConst},\ \theta[0] == 0,\ i[0] == -\frac{-\text{Bafter vapp0} - \text{B vapp0}\,\eta\,\text{N}^2 + \text{Ke N}\,\tau\text{app0}}{\text{Bafter R} + \text{Ke Kt}\,\eta\,\text{N}^2 + \text{B R}\,\eta\,\text{N}^2},\ \theta'[0] == -\frac{-\text{Kt vapp0}\,\eta\,\text{N}^2 - \text{R N}\,\tau\text{app0}}{\text{Bafter R} + \text{Ke Kt}\,\eta\,\text{N}^2 + \text{B R}\,\eta\,\text{N}^2}\right\}$$

```
ΔDiffEqFunctionsOfTime = Reap[Scan[If[MatchQ[#, _[t]], Sow[#]] &, ΔdiffEqns, Infinity]][[2, 1]] // Union
Δothers = Complement[ΔDiffEqFunctionsOfTime /. t → 0, ΔsomeInitialConditions[[All, 1]]]
ΔeqnsSubsKnownInitial = ΔdiffEqns /. t → 0 /. (ΔsomeInitialConditions /. Equal → Rule);
ΔsolvedInitialConditions = (uniqueSolve[ΔeqnsSubsKnownInitial, Δothers] /. Rule → Equal);
(ΔinitialConditions = ΔsolvedInitialConditions ~Join~ ΔsomeInitialConditions) // prettyPrint
```

```
{i[t], vg[t], Δvapp[t], Δτapp[t], θ[t], θafter[t], τ[t], τafter[t], i'[t], θ'[t], θafter'[t], θ''[t], θafter''[t]}
```

```
{vg[0], θafter[0], τ[0], τafter[0], i'[0], θafter'[0], θ''[0], θafter''[0]}
```

$$\text{vg}(0) = \frac{\text{Ke (Kt vapp0}\,\eta\,\text{N}^2 + R\ \tau\text{app0 N)}}{\text{Ke Kt}\,\eta\,\text{N}^2 + B\,R\,\eta\,\text{N}^2 + \text{Bafter}\,R}$$

$$\theta\text{after}(0) = 0$$

$$\tau(0) = -\frac{-B\,\text{Kt vapp0}\,\eta\,\text{N}^2 + \text{Ke Kt}\ \tau\text{app0 N} - \text{Bafter Kt vapp0}}{\text{Ke Kt}\,\eta\,\text{N}^2 + B\,R\,\eta\,\text{N}^2 + \text{Bafter}\,R}$$

$$\tau\text{after}(0) = \frac{\eta\,\text{N}\,(B\,\text{Kt vapp0}\,\eta\,\text{N}^2 - \text{Ke Kt}\ \tau\text{app0 N} + \text{Bafter Kt vapp0})}{\text{Ke Kt}\,\eta\,\text{N}^2 + B\,R\,\eta\,\text{N}^2 + \text{Bafter}\,R}$$

$$i'(0) = \frac{\Delta\text{vappConst}}{L}$$

$$\theta\text{after}'(0) = -\frac{-\text{Kt vapp0}\,\eta\,\text{N} - R\ \tau\text{app0}}{\text{Ke Kt}\,\eta\,\text{N}^2 + B\,R\,\eta\,\text{N}^2 + \text{Bafter}\,R}$$

$$\theta''(0) = \frac{\Delta\tau\text{appConst N}}{J\,\eta\,\text{N}^2 + \text{Jafter}}$$

$$\theta\text{after}''(0) = \frac{\Delta\tau\text{appConst}}{J\,\eta\,\text{N}^2 + \text{Jafter}}$$

$$\Delta\text{vapp}(0) = \Delta\text{vappConst}$$

$$\Delta\tau\text{app}(0) = \Delta\tau\text{appConst}$$

$$\theta(0) = 0$$

$$i(0) = -\frac{-B\,\text{vapp0}\,\eta\,\text{N}^2 + \text{Ke}\ \tau\text{app0 N} - \text{Bafter vapp0}}{\text{Ke Kt}\,\eta\,\text{N}^2 + B\,R\,\eta\,\text{N}^2 + \text{Bafter}\,R}$$

$$\theta'(0) = -\frac{-\text{Kt vapp0}\,\eta\,\text{N}^2 - R\ \tau\text{app0 N}}{\text{Ke Kt}\,\eta\,\text{N}^2 + B\,R\,\eta\,\text{N}^2 + \text{Bafter}\,R}$$

4. Motor Parameters

4.1. Motor Parameter Units

We develop infrastructure regarding the units in which each of our parameters are specified. This will help us sanity check the structure of our equations, as they must be internally consistent in their use of units (e.g.: no velocities added to masses, etc).

A note on radians as a unit. The Bureau International des Poids et Mesures, the intergovernmental organization that standardizes the SI, officially refers to a radian as “a special name is given to the unit one, in order to facilitate the identification of the quantity involved” (Ref: https://www.bipm.org/en/publications/si-brochure/section2-2-3.html). With that as background, we have here chosen to include radians in all places in which they logically belong, as doing so provides valuable insight regarding the quantities being manipulated. Thus, for example, you will find here torque having units of N m / radian instead of N m, and the like. As, from the SI perspective, this amounts to multiplying or dividing by the unit one, no semantic change is effected.

Aside: in the opinion of one of the authors (Atkinson), that “a special name given to the unit one” is in any way at all different than a first-class “unit” is a perspective reasonable only if one artificially and myopically limits one’s worldview to contain only the seven units whose dimension is necessarily *experimentally* calibrated as opposed to units whose dimension is mathematically or otherwise calibrated, such as radian, becquerel, motor encoder ticks, etc. And the world is chock-full of the latter. But this is a discussion for another time. See also Jacques E. Romain, “Angle as a Fourth Fundamental

Quantity", Journal of Research of the National Bureau of Standards-B. Mathematics and Mathematical Physics, Vol. 66B, No. 3, July- September 1962.

```
radiansUnits = parseUnit["Radians"];
secondsUnits = parseUnit["Seconds"];
siAngularPositionUnits = radiansUnits;
siAngularVelocityUnits = siAngularPositionUnits / secondsUnits;
siAngularAccelerationUnits = siAngularVelocityUnits / secondsUnits;
siTorqueUnits = parseUnit["N m"] / radiansUnits;
siAngularViscousDragUnits = siTorqueUnits / siAngularVelocityUnits;
siAngularInertialUnits = siTorqueUnits / siAngularAccelerationUnits;

parameterUnits = {
    R → parseUnit["Ohms"],
    L → parseUnit["Henrys"],
    i[t] → parseUnit["Amperes"],
    i'[t] → parseUnit["Amperes"] / secondsUnits,
    i''[t] → parseUnit["Amperes"] / secondsUnits / secondsUnits,
    Δvapp[t] → parseUnit["Volts"],
    ΔvappConst → parseUnit["Volts"],
    vg[t] → parseUnit["Volts"],
    vapp0 → parseUnit["Volts"],
    vapp0 → parseUnit["Volts"],

    J → siAngularInertialUnits,
    Jafter → siAngularInertialUnits,
    B → siAngularViscousDragUnits,
    Bafter → siAngularViscousDragUnits,

    Ke → parseUnit["Seconds Volts / Radians"],
    Kt → parseUnit["Newtons Meters"] / parseUnit["Amperes"] / radiansUnits,

    θ[t] → siAngularPositionUnits, θafter[t] → siAngularPositionUnits,
    θ'[t] → siAngularVelocityUnits, θafter'[t] → siAngularPositionUnits,
    θ''[t] → siAngularAccelerationUnits, θafter''[t] → siAngularPositionUnits,
    ω[t] → siAngularVelocityUnits,
    α[t] → siAngularAccelerationUnits,

    τ[t] → siTorqueUnits,
    τafter[t] → siTorqueUnits,
    Δτapp[t] → siTorqueUnits,
    ΔτappConst → siTorqueUnits,
    τapp0 → siTorqueUnits,
    τapp0 → siTorqueUnits,

    N → parseUnit["DimensionlessUnit"],
    η → parseUnit["DimensionlessUnit"]
   } // Association;
unitsToQuantities[units_] := Module[{rules, makeQuantity},
  rules = Normal[units];
  makeQuantity = Function[{param, unit}, Quantity[param, unit]];
  (#[[1]] → makeQuantity @@ # &/@ rules) // Association
 ]
parameterQuantities = unitsToQuantities[parameterUnits]
```

```
⟨|R → R Ω, L → L H, i[t] → i[t] A, i'[t] → i'[t] A/s, i''[t] → i''[t] A/s², Δvapp[t] → Δvapp[t] V,
 ΔvappConst → ΔvappConst V, vg[t] → vg[t] V, vapp0 → vapp0 V, J → J m s²N/rad², Jafter → Jafter m s²N/rad²,
 B → B m s N/rad², Bafter → Bafter m s N/rad², Ke → Ke s V/rad, Kt → Kt m N/(A rad), θ[t] → θ[t] rad,
 θafter[t] → θafter[t] rad, θ'[t] → θ'[t] rad/s, θafter'[t] → θafter'[t] rad, θ''[t] → θ''[t] rad/s²,
 θafter''[t] → θafter''[t] rad, ω[t] → ω[t] rad/s, α[t] → α[t] rad/s², τ[t] → τ[t] m N/rad, τafter[t] → τafter[t] m N/rad,
 Δτapp[t] → Δτapp[t] m N/rad, ΔτappConst → ΔτappConst m N/rad, τapp0 → τapp0 m N/rad, N → N, η → η |⟩
```

Let's look and see if our units are consistent:

```
ΔdiffEqns /. parameterQuantities // ColumnForm
```

$$\text{Kt}\, i[t]\ \text{mN/rad} == \tau[t]\ \text{mN/rad}$$

$$\text{vg}[t]\ \text{V} == \text{Ke}\, \theta'[t]\ \text{V}$$

$$(\text{R}\, i[t] + \text{vg}[t] + \text{L}\, i'[t])\ \text{A}\Omega == (\text{vapp0} + \Delta\text{vapp}[t])\ \text{V}$$

$$\frac{\text{N}\,(\tau\text{app0}+\Delta\tau\text{app}[t])+\eta\,\text{N}^2\,\tau[t]-\left(\text{Bafter}+\text{B}\,\eta\,\text{N}^2\right)\theta'[t]-\left(\text{Jafter}+\text{J}\,\eta\,\text{N}^2\right)\theta''[t]}{\eta\,\text{N}}\ \text{mN/rad} == 0$$

$$\tau[t]\ \text{mN/rad} == \frac{\tau\text{after}[t]}{\eta\,\text{N}}\ \text{mN/rad}$$

$$\theta\text{after}[t]\ \text{rad} == \frac{\theta[t]}{\text{N}}\ \text{rad}$$

$$\theta\text{after}'[t]\ \text{rad} == \frac{\theta'[t]}{\text{N}}\ \text{rad/s}$$

$$\theta\text{after}''[t]\ \text{rad} == \frac{\theta''[t]}{\text{N}}\ \text{rad/s}^2$$

Huzzah! They check out (if the units were incompatible, Mathematica would've told us). How about our initial conditions?

```
initialConditions /. parameterQuantities // siUnits // simplifyUnits // ColumnForm
Δinitial Conditions /. parameterQuantities // siUnits // simplifyUnits // ColumnForm
```

$$i[ss0] == -\frac{-\text{Bafter}\ \text{vapp0}-\text{B}\ \text{vapp0}\ \eta\ \text{N}^2+\text{Ke}\ \text{N}\ \tau\text{app0}}{\text{Bafter}\ \text{R}+\text{Ke}\ \text{Kt}\ \eta\ \text{N}^2+\text{B}\ \text{R}\ \eta\ \text{N}^2}\ \text{A}$$

$$\text{vg}[ss0] == -\frac{-\text{Ke}\ \text{Kt}\ \text{vapp0}\ \eta\ \text{N}^2-\text{Ke}\ \text{R}\ \text{N}\ \tau\text{app0}}{\text{Bafter}\ \text{R}+\text{Ke}\ \text{Kt}\ \eta\ \text{N}^2+\text{B}\ \text{R}\ \eta\ \text{N}^2}\ \text{V}$$

$$\Theta\text{after}[ss0] == 0$$

$$\tau[ss0] == \frac{\text{Kt}\ \left(\text{Bafter}\ \text{vapp0}+\text{B}\ \text{vapp0}\ \eta\ \text{N}^2-\text{Ke}\ \text{N}\ \tau\text{app0}\right)}{\text{Bafter}\ \text{R}+\text{Ke}\ \text{Kt}\ \eta\ \text{N}^2+\text{B}\ \text{R}\ \eta\ \text{N}^2}\ \text{kg}\,\text{m}^2/\left(\text{s}^2\text{rad}\right)$$

$$\tau\text{after}[ss0] == \frac{\text{Kt}\ \eta\ \text{N}\ \left(\text{Bafter}\ \text{vapp0}+\text{B}\ \text{vapp0}\ \eta\ \text{N}^2-\text{Ke}\ \text{N}\ \tau\text{app0}\right)}{\text{Bafter}\ \text{R}+\text{Ke}\ \text{Kt}\ \eta\ \text{N}^2+\text{B}\ \text{R}\ \eta\ \text{N}^2}\ \text{kg}\,\text{m}^2/\left(\text{s}^2\text{rad}\right)$$

$$\Theta'[ss0] == -\frac{-\text{Kt}\ \text{vapp0}\ \eta\ \text{N}^2-\text{R}\ \text{N}\ \tau\text{app0}}{\text{Bafter}\ \text{R}+\text{Ke}\ \text{Kt}\ \eta\ \text{N}^2+\text{B}\ \text{R}\ \eta\ \text{N}^2}\ \text{rad}/\text{s}$$

$$\Theta\text{after}'[ss0] == -\frac{-\text{Kt}\ \text{vapp0}\ \eta\ \text{N}-\text{R}\ \tau\text{app0}}{\text{Bafter}\ \text{R}+\text{Ke}\ \text{Kt}\ \eta\ \text{N}^2+\text{B}\ \text{R}\ \eta\ \text{N}^2}\ \text{rad}/\text{s}$$

$$\Theta\text{after}''[ss0] == 0$$

$$\text{vapp}[ss0] == \text{vapp0}\ \text{V}$$

$$\tau\text{app}[ss0] == \tau\text{app0}\ \text{kg}\,\text{m}^2/\left(\text{s}^2\text{rad}\right)$$

$$\Theta[ss0] == 0$$

$$\Theta''[ss0] == 0$$

$$i'[ss0] == 0$$

$$\text{vg}[0] == \frac{\text{Ke}\ \left(\text{Kt}\ \text{vapp0}\ \eta\ \text{N}^2+\text{R}\ \text{N}\ \tau\text{app0}\right)}{\text{Bafter}\ \text{R}+\text{Ke}\ \text{Kt}\ \eta\ \text{N}^2+\text{B}\ \text{R}\ \eta\ \text{N}^2}\ \text{V}$$

