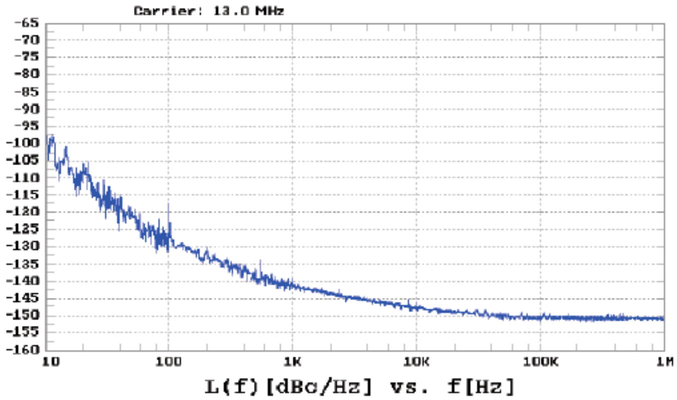
=====ACCUMULATED PHASE NOISE IN OSCILLATORS===========

Fig.8 is a Phase Noise plot of a real 13.0MHz Crystal Oscillator.

Fig. 8



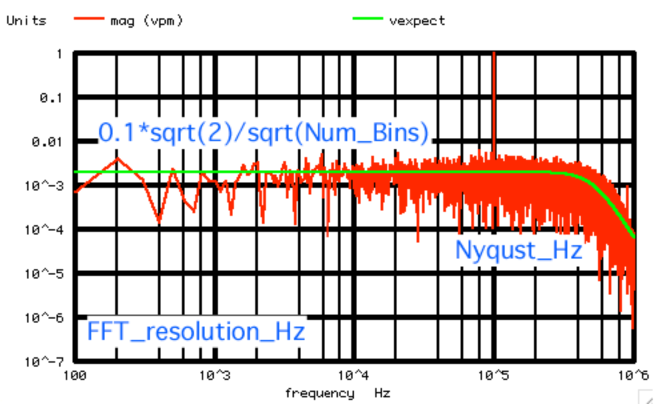
When looking at a Phase noise plot, it takes 100ms to view a 10Hz signal.
A 13Hz clock accumulates 1.3 million cycles of timing tolerance during this time.
So phase noise should be looked at in terms of an accumulation factor.

Accum_factor = Freq_Carrier_Hz/(Freq_Offset_Hz)
= Numbers of clock cycles accumulated

It is straight forward to predict the spectrum of phase modulation.
A write up as how to do that can be found here.
http://www.idea2ic.com/For_Better_Or_Worst/SIMPLE_RANDOM_PM_WAVEFORM_GENERATION.pdf

The flat_band_magnitude for 10% radian noise is is such....

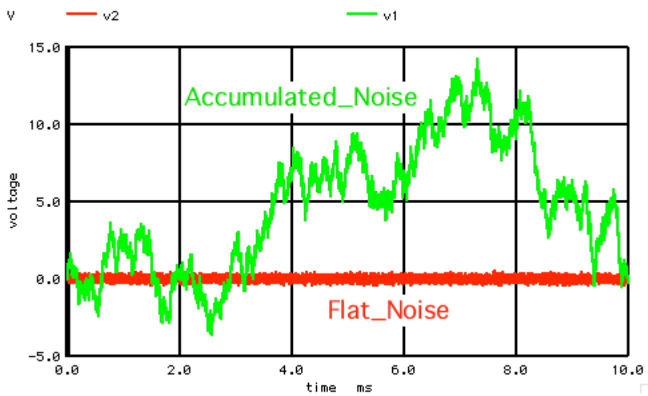
flat_band_magnitude = 0.1*sqrt(2)/sqrt(Num_Bins)



Now integrate or accumulate that same 10% radian noise,
and use it to phase modulate a 100KHz signal.
The rms level of accumulated noise is some what predictable.
A write up on how that is possible can be found here.
http://www.idea2ic.com/For_Better_Or_Worst/Accumulated_Randomness.pdf

