Motor Physics

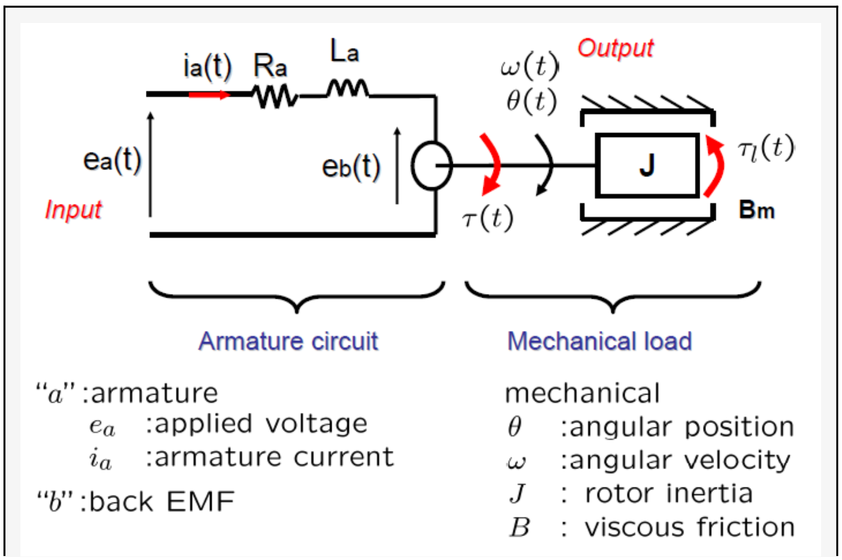
Robert Atkinson, Maddy Nguyen, Samantha Hordyk, Cole Welch, and John Fraser
20 July 2018

An exploration of the physics of motors. We begin by loading some handy utilities.

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

Core Motor Model

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



In addition to the the above, we allow here for an externally applied torque. At the moment, only a torque not dependent on other state (such as angular position) is modelled (e.g. a constant externally applied torque, but not one dependent on θ''); while useful for steady-state analysis, this will need to be enhanced in future.

With that considered, the state of the system can be described by the following variables.

- *ea,* the applied voltage
- *i,* the armature current
- *e,* the back EMF
- θ, the angular position
- τ, the torque driven by the armature current
- τa, the externally applied torque, if any

The parameter constants that characterize the motor are the following

- J, Moment of inertia
- b, Motor viscous friction constant
- Ke, Electromotive force constant
- Kt, Motor torque constant
- R, Electric resistance
- L Electric inductance

Further, the moment of inertia J and viscous friction constant b can be decomposed into additive contributions from both the motor and the load; we'll return to that later.

```
motorParameterNames = {J, b, Ke, Kt, R, L}
```

```
{J, b, Ke, Kt, R, L}
```

Differential Equations

Differential Equation Development

The system can be modelled as a set of differential equations

```
diffEqns = {}
```

```
{}
```

Drive torque is proportional to current.

```
diffEqns = Union[diffEqns, {τ[t] == Kt i[t]}]
```

```
{τ[t] == Kt i[t]}
```

The back EMF is proportional to velocity.

```
diffEqns = Union[diffEqns, {e[t] == Ke θ'[t]}]
```

```
{e[t] == Ke θ′[t], τ[t] == Kt i[t]}
```

Viscous friction reduces the torque, and what's left accelerates the inertial mass.

```
diffEqns = Union[diffEqns, {τ[t] + τa[t] - b θ'[t] == J θ''[t]}]
```

```
{e[t] == Ke θ′[t], τ[t] == Kt i[t], τ[t] + τa[t] - b θ′[t] == J θ″[t]}
```

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

```
diffEqns = Union[diffEqns, {ea[t] == R i[t] + L i'[t] + e[t]}]
```

```
{e[t] == Ke θ′[t], ea[t] == e[t] + R i[t] + L i′[t],
 τ[t] == Kt i[t], τ[t] + τa[t] - b θ′[t] == J θ″[t]}
```

```
diffEqns // prettyPrint
```

$e(t) = \mathrm{Ke}\,\theta'(t)$
$\mathrm{ea}(t) = e(t) + R\,i(t) + L\,i'(t)$
$\tau(t) = \mathrm{Kt}\,i(t)$
$\tau(t) + \tau\mathrm{a}(t) - b\,\theta'(t) = J\,\theta''(t)$

Initial Conditions

Some of the initial conditions we can intuit intellectually:

```
initialConditions = {
    θ[0] == 0,
    i[0] == 0,
    e[0] == 0,
    ea[0] == ν,
    τa[0] == T
  };
```

Others we can solve for:

```
others = {i'[0], θ'[0], θ''[0], τ[0]};
diffEqns /. t → 0 /. (initialConditions /. Equal → Rule)
(allInitialConditions =
   (uniqueSolve[%, others] /. Rule → Equal) ~Join~ initialConditions) // prettyPrint
```

```
{0 == Ke θ′[0], ν == L i′[0], τ[0] == 0, T + τ[0] - b θ′[0] == J θ″[0]}
```

$i'(0) = \frac{v}{L}$
$\theta'(0) = 0$
$\theta''(0) = \frac{T}{J}$
$\tau(0) = 0$
$\theta(0) = 0$
$i(0) = 0$
$e(0) = 0$
$\mathrm{ea}(0) = v$
$\tau\mathrm{a}(0) = \mathrm{T}$

Motor Parameters

We now take a moment to carefully work out the units of each parameter and variable. Some units are obvious and clear. Let’s write those down.

```
parameterUnits = {
   R → parseUnit["Ohms"],
   L → parseUnit["Henrys"],
   θ[t] → parseUnit["Radians"],
   θ'[t] → parseUnit["Radians per Second"],
   Ω[t] → parseUnit["Radians per Second"],
   θ''[t] → parseUnit["Radians per Second per Second"],
   α[t] → parseUnit["Radians per Second per Second"],
   i[t] → parseUnit["Amperes"],
   i'[t] → parseUnit["Amperes"] / parseUnit["Second"],
   i''[t] → parseUnit["Amperes"] / parseUnit["Second"] / parseUnit["Second"],
   ea[t] → parseUnit["Volts"],
   e[t] → parseUnit["Volts"],
   τ[t] → parseUnit["Newton Meters / Radians"],
   τa[t] → parseUnit["Newton Meters / Radians"]
  } // Association
unitsToQuantities[units_] := Module[{rules, makeQuantity},
  rules = Normal[units];
  makeQuantity = Function[{param, unit}, Quantity[param, unit]];
  (#[[1]] → makeQuantity @@ # & /@ rules) // Association
 ]
parameterQuantities = unitsToQuantities[parameterUnits]
```