$$\Theta\text{after}[0] == 0$$

$$\tau[0] == -\frac{-\text{Bafter}\ \text{Kt}\ \text{vapp0}-\text{B}\ \text{Kt}\ \text{vapp0}\ \eta\ \text{N}^2+\text{Ke}\ \text{Kt}\ \text{N}\ \tau\text{app0}}{\text{Bafter}\ \text{R}+\text{Ke}\ \text{Kt}\ \eta\ \text{N}^2+\text{B}\ \text{R}\ \eta\ \text{N}^2}\ \text{kg}\,\text{m}^2/\left(\text{s}^2\text{rad}\right)$$

$$\tau\text{after}[0] == \frac{\eta\ \text{N}\ \left(\text{Bafter}\ \text{Kt}\ \text{vapp0}+\text{B}\ \text{Kt}\ \text{vapp0}\ \eta\ \text{N}^2-\text{Ke}\ \text{Kt}\ \text{N}\ \tau\text{app0}\right)}{\text{Bafter}\ \text{R}+\text{Ke}\ \text{Kt}\ \eta\ \text{N}^2+\text{B}\ \text{R}\ \eta\ \text{N}^2}\ \text{kg}\,\text{m}^2/\left(\text{s}^2\text{rad}\right)$$

$$i'[0] == \frac{\Delta\text{vappConst}}{\text{L}}\ \text{A}/\text{s}$$

$$\Theta\text{after}'[0] == -\frac{-\text{Kt}\ \text{vapp0}\ \eta\ \text{N}-\text{R}\ \tau\text{app0}}{\text{Bafter}\ \text{R}+\text{Ke}\ \text{Kt}\ \eta\ \text{N}^2+\text{B}\ \text{R}\ \eta\ \text{N}^2}\ \text{rad}/\text{s}$$

$$\Theta''[0] == \frac{\Delta\tau\text{appConst}\ \text{N}}{\text{Jafter}+\text{J}\ \eta\ \text{N}^2}\ \text{rad}/\text{s}^2$$

$$\Theta\text{after}''[0] == \frac{\Delta\tau\text{appConst}}{\text{Jafter}+\text{J}\ \eta\ \text{N}^2}\ \text{rad}/\text{s}^2$$

$$\Delta\text{vapp}[0] == \Delta\text{vappConst}\ \text{V}$$

$$\Delta\tau\text{app}[0] == \Delta\tau\text{appConst}\ \text{kg}\,\text{m}^2/\left(\text{s}^2\text{rad}\right)$$

$$\Theta[0] == 0$$

$$i[0] == -\frac{-\text{Bafter}\ \text{vapp0}-\text{B}\ \text{vapp0}\ \eta\ \text{N}^2+\text{Ke}\ \text{N}\ \tau\text{app0}}{\text{Bafter}\ \text{R}+\text{Ke}\ \text{Kt}\ \eta\ \text{N}^2+\text{B}\ \text{R}\ \eta\ \text{N}^2}\ \text{A}$$

$$\Theta'[0] == -\frac{-\text{Kt}\ \text{vapp0}\ \eta\ \text{N}^2-\text{R}\ \text{N}\ \tau\text{app0}}{\text{Bafter}\ \text{R}+\text{Ke}\ \text{Kt}\ \eta\ \text{N}^2+\text{B}\ \text{R}\ \eta\ \text{N}^2}\ \text{rad}/\text{s}$$

Good. We save information regarding our parameters for later use.

```
saveDefinitions[outputDirectory <> "ParametersUnitsAndAssumptions.m",
 {parameterUnits, parameterQuantities, parameterAssumptions, radiansUnits, secondsUnits, siAngularPositionUnits,
  siAngularVelocityUnits, siAngularAccelerationUnits, siTorqueUnits, siAngularViscousDragUnits, siAngularInertialUnits}]
```

4.2. On Ke & Kt

A brief digression. It is often remarked that, in compatible units, Ke & Kt have the same magnitude and sign. However, while often true, this relation does not *always* hold.

Recall our differential equations:

```
diffEqns // ColumnForm
```

$$\begin{aligned}
&\text{Kt}\, i[t] == \tau[t]\\
&\text{vg}[t] == \text{Ke}\, \theta'[t]\\
&\text{R}\, i[t] + \text{vg}[t] + \text{L}\, i'[t] == \text{vapp}[t]\\
&\frac{\eta\, \text{N}^2\, \tau[t] + \text{N}\, \tau\text{app}[t] - \left(\text{Bafter} + \text{B}\, \eta\, \text{N}^2\right) \theta'[t] - \left(\text{Jafter} + \text{J}\, \eta\, \text{N}^2\right) \theta''[t]}{\eta\, \text{N}} == 0\\
&\tau[t] == \frac{\tau\text{after}[t]}{\eta\, \text{N}}\\
&\theta\text{after}[t] == \frac{\theta[t]}{\text{N}}\\
&\theta\text{after}'[t] == \frac{\theta'[t]}{\text{N}}\\
&\theta\text{after}''[t] == \frac{\theta''[t]}{\text{N}}
\end{aligned}$$

In steady state, the current and speed are constant.

```
(ssEqns = diffEqns /. {i'[t] → 0, θ''[t] → 0}) // ColumnForm
(ssEqns = Eliminate[ssEqns, {vg[t], B, θafter[t], θafter'[t], θafter''[t], τapp[t]}]) // ColumnForm
(ssEqns = ssEqns[[1 ;; 2]]) // ColumnForm
```

$$\begin{aligned}
&\text{Kt}\, i[t] == \tau[t]\\
&\text{vg}[t] == \text{Ke}\, \theta'[t]\\
&\text{R}\, i[t] + \text{vg}[t] == \text{vapp}[t]\\
&\frac{\eta\, \text{N}^2\, \tau[t] + \text{N}\, \tau\text{app}[t] - \left(\text{Bafter} + \text{B}\, \eta\, \text{N}^2\right) \theta'[t]}{\eta\, \text{N}} == 0\\
&\tau[t] == \frac{\tau\text{after}[t]}{\eta\, \text{N}}\\
&\theta\text{after}[t] == \frac{\theta[t]}{\text{N}}\\
&\theta\text{after}'[t] == \frac{\theta'[t]}{\text{N}}\\
&\theta\text{after}''[t] == 0
\end{aligned}$$

$$\begin{aligned}
&\text{Kt}\, i[t] == \tau[t]\\
&\text{vapp}[t] == \text{R}\, i[t] + \text{Ke}\, \theta'[t]\\
&\tau\text{after}[t] == \eta\, \text{N}\, \tau[t]\\
&\eta \neq 0\\
&\text{N} \neq 0
\end{aligned}$$

$$\begin{aligned}
&\text{Kt}\, i[t] == \tau[t]\\
&\text{vapp}[t] == \text{R}\, i[t] + \text{Ke}\, \theta'[t]
\end{aligned}$$

From a power point of view (ref: https://bit.ly/2Lxka0Z), we must have (electrical) power in = (mechanical) power out + (electrical + other) power losses:

```
powerEqns = And @@ {
    pwrIn == vapp[t] i[t],
    pwrOut == τ[t] θ'[t],
    pwrLosses == i[t]^2 R + pwrOtherPowerLosses,
    pwrIn == pwrOut + pwrLosses
   };
(ssPowerEqns = Union[ssEqns, powerEqns]) // prettyPrint
```

$$\begin{aligned}
&\text{pwrIn} = \text{pwrLosses} + \text{pwrOut}\\
&\text{pwrIn} = i(t)\, \text{vapp}(t)\\
&\text{pwrLosses} = R\, i(t)^2 + \text{pwrOtherPowerLosses}\\
&\text{pwrOut} = \tau(t)\, \theta'(t)\\
&\text{Kt}\, i(t) = \tau(t)\\
&\text{vapp}(t) = R\, i(t) + \text{Ke}\, \theta'(t)
\end{aligned}$$

Let's eliminate a few of the variables that we're not interested in.

```
kBalanceEqns = Eliminate[ssPowerEqns, {i[t], τ[t], θ'[t], vapp[t]}];
kBalanceEqns // prettyPrint
uniqueSolve[kBalanceEqns[[2]], pwrOtherPowerLosses] /. Rule → Equal
```

$$\begin{aligned}
&\text{pwrIn} = \text{pwrLosses} + \text{pwrOut}\\
&\text{Kt}\ \text{pwrOtherPowerLosses} = (\text{Ke} - \text{Kt})\ \text{pwrOut}
\end{aligned}$$

$$\left\{\text{pwrOtherPowerLosses} == \frac{(\text{Ke} - \text{Kt})\ \text{pwrOut}}{\text{Kt}}\right\}$$

We can conclude that if other power losses are zero, then Ke must equal Kt.

4.3. Experimental Motor Data

We have experimental data from several motors (Nguyen, Hordyk, & Fraser, 2017). We load in same. Note that the motor parameters herein have all been measured *after* the attached gearbox.

```
Clear[importMotorData, motorData]
importMotorData[] := importMotorData["Characterized Motors (Gears)v2.xlsx"];
importMotorData[file_] := Module[{},
  Import[NotebookDirectory[] <> file, {"Data", 1}][[2 ;; ]]
 ]
motorData = importMotorData[];
(motorData) // TableForm
```

Name	Model	R	L (uH)	L (H)	Ke	Kt (est)	J	B	Ticks/Rev	N	ηf (
AM 20 A	am-3102	2.3	691.	0.000691	0.351	0.351	9.011×10^{-6}	0.0022	560.	20.	0.9
AM 20 B		1.9	684.	0.000684	0.389	0.389	9.011×10^{-6}	0.0025	560.	20.	0.9
AM 20 C		5.1	717.	0.000717	0.385	0.385	8.931×10^{-6}	0.0028	560.	20.	0.9
AM 40 A	am-2964a	2.5	674.	0.000674	0.753	0.753	0.00002221	0.2269	1120.	40.	0.9
AM 40 B		3.8	705.	0.000705	0.705	0.705	0.00001741	0.56	1120.	40.	0.9
AM 40 C		2.1	716.	0.000716	0.763	0.763	0.00002471	0.018	1120.	40.	0.9
AM 60 A	am-3103	3.3	694.	0.000694	1.066	1.066	0.00001041	0.033	1680.	60.	0.9
AM 60 B		5.1	696.	0.000696	1.076	1.076	8.421×10^{-6}	0.02	1680.	60.	0.9
AM 3.7 A	am-3461	8.9	679.	0.000679	0.099	0.099	0.00002791	0.00014	44.4	3.7	0.9
AM 3.7 B		2.6	797.	0.000797	0.108	0.108	0.00003151	0.000176	44.4	3.7	0.9
AM 3.7 C		8.7	880.	0.00088	0.105	0.105	0.00003091	0.00017	44.4	3.7	0.9
Matrix A	14-0011	3.8	718.	0.000718	0.34	0.34	9.431×10^{-6}	0.00151	1478.4	52.8	0.9
Matrix B		7.8	777.	0.000777	0.363	0.363	7.761×10^{-6}	0.00191	1478.4	52.8	0.9
Matrix C		20.6	658.	0.000658	0.338	0.338	7.231×10^{-6}	0.00186	1478.4	52.8	0.9
CoreHex A	REV-41-1300	3.6	1356.	0.001356	0.822	0.822	0.0007331	0.0112	288.	72.	0.9
CoreHex B		11.3	1352.	0.001352	0.858	0.858	0.0006551	0.008	288.	72.	0.9
CoreHex C		5.6	1342.	0.001342	0.711	0.711	0.0004541	0.0078	288.	72.	0.9
Tetrix	W39530								1440.	52.	

We build a function to retrieve data from the table. We convert from floating point to rational in order to help delay floating point collapse later on down the line. We include load and external torque parameters at zero level so these can be easily added to later.

```
Clear[motorParameters];
Options[motorParameters] = { WithGearbox → True , Efficiency → "Forward" };
validateOption[motorParameters][WithGearbox, val_] := BooleanQ[val];
validateOption[motorParameters][Efficiency, val_] := MemberQ[{"Forward", "Reverse"}, val];

motorParameters[motorName_, opts : OptionsPattern[]] := motorParameters[motorData, motorName, opts]
motorParameters[motorData_, motorName_, opts : OptionsPattern[]] :=
  validateOptions[] @ Module[{row, paramValues, quantify, qty, quantifiedRow, result, gearbox},
    row = Select[motorData, #[[1]] == motorName &][[1]];
    paramValues = #[[1]] → toRational[ row[[#[[2]]]] ] & /@ {
       {R, 3}, {L, 5}, {Ke, 6}, {Kt, 7}, {J, 8}, {B, 9},
       {N, 11}, {η, If[OptionValue[Efficiency] == "Forward", 12, 13]} };
    quantify = Function[{name, value},
      qty = (name /. parameterQuantities);
      name → Quantity[value, QuantityUnit[qty]]
     ];
    quantifiedRow = quantify @@ # & /@ paramValues // Association;

    (* An empty container that we'll add to *)
    result = Association[];
    result[R] = quantifiedRow[R];
    result[L] = quantifiedRow[L];

    (* Figure out the gearing *)
    If[OptionValue[WithGearbox],
     result[N] = quantifiedRow[N];
     result[η] = quantifiedRow[η];
     ,
     result[N] = 1;
     result[η] = 1;
    ];
    (* WithGearbox being false means what it says: the load is directly on the armature. *)
    (* We still have to reflect the experimental parameters, either way. *)
    gearbox = {N → quantifiedRow[N], η → quantifiedRow[η]};

    (* In the result, we need to reflect measured values back before the gearbox *)
    result[Ke] = reflectAngleCoefficient[quantifiedRow[Ke]] /. gearbox;
    result[Kt] = reflectAngleCoefficient[quantifiedRow[Kt]] /. gearbox;
    result[B] = reflectDrag[quantifiedRow[B]] /. gearbox;
    result[J] = reflectInertia[quantifiedRow[J]] /. gearbox;

    (* add in defaults that we can later add thereto *)
    result[Jafter] = Quantity[0, siAngularInertialUnits];
    result[Bafter] = Quantity[0, siAngularViscousDragUnits];
    result[ΔτappConst] = Quantity[0, siTorqueUnits];

    result // siUnits // simplifyUnits
   ];

motorParameters["AM 60 A", WithGearbox → False]
motorParameters["AM 60 A", WithGearbox → True, Efficiency → "Reverse"]
```

$$\left\langle \left| R \to \frac{33}{10}\,W/A^2\,,\ L \to \frac{347}{500\,000}\,H\,,\ N \to 1,\ \eta \to 1,\ Ke \to \frac{533}{30\,000}\,kg\,m^2/(s^2\,A\,rad)\,,\ Kt \to \frac{533}{30\,000}\,kg\,m^2/(s^2\,A\,rad)\,,\ B \to \frac{11}{1\,080\,000}\,kg\,m^2/(s\,rad^2)\,,\right.\right.$$

$$\left.\left. J \to \frac{347}{108\,000\,000\,000}\,kg\,m^2/rad^2\,,\ Jafter \to 0\,kg\,m^2/rad^2\,,\ Bafter \to 0\,kg\,m^2/(s\,rad^2)\,,\ \Delta\tau appConst \to 0\,kg\,m^2/(s^2 rad) \right| \right\rangle$$

$$\left\langle\left| \mathtt{R} \to \frac{33}{10}\,\mathrm{W/A^2}\,,\ \mathtt{L} \to \frac{347}{500\,000}\,\mathrm{H}\,,\ \mathtt{N} \to 60,\ \eta \to \frac{4}{5},\ \mathtt{Ke} \to \frac{533}{30\,000}\,\mathrm{kg\,m^2/(s^2\,A\,rad)}\,,\ \mathtt{Kt} \to \frac{533}{30\,000}\,\mathrm{kg\,m^2/(s^2\,A\,rad)}\,,\ \mathtt{B} \to \frac{11}{960\,000}\,\mathrm{kg\,m^2/(s\,rad^2)}\,,\right.\right.$$

$$\left.\left. \mathtt{J} \to \frac{347}{96\,000\,000\,000}\,\mathrm{kg\,m^2/rad^2}\,,\ \mathtt{Jafter} \to 0\,\mathrm{kg\,m^2/rad^2}\,,\ \mathtt{Bafter} \to 0\,\mathrm{kg\,m^2/(s\,rad^2)}\,,\ \Delta\tau\mathtt{appConst} \to 0\,\mathrm{kg\,m^2/(s^2\,rad)} \right|\right\rangle$$

