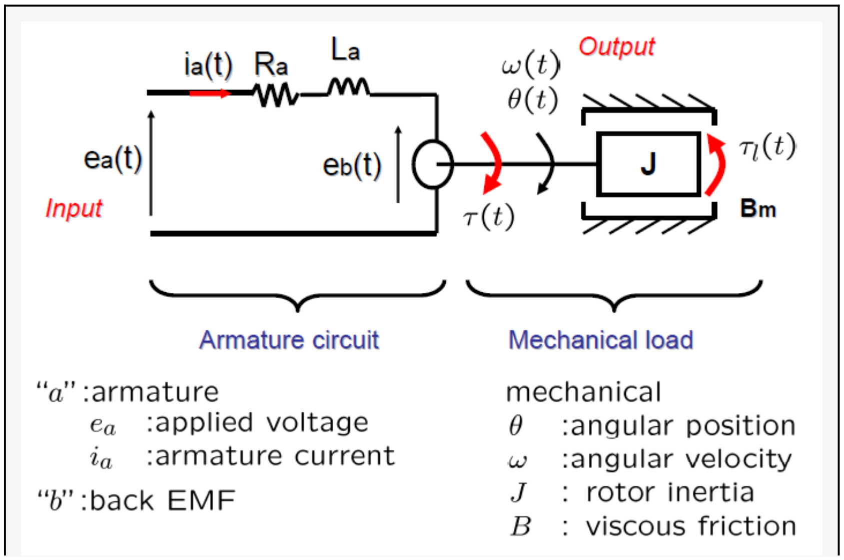
Motor Physics

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.



The angular velocity is the time derivative of the position, so the velocity is redundant. With that considered, we can see that the state of the system can be described by the following variables.

- *V,* the applied voltage
- *i,* the armature current
- *e,* the back EMF
- *θ*, the angular position

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 = {}
```

```
{}
```

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] - b θ'[t] == J θ''[t]}]
```

```
{e[t] == Ke θ′[t], τ[t] == Kt i[t], τ[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, {V[t] == R i[t] + L i'[t] + e[t]}]
```

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

```
diffEqns // prettyPrint
```

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

Initial Conditions

Some of the initial conditions we can intuit intellectually:

```
initialConditions = {
   θ[0] == 0,
   i[0] == 0,
   e[0] == 0,
   V[0]  == ν
  };
```

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, τ[0] - b θ′[0] == J θ′′[0]}
```

$$i'(0) = \frac{v}{L}$$
$$\theta'(0) = 0$$
$$\theta''(0) = 0$$
$$\tau(0) = 0$$
$$\theta(0) = 0$$
$$i(0) = 0$$
$$e(0) = 0$$
$$V(0) = v$$

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"],
   θ''[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"],
   V[t] → parseUnit["Volts"],
   e[t] → parseUnit["Volts"],
   τ[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 → Ohms, L → Henries, θ[t] → Radians, θ'[t] → $\frac{\text{Radians}}{\text{Seconds}}$, θ''[t] → $\frac{\text{Radians}}{\text{Seconds}^2}$, i[t] → Amperes, i′[t] → $\frac{\text{Amperes}}{\text{Seconds}}$, i″[t] → $\frac{\text{Amperes}}{\text{Seconds}^2}$, V[t] → Volts, e[t] → Volts, τ[t] → $\frac{\text{Meters Newtons}}{\text{Radians}}$ $|\rangle$

$\langle|$ R → R Ω, L → L H, θ[t] → θ[t] rad, θ'[t] → θ'[t] rad/s, θ''[t] → θ''[t] rad/s^2, i[t] → i[t] A, i′[t] → i′[t] A/s, i″[t] → i″[t] A/s^2, V[t] → V[t] V, e[t] → e[t] V, τ[t] → τ[t] mN/rad $|\rangle$

Let’s look at how far that takes us:

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

```
e[t] V == Ke (θ′[t] rad/s)
V[t] V == (e[t] + R i[t] + L i′[t]) AΩ
τ[t] mN/rad == Kt (i[t] A)
τ[t] mN/rad + b (−θ′[t] rad/s) == J (θ″[t] rad/s²)
```

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}]
```

```
⟨|R → R Ω, L → L H, θ[t] → θ[t] rad, θ′[t] → θ′[t] rad/s,
 θ″[t] → θ″[t] rad/s^2, i[t] → i[t] A, i′[t] → i′[t] A/s, i″[t] → i″[t] A/s^2,
 V[t] → V[t] V, e[t] → e[t] V, τ[t] → τ[t] mN/rad, J → J kg m^2/rad^2 |⟩
```

```
e[t] V == Ke (θ′[t] rad/s)
V[t] V == (e[t] + R i[t] + L i′[t]) A Ω
τ[t] mN/rad == Kt (i[t] A)
τ[t] mN/rad + b (-θ′[t] rad/s) == J θ″[t] kg m^2/(s^2 rad)
```

