*=======TRANSIENT_OSCILLATOR_PHASE_NOISE_JITTER============

Adding a piece wise linear noise source to a free running oscillator can show how the noise maps to jitter or PM or to a spectrum.

In this case a 2second/1kHz BW random PWL file is created.

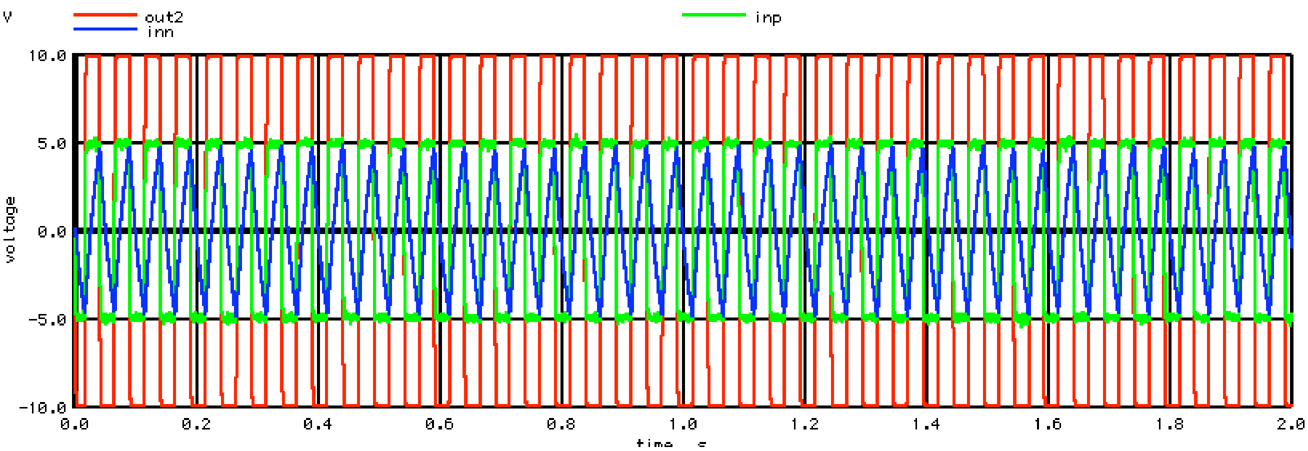
But in this case, a jitter plot will be taken and used to produce a PWL file, which then can be used to Phase Modulate a 20 Hz signal.

```
OSCILLATOR_WITH_PWL_Noise_Jitter
*                R1
*    _____/\  /\  /\__
*    |     \/  \/    _|_
* VFB|               ///
*    |___/\  /\  /\_______________
*    |    \/  \/         |        |
*    |          R2      _|_  C0   |
*   _|_                 ___       |   ___
*  / _ \ BPWL            |        |__|OUT|
* // \  \               _|_200uF  |  |___|
* \   \//               ///       |
*  \___/     INP  |\              |
*    |____________|+\             |
*                 |  \_/\  /\  /\_|
*    C1     INN   |  /   \/  \/   |
*      ___________|-/     R0 1 Ohm|
*    _|_   |      |/ BAMP         |
*    ___   |                      |
*     |    |    R3                |
*    _|_   |__/\  /\  /\__________|
*    ///       \/  \/
*
* VpwlT OUT 0        PWL( + 0.0005 0.988835 +.....
*=========Include_Noise_File===============================================
.include  PWL_File.inc
Rload     OUT    0      1k
BAMP      OUT1   0      V =    9.9*tanh( (V(INP)-V(INN))*10)
R0        OUT1   OUT2   1
C0        OUT2   0      500u
R1        VFB    0      1K
R2        VFB    OUT2   1K
R3        INN    OUT2   2.49K
BPWL      INP    VFB    V =    .1*V(OUT)
C1        INN    0      9.415u IC= .1
*=========Run_Simulation==================================================
*TRAN     TSTEP  TSTOP  TSTART TMAX      ?UIC?
.tran     5u     2      0      5u        UIC
.control
run
set       pensize =     2
plot      out2   inp    inn
*=================================================================
```

Noise from BPWL is applied to one of the inputs of the amplifier BAMP. Amplifier BAMP is clipping its output at +/- 10Volts.

Capacitor C1 is toggling between +/- 5volts. Noise source BPWL is messing up the timing. So in this case C1 has been adjusted to produce 40 whole cycles within 2 seconds.

