Optimizing rotational acceleration curves for minimum energy use in electric motors.

12/15/06
Fall '06 T&AM 491 Final Report, 3 Credits
Andrew Spielberg
Engineering Physics 2010
Contact Information:
E-mail: Aes89@Cornell.edu
Cornell Phone: 1-(516)-426-5809
Cornell Address: 5244 High Rise 5, Ithaca, NY, 14853
Home Phone: 1-(516)-221-2774
Home Address: 3036 Clovermere Road, Wantagh,

Purpose/Abstract:

The purpose of this project is to determine what the most efficient acceleration path for rotating a wheel is; that is, we will see which acceleration curve uses the least amount of energy. This report will present the four different methods for calculating the minimum energy value and its corresponding accelerations, each more accurate than the last as we factor in asymmetry, friction, and negligible generator effects. For the purposes of this experiment, we will assume that we are rotating the motor an arbitrary 60 degrees in an arbitrary .5 seconds. The most important part of this experiment is to create a working model to find the optimal efficiency of a motor for any input values of motor constants, distance, or time. In every instance we will calculate the energy using a summation of very small acceleration piecewise "steps." Calculations are done with the MATLAB function fminunc.

Introduction:

For walking robots that simulate humanoid gaits, oftentimes the goal is to preserve the incredible efficiency that humans have in the translation. In the most recent lab project, the goal was to make the world's longest walking robot. This was a large part of the inspiration to find which rotational acceleration curve for the used motors would be most efficient. We really were not sure what the optimal acceleration curve would be; all we had was the hypothesis that a constant acceleration would be optimal. There was no evidence for this, and so we sought to compute the actual answer (which would turn out to be different from our hypothesis).

For the actual task, we decided to simulate the energy cost of spinning a hypothetical wheel on a motor. We chose the specific motor related constants used in this experiment based on corresponding motors used in our lab for the walking robot project. Other variables, such as the distance needed to rotate this wheel and the time allowed for the rotation were chosen arbitrarily. We chose to measure the rotation of the wheel 60 degrees over the time span of half a second. These values can easily be changed to fit the need of the application with no necessary change in the method. Four different methods were then implemented to determine the optimal acceleration path, each with the intention of being more realistic and accurate than the last.

I will now explain the basics between the four different methods implemented in determining the optimal acceleration path. The first step was to make the problem as simple as possible. This meant making a number of assumptions. Firstly, the acceleration curves would be symmetrical. In other words, whatever was done in the first half of the rotation would be mirrored in the second half of the rotation, but negated. Secondly, there would be no motor friction. Thirdly, the motor does not act as a generator (this is not really a simplification since it is probably true, as the specific motor we are using in this experiment has a negligible generator effect. Later, however we will assume the motor acts like a generator for computational reasons). This means that we are only considering the first half of the symmetrical curve. After this first model, we will then factor in asymmetry in the second model, and assume it acts like a complete

generator for computational purposes. The third model will modify the second model to incorporate friction. Last, we will discuss the possibility of modifying the third model as if the motor did not act like a generator.

One other key simplification is made here though. Since the mathematics for solving a single equation for the optimal acceleration over time is difficult, we will instead model this through a large number of short constant acceleration curves. We will therefore get a step-function that could be connected with a smooth line, and should theoretically give us the smooth curve function for which we are looking.

Equations:

We will begin by formulating our energy formula. The formula for the power used by a motor is:

$$\mathbf{P=I_c^2R + T\omega}$$

Where P is the power, I_c is the current through the motor, R is the resistance, T is the torque of the motor, and ω is the angular velocity. This is neglecting friction (other than the I_c^2R term) or any external effects of the surroundings. By multiplying all values by *dt*, we can obtain:

$$\mathbf{E= I_c^2R}\, dt + \mathbf{T\omega}\, dt$$

Where E is the energy, and *dt* is the length of one "step" of the acceleration curve. In this simplification, *dt* is equal to the total time, one half a second, divided by 2n, where n is the number of "steps" in the acceleration curve. The 2 is there since we are only dealing with the first half of the curve (this is because of the symmetry. The 60 degree total distance will be halved to 30 degrees as well).