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

Let’s check how those work.

```
remainingUnits /. parameterQuantities
```

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

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,
 θ″[t] → θ″[t] rad/s^2, i[t] → i[t] A, i′[t] → i′[t] A/s,
 i″[t] → i″[t] A/s^2, V[t] → V[t] V, e[t] → e[t] V, τ[t] → τ[t] mN/rad,
 J → J kg m^2/rad^2, Ke → Ke s V/rad, Kt → Kt m N/(A rad), b → b m s N/rad^2 |⟩
```

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[V[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] - b (s LaplaceTransform[θ[t], t, s] - θ[0]) ==
  J (s^2 LaplaceTransform[θ[t], t, s] - s θ[0] - θ′[0])}
```

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

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[i[t], t, s], LaplaceTransform[V[t], t, s],
 LaplaceTransform[θ[t], t, s], LaplaceTransform[τ[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[V[t], t, s];
outputXorms = Complement[allXforms, {voltageXform}]
solvedXforms = uniqueSolve[leqns, outputXorms]
solvedXforms =
 solvedXforms /. (allInitialConditions /. Equal → Rule) // FullSimplify
```

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

$$\left\{\text{LaplaceTransform}[e[t],t,s]\to\frac{\text{Ke Kt LaplaceTransform}[V[t],t,s]}{\text{Ke Kt}+(b+J\,s)\,(R+L\,s)},\right.$$

$$\text{LaplaceTransform}[i[t],t,s]\to\frac{(b+J\,s)\ \text{LaplaceTransform}[V[t],t,s]}{\text{Ke Kt}+(b+J\,s)\,(R+L\,s)},$$

$$\text{LaplaceTransform}[\theta[t],t,s]\to\frac{\text{Kt LaplaceTransform}[V[t],t,s]}{s\,(\text{Ke Kt}+(b+J\,s)\,(R+L\,s))},$$

$$\left.\text{LaplaceTransform}[\tau[t],t,s]\to\frac{\text{Kt}\,(b+J\,s)\ \text{LaplaceTransform}[V[t],t,s]}{\text{Ke Kt}+(b+J\,s)\,(R+L\,s)}\right\}$$

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_, coefficient_] := TransferFunctionModel[
  (LaplaceTransform[var[t], t, s] * coefficient /. solvedXforms) / voltageXform,
  s,
  SystemsModelLabels → {"V", label}]
motorPositionModel = makeModel[θ]
motorVelocityModel = makeModel[θ, "Ω", s]
motorAccelerationModel = makeModel[θ, "α", s * s]
motorCurrentModel = makeModel[i]
motorEMFModel = makeModel[e]
motorTorqueModel = makeModel[τ]
```

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

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

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

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

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

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

Motor Parameters Redux

On Ke & Kt

It is a curious fact that, in compatible units, Ke & Kt have the same magnitude and sign (Further, if one ignores angular position (aka radians) as a unit, they can be considered to have the *same* units. But that’s a longer story). Recall our differential equations:

```
diffEqns // ColumnForm
```

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

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

From a power point of view, we must have (electrical) power in = (mechanical) power out + (electrical + mechanical) power losses:

```
powerEqns = And @@ {
    wIn == V[t] i[t],
    wOut == τ[t] θ'[t],
    wLosses == i[t]^2 R + wMechanicalPowerLosses,
    wIn == wOut + wLosses
   };
(ssPowerEqns = ssEqns ~Join~ powerEqns) // prettyPrint
```

$$V(t) = R\, i(t) + \text{Ke}\, \theta'(t)$$
$$\tau(t) = \text{Kt}\, i(t)$$
$$\text{wIn} = i(t)\, V(t)$$
$$\text{wOut} = \tau(t)\, \theta'(t)$$
$$\text{wLosses} = R\, i(t)^2 + \text{wMechanicalPowerLosses}$$
$$\text{wIn} = \text{wLosses} + \text{wOut}$$

