

User Guide to RealTime MC

Stephan Weber

Carl-Orff-Str. 9, 83052 Bruckmühl, Germany

E-mail: weberconnect@freenet.de

Tutorial: 1 st Steps	5
Yield and C_{PK} for More Functions & Circuits	14
Further Features: Cuts and Percentiles	15
C_{PK} Speed-up and Errors, and the Generalized C_{PK}	18
Further RealTime Sweeping Applications	26
C_{GPK} Flow for Yield Estimation	27
Bias Error in C_{PK} and C_{GPK}	28
RealTime MC as Guide for Advanced Statistical Techniques.....	30
RealTime MC Hints for Sampling Techniques	31
Bootstrap in RealTime MC	32
Bootstrap for Normal Data and C_{PK}	32
Appendix	35
Multivariate Capability Index.....	35

Mathematically spoken, the application RealTime MC is able to generate statistical data, and based on this it executes different analysis for yield and quantile prediction. From the engineering perspective, it makes a Monte-Carlo analysis, but instead of using an external simulator like SPICE, several circuits are directly built-in to allow a much higher speed (no netlisting, no tool loading, no communication between different tools, etc.). This way we can execute a realistic huge MC analysis within minutes and get a kind of golden reference, or we can run a MC analysis with 1000 points 250 times to inspect the differences between the different individual MC runs. This way you can really get an immediate “real-time” feeling for statistics, and this even related to highly complex circuit design problems!

The mathematical reason why this "equivalence" works is this: For histogram, yield and C_{PK} it does not matter at all whether the data is coming from a simple random number generator or from a huge circuit which may depend on millions of variables!

Note: In principle, it would be also possible to use a spreadsheet program to learn about statistics, but this would almost exclude the circuit simulation part, and the speed and the amount of data would be still limited. Of course, one advantage is that spreadsheets are quite easy to extend, therefore we included to the River Publishers book "Circuit Design: Anticipate, Analyze, Exploit Variations" also some Excel[®] sheets, e.g. the bootstrap technique is available in the MC app and as spreadsheet.

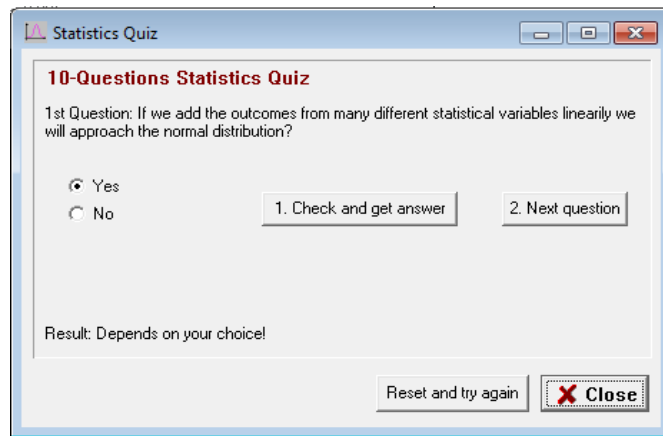
The major reason for creating the Monte-Carlo app is to give designers more insights to statistics in an intuitive way, like on MC itself, on estimates (like mean and median), on confidence intervals, on settings for histograms, etc. In modern design environments, you can do similar analysis as well, but with real circuits and real circuit simulations MC can be quite or even extremely time-consuming. So you may know from experience that MC results depend on chance, but it is hard to know how much! With RealTime MC you can learn about Monte-Carlo quickly and you can apply the know-how to your advantage in more complex, more realistic examples. Actually using MC can be often a *good motivation* to use more complex, highly advanced algorithms, like the ones inside Cadence[®] ADE GXL or other IC design environments (e.g., worst-case distances, MC with sample re-ordering, scaled-sigma sampling, etc.).

In addition, the data analysis itself within RealTime MC is significantly extended to what is typically available in design environments. This has been made to allow better yield estimation, for both normal and non-normal data. MC will provide guidance and answers like what-if analysis "how much do I need to change my specification to get 3-sigma yield (99.7%)", or "This is my yield and C_{PK} estimate but what is guaranteed with 95% confidence?" or "How many samples do I need to achieve a certain yield with certain confidence?", etc. This way MC can also act as a Monte-Carlo guide for users which are no statistics experts (yet).

By pure counting and thus calculating the sample yield $Y = n_{\text{pass}}/n_{\text{tot}}$ we can estimate the true yield without any systematic error. However, this method acts like a 1-bit ADC which has a bad SNR, and therefore it requires large (or even huge) sample counts for verification with sufficient confidence (like 3000 or more points for 0.13% yield loss and millions for 5-sigma or more). For normal Gaussian distributions you can get yield predictions with much less variance using the $C_{PK} = f(\text{Spec limit(s)}, \mu, \sigma)$ but that comes with systematic errors if the data is non-normal (e.g., having specs in dB or nonlinear measures such as delay, filter responses, ADC DNL, etc.). For this reason, the generalized C_{PK} has been created [Weber] and implemented to the app.

Note: Although your circuit *components* usually indeed vary according to a normal Gaussian distribution, the overall *output* measure of your usually nonlinear circuit often shows a non-normal histogram.

To test your knowledge about statistics and MC you may do a little quiz.



Statistic quiz in RealTime MC

We also include several guides to Monte-Carlo and advanced MC techniques to support you on setting up statistical analysis in true custom IC design environments. Actually there is no free lunch and if the user is aware of specific design characteristics he can usually speedup any kind of analysis! If you have run different statistical analysis several times, you usually get your own feeling quickly.

MC Guide for Advanced Techniques

Input

For pure MC we run the analysis and then do the result evaluation, maybe combined with a simple autostop on top. Advanced methods usually start with MC but then continue with other methods like optimization to get a speed-up.

For yield estimation pure random MC is not really impacted by nonlinearities or by design complexity, but convergence is slow (sqrt law). Some speed-up techniques like LDS can be applied with almost no risk, but other methods like using the Cpk are risky.

In conclusion, there is no single best method, e.g. some methods are optimized for accuracy and high yield targets, but speed-up is limited if you are below 3sigma.

Your primary constraints:

	Equivalent sigma	Equivalent loss
Equivalent true Cpk	2	6.00

☐ Distribution very close to normal

☐ Design complexity moderate (like op-amp in not too adv. technologies)

☒ Verify yield

☐ Get statistical corners

Output

Consider to use high-yield estimation via SSS. Also consider using the generalized Cpk.

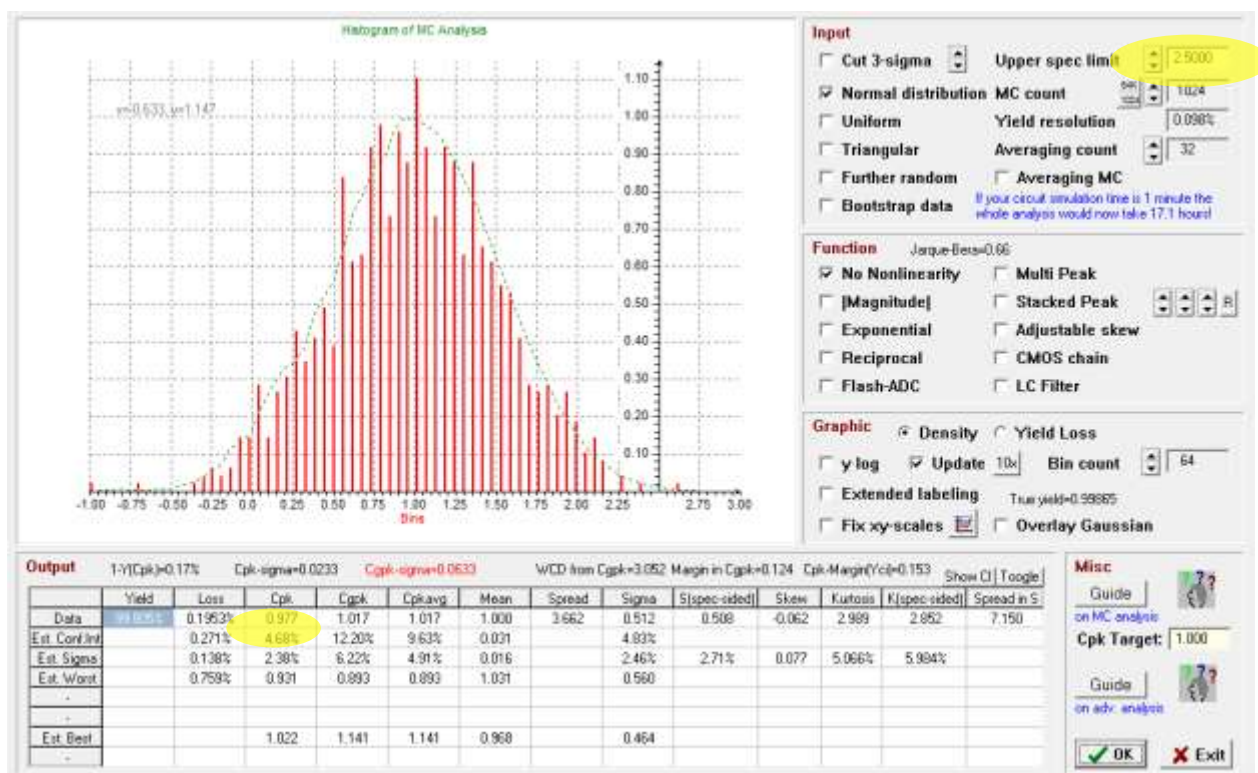
Close

RealTime guide for statistical analysis

Tutorial: 1st Steps

By default you see after startup a histogram with MC pure random data. The settings depend on MC.ini file which is saved in the Windows directory. The histogram appears with red bars as usual. On the left side a smoothed version of the data is shown (smoothing based on kernel densities KDE—like in many tools); on the right side the (green) fit from the *generalized* C_{GPK} ($= C_{PK}^*$). The y-axis is always placed at the specification limit, which is an upper limit (e.g., USL = 2.5 by default).

Comment: MC is built for single-sided upper specification limit. This is no real restriction, because for lower limits we could just multiply our data with -1, and for double-sided specs we could simple calculate two separate yield estimates and add the yield losses.

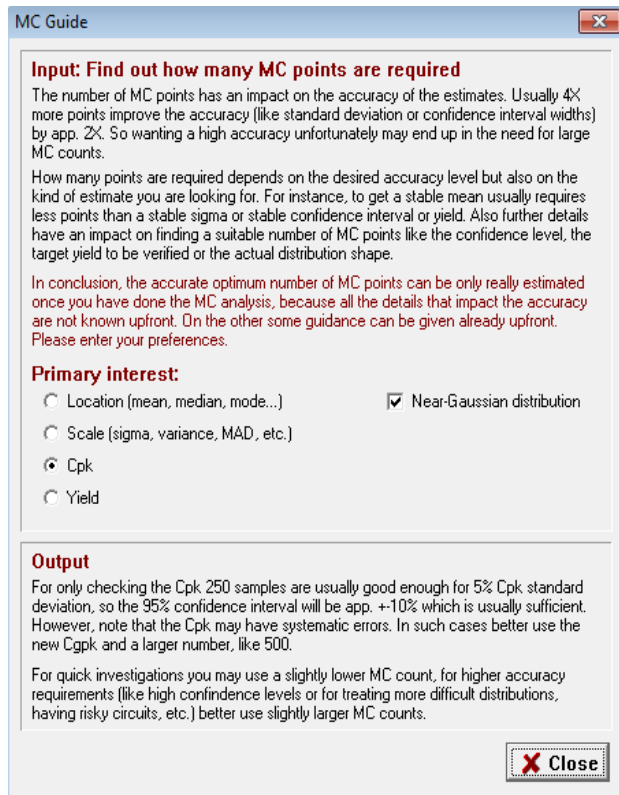


MC run from a normal Gaussian distribution

By default we calculate truly normal random data, with $\mu = 1$ and $\sigma = 0.5$. In the table below the graph you see important measures such as the sample yield Y , the process capability index C_{PK} , the sample mean (as mentioned true mean with defaults is 1.00), sample sigma (true sigma is 0.500 with defaults), etc.

Usually MC results become more stable when the MC count is increased, but not all measures require the same MC count, some are already usually quite stable for low MC counts (like the mean or median).

