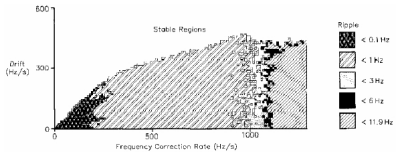
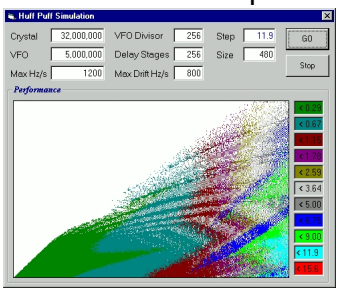


The Simulator

This simulator is designed to investigate Peter Lawton G7IXH's "fast" stabiliser design to determine the performance in terms of ripple and stabilisable drift parameters, with various number of shift register stages. The starting point for this experiment was to try to reproduce the same diagram that Peter has in his Nov 1998 QEX article at the bottom of the third page (figure 4). Readers should familiarise themselves with this excellent article, since it explains the basis of the "fast" Huff & Puff technique that concerns this simulator.

Peter's diagram is reproduced to the right. It shows the regions of stability for drift rates from 0 - 600Hz/s and frequency correction rates from 0 to about 1,250Hz/s. The diagram shading shows the amount of frequency ripple present on the output. This stabiliser is his "fast" design, with 5MHz VFO and 32MHz crystal reference oscillator, with 11.9Hz step size.

Intuitively one would imagine that in order for lock to occur, the drift rate cannot be greater than the correction rate, otherwise the stabiliser will not be capable of correcting the drift. This means we should see stability occurring only beneath a 45-degree diagonal line, which from the diagram is clearly the case. Higher rates of correction lead to higher ripple rates but also make it possible to correct worse drift. The white areas in these diagrams indicate regions where the stabiliser is not capable of correcting the VFO drift.



I wanted to investigate the effect of different numbers of shift register stages on the design. My simulator uses the same internal engine as my Java simulator which shows frequency. However this simulator is written in Visual Basic on MS Windows, and runs the simulation engine once at each drift vs correction rate point on the chart. The ripple is "measured" at the end of the simulation run and a colour coded point drawn on the chart.

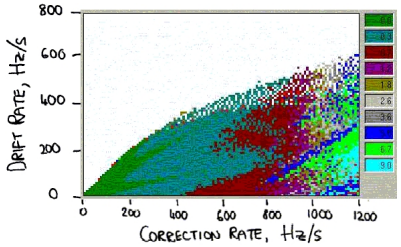
Here on the right is a screenshot of the simulator. Notice that all the parameters can be adjusted at will. The "size" parameter specifies plot resolution and just relates to how many simulations will be done along the horizontal (correction rate) axis. 480 produces a very detailed

plot, but with large numbers of shift register stages, the image can take a very long time to draw. To simulate larger shift registers, I generally choose a lower resolution.

The "step" text box (here showing 11.9Hz) is the only one which is not changeable by the user, it is a calculated output rather than an input parameter.

It is important to note that the simulator does not use any theoretical model to explain the behaviour of the Huff & Puff stabiliser. Instead, it contains a large array to simulate the shift register, and samples the state of an imaginary crystal reference oscillator at each instance of the divided VFO frequency, propagating this signal through to the final integrator and modelling its effect on the VFO frequency. This method gives a high degree of confidence in the results, since it does not rely on many theoretical assumptions. In any case, it's the only way I could think of...

The resulting image shows interesting patterns of stability. Try explaining those! There are many white pixels, which would normally indicate a lack of stable lock, but in the picture shown to the right are more likely to be due to some deficiency in the way I measure the "lock" condition and the frequency ripple. You can immediately see the similarity between my output and G7IXH's, which is very comforting.



To the right here is another image with the same settings, but this time using 120-pixel resolution. That means 12 times faster calculation compared to 480 pixels! Notice that the white specks are no longer present except near the top right of the curve, where stability might be expected to be somewhat marginal.

When writing this program, I didn't get as far as perfecting the display by adding legends and scale to the axes. In order to resolve any potential confusion, I have drawn the axes on the right-hand diagram by hand. This applies to all subsequent images on this page.