Next, we have a few motor equations.

$$\mathbf{I_c=T / K}$$

Where K is a motor constant relating the input current to the output torque in Newton meters per amp. Substituting this back into the energy equation:

$$\mathbf{E=T /K}\, dt \,\mathbf{/ K + T\omega}\, dt$$

Next, we know that $\mathbf{T=I_g\alpha}$, where I_g is the moment of inertia (which we will assume to be an arbitrary value, .1, for this model), and α is the rotational acceleration of a "step." Finally, we must scale this torque by the gear ratio (66). So $\mathbf{T= I_g\alpha/G}$, where G is the gear ratio. Therefore, we have our final energy formula for the first model in terms of acceleration:

$$\mathbf{E= \Sigma((I_g\alpha)^2R}\, dt \,\mathbf{/ (G^2\, K^2) + I_g\alpha\omega}\, dt)$$

The Σ denotes that this is a summation for every "step." One other detail should be made. ω is the average rotational velocity for a "step." Since acceleration is constant for a "step," we can take this to be the "middle" value on a "step."

Constants:

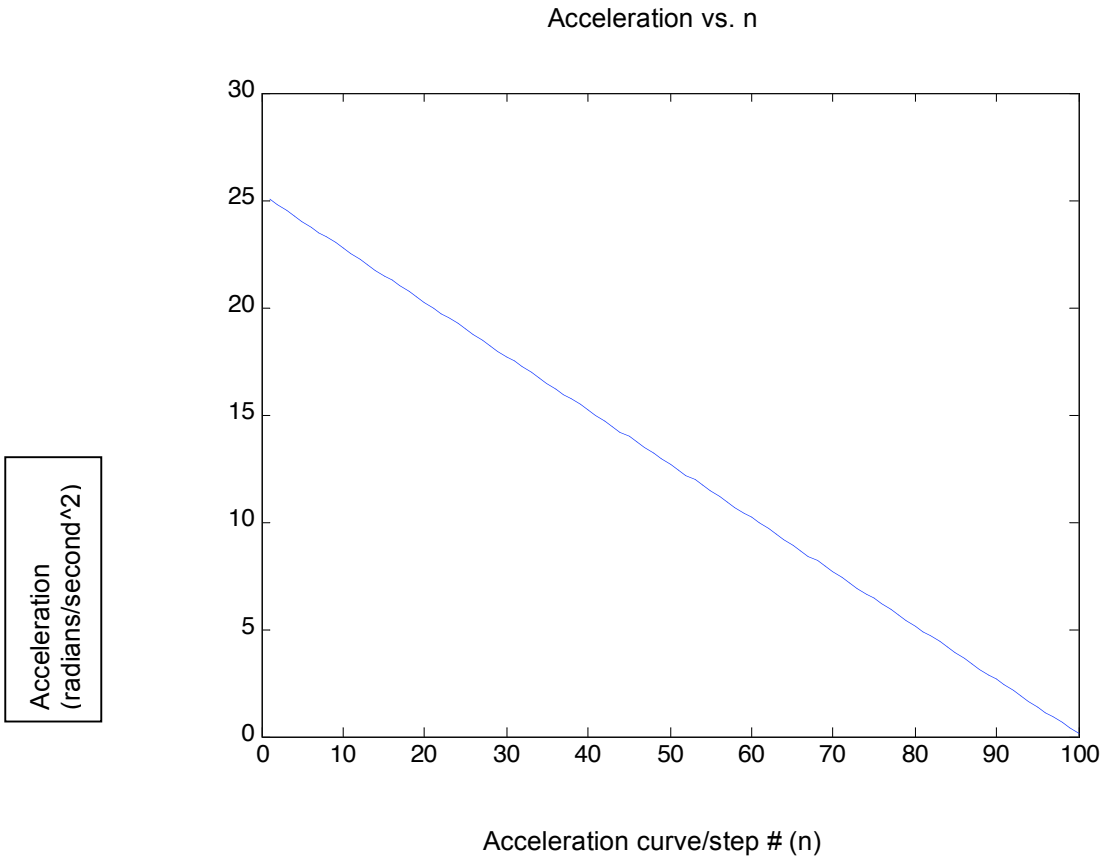
I_g = .1 (Kg m^2)
R=1.7 (ohms)
K=.01818
t, time for total rotation=.5 (seconds)
d, total distance of rotation = (π/3 radians)
c1, velocity dependent friction = .0155
c0, constant friction = .0729
Gear ratio = 66