Noise source BPWL is receiving a random 1VRMS signal with a bandwidth of 1KHz. This has been scaled down to +/- 100mV rms.



The Noise source BPWL can be seen on the inp input. But its PM effects are hard to see. A jitter plot can show better details.

```
*=========Create_AnySize_Arrays==========================================
compose    anysize     start = 0 stop = 99 step =1
let        num =       length(out2)-5
let i  =   0
let t  =   0
let n  =   0
*======================================================================
```

Assume the number of rising or falling edges are not known at this point. So array anysize will be used to store an unknown number of data points. The total number of output points (num) for the oscillator output is easy to find.

Some simple "if" statements can be used to find the timing for the edges.

```
*=========Find_Edge_Timing==============================================
repeat     $&num
if         ( out2[i] < 0 & out2[i+1] > 0)
let t =    time[i]
let        anysize[n]= t
echo       n=    $&n    out_rise= $&t
let        n =    n +1
endif
if         ( out2[i] > 0 & out2[i+1] < 0)
let t =    time[i]
let        anysize[n]= t
echo       n=    $&n    out_fall= $&t
let        n =    n +1
endif
let i =    i +1
endrepeat
let        n3 =   n -1
*======================================================================
```

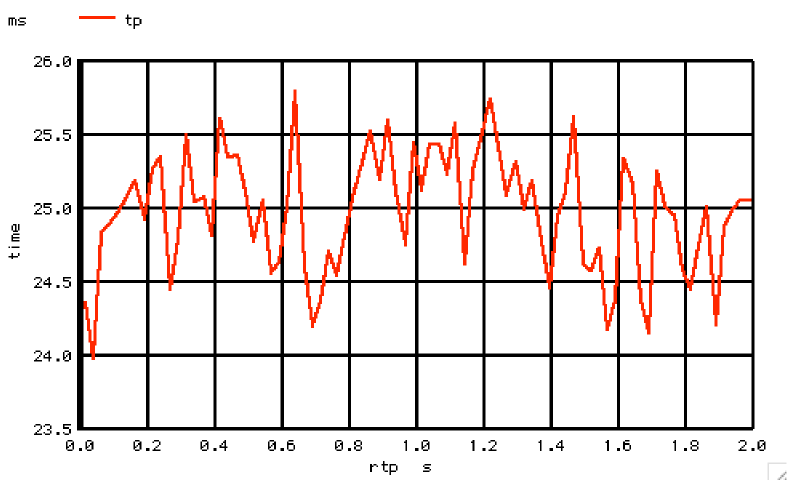
The MacSpice printout..

```
n = 0  out_rise = 0.0159425
n = 1  out_fall = 0.0402975
n = 2  out_rise = 0.0642775
n = 3  out_fall = 0.0891075
n = 4  out_rise = 0.114012
n = 5  out_fall = 0.138993
n = 6  out_rise = 0.164068
...
n = 76 out_rise = 1.91342
n = 77 out_fall = 1.9383
n = 78 out_rise = 1.96327
n = 79 out_fall = 1.98833
```

Now that the number of edge data points are known, some new arrays can be created to store and plot

the results.

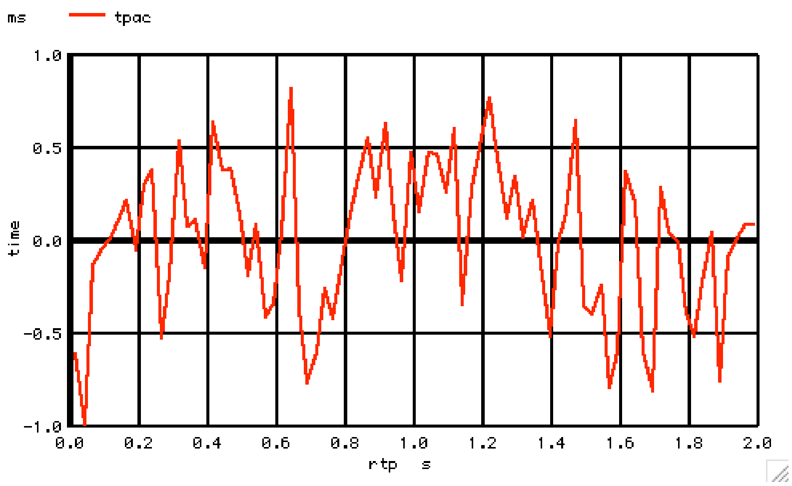
```
*=========Create_Edge_Time_Arrays===========================================
compose    tp            start = 0 stop = $&n3 step =1
compose    tpac          start = 0 stop = $&n3 step =1
compose    td            start = 0 stop = $&n3 step =1
compose    tdac          start = 0 stop = $&n3 step =1
compose    rtp           start = 0 stop = $&n3 step =1
compose    pmr           start = 0 stop = $&n3 step =1
*=========Transfer_Arrays===========================================
let i  =   0
repeat     $&n
let        rtp[i] = anysize[i]
let i =    i +1
endrepeat
let i  =   0
let n2 =   n -1
repeat     $&n2
let        tp[i] = rtp[i+1] -rtp[i]
let i =    i +1
endrepeat
let        tp[n2] = tp[n2-1]
plot       tp  vs rtp
*=====================================================================
```



