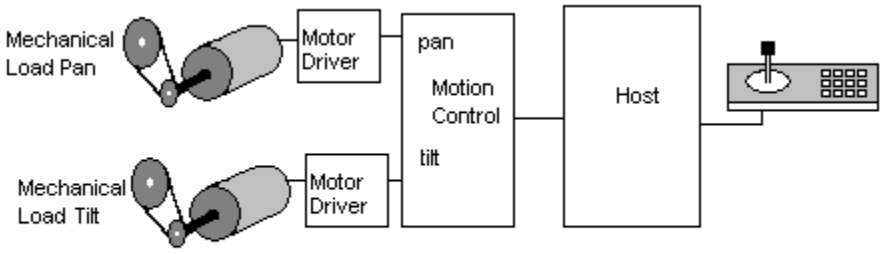
Dan DaSilva
January 22, 2004

Pelco Motion Control

This paper describes how motion control is implemented at Pelco and how to improve it.



Input from keyboard and joy stick

Keyboards with joysticks are the front end of motion control. This is where the user inputs the position commands to the system. The output is a value from 0 to 64 where 0=stopped, 63=fastest speed, and 64 = turbo speed (Turbo speed is pan only, no turbo allowed in tilt). This format is dictated by the communication protocols which are ASCII, P, D, or Coaxitron. All protocols have this format and therefore all motion control devices must adhere to it. This means that no matter how many speeds are possible by the motor and controller, only 65 speeds are available.

At the front end controller the value of the joystick potentiometer voltage is read by an A/D. The output is linear with 0 volts = speed zero, while ½ scale voltage = speed 31 and full scale voltage = speed 63 . Turbo speed (speed 64) is activated by a separate keyboard button.

Set preset:
The set preset command informs the position device (Spectra or Esprit for example) to store the current pan and tilt positions in non-volatile memory and assign it the preset number commanded by the keyboard.

Go to preset:
Keyboard sends preset number to positioning device which commands motion control to the position stored in memory for that preset number.

System Preset accuracy:
Pelco has advertised preset accuracy of 0.1 degree on Spectra III. With a 0.9 degree step motor and a 5 to 1 gear ratio each step is 0.18 degree. Therefore the accuracy required can be met if the motor is within ½ full step of the target. Other inaccuracies are listed in the table below.

Sources of Preset Inaccuracy

Mechanical	**Electrical /Software**
Dimension Tolerance in Belt	Speed of Home Detection in Software
Friction	Accuracy in LED Electrical Trip Point
Imbalanced load of camera	Micro Switch Repeatability
External Force, Bird or Wind	Motor Magnetic Hysteresis
Lack of Mechanical Stiffness	Lack of Motor Stiffness

Microstep accuracy level on the order of 1/64 of a step are not possible because the motor does not have the stiffness to hold the target to that tight a tolerance. If ½ step accuracy is required the motor does quite well in holding to that accuracy.

Patterns:
At the start of a pattern the host starts recording the keyboard commands. These commands are recorded serially in time and are played back at the same rate at which they were recorded. On a host with a PAL camera the resolution of recording is 50 times per second on a NTSC it is 60 times per second. One concern with patterns has been accuracy. The position accuracy suffers from the fact that the play back does not contain position information (It is not due to the factors listed above in Preset accuracy.) . In other words, pattern accuracy is based on dead reckoning. A possible improvement would be to record the absolute position a points in the pattern and then on playback insure that these points are repeated.

Motors used for Motion

2-phase stepper motors:
This type motor has been used for several years for all motion control at Pelco. These motors are used without feedback, only the home position has

to be found at power-up. If higher performance is someday required feedback can be use with this type of motor.

Others types of motor that have been used:
AC induction motors:
Provide only a single speed and are used in older designs. Required feedback in the form of a potentiometer to determine position. Single speed motors are no longer competitive.

DC brush motors:
These motor provide multiple speeds. Feedback is required in the form of a potentiometer to determine position. Brushes made this type of motor unreliable compared to brushless motors.

5-phase stepper motors:
Intercept uses these motors. They have an advantage over 2-phase motors in full and half stepping smoothness because the steps are smaller. This advantage disappears with microstepping. The inherent disadvantage is complexity in the driver which is require to drive five phases instead of two. Industry has standardized on the two phase stepper and the five phase is only produced by the Oriental Motor Company. All new designs of motion control at Pelco have used two phase motors.

Servo motors have not been used at Pelco. The use of servo motors is possible for new designs because they can provide higher acceleration, speed and accuracy than stepper motors but with a higher price. Servo motors usually require gear train reduction while stepper motor have more torque for a given size. Therefore steppers can be operated with less reduction such as a simple belt reduction drive which tends to be an advantage for steppers. There a not been any detail cost / performance studies on servo designs recently.

Host Processor

At power-up the host processor directs the motion controller to find home positions in both pan and tilt. Pan is commanded to travel more than one full revolution or until it passes the home position tab. The pan is commanded to return in the opposite direction to determine the tab width. By using the

width it is possible to detect the home position while traveling in either direction. The tilt is commanded up for a period of time long enough to insure that it has hit the tilt limit, this is the tilt home position.

The host processor also performs the following duties:
Saves preset values by position gotten form motion controller

Gives velocity commands based on input from keyboard

Converts the 0-63 velocity command into speed command by using a table for example in Spectra the table for pan is:

```
pan_speed[] =
{
   5,    5,    5,    5,    5,    5,    5,    5,
   5,    5,    6,    7,    7,    8,    9,   10,
  10,   11,   13,   14,   15,   17,   18,   20,
  22,   24,   26,   29,   32,   35,   38,   42,
  46,   50,   55,   60,   66,   73,   80,   87,
  96,  105,  115,  126,  139,  152,  167,  183,
 200,  220,  241,  264,  290,  318,  349,  382,
 419,  460,  504,  553,  607,  665,  729,  800
};
```

This array is accessed by the speed command from the keyboard. It has 64 values which are indexed by the 0-63 range of speed commands. Each value in table represents 1/10 degree per second at the camera therefore speed command of 0 gives a speed of 0.5 degrees second speed command of 63 gives a speed of 80 degrees second

If proportional pan speed is on the speed is reduced by the zoom of the camera. for example if zoom is 4 and speed command is 20 degrees per second then the proportional pan speed is computed as 20/4 = 5 and the actual camera speed will be 5 degrees per second.

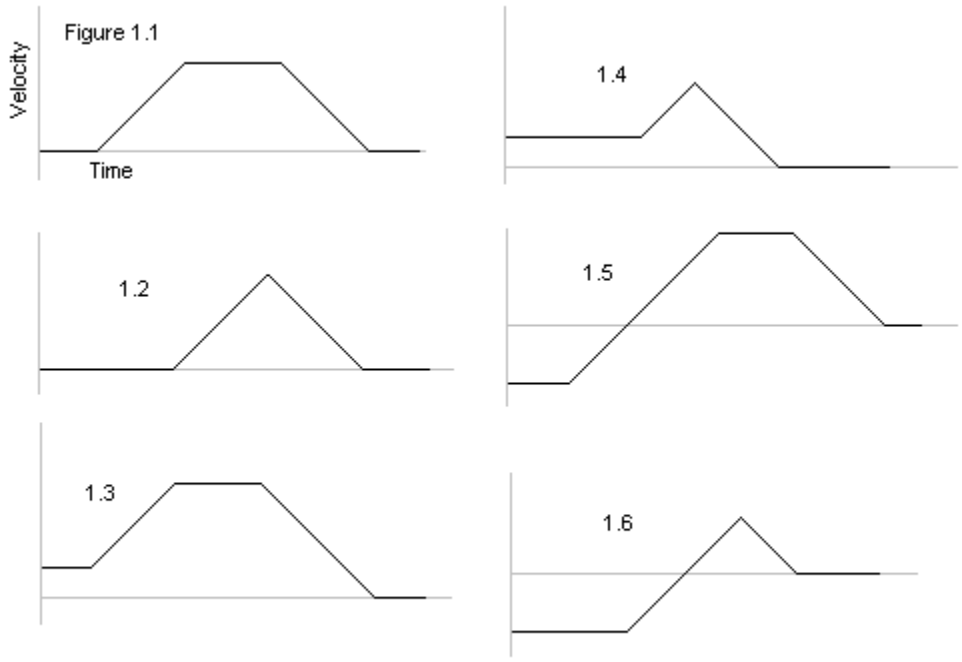
The jitter speeds table is specified as a range of speeds which are not allowed, for example if camera motion has jitter in the range of 3.1 to 4.3 degrees per second then according to the pan table the speed 3.2, 3.5, 3.8 and 4.2 will be skipped. The table combined with jitter speeds can produce a