R, K, c1, and c0 have been found experimentally.
I_g, t, and d are arbitrary values.

Simulations:

Since we can't assume anything about how the acceleration curve should look (besides the symmetry in this scenario), we're going to have to find the minimum energy curve through simulation alone. MATLAB's fminunc search will be most beneficial to us, since searching through every value otherwise would take far too long using a simulator like MATLAB (believe me, I tried). Let's assume for now that n will equal 100 "steps." Since we want the accelerations to add up to just enough to get the motor the distance it needs to go (30 degrees when assuming symmetry), we will allow the first 99 "steps" to be free values, and the last acceleration curve to be calculated so that the wheel will reach half the total distance just at the end of the first half of the symmetrical curve. To do this, we add the distance after each step by multiplying the acceleration by the time of the step squared. Then we take the required remaining distance, divide it by the time of the step to find the necessary average velocity, subtract that from the current velocity, and divide it by the time of the step again. This can be seen in the MATLAB code in appendix 1.

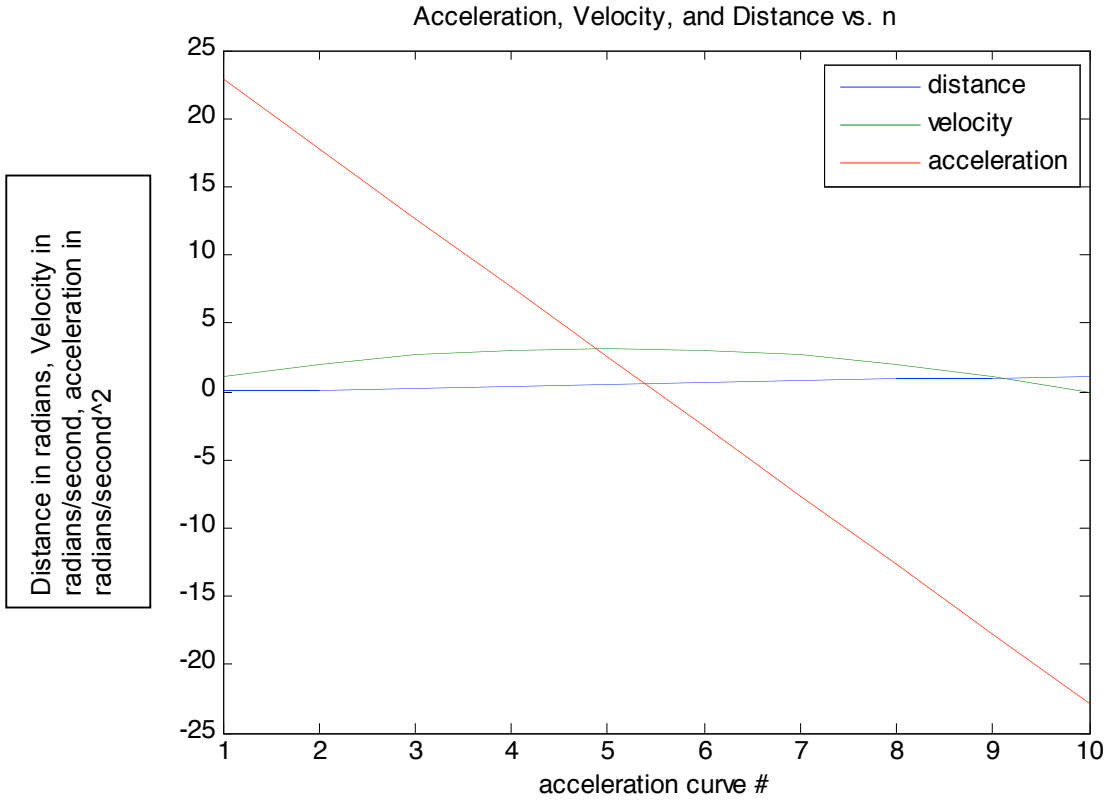
The result found for the first scenario, a non-generator, symmetrical acceleration curve with no friction, is shown below. As can be seen, for such a scenario, the most efficient method is to start with a high acceleration and linearly decrease it, and then mirror this with negative accelerations. Through extrapolation, there should be a slight vertical drop between the two halves, resulting in a non-smooth curve.