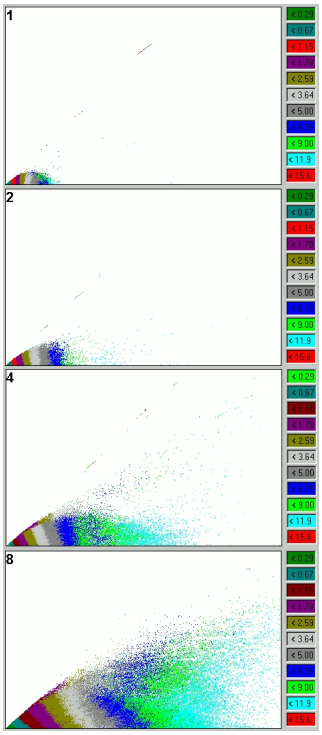
Simulation Results

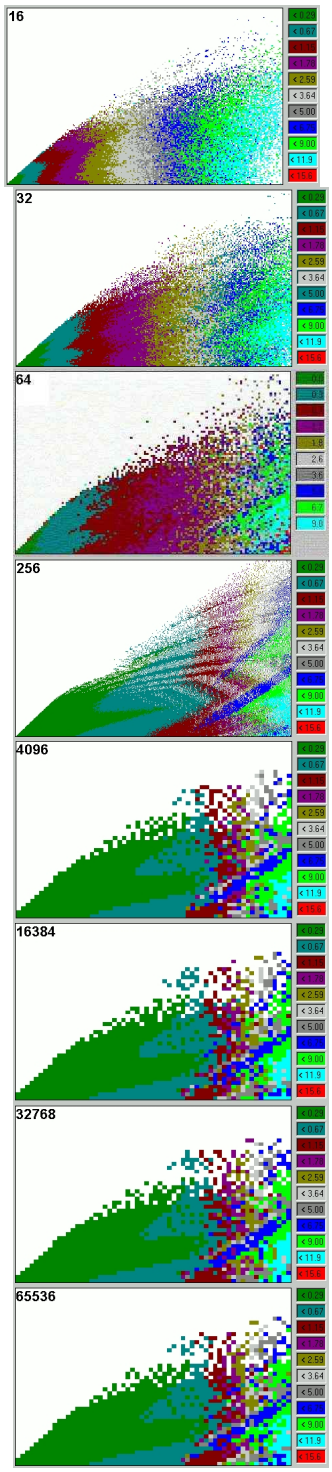
Now for some interesting results using this simulator! Each time the number of shift register delay states is reduced by a factor of 2, the VFO division ratio must be multiplied by a factor of two in order to maintain the same step size (distance between successive stable lock points) of 11.9Hz. All of the resulting screenshots below use the following parameters, in order to be comparable:

Crystal reference frequency
32 MHz
Base VFO frequency
5 MHz
Correction rate
0 to 1,200 Hz/s
Drift rate
0 to 800 Hz/s
Frequency step
11.9 Hz

The number of shift register delay stages is the variable parameter under investigation, and was varied from 1 to 65536. The VFO division factor was adjusted accordingly in order to keep the frequency step size at 11.9Hz (as in Peter G7IXH's article diagram). The frequency ripple colour coding is shown on right hand edge of each plot. The number of pixels was reduced for the very long shift registers, because otherwise the simulation run took too long. In each image I have cut out just the plot from the whole screenshot, in order to clarify and save space. I have indicated the number of shift register stages in the top left of each image. The axes should be labeled as above right.

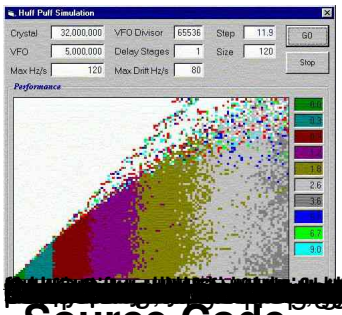
Warning: Not all the colour coding, or the boundaries between ripple colour regions, are the same. Apologies for this. These are the images I produced in June 2000 when I wrote the simulator, and I was probably still experimenting. I probably ought to re-run the simulator with completely consistent boundaries, but I haven't the time for that right now. In the meantime, I think you will get the idea.





Results Analysis

Straight away, it is clear that the "fast" approach offers massive advantages compared to the original "classic" Huff & Puff stabilisers. The first image shows a single shift register stage. This situation is equivalent to a "classic" stabiliser. You can see that by comparison with the "fast" stabiliser involving multiple shift register stages, the "classic" stabiliser is much less capable in terms of the amount of drift it is able to compensate.



Source Code

This simulator was written in Visual Basic 5.0 in June 2000. Here is the source code for fmMain.frm . Most of it concerns the definition of the user interface objects i.e. text boxes etc. If you scroll down to the bottom you will see the actual simulation calculation engine and that it is relatively simple.

If you have Visual Basic you could use this to rebuild the simulator, since this is the only source code file in the project. I also have available an executable file if you wish to run the simulator but do not have Visual Basic. This requires other supporting "VB Runtime" DLL's which may or may not be present on your system. I have a full installation using a setup.exe which is approximately 2MBytes and should work on Windows machines (but I can't offer any guarantees whatsover and accept no responsibility for anything which happens to your system, should things not work out). If you want these files you can Email Me .