very non-linear command to actual speed relationship. The jitter skipping is not desirable but is necessary. With better control of motor, jitter skipping would not be required.

The host processor also sets acceleration value, computes gear ratio and step size to compute values for deg/sec.

Motion Controller

The motion controller performs all calculations to execute the motion moves commanded by the Host Processor. The hardest and most calculation intensive task is the preset move. The following table show the 6 possible type of preset moves that must be taken into account:



The X axis is time, the Y axis is velocity in all Figures 1.1 – 1.6.

Figure 1.1
Move from stop to a preset ending in a stop. Max velocity is reached and held for some time.

Figure 1.2
Move from stop to a preset ending in a stop. Max velocity is never reached.

Figure 1.3
Move with a starting velocity to a preset ending in a stop. Initial velocity was in the correct direction to make the move. Max velocity is reached and held.

Figure 1.4
Move with a starting velocity to a preset ending in a stop. Initial velocity had to be reversed to make the move. Max speed is never reached.

Figure 1.5
Move with a starting velocity to a preset ending in a stop. Initial velocity had to be reversed to make the move. Max speed is reached and held for some time.

Figure 1.6
Move with a starting velocity to a preset ending in a stop. Initial velocity had to be reversed to make the move. Max speed is never reached.

Computing a preset and performing the move:
If started from stop this is a classical trapezoid move with constant acceleration. If move in the opposite direction first stop and the do the trapezoid move. PUD: If moving in the correct direction accelerate to first stop and the do the trapezoid move. PMD: Does a trapezoid move from a running start

The goal of the motion control system should be to produce constant velocity during constant speed moves and constant acceleration for during speed changes. This will result in smooth motion which is what is desired by the customer when he looks at the camera image while moving.

If the controller does not have the ability to change velocity after each microstep then it is possible that the target maybe undershot or overshot. The desired procedure is to always undershoot the target then to make a small adjustment at the end to reach the target.

The derivative of acceleration is jerk, in past designs acceleration has been kept constant but jerk has been allow to be infinite. The advantages of limiting jerk have not been observed during testing and all Pelco designs so far do not limited the amount of jerk.

Motor Driver

The Motors used by Pelco are Bipolar Hybrid two phase. In Bipolar motors the current direction is switched to change the direction of the motor. To reverse the direction of the field produced by a motor winding, we need to reverse the current through the winding. We could use a double-pole double throw switch to do this electromechanically; the electronic equivalent of such a switch is called an H-bridge and is shown in Figure 2.1:

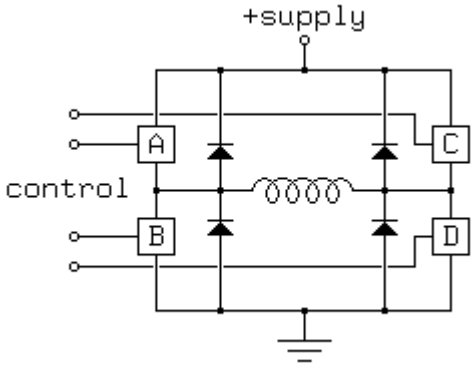


Figure 2.1

The switches used in the H-bridge must be protected from the voltage spikes caused by turning the power off in a motor winding. This is usually done with diodes, as shown in Figure 2.1. Note: Mosfets have the advantage of containing intrinsic diodes.

With 4 switches, the basic H-bridge offers 16 possible operating modes, 7 of which short out the power supply. The following operating modes are of interest:

Forward mode, switches A and D closed.
Reverse mode, switches B and C closed.

These are the usual operating modes, allowing current to flow from the supply, through the motor winding to ground. Figure 2 illustrates forward mode:

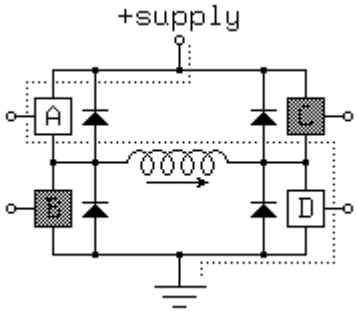


Figure 2.2

Fast decay (Four Quadrant Control) mode:

Any current flowing through the motor winding will be working against the full supply voltage, plus two diode drops, so current will decay quickly. Figure 2.3 illustrates the current flow immediately after switching from forward running mode to fast decay mode.

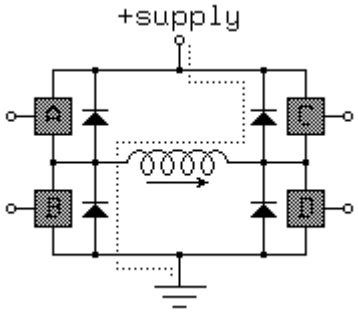


Figure 2.3

Slow decay (Two Quadrant Control) modes:

In these modes, current may circulate through the motor winding with minimum resistance. As a result, if current is flowing in a motor winding when one of these modes is entered, the current will decay slowly. Figure 2.4 illustrates one of the many useful slow-decay modes, with switch D closed; if the motor winding has recently been in forward running mode, the state of switch B may be either open or closed:

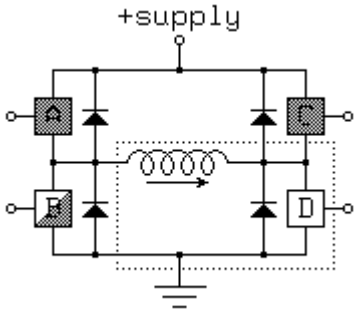


Figure 2.4

Most H-bridges are designed so that the logic necessary to prevent a short circuit is included at a very low level in the design. Figure 5 illustrates one arrangement:

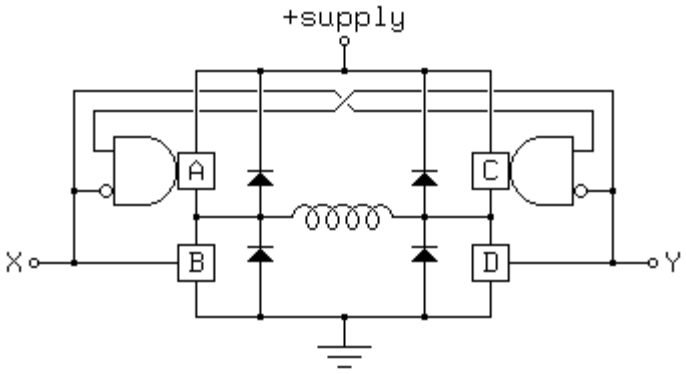
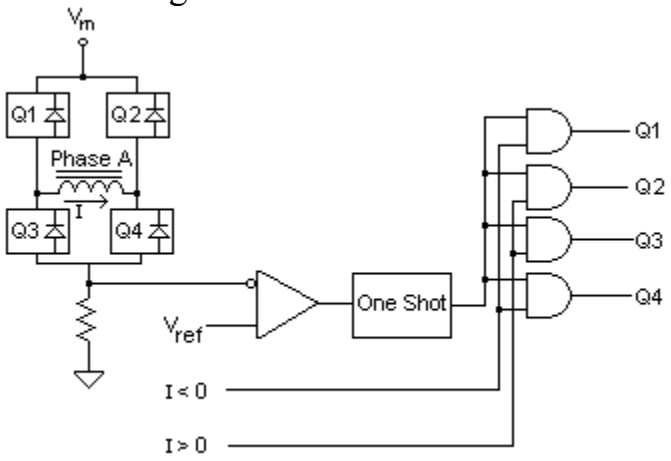


Figure 2.5

Peak Current Control:
Until recently all Pelco designs have used peak control to control current in the motor windings.



As shown in the figure above a comparator monitors the amplitude of the winding current by having as one of its inputs the voltage across the sensing resistor. The output of the comparator triggers a one-shot which implements the constant off time. The one-shot when triggered turns all switches in the bridge driver off. As a result, the winding current can only flow back into the power supply, it uses fast decay.

Improvements

Less vibration at the motor will produce less jitter at the camera. Excluding external sources, the motor is the only source of vibration in the system. The camera support and drive mechanics can be designed to minimize vibration but producing smoother motion solves the problem at its origin. In the past very little work has been done in trying to characterize the motor so that the best drive waveforms can be fed to it. Initial testing indicates that 95% of the velocity modulation or vibration can be eliminated by correcting current waveforms which drive the motor. In the past it has been assumed that sine waves should be used. This is definitely not the case. To calculate the correct waveform the motor is simulated by torque equations.

At Pelco many designs have been used which operate step motors at less than their rated current and therefore less than rated torque. This has increased the effect of detent torque which does not change with current. Because the effect of detent torque was never modeled the result is velocity modulation. The use of more and more powerful zoom on cameras has caused picture jitter to reach objectionable levels. The solution is to model detent and all other significant factors when designing a motion control system.

Torque Equation Model:

$$\mathbf{dT_a/d\emptyset = -i_a*N*(B - C/2*|i_a|)*N*(\cos N\emptyset + 3h_3*\cos 3N\emptyset)}$$
$$\mathbf{dT_b/d\emptyset = -i_b*N*(B - C/2*|i_b|)*N*(\sin N\emptyset - 3h_3*\sin 3N\emptyset)}$$
$$\mathbf{dT_d/d\emptyset = -4D*N*\cos 4N\emptyset}$$
$$\mathbf{dT_{Total} = dT_a/d\emptyset + dT_b/d\emptyset + dT_d/d\emptyset}$$

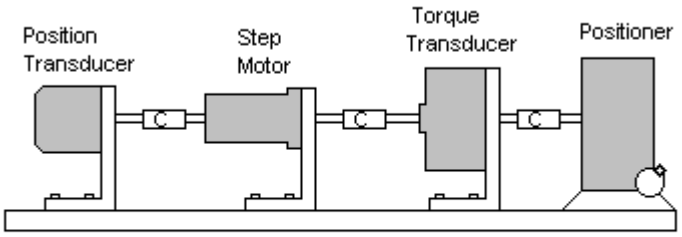
T = Torque
i = current

N = number of steps per revolution
B = Magnetic Flux linkage
D = Detent Torque Constant
C = Flux leakage Constant
h_3= 3rd harmonic content
Ø = Position angle of rotor to stator coil
a,b = Phase A and Phase B

By determining the values of D C and h_3 the motor can be accurately characterized for the purposes of model dynamic behavior and micro step current requirements.

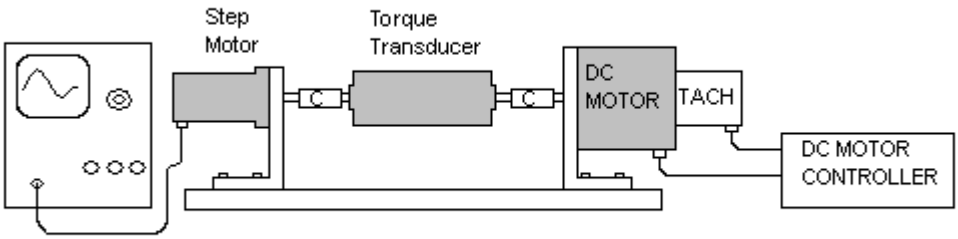
Determining the Motor Constants:
C = Saturation or Flux leakage Constant
D = Detent Torque Constant
h_3= 3rd harmonic content
B = Magnetic Flux linkage

The Step Motor System Design Handbook by Albert C. Leenhouts
has a program which computes motor constants given parameters which can be measured by the following procedures:



Torque Test Procedure:
Using test setup shown above with other instruments determine the following parameters:
Rotor Inertia
Winding resistance
Rated Power Dissipation
Detent Torque at various points
Holding torque at various points

Spin tests:



Note that the Torque transducer used in the above test must be dynamic as opposed to the transducer in the previous test that uses a static torque transducer.

Spin Test Procedure:
Spin tests measure the Back emf of the motor. Note Back emf and torque are intrinsically related. If there is no magnetic saturation then the Back EMF Constant K_E in units of Volts / (Radian / Second) is equal to the Torque Constant K_T in units of (Newton * Meter) / Ampere. This is a simple change in units:

K_E = 1 Volt / (Radian / Second)
K_T = 1 (Newton * Meter) / Ampere

1 Volt = 1 Joule / Coulomb
1 Coulomb = 1 Amp * Second
1 Joule = 1 Newton * Meter * Radian

Substituting and rearranging where necessary:
K_E = 1 Volt / (Radian / Second)
Substitution: 1 Volt = 1 Joule / Coulomb

K_E = 1 (Joule / Coulomb) / (Radian / Second)
Substitution: 1 Joule = 1 (Newton * Meter) * Radian

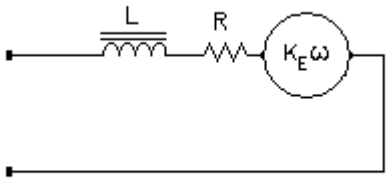
K_E = 1 (Newton * Meter * Radian / Coulomb) / (Radian / Second)

Cancel terms

K_E= 1 (Newton * Meter / (Coulomb / Second))
Substitution: (Coulomb / Second) = Ampere

K_E = 1 (Newton * Meter) / Ampere
therefore: $K_E = K_T$

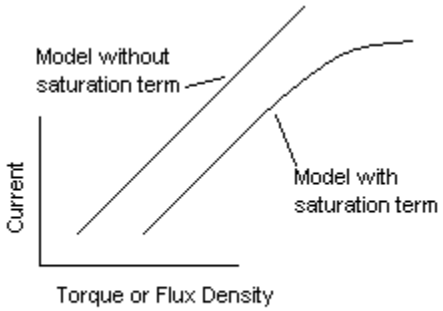
Using K_E, the electric model for one phase becomes:



Where K_E is the Back Emf constant and ω is the speed.