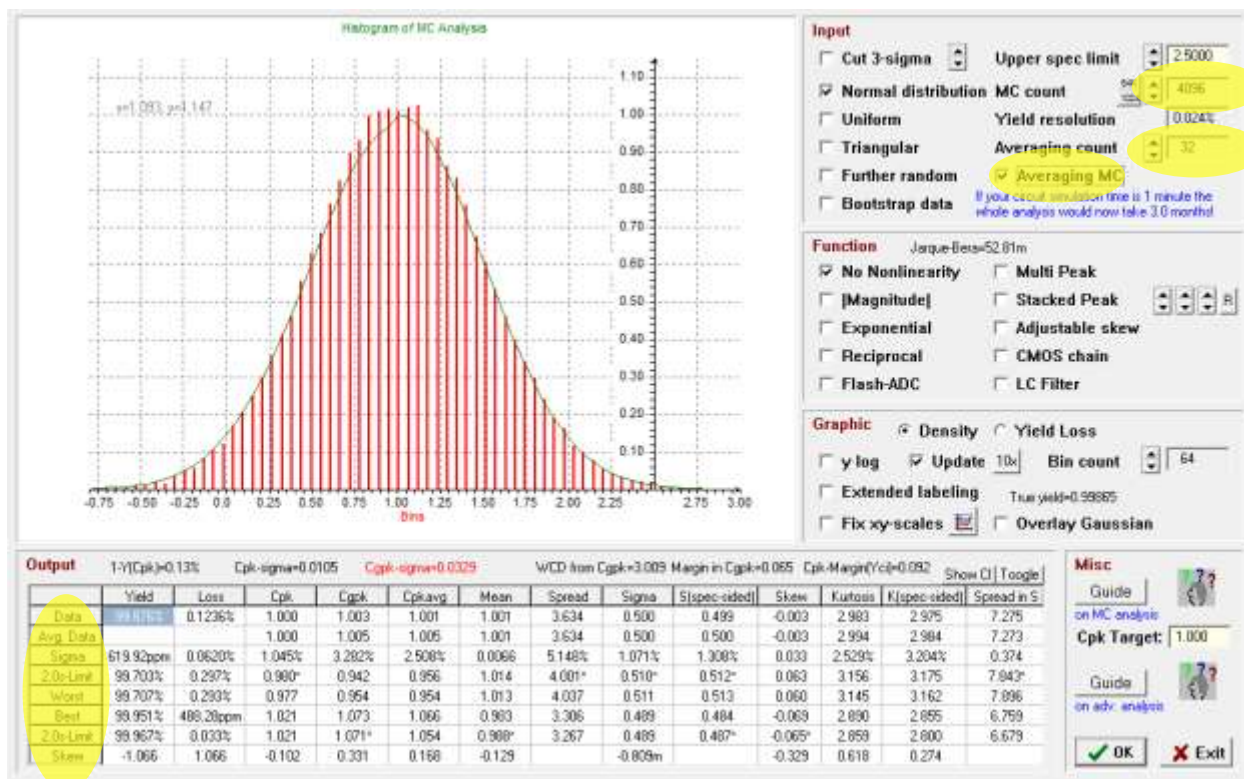
A little help window on this is available (button near the MC count entry), make your entries and you will immediately get setup hints.



MC guide to decide on the number of required points

Note: Here Near-Gaussian distribution is checked, so we could use the C_{PK} . The recommended number of MC points is 250–500—depending on yield level.

More stable values can be obtained for large MC counts or via averaging (like 32x, better use not much lower values):

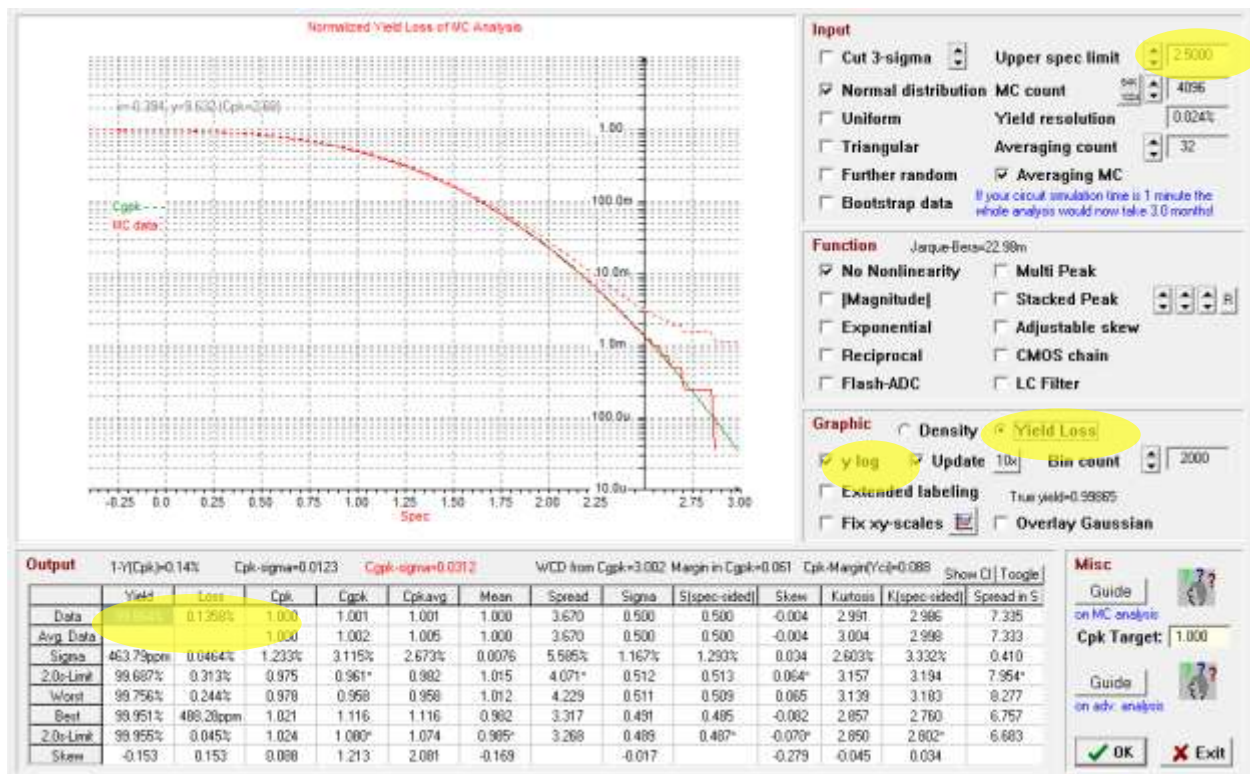


MC run with 32X averaging (normal Gaussian distribution).

The variances (across the 32 different MC runs) are now also calculated and reported in the table, including the confidence intervals (confidence level can be set in MC.ini, default is 95%—is equivalent to $\pm 2\sigma$) and some other measures, like the worst/best cases (in that MC run). For more stable values consider a larger averaging count (like 256, but better do not combine huge MC sample count with large averaging count, in such cases the calculation may take some minutes—be patient in such cases).

Another very interesting graph is the cumulated histogram, showing not the probability density but its integral—the yield—(by default log (1-Y)-axis is enabled). Click at “yield loss” radio button. In opposite to most other programs we provide a log y-axis so that the most interesting region—which is the high-yield region—can be inspected much better!

Note: To be correct not the true CDF is shown (which is usually unknown), but the so-called empirical CDF. Go for Wiki for more information.



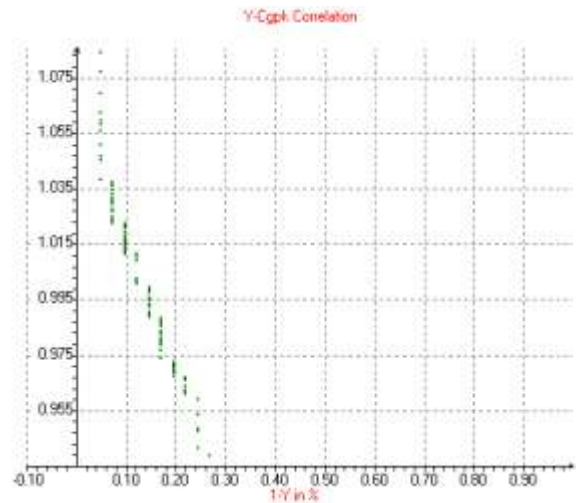
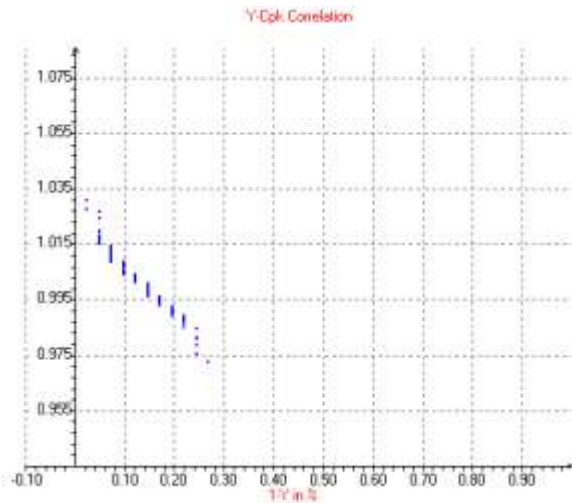
RealTime MC plot for yield (normal Gaussian distribution, averaging active))

The yield and C_{PK} depends on the specification limit USL, you can edit the upper spec limit (default is 2.5) at the top right entry field—either directly or by using the spin button aside. The spin buttons are very convenient to change the spec in small steps to find out which spec belongs to which yield (or which loos or C_{PK})! Look at the table result, when doing the spin sweep—till you reach what you want!

With no averaging the results are much less smooth and depend more on chance. Disable the averaging and also disable the “Update” checkbox. Enable “Fix xy-scales”, then press the “10X” button.



Now re-enable the “Update” checkbox and disable “Fix xy-scales”, and switch back to “Density”, also disable “y-log”. MC features also another nice plot to show how good the correlation of the C_{PK} with the sample yield is. Enable averaging and click at menu Tools—correlation Y-to- C_{PK} .



Correlation between yield loss and C_{PK} and C_{GPK} (normal distribution, $N = 4096$, averaging factor 128)

The monotonous behavior shows nicely the good behavior of the C_{PK} and the C_{GPK} , but also the problems with the sample yield, i.e., it is highly quantized. For lower MC counts maybe you will see no fail samples at all!

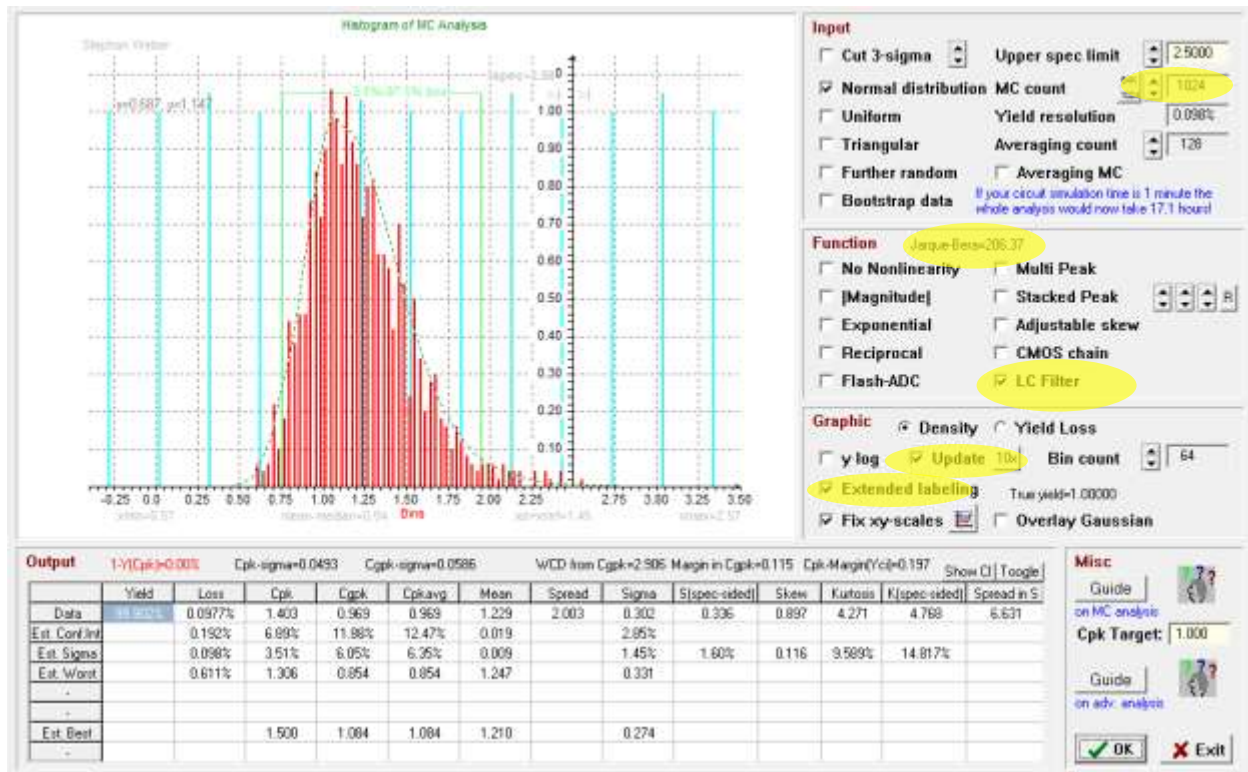
Note: Not in all cases we have such a strong correlation Y to C_{PK} . For instance, if we have an upper limit as spec, but outliers at the lower data part, then the standard deviation σ increases and the C_{PK} decreases, but the yield remains high! The C_{GPK} has no such problem! One real circuit example for this behavior is looking to PSRR, -300 dB is a good value, but quite an outlier if your mean is around -60 dB!!