Simulation 1 – Assuming symmetry, no friction, and negligible generator properties

Energy for half = .6290
Total energy = 1.2581

For the second scenario, we remove the assumption of asymmetry. However, this requires a new method of calculating the ending velocity and acceleration. Instead of having n-1 fixed acceleration "steps," now there are n-2 fixed "steps." The code for calculating this is included in Appendix 2. This simulation yields similar results to the first, in that the acceleration curve is that of a constant linear deceleration, but since we are not assuming symmetry the linear deceleration continues throughout the entire 60 degrees of rotation, without a break.



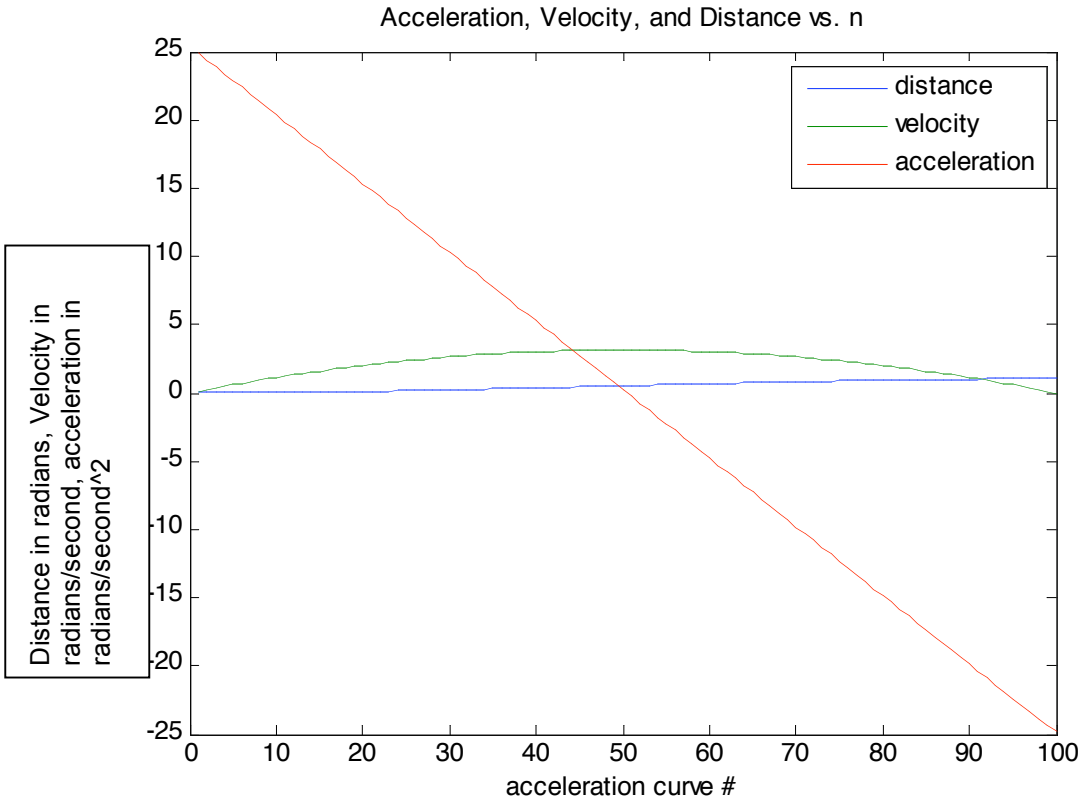
Simulation 2 – Assuming asymmetry, no friction, and generator properties

