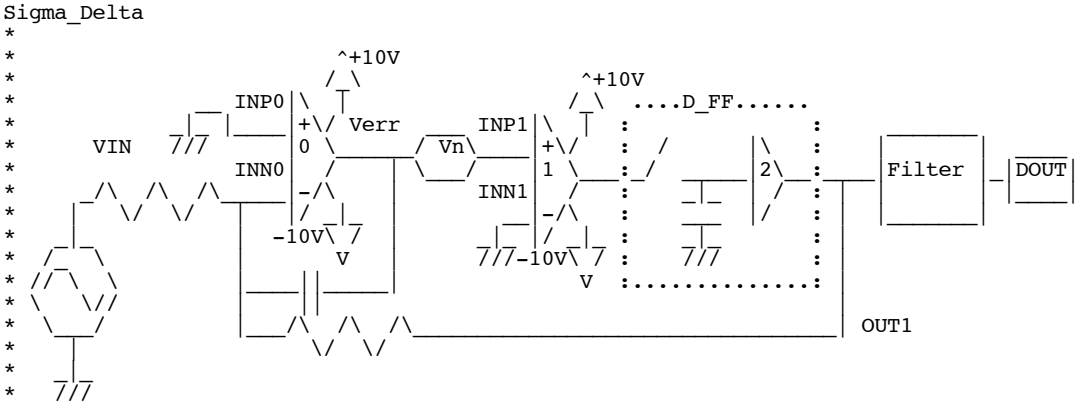
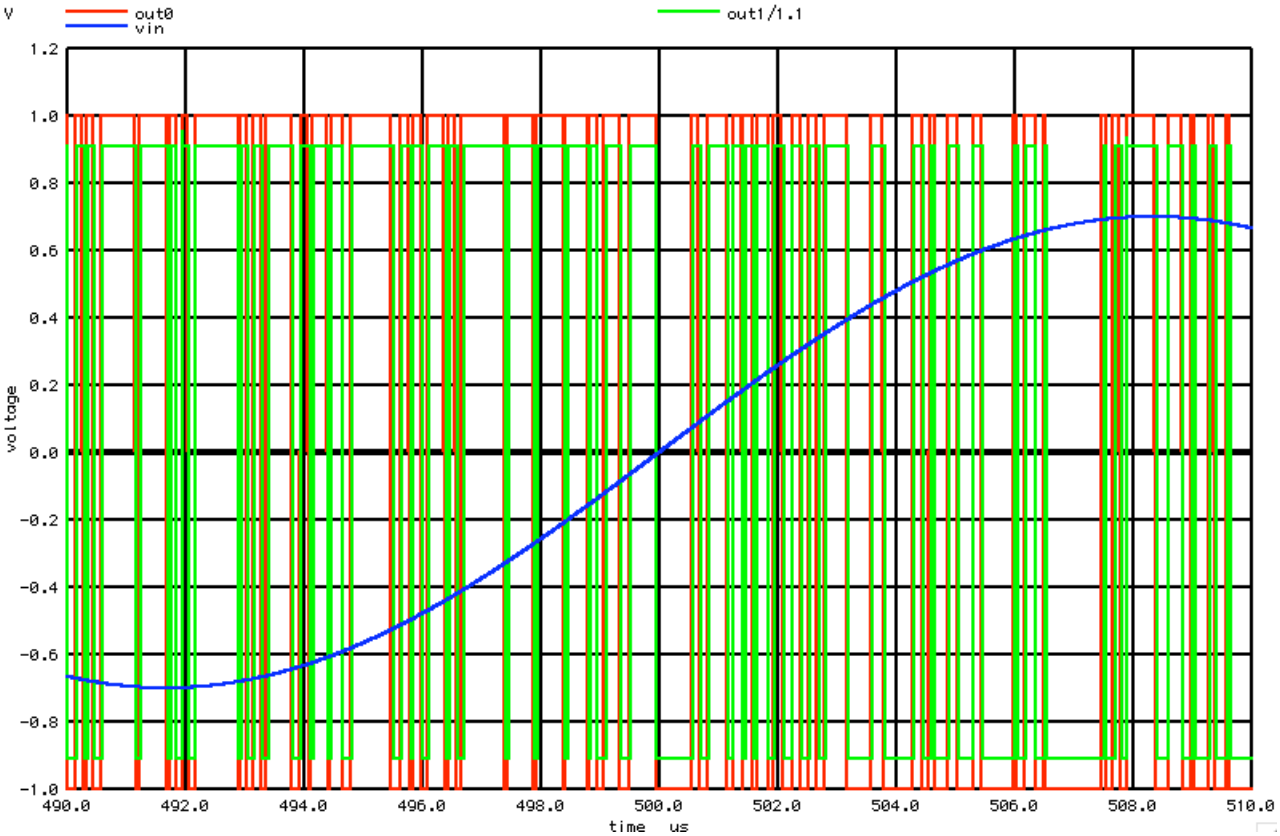
Sigma Delta First Order Noise Shaping