Now let's play with other (non-normal) distributions, and one from a real circuit. Reduce the points to 1024 and enable "LC filter". This runs internally an 8-element LC band-pass filter circuit simulation. To be more accurate an AC simulation is done and we calculate the worst-case pass-band loss in dB; the elements itself have normal Gaussian component variations (by default). Actually the circuit is the same as the one available in RealTime Match (ANPASS.EXE).

RealTime Match is able to design matching networks of many kinds.

RealTime Match is able to design matching networks of many kinds.

As you can see the histogram is skewed (longer tail at the right side, where the spec is placed)—although the filter LC elements itself vary normal (so without skew)!



Realtime MC with simulation of a LC filter

In the table you can read out that the skew is app. 1.077 (normal PDF has no skew, so $s = 0$) and the kurtosis is app. 5.2 (normal data would give $k = 3.0$).

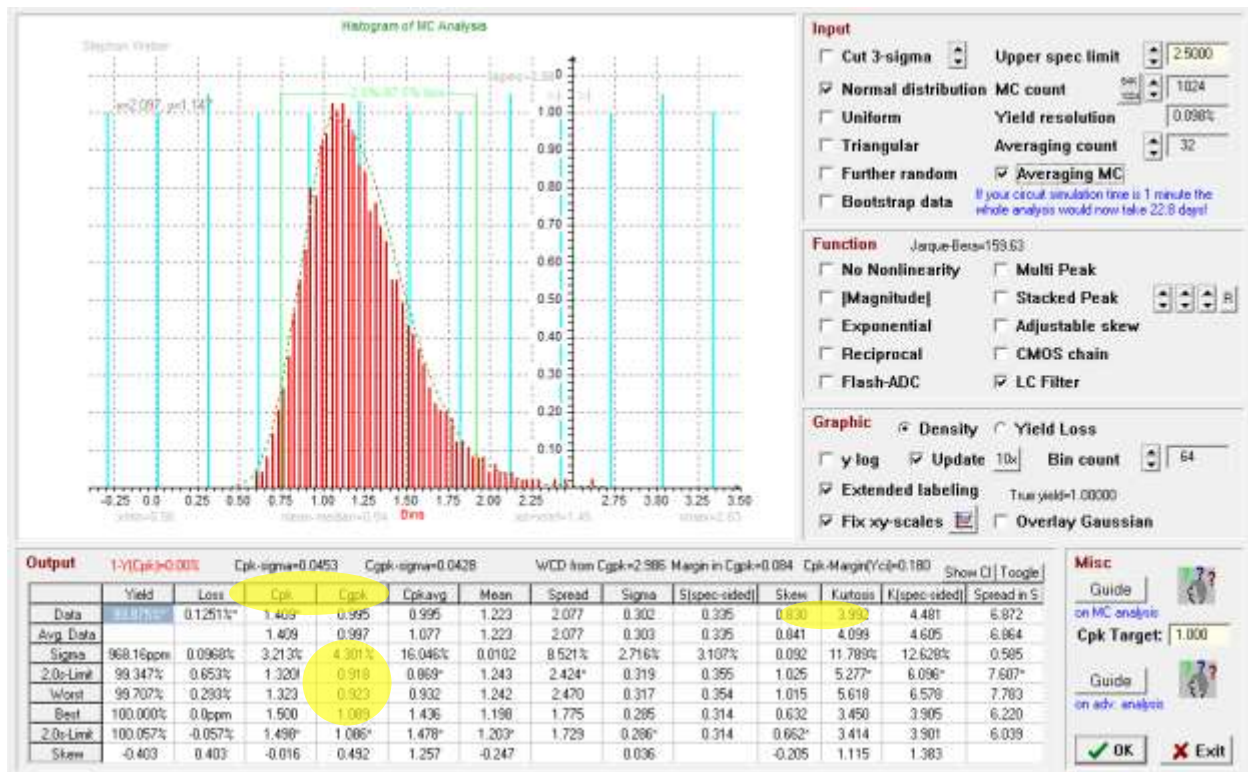
Note: Also many programs and Excel report skew and kurtosis. There is also a measure of normality called Jarque-Bera, which combines skew and kurtosis in one value. JB is also reported by MC (values larger than app. 7 indicate usually significant non-normality! Go for Wiki for more information).

Actually the histogram center samples are close to the nominal Butterworth design and we readout more or less the 1 dB-bandwidth. The few better samples are close to a Chebycev filter (with ripple of app. 0.7 dB) which is able to extend the BW at the cost of some ripple. The bad samples in the right side tail are mostly samples with asymmetric or shifted frequency response. Overall this the pass-band ripple measure creates not a certain standard probability distribution, but really “something new”, i.e., not available in any math package.

Remark: If we would the same analysis to the 1 dB-BW instead of 0.5 dB we would also find some skew, but slightly less. Mainly the 3dB-BW behaves “as expected” showing at least a near-Gaussian behavior. So in general, the same circuit can have both near-Gaussian and highly non-Gaussian behavior (of course

even more if we extend the testbench to nonlinear DC or transient simulations). Note: Skew and kurtosis can be calculated in Excel too, but Excel gives out $k-3$, so $k = 0$ for the normal case.

The C_{PK} is 1.41 but the yield is only around 0.4%, so obviously the C_{PK} is far too optimistic (error in the order of $0.5 = 1.5\sigma$)! For more stable results best enable “Average MC” to see that really a systematical error occurs for the standard C_{PK} . Now the calculation takes longer, but of course still much less than with an external circuit simulator.



MC averaging run on the LC filter

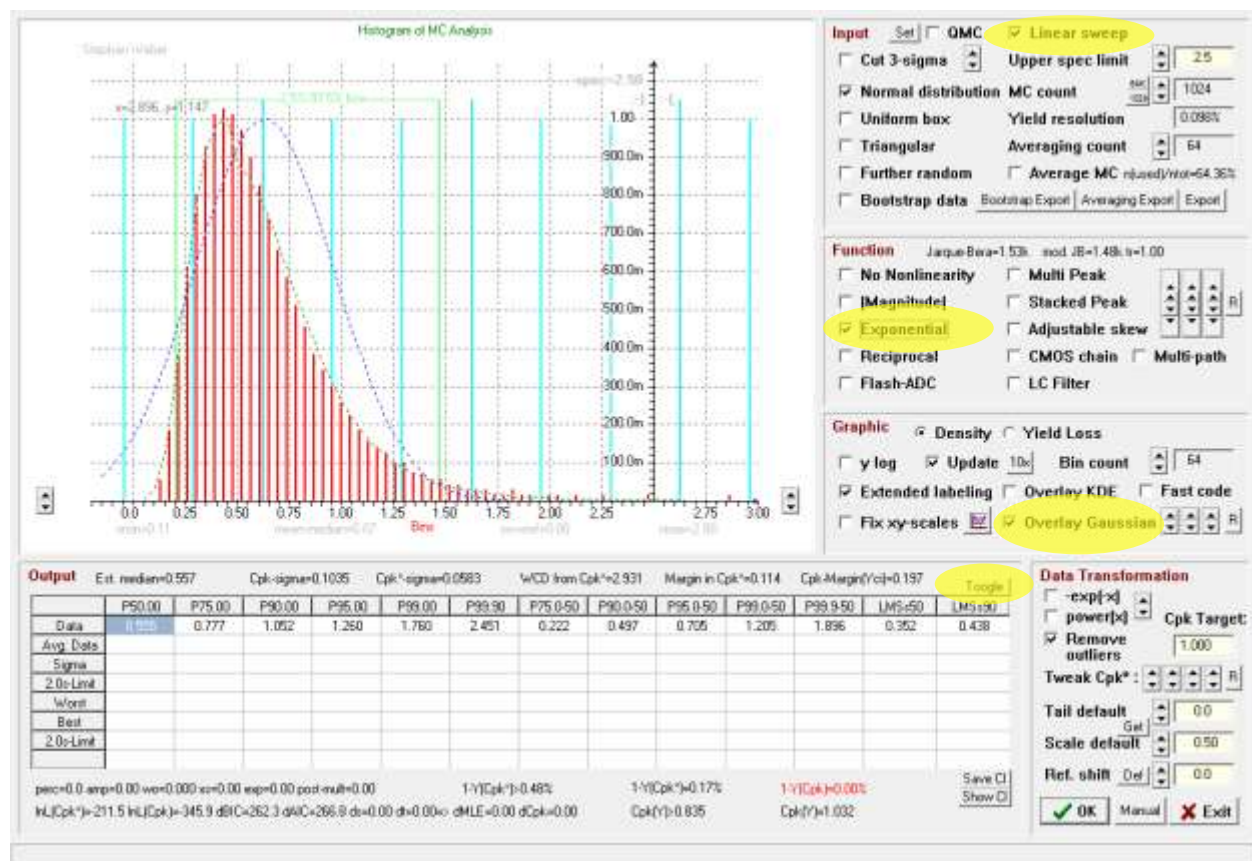
The C_{PK} is 1.41 but the “true C_{PK} ” from the (sample) yield would be only 1.024! The generalized C_{GPK} is 0.992 and thus has a much smaller systematical error and gives no over-optimistic results! Of course, on top of the systematical errors we have also *statistical* variations. In the table you get also an estimation of confidence intervals as 2σ -limits*, like 0.903 to 1.095. So to get sure by 95% confidence level (set in MC.ini) the worst-case C_{GPK} is app. 0.1 worse than the mean C_{GPK} . That should be our design margin; such margin is a statistical margin, similar to a margin you need to take into account for production measurements. This would lead to some over-design, but a margin like 0.1 in C_{PK} or 0.3σ should be feasible. If you need a small margin (like your ultra-high-performance design is already at the technology limit), we have to use a larger MC count or to improve our circuit concept (like adding a calibration, spending more power, etc.).

*Note: Behind the C_{PK} a normal Gaussian fit is used. For the C_{GPK} we use a Student's t for $k > 3$ and for $k < 3$ an approximated Irwin-Hall distribution (to model a triangular or uniform distribution). Also note that the confidence limits are not just $\text{mean} \pm 2\sigma$ but a more complex formula is used internally.

Yield and C_{PK} for More Functions & Circuits

The program has many further features, e.g.

- Get further distributions like log-normal by applying “Exponential” or doing an ADC DNL simulation, etc. (under the “Function” entry)
Note: ADC DNL depends on offset voltages, which are typically normal distributed. However, in a flash-DC the DNL is related to the maximum offset difference in the set of 255 comparators, and the maximum offset is not normal distributed!
- Overlay a Gaussian fit (as blue curve) via “Overlay Gaussian; that is the normal Gaussian fit used by the usual C_{PK} !



MC run from a log-normal distribution (non-random mode, full version)

Actually such log-normal distribution (or something at least very similar) is occurring quite frequently in circuit design, so let us do some investigations on yield and C_{PK} . You can get such distribution for leakage or other nonlinear simulations, so if such nonlinear simulation is time-consuming, you are maybe able to run only 256 points. Let's inspect this typical situation, e.g., it could happen that our design is quite good

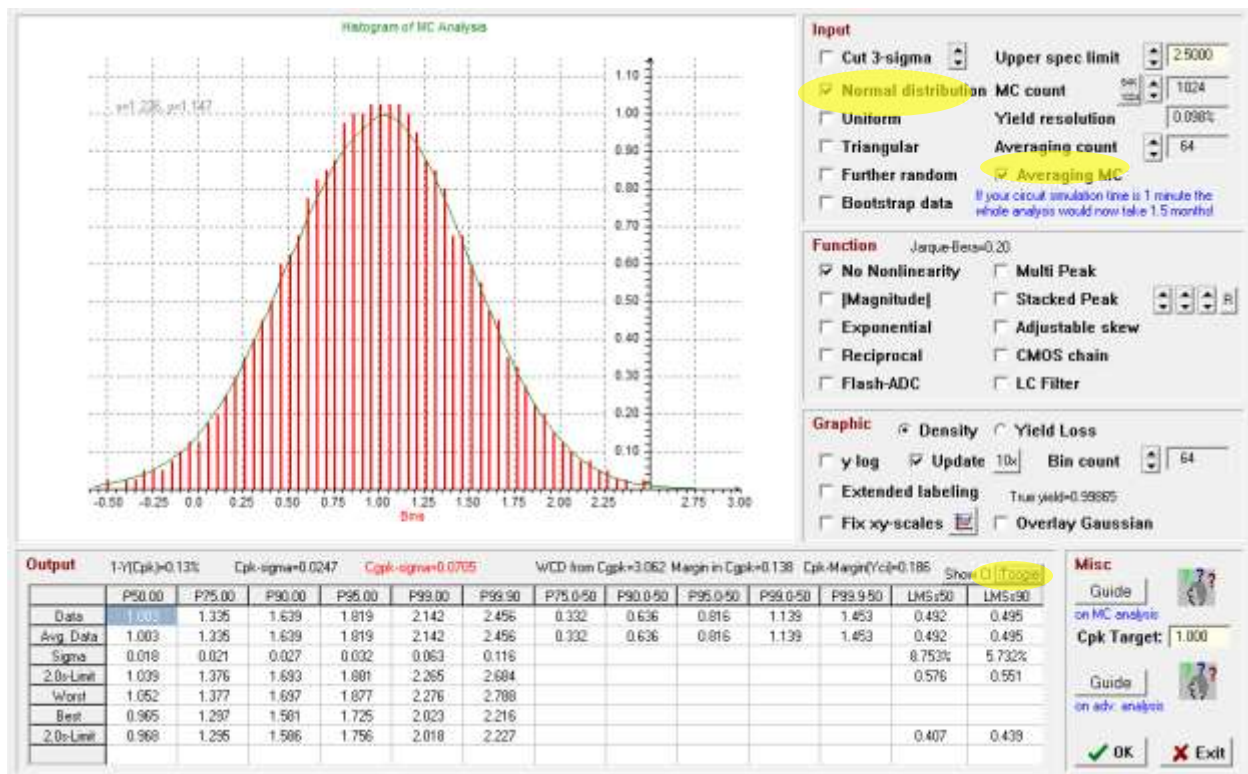
and we have no fails in our MC run. So the yield would be 100%, and the C_{PK} may look indeed more realistic like close to 1.0, which is equivalent to 99.87%—which is right? Actually NONE!
The problem is that the situation would usually only improve by either using advanced statistical techniques or spending (much) more MC points!