To measure back emf the step motor is driven by another motor, and we observe the voltages that appear at the winding terminals, or the current that flows in a short circuit winding. The test results are used to calculate losses, and the large signal inductance.

Saturation and Flux Leakage:

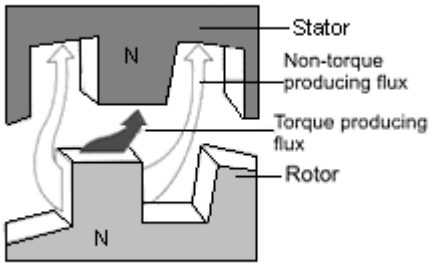


As the flux density in iron increases it reaches a point of saturation and for increases in flux density beyond that point the iron starts to behave like free space. To model this effect a saturation flux leakage term (**C**) is produced

whose effect varies as the square of the current. At low values of current the effect of the saturation term is near zero. However at higher currents it starts to subtract from the torque of the system. The equation is shown below (without the 3rd term)

$dT_a/d\emptyset = -i_a*N*(B - C/2*|i_a|)*N*(\cos N\emptyset)$

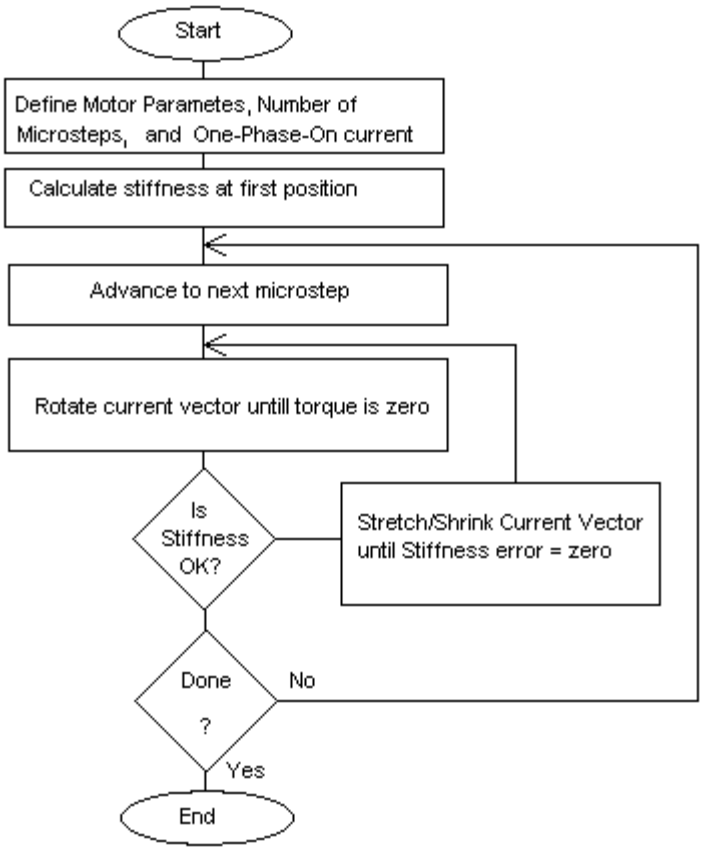
As saturation takes place more and more flux leakage occurs:



The net effect of flux leakage and saturation is to reduce torque as shown in the above torque equation

Using the book The Art and Practice Of Step Motor Control by Albert C. Leenhouts, these values are entered into a program and the Motor Constants are determined. The actual calculations used to obtain these constants are not explained and a executable program without source does the calculation. With some effort it should be possible to determine the calculation method and gain a understanding of the process.

Determining Micro Step Table:



Source Code For determining Microstep Table:

```
// PROGRAM CURRENTS
// CALCULATES CURRENT VALUES FOR MICROSTEPPING A TWO PHASE STEP MOTOR
//
// CONSTANTS ************************************************
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
void torque();
void current();
void stiffness();
void calculate_current();
const double  PI = 3.1415927;

//MOTOR PARAMETERS *****************************************
//
double N  =  50.000;   // number of poles
double NB =   0.350;   // flux

// sine
```

```
double D  =   0.020;
double NC =   0.0;   // leakage flux
double H3 =   0.0;   // 3rd harmonic

//
// USER SELECTED PARAMETERS ***********************************
//
const double I =   1.0;   // one phase on current

#define STEPS  256
const double Z =  STEPS;
int i;

double I1,I2,Q,K1,K2,S,S1,S2,
S3,S7,S8,SS,T,T1,
T2,T3,T7,T8;
////////////////////////////////////////////////////////////
int main()
{
      printf(" THIS IS PROGRAM CURRENTS \n\n");
      printf(" MOTOR PARAMETERS: \n");
      printf(" NB = %f N.M./A.",NB);
      printf("      NC = %f N.M./A.^2 \n",NC);
      printf("  D = %f N.M.",D);
      printf("         H3 = %f percent", H3*100);
      printf("\n\n");

      printf(" 1 PHASE ON CURRENT = %1.2f A",I);
      printf("\n NUMBER OF MICROSTEP/CYCLE = %2.0f",Z);
      printf("\n\n");

// STIFFNESS CALCULATION **************************************
    Q  = 0.0;
    K1 = I;
    K2 = Q;

    current();
    stiffness();
    // compute the starting stiffness to be used as the standard
    SS = S;

// MAIN PROGRAM ***********************************
    printf(" STEP #    CURRENT 1          CURRENT 2\n");

    i=0;
    Q=0;
    for (;;)
    {
        calculate_current();
        printf(" %2.0f         %2.4f             %2.4f \n",
               ( (Q/(2*PI) ) *Z), I1, I2 );
        i++;
        Q += (2*PI)/Z;
        if (i>= Z)
            break;
    }
```

```
    return 0;
}

//////////////////////////////////////////
void calculate_current()
{
    K2 = Q;
    // while stiffness error is to large
    for (;;)
    {
        for (;;)
        {
            current();
            torque();
            if ( fabs(T) < (0.001 * NB * I) )
                break;
            T7 = T;
            K2 = K2 + 0.0001;
            current();
            torque();
            T8 = T;
            K2 = K2 - 0.0001;
            K2 = K2 + 0.0001 * (T7/(T7-T8));
        }
        stiffness();

        if ( fabs(S - SS) < (-0.001*SS) )
            break;
        S7 = S;
        K1 = K1 + 0.0001;
        current();
        stiffness();
        S8 = S;
        K1 = K1 - 0.0001 + 0.0001*( (S7 -SS) / (S7 -S8) );
    }
    return;
}

//////////////////////////////////////////
void torque()
{
    T1 = -( NB*I1 -(NC/2) * I1 * fabs(I1) )  * (sin(Q) + H3*sin(3*Q));
    T2 =  ( NB*I2 -(NC/2) * I2 * fabs(I2) )  * (cos(Q) - H3*cos(3*Q));
    T3 = -D*sin(4*Q);

    T  = T1 + T2 + T3;

    return;
}
//////////////////////////////////////////
void current()
{
// CALCULATE CURRENTS ****************************************
    I1 = K1 * cos(K2);
    I2 = K1 * sin(K2);
    return;
```