$\langle| R \to \text{Ohms}, L \to \text{Henries}, \theta[t] \to \text{Radians}, \theta'[t] \to \frac{\text{Radians}}{\text{Seconds}}, \Omega[t] \to \frac{\text{Radians}}{\text{Seconds}}, \theta''[t] \to \frac{\text{Radians}}{\text{Seconds}^2}, \alpha[t] \to \frac{\text{Radians}}{\text{Seconds}^2}, i[t] \to \text{Amperes}, i'[t] \to \frac{\text{Amperes}}{\text{Seconds}}, i''[t] \to \frac{\text{Amperes}}{\text{Seconds}^2}, ea[t] \to \text{Volts}, e[t] \to \text{Volts}, \tau[t] \to \frac{\text{Meters Newtons}}{\text{Radians}}, \tau a[t] \to \frac{\text{Meters Newtons}}{\text{Radians}} |\rangle$

$\langle| R \to R\,\Omega, L \to L\,\text{H}, \theta[t] \to \theta[t]\,\text{rad}, \theta'[t] \to \theta'[t]\,\text{rad/s}, \Omega[t] \to \Omega[t]\,\text{rad/s}, \theta''[t] \to \theta''[t]\,\text{rad/s}^2, \alpha[t] \to \alpha[t]\,\text{rad/s}^2, i[t] \to i[t]\,\text{A}, i'[t] \to i'[t]\,\text{A/s}, i''[t] \to i''[t]\,\text{A/s}^2, ea[t] \to ea[t]\,\text{V}, e[t] \to e[t]\,\text{V}, \tau[t] \to \tau[t]\,\text{mN/rad}, \tau a[t] \to \tau a[t]\,\text{mN/rad} |\rangle$

Let’s look at how far that takes us:

```
diffEqns /. parameterQuantities // ColumnForm
```

$$e[t]\,\mathrm{V} == Ke\left(\theta'[t]\,\mathrm{rad/s}\right)$$
$$ea[t]\,\mathrm{V} == \left(e[t] + R\,i[t] + L\,i'[t]\right)\mathrm{A\Omega}$$
$$\tau[t]\,\mathrm{mN/rad} == Kt\left(i[t]\,\mathrm{A}\right)$$
$$\left(\tau[t] + \tau a[t]\right)\mathrm{mN/rad} + b\left(-\theta'[t]\,\mathrm{rad/s}\right) == J\left(\theta''[t]\,\mathrm{rad/s^2}\right)$$

We can see that if we write down the units for either b or J, the remaining units will be fully determined. We choose to specify J.

```
parameterUnits[J] = parseUnit["kg m^2"] / parseUnit["Radians^2"];
parameterQuantities = unitsToQuantities[parameterUnits]
diffEqns /. parameterQuantities // ColumnForm
remainingUnits = uniqueSolve[diffEqns[[{1, 3, 4}]], {Ke, Kt, b}]
```

$$\langle| R \to R\,\Omega,\ L \to L\,\mathrm{H},\ \theta[t] \to \theta[t]\,\mathrm{rad},\ \theta'[t] \to \theta'[t]\,\mathrm{rad/s},$$
$$\Omega[t] \to \Omega[t]\,\mathrm{rad/s},\ \theta''[t] \to \theta''[t]\,\mathrm{rad/s^2},\ \alpha[t] \to \alpha[t]\,\mathrm{rad/s^2},$$
$$i[t] \to i[t]\,\mathrm{A},\ i'[t] \to i'[t]\,\mathrm{A/s},\ i''[t] \to i''[t]\,\mathrm{A/s^2},\ ea[t] \to ea[t]\,\mathrm{V},$$
$$e[t] \to e[t]\,\mathrm{V},\ \tau[t] \to \tau[t]\,\mathrm{mN/rad},\ \tau a[t] \to \tau a[t]\,\mathrm{mN/rad},\ J \to J\,\mathrm{kg\,m^2/rad^2} |\rangle$$

$$e[t]\,\mathrm{V} == Ke\left(\theta'[t]\,\mathrm{rad/s}\right)$$
$$ea[t]\,\mathrm{V} == \left(e[t] + R\,i[t] + L\,i'[t]\right)\mathrm{A\Omega}$$
$$\tau[t]\,\mathrm{mN/rad} == Kt\left(i[t]\,\mathrm{A}\right)$$
$$\left(\tau[t] + \tau a[t]\right)\mathrm{mN/rad} + b\left(-\theta'[t]\,\mathrm{rad/s}\right) == J\,\theta''[t]\,\mathrm{kg\,m^2/(s^2 rad)}$$

$$\left\{Ke \to \frac{e[t]}{\theta'[t]},\ Kt \to \frac{\tau[t]}{i[t]},\ b \to \frac{\tau[t] + \tau a[t] - J\,\theta''[t]}{\theta'[t]}\right\}$$

Let’s check how those work.

```
remainingUnits /. parameterQuantities
```

$$\left\{Ke \to \frac{e[t]}{\theta'[t]}\,\mathrm{sV/rad},\ Kt \to \frac{\tau[t]}{i[t]}\,\mathrm{mN/(A\,rad)},\ b \to \frac{\tau[t] + \tau a[t] - J\,\theta''[t]}{\theta'[t]}\,\mathrm{m\,s\,N/rad^2}\right\}$$