In this case rtp stands for time reference point. That is the time when the transition happened. The value tp stands for time period. This is the actual time between edges. Notice that the average time period is 25msec. A 20Hz square wave has two transitions within 50msec.

It is easy to do some further math on the data.

```
*=========Remove_Average_Time_Period===========================================
let        tpave = mean(tp)
let        tpac =  tp -tpave
plot       tpac vs rtp
*=========Find_RMS_Vtpac=========================
let        i  =     0
let        vpwr =   0
repeat     $&n2
let        i =      i +1
let        vpwr =   vpwr   + (mag(tpac[i])*mag(tpac[i]))/n2
end
let        vrms1 = sqrt(vpwr)
echo       Edge2Edge_Period $&tpave TPAC RMS SQUARE =  $&vrms1
*=====================================================================
```



In this case **tpac** **stands for** **time reference point AC.**
The average time period has been stripped away.

Now it is easy to do a RMS of the data and print out both the average and RMS levels.

The MacSpice printout...

Edge2Edge_Period 0.024968 TPAC RMS SQUARE = 0.000404478

Consider the ratio of the RMS value to the Average value.

TimePeriod_RMS/AVE = 0.0162

```
OSCILLATOR_WITH_PWL_Noise
*                R1
*    _____/\  /\  /\__
*    |      \/  \/    _|_
* VFB|                ///
*    |___/\  /\  /\__________________
*    |      \/  \/          _|_     |
*    |           R2         ___ C0  |
*   _|_  BPWL               _|_     |
*  /   \ BPWL                |      |__|OUT|
* //^\  \                   _|_200uF|  |___|
* \   \//                   ///     |
*  \___/     INP   |\               |
*    |_____________|+\              |
*                  |  \_/\  /\  /\_ |
*    C1     INN    |  /   \/  \/    |
*                  |-/      RO 1 Ohm|
*      _________   |/ BAMP          |
*     _|_    |                      |
*     ___    |       R3             |
*     _|_    |___/\  /\  /\_________|
*     ///          \/  \/
```

Capacitor C1 is swinging between -5V and +5V.
And at each end there is an uncertainty of +/- 100mV rms.
So in this case, the ratio of the RMS value to Average value is.

C1_TimePeriod_RMS/AVE = sqrt(2)*100mv/10V = 0.01414

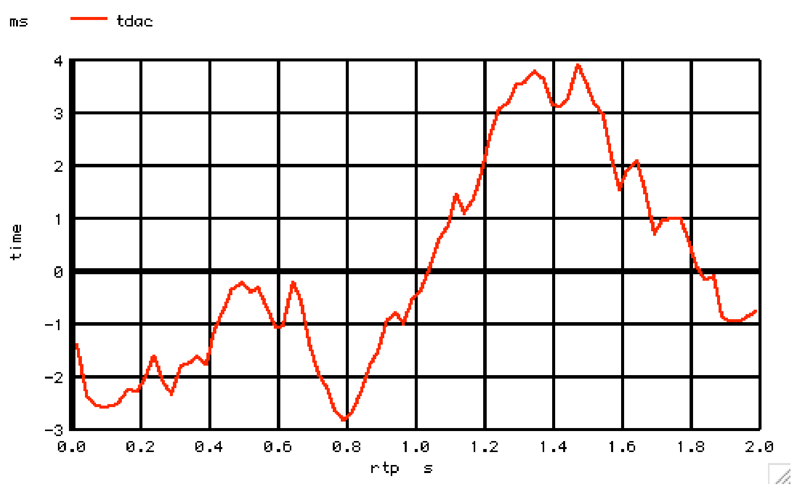
The two **ratios of uncertainty to average** **value should come close to each other. The amplifier** **BAMP** **is in effect sampling two 100mV rms random points to be compared to a 10volt swing. That ratio of uncertainty to the average value gets directly mapped to the time period uncertainty.**