**The Delta Modulation method first integrates the digital signal,
and then subtracts by comparing it to the input signal.**

**Sigma_Delta instead first subtracts the output digital from the analog input,
then it integrates, then it compares.
Simply changing the order of things done in the feedback loop
adds some important improvements.**

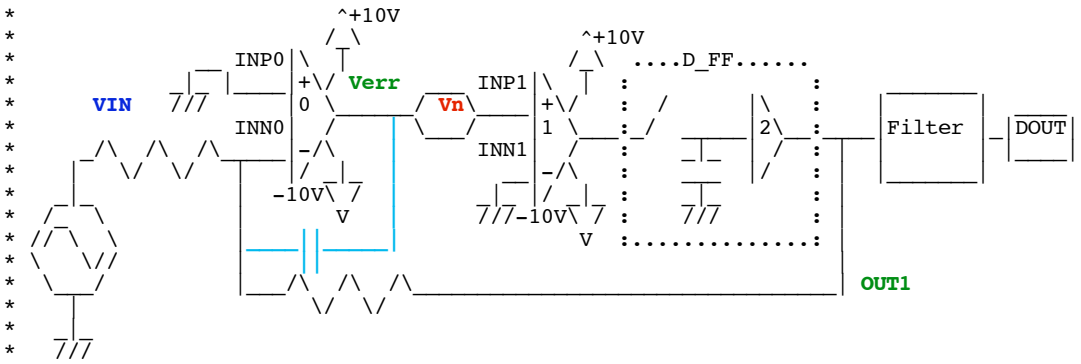


**In the Sigma_Delta, the digital output voltage
is actually a duty cycle pulse width modulation of the input signal.**

**Think of the comparator now functioning as a pulse width modulator
which is being dithered by its own referred to input noise.**

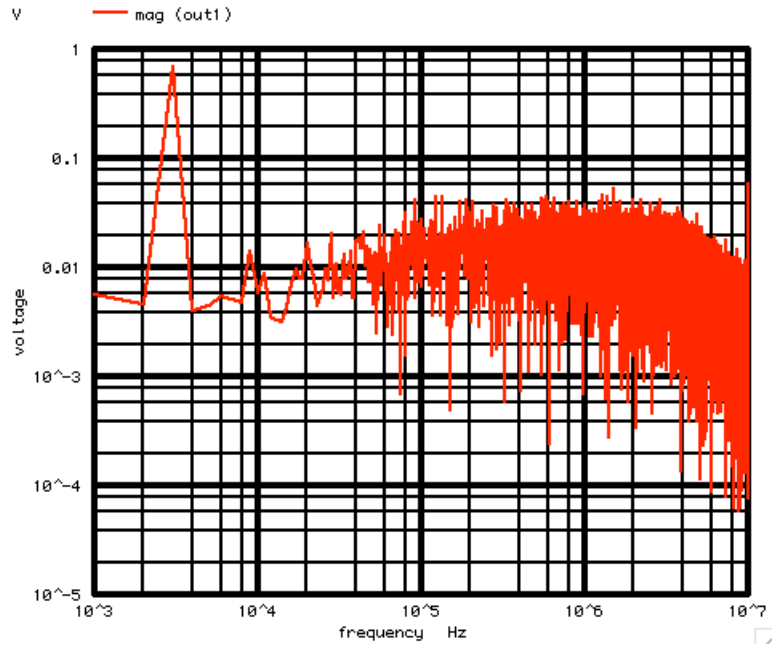
**How well the duty cycle of the digital output
matches the analog input signal is now integrated.
This error signal in turn adjusts the modulation
of the duty cycle of the comparator to minimize the error signal.**