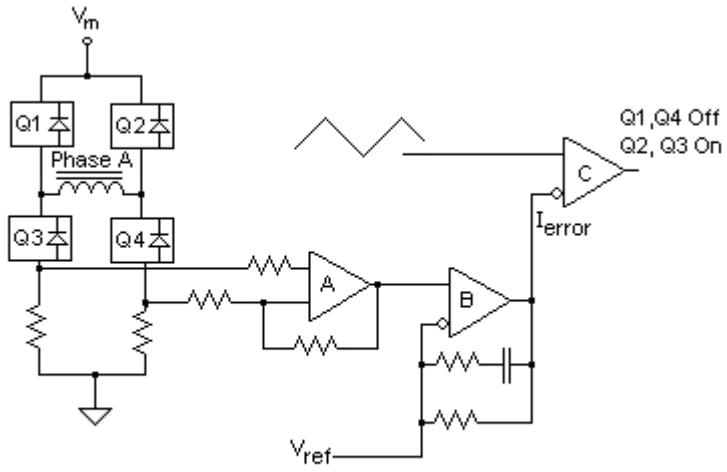
```
}
//////////////////////////////////////////
void stiffness()
{

// CALCULATE STIFFNESS ***************************************
// H3 is the third harmonic term,  NC is the leakage flux term
    S1= (cos(Q) + 3*H3 * cos(3*Q)) * (NB*I1-(NC/2) * I1 * fabs(I1));
    S2= (sin(Q) - 3*H3 * sin(3*Q)) * (NB*I2-(NC/2) * I2 * fabs(I2));

    // detent torque
    S3= 4 * D * cos(4 * Q);

    // compute total stiffness
    S= (-S1-S2-S3) * N;
    return;
}
```

P.W.M Average Current Control Driver:
With this type of drive the effects of non-linearity around zero current level are reduced to an insignificant level. Previously all Pelco design have used peak current control. Project 125 is using a P.W.M Average Current Control Driver.



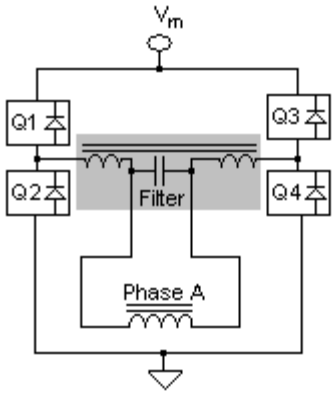
In Pulse Width Modulated drivers the average, current rather than the instantaneous, winding current is monitored. Then, a closed loop controller generates the On/Off commands to the driver circuit based on the difference between the observed and the desired current. The figure above shows the sensing and control circuit.

In this circuit there are two current sensing resistors. At any point in time one monitors the current coming from the power supply, the other monitors the current being returned to it. Both of these currents flow through the phase winding. The operational Amplifier A produces a fairly accurate replica of the winding current. This output is averaged and compared with the reference voltage which reflects the desired current level by Amplifier B. The error signal I_{error} is compared with a saw tooth voltage and generates the On/Off signals to the switch in the bridge. With proper averaging the effects of eddy currents and non-linearity around zero current level can be reduced to an insignificant level.

All switching circuits, but especially Four Quadrant Choppers (fast decay) and the P.W.M. drivers cause significant eddy current losses in the motor. Two Quadrant (slow decay) choppers have less loss, because the voltage applied to the winding is for some time near zero. Eddy currents in the magnetic components of the motor, including the steel, but, in some cases also the magnet, cause heating of the motor. Ultimately, the torque and power output of the motor are limited by its heat dissipation. Smarter circuits reduce the dissipation caused by eddy currents by combining the characteristics of the Two Quadrant and Four Quadrant Chopper into hybrid forms that change mode as necessary to minimize heat in the motor.

Note: An implementation of the PWM driver is described in the data sheet for the L6258 form ST Microelectronics.

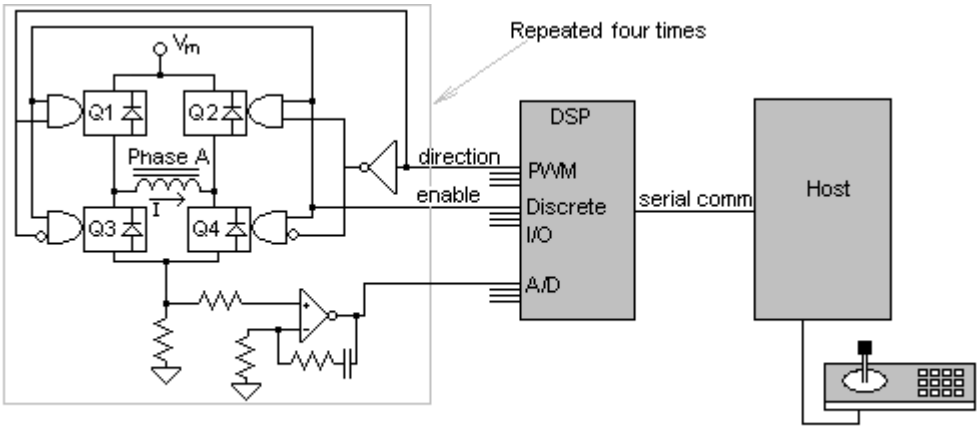
In P.W.M. Drivers we also have the option of placing a high frequency filter between the motor and the driver. A general circuit arrangement is shown in figure below. This filter circuit has the added advantage of containing high frequency electro-magnetic interference signals in the driver, rather than having them wandering all over the system along the motor cables. This type of driver lowers the eddy current losses in the motor to minimum, and lets us obtain the maximum torque from the motor for a give heat dissipation level.



New Pelco Motion Controller/Driver

Hardware:

The P.W.M Average Current Control Driver circuit can be modified so that the current feedback is read by an A/D converter and the control is done by PWM on microcontroller or DSP. Possible system configuration shown below:



Commands:

Set Axis:

Set the axis (pan or tilt) that the commands will refer to.

Set Acceleration:

16 bit unsigned value of the acceleration used for all moves.

Set Velocity:

32 signed value of the commanded velocity.

Set Preset Velocity:
Set the maximum velocity for preset moves.

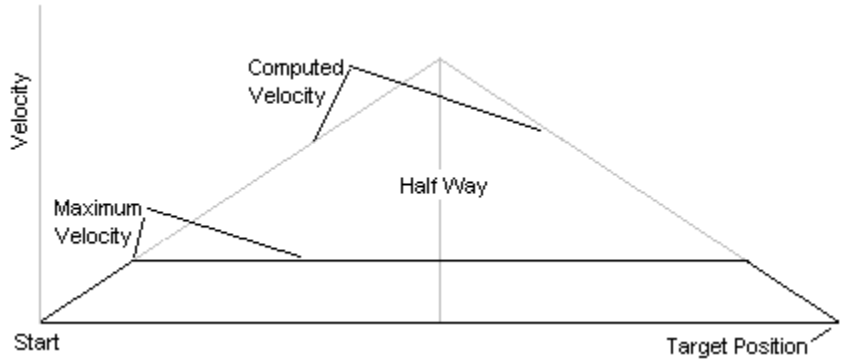
Goto Preset Position
Move to the 32 bit signed value of position gotten from host.

Report position
Return the 32 bit signed value of the current to the host.

Download Table
Load from host the microstep table

Acquire Home Position.

Algorithm for preset moves



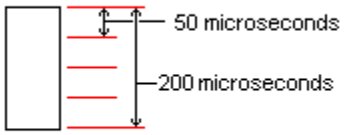
When performing preset moves first come to a stop if moving, then compute the total distance for the move. Start accelerating to the target position. If maximum velocity is reached then limit the actual velocity to this value but also keep computing the calculated velocity as if it was not limited. When half way is reached start reducing the computed velocity . As the computed velocity is reduced it will eventually match the maximum actual velocity at that time the actual velocity is reduced until it reaches zero.