Let’s eliminate a few of the variables

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

wIn = wLosses + wOut

Kt wMechanicalPowerLosses = (Ke – Kt) wOut

$$\left\{\text{wMechanicalPowerLosses} == \frac{(\text{Ke} - \text{Kt})\ \text{wOut}}{\text{Kt}}\right\}$$

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

Experimental Motor Data

We have experimental data from several motors. 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"];

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]]
```

$\langle| J \to \frac{100001041}{100000000}\,\mathrm{kg\,m^2/rad^2},\ b \to \frac{33}{1000}\,\mathrm{m\,s\,N/rad^2},$

$Ke \to \frac{533}{500}\,\mathrm{s\,V/rad},\ Kt \to \frac{533}{500}\,\mathrm{m\,N/(A\,rad)},\ R \to \frac{33}{10}\,\Omega,\ L \to \frac{347}{500000}\,\mathrm{H} |\rangle$

Time Domain Functions

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]]
```

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

$\langle| J \to \frac{100001041}{100000000}\,\mathrm{kg\,m^2/rad^2},\ b \to \frac{33}{1000}\,\mathrm{m\,s\,N/rad^2},$

$Ke \to \frac{533}{500}\,\mathrm{s\,V/rad},\ Kt \to \frac{533}{500}\,\mathrm{m\,N/(A\,rad)},\ R \to \frac{33}{10}\,\Omega,\ L \to \frac{347}{500000}\,\mathrm{H} |\rangle$

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

```
{J → J kg m^2/rad^2 , b → b m s N/rad^2 , Ke → Ke s V/rad , Kt → Kt m N/(A rad) , R → R Ω , L → L H }
```

We make time domain functions for each of the values of interest

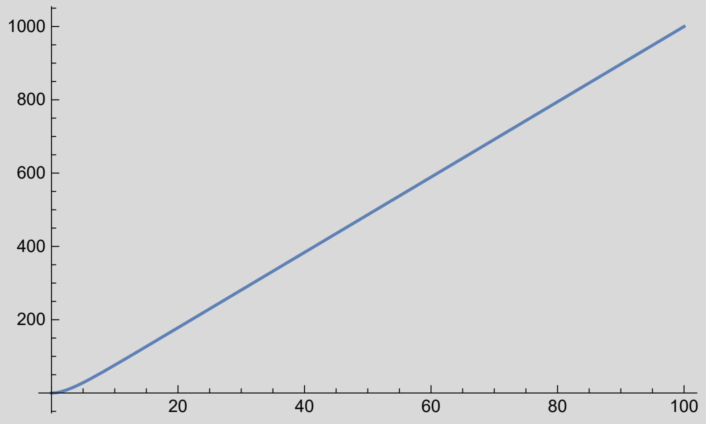
```
Clear[makeTimeDomainFunction, motorPosition,
 motorVelocity, motorCurrent, motorEMF, motorTorque]
makeTimeDomainFunction[parameters_, tranferFunctionModel_, stimulus_] := Module[
  {transferExpr, inputExpr, outputExpr, s = Unique[], t = Unique[], tExpr},
  transferExpr  = tranferFunctionModel[s][[1, 1]] /. parameters // siUnits ;
  inputExpr = LaplaceTransform[stimulus[t], t, s];
  outputExpr = transferExpr inputExpr;
  tExpr = InverseLaplaceTransform[outputExpr, s, t];
  tExpr /. t → # &]

motorPosition[parameters_, V_] :=  motorPosition[parameters, V] =
  makeTimeDomainFunction[parameters, motorPositionModel, V UnitStep[#] &]
motorVelocity[parameters_, V_] := motorVelocity[parameters, V] =
  makeTimeDomainFunction[parameters, motorVelocityModel, V UnitStep[#] &]
motorCurrent[parameters_, V_] := motorCurrent[parameters, V] =
  makeTimeDomainFunction[parameters, motorCurrentModel, V UnitStep[#] &]
motorEMF[parameters_, V_] :=  motorEMF[parameters, V] =
  makeTimeDomainFunction[parameters, motorEMFModel, V UnitStep[#] &]
motorTorque[parameters_, V_] := motorTorque[parameters, V] =
  makeTimeDomainFunction[parameters, motorTorqueModel, V UnitStep[#] &]
```