Energy=1.2556

For the third scenario, friction is being factored in, and any "negative" energy is then "added" to the total energy consumed, resulting in a lower energy than if we were assuming the motor not to be a generator. The truth is that the motor being used probably is a negligible generator, but neither the MATLAB function fminunc nor our current homebrewed gradient descent program is able to calculate an optimal path if we use an *if* statement to assign negative energies of "steps" to zero. Data and results are shown on the next page, code is in Appendix 3. The equation for torque with the inclusion of motor friction is:

$$\mathbf{T=I\alpha + c1\ \omega + c0}$$

Where c1 is a velocity related friction and c0 is a constant friction. This should be divided by the gear ratio and then substituted back into the energy equation for torque.



Simulation 3 – Assuming friction, asymmetry, and generator properties

Energy=1.3670

As expected, this costs more energy than the second simulation because of friction.

Discussion of results:

As expected, the total energy is greater with friction than without, and the minimum energy is less when we don't force a non-optimal symmetry. There are two differences in the energy calculation between models 1 and 2. First, for model 1, when you extrapolate the second half of the first graph (remember that this is only the first half of the symmetrical curve) there is a vertical drop between the first and second halves rather than a smooth line. Also, model 1 assumes that the motor is a negligible generator, whereas model 2 (and 3) assumes that it is a generator. One of the notable things about this final data is just how linearly consistent it actually is. In all scenarios, the difference in the rate of change does not fluctuate by more than .0002. The slope of the acceleration curve is approximately -.5028 over 100 "steps," denoting a total change of 50.28. Any irregularities such as a miniscule bump would probably smooth out if the number of

"steps" was to approach infinity and the length of each "step" was to approach zero. Additionally, there is a slight asymmetry to the results; the magnitude for the last acceleration curve is approximately 1 radian per second^2. There is a possibility that this asymmetry would disappear as the number of steps were to increase to a very high number, or perhaps this actually is more optimal than a balanced line.

A possible generalization we can make is that as long as that troublesome *if* statement of energy being less than zero does not exist, the optimal acceleration path for motor efficiency is a negative linear relationship. With the *if* statement, it is unknown if this generalization holds true.

Possible improvements:

The final simulation is probably the most important, but unfortunately, I'm yet to figure out a way to compute this. In the future, it would definitely be beneficial to find either a method of computing the optimal efficiency other than the gradient or line-search search, one that can compute with the problematic if "Energy<0" statement still as a part of the equation and finish the fourth simulation. Additionally, for a direct application this model will be somewhat inaccurate. That's because the model being used here is actually only for a wheel as opposed to an actual robotic leg. If there's a leg involved and not simply a wheel, the calculations become vastly more complicated as there is more to factor in, such as air resistance, different moments of inertia, etc. Inclusion of these variables would increase the accuracy of this experiment.

Conclusion:

In the end we did manage to find a minimum energy and a corresponding acceleration curve for rotating our hypothetical wheel 60 degrees in .5 seconds, one that was different than our original guess. When factoring in asymmetry and friction, we found that the minimum energy it would take for the rotation would be 1.3670 Joules, and that the most efficient way to do this is to start the wheel off at a high acceleration of about 24 radians per second^2 and then to lower it back down to approximately the same negative acceleration. This method of finding the optimal acceleration can be applied to any future necessary motor based projects. Additionally, perhaps the best information that we can get out of this is that using the above specification, it seems a negative linear path is most likely to be optimal for motor efficiency.

