Ripple Current in Asynchronous Sign Magnitude HBridges

Robert Atkinson
26 August 2018

Here we investigate the ripple current of asynchronous sign magnitude H bridges. The development here is based on earlier work by Eric Chin (private communication), and on the work of Andras Tantos at the H Bridge Secrets site (http://modularcircuits.tantosonline.com/blog/articles/h-bridge-secrets/asynchronous-sign-magnitude-drive/). Our most notable result is exhibiting the relationship between PWM duty cycle and steady state motor velocity for asynchronous sign magnitude H bridges.

1. Introduction

1.1. Administrivia

Before we begin, we load in some previously computed logic (Ref: https://github.com/rgatkinson/RobotPhysics/blob/master/MotorPhysics-GearsInitialConditions.pdf)

```
Get[NotebookDirectory[] <> "Utilities.m"]
inputDirectory = FileNameJoin[{NotebookDirectory[], "MotorPhysicsGearsInitialConditions.Output"}] <> $PathnameSeparator;
Get[inputDirectory <> "ParametersUnitsAndAssumptions.m"];
Get[inputDirectory <> "MotorModels.m"];
Get[inputDirectory <> "MotorTimeDomainFunctions.m"];
Get[inputDirectory <> "Misc.m"];
```

```
SetOptions[Plot, LabelStyle → Directive[Background → None]];
```

2. Examples

Before we get going, we take a moment to load in parameters of an example motor, here an AndyMark NeveRest 60. To that we attach a flywheel, as is our usual practice (that said, it turns out in the below that the *inertial* load is irrelevant to our results, only the *drag* load is).

```
aMotor = motorParameters["AM 60 A"];
unitlessMotor = aMotor // siUnits // clearUnits // N
aMotorWithLoad = addMotorLoad[aMotor, flywheel[Quantity[5, "kg"], Quantity[10, "cm"]]];
unitlessMotorWithLoad = aMotorWithLoad // siUnits // clearUnits // N;
unitlessMotorWithLoad = unitlessMotorWithLoad // gearboxlessEquivalent
```

```
⟨|R → 3.3, L → 0.000694, ℕ → 60., η → 0.9, Ke → 0.0177667, Kt → 0.0177667,
 B → 0.0000101852, J → 3.21296×10^-9, Jafter → 0., Bafter → 0., ΔτappConst → 0.|⟩
```

```
⟨|R → 3.3, L → 0.000694, ℕ → 1, η → 1, Ke → 0.0177667, Kt → 0.0177667,
 B → 0.0000101852, J → 7.71926×10^-6, Jafter → 0, Bafter → 0, ΔτappConst → 0.|⟩
```

We also define some parameters that mirror our environment; the battery voltage, the voltage drop across a silicon diode, and the PWM frame period used in the First Tech Challenge REV Expansion Hub firmware (Ref: http://www.revrobotics.com/rev-31-1153/).

```
env = {vbat → 12, ΔvappConst → 12, vdiode → 0.7, p → 0.0001 (*10 kHz*)} // Association
```

```
⟨|vbat → 12, ΔvappConst → 12, vdiode → 0.7, p → 0.0001|⟩
```

A handy figure to keep in mind: what is the maximum velocity of this example motor (in rad/s)?

```
velMaxExample = ssVel /. simpleSs /. Join[unitlessMotorWithLoad, env]
```

```
610.424
```

3. Ripple Current Analysis

3.1. Current at the Start and End of the PWM Cycle

Our examination of ripple current is rooted in the voltage differential equation of our motor model. Here, we will assume that due to the short time intervals involved in a PWEM cycle that velocity is constant. Thus, *vg[t]* will be a constant *Ke* Ω, where Ω is the current instantaneous velocity.

```
diffEqns[[3]] /. {vg[t] → Ke Ω}
```

```
Ke Ω + R i[t] + L i′[t] == vapp[t]
```

3.1.1. PWM On Interval

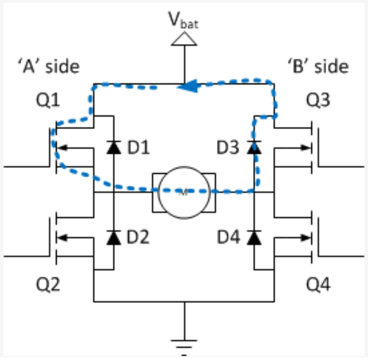
In the on part of the PWM cycle, the applied voltage *vapp[t]* is the battery voltage. Given a value for the current at the start of the interval, we solve for current throughout the interval.

```
Clear[currentPwmOn]
currentPwmOn[initialCurrent_] := Module[{t, eqns, initialConditions, result},
  eqns = {R i[t] + Ke Ω + L i'[t] == vbat};
  initialConditions = {i[0] == initialCurrent};
  result = DSolve[eqns ~Join~ initialConditions, i[t], t] // Flatten // FullSimplify;
  Function[tt, i[t] /. result /. t → tt]]
currentPwmOn[initialCurrentOn][t]
```

$$\frac{e^{-\frac{R t}{L}}\left(\text{initialCurrentOn}\,R+\left(-1+e^{\frac{R t}{L}}\right)(\text{vbat}-\text{Ke}\,\Omega)\right)}{R}$$

3.1.2. PWM Off Interval

In the off part of the cycle, the applied voltage is zero. Also, in our H bridges, in the off part of the cycle a voltage drop across a diode is involved (Ref: http://modularcircuits.tantosonline.com/blog/articles/h-bridge-secrets/asynchronous-sign-magnitude-drive/)