Huzzah! They check out (if the units were incompatible, Mathematica would’ve told us). Let’s the remaining units to our kit.

```
(parameterUnits[#] = QuantityUnit[
     # /. (remainingUnits /. parameterQuantities)]) & /@ remainingUnits[[All, 1]]
parameterQuantities = unitsToQuantities[parameterUnits]
```

$$\left\{\frac{\text{Seconds Volts}}{\text{Radians}}, \frac{\text{Meters Newtons}}{\text{Amperes Radians}}, \frac{\text{Meters Newtons Seconds}}{\text{Radians}^2}\right\}$$

```
⟨|R → R Ω, L → L H, θ[t] → θ[t] rad, θ′[t] → θ′[t] rad/s,
 Ω[t] → Ω[t] rad/s, θ″[t] → θ″[t] rad/s², α[t] → α[t] rad/s²,
 i[t] → i[t] A, i′[t] → i′[t] A/s, i″[t] → i″[t] A/s², ea[t] → ea[t] V,
 e[t] → e[t] V, τ[t] → τ[t] mN/rad, τa[t] → τa[t] mN/rad,
 J → J kg m²/rad², Ke → Ke s V/rad, Kt → Kt m N/(A rad), b → b m s N/rad²|⟩
```

Laplace Transforms and Transfer Function Models

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 LaplaceTransform[] function to each equation. It automatically makes use of the linearity 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] & /@ diffEqns
leqns // prettyPrint
```

```
{LaplaceTransform[e[t], t, s] == Ke (s LaplaceTransform[θ[t], t, s] - θ[0]),
 LaplaceTransform[ea[t], t, s] == LaplaceTransform[e[t], t, s] +
   R LaplaceTransform[i[t], t, s] + L (-i[0] + s LaplaceTransform[i[t], t, s]),
 LaplaceTransform[τ[t], t, s] == Kt LaplaceTransform[i[t], t, s],
 LaplaceTransform[τ[t], t, s] + LaplaceTransform[τa[t], t, s] -
   b (s LaplaceTransform[θ[t], t, s] - θ[0]) ==
  J (s² LaplaceTransform[θ[t], t, s] - s θ[0] - θ′[0])}
```

$$\mathcal{L}_t[e(t)](s) = \text{Ke}\,(s\,(\mathcal{L}_t[\theta(t)](s)) - \theta(0))$$
$$\mathcal{L}_t[\text{ea}(t)](s) = \mathcal{L}_t[e(t)](s) + R\,(\mathcal{L}_t[i(t)](s)) + L\,(s\,(\mathcal{L}_t[i(t)](s)) - i(0))$$
$$\mathcal{L}_t[\tau(t)](s) = \text{Kt}\,(\mathcal{L}_t[i(t)](s))$$
$$\mathcal{L}_t[\tau(t)](s) + \mathcal{L}_t[\tau\text{a}(t)](s) - b\,(s\,(\mathcal{L}_t[\theta(t)](s)) - \theta(0)) = J\left((\mathcal{L}_t[\theta(t)](s))\,s^2 - \theta(0)\,s - \theta'(0)\right)$$

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[e[t], t, s], LaplaceTransform[ea[t], t, s],
 LaplaceTransform[i[t], t, s], LaplaceTransform[θ[t], t, s],
 LaplaceTransform[τ[t], t, s], LaplaceTransform[τa[t], t, s]}
```

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

```
voltageXform = LaplaceTransform[ea[t], t, s];
τinXform = LaplaceTransform[τa[t], t, s]
outputXorms = Complement[allXforms, {voltageXform, τinXform}]
solvedXforms = uniqueSolve[leqns, outputXorms]
solvedXforms =
 solvedXforms /. (allInitialConditions /. Equal → Rule) // FullSimplify
```

```
LaplaceTransform[τa[t], t, s]
```

```
{LaplaceTransform[e[t], t, s], LaplaceTransform[i[t], t, s],
 LaplaceTransform[θ[t], t, s], LaplaceTransform[τ[t], t, s]}
```

```
{LaplaceTransform[e[t], t, s] →
  -((-Ke Kt L i[0] - Ke Kt LaplaceTransform[ea[t], t, s] - Ke R LaplaceTransform[τa[t],
          t, s] - Ke L s LaplaceTransform[τa[t], t, s] - J Ke R θ′[0] - J Ke L s θ′[0]) /
     (Ke Kt + b R + b L s + J R s + J L s^2)), LaplaceTransform[i[t], t, s] →
  -((-b L i[0] - J L s i[0] - b LaplaceTransform[ea[t], t, s] -
       J s LaplaceTransform[ea[t], t, s] + Ke LaplaceTransform[τa[t], t, s] +
       J Ke θ′[0]) / (Ke Kt + b R + b L s + J R s + J L s^2)),
 LaplaceTransform[θ[t], t, s] → -1/(s (Ke Kt + b R + b L s + J R s + J L s^2))
   (-Kt L i[0] - Kt LaplaceTransform[ea[t], t, s] - R LaplaceTransform[τa[t], t, s] -
    L s LaplaceTransform[τa[t], t, s] - Ke Kt θ[0] - b R θ[0] - b L s θ[0] -
    J R s θ[0] - J L s^2 θ[0] - J R θ′[0] - J L s θ′[0]), LaplaceTransform[τ[t], t, s] →
  -((-b Kt L i[0] - J Kt L s i[0] - b Kt LaplaceTransform[ea[t], t, s] -
       J Kt s LaplaceTransform[ea[t], t, s] + Ke Kt LaplaceTransform[τa[t], t, s] +
       J Ke Kt θ′[0]) / (Ke Kt + b R + b L s + J R s + J L s^2))}
```

```
{LaplaceTransform[e[t], t, s] →
  Ke (Kt LaplaceTransform[ea[t], t, s] + (R + L s) LaplaceTransform[τa[t], t, s])
  ───────────────────────────────────────────────────────────────
                    Ke Kt + (b + J s) (R + L s)                               ,
 LaplaceTransform[i[t], t, s] →
  (b + J s) LaplaceTransform[ea[t], t, s] - Ke LaplaceTransform[τa[t], t, s]
  ───────────────────────────────────────────────────────────────
                    Ke Kt + (b + J s) (R + L s)                               ,
 LaplaceTransform[θ[t], t, s] →
  Kt LaplaceTransform[ea[t], t, s] + (R + L s) LaplaceTransform[τa[t], t, s]
  ───────────────────────────────────────────────────────────────
                  s (Ke Kt + (b + J s) (R + L s))                             ,
 LaplaceTransform[τ[t], t, s] →
  Kt ((b + J s) LaplaceTransform[ea[t], t, s] - Ke LaplaceTransform[τa[t], t, s])
  ───────────────────────────────────────────────────────────────
                    Ke Kt + (b + J s) (R + L s)                               }
```