4.4. Loads on the Motor

motorLoad[] creates a motor load given explicit internal and drag parameters. We allow the 'after' suffix but do not require its use.

```
Clear[motorLoad]
Options[motorLoad] = {
   J → Quantity[0, siAngularInertialUnits], Jafter → Quantity[0, siAngularInertialUnits],
   B → Quantity[0, siAngularViscousDragUnits], Bafter → Quantity[0, siAngularViscousDragUnits],
   ΔτappConst → Quantity[0, siTorqueUnits]
  };
motorLoad[opts : OptionsPattern[]] := Module[{assoc = Association[]},
  assoc[Jafter] = OptionValue[J] + OptionValue[Jafter];
  assoc[Bafter] = OptionValue[B] + OptionValue[Bafter];
  assoc[ΔτappConst] = OptionValue[ΔτappConst];
  assoc
 ]
```

flywheel[] creates a motor load given mass and radius parameters

```
Clear[flywheel]
flywheel[mass_, radius_] := motorLoad[J -> mass * radius ^2 / 2 / Quantity[1, "Radians^2"]]
flywheel[Quantity[2, "kg"], Quantity[5, "cm"]]
```

$$\left\langle\left| \mathtt{Jafter} \to 25\,\mathrm{kg\,cm^2/rad^2}\,,\ \mathtt{Bafter} \to 0\,\mathrm{m\,s\,N/rad^2}\,,\ \Delta\tau\mathtt{appConst} \to 0\,\mathrm{m\,N/rad} \right|\right\rangle$$

addMotorLoad[] takes motor parameters and a load and returns new parameters that include the load.

```
addMotorLoad[motor_, load_] := Module[{assoc},
  assoc = Association[motor]; (*explicitly create copy*)
  assoc[Jafter] = (Jafter /. motor) + (Jafter /. load);
  assoc[Bafter] = (Bafter /. motor) + (Bafter /. load);
  assoc[ΔτappConst] = (ΔτappConst /. motor) + (ΔτappConst /. load);
  assoc
 ]
addMotorLoad[motorParameters["AM 60 A"], flywheel[Quantity[2, "kg"], Quantity[5, "cm"]]]
```

$$\left\langle\left| \mathtt{R} \to \frac{33}{10}\,\mathrm{W/A^2}\,,\ \mathtt{L} \to \frac{347}{500\,000}\,\mathrm{H}\,,\ \mathtt{N} \to 60,\ \eta \to \frac{9}{10},\ \mathtt{Ke} \to \frac{533}{30\,000}\,\mathrm{kg\,m^2/(s^2\,A\,rad)}\,,\ \mathtt{Kt} \to \frac{533}{30\,000}\,\mathrm{kg\,m^2/(s^2\,A\,rad)}\,,\ \mathtt{B} \to \frac{11}{1\,080\,000}\,\mathrm{kg\,m^2/(s\,rad^2)}\,,\right.\right.$$

$$\left.\left. \mathtt{J} \to \frac{347}{108\,000\,000\,000}\,\mathrm{kg\,m^2/rad^2}\,,\ \mathtt{Jafter} \to 25\,\mathrm{kg\,cm^2/rad^2}\,,\ \mathtt{Bafter} \to 0\,\mathrm{kg\,m^2/(s\,rad^2)}\,,\ \Delta\tau\mathtt{appConst} \to 0\,\mathrm{m\,N/rad} \right|\right\rangle$$

4.5. Mass on pulley

Here, we model a mass suspended on a massless "rigid string" from a massless pulley attached to the motor shaft. That string acts at a wrench with a length of the radius of the pulley to deliver a torque to the shaft. τMassOnPulley[] computes that torque.

There is a force pulling down on the mass due to gravity and up due to acceleration in the string that results from the driven angular acceleration of the motor shaft. The sum of these forces is the tension in the string. If the net acceleration is negative (ie: downward), a real (flexible) string would clamp at zero and have the mass in free-fall; we choose not to model that for now. Instead, here, our hypothetical rigid string continues to impart acceleration from the pulley even for negative accelerations. This is a bit odd for a string, but more reasonable if the string were in fact a rigid arm being lifted, or some such.

```
Clear[massOnPulley]
massOnPulley[mass_, radius_] := Module[
  {gravityAcceleration, gravityForce, angularAcceleration,
   tangentialAcceleration, tangentialForce, tension, torque, unit, parts, radians = Quantity["Radians"]},

  (* gravity exerts a constant downard force by acting on the mass *)
  gravityAcceleration = Quantity["StandardAccelerationOfGravity"];
  gravityForce = gravityAcceleration * mass;

  (* angular accelearation, converted to tangential acceration also acts on the mass to create an upward force *)
  angularAcceleration = Quantity[θ''[t], siAngularAccelerationUnits];
  tangentialAcceleration = angularAcceleration * radius / radians;
  tangentialForce = tangentialAcceleration * mass;

  (* the net force (the tension in the string) is the sum of the two forces*)
  tension = gravityForce + tangentialForce;

  (* the tension acts like a wrench with lever arm 'radius' *)
  torque = tension * radius / radians;

  (* we decompose the torque into constant and acceleration-dependent parts for output *)
  unit = QuantityUnit[torque];
  parts = List @@ (QuantityMagnitude[torque] // Apart);
  motorLoad[ΔτappConst → Quantity[parts[[1]], unit], Jafter → Quantity[parts[[2]], unit] / angularAcceleration]
 ]
massOnPulley[Quantity[weightLbs, "Pounds"], Quantity[rInches, "Inches"]]
% // siUnits // clearUnits // N
```

$$\langle\left|\text{Jafter} \to \text{rInches}^2\ \text{weightLbs}\ \text{lb}\,\text{in}^2/\text{rad}^2\ ,\ \text{Bafter} \to 0\,\text{m}\,\text{s}\,\text{N}/\text{rad}^2\ ,\ \Delta\tau\text{appConst} \to \frac{196\,133\ \text{rInches}\ \text{weightLbs}}{508}\ \text{lb}\,\text{in}^2/(\text{s}^2\text{rad})\right|\rangle$$

$$\langle\left|\text{Jafter} \to 0.00029264\ \text{rInches}^2\ \text{weightLbs},\ \text{Bafter} \to 0.,\ \Delta\tau\text{appConst} \to 0.112985\ \text{rInches}\ \text{weightLbs}\right|\rangle$$

4.6. Example Motors

We define motors and loads that we'll use to illustrate examples.

```
aFlywheel = flywheel[Quantity[10, "kg"], Quantity[10, "cm"]] // siUnits
aPulley = massOnPulley[Quantity[3, "Pounds"], Quantity[2, "Inches"]] // siUnits
```

$$\langle\left|\text{Jafter} \to \frac{1}{20}\ \text{kg}\,\text{m}^2/\text{rad}^2\ ,\ \text{Bafter} \to 0\,\text{kg}\,\text{m}^2/(\text{s}\,\text{rad}^2)\ ,\ \Delta\tau\text{appConst} \to 0\,\text{kg}\,\text{m}^2/(\text{s}^2\text{rad})\right|\rangle$$

$$\langle\left|\text{Jafter} \to \frac{2\,194\,797\,400\,719}{625\,000\,000\,000\,000}\ \text{kg}\,\text{m}^2/\text{rad}^2\ ,\ \text{Bafter} \to 0\,\text{kg}\,\text{m}^2/(\text{s}\,\text{rad}^2)\ ,\ \Delta\tau\text{appConst} \to \frac{3\,389\,544\,870\,828\,501}{5\,000\,000\,000\,000\,000}\ \text{kg}\,\text{m}^2/(\text{s}^2\text{rad})\right|\rangle$$

```
aMotor = motorParameters["AM 60 A"];
aMotorWithFlywheel = addMotorLoad[aMotor, aFlywheel] // siUnits;
aUnitlessMotorWithFlywheel = aMotorWithFlywheel // clearUnits // N
aMotorWithFlywheelNoGear = addMotorLoad[motorParameters["AM 60 A", WithGearbox → False], aFlywheel];
aUnitlessMotorWithFlywheelNoGear = aMotorWithFlywheelNoGear // siUnits // clearUnits // N
```

$$\langle\left|\text{R} \to 3.3,\ \text{L} \to 0.000694,\ \text{N} \to 60.,\ \eta \to 0.9,\ \text{Ke} \to 0.0177667,\ \text{Kt} \to 0.0177667,\right.$$
$$\left.\text{B} \to 0.0000101852,\ \text{J} \to 3.21296 \times 10^{-9},\ \text{Jafter} \to 0.05,\ \text{Bafter} \to 0.,\ \Delta\tau\text{appConst} \to 0.\right|\rangle$$

$$\langle\left|\text{R} \to 3.3,\ \text{L} \to 0.000694,\ \text{N} \to 1.,\ \eta \to 1.,\ \text{Ke} \to 0.0177667,\ \text{Kt} \to 0.0177667,\right.$$
$$\left.\text{B} \to 0.0000101852,\ \text{J} \to 3.21296 \times 10^{-9},\ \text{Jafter} \to 0.05,\ \text{Bafter} \to 0.,\ \Delta\tau\text{appConst} \to 0.\right|\rangle$$

5. Laplace Transforms and Transfer Function Models

5.1. Laplace Transforms

Returning to the differential equations, we form the Laplace Transform of each, then solve the for the various transforms. First, we apply the Laplace-Transform[] function to each equation. It automatically makes use of the linearity and differential properties of the transform, and insinuates itself just around the time dependent parts (ie: the parts dependent on the time variable we told it about, namely *t*).

```
leqns = LaplaceTransform[#, t, s] &/@ ΔdiffEqns;
leqns // prettyPrint
```

$$\text{Kt}\,(\mathcal{L}_t[i(t)]\,(s)) = \mathcal{L}_t[\tau(t)]\,(s)$$
$$\mathcal{L}_t[\text{vg}(t)]\,(s) = \text{Ke}\,(s\,(\mathcal{L}_t[\theta(t)]\,(s)) - \theta(0))$$
$$R\,(\mathcal{L}_t[i(t)]\,(s)) + L\,(s\,(\mathcal{L}_t[i(t)]\,(s)) - i(0)) + \mathcal{L}_t[\text{vg}(t)]\,(s) = \frac{\text{vapp0}}{s} + \mathcal{L}_t[\Delta\text{vapp}(t)]\,(s)$$
$$\frac{\eta\,(\mathcal{L}_t[\tau(t)]\,(s))\,\text{N}^2 + \frac{\text{rapp0 N}}{s} + (\mathcal{L}_t[\Delta\text{rapp}(t)]\,(s))\,\text{N} - (B\,\eta\,\text{N}^2 + \text{Bafter})\,(s\,(\mathcal{L}_t[\theta(t)]\,(s)) - \theta(0)) - (J\,\eta\,\text{N}^2 + \text{Jafter})\,((\mathcal{L}_t[\theta(t)]\,(s))\,s^2 - \theta(0)\,s - \theta'(0))}{\eta\,\text{N}} = 0$$
$$\mathcal{L}_t[\tau(t)]\,(s) = \frac{\mathcal{L}_t[\text{rafter}(t)]\,(s)}{\eta\,\text{N}}$$
$$\mathcal{L}_t[\theta\text{after}(t)]\,(s) = \frac{\mathcal{L}_t[\theta(t)]\,(s)}{\text{N}}$$
$$s\,(\mathcal{L}_t[\theta\text{after}(t)]\,(s)) - \theta\text{after}(0) = \frac{s\,(\mathcal{L}_t[\theta(t)]\,(s)) - \theta(0)}{\text{N}}$$
$$(\mathcal{L}_t[\theta\text{after}(t)]\,(s))\,s^2 - \theta\text{after}(0)\,s - \theta\text{after}'(0) = \frac{(\mathcal{L}_t[\theta(t)]\,(s))\,s^2 - \theta(0)\,s - \theta'(0)}{\text{N}}$$

Next, we walk those equations, picking up those insinuations and sowing them to the wind, reaping them on the outside, and finally removing duplicates.

```
allXforms = Reap[Scan[(If[MatchQ[#, _LaplaceTransform], Sow[#]]) &, leqns, Infinity]][[2, 1]] // Union
```

```
{LaplaceTransform[i[t], t, s], LaplaceTransform[vg[t], t, s], LaplaceTransform[Δvapp[t], t, s], LaplaceTransform[Δτapp[t], t, s],
 LaplaceTransform[θ[t], t, s], LaplaceTransform[θafter[t], t, s], LaplaceTransform[τ[t], t, s], LaplaceTransform[τafter[t], t, s]}
```