RMS_level_Expect 0.1*0.707*sqrt(Numb_Cycles)/2

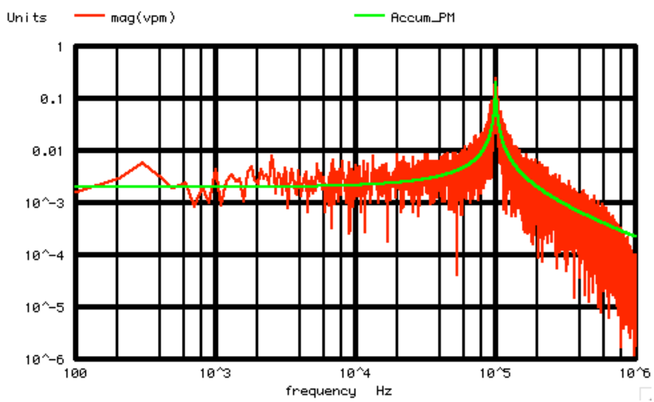
```
=======================Want_10000_1us_steps===============================
Total_Period_s =        0.01
Bin_Resolutio_Hz =      100
Sample_Period_s =       1E-06
Nyquist_Hz =            500000
Total_Bins =            5000
=======================Create_PWL_array_and_Accum_and_Plot==============
=======================Add_.1Vrms_Noise_to_PWL_array====================
=======================Adjust_Endpoint===================================
=======================Find_Ave_Rms_V1===================================
RMS_level_Expect        0.1*0.707*sqrt(10000)/2
RMS_level_Expect        3.535
RMS_level_RM            3.00162
```



The accumulation factor is a function of frequency as such..

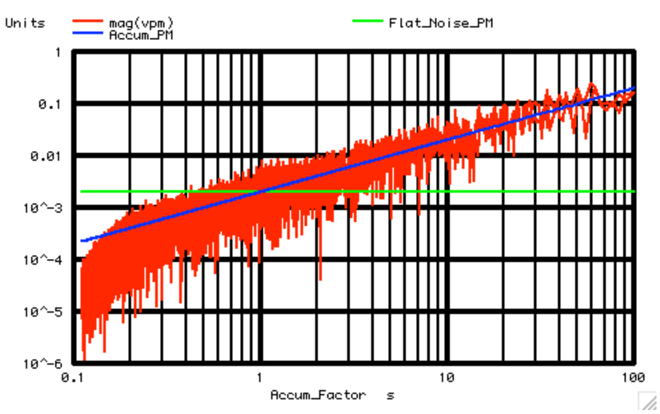
Accum_factor(Frequency_Hz) = Clock_Hz/(mag(Frequency_Hz-Clock_Hz))

Accumlated_PM_Noise_curve = flat_band_magnitude*Accum_factor(Frequency_Hz)



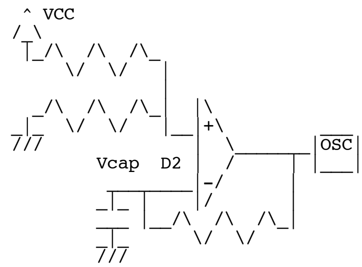
Scaling the Accumulation_factor curve by the flat_band_magnitude.
This will generate an Accumlated_PM_Noise_curve
which comes close predicting the actual Phase Noise of the clock.
At least until Nyquist.

Now plot full spectrum, flat_band_magnitude, and Accumlated_PM_Noise,
all versus the Accumulation_factor.



Notice that the full spectrum and the flat_band_magnitude
both cross each other at an Accumulation_factor of one. This make sense.
There is little accumulation of random phase error over one clock period.

But over ten times more time, the flat noise is getting integrated.
It produces a noise level that is 20dB higher at a ten times lower frequency.



Timing tolerance is signal level divided by noise level.
This timing tolerance ratio can be in percent, or as radians, or in terms of time.

The comparison process samples the randomness over full Nyquist.
This pre-accumulated level times the accumulation factor predicts the phase noise.