We explore same. Notice how the units come out correctly. Huzzah.

```
pos = motorPosition[aMotor, Quantity[12, "Volts"]][Quantity[t, "Seconds"]] // N //
  FullSimplify
pos // siUnits
Plot[pos, {t, 0, 100}]
```

$$e^{-4754.7\,t}\left(-1.7149\times 10^{-7}\,\mathrm{s^3\,A\,rad\,V/(kg\,m^2)}\right) + e^{-0.377374\,t}\left(27.2234\,\mathrm{s^3\,A\,rad\,V/(kg\,m^2)}\right) + (-27.2234 + 10.2726\,t)\,\mathrm{s^3\,A\,rad\,V/(kg\,m^2)}$$

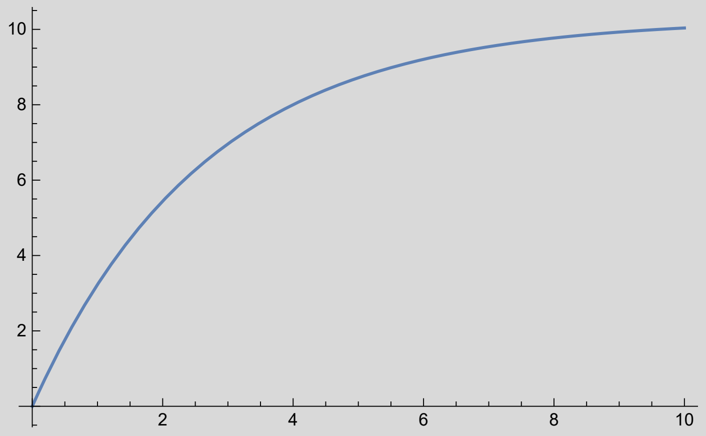
$$e^{-4754.7\,t}\left(-1.7149\times 10^{-7}\,\mathrm{rad}\right) + e^{-0.377374\,t}\left(27.2234\,\mathrm{rad}\right) + (-27.2234 + 10.2726\,t)\,\mathrm{rad}$$



```
vel = motorVelocity[aMotor, Quantity[12, "Volts"]][Quantity[t, "Seconds"]] // N //
  FullSimplify
vel // siUnits
Plot[vel, {t, 0, 10}]
```

$$e^{-0.377374\,t}\left(-10.2734\ \mathrm{s^2 A\,rad\,V/(kg\,m^2)}\right)+ e^{-4754.7\,t}\left(0.000815385\ \mathrm{s^2 A\,rad\,V/(kg\,m^2)}\right)+10.2726\ \mathrm{s^2 A\,rad\,V/(kg\,m^2)}$$

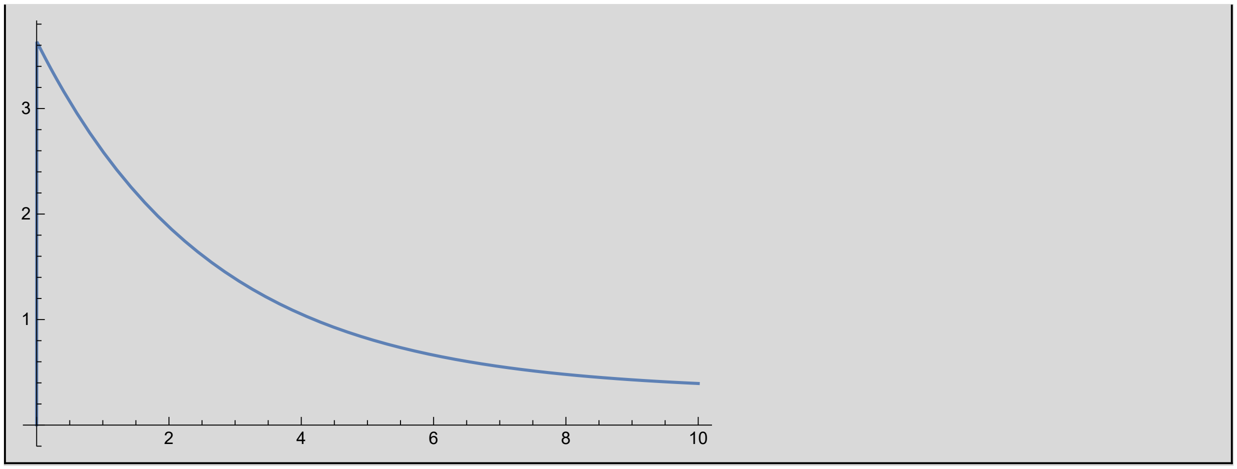
$$e^{-0.377374\,t}\left(-10.2734\ \mathrm{rad/s}\right)+e^{-4754.7\,t}\left(0.000815385\ \mathrm{rad/s}\right)+10.2726\ \mathrm{rad/s}$$



```
cur = motorCurrent[aMotor, Quantity[12, "Volts"]][Quantity[t, "Seconds"]] // N //
  FullSimplify
cur // siUnits // simplifyUnits
Plot[cur, {t, 0, 10}]
```