Now this 100mV rms noise has a 1kHz bandwidth. Even though a grand total of 80 samples of this noise is taken over 2 seconds, the RMS value for all samples is still 100mV. This is a case

of sampling without an anti-aliasing filter. So the 1kHz noise just got all alaised down to within a 20Hz bandwidth.

But variation in time period is really frequency modulation. All the ac time periods need to be added up to see the overall phase timing.

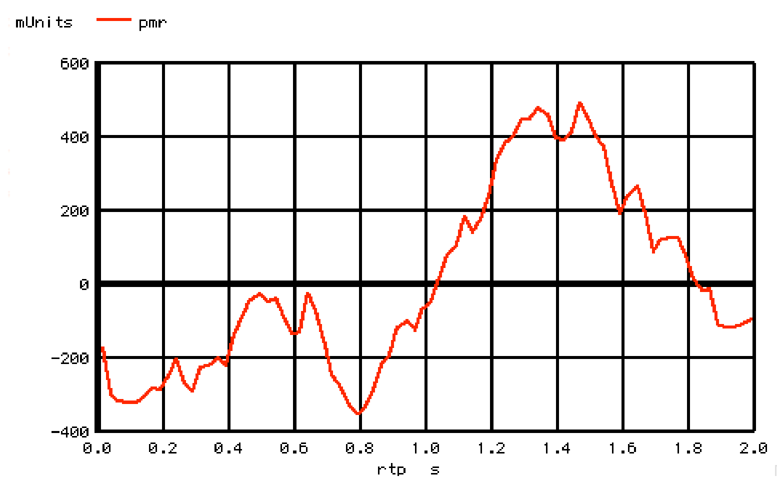
```
*=========Convert_FM_to_PM========================================
let i  =  1
let n2 =  n -1
repeat    $&n2
let       td[i] = td[i-1] +tpac[i]
let i =   i +1
endrepeat
plot      td    vs rtp
*=========Remove_Average_Phase============================================
let       tdave = mean(td)
let       tdac  =  td -tdave
plot      tdac  vs rtp

*==========================================================================
```



In this case tdac stands for time delay AC. This is how much each edge is "delayed" in time compared to a perfect 20Hz square wave.

This can further be converted to a phase modulation format in terms of radians.

```
*=========Convert_to_PM_radian============================================
let       pmr =3.14159*tdac/tpave
plot      pmr  vs rtp
*==========================================================================
```



Now pmr stands for phase modulation radians.
In this format, the jitter can be treated like a modulation signal which can be exported to a piece wise linear file.

```
*=========Write_To_PWL_File============================================
set       outfile = "PWL_FileJitter.inc"
echo      "VpwlT OUT 0 PWL(" > $outfile
let       i   =  1
let       t   =  0
let       ph  =  0
repeat    $&n2
let       t  =  rtp[i]
let       ph  = pmr[i]
echo      "+ $&t $&ph" >> $outfile
let       i =   i +1
endrepeat
echo      "+ )" >> $outfile
*=========Wrap_Up======================================================
.endc
.end
*=====================================================================
```