The voltage transform is input, as is any autonomous externally applied torque; all the others are outputs. Solve for the outputs. Finally substitute what we know about initial conditions, and simplify as much as we can.

```
ΔvappXform = LaplaceTransform[Δvapp[t], t, s];
ΔτappXform = LaplaceTransform[Δτapp[t], t, s];
inputXforms = {ΔvappXform, ΔτappXform}
inputLabels = {"1", "vapp", "τapp"}
(outputXorms = Complement[allXforms, inputXforms]) // ColumnForm
```

```
{LaplaceTransform[Δvapp[t], t, s], LaplaceTransform[Δτapp[t], t, s]}
```

```
{1, vapp, τapp}
```

```
LaplaceTransform[i[t], t, s]
LaplaceTransform[vg[t], t, s]
LaplaceTransform[θ[t], t, s]
LaplaceTransform[θafter[t], t, s]
LaplaceTransform[τ[t], t, s]
LaplaceTransform[τafter[t], t, s]
```

```
(leqnsToSolve = leqns /. (Δinitialconditions /. Equal → Rule)) // prettyPrint
```

$$\text{Kt}\,(\mathcal{L}_t[i(t)]\,(s)) = \mathcal{L}_t[\tau(t)]\,(s)$$
$$\mathcal{L}_t[\text{vg}(t)]\,(s) = \text{Ke}\,s\,(\mathcal{L}_t[\theta(t)]\,(s))$$
$$R\,(\mathcal{L}_t[i(t)]\,(s)) + L\left(\frac{-B\,\text{vapp0}\,\eta\text{N}^2+\text{Ke}\,\text{rapp0}\,\text{N}-\text{Bafter}\,\text{vapp0}}{\text{Ke}\,\text{Kt}\,\eta\text{N}^2+B\,R\,\eta\text{N}^2+\text{Bafter}\,R} + s\,(\mathcal{L}_t[i(t)]\,(s))\right) + \mathcal{L}_t[\text{vg}(t)]\,(s) = \frac{\text{vapp0}}{s} + \mathcal{L}_t[\Delta\text{vapp}(t)]\,(s)$$
$$\frac{\eta\,(\mathcal{L}_t[\tau(t)]\,(s))\,\text{N}^2+\frac{\text{rapp0}\,\text{N}}{s}+(\mathcal{L}_t[\Delta\,\text{rapp}(t)]\,(s))\,\text{N}-s\,(B\,\eta\text{N}^2+\text{Bafter})\,(\mathcal{L}_t[\theta(t)]\,(s))-(J\,\eta\text{N}^2+\text{Jafter})\left((\mathcal{L}_t[\theta(t)]\,(s))\,s^2+\frac{-\text{Kt}\,\text{vapp0}\,\eta\text{N}^2-R\,\text{rapp0}\,\text{N}}{\text{Ke}\,\text{Kt}\,\eta\text{N}^2+B\,R\,\eta\text{N}^2+\text{Bafter}\,R}\right)}{\eta\text{N}} = 0$$
$$\mathcal{L}_t[\tau(t)]\,(s) = \frac{\mathcal{L}_t[\text{rafter}(t)]\,(s)}{\eta\text{N}}$$
$$\mathcal{L}_t[\theta\text{after}(t)]\,(s) = \frac{\mathcal{L}_t[\theta(t)]\,(s)}{\text{N}}$$
$$s\,(\mathcal{L}_t[\theta\text{after}(t)]\,(s)) = \frac{s\,(\mathcal{L}_t[\theta(t)]\,(s))}{\text{N}}$$
$$(\mathcal{L}_t[\theta\text{after}(t)]\,(s))\,s^2 + \frac{-\text{Kt}\,\text{vapp0}\,\eta\text{N}-R\,\text{rapp0}}{\text{Ke}\,\text{Kt}\,\eta\text{N}^2+B\,R\,\eta\text{N}^2+\text{Bafter}\,R} = \frac{(\mathcal{L}_t[\theta(t)]\,(s))\,s^2+\frac{-\text{Kt}\,\text{vapp0}\,\eta\text{N}^2-R\,\text{rapp0}\,\text{N}}{\text{Ke}\,\text{Kt}\,\eta\text{N}^2+B\,R\,\eta\text{N}^2+\text{Bafter}\,R}}{\text{N}}$$

```
(rawSolvedXforms = uniqueSolve[leqnsToSolve, outputXorms] // Simplify) // prettyPrint
```

$$\mathcal{L}_t[i(t)]\,(s) \to \frac{\text{Bafter}\,\text{vapp0}+\text{N}\,(B\,\text{vapp0}\,\eta\text{N}-\text{Ke}\,\text{rapp0})}{s\,((\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R)} + \frac{((B+J\,s)\,\eta\text{N}^2+\text{Bafter}+\text{Jafter}\,s)\,(\mathcal{L}_t[\Delta\text{vapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)} - \frac{\text{Ke}\,\text{N}\,(\mathcal{L}_t[\Delta\,\text{rapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)}$$
$$\mathcal{L}_t[\text{vg}(t)]\,(s) \to \text{Ke}\,\text{N}\left(\frac{\text{Kt}\,\text{vapp0}\,\eta\text{N}+R\,\text{rapp0}}{s\,((\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R)} + \frac{\text{Kt}\,\eta\,\text{N}\,(\mathcal{L}_t[\Delta\text{vapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)} + \frac{(R+L\,s)\,(\mathcal{L}_t[\Delta\,\text{rapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)}\right)$$
$$\mathcal{L}_t[\theta(t)]\,(s) \to \frac{\text{N}\left(\frac{\text{Kt}\,\text{vapp0}\,\eta\text{N}+R\,\text{rapp0}}{(\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R} + \frac{\text{Kt}\,s\,\eta\,\text{N}\,(\mathcal{L}_t[\Delta\text{vapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)} + \frac{s\,(R+L\,s)\,(\mathcal{L}_t[\Delta\,\text{rapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)}\right)}{s^2}$$
$$\mathcal{L}_t[\theta\text{after}(t)]\,(s) \to \frac{\frac{\text{Kt}\,\text{vapp0}\,\eta\text{N}+R\,\text{rapp0}}{(\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R} + \frac{\text{Kt}\,s\,\eta\,\text{N}\,(\mathcal{L}_t[\Delta\text{vapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)} + \frac{s\,(R+L\,s)\,(\mathcal{L}_t[\Delta\,\text{rapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)}}{s^2}$$
$$\mathcal{L}_t[\tau(t)]\,(s) \to \text{Kt}\left(\frac{\text{Bafter}\,\text{vapp0}+\text{N}\,(B\,\text{vapp0}\,\eta\text{N}-\text{Ke}\,\text{rapp0})}{s\,((\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R)} + \frac{((B+J\,s)\,\eta\text{N}^2+\text{Bafter}+\text{Jafter}\,s)\,(\mathcal{L}_t[\Delta\text{vapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)} - \frac{\text{Ke}\,\text{N}\,(\mathcal{L}_t[\Delta\,\text{rapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)}\right)$$
$$\mathcal{L}_t[\text{rafter}(t)]\,(s) \to -\frac{\text{rapp0}}{s} - \frac{(J\,\eta\text{N}^2+\text{Jafter})\,(\text{Kt}\,\text{vapp0}\,\eta\text{N}+R\,\text{rapp0})}{(\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R} - \mathcal{L}_t[\Delta\,\text{rapp}(t)]\,(s) + \frac{((B+J\,s)\,\eta\text{N}^2+\text{Bafter}+\text{Jafter}\,s)\,(((\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s))\,(\text{Kt}\,\text{v}}{s\,((\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R)}$$

We separate the transforms into applied-voltage- and applied-torque-dependent parts.

```
Clear[separateXforms]
separateXforms[xform_] := Module[{xformVarRules, invertedRules, xformVars, vard, rawCollected, coeffs, const, terms},
  xformVarRules = # → Unique["laplaceTransform"] & /@ inputXforms;
  invertedRules = Reverse /@ xformVarRules;
  xformVars = xformVarRules[[All, 2]];
  vard = xform /. xformVarRules;
  rawCollected = Collect[vard, xformVars] /. invertedRules;
  coeffs = Coefficient[rawCollected, #] & /@ inputXforms;
  const = Together[rawCollected /. {# → 0 & /@ inputXforms}];
  (* we multiply const by s to make the units come out right (?) *)
  const = {const[[1]] * s // Simplify};
  terms = const ~Join~ (#[[1]] * #[[2]] &/@ Transpose[{coeffs, inputXforms}]);
  Plus @@ terms
 ]
solvedXforms = #[[1]] → separateXforms[#[[2]]] & /@ rawSolvedXforms;
% // prettyPrint
```

$$\mathcal{L}_t[i(t)]\,(s) \to \frac{\text{Bafter}\,\text{vapp0}+\text{N}\,(B\,\text{vapp0}\,\eta\text{N}-\text{Ke}\,\text{rapp0})}{(\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R} + \frac{((B+J\,s)\,\eta\text{N}^2+\text{Bafter}+\text{Jafter}\,s)\,(\mathcal{L}_t[\Delta\text{vapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)} - \frac{\text{Ke}\,\text{N}\,(\mathcal{L}_t[\Delta\,\text{rapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)}$$
$$\mathcal{L}_t[\text{vg}(t)]\,(s) \to \frac{\text{Ke}\,\text{Kt}\,\eta\,(\mathcal{L}_t[\Delta\text{vapp}(t)]\,(s))\,\text{N}^2}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)} + \frac{\text{Ke}\,(\text{Kt}\,\text{vapp0}\,\eta\text{N}+R\,\text{rapp0})\,\text{N}}{(\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R} + \frac{\text{Ke}\,(R+L\,s)\,(\mathcal{L}_t[\Delta\,\text{rapp}(t)]\,(s))\,\text{N}}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)}$$
$$\mathcal{L}_t[\theta(t)]\,(s) \to \frac{\text{Kt}\,\eta\,(\mathcal{L}_t[\Delta\text{vapp}(t)]\,(s))\,\text{N}^2}{s\,((\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s))} + \frac{(\text{Kt}\,\text{vapp0}\,\eta\text{N}+R\,\text{rapp0})\,\text{N}}{s\,((\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R)} + \frac{(R+L\,s)\,(\mathcal{L}_t[\Delta\,\text{rapp}(t)]\,(s))\,\text{N}}{s\,((\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s))}$$
$$\mathcal{L}_t[\theta\text{after}(t)]\,(s) \to \frac{\text{Kt}\,\text{vapp0}\,\eta\text{N}+R\,\text{rapp0}}{s\,((\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R)} + \frac{\text{Kt}\,\eta\,\text{N}\,(\mathcal{L}_t[\Delta\text{vapp}(t)]\,(s))}{s\,((\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s))} + \frac{(R+L\,s)\,(\mathcal{L}_t[\Delta\,\text{rapp}(t)]\,(s))}{s\,((\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s))}$$
$$\mathcal{L}_t[\tau(t)]\,(s) \to \frac{\text{Kt}\,(\text{Bafter}\,\text{vapp0}+\text{N}\,(B\,\text{vapp0}\,\eta\text{N}-\text{Ke}\,\text{rapp0}))}{(\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R} + \frac{\text{Kt}\,((B+J\,s)\,\eta\text{N}^2+\text{Bafter}+\text{Jafter}\,s)\,(\mathcal{L}_t[\Delta\text{vapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)} - \frac{\text{Ke}\,\text{Kt}\,\text{N}\,(\mathcal{L}_t[\Delta\,\text{rapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)}$$
$$\mathcal{L}_t[\text{rafter}(t)]\,(s) \to \frac{\text{Kt}\,\eta\,\text{N}\,(\text{Bafter}\,\text{vapp0}+\text{N}\,(B\,\text{vapp0}\,\eta\text{N}-\text{Ke}\,\text{rapp0}))}{(\text{Ke}\,\text{Kt}+B\,R)\,\eta\text{N}^2+\text{Bafter}\,R} + \frac{\text{Kt}\,\eta\,\text{N}\,((B+J\,s)\,\eta\text{N}^2+\text{Bafter}+\text{Jafter}\,s)\,(\mathcal{L}_t[\Delta\text{vapp}(t)]\,(s))}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)} + \left(\frac{(R+L\,s)\,((B+J\,s)\,\eta\text{N}^2+\text{Bafter}+\text{Jafter}\,s)}{(\text{Ke}\,\text{Kt}+(B+J\,s)\,(R+L\,s))\,\eta\text{N}^2+\text{Bafter}\,(R+L\,s)+\text{Jafter}\,s\,(R+L\,s)}\right.$$

5.2. Transfer Function Models

We use the solved transforms to create TransferFunctionModel[]s for every variable of interest.

```
Clear[divideTerms]
divideTerms[sum_, termDivisors_] := Module[{list},
  list = List @@ sum;
  #[[1]] / #[[2]] & /@ Transpose[{list, termDivisors}]]
divideTerms[a x + b y, {x, y}]
```

```
{a, b}
```

```
Clear[makeModel]
makeModel[var_] := makeModel[var, ToString[var], 1]
makeModel[var_, outputLabel_, factor_] := TransferFunctionModel[
    factor * Function[{sum}, divideTerms[sum, {1} ~Join~ inputXforms]]
      [LaplaceTransform[var[t], t, s] /. solvedXforms],
    s,
    SystemsModelLabels → {inputLabels, outputLabel}] // FullSimplify // Simplify
```

```
motorPositionModel = makeModel[θ]
motorVelocityModel = makeModel[θ, "ω", s]
motorAccelerationModel = makeModel[θ, "α", s * s]
motorCurrentModel = makeModel[i]
motorEMFModel = makeModel[vg, "vg", 1]
motorTorqueModel = makeModel[τ]
```

$$\begin{array}{c|ccc} & 1 & \text{vapp} & \tau\text{app} \\ \theta & \frac{N\,(Kt\,vapp0\,\eta\,N + R\,\tau app0)}{\left(Bafter\,R + (Ke\,Kt + B\,R)\,\eta\,N^2\right)\,s} & \frac{Kt\,\eta\,N^2}{s\,\left(Ke\,Kt\,\eta\,N^2 + (R + L\,s)\,\left(Bafter + Jafter\,s + \eta\,N^2\,(B + J\,s)\right)\right)} & \frac{N\,(R + L\,s)}{s\,\left(Ke\,Kt\,\eta\,N^2 + (R + L\,s)\,\left(Bafter + Jafter\,s + \eta\,N^2\,(B\right.\right.} \end{array}$$

$$\begin{array}{c|ccc} & 1 & \text{vapp} & \tau\text{app} \\ \omega & \frac{N\,(Kt\,vapp0\,\eta\,N + R\,\tau app0)}{Bafter\,R + (Ke\,Kt + B\,R)\,\eta\,N^2} & \frac{Kt\,\eta\,N^2}{Ke\,Kt\,\eta\,N^2 + (R + L\,s)\,\left(Bafter + Jafter\,s + \eta\,N^2\,(B + J\,s)\right)} & \frac{N\,(R + L\,s)}{Ke\,Kt\,\eta\,N^2 + (R + L\,s)\,\left(Bafter + Jafter\,s + \eta\,N^2\,(B + J\,s)\right)} \end{array}$$

$$\begin{array}{c|ccc} & 1 & \text{vapp} & \tau\text{app} \\ \alpha & \frac{N\,(Kt\,vapp0\,\eta\,N + R\,\tau app0)\,s}{Bafter\,R + (Ke\,Kt + B\,R)\,\eta\,N^2} & \frac{Kt\,\eta\,N^2\,s}{Ke\,Kt\,\eta\,N^2 + (R + L\,s)\,\left(Bafter + Jafter\,s + \eta\,N^2\,(B + J\,s)\right)} & \frac{N\,s\,(R + L\,s)}{Ke\,Kt\,\eta\,N^2 + (R + L\,s)\,\left(Bafter + Jafter\,s + \eta\,N^2\,(B + J\,s)\right)} \end{array}$$