Further Features: Cuts and Percentiles

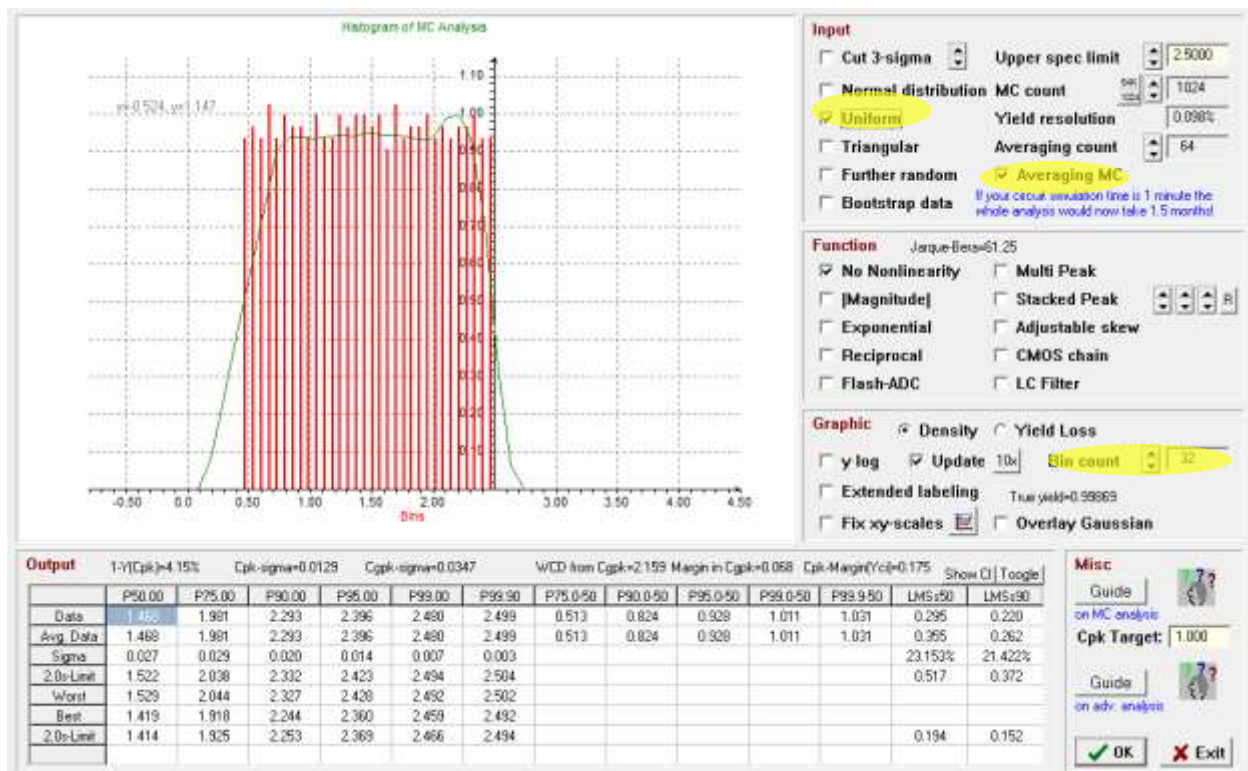
Beside yield estimations via sample yield, C_{PK} and C_{GPK} , we can make many statistical experiments. For instance, many use cuts for the process parameters, at 6σ . With MC you can inspect how the C_{PK} and C_{GPK} behave if you enable cutting. Also you can check how many points are required to get a stable sample standard deviation and C_{PK} , like $\pm 10\%$.

You can also check the central limit theorem, which tells us that if we add the samples from many statistical variables we will approach the normal distribution to a high degree—even when the original samples came from a uniform distribution! The so-called Irwin-Hall distribution can be created by summing random variables from n uniform distributions. For instance, for $n = 2$ we obtain a triangular distribution and for $n = 3$ a parabolic pdf. The IH-distribution is built-in to MC and with the spin buttons you can tweak the parameter n . This way you can interactively see how many uniform variables we need to add to get a good approximation to the normal distribution. Best check the normality in the histogram, in the normal quantile plot and by inspecting the kurtosis k .

As mentioned, some estimates do not converge in the general case, like the spread or the mid-point of a normal distribution. With MC you can also check the behavior of the sample mean or standard deviation for different distributions like a Student-t or a uniform one. Also inspect the spread or the 99% percentile p_{99} ; for a uniform it is much more stable than the median p_{50} , but for a normal distribution and many others it is vice versa. The investigations on the uniform are similar to the famous taxi count problem mentioned in the book.



MC averaging run on a normal distribution with percentile table output



MC averaging run on a uniform distribution with percentile table output

To interpret the results let us look to p50 and p99.9 in the Gaussian and in the uniform case. The variance for the percentiles is available in the 3rd row. For p50 we read out 0.018 and 0.032, respectively—so no big differences! However, for p99.9 the situation changes completely and we get 0.106 and 0.003! So the outer values are much less stable for the Gaussian case, and for the uniform we get the opposite behavior. To get a feeling for this and quick answer to many statistical questions this app has been created for.

If the distribution of a certain output performance is not Gaussian, we can still make estimations on yield. For instance you may assume another specific distribution (like log-normal) or use more general theorems like Chebyshev theorem, which makes no specific pdf shape assumptions. If we know the variance V we can estimate how many samples can be beyond a certain limit like $k\sqrt{V}$ beyond the mean. Another, usually better method, is using the generalized C_{PK} .

C_{PK} Speed-up and Errors, and the Generalized C_{PK}

As mentioned, the C_{PK} can be highly wrong for non-normal data, but for normal data it is a very good concept. Under “Help” menu you can calculate how many points you need for a certain yield verification using the counting method and the C_{PK} . The C_{PK} needs much less points, so it gives the user a speed-up for verifications.

Note: Also a plot for C_{PK} speed-up is available.

Here is an example, for a normal distribution: We want 1.0 for C_{PK} and can allow a margin of 0.25. Use the spin button to find the toggle point where the counting method label gets from red to green! This happens at app. 2650 points! So the counting method needs 2650 samples, but the C_{PK} needs 45 times less points (like 60 vs. 2650)!

It is well-known that Monte-Carlo simulation needs a lot of samples to offer stable results, e.g. on yield, mean and sigma. Therefore this program has been created to do MC investigations quickly.

Real random processes have the tendency to generate near-Gaussian data, due to central-limit theorem. However, if random effects do not add up, but using more complex formulas, then non-Gaussian distributions will be generated.

Beside the yield itself often C_{pk} is used as a monitor. However C_{pk} to yield correlation is not accurate for non-Gaussian data. At $C_{pk}=1$ an error of 0.1 in C_{pk} means a yield loss error of app. 2.5%.

Mean varies among MC runs. An estimation is $\sigma(\text{mean}) = \sigma / \sqrt{N}$. Other data like sigma, skew or kurtosis is less stable. To get $\pm 3\sigma$ samples you typ. need 256 MC samples (if kurtosis is not too small).

If you run a MC with 128 samples and you want to guarantee 0.13% yield (or $C_{pk}=1$), then you need a margin of app. 0.3 C_{pk} (or 0.9 sigma) for your design. If the distribution is highly skewed and having large kurtosis, you need either more samples or more margin! For low kurtosis or out-Gaussian you would need less sample and/or margin.

Few hints:
This program and C_{pk}^* definition is optimized for yield from app. 50-99.997% because this range is most interesting for MC application and real designs.
Skew give information how asymmetric the distribution is. Kurtosis describes if a distribution has a long tail or not. For Gaussian kurtosis is 3. If yield or C_{pk} data is not very accurate, the values in the table will be indicated with "****".
 C_{pk}^* is an improved definition which takes skew, kurtosis and number of samples into account. Another criteria is the worst-case distance $WCD = C_{pk}^* \cdot 3$.
By default the confidence level is 95%. This is two-sided, i.e. for the minimum C_{pk} it is 2.5% (which gives some extra margin).

Graph hints:
For the yield graph an errors are reported. The C_{pk} validity is defined so that the rms error is smaller than 0.05. The errors are taken up to the last sample or up to the target $C_{pk} + C_{pk}$ margin (which ever is larger wrt yield).

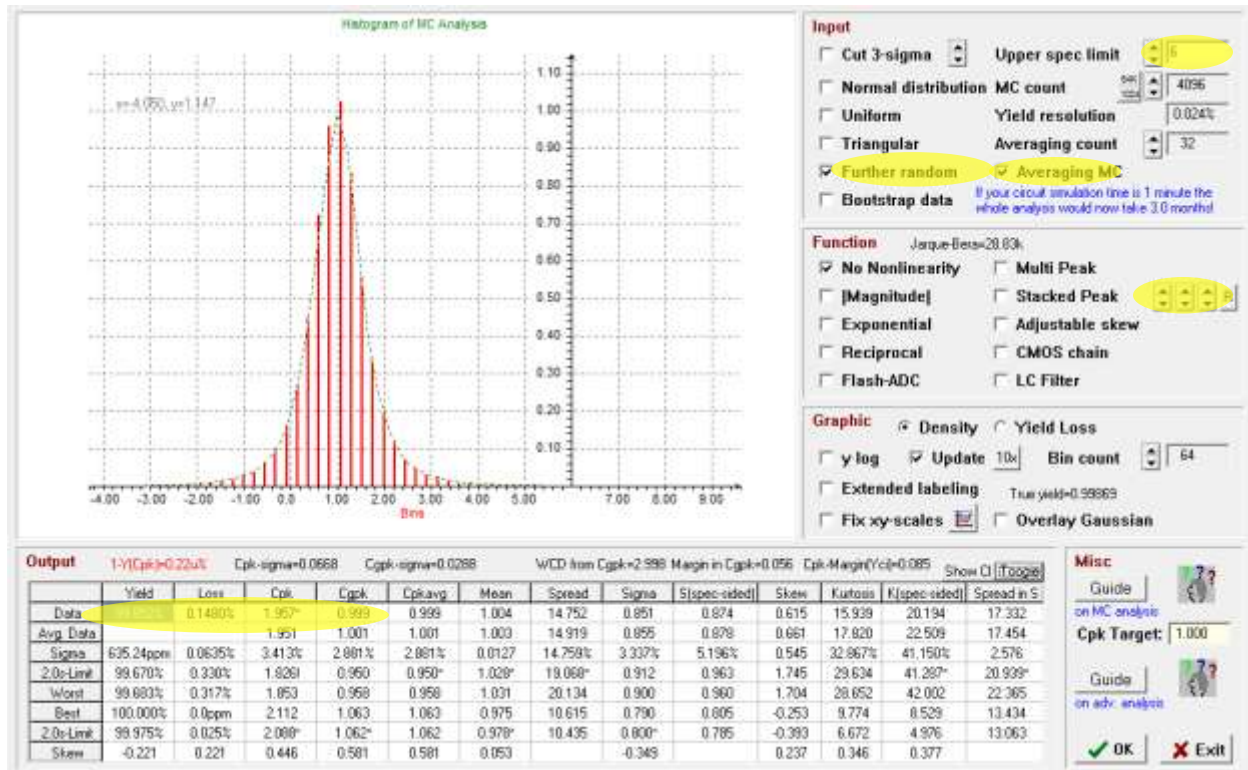
Cpk-Speed-up Calculation

Target yield loss in sigma	3.0
equivalent Cpk	1.000
equivalent loss in %	0.1350
equivalent loss in ppm	1349.90
Acc. design margin in Cpk	0.25
MC sample count	2657
Needed yield in sigma	3.75
This should be the result of your MC run	
equivalent Cpk	1.25
equivalent loss in ppm	88.42
equivalent loss in %	0.0088
Counting method: sigmaCpk=0.123	
95% confidence limit YL in %	0.130009
equivalent Cpk	1.004
Distance method: sigmaCpk=0.018	
95% confidence limit YL in %	0.013630
equivalent Cpk	1.213
Cpk sigma scaling factor	1.0
Systematic Cpk error	0
Speed-up distance vs counting: 45.12	
Accuracy improvement : 6.717	

C_{PK} speed-up calculation in RealTime MC

The only pity is that the C_{PK} can be so wrong that this speed-up disappears. If we have a systematic error of 0.2 (enter at lowest field), the speed-up would disappear! Remember: In our LC-filter test case the error was already 0.5! Therefore, the generalized C_{GPK} has been created, having much lower systematical error (but a bit larger variance)!