Given a value for the current at the start of the interval, we solve for the current throughout the interval.

```
Clear[currentPwmOff]
currentPwmOff[initialCurrent_] := Module[{t, eqns, initialConditions, result},
  eqns = {R i[t] + Ke Ω + L i'[t] + vdiode == 0};
  initialConditions = {i[0] == initialCurrent};
  result = DSolve[eqns ~Join~ initialConditions, i[t], t] // Flatten // FullSimplify;
  Function[tt, i[t] /. result /. t → tt]]
currentPwmOff[initialCurrentOff][t]
```

$$-\frac{\text{vdiode}+\text{Ke}\,\Omega-e^{-\frac{R\,t}{L}}\left(\text{initialCurrentOff}\,R+\text{vdiode}+\text{Ke}\,\Omega\right)}{R}$$

3.1.3. Constraints

As the PWM cycle is, well, cyclical, we require that the current values at the end of the on and off intervals match at both ends. We capture those constraints as a pair of equations. Here, *d* is the duration of the on part of the cycle (known hereinafter as “on-duration”), and *p* is the duration of the overall PWM frame period (“frame duration”).

```
currentPwmOn[initialCurrentOn][d] == currentPwmOff[initialCurrentOff][0]
currentPwmOn[initialCurrentOn][0] == currentPwmOff[initialCurrentOff][p - d]
constraints = {%, %%};
```

$$\frac{e^{-\frac{d\,R}{L}}\left(\text{initialCurrentOn}\,R+\left(-1+e^{\frac{d\,R}{L}}\right)\left(\text{vbat}-\text{Ke}\,\Omega\right)\right)}{R}==\text{initialCurrentOff}$$

$$\text{initialCurrentOn}==-\frac{\text{vdiode}+\text{Ke}\,\Omega-e^{-\frac{(-d+p)\,R}{L}}\left(\text{initialCurrentOff}\,R+\text{vdiode}+\text{Ke}\,\Omega\right)}{R}$$

3.1.4. Solving for initial/final current as a function of duty cycle, velocity, motor, and environment

We’re not actually interested in the current at the start of the off period, so we eliminate it from the equation.

```
Eliminate[constraints, initialCurrentOff] /. And → List
initialCurrentOnEqn = FullSimplify[Eliminate[constraints, initialCurrentOff], parameterAssumptions]
```

$$\Big\{e^{-\frac{(-d+p)\,R}{L}}\left(-1+e^{-\frac{d\,R}{L}}\right)\text{vbat}==e^{-\frac{(-d+p)\,R}{L}}\left(e^{-\frac{d\,R}{L}}\,\text{initialCurrentOn}\,R-e^{-\frac{(d-p)\,R}{L}}\,\text{initialCurrentOn}\,R+\text{vdiode}-e^{-\frac{(d-p)\,R}{L}}\,\text{vdiode}+e^{\frac{d\,R}{L}}\,\text{Ke}\,\Omega-e^{-\frac{(d-p)\,R}{L}}\,\text{Ke}\,\Omega\right),$$
$$e^{-\frac{(-d+p)\,R}{L}}\left(-1+e^{\frac{d\,R}{L}}\right)\text{vbat}==e^{-\frac{(-d+p)\,R}{L}}\left(-\text{initialCurrentOn}\,R+e^{\frac{d\,R}{L}-\frac{(d-p)\,R}{L}}\,\text{initialCurrentOn}\,R-e^{\frac{d\,R}{L}}\,\text{vdiode}+e^{\frac{d\,R}{L}-\frac{(d-p)\,R}{L}}\,\text{vdiode}-\text{Ke}\,\Omega+e^{\frac{d\,R}{L}-\frac{(d-p)\,R}{L}}\,\text{Ke}\,\Omega\right),$$
$$R\neq 0\Big\}$$

$$\text{vbat}+e^{\frac{p\,R}{L}}\left(\text{initialCurrentOn}\,R+\text{vdiode}+\text{Ke}\,\Omega\right)==\text{initialCurrentOn}\,R+e^{\frac{d\,R}{L}}\left(\text{vbat}+\text{vdiode}\right)+\text{Ke}\,\Omega$$

We solve for the initial current, *initialCurrentOn*, as a function of the remaining parameters. This is an important result that we will use frequently in the below. It embodies the periodic nature of the PWM cycle, telling us what the current must be at the start (and end) of that cycle, given the other parameters (Ω, R, L, Ke, p, vbat, vdiode).

```
initialCurrentOnExpr = initialCurrentOn /. uniqueSolve[initialCurrentOnEqn, initialCurrentOn];
initialCurrentOnExpr = initialCurrentOnExpr // FullSimplify
```

$$\frac{\left(-1+e^{\frac{d\,R}{L}}\right)\text{vbat}+e^{\frac{d\,R}{L}}\,\text{vdiode}+\text{Ke}\,\Omega-e^{\frac{p\,R}{L}}\left(\text{vdiode}+\text{Ke}\,\Omega\right)}{\left(-1+e^{\frac{p\,R}{L}}\right)R}$$

