

PHASE LOCKED LOOP TECHNIQUES: INTERACTIVE SIMULATION
WITH THE ADVANCED CONTINUOUS SIMULATION LANGUAGE (ACSL)

Edward E. L. Mitchell
Mitchell and Gauthier Assoc. Inc., Concord, Mass.

SUMMARY

Describes the implementation of a non-linear model of a phase locked loop and its use in the design of a switched frequency generator. Phase plane trajectories are determined for a generic one-parameter locked loop.

INTRODUCTION

This paper describes some results of modelling a simple phase locked loop and comparing linear and non-linear models. Phase locked loops (PLL's) can be analyzed for small phase differences with linear theory that corresponds directly to the design of a simple, low order, control system. Straight forward techniques can be used to specify resistor and capacitor values for a design that has adequate response and damping. Outside the linear region, however, the mathematics becomes more difficult and the analysis of such things as lock time and noise drop out can be better performed by simulation.

The loop described herein is a design proposed for a frequency synthesizer, the output being locked to integer multiples of a reference frequency. Typical specifications call for a particular settling time after the divider is switched. The start up action is also of interest, although not examined in detail here, where voltage is first applied to the circuit elements.

Other uses of phase locked loops are described in detail in a Signetics Application Book (1). Typical examples are the so-called flywheel synchronizer for TV horizontal and vertical sweeps and high accuracy FM demodulation. Signal tracking of doppler-shifted very narrow band signals from deep space probes is treated by Gardner (2) and Viterbi (3).

FREQUENCY SYNTHESIZER

Figure 1 shows the typical arrangement of a PLL and its essential components: The input is a signal of a particular frequency f_I which is compared in phase with the output f_0 of the voltage controlled oscillator (VCO). The output of the phase detector is filtered and becomes the input to the VCO - so changing the output frequency f_0 . If the loop is well designed and f_0 and f_I are not too far apart, the output frequency will be drawn towards the input frequency, until after a settling period they eventually become identical.

Phase detectors work in different ways - a typical one operating by multiplying the two wave forms together and filtering the output. i.e.

$$\epsilon = V_1 V_0 \sin(2\pi f_1 t) \sin(2\pi f_0 t)$$

which has sum and difference frequency outputs. The important output is the phase error or difference frequency i.e.

$$\begin{aligned} \epsilon &= \frac{V_1 V_0}{2} \cos(2\pi(f_1 - f_0)t) \\ &= K_p \cos(2\pi(f_1 - f_0)t) \end{aligned}$$

When the output frequency has exactly the same frequency as the input and differs in phase by 90 degrees, then the phase error will be zero.

When the frequencies are varying, the term $2\pi(f_1 - f_0)t$

must be changed to the integral of the difference or

$$\theta = \int 2\pi(f_1 - f_0) dt - \frac{\pi}{2}$$

It is usual to change the meaning of phase difference θ by adding an initial condition as above so that the phase error can be written using the sine function

$$\epsilon = K_p \sin(\theta)$$

since this linearizes about zero rather than 90 degrees.

The filter that drives the VCO has the job of removing the sum frequency and also stabilizing the loop. Typical circuits are a simple RC lag, a lead over an integrator or a double lead over two integrators. Call this transfer function $G(s)$, then figure 2 shows the block diagram loop structure where we've added a frequency divider in the feedback path for the frequency synthesis application.

The open loop, small signal, transfer function of this system is

$$\frac{2\pi K_p K_F K_V G(s)}{K_H s}$$

and the closed loop behavior is determined by the roots of

$$1 + K \frac{G(s)}{s}$$

where K is the loop gain given by

$$K = \frac{2\pi K_p K_F K_V}{K_H}$$

Now consider a design example where we are to determine the desired characteristics for a frequency synthesizer that can be switched from 2 Mhz to 3 Mhz in increments of 0.1 Mhz by changing the frequency divider K_H in figure 2. We'd like the output to be phase coherent with the input or reference frequency so the steady state phase error should be zero. We'd also like the lock-up time between channels to be less than 1 msec with an overshoot of 20% or less.

The frequency range and increment lead to a reference frequency of 100 KHz and a divider ratio ranging from 20, for an output frequency of 2 Mhz, to 30 for 3 Mhz.

The phase coherence specification requires the filter to contain an integrator or pole at the origin so that the phase will eventually be reduced to zero - if lock up occurs. This requirement leads to a type 2 loop and a lead is necessary for stability. The filter will look like

$$\frac{K_F(\tau s + 1)}{s}$$

Implementation is by resistor/capacitor network around a high gain amplifier as shown in figure 3.

Commercial VCO's and phase comparators can be obtained as off the shelf items, it being necessary to specify the gain before finalizing the loop design. A typical VCO (Motorola MC4324 100pF feedback capacitor) has a frequency/voltage plot

as shown in figure 4 which is fairly linear over the operating range of interest, 2-3 Mhz. The complete curve is needed when studying the start-up or initial voltage turn-on transient. The important characteristic, as far as the loop response is concerned, is the gain, K_V , which in this case is about 2 Mhz/volt.