Sigma_Delta



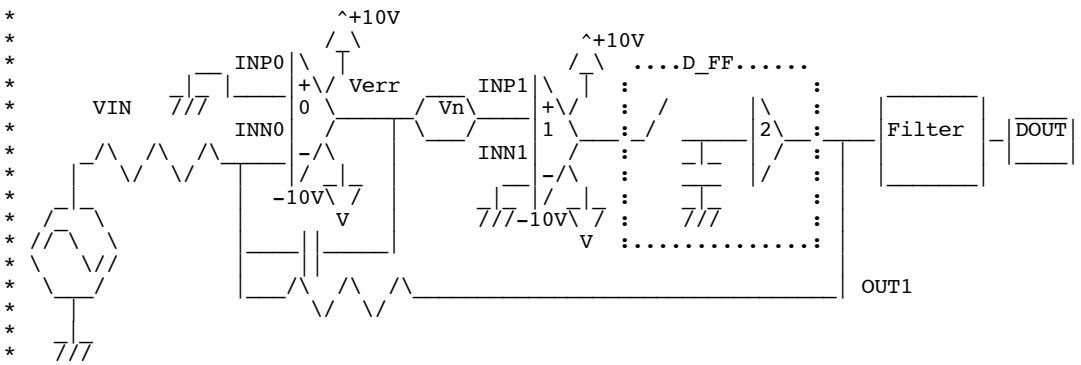
It should be noted that the integration stage means lower loop gain at high frequency than at low frequencies.

This means that the Verr output of the integrator can do noise cancelation on the comparator's input noise Vn at low frequencies. This is called noise shaping.



The spectrum of the digital PWM output signal is shown above. The loop gain is increasing by a factor of ten for every decade decrease in input frequency.

The power of the noise from the comparator will be dropping by this amount. But the noise voltage level will be dropping by the square root of that amount.