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

```
solvedXforms = #[[1]] → Apart[#[[2]]] & /@ solvedXforms
```

$$\left\{\text{LaplaceTransform}[e[t], t, s] \to \frac{Ke\,Kt\,\text{LaplaceTransform}[ea[t], t, s]}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} + \frac{Ke\,(R + L\,s)\,\text{LaplaceTransform}[\tau a[t], t, s]}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2},\right.$$
$$\text{LaplaceTransform}[i[t], t, s] \to \frac{(b + J\,s)\,\text{LaplaceTransform}[ea[t], t, s]}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} - \frac{Ke\,\text{LaplaceTransform}[\tau a[t], t, s]}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2},\ \text{LaplaceTransform}[\theta[t], t, s] \to$$
$$\frac{Kt\,\text{LaplaceTransform}[ea[t], t, s]}{s\,(Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2)} + \frac{(R + L\,s)\,\text{LaplaceTransform}[\tau a[t], t, s]}{s\,(Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2)},$$
$$\left.\text{LaplaceTransform}[\tau[t], t, s] \to \frac{Kt\,(b + J\,s)\,\text{LaplaceTransform}[ea[t], t, s]}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} - \frac{Ke\,Kt\,\text{LaplaceTransform}[\tau a[t], t, s]}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2}\right\}$$

Transfer Function Models

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

```
Clear[makeModel]
makeModel[var_] := makeModel[var, ToString[var], 1]
makeModel[var_, label_, factor_] := TransferFunctionModel[
  factor * Function[{sum}, {sum[[1]] / voltageXform, sum[[2]] / τinXform}]
    [LaplaceTransform[var[t], t, s] /. solvedXforms],
  s,
  SystemsModelLabels → {{"V", "τin"}, label}]
motorPositionModel = makeModel[θ]
motorVelocityModel = makeModel[θ, "Ω", s]
motorAccelerationModel = makeModel[θ, "α", s * s]
motorCurrentModel = makeModel[i]
motorEMFModel = makeModel[e]
motorTorqueModel = makeModel[τ]
```

$$\left(\begin{array}{c|cc} & V & \tau in \\ \hline \theta & \frac{Kt}{s\,(Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2)} & \frac{R + L\,s}{s\,(Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2)} \end{array}\right)^{\mathcal{T}}$$

$$\left(\begin{array}{c|cc} & V & \tau in \\ \hline \Omega & \frac{Kt}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} & \frac{R + L\,s}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} \end{array}\right)^{\mathcal{T}}$$

$$\left(\begin{array}{c|cc} & V & \tau in \\ \hline \alpha & \frac{Kt\,s}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} & \frac{s\,(R + L\,s)}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} \end{array}\right)^{\mathcal{T}}$$

$$\left(\begin{array}{c|cc} & V & \tau in \\ \hline i & \frac{b + J\,s}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} & -\frac{Ke}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} \end{array}\right)^{\mathcal{T}}$$

$$\left(\begin{array}{c|cc} & V & \tau in \\ \hline e & \frac{Ke\,Kt}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} & \frac{Ke\,(R + L\,s)}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} \end{array}\right)^{\mathcal{T}}$$

$$\left(\begin{array}{c|cc} & V & \tau in \\ \hline \tau & \frac{Kt\,(b + J\,s)}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} & -\frac{Ke\,Kt}{Ke\,Kt + b\,R + b\,L\,s + J\,R\,s + J\,L\,s^2} \end{array}\right)^{\mathcal{T}}$$

Motor Parameters Redux

On Ke & Kt

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
```

```
e[t] == Ke θ′[t]
ea[t] == e[t] + R i[t] + L i′[t]
τ[t] == Kt i[t]
τ[t] + τa[t] - b θ′[t] == J θ′′[t]
```

In steady state, the current and speed are constant.

```
(ssEqns = diffEqns /. {i'[t] → 0, θ''[t] → 0}) // ColumnForm
(ssEqns = Eliminate[ssEqns, {e[t], b }]) // ColumnForm
```

```
e[t] == Ke θ′[t]
ea[t] == e[t] + R i[t]
τ[t] == Kt i[t]
τ[t] + τa[t] - b θ′[t] == 0
```

```
ea[t] == R i[t] + Ke θ′[t]
τ[t] == Kt i[t]
```

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 == ea[t] i[t],
    pwrOut == τ[t] θ'[t],
    pwrLosses == i[t]^2 R + pwrOtherPowerLosses,
    pwrIn == pwrOut + pwrLosses
   };
(ssPowerEqns = ssEqns ~Join~ powerEqns) // prettyPrint
```

$\mathrm{ea}(t) = R\, i(t) + \mathrm{Ke}\, \theta'(t)$
$\tau(t) = \mathrm{Kt}\, i(t)$
$\mathrm{pwrIn} = \mathrm{ea}(t)\, i(t)$
$\mathrm{pwrOut} = \tau(t)\, \theta'(t)$
$\mathrm{pwrLosses} = R\, i(t)^2 + \mathrm{pwrOtherPowerLosses}$
$\mathrm{pwrIn} = \mathrm{pwrLosses} + \mathrm{pwrOut}$

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

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

pwrIn = pwrLosses + pwrOut
Kt pwrOtherPowerLosses = (Ke – Kt) pwrOut

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

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

Experimental Motor Data

We have experimental data from several motors (Nguyen, Hordyk, & Fraser, 2017). We load in same.

```
(motorData = Import[
    NotebookDirectory[] <> "Characterized Motors.xlsx", {"Data", 1}]) // TableForm
```

