3-Phase Oscillator

The basic idea is to provide a linear approximation to the progression of phases so that an iterative calculation can be done (hopefully without the use of hardware multipliers) to compute the drive phase as a function of time. We plan to encode the state variables as Sigma/Delta streams to lower the hardware cost and to provide a ready digitized encoding for the drive modulator. It is very desirable that the states used be the motor drive output level as this simplifies the implementation of controls. (The generic plan is to make use of the fact that the motor is effectively synchronous with the phases – so that motor winding current returns can be directly compared to the desired motor state. Encoder returns will need to be converted to relative angle – but that is available as a linear approximation from the integration of the scalar angular velocity (Rad/sec)).

State

Assuming we can choose the obvious one:

$$f[t\_]:=\{\text{Cos}[t],\text{Cos}[t+2\text{Pi}/3],\text{Cos}[t+4\text{Pi}/3]\}$$

This is a simple vector of the phases – and will be the state. Initialization is f(0):

$$f[0]\ \left\{1,-\frac{1}{2},-\frac{1}{2}\right\}$$

Projector

It is easiest to keep track of the trig if we look at the update of the first term:

$$\text{a1}[e\_]:=\text{Cos}[t+e]$$

Here we assume that e is going to be small, and t is the current scalar position ordinate.

$$\text{a1}[e]=\text{Cos}[e]\text{Cos}[t]-\text{Sin}[e]\text{Sin}[t]$$

Every term here is fine except Sin(t) – we don't have it and need to find a way to express it in terms of the state (phases) we do have.

$$\text{Cos}[t+2\text{Pi}/3]=-\frac{\text{Cos}[t]}{2}-\frac{1}{2}\sqrt{3}\text{Sin}[t]$$

Here is the clue – we know Cos(t) as it is phase1, so Sin(t) can be expressed in terms of phase1 and phase2. Similarly it can be expressed in terms of phase1 and phase3. We will choose to use all three phases to enhance the symmetry and thus the numerical stability of the result.

$-1/\text{Sqrt}[3] * (\text{Cos}[t + 2\text{Pi}/3] - \text{Cos}[t + 4\text{Pi}/3]) = \text{Sin}[t]$

So the required term is a linear scaled sum of phase2 and phase3. Applying 39 above to 35 we get the exact relation:
Cos(t+e)=Cos(t)Cos(t)+Sin(e)/Sqrt(3)*(Cos(t+2Pi/3)-Cos(t+4Pi/3))
and thus:

$$A[e] = \begin{pmatrix} \text{Cos}[e] & \frac{\text{Sin}[e]}{\sqrt{3}} & -\frac{\text{Sin}[e]}{\sqrt{3}} \\ -\frac{\text{Sin}[e]}{\sqrt{3}} & \text{Cos}[e] & \frac{\text{Sin}[e]}{\sqrt{3}} \\ \frac{\text{Sin}[e]}{\sqrt{3}} & -\frac{\text{Sin}[e]}{\sqrt{3}} & \text{Cos}[e] \end{pmatrix}$$

Matrix A(e) will take state f(t) to state f(t+e), exactly:

$f[t + e] - A[e].\ f[t] = \{0, 0, 0\}$

Performance

It is desirable to have a slight positive numerical gain during the calculation so that the saturation (provided by special logic for overflow) keeps the approximation stable. Building the projector using both other phases is intended to maintain strict phase relative values at all times. The simplest approximation is small angle: Sin(e) -> e Cos(e)-> 1. Note that The actual Cos(e) term is on the diagnonal and is slightly and phase-symmetrically larger as it misses the -e^2/2 term. a[e] is then the approximate projector.

$$a[e\_] := \begin{pmatrix} 1 & \frac{e}{\sqrt{3}} & -\frac{e}{\sqrt{3}} \\ -\frac{e}{\sqrt{3}} & 1 & \frac{e}{\sqrt{3}} \\ \frac{e}{\sqrt{3}} & -\frac{e}{\sqrt{3}} & 1 \end{pmatrix}$$

This is cool – if we scale the increment, only small changes need be applied to update the 3 states, making the computation exactly integration of state (note 1 on diagonal) with a sigma-delta encoded update of the other states times a small constant...

$A[0.1] - a[0.1] =$

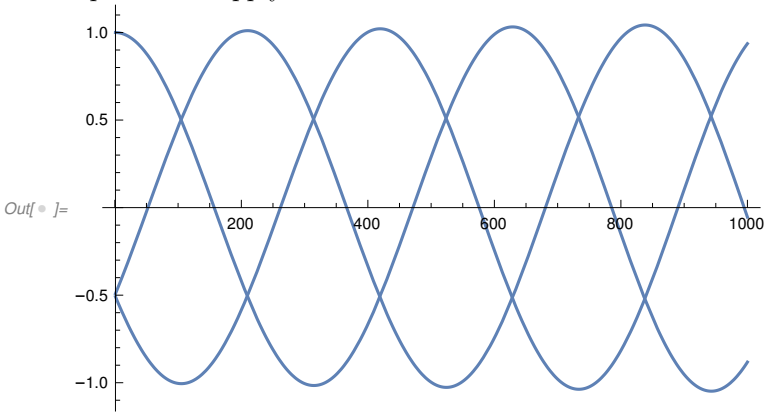
$\{\{-0.00499583, -0.0000961769, 0.0000961769\},$

$\{0.0000961769, -0.00499583, -0.0000961769\},$

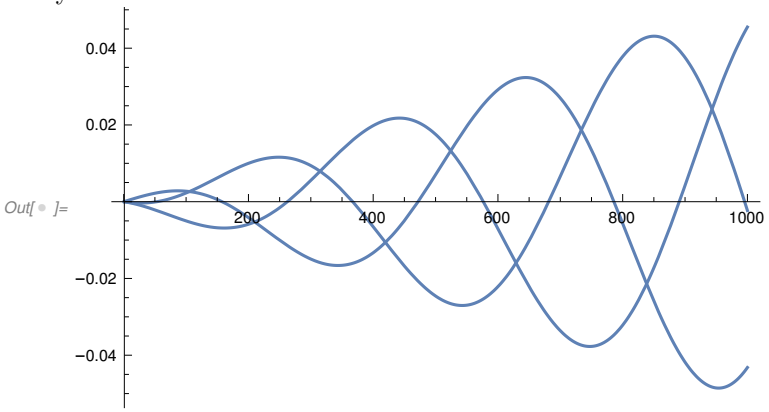
$\{-0.0000961769, 0.0000961769, -0.00499583\}\}$

As expected, errors are small for 74mR rotation, and diagonal has positive gain of 0.005 for this rotation. To test the approximate matrix, simply take it

to the nth power and apply it to the initial condition:



Direct integration looks pretty nice – with a slight growth in amplitude over 1000 iterations at 7mR/step – so just over 1 complete rotation. Again, we plan to saturate the maximum values to keep this under control. To see the error explicitly:



The error is gain exactly as planned, with first saturation of about 0.005 for this step size. At 12,000 RPM, we have 12000/60 = 200Rot/Sec, a step of 0.01/Sqrt(3)Rad $\sim$ 1100 steps/rot
or 220kHz. We are likely to run at several MHz internally, so this is no problem at all and errors should be below practical bit threshold.
Logic circuit is now just 3 states, 3 SD encoders, 6 muxes (scaled update) and adders...