Current control:
This design proposal is based on the PWM capabilities of the Motorola DSP56F802. The PWM will run at 40KHz with a 11 bit pwm the frequency of clock must be 40MHz

Fastest Speed Operation:

Diagram of fastest speed, one full step shown

Diagram of fastest speed operation at 200 microseconds or 5k full step/second



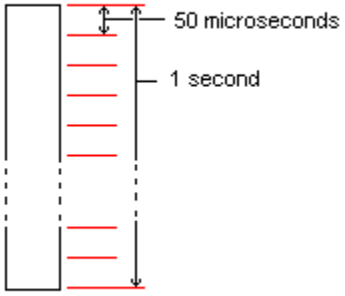
Using a 40 MHz clock and 11 bits of PWM resolution the frequency is 40MHz/2048 = 20KHz. The period of the PWM is therefore 50 microseconds

The PWM will update the four times at the fastest step speed. At this speed the controller will be full stepping.

Slowest Speed Operation:

Diagram of one microstep at slow speed

Diagram of slow speed operation at 1 microstep/second



Using a 40 MHz clock and 11 bits of PWM resolution the frequency is 40MHz/2048 = 20KHz. The period of the PWM is therefore 50 microseconds

Therefore at slow speed of 1 microstep per second each microstep will have 20,000 PWM cycles

Speed range: From 1 microstep/second (1/64 step/second) to 5000 step/second gives a range of 320,000 to 1 which will require a 32 word.

Software Control Loops
Fast current control loop:
Load the desired current level determined by the 200 microsecond loop. Calculated the PWM value to the desired value and the feedback value using a proportional, integral, derivative (PID) or other algorithm.

Slow motion control loop:
Using the acceleration values and velocity values find the desired current value using the waveform table.

PWM Resolution:
At very slow speed the PWM will lose some resolution because it will be operating at only at fraction of its full range. For example, if the supply voltage is 32 volts and the motor has 4 ohms and operates at 1 ampere then the PWM must cut the supply voltage back to an effective value of 4 volts.

This means that the PWM will be operating at 4/32 = 1/8 of full scale. Therefore if the PWM has a resolution of 11 bits, at slow speeds or DC its resolution will be only 8 bits ($2^{11}/ 8 = 2^8$). To add resolution the PWM could alternate between two values. For example to get 9 bits of resolution with a value of 250.5 a 8 bit PWM could alternate between 250 and 251. This would be in place of just outputting a constant values of 250 which would be to low or 251 which would be too high.

Processor:
To perform the motion control task the following processor requirements are needed: $5 or under, 16 Bit or more, capable of 40Mips or more, 4 ADC, 4 PWM, 4k Flash, Serial communication. Two controllers were found that meet his requirement:

Motorola DSP56F802 $4.70 in volume of 1000.
40 MIPS 16 Bit DSP
6 channels PWM
Two 4-Channel 12-bit ADC's
Serial Communication Interface
8K x 16 bit Program Flash
1K x 16 bit Program RAM

Texas Instruments TMS320LF2401A $5.20 in volume of 1000.
40 MIPS 16 Bit DSP
7 channels PWM
4 channels of 10 bit ADC
8K x 16 bit Program Flash
1K x 16 bit Program RAM

Models

Model of Motor Load

A good motion control design begins by describing the load that must be moved , and the motion trajectory that this load should follow. This is rotary version of F = MA with friction and gravity added.

$$T = (J_L + J_M)\ \alpha + T_V\omega + T_C\ (\omega / |\omega|) + T_G$$

α is the motor acceleration in radians / second^2
ω is the velocity of the motor in radians/ second
T_M torque of the motor
J_L is the moment of inertia of the load
T_V is the viscous friction coefficient in (Newton Meter) / (radian / second)
T_C is the Coulomb friction in Newton Meters
T_G is the Gravitational torque in Newton Meters

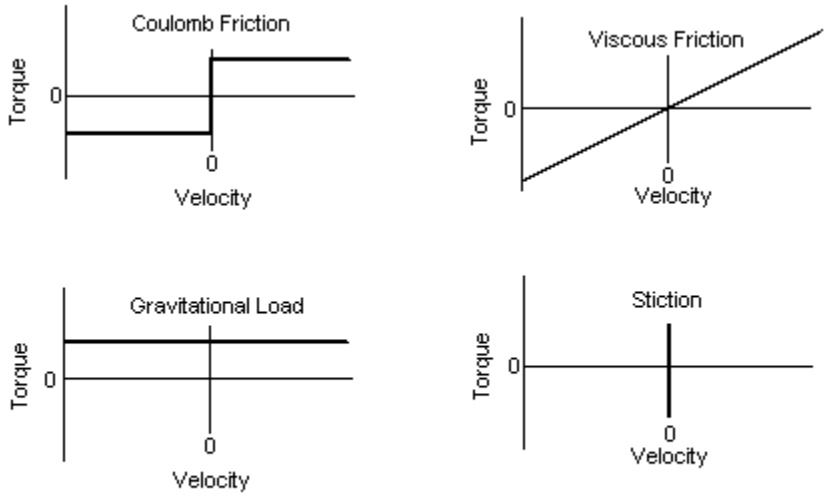
solving for
$\alpha = T_M - T_V - T_C (\omega / |\omega|) - T_G / (J_L + J_M)$

Viscous friction opposes motion, and is proportional to velocity

Coulomb friction also opposes motion, but has an amplitude that is independent of velocity. The sign of this friction changes abruptly with the direction of velocity. This is what the (ω / |ω|) is doing.

Gravitational Load Torque is due to gravity

Stiction is the term used to describe the phenomenon where a high is need to imitate motion. Once the load is moving stiction is zero.

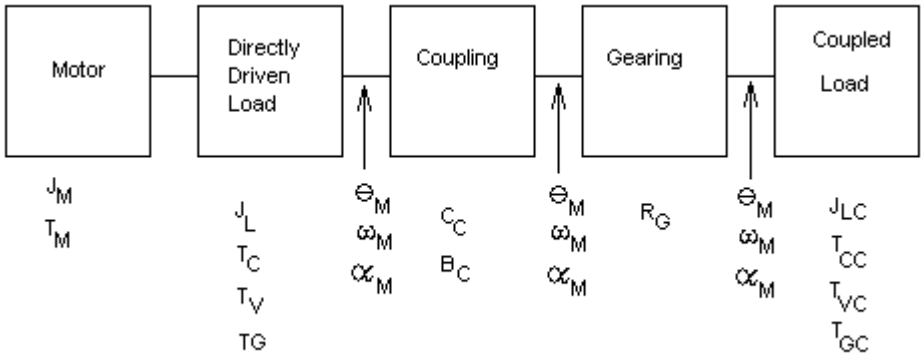


Gearing and Compliant Couplings

If the load is driven through a compliant coupling and or a gearing mechanism, we must consider additional load parameters. Now, the load consists of the directly coupled load elements described above, coupling compliance and damping , a gear ratio and a second set of load parameters, similar to those used to describe the first set.

If we apply a torque across a compliant coupling a position difference between the input and output results. This deflection is proportional with the coupling compliance C_C and the applied torque. Neoprene or polyurethane show substantial damping torque proportional to the difference in velocity between the two ends of the coupling and the damping coefficient B_C .

The gear ratio R_G is defined by diving the speed of the motor by the speed of the coupled load. The efficiency of the gearing is best described by losses described by the parameters describing the directly driven load and the coupled load.