3.1.5. When is the initial/final current zero?

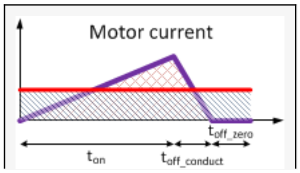
In the below, we will have particular interest in when the current at the start (and end) of the PWM cycle is zero. For a given velocity and on-duration, we compute the minimum PWM frame duration for which the initial (and final) current is zero. If the actual frame duration is longer than this value, the current will be discontinuous.

```
zeroCurrentFrameDurationFromVelocity[ΩΩ_, dd_] := Module[{soln },
  soln = uniqueSolve[initialCurrentOnExpr == 0, p] /. C[1] → 0; (*constant is zero to force real result, not complex*)
  (p /. soln) /. {Ω → ΩΩ, d → dd} (*// FullSimplify*)
 ]
zeroCurrentFrameDurationFromVelocity[Ω, d]
```

$$\frac{\mathrm{L}\,\mathrm{Log}\left[\frac{-\mathrm{vbat}+e^{\frac{\mathrm{d}\,\mathrm{R}}{\mathrm{L}}}\,\mathrm{vbat}+e^{\frac{\mathrm{d}\,\mathrm{R}}{\mathrm{L}}}\,\mathrm{vdiode}+\mathrm{Ke}\,\Omega}{\mathrm{vdiode}+\mathrm{Ke}\,\Omega}\right]}{\mathrm{R}}$$

3.2. Waveform of Full PWM Frame

As is observed in the H Bridge Secrets site, asynchronous sign magnitude bridges can, depending on the parameters involved, exhibit either *continuous* or *discontinuous* current through each of their PWM cycles, according to whether current continues to flow throughout the whole PWM frame or whether it becomes zero before the frame is complete. Understanding the parameters that determine whether we are in a continuous or a discontinuous regime is a primary focus of ours here. (Ref: http://modularcircuits.tantosonline.com/blog/articles/h-bridge-secrets/asynchronous-sign-magnitude-drive/).



While we have modeled the voltage drop across the diode in our development above, we did not model the diode’s rectifying behavior. That is, the differential equations above are in fact **invalid** and **incorrect** if the current ever becomes negative. Thus, we cannot correctly use them on their own by themselves to model the system when in discontinuous operation. Rather, we must take care by *other* means to determine when the current would go negative and modify our waveforms appropriately.

With that caution in mind, we our on-cycle and off-cycle results together to produce a function that shows the full PWM current waveform over an entire PWM cycle. Ensuring that the computed initial current is non-negative and that our PWM-on and PWM-off waveforms never exceed the zero-current frame duration is sufficient to produce correct results.

```
Clear[pwmCurrent]
pwmCurrent[ΩΩ_, dd_, pp_] := Module[
  {rules = Association[{Ω → ΩΩ, d → dd, p → pp}], initialCurrentOn, initialCurrentOff, pZeroCurrent, on, off, cycle},
  (* Clamp the initial current to be non-negative. Needed if we're in discontinuous current operation. *)
  initialCurrentOn = Max[0, initialCurrentOnExpr];
  initialCurrentOff = currentPwmOn[initialCurrentOn][d];
  pZeroCurrent = zeroCurrentFrameDurationFromVelocity[ΩΩ, dd];
  on = Piecewise[{{currentPwmOn[initialCurrentOn][t], 0 ≤ t ≤ d}}, 0];
  off = Piecewise[{{currentPwmOff[initialCurrentOff][t - d], d < t ≤ p}}, 0];
  cycle = Piecewise[{
     {currentPwmOn[initialCurrentOn][t], 0 ≤ t ≤ d && t ≤ pZeroCurrent}, (*clip overly-long on-duration*)
     {currentPwmOff[initialCurrentOff][t - d], d ≤ t ≤ p && t ≤ pZeroCurrent} (*clip overly-long frame-duration*)
    },
    0];
  cycle = cycle /. rules;
  cycle
 ]
pwmCurrent[Ω, d, p] // prettyPrint
```

$$\begin{cases} \frac{1}{R}e^{-\frac{R t}{L}}\left(R \max\left(0, \left(\mathrm{vbat}\left(e^{\frac{d R}{L}}-1\right)+\mathrm{vdiode}\, e^{\frac{d R}{L}}-(\mathrm{Ke}\,\Omega+\mathrm{vdiode})\, e^{\frac{p R}{L}}+\mathrm{Ke}\,\Omega\right)\Big/\left(R\left(e^{\frac{p R}{L}}-1\right)\right)\right)+ \right. \\ \qquad \left. (\mathrm{vbat}-\mathrm{Ke}\,\Omega)\left(e^{\frac{R t}{L}}-1\right)\right) & 0 \le t \le d \land t \le \frac{L \log\left(\frac{\mathrm{vbat}\, e^{\frac{d R}{L}}+\mathrm{vdiode}\, e^{\frac{d R}{L}}+\mathrm{Ke}\,\Omega-\mathrm{vbat}}{\mathrm{Ke}\,\Omega+\mathrm{vdiode}}\right)}{R} \\ -\frac{1}{R}\left(-e^{-\frac{R t}{L}}\left(R \max\left(0, \left(\mathrm{vbat}\left(e^{\frac{d R}{L}}-1\right)+\mathrm{vdiode}\, e^{\frac{d R}{L}}-(\mathrm{Ke}\,\Omega+\mathrm{vdiode})\, e^{\frac{p R}{L}}+\mathrm{Ke}\,\Omega\right)\Big/\left(R\left(e^{\frac{p R}{L}}-1\right)\right)\right)+ \right.\right. \\ \qquad \left.\left. (\mathrm{vbat}+\mathrm{vdiode})\, e^{\frac{d R}{L}}+\mathrm{Ke}\,\Omega-\mathrm{vbat}\right)+\mathrm{Ke}\,\Omega+\mathrm{vdiode}\right) & d \le t \le p \land t \le \frac{L \log\left(\frac{\mathrm{vbat}\, e^{\frac{d R}{L}}+\mathrm{vdiode}\, e^{\frac{d R}{L}}+\mathrm{Ke}\,\Omega-\mathrm{vbat}}{\mathrm{Ke}\,\Omega+\mathrm{vdiode}}\right)}{R} \end{cases}$$