$$\begin{array}{c|ccc} & 1 & \text{vapp} & \tau\text{app} \\ i & \frac{Bafter\,vapp0 + N\,(B\,vapp0\,\eta\,N - Ke\,\tau app0)}{Bafter\,R + (Ke\,Kt + B\,R)\,\eta\,N^2} & \frac{Bafter + Jafter\,s + \eta\,N^2\,(B + J\,s)}{Ke\,Kt\,\eta\,N^2 + (R + L\,s)\,\left(Bafter + Jafter\,s + \eta\,N^2\,(B + J\,s)\right)} & -\frac{Ke\,N}{Ke\,Kt\,\eta\,N^2 + (R + L\,s)\,\left(Bafter + Jafter\,s + \eta\,N\right.} \end{array}$$

$$\begin{array}{c|ccc} & 1 & \text{vapp} & \tau\text{app} \\ vg & \frac{Ke\,N\,(Kt\,vapp0\,\eta\,N + R\,\tau app0)}{Bafter\,R + (Ke\,Kt + B\,R)\,\eta\,N^2} & \frac{Ke\,Kt\,\eta\,N^2}{Ke\,Kt\,\eta\,N^2 + (R + L\,s)\,\left(Bafter + Jafter\,s + \eta\,N^2\,(B + J\,s)\right)} & \frac{Ke\,N\,(R + L\,s)}{Ke\,Kt\,\eta\,N^2 + (R + L\,s)\,\left(Bafter + Jafter\,s + \eta\,N^2\,(B + J\,s)\right)} \end{array}$$

$$\begin{array}{c|ccc} & 1 & \text{vapp} & \tau\text{app} \\ \tau & \frac{Kt\,(Bafter\,vapp0 + N\,(B\,vapp0\,\eta\,N - Ke\,\tau app0))}{Bafter\,R + (Ke\,Kt + B\,R)\,\eta\,N^2} & \frac{Kt\,\left(Bafter + Jafter\,s + \eta\,N^2\,(B + J\,s)\right)}{Ke\,Kt\,\eta\,N^2 + (R + L\,s)\,\left(Bafter + Jafter\,s + \eta\,N^2\,(B + J\,s)\right)} & -\frac{Ke\,Kt\,N}{Ke\,Kt\,\eta\,N^2 + (R + L\,s)\,\left(Bafter + Jafter\right.} \end{array}$$

```
motorPositionAfterModel = makeModel[θafter]
motorVelocityAfterModel = makeModel[θafter, "ωafter", s]
motorAccelerationAfterModel = makeModel[θafter, "αafter", s * s]
motorTorqueAfterModel = makeModel[τafter]
```

$$\begin{pmatrix} & 1 & \text{vapp} & \tau\text{app} \\ \theta\text{after} & \frac{\text{Kt vapp0}\,\eta\,\text{N}+\text{R}\,\tau\text{app0}}{\left(\text{Bafter R}+\left(\text{Ke Kt}+\text{B R}\right)\,\eta\,\text{N}^2\right)\,\text{s}} & \frac{\text{Kt}\,\eta\,\text{N}}{\text{s}\,\left(\text{Ke Kt}\,\eta\,\text{N}^2+\left(\text{R}+\text{L s}\right)\,\left(\text{Bafter}+\text{Jafter s}+\eta\,\text{N}^2\,\left(\text{B}+\text{J s}\right)\right)\right)} & \frac{\text{R}+\text{L s}}{\text{s}\,\left(\text{Ke Kt}\,\eta\,\text{N}^2+\left(\text{R}+\text{L s}\right)\,\left(\text{Bafter}+\text{Jafter s}+\eta\,\text{N}\right.\right.} \end{pmatrix}$$

$$\begin{pmatrix} & 1 & \text{vapp} & \tau\text{app} \\ \omega\text{after} & \frac{\text{Kt vapp0}\,\eta\,\text{N}+\text{R}\,\tau\text{app0}}{\text{Bafter R}+\left(\text{Ke Kt}+\text{B R}\right)\,\eta\,\text{N}^2} & \frac{\text{Kt}\,\eta\,\text{N}}{\text{Ke Kt}\,\eta\,\text{N}^2+\left(\text{R}+\text{L s}\right)\,\left(\text{Bafter}+\text{Jafter s}+\eta\,\text{N}^2\,\left(\text{B}+\text{J s}\right)\right)} & \frac{\text{R}+\text{L s}}{\text{Ke Kt}\,\eta\,\text{N}^2+\left(\text{R}+\text{L s}\right)\,\left(\text{Bafter}+\text{Jafter s}+\eta\,\text{N}^2\,\left(\text{B}+\text{J s}\right)\right)} \end{pmatrix}$$

$$\begin{pmatrix} & 1 & \text{vapp} & \tau\text{app} \\ \alpha\text{after} & \frac{\left(\text{Kt vapp0}\,\eta\,\text{N}+\text{R}\,\tau\text{app0}\right)\,\text{s}}{\text{Bafter R}+\left(\text{Ke Kt}+\text{B R}\right)\,\eta\,\text{N}^2} & \frac{\text{Kt}\,\eta\,\text{N}\,\text{s}}{\text{Ke Kt}\,\eta\,\text{N}^2+\left(\text{R}+\text{L s}\right)\,\left(\text{Bafter}+\text{Jafter s}+\eta\,\text{N}^2\,\left(\text{B}+\text{J s}\right)\right)} & \frac{\text{s}\,\left(\text{R}+\text{L s}\right)}{\text{Ke Kt}\,\eta\,\text{N}^2+\left(\text{R}+\text{L s}\right)\,\left(\text{Bafter}+\text{Jafter s}+\eta\,\text{N}^2\,\left(\text{B}+\text{J s}\right)\right)} \end{pmatrix}$$

$$\begin{pmatrix} & 1 & \text{vapp} & \tau\text{app} \\ \tau\text{after} & \frac{\text{Kt}\,\eta\,\text{N}\,\left(\text{Bafter vapp0}+\text{N}\,\left(\text{B vapp0}\,\eta\,\text{N}-\text{Ke}\,\tau\text{app0}\right)\right)}{\text{Bafter R}+\left(\text{Ke Kt}+\text{B R}\right)\,\eta\,\text{N}^2} & \frac{\text{Kt}\,\eta\,\text{N}\,\left(\text{Bafter}+\text{Jafter s}+\eta\,\text{N}^2\,\left(\text{B}+\text{J s}\right)\right)}{\text{Ke Kt}\,\eta\,\text{N}^2+\left(\text{R}+\text{L s}\right)\,\left(\text{Bafter}+\text{Jafter s}+\eta\,\text{N}^2\,\left(\text{B}+\text{J s}\right)\right)} & -\frac{\text{Ke Kt}\,\eta\,\text{N}}{\text{Ke Kt}\,\eta\,\text{N}^2+\left(\text{R}+\text{L s}\right)\,\left(\text{Bafter}+\text{J}\right.} \end{pmatrix}$$

Save the models to a file for later use.

```
saveDefinitions[outputDirectory <> "MotorModels.m",
 {diffEqns, ΔdiffEqns, initialConditions, ΔinitialConditions, motorPositionModel,
  motorVelocityModel, motorAccelerationModel, motorCurrentModel, motorEMFModel, motorTorqueModel,
  motorPositionAfterModel, motorVelocityAfterModel, motorAccelerationAfterModel, motorTorqueAfterModel}]
```

5.3. Units Verification

We do a quick check on one of the transforms to see if units are internally consistent. Mathematica would complain if there were not (and has done so in the past, during development; indeed, that's why this check is here).

```
motorVelocityModel[s][[1]]
modelTest = % /. parameterQuantities /. s → Quantity[s, "per second"];
inputs = {1, Quantity[vapp, "Volts"], Quantity[τapp, siTorqueUnits]}
modelTest.inputs
```

$$\left\{\frac{\text{N}\,\left(\text{Kt vapp0}\,\eta\,\text{N}+\text{R}\,\tau\text{app0}\right)}{\text{Bafter R}+\left(\text{Ke Kt}+\text{B R}\right)\,\eta\,\text{N}^2},\ \frac{\text{Kt}\,\eta\,\text{N}^2}{\text{Ke Kt}\,\eta\,\text{N}^2+\left(\text{R}+\text{L s}\right)\,\left(\text{Bafter}+\text{Jafter s}+\left(\text{B}+\text{J s}\right)\,\eta\,\text{N}^2\right)},\ \frac{\left(\text{R}+\text{L s}\right)\,\text{N}}{\text{Ke Kt}\,\eta\,\text{N}^2+\left(\text{R}+\text{L s}\right)\,\left(\text{Bafter}+\text{Jafter s}+\left(\text{B}+\text{J s}\right)\,\eta\,\text{N}^2\right)}\right\}$$

$$\left\{1,\ \text{vapp V},\ \tau\text{app mN/rad}\right\}$$

$$\left(\frac{\text{Kt vapp}\,\eta\,\text{N}^2}{\text{Ke Kt}\,\eta\,\text{N}^2+\left(\text{R}+\text{L s}\right)\,\left(\text{Bafter}+\text{Jafter s}+\left(\text{B}+\text{J s}\right)\,\eta\,\text{N}^2\right)}+\frac{\left(\text{R}+\text{L s}\right)\,\text{N}\,\tau\text{app}}{\text{Ke Kt}\,\eta\,\text{N}^2+\left(\text{R}+\text{L s}\right)\,\left(\text{Bafter}+\text{Jafter s}+\left(\text{B}+\text{J s}\right)\,\eta\,\text{N}^2\right)}+\frac{\text{N}\,\left(\text{Kt vapp0}\,\eta\,\text{N}+\text{R}\,\tau\text{app0}\right)}{\text{Bafter R}+\left(\text{Ke Kt}+\text{B R}\right)\,\eta\,\text{N}^2}\right)\ \text{rad/s}$$

6. Time Domain Functions

The time domain functions we seek model various states of the motor as a direct function of time. Generally speaking, we may obtain them from the frequency domain by (matrix) multiplying the transfer function by the Laplace transform of the input signal, then taking the inverse Laplace transform of the result. However, if we're starting with the (frequency domain) transfer function and the (time domain) input signal in-hand, we can compute the time domain function more efficiently using the Laplace transform convolution theorem.

6.1. Time Domain Output Functions

6.1.1. makeTimeDomainFunctionConvolve[]

makeTimeDomainFunctionConvolve[] uses convolution and the theorem that the Laplace transform of a convolution is the product of the Laplace transform of the factors. Ref1: http://mathfaculty.fullerton.edu/mathews/c2003/laplaceconvolutionmod.html.

With the convolution approach, we can make explicit use of the fact that in the uses we have we know we want to take the inverse Laplace transform of a product, one of whose factors we already know the time domain inverse for, and thus we can avoid having LaplaceTransform[] do as much heavy lifting as we might ask it to do if we were to transform the input, multiply, then inverse-transform the whole result. This convolution approach also has been shown to be more efficient than the OutputRespose[] builtin function.

```
Clear[makeTimeDomainFunctionConvolve, convolve]
parameterAssumptions

convolve[fExpr_, gExpr_, exprVar_, t_] := Module[{τ(*=Unique["τ"]*)},
  Assuming[Union[{t ≥ 0}, parameterAssumptions], Integrate[(fExpr /. exprVar → τ) * (gExpr /. exprVar → t - τ), {τ, 0, t}]]
 ]

makeTimeDomainFunctionConvolve[model_, tInputFunctions_] := Module[
  {s(* = Unique["s"]*), τ(* = Unique["τ"]*), modelExpr},
  modelExpr = InverseLaplaceTransform[model[s], s, τ][[1]];
  Function[{t}, Module[{convolutions},
    convolutions = convolve[modelExpr[[#]], (tInputFunctions[[#]])[τ], τ, t] & /@ Range[1, Length[tInputFunctions]];
    {Total[convolutions]} (* put in list to mirror InverseLaplaceTransform[] output structure *)
   ]
  ]
 ]
```

```
{Bafter ∈ ℝ, Jafter ∈ ℝ, Null ∈ ℝ, vapp0 ∈ ℝ, ΔvappConst ∈ ℝ, ΔτappConst ∈ ℝ, τapp0 ∈ ℝ, i[_] ∈ ℝ, vg[_] ∈ ℝ, α[_] ∈ ℝ, Δvapp[_] ∈ ℝ,
 Δτapp[_] ∈ ℝ, Θ[_] ∈ ℝ, τ[_] ∈ ℝ, τafter[_] ∈ ℝ, ω[_] ∈ ℝ, Ke > 0, Kt > 0, L > 0, R > 0, η > 0, N > 0, B ≥ 0, J ≥ 0, t ≥ 0}
```

6.1.2. makeMotorTimeDomainFunction[]

makeMotorTimeDomainFunction[] takes a transfer function model and returns a function that, when provided with ea and τa time-domain input functions, returns a function of time.

```
Clear[makeMotorTimeDomainFunction]
makeMotorTimeDomainFunction[model_] := makeMotorTimeDomainFunction[model, makeTimeDomainFunctionConvolve]
makeMotorTimeDomainFunction[model_, builder_] := makeMotorTimeDomainFunction[model, builder] = (*cache the result*)
  Module[{t(*=Unique["t"]*)},
   Function[{vappfn, τappfn}, Module[{fn},
     fn = Function[{aTime},
       builder[model, {1 &, vappfn[#] &, τappfn[#] &}] [aTime] [[1]] (*unwrap output structure mentioned above*)];
     fn
    ]]]
```

6.1.3. Time Domain Output Functions for Motor State

We define time domain functions for all the motor state of interest.

```
Clear[motorPosition, motorVelocity, motorAcceleration, motorCurrent, motorEMF, motorTorque]
```

```
motorPosition = makeMotorTimeDomainFunction[motorPositionModel];
```

```
motorVelocity = makeMotorTimeDomainFunction[motorVelocityModel];
```

```
motorAcceleration = makeMotorTimeDomainFunction[motorAccelerationModel];
```

```
motorCurrent = makeMotorTimeDomainFunction[motorCurrentModel];
```

```
motorEMF = makeMotorTimeDomainFunction[motorEMFModel];
```

```
motorTorque = makeMotorTimeDomainFunction[motorTorqueModel];
```

The *output* too.

```
Clear[motorPositionAfter, motorVelocityAfter, motorAccelerationAfter, motorTorqueAfter]
```

```
motorPositionAfter = makeMotorTimeDomainFunction[motorPositionAfterModel];
```

```
motorVelocityAfter = makeMotorTimeDomainFunction[motorVelocityAfterModel];
```

```
motorAccelerationAfter = makeMotorTimeDomainFunction[motorAccelerationAfterModel];
```

```
motorTorqueAfter = makeMotorTimeDomainFunction[motorTorqueAfterModel];
```

Save these time domain functions so we can load them in later without having to develop them from scratch.

```
saveDefinitions[outputDirectory <> "MotorTimeDomainFunctions.m",
 {
  motorPosition, motorVelocity, motorAcceleration, motorCurrent, motorEMF, motorTorque,
  motorPositionAfter, motorVelocityAfter, motorAccelerationAfter,
  makeMotorTimeDomainFunction, makeTimeDomainFunctionConvolve
 }]
```

6.2. Time Domain Input Functions

6.2.1. Typical Input Examples

In analog to aMotor above, we now define some typical inputs. The first (and only, for now) is a constant 12V input with no externally applied torque.

```
applyTwelveVolts = Association[t → Quantity[t, "Seconds"],
  vapp0 → Quantity[0, "Volts"], τapp0 → Quantity[0, siTorqueUnits], ΔvappConst → Quantity[12, "Volts"]]
```

```
⟨|t → t s, vapp0 → 0 V, τapp0 → 0 mN/rad, ΔvappConst → 12 V|⟩
```

```
steadyTwelveVolts = Association[t → Quantity[t, "Seconds"],
  vapp0 → Quantity[12, "Volts"], τapp0 → Quantity[0, siTorqueUnits], ΔvappConst → Quantity[0, "Volts"]]
```

```
⟨|t → t s, vapp0 → 12 V, τapp0 → 0 mN/rad, ΔvappConst → 0 V|⟩
```

```
brakeFromTwelveVolts = Association[t → Quantity[t, "Seconds"],
  vapp0 → Quantity[12, "Volts"], τapp0 → Quantity[0, siTorqueUnits], ΔvappConst → Quantity[-12, "Volts"]]
```

```
⟨|t → t s, vapp0 → 12 V, τapp0 → 0 mN/rad, ΔvappConst → -12 V|⟩
```

6.3. Examples

We try out our time-domain functions on our example data. Note that the time axes in the plots here vary in scale.