θ_M is the position of the motor and the directly coupled load in Radians

ω_M is the velocity of the motor and the directly coupled load in Radians/Second

α_M is the acceleration of the motor and the directly coupled load in Radians/Second^2

C_C is the coupling compliance in
Radians / Newton Meter

B_C is the damping coefficient of the coupling in
Newton Meters / (Radians / Second)

θ_C is the position at the output shaft of coupling in
Radians

ω_C is the velocity at the output shaft of coupling in
Radians / Second

α_C is the acceleration at the output shaft of coupling in
Radians / Second^2

R_G is the ratio between the input and the output of the gearing unit

$\theta_{L\,C}$ is the position of the coupled load in
Radians

ω_{LC} is the velocity of the coupled load in
Radians / Second

α_{LC} is the acceleration of the coupled load in
Radians / Second^2

J_{LC} is the moment of inertia of the coupled load in
Kilogram / Meter^2

T_{CC} is the Coulomb friction at the coupled load in
Newton Meters

T_{VC} is the vicious friction coefficient of the coupled load in

(Newton Meters) / (Radians / Second)

T_{GC} = is the gravitational torque at the coupled load in Newton Meters

$$\alpha_M = [TM - TV - T_C (\omega / |\omega|) - T_G - (\theta_M - \theta_C)/C_C - B_C(\omega_M / \omega_C)] / (J_L + J_M)$$

$$\alpha_L = R_G[((\theta_M - \theta_C) / C_C - B_C(\omega_M / \omega_C)) - T_{VC}\, \omega LC - T_{CC} (\omega LC / |\omega LC|)] - T_{GC} / J_{LC}$$

Velocity and Position can be found by repeated integration

Model of Motor

To get a model of the motor the pull out torque of the motor is measured a several speeds. Pullout torque can be measure by slowly increasing the torque on a motor until it stalls the maximum torque recorded is the pullout torque. Since the pullout torque of the motor includes the motor driver the whole setup is modeled as a unit. This testing will produce a table of pullout torque versus speed. For example

0 Rev/Sec	0.970 NM
1 Rev/Sec	0.970 NM
2 Rev/Sec	0.965 NM
3.2 Rev/Sec	0.962 NM
5 Rev/Sec	0.920 NM
8 Rev/Sec	0.660 NM
10 Rev/Sec	0.523 NM

$$T_P = T_{PO} (\pi / (N_V \sin \pi/N_V)$$

Where :

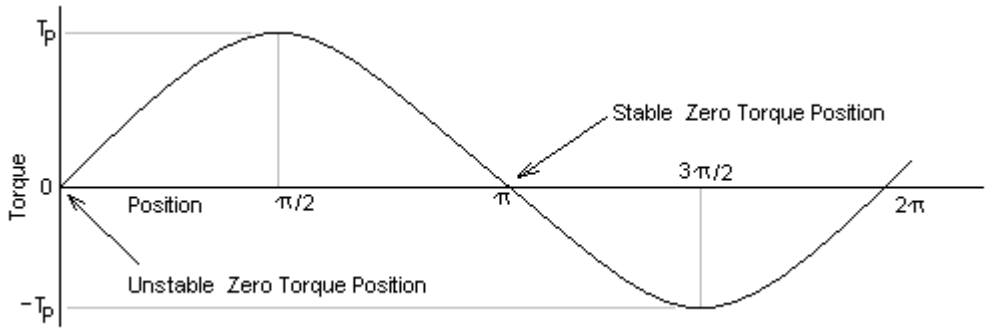
T_P = Peak Torque
T_{PO} = Pullout Torque from table
N_V = Microsteps per full step

The torque of the motor is the sum of the current dependant torque a N cycles

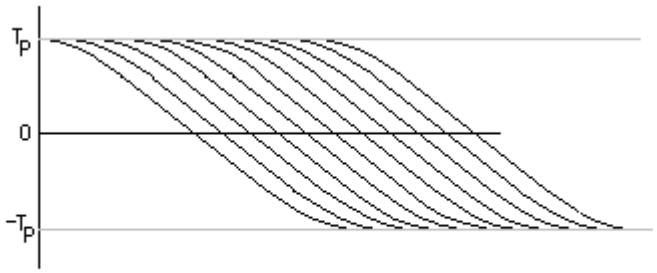
In microstepping systems with 64 microstep per step the peak torque is equal to pullout torque as can be seen by the equation. We now have a torque model of the motor and the drive system which can be used to do simulations.

Torque Curves
A torque verus position curve for a step motor show how the torque on a motor changes as the position changes. The zero position which has a positive slope on the torque curve is not stable because the torque produced drives the position away from zero. Conversely torque produced on the stable position always drive the motor to that stable position.

$T_{RQ} = (J_L + J_M)\ \alpha + T_V\omega + T_C\ (\omega / |\omega|) + T_G$
$T_M = T_p \sin(\pi + N\theta - \theta_e) = T_p \sin(\theta_e - N\theta)$



Above the motor torque (T_m) curve shown from plus and minus maximum torque from its stable rest position. The stable rest position must be a zero torque and the slope of the curve must be negative.



Above is a set of microstep torque curves. Each curve represents the torque available at one microstep value. For value of position on the horizontal axis there are several value of torque which can be selected by changing the microstep.

$N\theta$ is the angle of the motor shaft while θ_e is the angle of the current vector.
$\theta_e = 2\pi (V_c / N_v)$

V_c is the Step State or also know as the microstep number. In a 64 microstep per full step system this number will range from 0 to 63 and then start again a zero. Point on the motor where the detent torque is stable a zero is the V_c = 0 or the first microstep position.

N_v is the number of microsteps per full step.
Vc with vary as the motor is microstep from 0 to (N_v –1). In a 64 microstep system the step state will vary form 0 to 63.

θ_e is the angle at which the motor would rest if no torque was applied to it. To produce torque there must be a displacement between this position and the actual position. This displacement is(θ_e - $N\theta$). As shown by the equation the maximum torque is T_p and it is produce at (θ_e - $N\theta$) = $\pi/2$ radians.
T_p is defined by the table speed torque table. When the system is being simulated we know that:

$T_M = T_{RQ}$

The microstep number V_C is computed by:

$V_C = N_V N_\theta/2\pi + (N_V/2\pi) \sin^{-1}(T_{RQ} / T_P)$

The first term is the position of the motor. This the position at which the motor will produce zero torque. The second term is the displacement from the zero torque position required to produce the torque needed. This is torque feedforward. Without torque feedforward the motor will advance steps based only on constant acceleration or velocity. Torque feedforward modifies this so that the motor advances to produce the required torque to move the load. The simulation can operate with torque feedforward off or on. Making a controller with torque feedforward will require more computation power. When calculating the microstep position (V_C) the result is rounded to the nearest microstep. As you can see the more microstep the closest the result will match. If the calculation of torque feedforward not possible in real time it could be calculated in advance.

$T_M = T_P \sin [2\pi N(\theta + \omega (\Delta T/2) -2\pi V_C/N_V N + \pi]$
$= T_P \sin [2\pi V_C/N_V N -2\pi N(\theta + \omega (\Delta T/2)]$

This equation indicates that the torque produced by the motor is equal to its location on the torque curve. The position on the torque curve is determined by the difference of microstep position (this is the $2\pi V_C/N_V$ term) and the position of the system as it moves (this is the $2\pi N[\theta + \omega (\Delta T /2)]$ term). A reversal in sign occurs because of the torque curve operating points offset by π radians from position at 0 radians.

A review of variables
ΔT is the time tick at which each new position is calculated
$\theta$is the angular position of the load
ω is angular speed of the load.
N is the number of full step in one revolution of the motor.
N_v is the number of microstep per full step.
V_C is the microstep number

$\omega_{new} = \omega_{old} + \alpha \Delta T$
The new velocity is equal to the old velocity plus the acceleration times the time tick.