Acknowledgements:

Special thanks to Daniël Karssen for some of the motor formulas as well as guiding me through the process, Pranav Bhounsule for helping with MATLAB's methods of computing, Carlos Arango for his motor constants, and Andy Ruina and Jason Cortell for the opportunity to work in their lab.

Appendix 1: MATLAB code for first model:

```
function [Energy, a_last]=minEnergyFunction(n, a)
I=.1;
R=1.7;
K=.01818;
T=0.5;
dt=T/2/n;
d=pi/3;
gear=66;

vel_avg = 0;
vel_end = 0;
pos_end = 0;
Energy=0;

for k=1:n
    if k<n
        vel_avg = vel_end + a(k)/2*dt;
        vel_end = vel_end + a(k)*dt;
        pos_end = pos_end + vel_avg*dt;
    else
        vel_avg = (d/2 - pos_end)/dt;
        a_last = (vel_avg - vel_end)/dt*2;
        a(k) = a_last;
        pos_end = pos_end + vel_avg*dt;
    end
    Energy = Energy + (a(k)^2*I^2/K^2/gear^2*R*dt) +
(I*a(k)*vel_avg*dt/gear);
end
```

Appendix 2: MATLAB code for second model:

```
function [Energy, data]=minEnergyFunction2(n, a)
I=.1;
R=1.7;
K=.01818;
T=0.5;
dt=T/n;
d=pi/3;
gear=66;

vel_avg = zeros(n,1);
vel_end = zeros(n,1);
pos_end = zeros(n,1);
Energy=0;

for k=1:n
    if k==(n-1)
        a_nexttolast=-vel_end(k-1)*3/2/dt+(d-pos_end(k-1))/(dt^2);
        a(k)=a_nexttolast;
    elseif k==n
        a_last=-vel_end(k-1)/dt;
        a(k)=a_last;
    end
    if k>1
        vel_avg(k) = vel_end(k-1) + a(k)/2*dt;
        vel_end(k) = vel_end(k-1) + a(k)*dt;
        pos_end(k) = pos_end(k-1) + vel_avg(k)*dt;
    else
        vel_avg(k) = a(k)/2*dt;
        vel_end(k) = a(k)*dt;
        pos_end(k) = vel_avg(k)*dt;
    end
    Energy = Energy + a(k)^2*I^2/K^2*R/gear^2*dt +
I*a(k)*vel_avg(k)*dt/gear;
end

data = [pos_end, vel_end, a'];
```

Appendix 3: MATLAB code for models 3 and 4:

```
function [Energy, data]=minEnergyFunction3(n, a)
I=.1;
R=1.7;
K=.01818;
T=0.5;
dt=T/n;
d=pi/3;
c1=.0155;
c0=.0729;
gear=66;

vel_avg = zeros(n,1);
vel_end = zeros(n,1);
pos_end = zeros(n,1);
Energy=0;

for k=1:n
    if k==(n-1)
        a_nexttolast=-vel_end(k-1)*3/2/dt+(d-pos_end(k-1))/(dt^2);
        a(k)=a_nexttolast;
    elseif k==n
        a_last=-vel_end(k-1)/dt;
        a(k)=a_last;
    end
    if k>1
        vel_avg(k) = vel_end(k-1) + a(k)/2*dt;
        vel_end(k) = vel_end(k-1) + a(k)*dt;
        pos_end(k) = pos_end(k-1) + vel_avg(k)*dt;
    else
        vel_avg(k) = a(k)/2*dt;
        vel_end(k) = a(k)*dt;
        pos_end(k) = vel_avg(k)*dt;
    end


    T=(a(k)*I+c1*vel_avg(k)+c0)/gear;
    Add_Energy(k)=T^2/K^2*R*dt + T*vel_avg(k)*dt*gear;
%     if Add_Energy(k)<0
%         Add_Energy(k)=0;    <- - - - - The Energy<0 problem.
%     end         If including these lines, the code can't be optimized

    Energy = Energy+Add_Energy(k);
end

data = [pos_end, vel_end, a'];
```