The PWL_FileJitter.inc file comes out looking like so.

```
VpwlT OUT 0 PWL(
+ 0.0402975 -0.298113
+ 0.0642775 -0.315477
+ 0.0891075 -0.3235
+ 0.114012  -0.321894
.....
+ 1.88921   -0.107741
+ 1.91342   -0.119442
+ 1.9383    -0.118562
+ 1.96327   -0.107615
+ 1.98833   -0.0966681
+ )
```

```
===========Full_Netlist_For_Copy_Paste========================
OSCILLATOR_WITH_PWL_Noise_Jitter
*                R1
*    _____/\  /\  /\__
*    |      \/  \/   _|_
* VFB|               ///
*    |___/\  /\  /\______________
*    |     \/  \/         _|_   |
*   _|_           R2      ___ CO|
*  /_  \ BPWL              |    |__|OUT|
* //\   \                 _|_200uF |___|
* \   \//                 ///   |
*  \___/    INP |\              |
*    |__________|+\             |
*               |  \_/\  /\  /\_|
*    C1    INN  |  /   \/  \/   |
*       ________|-/     RO 1 Ohm|
*    _|_   |    |/ BAMP         |
*    ___   |                    |
*     |    |       R3           |
*    _|_   |__/\  /\  /\________|
*    ///       \/  \/
*
* VpwlT OUT 0        PWL( + 0.0005 0.988835 +.....
.include  PWL_File.inc
Rload     OUT    0      1k
BAMP      OUT1   0      V =    9.9*tanh( (V(INP)-V(INN))*10)
RO        OUT1   OUT2   1
CO        OUT2   0      500u
R1        VFB    0      1K
R2        VFB    OUT2   1K
R3        INN    OUT2   2.49K
BPWL      INP    VFB    V =    .1*V(OUT)
C1        INN    0      9.415u IC= .1
*TRAN     TSTEP  TSTOP  TSTART TMAX      ?UIC?
.tran     5u     2      0      5u        UIC
.control
run
set       pensize =     2
```

```
plot      out2   inp    inn

*=========Create_AnySize_Arrays===========================================
compose   anysize     start = 0 stop = 99 step =1
let       num =       length(out2)-5
let i  =  0
let t  =  0
let n  =  0
*=========Find_Edge_Timing===========================================
repeat    $&num
if        ( out2[i] < 0 & out2[i+1] > 0)
let t =   time[i]
let       anysize[n]= t
echo      n=   $&n   out_rise= $&t
let       n =   n +1
endif
if        ( out2[i] > 0 & out2[i+1] < 0)
let t =   time[i]
let       anysize[n]= t
echo      n=   $&n   out_fall= $&t
let       n =   n +1
endif
let i =   i +1
endrepeat
let       n3 =  n -1

*=========Create_Edge_Time_Arrays===========================================
compose   tp           start = 0 stop = $&n3 step =1
compose   tpac         start = 0 stop = $&n3 step =1
compose   td           start = 0 stop = $&n3 step =1
compose   tdac         start = 0 stop = $&n3 step =1
compose   rtp          start = 0 stop = $&n3 step =1
compose   pmr          start = 0 stop = $&n3 step =1
*=========Transfer_Arrays===========================================
let i  =  0
repeat    $&n
let       rtp[i] = anysize[i]
let i =   i +1
endrepeat
let i  =  0
let n2 =  n -1
repeat    $&n2
let       tp[i] = rtp[i+1] -rtp[i]
let i =   i +1
endrepeat
let       tp[n2] = tp[n2-1]
plot      tp  vs rtp
*=========Remove_Average_Time_Period===========================================
let       tpave = mean(tp)
let       tpac =  tp -tpave
plot      tpac vs rtp
*=========Find_RMS_Vtpac=========================
let       i  =    0
let       vpwr =  0
repeat    $&n2
let       i =     i +1
let       vpwr =  vpwr  + (mag(tpac[i])*mag(tpac[i]))/n2
end
let       vrms1 = sqrt(vpwr)
*echo     TPAC RMS SQUARE =  $&vrms1
echo      Edge2Edge_Period $&tpave TPAC RMS SQUARE =  $&vrms1
*=========Convert_FM_to_PM===========================================
let i  =  1
let n2 =  n -1
repeat    $&n2
let       td[i] = td[i-1] +tpac[i]
let i =   i +1
endrepeat
plot      td    vs rtp
*=========Remove_Average_Phase===========================================
let       tdave = mean(td)
let       tdac  =  td -tdave
plot      tdac  vs rtp
*=========Convert_to_PM_radian===========================================
let       pmr =3.14159*tdac/tpave
plot      pmr  vs rtp
*=========Write_To_PWL_File===========================================
set       outfile = "PWL_FileJitter.inc"
echo      "VpwlT OUT 0 PWL(" > $outfile
let       i   =  1
let       t   =  0
let       ph  =  0
repeat    $&n2
let       t  =  rtp[i]
```

```
let       ph  = pmr[i]
echo      "+ $&t $&ph" >> $outfile
let       i =   i +1
endrepeat
echo      "+ )" >> $outfile
*=========Wrap_Up=======================================================
.endc
.end
```

2.18.10_12.15PM

dsauersanjose@aol.com
Don Sauer
http://www.idea2ic.com/