$\theta_{new} = \theta_{old} + [\ (\omega_{new} + \omega_{old}) / 2] \Delta T$
The new position is equal to the old position plus the average of the old and new velocity times the time tick.

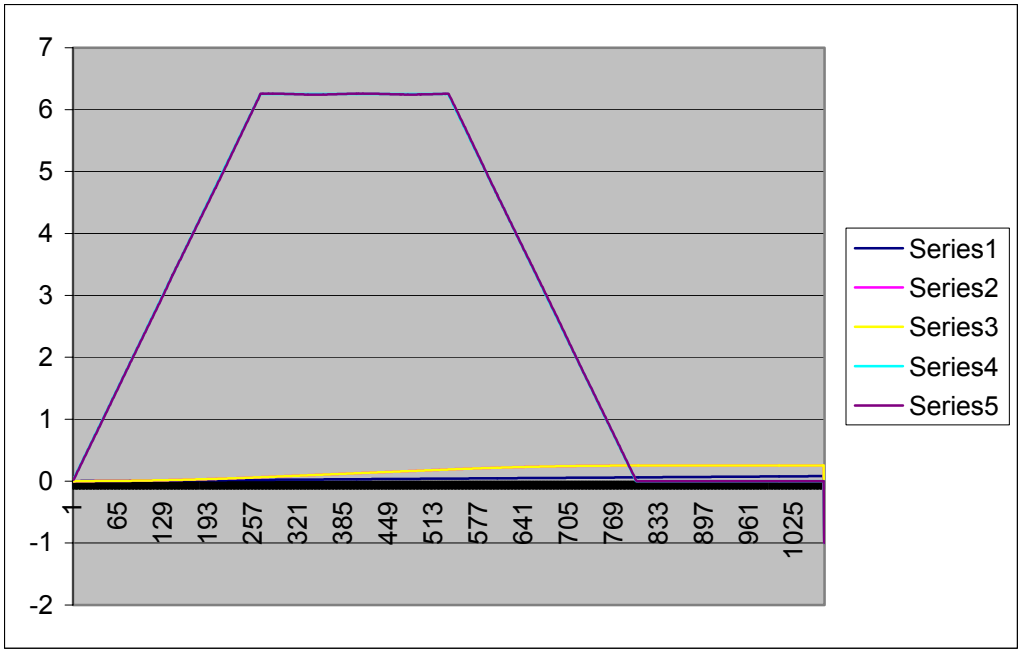
Example:

Inertia:	0.00005 Kilogram Meter^2
Friction:	0.05 Newton Meters
Viscous:	0
Gravitational Load:	0

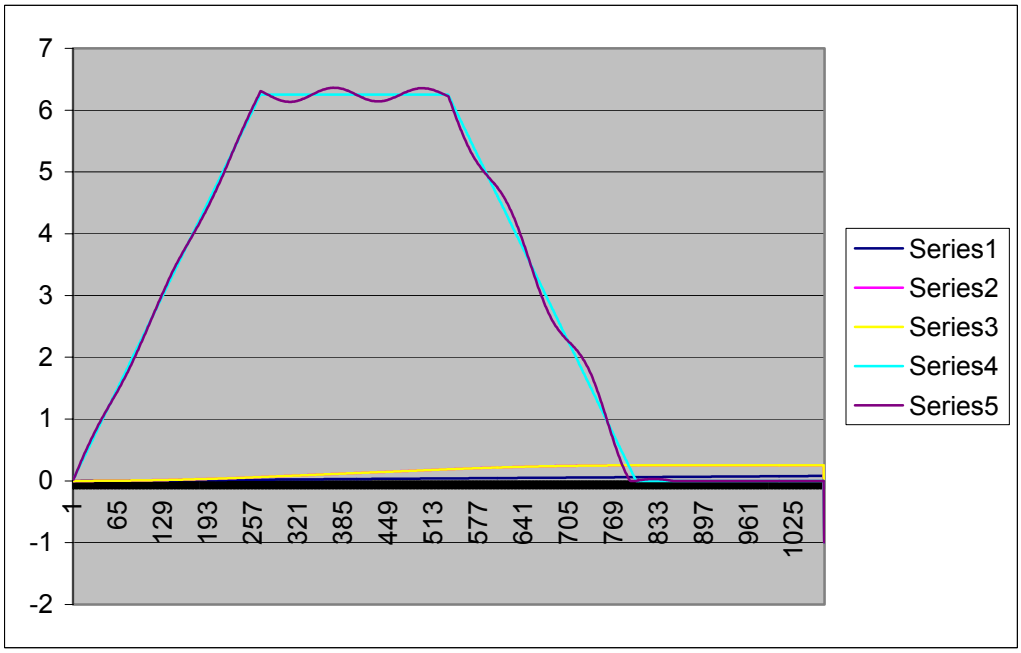
Size 23 Motor Double Stack
24 volt 2 Amp Microstep Drive
Pull defined in a Table as:

Rev/Sec	Pullout Torque
1	0.74
5	0.61
10	0.32
15	0.195

System response with 0.00005 Kilogram Meter^2 :

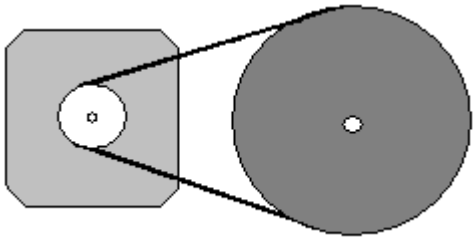


System response not as good with 0.00004 Kilogram Meter^2 :



Simulation using compliant couplings:

The previous simulation used direct coupled load the simulation available in Leenhouts book allow compliant coupling also.



At Pelco the coupling is a toothed belt and sprocket. This is a comliiant coupling.

The equations for system with compliant coupling:

θ_M is the position of the motor and the directly coupled load in Radians

ω_M is the velocity of the motor and the directly coupled load in Radians/Second

θ_C is the position at the output shaft of coupling in Radians

ω_C is the velocity at the output shaft of coupling in Radians / Second

$T_M = T_P \sin[(2\pi V_C/N_C) \ -N(\theta_M + \omega_M (\Delta t/2))]$

The acceleration of the motor, the directly coupled load and the coupling input equals:

$T_{RQ} = (J_L + J_M) \ \alpha + T_V\omega + T_C (\omega/ |\omega|) + T_G$

$\alpha_M = [T_M \ - T_V \ \omega_M - T_C(\omega_M/|\omega|) - T_G - T_{CP}] / (J_M + J_L)$

$T_{CP} = (\theta_M - \theta_C)/ C_C + B_C(\omega_M - \omega_C)$

$\omega_{M\ new} = \omega_{M\ old} + \alpha_M \Delta t$

$\theta_{M\ new} = \theta_{M\ old} + [(\omega_{M\ new} + \omega_{M\ old}) / 2] \Delta t$

The acceleration of the coupled load:

$\alpha_C = [-T_{CP} - T_{VC}\ \omega c - T_{CL}\ (\omega_M/|\omega|) - T_{GC}] / J_{LC}$

$\omega_{C\ new} = \omega_{C\ old} + \alpha_C \Delta t$

$\theta_{C\ old} + [(\omega_{C\ new} + \omega_{C\ old})/2] \Delta t$

Take the previous simulation and adding a compliant, lossy coupling between the motor and load.

Compliance = 0.1 Radian/ (Newton Meter)
Damping = 0.005 (Newton Meter) / (Radian / Second)