3.3. Mean Current over PWM Frame

With pwmCurrent[] in hand, the mean current across a PWM duty cycle can be computed as the integral of current over the PWM frame divided by the duration of the frame. We provide two forms, one fully general, and one numeric.

```
parameterAssumptions = Union[parameterAssumptions, {Ω ≥ 0, 0 ≤ d ≤ p, p > 0 }];
Clear[meanCurrent, numericalMeanCurrent]
meanCurrent[Ω_, d_, p_] := meanCurrent[Ω, d, p, {}];
meanCurrent[Ω_, d_, p_, rules_] := meanCurrent[Ω, d, p, rules] = Module[{expr, pVal, range, assumptions, result},
   expr = pwmCurrent[Ω, d, p] /. rules; (*Print[expr];*)
   pVal = p /. rules;
   range = {t, 0, pVal};
   assumptions = parameterAssumptions;
   result = (Assuming[assumptions, Integrate[expr, range]] / pVal);
   result]
numericalMeanCurrent[Ω_, d_, p_] := numericalMeanCurrent[Ω, d, p, {}];
numericalMeanCurrent[Ω_, d_, p_, rules_] :=
 numericalMeanCurrent[Ω, d, p, rules] = Module[{expr, pVal, range, assumptions, result},
   expr = pwmCurrent[Ω, d, p] /. rules; (*Print[expr];*)
   pVal = p /. rules;
   range = {t, 0, pVal};
   assumptions = parameterAssumptions;
   result = (Assuming[assumptions, NIntegrate[expr, range]] / pVal);
   result]
Block[{prettyPrintFontSize = 16}, meanCurrent[Ω, d, p] // prettyPrint]
```

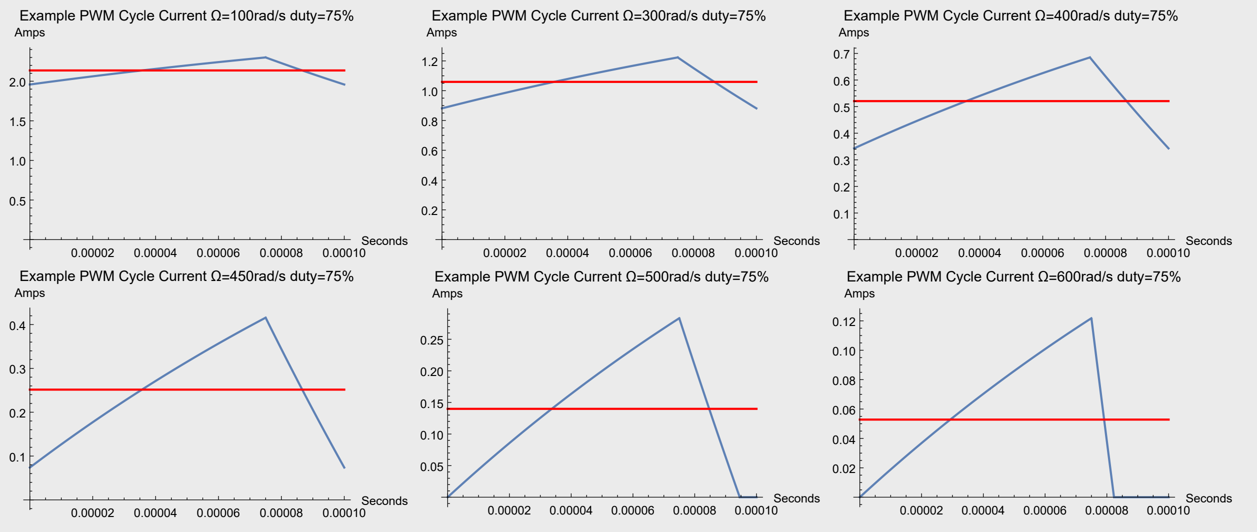
$$\frac{1}{p}\Bigg(\int_0^p\Bigg(\begin{cases} \begin{array}{l} \frac{1}{R}e^{-\frac{Rt}{L}}\bigg(\Big(-1+e^{\frac{Rt}{L}}\Big)(\text{vbat}-\text{Ke}\,\Omega)+ \\ \qquad R\max\bigg(0,\Big(\Big(-1+e^{\frac{dR}{L}}\Big)\text{vbat}+e^{\frac{dR}{L}}\,\text{vdiode}+\text{Ke}\,\Omega- \\ \qquad\qquad e^{\frac{pR}{L}}(\text{vdiode}+\text{Ke}\,\Omega)\Big)\Big/\Big(\Big(-1+e^{\frac{pR}{L}}\Big)R\Big)\bigg)\bigg) \end{array} & \begin{array}{l} 0\le t\le d\land t\le\frac{1}{R} \\ \quad L\log\Big(\Big(e^{\frac{dR}{L}}\,\text{vbat}-\text{vbat}+e^{\frac{dR}{L}}\,\text{vdiode}+\text{Ke}\,\Omega\Big)\Big/(\text{vdiode}+\text{Ke} \end{array} \\ \begin{array}{l} -\frac{1}{R}\bigg(\text{vdiode}+\text{Ke}\,\Omega-e^{-\frac{Rt}{L}}\bigg(-\text{vbat}+e^{\frac{dR}{L}}(\text{vbat}+\text{vdiode})+\text{Ke}\,\Omega+ \\ \qquad R\max\bigg(0,\Big(\Big(-1+e^{\frac{dR}{L}}\Big)\text{vbat}+e^{\frac{dR}{L}}\,\text{vdiode}+\text{Ke}\,\Omega- \\ \qquad\qquad e^{\frac{pR}{L}}(\text{vdiode}+\text{Ke}\,\Omega)\Big)\Big/\Big(\Big(-1+e^{\frac{pR}{L}}\Big)R\Big)\bigg)\bigg)\bigg) \end{array} & \begin{array}{l} d\le t\le p\land t\le\frac{1}{R} \\ \quad L\log\Big(\Big(e^{\frac{dR}{L}}\,\text{vbat}-\text{vbat}+e^{\frac{dR}{L}}\,\text{vdiode}+\text{Ke}\,\Omega\Big)\Big/(\text{vdiode}+\text{Ke} \end{array} \end{cases}$$