Name	J	b	K	R	L (uH)	L (H)	J
AM 20 A	9.0110-6	0.0022	0.351	2.3	691.	0.000691	9.011×10^{-6}
AM 20 B	9.0110-6	0.0025	0.389	1.9	684.	0.000684	9.011×10^{-6}
AM 20 C	8.9310-6	0.0028	0.385	5.1	717.	0.000717	8.931×10^{-6}
AM 40 A	2.2210-5	0.2269	0.753	2.5	674.	0.000674	0.00002221
AM 40 B	1.7410-5	0.56	0.705	3.8	705.	0.000705	0.00001741
AM 40 C	2.4710-5	0.018	0.763	2.1	716.	0.000716	0.00002471
AM 60 A	1.0410-5	0.033	1.066	3.3	694.	0.000694	0.00001041
AM 60 B	8.4210-6	0.02	1.076	5.1	696.	0.000696	8.421×10^{-6}
AM 3.7 A	2.7910-5	0.00014	0.099	8.9	679.	0.000679	0.00002791
AM 3.7 B	3.1510-5	0.000176	0.108	2.6	797.	0.000797	0.00003151
AM 3.7 C	3.0910-5	0.00017	0.105	8.7	880.	0.00088	0.00003091
Matrix A	9.4310-6	0.00151	0.34	3.8	718.	0.000718	9.431×10^{-6}
Matrix B	7.7610-6	0.00191	0.363	7.8	777.	0.000777	7.761×10^{-6}
Matrix C	7.2310-6	0.00186	0.338	20.6	658.	0.000658	7.231×10^{-6}
CoreHex A	7.3310-4	0.0112	0.822	3.6	1356.	0.001356	0.0007331
CoreHex B	6.5510-4	0.008	0.858	11.3	1352.	0.001352	0.0006551
CoreHex C	4.5410-4	0.0078	0.711	5.6	1342.	0.001342	0.0004541

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 allow for optional load inertia and viscous friction.

```
siAngularInertialUnits = parseUnit["kg m^2"] / parseUnit["Radians^2"];
siAngularViscousFrictionUnits = parseUnit["N m s"] / parseUnit["Radians^2"];
siTorqueUnits = parseUnit["N m"] / parseUnit["Radians"];

Clear[motorParameters]
Options[motorParameters] = {
   JLoad → Quantity[0, siAngularInertialUnits],
   bLoad → Quantity[0, siAngularViscousFrictionUnits]
  };
motorParameters[motorName_, opts : OptionsPattern[]] :=
  Module[{row, paramValues, quantify, assoc},
   row = Select[motorData, #[[1]] == motorName &][[1]];
   paramValues = #[[1]] → toRational[ row[[#[[2]]]] ] &/@
     {{J, 8}, {b, 3}, {Ke, 4}, {Kt, 4}, {R, 5}, {L, 7}};
   quantify = Function[{name, value},
     name → ((name /. parameterQuantities) /. name → value)
    ];
   assoc = quantify @@ # &/@ paramValues // Association;
   assoc[J] = assoc[J] + OptionValue[JLoad];
   assoc[b] = assoc[b] + OptionValue[bLoad];
   assoc
  ];

motorParameters["AM 60 A", JLoad → Quantity[1, siAngularInertialUnits]]
```

$\langle| J \to \frac{100\,001\,041}{100\,000\,000}\,\mathrm{kg\,m^2/rad^2},\ b \to \frac{33}{1000}\,\mathrm{m\,s\,N/rad^2},$
$Ke \to \frac{533}{500}\,\mathrm{s\,V/rad},\ Kt \to \frac{533}{500}\,\mathrm{m\,N/(A\,rad)},\ R \to \frac{33}{10}\,\Omega,\ L \to \frac{347}{500\,000}\,\mathrm{H} |\rangle$

Time Domain Functions

Development

We define a motor with a load that we'll use to illustrate examples. We also define a generic, abstract motor that can helps explore things symbolically

```
siAngularInertialUnits
aMotor = motorParameters["AM 60 A", JLoad → Quantity[1, siAngularInertialUnits]]
```

$$\frac{\text{Kilograms Meters}^2}{\text{Radians}^2}$$

$$\left\langle \left| J \to \frac{100001041}{100000000}\,\text{kg m}^2/\text{rad}^2,\ b \to \frac{33}{1000}\,\text{m s N}/\text{rad}^2,\ Ke \to \frac{533}{500}\,\text{s V}/\text{rad},\ Kt \to \frac{533}{500}\,\text{m N}/(\text{A rad}),\ R \to \frac{33}{10}\,\Omega,\ L \to \frac{347}{500000}\,\text{H} \right| \right\rangle$$

```
genericMotor = # → (# /. parameterQuantities) & /@ motorParameterNames
```

$$\{J \to J\,\text{kg m}^2/\text{rad}^2,\ b \to b\,\text{m s N}/\text{rad}^2,\ Ke \to Ke\,\text{s V}/\text{rad},\ Kt \to Kt\,\text{m N}/(\text{A rad}),\ R \to R\,\Omega,\ L \to L\,\text{H}\}$$

makeTimeDomainFunction[] takes a model and time-domain input functions and returns a time-domain output function.

```
Clear[makeTimeDomainFunction]
makeTimeDomainFunction[model_, tInputFunctions_] := Module[
  {s = Unique["s"], t = Unique["t"], sInputs, sOutput, tOutput},
  sInputs = LaplaceTransform[#[t], t, s] & /@ tInputFunctions;
  sOutput = model[s] . sInputs;
  tOutput = InverseLaplaceTransform[sOutput, s, t];
  tOutput /. t → # &
 ]
makeTimeDomainFunction[motorVelocityModel, {ea[#] &, τa[#] &}][t]
```

$$\left\{\text{InverseLaplaceTransform}\left[\frac{Kt\ \text{LaplaceTransform}[ea[t], t, s11]}{Ke\,Kt + b\,R + b\,L\,s11 + J\,R\,s11 + J\,L\,s11^2} + \frac{(R + L\,s11)\ \text{LaplaceTransform}[\tau a[t], t, s11]}{Ke\,Kt + b\,R + b\,L\,s11 + J\,R\,s11 + J\,L\,s11^2}, s11, t\right]\right\}$$

makeMotorTimeDomainFunction[] returns a function that, when invoked with ea and τa functions, returns a function of time.

```
Clear[makeMotorTimeDomainFunction]
makeMotorTimeDomainFunction[model_] :=
 Module[{ea = Unique["ea"], τa = Unique["τa"], t = Unique["t"], expr},
  expr = makeTimeDomainFunction[model, {ea[#] &, τa[#] &}][t];
  Function[{eaActual, τaActual}, Module[{exprT},
    exprT = expr /. {ea → eaActual, τa → τaActual};
    Function[{tActual}, exprT /. t → tActual]]
  ]]
makeMotorTimeDomainFunction[motorVelocityModel][12 &, 0 &][t]
```