===================MacSpiceCode==================================

```
MAPPING_PHASE_NOISE
*=========Create_Signal==================
VTime     VTime  0      DC     0    PWL(    0      0    1      1)
Vfreq1    Vfreq1 0      DC     2
V1        V1     0      DC     0
V2        V2     0      DC     0
BMOD      VMOD   0      V      = cos(6.2831853*2000*V(VTime))
BPM       VPM    0      V      = 1*cos(6.2831853*100k*V(VTime)+1*V(V1))
BCOS      VCOS   0      V      = 1*cos(6.2831853*100k*V(VTime))

.control
*TRAN            TSTEP  TSTOP  TSTART TMAX   ?UIC?

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
let binsTotal=   nyquist/Bin_Hz
echo             "Total_Period_s =       $&period_t"
echo             "Bin_Resolutio_Hz =     $&Bin_Hz"
echo             "Sample_Period_s =      $&tstep"
echo             "Nyquist_Hz =           $&nyquist"
echo             "Total_Bins =           $&binsTotal"
echo             "======================Create_PWL_array_and_Index_and_Plot=============="
let pwl_1 =      vector(2*n)*tstep*0.5
let pwl_2 =      vector(2*n)*tstep*0.5
let ii =         vector(2*$&n)
echo             "======================Add_.1Vrms_Noise_to_PWL_array====================="

let n2 =         n-1
let              pwl_1[0] = 0
let index =      1
repeat           $&n2
let vnoise =     .1414*(rnd(127)+rnd(127)+rnd(127)+rnd(127)+rnd(127)+rnd(127)+rnd(127)+rnd(127)-507.5)/102.879
let              pwl_1[1+2*index] = pwl_1[-1+2*index] + vnoise
let              pwl_2[1+2*index] = vnoise
let index =      index + 1
end
echo             "======================Adjust_Endpoint====================================="

let endpt =      pwl_1[19999]
*print  endpt
let index =      1
repeat           $&n2
let              pwl_1[1+2*index] = pwl_1[1+2*index] -1*endpt*index/10000
let index =      index + 1
end
let endpt =      pwl_1[1999]
*print  endpt

echo             "======================Find_Ave_Rms_V1====================================="
let averVal =    mean(pwl_1)
let noisAC =     pwl_1 - averVal
let RmsVal =     1*sqrt(mean(noisAC* noisAC))
let rms_exp =    0.1*.707*sqrt(10000)/2
echo             "RMS_level_Expect       0.1*0.707*sqrt(10000)/2  "
echo             "RMS_level_Expect       $&rms_exp  "
echo             "RMS_level_RM           $&RmsVal  "
unlet averVal
unlet RmsVal

echo             "======================Install_the_PWL_array==========================="
alter            @v1[pwl] = pwl_1
alter            @v2[pwl] = pwl_2
tran             .1u    10m     0       .1u
set              pensize = 2
plot             v2 v1

echo             "======================FFT_and_Plot_VPM===================================="
linearize
let              FFT_BandWidth_Hz =      1meg
let              FFT_resolution_Hz =     100
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(vpm)
let              Flat_Noise_PM =         .1*sqrt(2)/sqrt(5000)
echo             "Flat_Noise_PM          .1*sqrt(2)/sqrt(5000) "
echo             "Flat_Noise_PM          $&Flat_Noise_PM  "
let              Accum_Factor =          100k/(mag(frequency-100k)+1k)
echo             "Accum_Factor           100k/(mag(frequency-100k)+1k)"
echo             "Accumulated_PM         Flat_Noise_PM*Accum_Factor"
let              Accum_PM =              Flat_Noise_PM*Accum_Factor
plot             mag(vpm) Flat_Noise_PM  Accum_PM  vs Accum_Factor loglog
plot             mag(vpm) Accum_PM        loglog
echo             "=======================done=================================================="

.endc
.end
```

4.18.11_1.22PM

dsauersanjose@aol.com
Don Sauer