3.4. Example

To get a sense of it all, we plot the PWM cycle for our example motor at a sequence of increasing velocities. We also plot the mean current over the cycle. Note that the vertical scales are different in each of the plots.

```
Clear[plotCycle, plotMean, plotBoth]
plotCycle[ΩΩ_, dd_, pp_, motor_] := Module[
  {pRules = {Ω → ΩΩ, d → dd, p → pp} // Association, rules, cycle, duty},
  rules = Join[pRules, motor, env];
  cycle = pwmCurrent[ΩΩ, dd, pp] //. rules;
  duty = (dd / pp) //. rules;
  Plot[cycle, Evaluate[{t, 0, (pp //. rules)}],
   AxesOrigin → {0, 0}, AxesLabel → {"Seconds", "Amps"},
   ImageSize → 300,
   PlotLabel →
    Evaluate["Example PWM Cycle Current Ω=" <> ToString[ΩΩ] <> "rad/s duty=" <> ToString[duty * 100 // Rationalize] <> "%" ]]
 ]
plotMean[ΩΩ_, dd_, pp_, motor_] := Module[{mean},
   mean = numericalMeanCurrent[ΩΩ, dd, pp, (motor ~ Join ~ env)];
   Plot[mean, Evaluate[{t, 0, pp //. env}, PlotStyle → Red]]];
plotBoth[Ω_, d_, p_, motor_] := Module[{mean, cycle},
   mean = plotMean[Ω, d, p, motor];
   cycle = plotCycle[Ω, d, p, motor];
   Show[cycle, mean]];
```

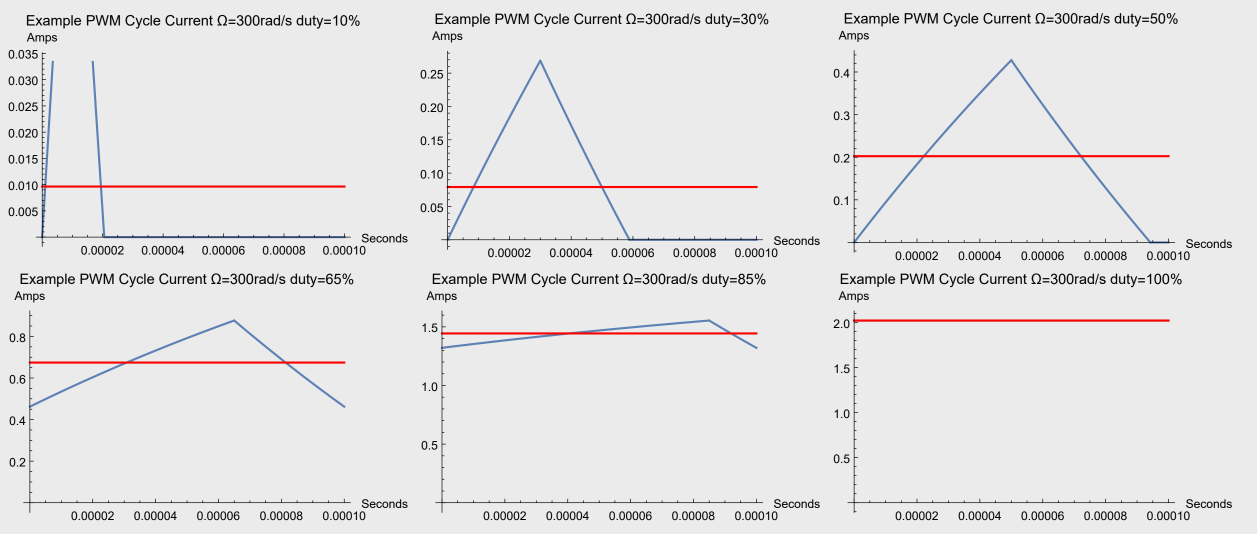
```
plots = plotBoth[#, 0.75 p, p, unitlessMotorWithLoad] & /@ {100, 300, 400, 450, 500, 600};
Grid[{{plots[[1]], plots[[2]], plots[[3]]}, {plots[[4]], plots[[5]], plots[[6]]}}]
```



Of particular interest is that at higher velocities, the current returns to zero before the PWM framing period is complete. In those situations, we have discontinuous current.