There are even some cases, where the C_{PK} simply does not converge at all. To get an example click at “Further random” and click to the 3rd lower function spin button ten times. This selects the Student-t distribution. By default, degree of freedom is set to a large value; so the Student-t is close to the normal distribution, but now also click 7x the left upper function spin button (this lowers the degrees of freedom). Set the spec e.g. at 6 (to have again a true C_{PK} close to 1.0) and N to 4K. Also enable averaging.

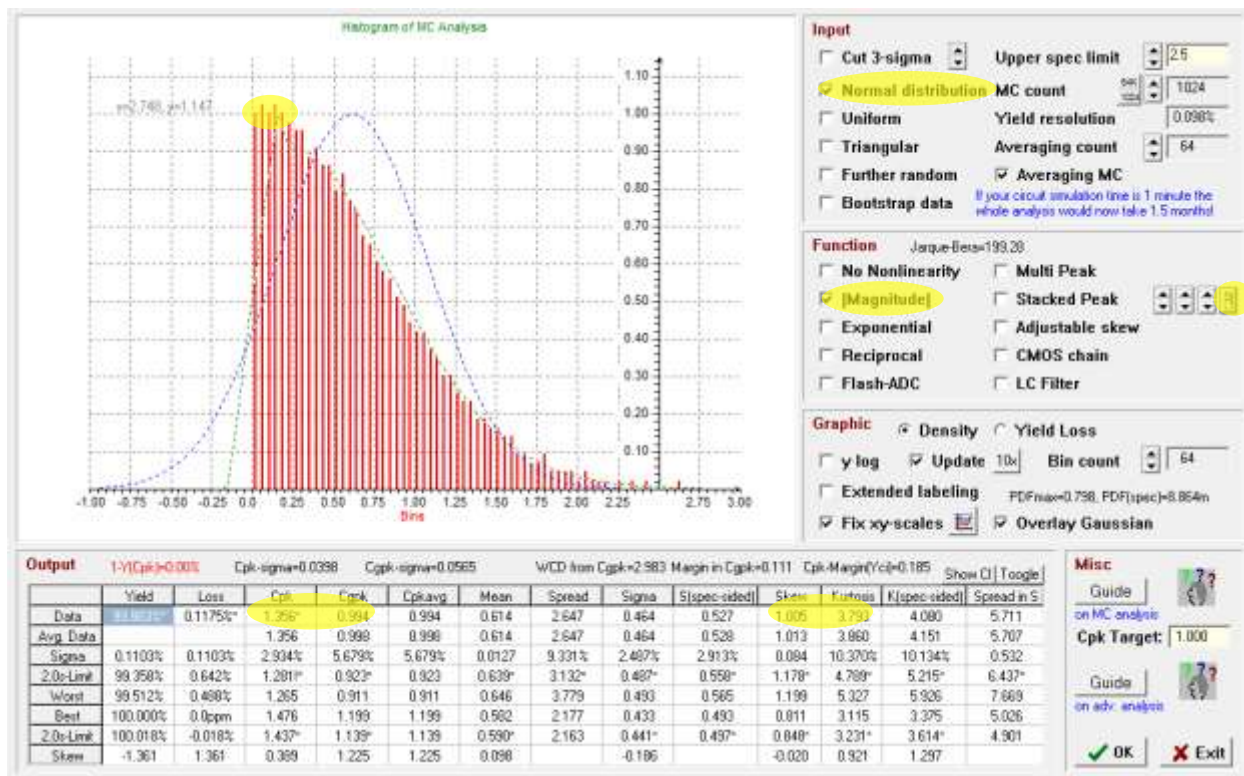


MC run on a Student-t distribution (infinite kurtosis)

The distribution looks still near-Gaussian, but the C_{PK} variance is 12.7%, but the C_{GPK} is more stable, having 2.4%! Also we have again a significant C_{PK} systematical error, in the order of 0.2. If we set now the MC count to 64K (takes some minutes!), we get still a C_{PK} sigma of 5.6%, so the sigma went down by 2X only although we spent 16X more points.

The reason for this bad C_{PK} behavior is the large sample kurtosis of >50. Also the linear histogram gives a bad impression, to really see the problem switch to “y log” axis. This way you can see the long tails of the Student-t distribution.

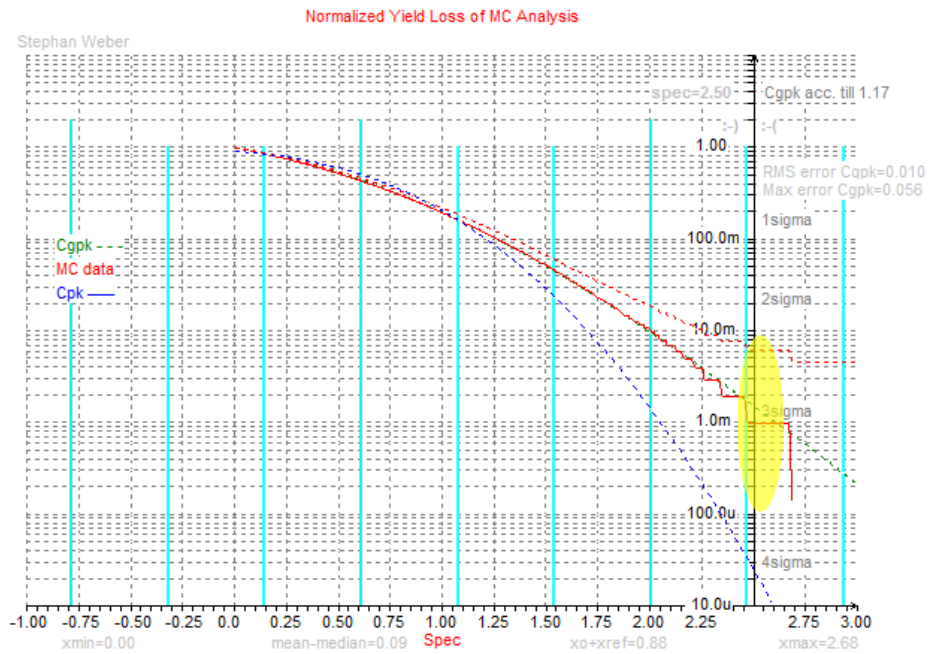
Another good example for the better behavior of the newer C_{GPK} (dashed green) is a folded Gaussian distribution. Set N to 1K and go back to a normal distribution, then enable “Magnitude”. Also set the spec to back 2.5. Also press the “R” button to reset the spin button settings and overlay the (blue) Gaussian C_{PK} fit again.



MC run from a half-normal distribution (averaging active)

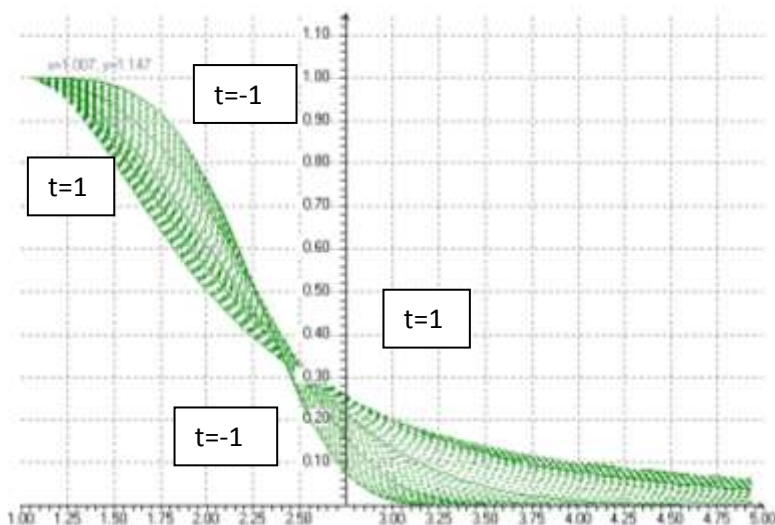
The shape is here really nicely normal, but we have only 50% of the normal distribution, like if we would have a spec like $|V_{offset}| < 2.5$ mV. The C_{PK} makes the blue fit, which is obviously a bad one. The C_{GPK} fit starts from the mode (peak) and uses not the mean as reference location, so the systematic error is much smaller, like 0.01 compared to 0.35 for the C_{PK} .

In many cases, the C_{PK} is too optimistic, which is of course dangerous for design decisions. This can be best seen again in the (logarithmic) yield loss plot (giving $-x^2$ dependence for the blue line).



Yield loss plot to show how wrong the C_{PK} (blue) extrapolates

In opposite the dashed green curve for the C_{GPK} and the full red line for the true data are very close to each other (if we use averaging and large MC counts).



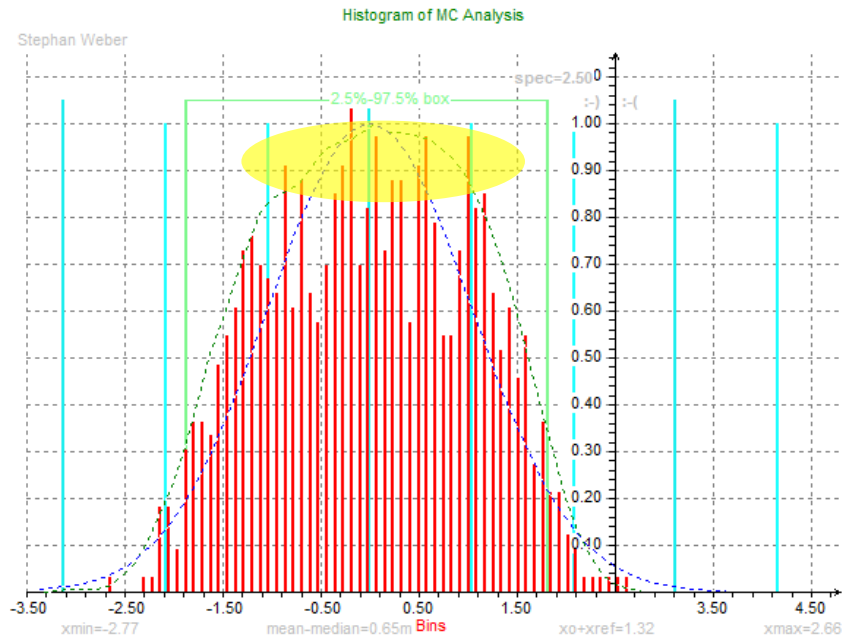
The C_{GPK} model with tail swept from -1 to 1 (in steps of 0.05, center and scale set to unity)

Note: $t = 1$ gives the Cauchy, $t = 0$ the normal distribution, $t = -1$ a good polynomial-style approximation to a trapezoidal distribution

The upper dashed red curve is the sample yield confidence interval, which gives unfortunately much smaller sigma values (so being very pessimistic).

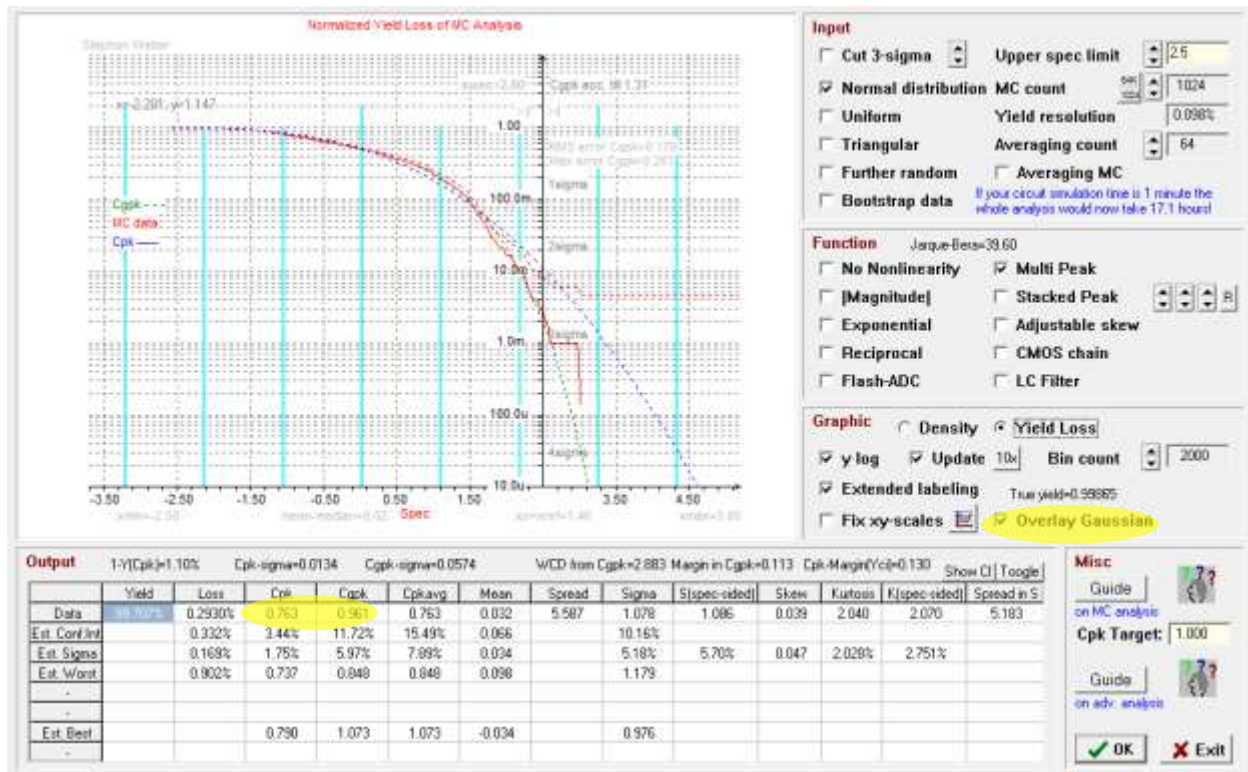
Of course there are also many other difficult cases, and sometimes the old C_{PK} is too pessimistic. Switch to “Multi Peak”. For best speed disable the averaging and enable QMC.