$$\left\{12\,\mathrm{Kt}\left(\frac{1}{\mathrm{Ke}\,\mathrm{Kt}+b\,R}-\left(-b\,e^{\left(-\frac{b}{2J}-\frac{R}{2L}-\frac{\sqrt{-4\,J\,\mathrm{Ke}\,\mathrm{Kt}\,L+b^2L^2-2\,b\,J\,L\,R+J^2R^2}}{2\,J\,L}\right)t}L+b\,e^{\left(-\frac{b}{2J}-\frac{R}{2L}+\frac{\sqrt{-4\,J\,\mathrm{Ke}\,\mathrm{Kt}\,L+b^2L^2-2\,b\,J\,L\,R+J^2R^2}}{2\,J\,L}\right)t}L-e^{\left(-\frac{b}{2J}-\frac{R}{2L}-\frac{\sqrt{-4\,J\,\mathrm{Ke}\,\mathrm{Kt}\,L+b^2L^2-2\,b\,J\,L\,R+J^2R^2}}{2\,J\,L}\right)t}J\,R+e^{\left(-\frac{b}{2J}-\frac{R}{2L}+\frac{\sqrt{-4\,J\,\mathrm{Ke}\,\mathrm{Kt}\,L+b^2L^2-2\,b\,J\,L\,R+J^2R^2}}{2\,J\,L}\right)t}J\,R+e^{\left(-\frac{b}{2J}-\frac{R}{2L}-\frac{\sqrt{-4\,J\,\mathrm{Ke}\,\mathrm{Kt}\,L+b^2L^2-2\,b\,J\,L\,R+J^2R^2}}{2\,J\,L}\right)t}\sqrt{-4\,J\,\mathrm{Ke}\,\mathrm{Kt}\,L+b^2L^2-2\,b\,J\,L\,R+J^2R^2}+e^{\left(-\frac{b}{2J}-\frac{R}{2L}+\frac{\sqrt{-4\,J\,\mathrm{Ke}\,\mathrm{Kt}\,L+b^2L^2-2\,b\,J\,L\,R+J^2R^2}}{2\,J\,L}\right)t}\sqrt{-4\,J\,\mathrm{Ke}\,\mathrm{Kt}\,L+b^2L^2-2\,b\,J\,L\,R+J^2R^2}\right)\Big/\left(2\left(\mathrm{Ke}\,\mathrm{Kt}+b\,R\right)\sqrt{-4\,J\,\mathrm{Ke}\,\mathrm{Kt}\,L+b^2L^2-2\,b\,J\,L\,R+J^2R^2}\right)\right)\right\}$$

We define such functions for all the 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];
```

Inputs

In analogy to parameterQuantities above, we define inputQuantities, which adds units to our input parameters.

```
inputQuantities = {ea → Quantity[ea, "Volts"],
  τa → Quantity[τa, siTorqueUnits], t → Quantity[t, "Seconds"]}
```

{ea → ea V , τa → τa mN/rad , t → t s }

Similarly, in analogy to aMotor above, we define some typical inputs. The first has no externally applied torque; the second has a torque corresponding to a (non-accelerating) mass suspended on a massless string from a massless pulley attached to the motor shaft. To develop the latter, we first consider the pulley disconnected from the motor and as stationary. Then, there is a force pulling down due to gravity; this is also the tension in the string. That string acts at a wrench with a length that of the radius of the pulley to deliver a torque to the shaft. We convert everything to SI units to check unit consistency.

```
{ (τMassOnPulley =
    Quantity[weightLbs, "Pounds"]
     * Quantity["StandardAccelerationOfGravity"]
     * Quantity[rInch, "Inches / Radians"])
  == Quantity[τ, siTorqueUnits]}
% // siUnits
```

```
{ rInch weightLbs lb in g/rad == τ m N/rad }
```

$$\left\{ \frac{1\,129\,848\,290\,276\,167\ \text{rInch weightLbs}}{10\,000\,000\,000\,000\,000}\ \text{kg}\,\text{m}^2/(\text{s}^2\text{rad}) == \tau\ \text{kg}\,\text{m}^2/(\text{s}^2\text{rad}) \right\}$$

We defined the aforementioned typical inputs.

```
anInput = {ea → Quantity[12, "Volts"],
  τa → Quantity[0, siTorqueUnits], t → Quantity[t, "Seconds"]}
anInputτ = {ea → Quantity[12, "Volts"],
  τa → (τMassOnPulley /. {weightLbs → 3, rInch → 2}), t → Quantity[t, "Seconds"]}
aGenericInputτ = {ea → Quantity[12, "Volts"],
  τa → (τMassOnPulley), t → Quantity[t, "Seconds"]}
```

```
{ea → 12 V, τa → 0 m N/rad, t → t s}
```

```
{ea → 12 V, τa → 6 lb in g/rad, t → t s}
```

```
{ea → 12 V, τa → rInch weightLbs lb in g/rad, t → t s}
```

Examples

We try out our time-domain functions on our example data. Notice how the units come out correctly: they flow automatically from beginning to end

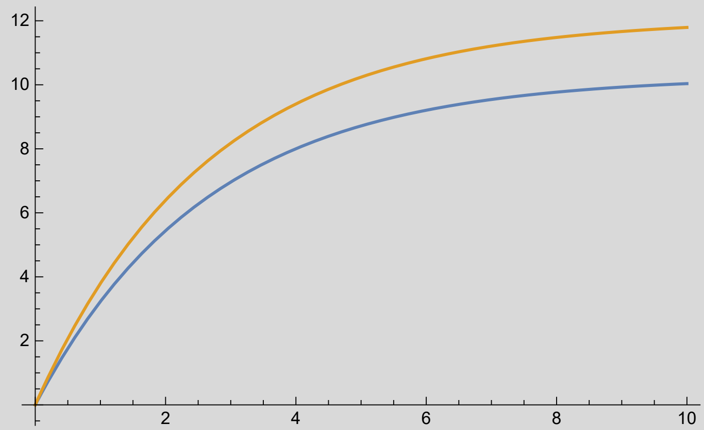
```
velUnits =
 motorVelocity[ea &, τa &][t] /. aMotor /. anInput // N // siUnits // FullSimplify
```

$$\left\{ e^{-0.377374\,t}\,(-10.2734\ \text{rad/s}) + e^{-4754.7\,t}\,(0.000815385\ \text{rad/s}) + 10.2726\ \text{rad/s} \right\}$$