Let’s also examine a fixed velocity while we vary duty cycle.

```
plots = plotBoth[300, # p, p, unitlessMotorWithLoad] & /@ {.1, .3, .5, .65, .85, 1};
Grid[{{plots[[1]], plots[[2]], plots[[3]]}, {plots[[4]], plots[[5]], plots[[6]]}}]
```



That makes sense: for fixed velocity, as duty cycle increases, we eventually transition from discontinuous to continuous current once the duty cycle is sufficiently large to develop enough energy during the on part of the cycle so that it is not exhausted by cycle's end.

However, except in transients, we don't actually get to vary duty cycle and velocity independently as these plots would imply. It is that relationship to which we now turn our attention.

4. Continuity, Discontinuity, Duty Cycle, and Velocity

Our ultimate interest here is to understand the relationship between duty cycle and velocity in asynchronous sign magnitude H bridges. In particular, we're interested in the nature of any non-linearities involved.

4.1. Mean Current, Duty Cycle, and Steady State Velocity

So far we have treated velocity (Ω) and duty cycle as independent variables. However, they are in fact related: for a given duty cycle (and other parameters), there is a limit to the maximum velocity which may be achieved. We seek now to compute that value. To do that, we recall the steady state current and velocity results from the motor modeling work.

```
ssCur == (ssCur /. simpleSs)
ssVel == (ssVel /. simpleSs)
```

$$\mathrm{ssCur} == \frac{\mathrm{B}\,\triangle\mathrm{vappConst}}{\mathrm{Ke}\,\mathrm{Kt} + \mathrm{B}\,\mathrm{R}}$$

$$\mathrm{ssVel} == \frac{\mathrm{Kt}\,\triangle\mathrm{vappConst}}{\mathrm{Ke}\,\mathrm{Kt} + \mathrm{B}\,\mathrm{R}}$$

From those, we define a function that computes the steady state velocity for a given steady state current. Notice that the result is not dependent on the inertia of the load & motor, only the drag.

```
ssVelFromCurrent[current_] := Module[{},
  ssVel /. simpleSs /. uniqueSolve[ssCur == (ssCur /. simpleSs), ΔvappConst] /. ssCur → current
 ]
ssVelFromCurrent[steadyStateCurrent]
```

$$\frac{\mathrm{Kt}\ \mathrm{steadyStateCurrent}}{\mathrm{B}}$$

We can now compute the steady state velocity for a given duty cycle. First, we compute the mean current in terms of duty cycle, velocity and other parameters, and from that compute the implied steady state velocity. At the actual steady state velocity for the duty cycle, that implied velocity must in fact be the same as the velocity we used to compute the mean current in the first place. To find that value, we subtract the velocity from the implied velocity, then numerically find the root.

```
Clear[ssVelFromPwmCycle]
ssVelFromPwmCycle[d_, p_, motor_] := ssVelFromPwmCycle[d, p, motor, env]
ssVelFromPwmCycle[d_, p_, motor_, env_] := Module[{rules = (motor ~Join~ env), dVal, pVal, Ω, mean, eqn, soln, result},
  pVal = p //. rules; (*If[!NumberQ[pVal], Print["warning: p should be numeric: "<> ToString[pVal]]];*)
  dVal = d //. rules; (*If[!NumberQ[dVal], Print["warning: d should be numeric: " <> ToString[dVal]]];*)
  (*Print[Riffle[{dVal, pVal}, " "]];*)
  mean = meanCurrent[Ω, dVal, pVal, rules] // FullSimplify;
  eqn = (ssVelFromCurrent[mean] - Ω) /. rules;
  eqn = PiecewiseExpand[eqn, Ω > 0];
  soln = FindRoot[eqn, {Ω, 300}];
  result = Ω /. soln;
  (*Print[dVal/pVal, " ", result];*)
  result]
ssVelFromPwmCycle[# p, p, unitlessMotorWithLoad] & /@ {1/4, 1/2, 3/4, 1}
```

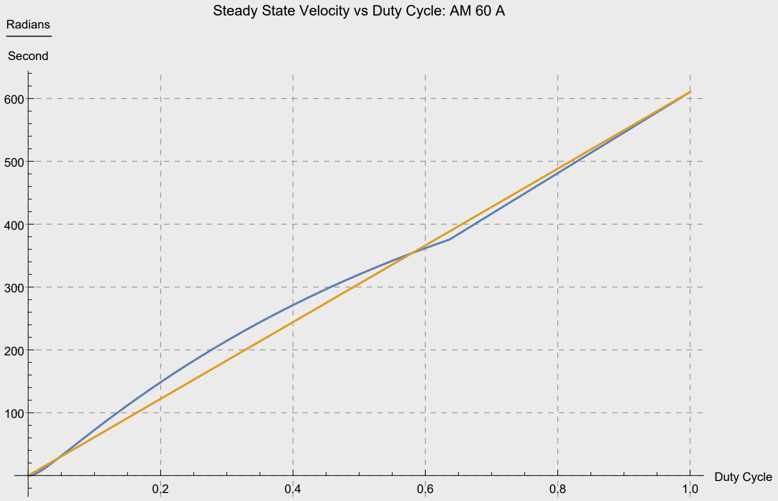
```
{182.72, 320.085, 448.916, 610.424}
```