$$e^{-4754.7\,t}\left(-3.63689\ \mathrm{s^3A^2V/(kg\,m^2)}\right) + 0.318007\ \mathrm{s^3A^2V/(kg\,m^2)} + e^{-0.377374\,t}\left(3.31888\ \mathrm{s^3A^2V/(kg\,m^2)}\right)$$

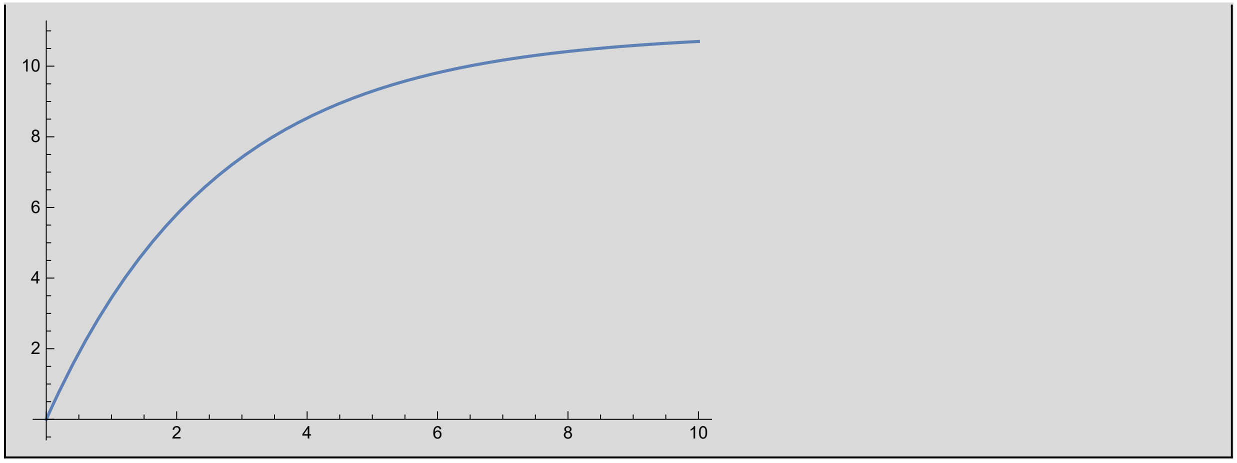
$$e^{-4754.7\,t}\left(-3.63689\ \mathrm{A}\right) + 0.318007\ \mathrm{A} + e^{-0.377374\,t}\left(3.31888\ \mathrm{A}\right)$$



```
emf = motorEMF[aMotor, Quantity[12, "Volts"]][Quantity[t, "Seconds"]] // N //
  FullSimplify
emf // siUnits // simplifyUnits
Plot[emf, {t, 0, 10}]
```

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

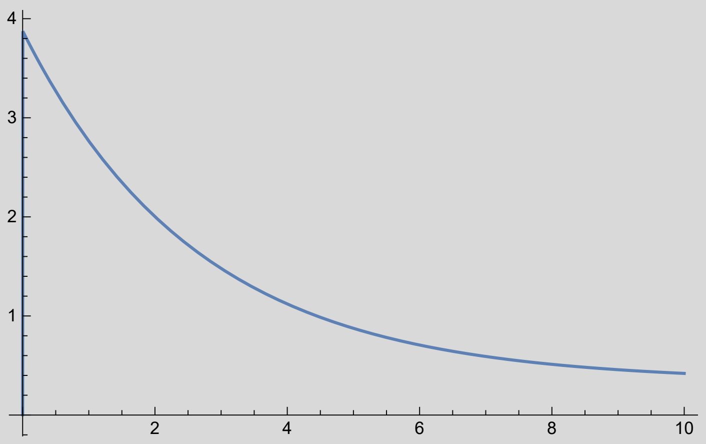
$$e^{-0.377374\,t}\left(-10.9514\ \mathrm{V}\right) + e^{-4754.7\,t}\left(0.000869201\ \mathrm{V}\right) + 10.9506\ \mathrm{V}$$



```
tor = motorTorque[aMotor, Quantity[12, "Volts"]][Quantity[t, "Seconds"]] // N //
  FullSimplify
tor // siUnits // simplifyUnits
Plot[tor, {t, 0, 10}]
```

$$e^{-4754.7\,t}\left(-3.87693\ \text{s A V/rad}\right) + 0.338995\ \text{s A V/rad} + e^{-0.377374\,t}\left(3.53793\ \text{s A V/rad}\right)$$