```
vel = velUnits // clearUnits
velτ =
 motorVelocity[ea &, τa &][t] /. aMotor /. anInputτ // N // siUnits // clearUnits //
  FullSimplify
Plot[{vel, velτ}, {t, 0, 10}]
```

$\{10.2726 + 0.000815385\, e^{-4754.7\, t} - 10.2734\, e^{-0.377374\, t}\}$

$\{12.0691 + 0.000815396\, e^{-4754.7\, t} - 12.0699\, e^{-0.377374\, t}\}$



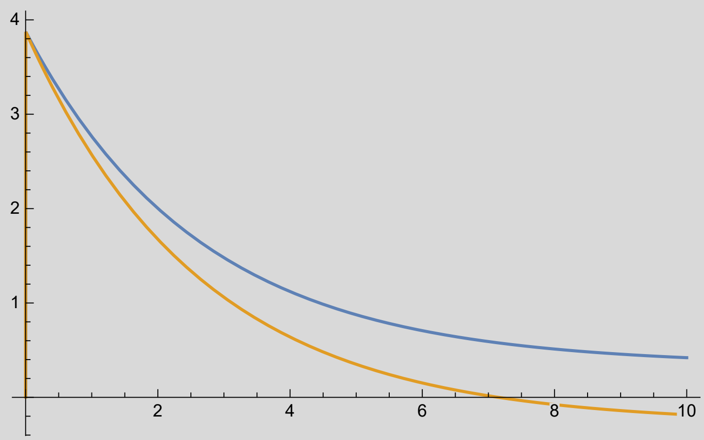
Good: the model with the (positive) external torque achieves greater velocity, as it should.

```
torUnits =
 motorTorque[ea &, τa &][t]  /. aMotor /. anInput // N // siUnits // FullSimplify
torτ = motorTorque[ea &, τa &][t]  /. aMotor /. anInputτ // N // siUnits //
   clearUnits // FullSimplify
tor = torUnits // clearUnits
Plot[{tor, torτ}, {t, 0, 10}]
```

$$\left\{e^{-4754.7\,t}\left(-3.87693\,\mathrm{kg\,m^2/(s^2\,rad)}\right)+0.338995\,\mathrm{kg\,m^2/(s^2\,rad)}+e^{-0.377374\,t}\left(3.53793\,\mathrm{kg\,m^2/(s^2\,rad)}\right)\right\}$$

$$\left\{-0.279629-3.87697\,e^{-4754.7\,t}+4.1566\,e^{-0.377374\,t}\right\}$$

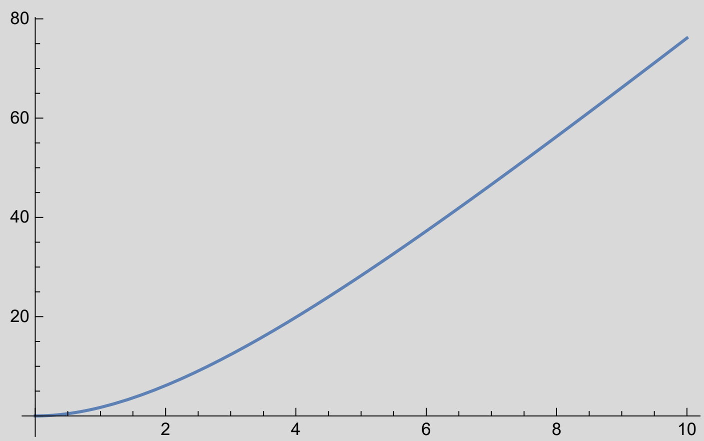
$$\left\{0.338995-3.87693\,e^{-4754.7\,t}+3.53793\,e^{-0.377374\,t}\right\}$$



```
posUnits =
 motorPosition[ea &, τa &][t]  /. aMotor /. anInput // N // siUnits // FullSimplify
pos = posUnits // clearUnits
Plot[{pos}, {t, 0, 10}]
```

$\left\{e^{-4754.7\,t}\left(-1.7149\times10^{-7}\,\text{rad}\right)+e^{-0.377374\,t}\left(27.2234\,\text{rad}\right)+\left(-27.2234+10.2726\,t\right)\,\text{rad}\right\}$

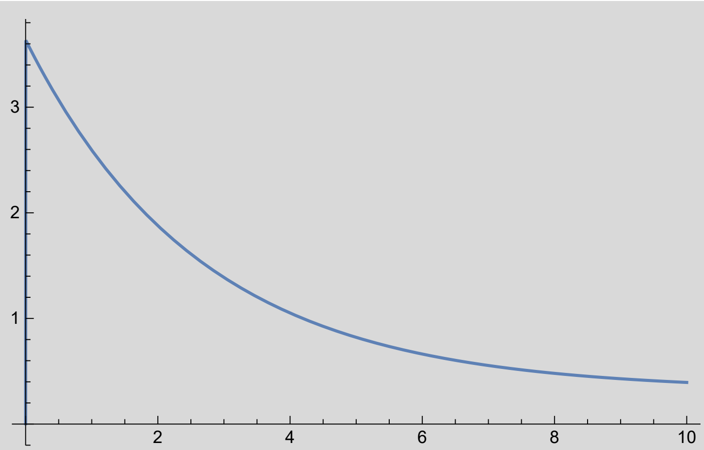
$\left\{-27.2234-1.7149\times10^{-7}\,e^{-4754.7\,t}+27.2234\,e^{-0.377374\,t}+10.2726\,t\right\}$