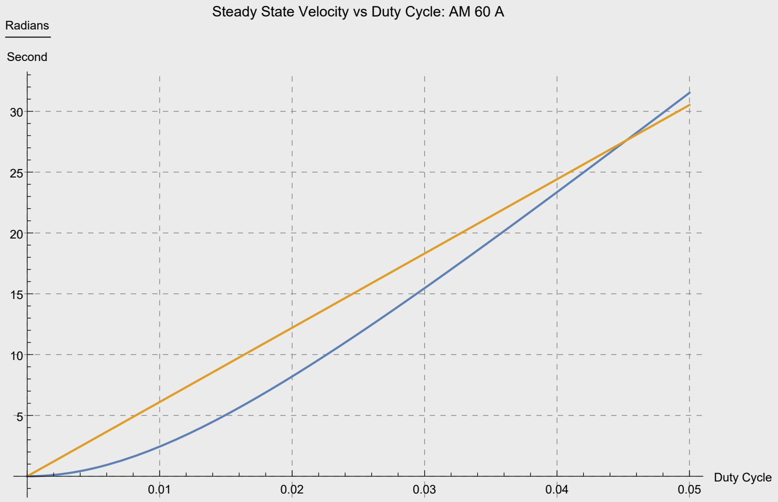
4.2. Steady State Velocity vs Duty Cycle

We’re now ready to look at how steady state velocity varies with duty cycle, which is one of our primary concerns here. We plot that curve for several different FTC motors; for each, we look at both the full curve and at the initial interval. We compare with a perfectly linear response as a reference.

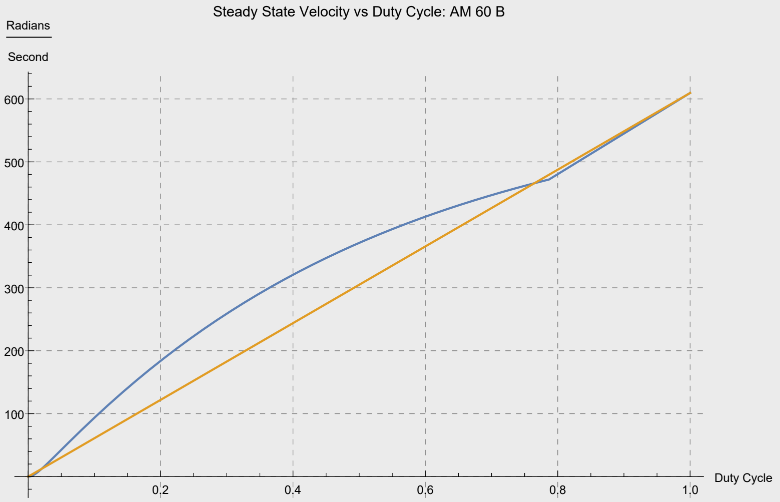
```
Clear[plotSsVelFromPwmCycle]
Options[plotSsVelFromPwmCycle] = { Title → "" };
plotSsVelFromPwmCycle[motor_, opts : OptionsPattern[]] := plotSsVelFromPwmCycle[motor, env, opts]
plotSsVelFromPwmCycle[motor_, env_, opts : OptionsPattern[]] := Module[{title, plotIt},
  title = "Steady State Velocity vs Duty Cycle" <> If[StringLength[OptionValue[Title]] > 0, ": " <> OptionValue[Title], ""];
  plotIt[tMax_] := Plot[
    {
     ssVelFromPwmCycle[duty p, p, motor, env],
     line[{0, 0}, {1, ssVel /. simpleSs /. Join[motor, env]}][duty]
    }, {duty, 0, tMax},
    AxesOrigin → {0, 0},
    AxesLabel → {"Duty Cycle", HoldForm[Radians / Second]},
    PlotLabel → title,
    ImageSize → Large,
    GridLines → Automatic,
    GridLinesStyle → Directive[Gray, Dashed]
   ];
  Row[{plotIt[1], plotIt[0.05]}]]
```

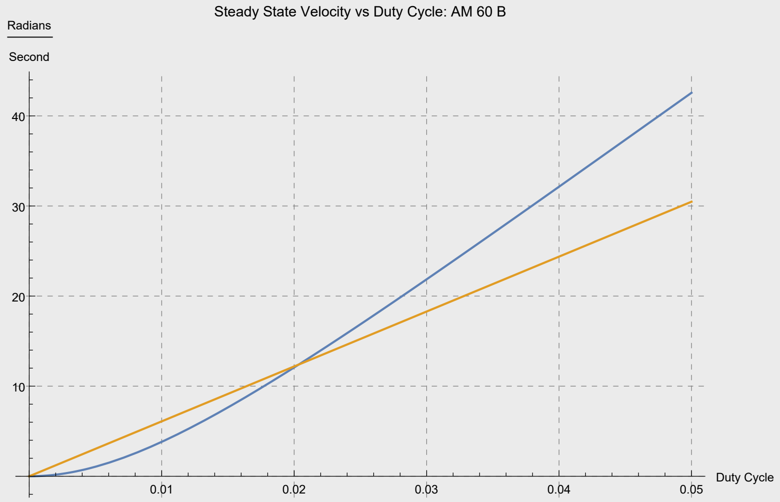
```
plotSsVelFromPwmCycle[motorParameters["AM 60 A"] // clearUnits // N, Title → "AM 60 A"]
```