Note: The bin count is set to a quite large value, because using a too small value (like 10) would smooth out the three peaks too much! Actually the optimum bin count is depending on distribution type and MC count!



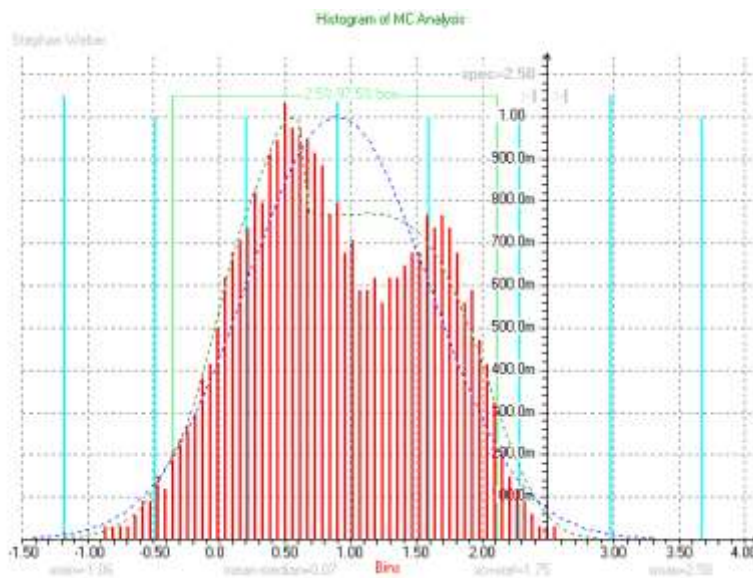
Histogram of a tri-model Gaussian mix and the normal Gaussian fit (blue)

Again the blue C_{PK} fit is worse; and to see what does it mean in terms of yield and C_{PK} best switch to the yield plot:



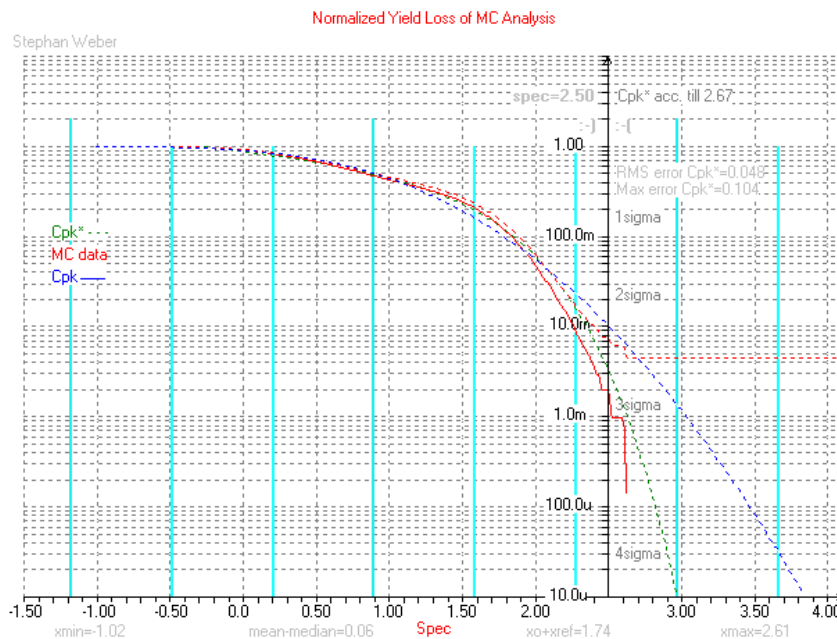
Yield plot on the Gaussian mix (C_{GPK} fit—green—is reasonable).

Now the C_{PK} gives only 0.782 but the more correct value is 1.076 obtained from C_{GPK} . There also difficult case for the C_{GPK} , like this (click the right function down spin button).



Bi-model example and fit by C_{PK} and C_{GPK}

Now both fits are far from perfect, but to see which one is better switch to the yield graph.

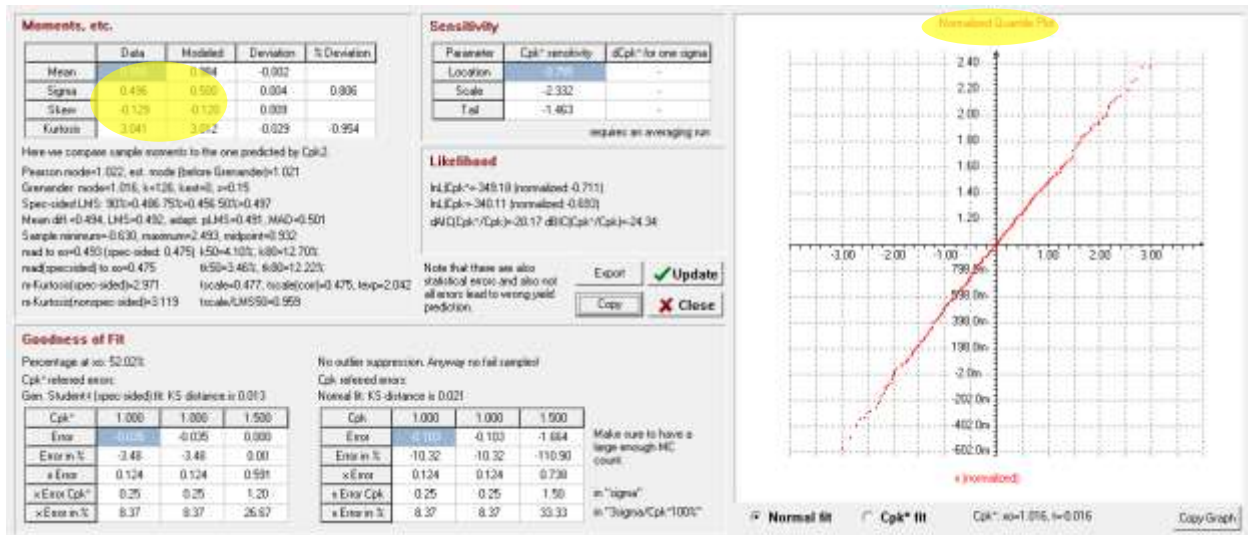


Yield plot for the difficult bimodal example

Again the winner is the C_{GPK} ; only for pure normal data indeed the C_{PK} is better; both have zero systematical errors, but the C_{PK} has app. 2.5X less standard deviation in this case (but in many other

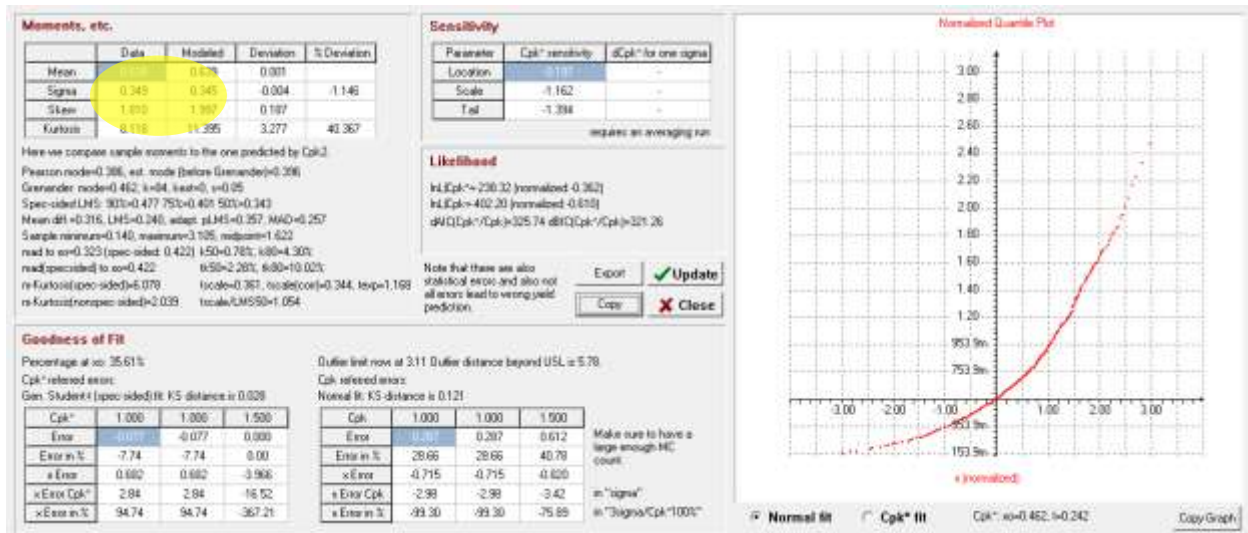
cases they are head to head, or the C_{PK} can be also worse, like for high kurtosis cases, remember the Student-t example).

A good plot to check if a distribution is normal (or not) is the quantile plot; so switch back to a pure normal distribution and bring it up via menu “Info” (under Linux the entry is under the Tools menu, and the quantile plot is in the main menu).



Normal quantile plot and further statistical parameters in RealTime MC (full version)

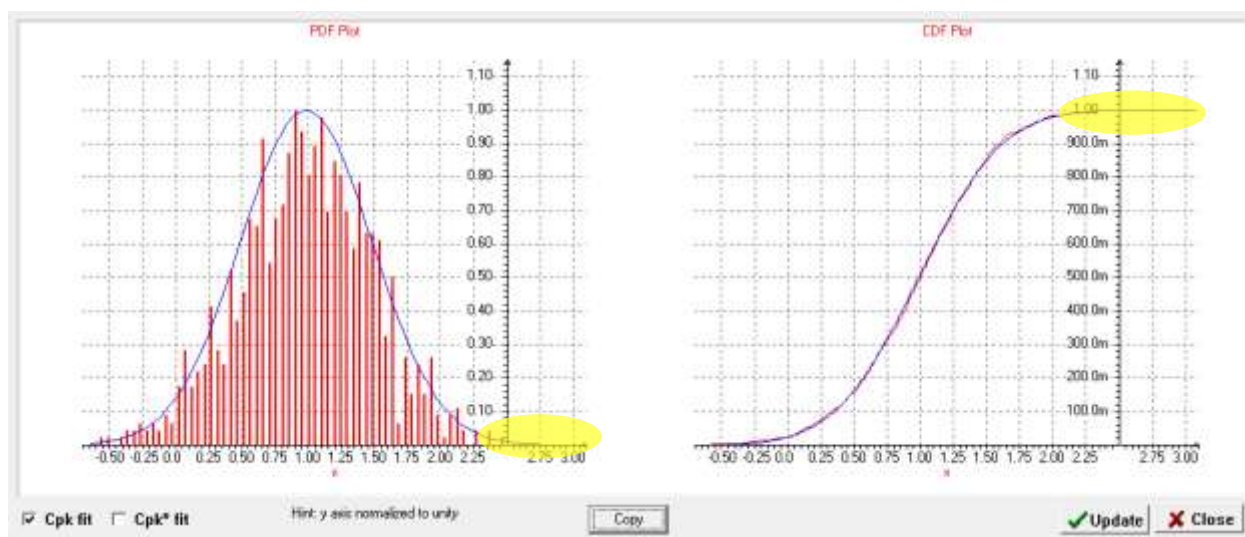
The graph shows a straight line, and the skew is small and kurtosis close to 3.0. Now let us switch to a non-normal plot, like enabling “Exponential” (giving log-normal data).



Log-normal data in normal quantile plot

Now we have non-normal skew and kurtosis and also the quantile plot is distorted!

By default you get in MC the histogram plot and a logarithmic yield loss plot. You can also get the more usual ones under menu “Tools –Histogram, PDF, CDF”.