```
curUnits =
 motorCurrent[ea &, τa &][t]  /. aMotor /. anInput // N // siUnits // FullSimplify
cur = curUnits // clearUnits
Plot[{cur}, {t, 0, 10}]
```

$\left\{e^{-4754.7\,t}\left(-3.63689\,\text{A}\right)+0.318007\,\text{A}+e^{-0.377374\,t}\left(3.31888\,\text{A}\right)\right\}$

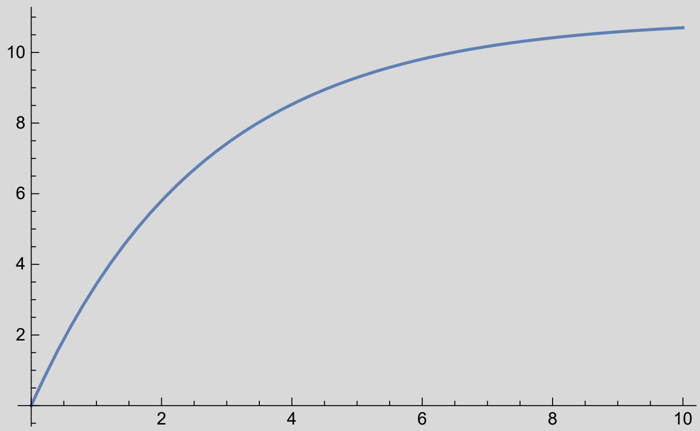
$\left\{0.318007-3.63689\,e^{-4754.7\,t}+3.31888\,e^{-0.377374\,t}\right\}$



```
emfUnits =
 motorEMF[ea &, τa &][t] /. aMotor /. anInput // N // siUnits // FullSimplify
emf = emfUnits // clearUnits
Plot[{emf}, {t, 0, 10}]
```

$$\left\{e^{-0.377374\,t}\left(-10.9514\,\mathrm{kg\,m^2/(s^3A)}\right) + e^{-4754.7\,t}\left(0.000869201\,\mathrm{kg\,m^2/(s^3A)}\right) + 10.9506\,\mathrm{kg\,m^2/(s^3A)}\right\}$$

$$\left\{10.9506 + 0.000869201\,e^{-4754.7\,t} - 10.9514\,e^{-0.377374\,t}\right\}$$



Steady State

Calculating values

We explore the steady state behavior. We can compute same either using the final value theorem, or directly by taking the limit (not shown here; previously verified). The former is definitely more efficient.

```
Clear[ssValue]
ssValue[model_, inputs_] := Module[{s = Unique[], t = Unique[], expr},
  expr = model[s] . (LaplaceTransform[#[t], t, s] & /@ inputs);
  Limit[s expr, s → 0][[1]]
 ]
```

```
(ss = {
    ssPos → ssValue[motorPositionModel, {ea &, τa &}],
    ssVel → ssValue[motorVelocityModel, {ea &, τa &}],
    ssAcc → ssValue[motorAccelerationModel, {ea &, τa &}],
    ssEmf → ssValue[motorEMFModel, {ea &, τa &}],
    ssCur → ssValue[motorCurrentModel, {ea &, τa &}],
    ssTor → ssValue[motorTorqueModel, {ea &, τa &}]
   }) // prettyPrint
```

ssPos → Indeterminate

ssVel → $\frac{\text{ea Kt}+R\,\tau\text{a}}{\text{Ke Kt}+b\,R}$

ssAcc → 0

ssEmf → $\frac{\text{ea Ke Kt}+\text{Ke}\,R\,\tau\text{a}}{\text{Ke Kt}+b\,R}$

ssCur → $\frac{b\,\text{ea}-\text{Ke}\,\tau\text{a}}{\text{Ke Kt}+b\,R}$

ssTor → $\frac{b\,\text{ea Kt}-\text{Ke Kt}\,\tau\text{a}}{\text{Ke Kt}+b\,R}$

Back EMF vs Applied Voltage

We want to know the steady-state velocity at which the back EMF balances the input voltage. Thus, we need EMF in terms of speed. We have EMF from voltage, and speed from voltage. So we need to invert the latter, then compose.

```
Clear[ssAppliedVoltageFromVelocity]
ssAppliedVoltageFromVelocity[velocity_] := Module[{eqn, velSym = Unique["vel"]},
  eqn = velSym == (ssVel /. ss);
  ea /. uniqueSolve[eqn, ea][[1]] /. velSym → velocity
 ]
ssAppliedVoltageFromVelocity[Ω]
```

$$\frac{-R\,\tau\text{a}+\text{Ke Kt}\,\Omega+\text{b}\,R\,\Omega}{\text{Kt}}$$

```
Clear[ssEmfFromAppliedVoltage, ssEmfFromVelocity]
ssEmfFromAppliedVoltage[voltage_] := (ssEmf /. ss) /. ea → voltage
ssEmfFromVelocity[velocity_] :=
 ssEmfFromAppliedVoltage[ssAppliedVoltageFromVelocity[velocity]]
ssEmfFromVelocity[Ω]
% // FullSimplify
```

$$\frac{\text{Ke}\,R\,\tau\text{a}+\text{Ke}\,(-R\,\tau\text{a}+\text{Ke Kt}\,\Omega+\text{b}\,R\,\Omega)}{\text{Ke Kt}+\text{b}\,R}$$

$$\text{Ke}\,\Omega$$

Well, that's a result now, isn't it? So, at what velocity do the two voltages balance?

```
ssEmfFromVelocity[Ω] == ssAppliedVoltageFromVelocity[Ω] // FullSimplify
uniqueSolve[%, Ω]
emfThresholdVelocity = %[[1]]
```

$$\frac{R\,(\tau a - b\,\Omega)}{Kt} == 0$$

$$\left\{\Omega \to \frac{\tau a}{b}\right\}$$

$$\Omega \to \frac{\tau a}{b}$$

That's interesting: the equalizing velocity is dependent on the motor (through its viscous friction parameter), the external system (through the externally applied torque) but *not* the externally applied voltage. Also: if there is no external torque, then the threshold velocity is zero, which is as it should be, since in that situation the back EMF can never match the externally applied voltage due to losses.

A quick check that the units are correct:

```
emfThresholdVelocity /. {Ω → Ω[t], τa → τa[t]} /. parameterQuantities
```

$$\Omega[t]\ \text{rad/s} \to \frac{\tau a[t]}{b}\ \text{rad/s}$$

What does this look like for an actual motor with our example weight attached?

```
Ω /. emfThresholdVelocity /. aMotor /. anInputτ // siUnits // N
UnitConvert[%, "rpm"]
UnitConvert[%, "Revolutions / Second"] *
 Quantity[1120, IndependentUnit["Ticks"] / "Revolutions"]
```

20.5427 rad/s

196.168 rev/min

3661.81 Ticks /s