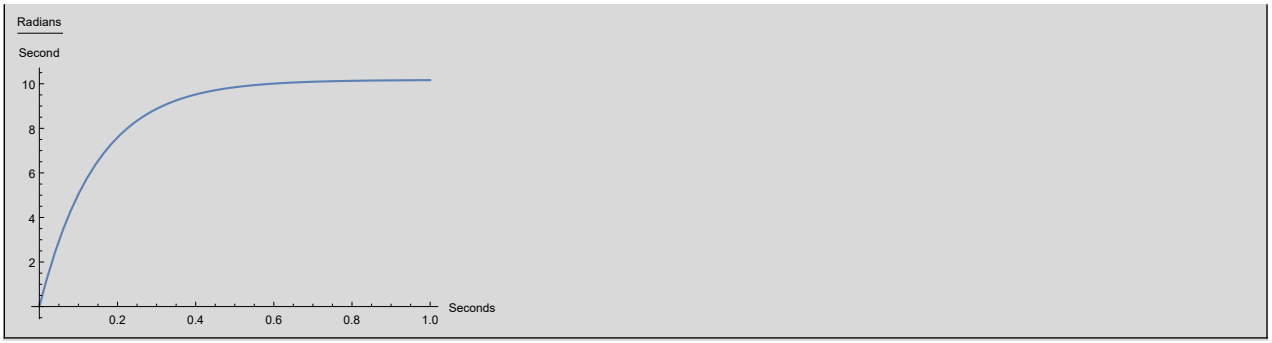
```
Clear[stepMotorTimeFunction, stepMotorTimeFunctionGeneric]
stepMotorTimeFunctionGeneric[motorTimeFunction_, t_] := stepMotorTimeFunctionGeneric[motorTimeFunction, t] = Module[{},
   motorTimeFunction[ΔvappConst &, ΔτappConst &][t]]

stepMotorTimeFunction[motorTimeFunction_, motorWithLoad_, input_, t_] :=
 stepMotorTimeFunction[motorTimeFunction, motorWithLoad, input, t] = Module[{generic, withUnits, unitless},
   generic = stepMotorTimeFunctionGeneric[motorTimeFunction, t];
   withUnits = generic /. (input ~Join~ motorWithLoad) // siUnits // N // FullSimplify;
   unitless = withUnits // clearUnits // FullSimplify;
   {generic, withUnits, unitless}]
```

6.3.1. Velocity, after gearbox

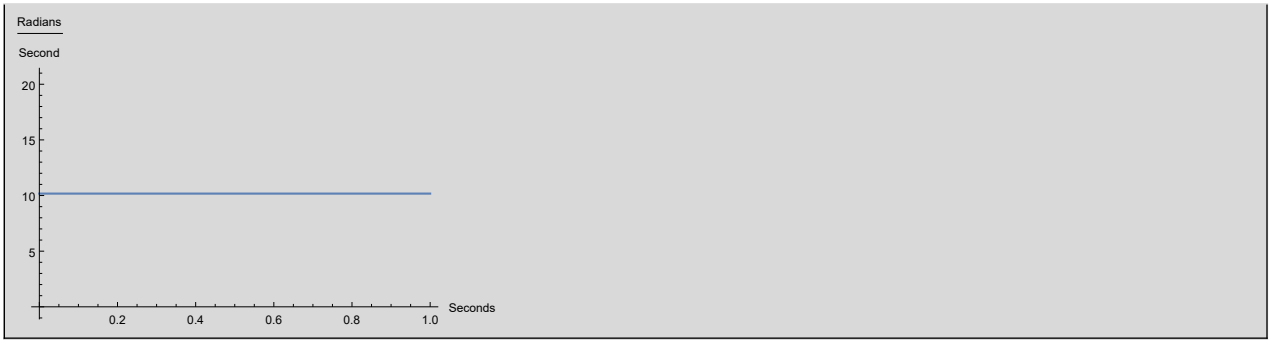
```
{velAfterStepGeneric, velAfterUnits, velAfter} =
  stepMotorTimeFunction[motorVelocityAfter, aMotorWithFlywheel, applyTwelveVolts, t];
velAfterUnits // N
Plot[velAfter, {t, 0, 1.0}, AxesLabel → {"Seconds", HoldForm[Radians / Second]}, PlotRange → All, AxesOrigin → {0, 0}]
```

$$2.71828^{-6.86584\,t}\left(-10.1885\ \text{rad/s}\right)+2.71828^{-4748.84\,t}\left(0.0147304\ \text{rad/s}\right)+10.1737\ \text{rad/s}$$



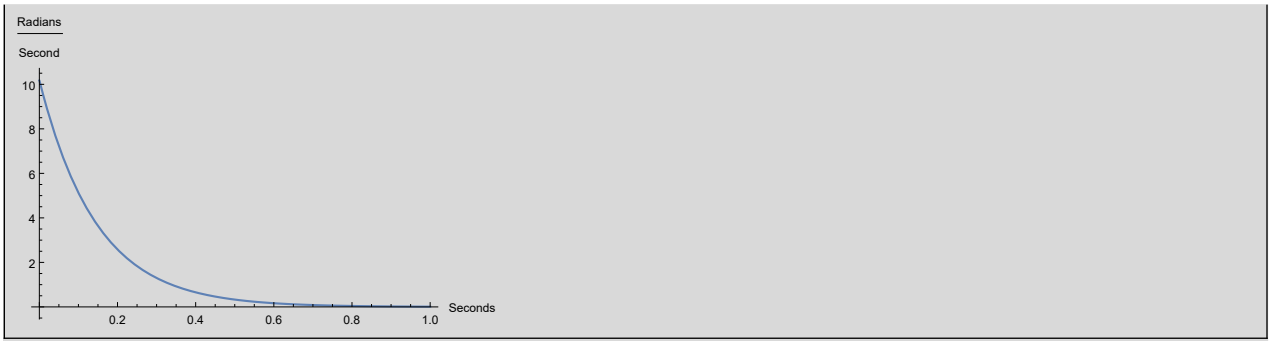

```
{steadyVelAfterStepGeneric, steadyVelAfterUnits, steadyVelAfter} =
  stepMotorTimeFunction[motorVelocityAfter, aMotorWithFlywheel, steadyTwelveVolts, t];
steadyVelAfterUnits // N
Plot[steadyVelAfter, {t, 0, 1.0}, AxesLabel → {"Seconds", HoldForm[Radians / Second]}, PlotRange → All, AxesOrigin → {0, 0}]
```

$$\left(2.71828^{-4748.84\,t}+2.71828^{-6.86584\,t}\right)\left(0.\ \text{rad/s}\right)+10.1737\ \text{rad/s}$$



```
{brakeVelAfterStepGeneric, brakeVelAfterUnits, brakeVelAfter} =
  stepMotorTimeFunction[motorVelocityAfter, aMotorWithFlywheel, brakeFromTwelveVolts, t];
brakeVelAfterUnits // N
Plot[brakeVelAfter, {t, 0, 1}, AxesLabel → {"Seconds", HoldForm[Radians / Second]}, PlotRange → All, AxesOrigin → {0, 0}]
```

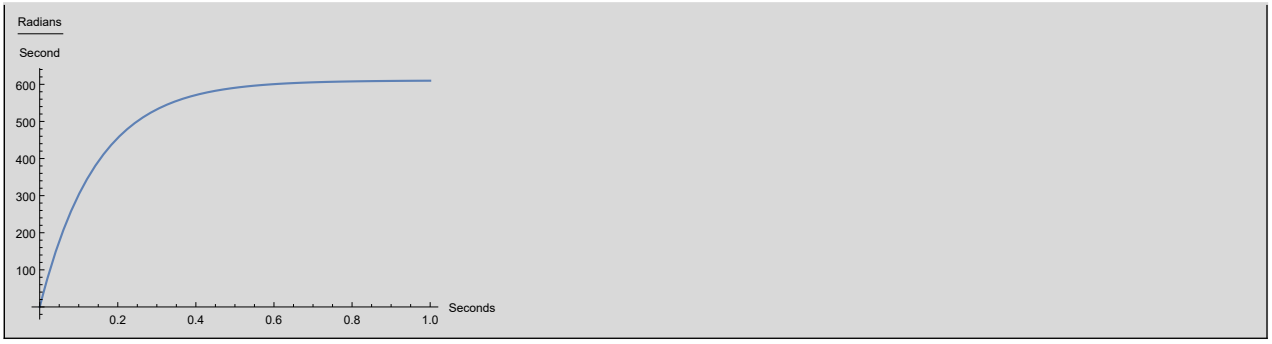
$$2.71828^{-4748.84\,t}\left(-0.0147304\ \text{rad/s}\right) + 7.85934\times10^{-14}\ \text{rad/s} + 2.71828^{-6.86584\,t}\left(10.1885\ \text{rad/s}\right)$$



6.3.2. Velocity, before gearbox

```
{velStepGeneric, velUnits, vel} = stepMotorTimeFunction[motorVelocity, aMotorWithFlywheel, applyTwelveVolts, t];
velUnits // N
Plot[vel, {t, 0, 1.0}, AxesLabel → {"Seconds", HoldForm[Radians / Second]}, PlotRange → All, AxesOrigin → {0, 0}]
```

$$2.71828^{-6.86584\,t}\left(-611.308\ \text{rad/s}\right) + 2.71828^{-4748.84\,t}\left(0.883825\ \text{rad/s}\right) + 610.424\ \text{rad/s}$$

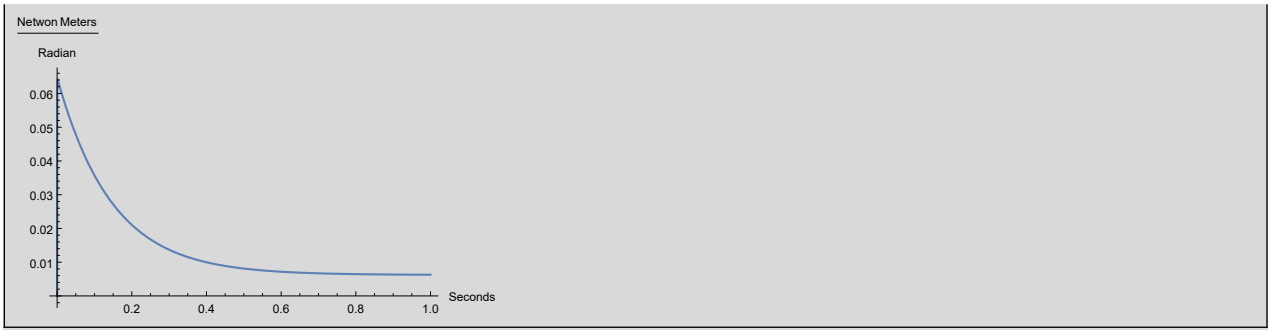


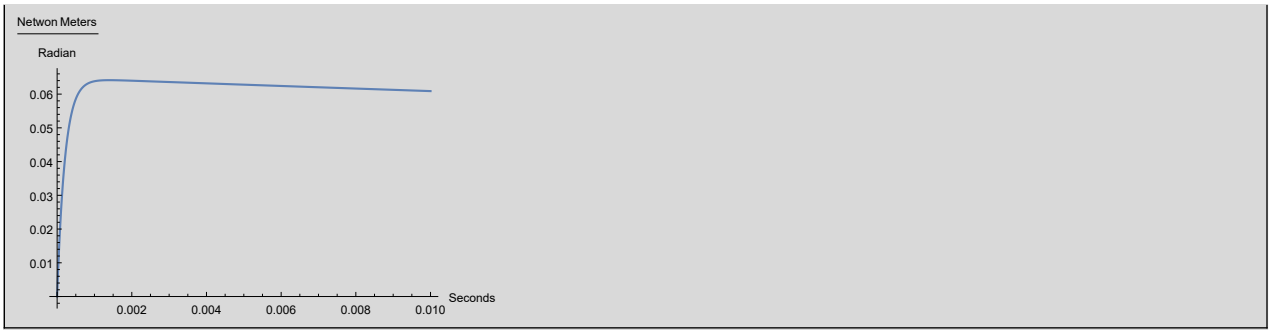
6.3.3. Torque, before gearbox

Note that the torque first rises, then descends.

```
{torStepGeneric, torUnits, tor} = stepMotorTimeFunction[motorTorque, aMotorWithFlywheel, applyTwelveVolts, t];
torUnits // N
Plot[tor, {t, 0, 1}, AxesLabel → {"Seconds", HoldForm[Netwon Meters / Radian]}, PlotRange → All, AxesOrigin → {0, 0}]
Plot[tor, {t, 0, .01}, AxesLabel → {"Seconds", HoldForm[Netwon Meters / Radian]}, PlotRange → All, AxesOrigin → {0, 0}]
```

$$2.71828^{-4748.84\,t}\left(-0.0647752\ \mathrm{kg\,m^2/(s^2 rad)}\right)+0.00621728\ \mathrm{kg\,m^2/(s^2 rad)}+2.71828^{-6.86584\,t}\left(0.0585579\ \mathrm{kg\,m^2/(s^2 rad)}\right)$$