It is the noise at the lower frequencies which is usually important.
The higher frequency noise can be digitally filtered out.

```
===================MacSpiceCode=============================================
Sigma_Delta
*
*                        ^+1V
*                       /_\                ^+1V
*               __ INP0|\  T             /_\  ....D_FF......
*             _|_ |____|+\/ Verr   ___ INP1|\  T  :              :   _______
*      VIN   ///       |0 \______/ Vn\____|+\/   :  /      |\    :  |       |
*                  INN0|  /      |\___/   |1 \___:_/  _____|2\___:__|Filter |_|DOUT|
*     _/\  /\  /\______|-/\      |    INN1|  /   :   _|_   |  /  : ||       | |____|
*    |   \/  \/   |    |/ _|_    |       _|-/\   :   ___   | /   : ||_______|
*   _|_           |  -1V\ /      |      _|_|/ _|_ :   _|_        : |
*  /_  \          |       V      |      /// -1V\ / :   ///        : |
* //_\  \         |_____||_______|             V  :..............:|
* \    \//        |     ||                                         |
*  \___/          |___/\  /\  /\_______________________________| OUT1
*                        \/  \/
*    _|_
*    ///

*================Need_A_voltage_Source_to_alter====================="
VN1               VERR    INP1   DC     0
*V_PULSE#         NODE_P  NODE_N DC     VALUE  PULSE( VINIT  VPULSE TDELAY TRISE  TFALL  PWIDTH PERIOD )
VCNTL             CNTL    0      DC     0      PULSE( 0      1      .5p    1n     1n     5n     50n     )
*V_SIN#           NODE_P  NODE_N DC     VALUE  SIN(   V_DC   AC_MAG FREQ   DELAY  FDamp)
VIN               VIN     0      DC     0      SIN(   0      .7     30k     )

R0                VIN     INN0   300k
B0                VERR    0      V =    1*tanh(100*tanh(100*tanh(100*tanh(100*tanh( -V(INN0))))))
C1                INN0    VERR   1n
R1                INN0    OUT1   300k
B1                OUT0    0      V =    1*tanh(100*tanh(100*tanh(100*tanh(100*tanh( V(INP1)  )))))

*S_NUMB           NODE1   NODE2  CNTL_P CNTL_N MODEL  ON/OFF
BCNTL             NCNTL   0      V  =   1-V(CNTL)
SA                OUT0    OUTA   CNTL   0      SW
CA                OUTA    0      100p
BA                OUTB    0      V  =   V(OUTA)
SB                OUTB    OUTC   NCNTL  0      SW
CC                OUTC    0      100p
BC                OUT1    0      V  =   V(OUTC)

.MODEL            SW      SW(    VT=.2 VH=.1m  RON=10m  ROFF=1MEG)

.control
set               pensize = 2
echo              "=======================Want_10000_.1us_steps==============================="
let n =           10000
let tstep =       .1us
let period_t =    n*tstep
let Bin_Hz =      1/period_t
let nyquist =     .5/tstep
echo              "Total_Period_s =      $&period_t"
echo              "Bin_Resolutio_Hz =    $&Bin_Hz"
echo              "Sample_Period_s =     $&tstep"
echo              "Nyquist_Hz =          $&nyquist"
echo              "=======================Create_PWL_array_and_Index_and_Plot=============="
let pwl_1 =       vector(2*n)*tstep*0.5
let ii =          vector(2*$&n)
*plot             pwl_1  vs ii
echo              "=======================Add_1Vrms_Noise_to_PWL_array====================="
let index =       0
repeat            $&n
let               pwl_1[1+2*index] = 10m*(rnd(127)+rnd(127)+rnd(127)+rnd(127)+rnd(127)+rnd(127)+rnd(127)+
rnd(127)-507.5)/102.879
let index =       index + 1
end
echo              "=======================Install_the_PWL_array============================="
alter             @vn1[pwl] = pwl_1
echo              "=======================Run_and_Plot======================================"
let period_s =    tstep/2
let trans_per =   tstep/20
tran              $&trans_per $&period_t   0  $&trans_per
plot              inp1 vinn

plot              out1 vin
plot              out0   out1/1.1  cntl vin xlimit .495m .505m
plot              out0   out1/1.1  vin xlimit .49m .51m
```

```
.endc
.end


echo             "======================FFT_and_Plot====================================="
linearize
let              FFT_BandWidth_Hz =      10Meg
let              FFT_resolution_Hz =     1k
echo             "FFT_BandWidth_Hz=      $&FFT_BandWidth_Hz"
echo             "FFT_resolution_Hz=     $&FFT_resolution_Hz"
set              specwindow=             "rectangular"
spec             $&FFT_resolution_Hz     $&FFT_BandWidth_Hz   $&FFT_resolution_Hz     v(out1)
let expect_V =   (sqrt(2)/sqrt(500k/1k))/(1+(frequency/550k)*(frequency/500k)*(frequency/500k)*(frequency/
500k)*(frequency/500k))
plot             mag (out1) loglog
echo             "==================done============================================="


7.22.11_12.56PM
dsauersanjose@aol.com
Don Sauer
```