The usual (linear) PDF and CDF plots in RealTime MC

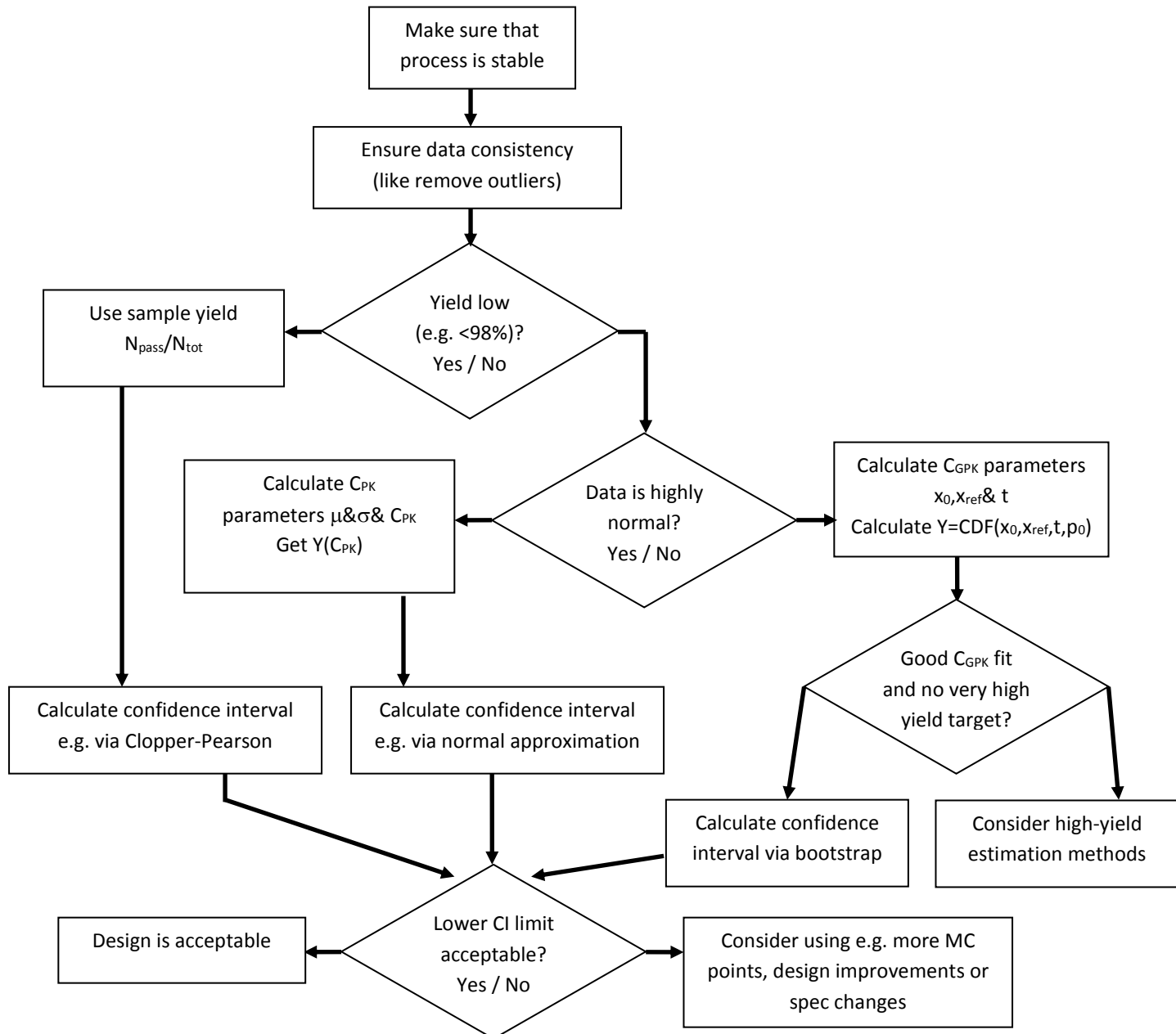
As you can see it is unfortunately hard to inspect the high-yield region in these linear graphs. Also the CDF plot looks often quite smooth, and the user may expect very stable value, but that is actually not true for the high-yield region and moderate MC counts (like 256 or even 1024).

Further RealTime Sweeping Applications

Many input fields come with spin buttons aside; if you use it for the spec limit, you can watch in parallel to any output you want, e.g., the C_{PK} or yield in the table, and check how it changes during the spec tweaks! This way you can find which spec value is equivalent to a certain yield (or yield CI limit)—even for non-normal data and in real-time! Or you may use the spin for the bin entry and look to the histogram till you feel the histogram looks meaningful, the same you can do for many other entries like the number of MC points, e.g., sweep it till you think the confidence interval is tight enough or till histogram looks stable enough.

C_{GPK} Flow for Yield Estimation

This flow chart is also available from the MC program.

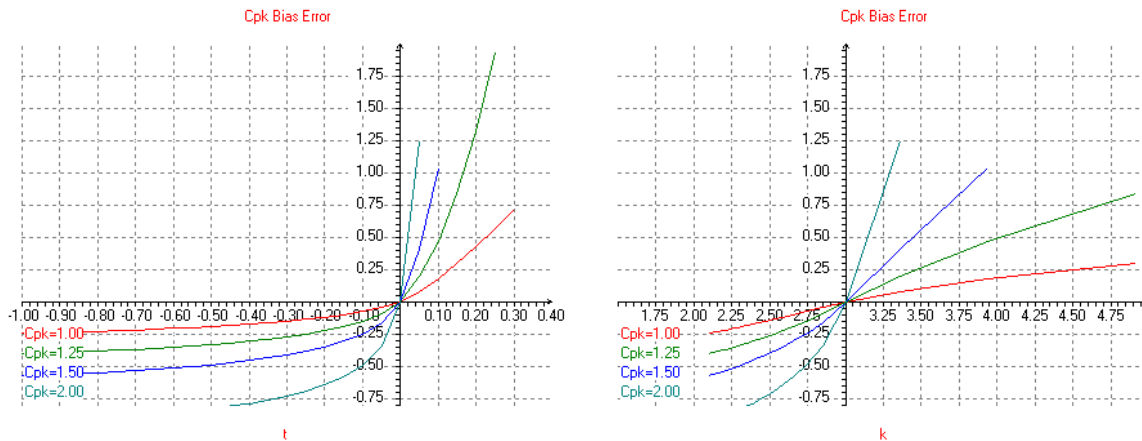


Bias Error in C_{PK} and C_{GPK}

You can also get a plot of the systematical C_{PK} errors vs tail parameter or versus kurtosis for symmetrical cases:

The C_{PK} bias error (and the minimum design margin) is already beyond 0.45 (1.35σ) even for moderate t values (like $t = 0.1$ giving kurtosis still below 4) and target yields (like true $C_{PK} = 1.25$).

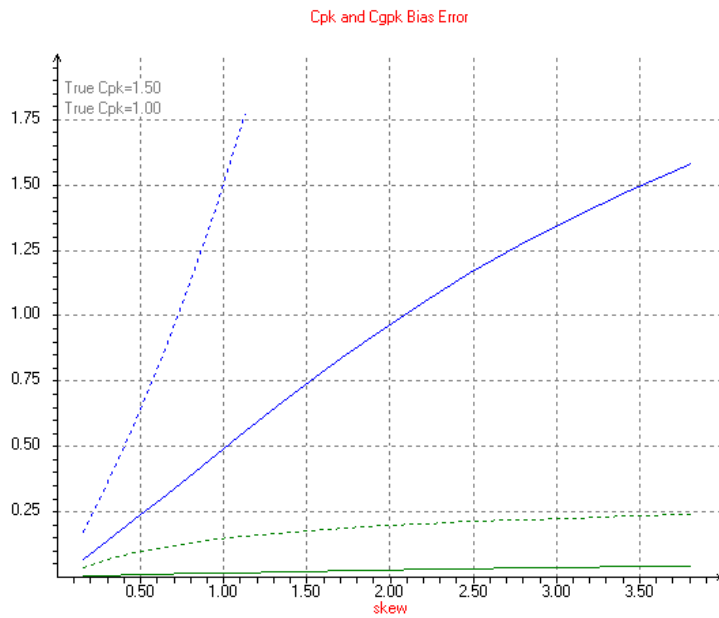
C_{PK} bias error vs t and k for different true C_{PK} s (for large t the C_{PK} becomes inconsistent):



Typical bias error of the C_{PK}

This clearly shows that even quite moderate deviations to the normal distribution can lead to significant C_{PK} errors, whereas the C_{GPK} is much more accurate.

For asymmetrical both the C_{PK} and the C_{GPK} have some bias, but again the C_{GPK} behaves much better and reduces the error by 10X in average:



Typical bias error of the C_{PK} and C_{GPK} for log-normal data

Also in opposite to the C_{PK} , the C_{GPK} is not too optimistic.

The menu "Xo" displays a window with sweep on the reference location parameter (Windows only, not Linux). That can be of interest for difficult distributions like the folded normal distribution or the multimodal distributions.

RealTime MC as Guide for Advanced Statistical Techniques

As mentioned, RealTime MC can guide the user to get his own gut feeling and trust to statistics and MC. We also added helping pages to guide the user for general MC simulation and for advanced methods, like C_{GPK} or special high-yield estimation methods.

The MC guide #1 for MC is asking for your intents for the analysis, like you want to check the standard deviation or the sample yield. Then it immediately gives you setup hints for your MC analysis, e.g., how many numbers are required by the different methods.

The MC guide #2 is doing something very similar but for the more advanced statistical techniques like statistical corners or high-yield verification. It provides you setup hints for your MC analysis, e.g., which technique is well-suited—often it is not only one, so you can run both and double-check.

MC Guide for Advanced Techniques

Input

For pure MC we run the analysis and then do the result evaluation, maybe combined with a simple autostop on top. Advanced methods usually start with MC but then continue with other methods like optimization to get a speed-up.

For yield estimation pure random MC is not really impacted by nonlinearities or by design complexity, but convergence is slow (sqrt law). Some speed-up techniques like LDS can be applied with almost no risk, but other methods like using the Cpk are risky.

In conclusion, there is no single best method, e.g. some methods are optimized for accuracy and high yield targets, but speed-up is limited if you are below 3sigma.

Your primary constraints:

Equivalent true Cpk: Equivalent sigma: Equivalent loss:

☒ Distribution very close to normal

☒ Design complexity moderate (like op-amp in not too adv. technologies)

☒ Verify yield I

☐ Get statistical corners (good for debugging and circuit improvements)

Output

Consider to use high-yield estimation via WCD or sigma-scaled sampling SSS. Also consider using the Cpk. Look-up your problem is difficult! Note that the number of required simulations is usually lower in advanced methods compared to sample yield, but for concrete numbers best look to tool recommendations.

X Close

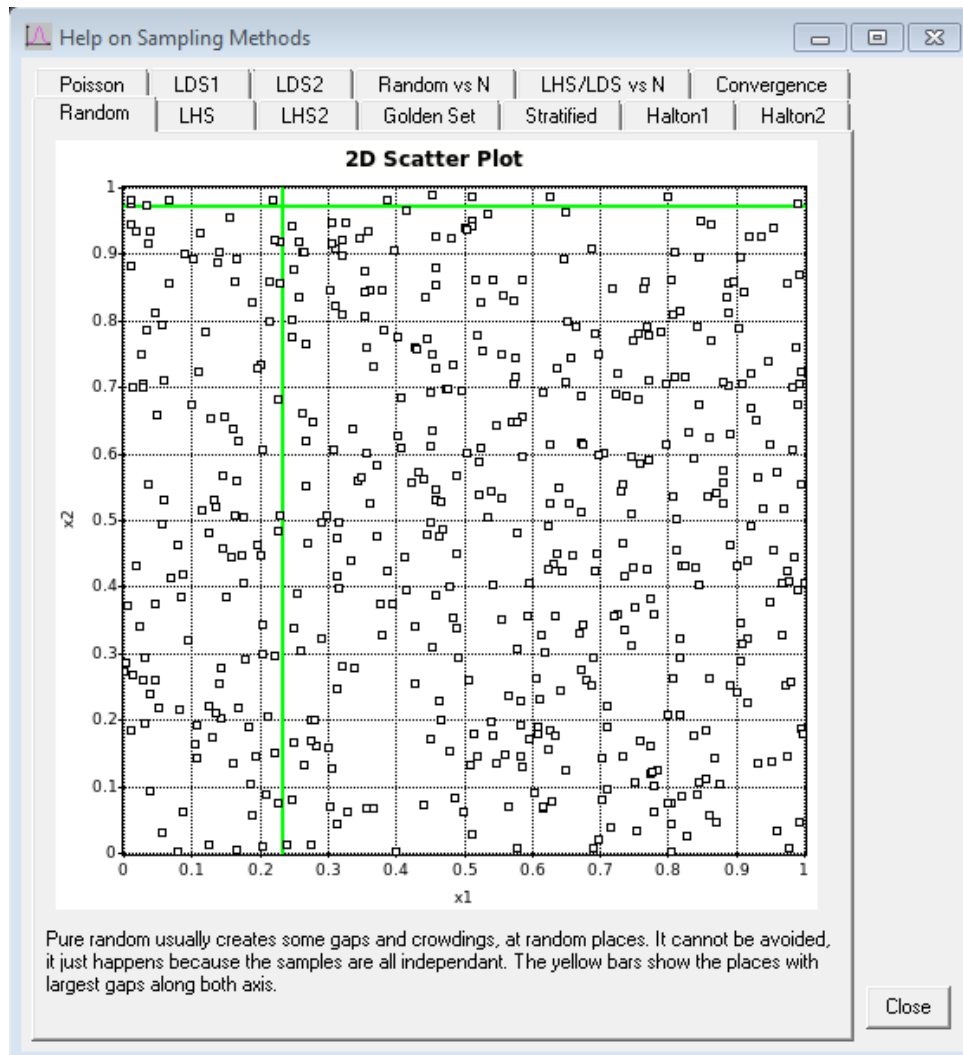
Guide for advanced techniques

Here the user select that the distribution is close to normal and design complexity is high, so we can use the C_{PK} method. Also SSS and WCD are well-suited, but you can expect longer runtimes because of problem complexity. If you switch to “Get statistical corners”, the C_{PK} method cannot be used.