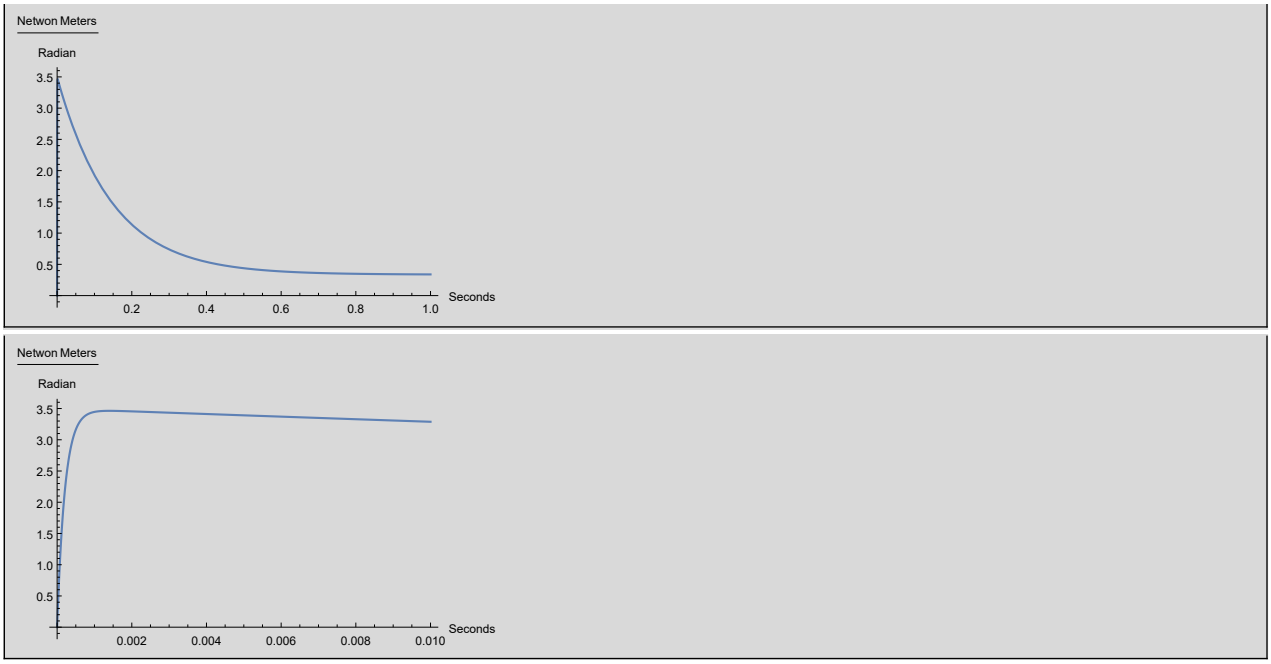


6.3.4. Torque, after gearbox

Note that the torque first rises, then descends.

```
{torAfterStepGeneric, torAfterUnits, torAfter} =
  stepMotorTimeFunction[motorTorqueAfter, aMotorWithFlywheel, applyTwelveVolts, t];
torAfterUnits // N
Plot[torAfter, {t, 0, 1}, AxesLabel → {"Seconds", HoldForm[Netwon Meters / Radian]}, PlotRange → All, AxesOrigin → {0, 0}]
Plot[torAfter, {t, 0, .01}, AxesLabel → {"Seconds", HoldForm[Netwon Meters / Radian]}, PlotRange → All, AxesOrigin → {0, 0}]
```

$$2.71828^{-4748.84\,t}\left(-3.49786\,\mathrm{kg\,m^2/(s^2 rad)}\right)+0.335733\,\mathrm{kg\,m^2/(s^2 rad)}+2.71828^{-6.86584\,t}\left(3.16213\,\mathrm{kg\,m^2/(s^2 rad)}\right)$$

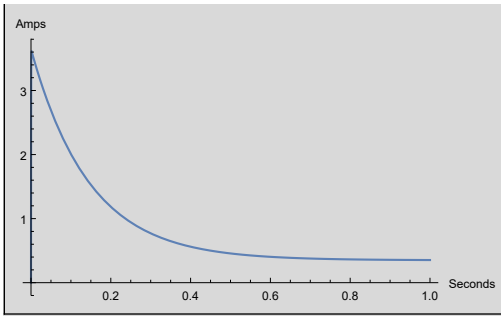


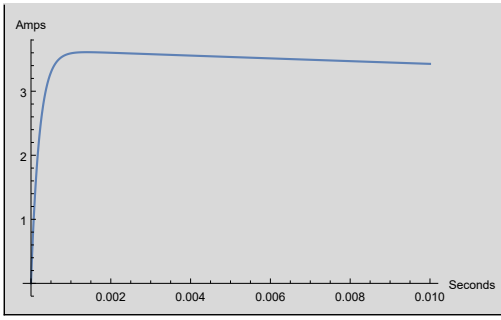
6.3.5. Current

Note that the current first rises, then descends.

```
{curStepGeneric, curUnits, cur} = stepMotorTimeFunction[motorCurrent, aMotorWithFlywheel, applyTwelveVolts, t];
curUnits // N
Plot[cur, {t, 0, 1}, AxesLabel → {"Seconds", HoldForm[Amps]}, PlotRange → All, AxesOrigin → {0, 0}]
Plot[cur, {t, 0, 0.01}, AxesLabel → {"Seconds", HoldForm[Amps]}, PlotRange → All, AxesOrigin → {0, 0}]
```

$$2.71828^{-4748.84\,t}\left(-3.64588\ \text{A}\right) + 0.349941\ \text{A} + 2.71828^{-6.86584\,t}\left(3.29594\ \text{A}\right)$$

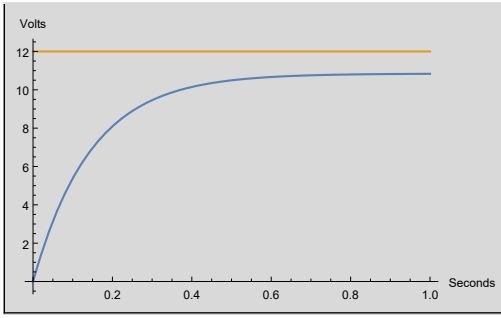




6.3.6. Back EMF

```
{emfStepGeneric, emfUnits, emf} = stepMotorTimeFunction[motorEMF, aMotorWithFlywheel, applyTwelveVolts, t];
emfUnits // N // simplifyUnits
Plot[{emf, ΔvappConst /. applyTwelveVolts}, {t, 0, 1},
 AxesLabel → {"Seconds", HoldForm[Volts]}, PlotRange → All, AxesOrigin → {0, 0}]
```

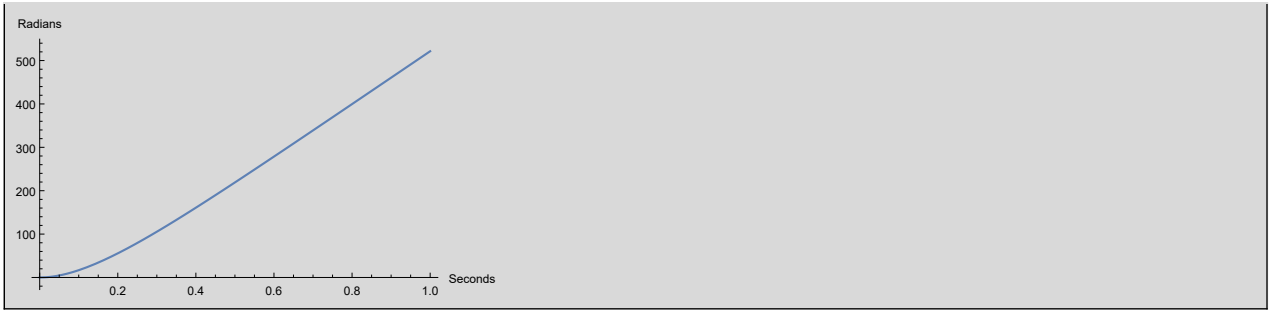
$$2.71828^{-6.86584\,t}\left(-10.8609\ \text{V}\right) + 2.71828^{-4748.84\,t}\left(0.0157026\ \text{V}\right) + 10.8452\ \text{V}$$



6.3.7. Position, before gearbox

```
{posStepGeneric, posUnits, pos} = stepMotorTimeFunction[motorPosition, aMotorWithFlywheel, applyTwelveVolts, t];
posUnits // N
Plot[pos, {t, 0, 1}, AxesLabel → {"Seconds", HoldForm[Radians]}, PlotRange → All, AxesOrigin → {0, 0}]
```

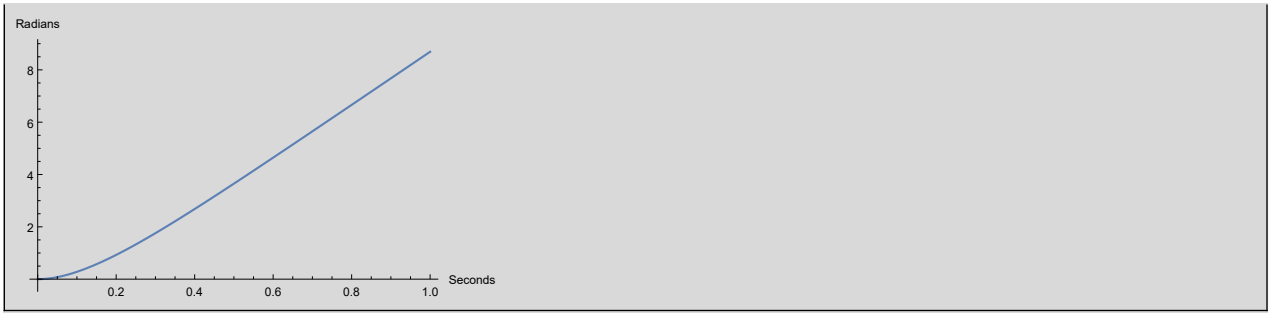
$$2.71828^{-4748.84\,t}\left(-0.000186114\ \text{rad}\right)+2.71828^{-6.86584\,t}\left(89.0361\ \text{rad}\right)+\left(-89.0359+610.424\,t\right)\ \text{rad}$$



6.3.8. Position, after gearbox

```
{posAfterStepGeneric, posAfterUnits, posAfter} =
  stepMotorTimeFunction[motorPositionAfter, aMotorWithFlywheel, applyTwelveVolts, t];
posAfterUnits // N
Plot[posAfter, {t, 0, 1}, AxesLabel → {"Seconds", HoldForm[Radians]}, PlotRange → All, AxesOrigin → {0, 0}]
```

$$2.71828^{-4748.84\,t}\left(-3.1019\times10^{-6}\ \text{rad}\right)+2.71828^{-6.86584\,t}\left(1.48394\ \text{rad}\right)+\left(-1.48393+10.1737\,t\right)\ \text{rad}$$



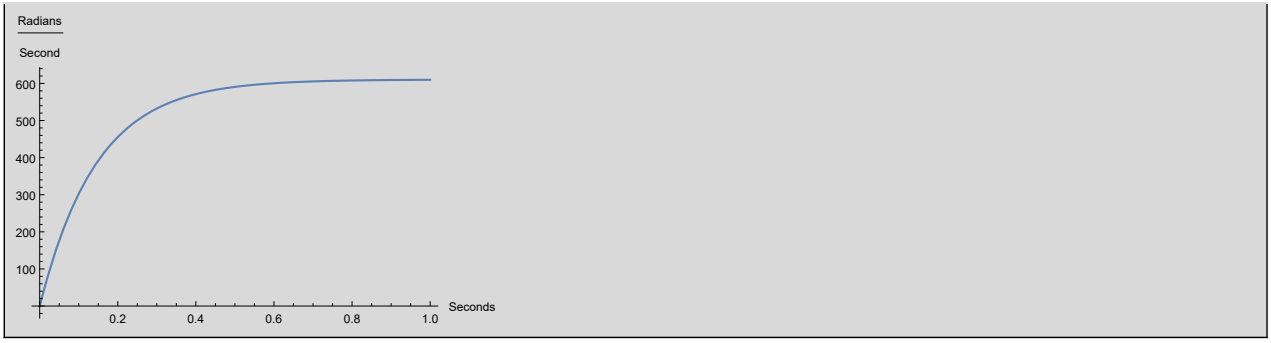
6.4.0. Examples with unitless loads

The unitless loads compute faster, but provide less insight.

6.4.1. With gear, using flywheel load

```
applyTwelveVolts // siUnits // clearUnits // N
{velUnits2StepGeneric, velUnits2, vel2} =
  stepMotorTimeFunction[motorVelocity, aUnitlessMotorWithFlywheel, applyTwelveVolts // siUnits // clearUnits // N, t];
Plot[vel2, {t, 0, 1}, AxesLabel → {"Seconds", HoldForm[Radians / Second]}, PlotRange → All, AxesOrigin → {0, 0}]
```

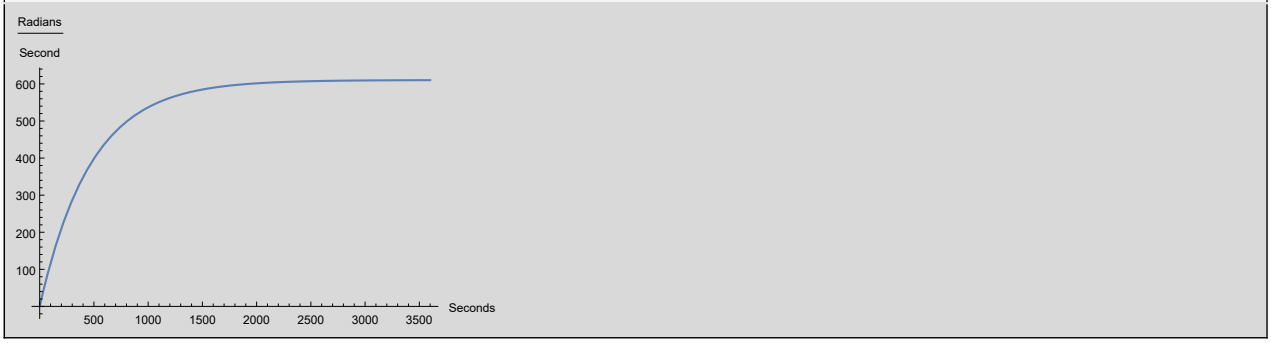
```
⟨|t → t, vapp0 → 0., τapp0 → 0., ΔvappConst → 12.|⟩
```



6.4.2. Without gear, using flywheel load

Note the time axis.

```
{velUnits3StepGeneric, velUnits3, vel3} =
  stepMotorTimeFunction[motorVelocity, aUnitlessMotorWithFlywheelNoGear, applyTwelveVolts // siUnits // clearUnits // N, t];
Plot[vel3, {t, 0, 3600}, AxesLabel → {"Seconds", HoldForm[Radians / Second]}, PlotRange → All, AxesOrigin → {0, 0}]
```



7. Steady State

7.1. Calculating Steady State Values : The Final Value Theorem

7.1.1. The general case

We explore the steady state behavior, the limit of the time domain functions as time goes to infinity.