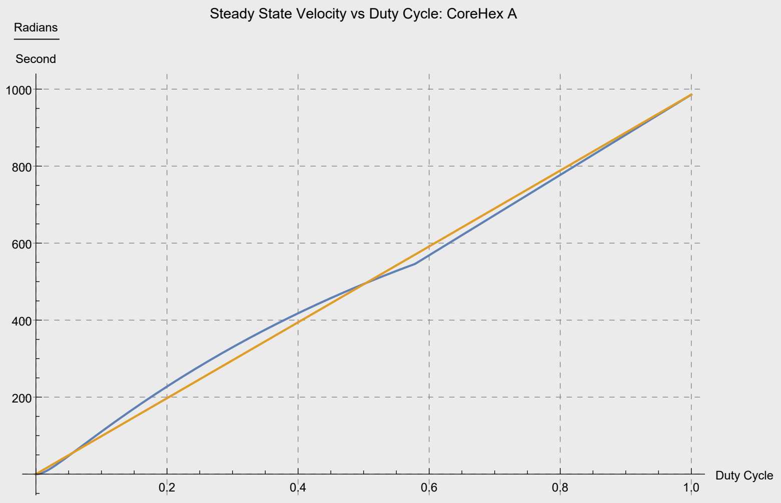


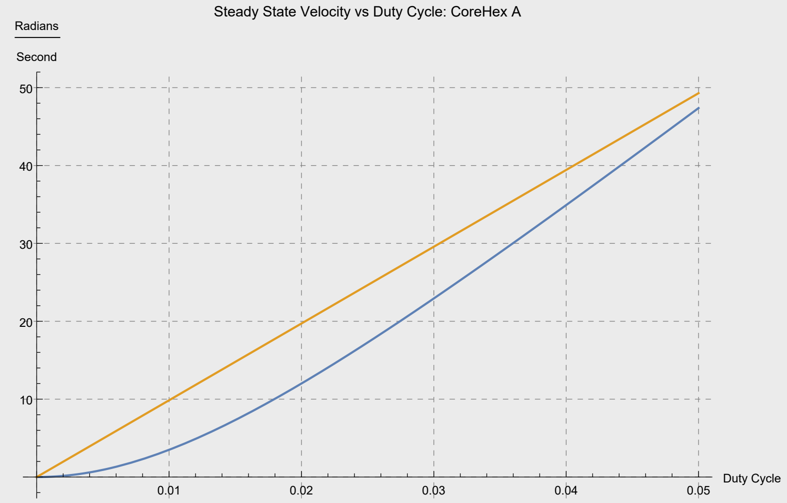
```
plotSsVelFromPwmCycle[motorParameters["AM 60 B"] // clearUnits // N, Title → "AM 60 B"]
```





```
plotSsVelFromPwmCycle[motorParameters["CoreHex A"] // clearUnits // N, Title → "CoreHex A"]
```





These are interesting results. To investigate them further, we first need a function that gives the zero current frame duration for the steady state velocity of a given on duration.

```
Clear[zeroCurrentFrameDurationFromOnDuration]
zeroCurrentFrameDurationFromOnDuration[d_, p_, motor_] := zeroCurrentFrameDurationFromOnDuration[d, p, motor, env]
zeroCurrentFrameDurationFromOnDuration[d_, p_, motor_, env_] := Module[{Ω, pZero},
  Ω = ssVelFromPwmCycle[d, p, motor, env];
  pZero = zeroCurrentFrameDurationFromVelocity[Ω, d] /. Join[motor, env];
  pZero]
```

We evaluate the zero current duration for the 40% duty cycle of each of the above examples.

```
zeroCurrentFrameDurationFromOnDuration[.4 p, p, motorParameters["AM 60 A"] // clearUnits // N]
zeroCurrentFrameDurationFromOnDuration[.4 p, p, motorParameters["AM 60 B"] // clearUnits // N]
zeroCurrentFrameDurationFromOnDuration[.4 p, p, motorParameters["CoreHex A"] // clearUnits // N]
```

```
0.0000827259
```

```
0.0000700376
```

```
0.0000870958
```

Each of these frame durations is less than the actual frame duration of .0001 seconds; this is indicative of discontinuity. That is, each motor operates in a discontinuous regime for a substantially-sized initial part of the duty cycle range. While discontinuous, the steady state velocity response is convexly supra-linear (except for a very short initial part of the cycle), but not *overly* strongly so (maybe a few tens of percent). In the upper part of the duty cycle range, there is a transition to continuity; thereafter, the incremental response is linear. While the overall non-linearity is tame enough that it will be adapted to by closed-loop control system, users of these motors in open-loop systems may wish to take note.

We complete our study by locating the duty cycle at which the transition from discontinuous to continuous operation occurs. Having defined the function that gives the zero current frame duration for the steady state velocity of a given on duration, we numerically search for where that zero current frame duration is the actual frame duration.

```
Clear[findContinuityTransition]
findContinuityTransition[title_] := findContinuityTransition[motorParameters[title], title]
findContinuityTransition[motorParam_, title_] := Module[{motor, pVal, fn, transitionFrameDuration},
  motor = motorParam // clearUnits // N;
  pVal = p /. env;
  fn[duty_?NumberQ] := zeroCurrentFrameDurationFromOnDuration[duty pVal, pVal, motor, env] - pVal;
  transitionFrameDuration = FindRoot[fn[duty], {duty, 0.5}(*, EvaluationMonitor :→ Print[duty]*)];
  title → (duty /. transitionFrameDuration)
 ]
```

```
findContinuityTransition[#] & /@ {"AM 60 A", "AM 60 B", "CoreHex A"} // ColumnForm
```

```
AM 60 A → 0.636524
AM 60 B → 0.786845
CoreHex A → 0.578523
```

5. Revision History

- 2018.08.26. Initial version.