RealTime MC Hints for Sampling Techniques

Within the app pure so-called pseudo-random numbers are used, but MC can be speed-up by using advanced sampling schemes like Latin hypercube or low-discrepancy sampling. In one dimension this is trivial, just like integration by rectangular rule ($1/n$ convergence) vs pure random ($1/\sqrt{n}$ speed only).

At the help menu you can find further hints, and many examples of advanced point sets in two dimensions. Actually there is no single best method in general, but usually designers have some benefits when switching from random to LDS.



The reason is that interestingly pure random does not lead to a coverage of the statistical space; you will almost always observe point crowds and spaces with gaps! Random sampling is independent, so nothing preserves us unfortunately for running similar points again. On the other hand the theoretical speed-up to $1/n$ convergence is only possible in less complex cases, and in real difficult circuits it is maybe 2x to 5x.

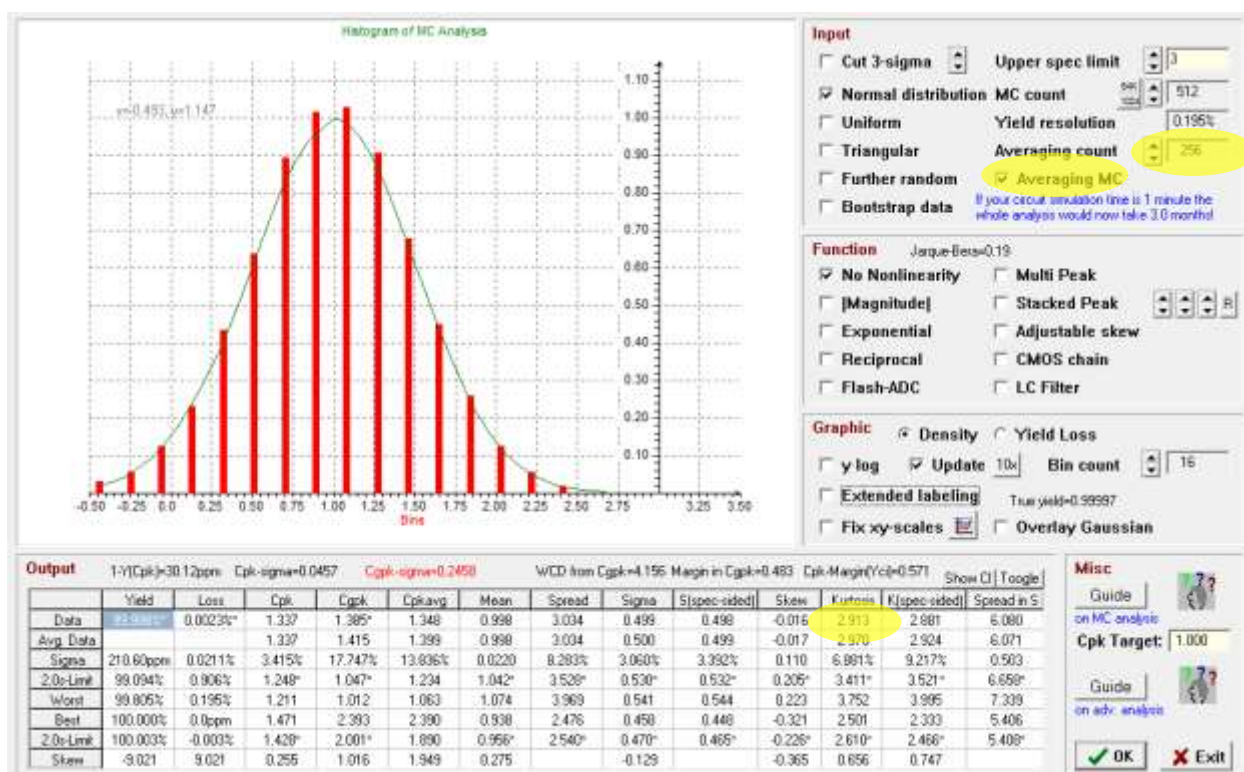
Bootstrap in RealTime MC

We mentioned that confidence intervals calculated from MC data directly—if the distribution type is known (like normal or log-normal) or in the general case (like coming from real circuits) by doing a repeated MC analysis. The 1st method is limited and risky for unknown data, and the 2nd method is extremely time-consuming.

Bootstrap is a method which is quite fast and also applicable to non-normal data and difficult estimates—and it is built-in to RealTime Match. Bootstrap is also used intensively in modern design environments like ADE GXL (for confidence interval calculations, internal error checking, to check for convergence, etc.)

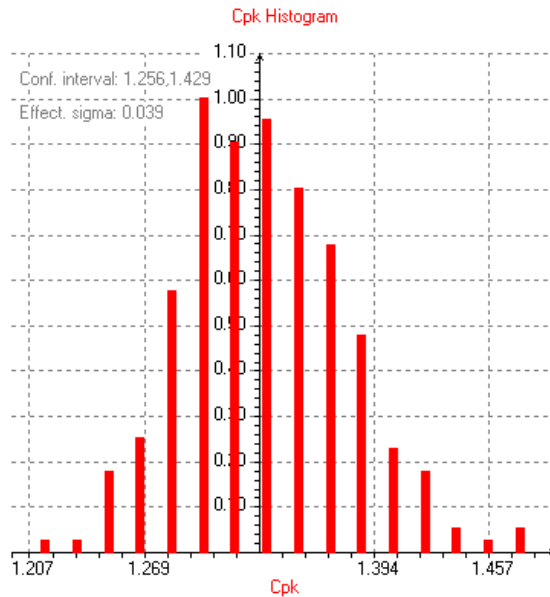
Bootstrap for Normal Data and C_{PK}

Standard bootstrap uses plain “re-sampling with replacement” and is not very accurate for long-tail distributions. Therefore, different options are available in the Options menu to get more accurate confidence intervals from bootstrap. To get a feeling for bootstrap let us first run a true repeated MC analysis with pure normal Gaussian data. It will result in a nice histogram with a kurtosis close to 3.0.



Histogram from repeated MC run with kurtosis close to 3.0

In each of the repeated MC runs we can also calculate the C_{PK} , so that we can also get a distribution for the C_{PK} ! To get this click at “Tools - Histogram for C_{PK} ”.

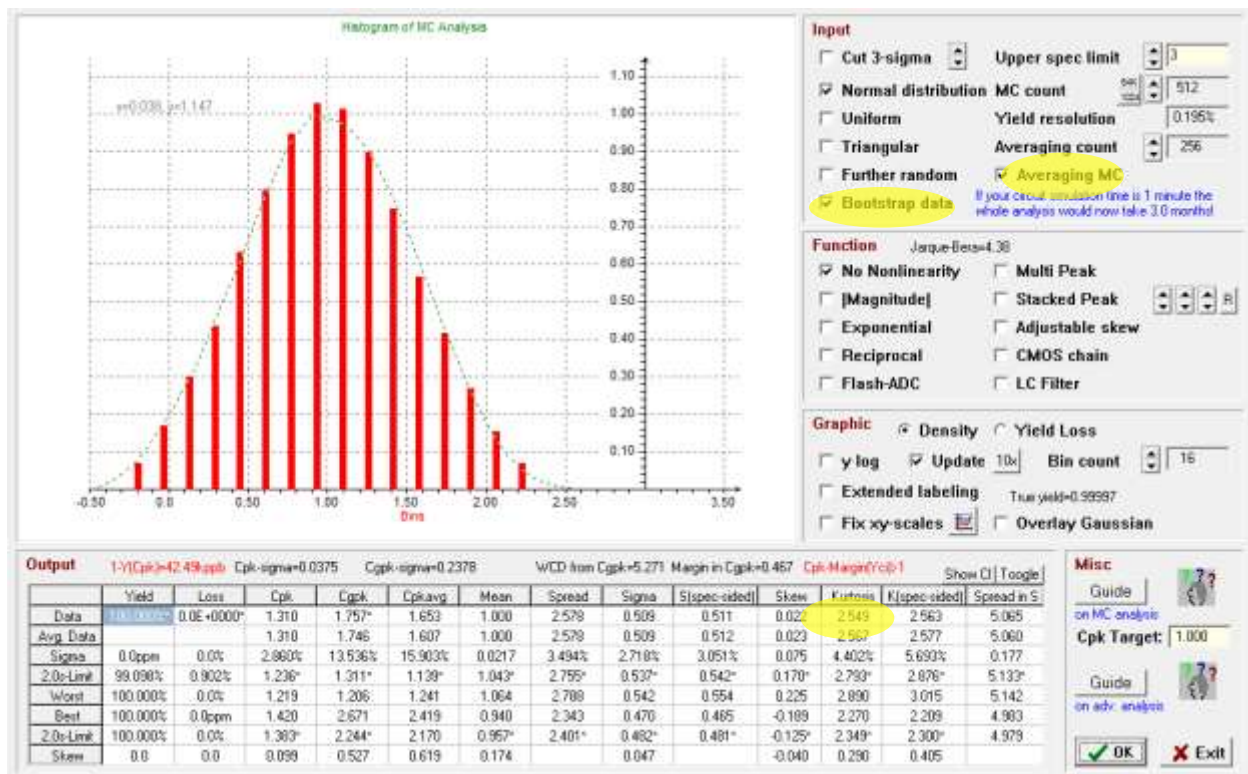


Histogram for the C_{PK} distribution from repeated MC

As you can see also the C_{PK} histogram is an almost normal distribution, and we could readout the 95% confidence interval as the part including 95% of the center samples or by using the $\pm 2\sigma$ rule!

Now enable the “Bootstrap” checkbox. Internally we do now no new fresh MC analysis anymore, but simply re-use the result from the repeated MC run which we made before! The result is a histogram which looks very similar, and only the kurtosis is slightly reduced, e.g., to app. 2.8. Again we can inspect the C_{PK} histogram and can again calculate a C_{PK} confidence interval. In RealTime Match both truly repeated MC and bootstrap have similar speed, but with a true simulator the repeated MC run would maybe take days, whereas the bootstrap takes less than a minute—although the results from both are quite similar!

This is the promise and value of bootstrap! But now let us also inspect the bootstrap results in more detail. We mention the kurtosis is slightly inaccurate in bootstrap, so if you bootstrap again and again the kurtosis will decrease further, and the data becomes more and more short-tailed! This is because by default RealTime Match is doing the next bootstrap from the previous bootstrap data, so that we can see an accumulation of the bootstrap error.



Histogram from repeated bootstrapping with kurtosis close to 2.5

Luckily, the kurtosis error in normal bootstrap would not add up again and again, because we would do the bootstrap only from the original data. In addition, there are some methods making the bootstrap more accurate. For instance, we can add noise to the bootstrap samples to get also sample values which are not part of the original MC data set. This makes also the variance slightly larger (and more correct), and will also increase the tails a bit.

Further improvements are possible with “conservative bootstrap” mode. This re-uses the more extreme samples more often which further helps to maintain accuracy also in long-tail cases. You may play further with the different options and look to the kurtosis or histogram.

Appendix

Multivariate Capability Index

RealTime MC treats one specification and one data set at the time. Usually a datasheet contains many specifications, but usually only a few of them cause the biggest headache for designers. With the new C_{GPK} they can treat the specifications efficiently, but there are also cases in which the yield on the individual performances are quite easy to fulfill, but the overall yield might be still low, because the performance characteristics compete and are hard to fulfill on the same time. So-called “multivariate C_{PK} ” can be calculated taking also correlations into account, but we focus on the univariate case. In the univariate case there is a one-to-one relation between the yield and the specification limit (acc. to inverse CDF or percentile function), whereas for the general case infinite specification combinations exist to obtain a certain yield! This and several other reasons make the application of multivariate C_{PK} ’s a bit more difficult, especially as in the design phase not all specifications might be fully clear. Therefore, there is also yet no standard on multivariate C_{PK} ’s.