One way to calculate said limit is to brute-force calculate the limit in question. While that works, and we can and have previously calculated same (Ref: https://github.com/rgatkinson/RobotPhysics/blob/master/MotorPhysics.pdf), it is very slow to do so.

A more efficient way to calculate the steady state behavior is to avail ourselves of the use of the Final Value Theorem (Ref: https://en.wikipedia.org/wiki/-Final_value_theorem). The final value theorem tells us that if the time-domain limit exists, it is equal to the limit of the Laplace transform of the time domain function as $s \to 0$.

```
Clear[ssFinalValueTheorem]
ssFinalValueTheorem[model_, inputs_] := Module[{s(*= Unique[]*), t (*= Unique[]*), xformedInputs, expr},
  xformedInputs = LaplaceTransform[#[t], t, s] &/@ inputs;
  expr = (model[s] .xformedInputs)[[1]];
  (*Limit is really slow if we include assumptions (!?), so don't*)
  Block[{$Assumptions = {}}, Limit[s expr, s → 0] // FullSimplify]
 ]
```

```
(ss = {
     ssPos → ssFinalValueTheorem[motorPositionModel, {1 &, ΔvappConst &, ΔτappConst &}] // Factor,
     ssVel → ssFinalValueTheorem[motorVelocityModel, {1 &, ΔvappConst &, ΔτappConst &}] // Factor,
     ssAcc → ssFinalValueTheorem[motorAccelerationModel, {1 &, ΔvappConst &, ΔτappConst &}] // Factor,
     ssEmf → ssFinalValueTheorem[motorEMFModel, {1 &, ΔvappConst &, ΔτappConst &}] // Factor,
     ssCur → ssFinalValueTheorem[motorCurrentModel, {1 &, ΔvappConst &, ΔτappConst &}] // Factor,
     ssTor → ssFinalValueTheorem[motorTorqueModel, {1 &, ΔvappConst &, ΔτappConst &}] // Factor,

     ssPosAfter → ssFinalValueTheorem[motorPositionAfterModel, {1 &, ΔvappConst &, ΔτappConst &}] // Factor,
     ssVelAfter → ssFinalValueTheorem[motorVelocityAfterModel, {1 &, ΔvappConst &, ΔτappConst &}] // Factor,
     ssAccAfter → ssFinalValueTheorem[motorAccelerationAfterModel, {1 &, ΔvappConst &, ΔτappConst &}] // Factor,
     ssTorAfter → ssFinalValueTheorem[motorTorqueAfterModel, {1 &, ΔvappConst &, ΔτappConst &}] // Factor
    } // FullSimplify) // prettyPrint
(simpleSs = ss /. {N → 1, η → 1, Bafter → 0, ΔτappConst → 0, τapp0 → 0, vapp0 → 0} // FullSimplify) // prettyPrint
```

$$\text{ssPos} \to \text{Indeterminate}$$
$$\text{ssVel} \to \frac{N\,(Kt\,(vapp0+\Delta vappConst)\,\eta\,N+R\,(\Delta\tau appConst+\tau app0))}{(Ke\,Kt+B\,R)\,\eta\,N^2+Bafter\,R}$$
$$\text{ssAcc} \to 0$$
$$\text{ssEmf} \to \frac{Ke\,N\,(Kt\,(vapp0+\Delta vappConst)\,\eta\,N+R\,(\Delta\tau appConst+\tau app0))}{(Ke\,Kt+B\,R)\,\eta\,N^2+Bafter\,R}$$
$$\text{ssCur} \to \frac{Bafter\,(vapp0+\Delta vappConst)+N\,(B\,(vapp0+\Delta vappConst)\,\eta\,N-Ke\,(\Delta\tau appConst+\tau app0))}{(Ke\,Kt+B\,R)\,\eta\,N^2+Bafter\,R}$$
$$\text{ssTor} \to \frac{Kt\,(Bafter\,(vapp0+\Delta vappConst)+N\,(B\,(vapp0+\Delta vappConst)\,\eta\,N-Ke\,(\Delta\tau appConst+\tau app0))))}{(Ke\,Kt+B\,R)\,\eta\,N^2+Bafter\,R}$$
$$\text{ssPosAfter} \to \text{Indeterminate}$$
$$\text{ssVelAfter} \to \frac{Kt\,(vapp0+\Delta vappConst)\,\eta\,N+R\,(\Delta\tau appConst+\tau app0)}{(Ke\,Kt+B\,R)\,\eta\,N^2+Bafter\,R}$$
$$\text{ssAccAfter} \to 0$$
$$\text{ssTorAfter} \to \frac{Kt\,\eta\,N\,(Bafter\,(vapp0+\Delta vappConst)+N\,(B\,(vapp0+\Delta vappConst)\,\eta\,N-Ke\,(\Delta\tau appConst+\tau app0)))}{(Ke\,Kt+B\,R)\,\eta\,N^2+Bafter\,R}$$

$$\text{ssPos} \to \text{Indeterminate}$$
$$\text{ssVel} \to \frac{Kt\,\Delta vappConst}{Ke\,Kt+B\,R}$$
$$\text{ssAcc} \to 0$$
$$\text{ssEmf} \to \frac{Ke\,Kt\,\Delta vappConst}{Ke\,Kt+B\,R}$$
$$\text{ssCur} \to \frac{B\,\Delta vappConst}{Ke\,Kt+B\,R}$$
$$\text{ssTor} \to \frac{B\,Kt\,\Delta vappConst}{Ke\,Kt+B\,R}$$
$$\text{ssPosAfter} \to \text{Indeterminate}$$
$$\text{ssVelAfter} \to \frac{Kt\,\Delta vappConst}{Ke\,Kt+B\,R}$$
$$\text{ssAccAfter} \to 0$$
$$\text{ssTorAfter} \to \frac{B\,Kt\,\Delta vappConst}{Ke\,Kt+B\,R}$$

The “Indeterminate” result regarding position is Mathematica informing us that it has proven to itself that the limit does not exist.

7.1.2. With specific examples

```
ss /. aUnitlessMotorWithFlywheel /. applyTwelveVolts // siUnits // clearUnits
```

```
{ssPos → Indeterminate, ssVel → 610.424, ssAcc → 0, ssEmf → 10.8452, ssCur → 0.349941,
 ssTor → 0.00621728, ssPosAfter → Indeterminate, ssVelAfter → 10.1737, ssAccAfter → 0, ssTorAfter → 0.335733}
```

```
ss /. aUnitlessMotorWithFlywheelNoGear /. applyTwelveVolts // siUnits // clearUnits
```

```
{ssPos → Indeterminate, ssVel → 610.424, ssAcc → 0, ssEmf → 10.8452, ssCur → 0.349941,
 ssTor → 0.00621728, ssPosAfter → Indeterminate, ssVelAfter → 610.424, ssAccAfter → 0, ssTorAfter → 0.00621728}
```

```
aMotorWithFlywheel
```

$$\left\langle\left| \mathtt{R} \to \frac{33}{10}\,\mathrm{kg\,m^2/(s^3A^2)}\,,\ \mathtt{L} \to \frac{347}{500\,000}\,\mathrm{kg\,m^2/(s^2A^2)}\,,\ \mathtt{N} \to 60,\ \eta \to \frac{9}{10},\ \mathtt{Ke} \to \frac{533}{30\,000}\,\mathrm{kg\,m^2/(s^2A\,rad)}\,,\right.\right.$$

$$\mathtt{Kt} \to \frac{533}{30\,000}\,\mathrm{kg\,m^2/(s^2A\,rad)}\,,\ \mathtt{B} \to \frac{11}{1\,080\,000}\,\mathrm{kg\,m^2/(s\,rad^2)}\,,\ \mathtt{J} \to \frac{347}{108\,000\,000\,000}\,\mathrm{kg\,m^2/rad^2}\,,$$

$$\left.\left.\mathtt{Jafter} \to \frac{1}{20}\,\mathrm{kg\,m^2/rad^2}\,,\ \mathtt{Bafter} \to 0\,\mathrm{kg\,m^2/(s\,rad^2)}\,,\ \Delta\tau\mathtt{appConst} \to 0\,\mathrm{kg\,m^2/(s^2rad)}\right|\right\rangle$$

7.2. On the Applicability of the Final Value Theorem

Can we always use the Final Value Theorem?

No, the Final Value Theorem only applies under certain conditions. Specifically: that all non-zero poles of the transfer function must have negative real parts, and the transfer function must have at most one pole at the origin (Ref: http://www-personal.umich.edu/~dsbaero/others/39-FVTrevisited.pdf). The predicate exhibited in the ConditionalExpression of the brute-force result captures exactly these conditions. When using the Final Value Theorem on any concrete system, we will need to test the conditions in any particular application. Lets illustrate by looking at the velocity model.

```
motorVelocityModel
(velPoles = TransferFunctionPoles[motorVelocityModel] // Flatten // Union)
```

$$\left(\begin{array}{c|ccc} & 1 & \mathtt{vapp} & \tau\mathtt{app} \\ \omega & \dfrac{\mathtt{N}\,(\mathtt{Kt}\,\mathtt{vapp0}\,\eta\,\mathtt{N}+\mathtt{R}\,\tau\mathtt{app0})}{\mathtt{Bafter}\,\mathtt{R}+(\mathtt{Ke}\,\mathtt{Kt}+\mathtt{B}\,\mathtt{R})\,\eta\,\mathtt{N}^2} & \dfrac{\mathtt{Kt}\,\eta\,\mathtt{N}^2}{\mathtt{Ke}\,\mathtt{Kt}\,\eta\,\mathtt{N}^2+(\mathtt{R}+\mathtt{L}\,\mathtt{s})\,(\mathtt{Bafter}+\mathtt{Jafter}\,\mathtt{s}+\eta\,\mathtt{N}^2\,(\mathtt{B}+\mathtt{J}\,\mathtt{s}))} & \dfrac{\mathtt{N}\,(\mathtt{R}+\mathtt{L}\,\mathtt{s})}{\mathtt{Ke}\,\mathtt{Kt}\,\eta\,\mathtt{N}^2+(\mathtt{R}+\mathtt{L}\,\mathtt{s})\,(\mathtt{Bafter}+\mathtt{Jafter}\,\mathtt{s}+\eta\,\mathtt{N}^2\,(\mathtt{B}+\mathtt{J}\,\mathtt{s}))} \end{array}\right)$$

$$\left\{\left(-\mathtt{Bafter}\,\mathtt{L}-\mathtt{Jafter}\,\mathtt{R}-\mathtt{B}\,\mathtt{L}\,\eta\,\mathtt{N}^2-\mathtt{J}\,\mathtt{R}\,\eta\,\mathtt{N}^2-\sqrt{\left(-4\left(\mathtt{Jafter}\,\mathtt{L}+\mathtt{J}\,\mathtt{L}\,\eta\,\mathtt{N}^2\right)\left(\mathtt{Bafter}\,\mathtt{R}+\mathtt{Ke}\,\mathtt{Kt}\,\eta\,\mathtt{N}^2+\mathtt{B}\,\mathtt{R}\,\eta\,\mathtt{N}^2\right)+\left(\mathtt{Bafter}\,\mathtt{L}+\mathtt{Jafter}\,\mathtt{R}+\mathtt{B}\,\mathtt{L}\,\eta\,\mathtt{N}^2+\mathtt{J}\,\mathtt{R}\,\eta\,\mathtt{N}^2\right)^2\right)}\right)\Big/\left(2\left(\mathtt{Jafter}\,\mathtt{L}+\mathtt{J}\,\mathtt{L}\,\eta\,\mathtt{N}^2\right)\right),\right.$$
$$\left.\left(-\mathtt{Bafter}\,\mathtt{L}-\mathtt{Jafter}\,\mathtt{R}-\mathtt{B}\,\mathtt{L}\,\eta\,\mathtt{N}^2-\mathtt{J}\,\mathtt{R}\,\eta\,\mathtt{N}^2+\sqrt{\left(-4\left(\mathtt{Jafter}\,\mathtt{L}+\mathtt{J}\,\mathtt{L}\,\eta\,\mathtt{N}^2\right)\left(\mathtt{Bafter}\,\mathtt{R}+\mathtt{Ke}\,\mathtt{Kt}\,\eta\,\mathtt{N}^2+\mathtt{B}\,\mathtt{R}\,\eta\,\mathtt{N}^2\right)+\left(\mathtt{Bafter}\,\mathtt{L}+\mathtt{Jafter}\,\mathtt{R}+\mathtt{B}\,\mathtt{L}\,\eta\,\mathtt{N}^2+\mathtt{J}\,\mathtt{R}\,\eta\,\mathtt{N}^2\right)^2\right)}\right)\Big/\left(2\left(\mathtt{Jafter}\,\mathtt{L}+\mathtt{J}\,\mathtt{L}\,\eta\,\mathtt{N}^2\right)\right)\right\}$$

In order to get a feel for where these zeros are in practice, let's look at them for an actual example motor. We compute where the poles occur for one of our examples;

```
velPoles /. aMotorWithFlywheel // siUnits // clearUnits // N
```

```
{-4748.84, -6.86584}
```

This, for this example, the Final Value Theorem applies (at least for velocity). As another example, adding a ridiculous additional load moves the poles, but they remain in the LHS of the complex plane.

```
velPoles /. addMotorLoad[aMotor, flywheel[Quantity[1000, "kg"], Quantity[1, "m"]]] // siUnits // clearUnits // N
```

```
{-4755.04, -0.000685831}
```

An analogous analytic development awaits a future revision to this document, as do examples that involve gears or externally applied torque.

8. More Administrivia

```
saveDefinitions[outputDirectory <> "Misc.m",
 {importMotorData, motorParameters, motorLoad, flywheel, addMotorLoad, massOnPulley,
  stepMotorTimeFunctionGeneric, stepMotorTimeFunction, ss, simpleSs,
  velAfterStepGeneric, velStepGeneric, torAfterStepGeneric,
  torStepGeneric, curStepGeneric, emfStepGeneric, posStepGeneric, posAfterStepGeneric
 }]
```

9. Revision History

- 2018.07.23 Added acceleration-dependant external torque. Corrected “viscous friction” to “viscous drag”. Clarified usage of units.
- 2018.08.06 Added gearbox support, removed time consuming internal tests. Regularized state variable names. Removed (flawed) “Back EMF vs Applied Voltage” discussion
- 2018.08.06 Renamed variables. Remove setting $Assumptions (that can slow things down)
- 2018.08.11 Added initial conditions support
- 2018.08.12 Bug fixes, miscellaneous polish
- 2018.08.13 Introduced ‘step’ terminology
- 2018.08.16 Adjusted to new schema in motor parameters spreadsheet