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



Steady State

We explore the steady state behavior. We can compute same either using the final value theorem, or directly by taking the limit. The former is definitely more efficient.

```
Clear[ssValue]
ssValue[model_, stimulus_] := Module[{s = Unique[], t = Unique[], expr},
  expr = model[s] * LaplaceTransform[stimulus[t], t, s];
  Limit[s expr, s → 0]
 ]
ssPos → ssValue[motorPositionModel, ν UnitStep[#] &]
ssVel → ssValue[motorVelocityModel, ν UnitStep[#] &]
ssAcc → ssValue[motorAccelerationModel, ν UnitStep[#] &]
ssEmf → ssValue[motorEMFModel, ν UnitStep[#] &]
ssCur → ssValue[motorCurrentModel, ν UnitStep[#] &]
ssTor → ssValue[motorTorqueModel, ν UnitStep[#] &]
```

```
ssPos → {{Indeterminate}}
```

$$\text{ssVel} \to \left\{\left\{\frac{\text{Kt}\,\nu}{\text{Ke}\,\text{Kt} + \text{b}\,\text{R}}\right\}\right\}$$

```
ssAcc → {{0}}
```

$$\text{ssEmf} \to \left\{\left\{\frac{\text{Ke}\,\text{Kt}\,\nu}{\text{Ke}\,\text{Kt} + \text{b}\,\text{R}}\right\}\right\}$$

$$\text{ssCur} \to \left\{\left\{\frac{\text{b}\,\nu}{\text{Ke}\,\text{Kt} + \text{b}\,\text{R}}\right\}\right\}$$

$$\text{ssTor} \to \left\{\left\{\frac{\text{b}\,\text{Kt}\,\nu}{\text{Ke}\,\text{Kt} + \text{b}\,\text{R}}\right\}\right\}$$

For comparison, we also exhibit velocity calculated the hard way.

```
genericVelExpr =
  motorVelocity[genericMotor, Quantity[v, "Volts"]][Quantity[t, "Seconds"]];
genericVelExpr = genericVelExpr // siUnits // clearUnits
parameterAssumptions  = # > 0 &/@ Select[ Keys[parameterQuantities], unboundQ[#] &]
ssVelHardWay = Limit[genericVelExpr, t → Infinity, Assumptions → parameterAssumptions]
```

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

```
{R > 0, L > 0, J > 0, Ke > 0, Kt > 0, b > 0}
```

$$\text{ConditionalExpression}\left[\frac{\text{Kt}\,v}{\text{Ke}\,\text{Kt}+b\,R},\right.$$
$$\left(\frac{1}{\sqrt{-4 J\,\text{Ke}\,\text{Kt}\,L+b^2 L^2-2 b J L R+J^2 R^2}}\left|\sqrt{-4 J\,\text{Ke}\,\text{Kt}\,L+b^2 L^2-2 b J L R+J^2 R^2}\right|\right.$$
$$\left.\sqrt{-4 J L\left(\text{Ke}\,\text{Kt}+b R\right)+\left(b L+J R\right)^2}\;\left|v\right|\right)\in\mathbb{R}\;\&\&$$
$$\sqrt{-4 J\,\text{Ke}\,\text{Kt}\,L+b^2 L^2-2 b J L R+J^2 R^2}>0\;\&\&\;\sqrt{-4 J\,\text{Ke}\,\text{Kt}\,L+b^2 L^2-2 b J L R+J^2 R^2}<b L+J R\;\&\&$$
$$\left.b L+J R+\sqrt{-4 J\,\text{Ke}\,\text{Kt}\,L+b^2 L^2-2 b J L R+J^2 R^2}>0\right]$$

Good: the two limits match. Also, the conditional predicate in the hard-way approach illustrates the conditions under which the final value theorem is applicable, namely (a) that all non-zero roots of the denominator of the transform must have negative real parts, and (b) the transform must not have more than one pole at the origin. See here for more on that topic.