The phase comparator can be of various forms including the basic sine operator, a triangle wave derived from a squared up switched multiplier or a more complicated device that extends the linear range from $-\pi$ to $+\pi$, a sawtooth wave form response. In this case, we'll look at a simple sine converter with a gain K_P volt/rad of phase difference.

The loop characteristic equation can now be written as

$$\frac{2\pi K_P K_F K_V}{K_N} \frac{(s\tau + 1)}{s^2} + 1 = 0$$

or if

$$K = \frac{2\pi K_P K_F K_V}{K_N}$$

$$s^2 + K\tau s + K = 0$$

From comparison with a standard second order filter

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$$

we can write

$$K = \omega_n^2$$

$$\tau = 2\zeta/\omega_n$$

From conventional servo theory, it can be shown that this will settle to within 5% at $\omega_n t = 4.5$. Since lock-up time is to be less than 1 msec, then

$$\omega_n \leq 4.5/0.001 = 4500 \text{ rad/sec}$$

$$\text{or } K \leq 2.025 \cdot 10^7 \text{ sec}^{-2}$$

A damping factor ζ of 0.8 is required to keep overshoot below 20% so the lead time constant is given by

$$\tau = 2 \times 0.8 / 4500 = 0.36 \text{ msec}$$

The filter can be configured from an operational amplifier and a resistor capacitor network shown in figure 3. If we write $1/Cs$ as the equivalent resistance of the capacitor, the transfer function can be written as

$$\frac{V_{OUT}}{V_{IN}} = \frac{1}{R_1} \frac{R_2 Cs + 1}{\left(\frac{R_2}{A} + 1\right)Cs + \frac{1}{A}}$$

This equation has the form

$$\frac{X}{Y} = \frac{K_F(T_{LED}s + 1)}{(T_{LAG}s + 1)}$$

but the lag time constant T_{LAG} can be made very large if the amplifier gain is very high. It's shown like this because the difference can be important in the non-linear lock-up transient response. A pure integrator, given sufficient time, will always lock up. The lag time-constant above can overcome the drift towards lock-up if frequency deviation is high. Analysis is by the non-linear simulation model.

For this design synthesis, assume that A is very large so that the filter gain K_F is given by

$$K_F = \frac{1}{R_1 C}$$

and the lead time constant by

$$T_{LED} = R_2 C$$

If we try a capacitor value of 0.5uF, then R_2 has to be 720 ohms to give the lead time constant of 0.36 msec and $R_1 = 4598$ ohms to make the filter gains come to 435 volts/volt. Standard resistor values that can be used are 680 ohms for R_1 and 4.7K for R_2 . Substituting back we get a gain K_F of 426 - equivalent to a natural frequency ω_n of 4451 - and a time constant T_{LED} of .35 msec - equivalent to a damping factor ζ of 0.78. These gain values were obtained at a feed back ratio K_N of 30. At 20, the natural frequency and damping are increased by the square root of 30/20 to 5451 and 0.96 respectively - an increase of 22%.

SIMULATION PROGRAM

The loop was modelled using the Advanced Continuous Simulation Language (ACSL). Designed for representing the behavior of systems described by differential equations, ACSL readily lends itself to the interactive analysis of both the linear and non-linear effects of the phase locked loop.

The simulation program is listed in Figure 5 which shows the model definition code that describes the structure of the simulation - it won't execute unless given what are called run-time drive commands described later. The heart of the model is contained in the four lines of code that define the phase error, E , voltages $V1$ and $V2$, and the output frequency $F0$, but other specification statements extend the program. Roughly half the text is in the form of comments - delimited by quotes - to explain the operation of the model.

The first active line in the model, the TABLE statement, defines the VCO frequency as a non-linear function of input voltage. Nine argument breakpoints are given for input voltages ranging from 1 to 5.5 volts: Corresponding frequency values, in hertz, follow the argument breakpoint definition. ACSL allows arbitrary functions of up to three independent variables to be defined.

After the TABLE definition, some type statements are given and parameters are preset by the CONSTANT statement. The integer values specified are related to the divide ratio in the feedback path for frequency. In the program real numbers could just as well have been used, but the integers were kept to show how they were defined. ACSL, by default, considers all variables real unless explicitly typed otherwise.

The next section describes the integration and data recording parameters using ACSL keywords. The first statement defines the integration algorithm to be used by

ALGORITHM IALG=5

which says the name assigned to the algorithm specification variable is IALG with a starting or preset value of 5 - this means Runga-Kutta 4th order, fixed step size. The name so defined can be changed at run-time to any of the other available algorithms. These include two variable step, variable order (Adams-Moulton a la Gear and Gear's Stiff), Euler first order or rectangular, Runga-Kutta second order or trapezoidal, Runga-Kutta fourth order and lastly a hook to a user defined algorithm.

The integration step size is defined by specifying the maximum step so

MAXTERVAL *AXT=1.0E-5

which says take a step of 10 usec unless made smaller by the recording interval (CINT) constraint. In this case, the recording interval is specified to be 20 usec by

CINTERVAL CIHT=2.0E-5

All these parameters may be changed later at run-time.

In order to determine the phase error, we must integrate the difference between the input or reference frequency and the divided output frequency. The statement that performs this operation is at the heart of the continuous simulation system - integration i.e.

E=INTEG(THOPI*(FC-F0/KH),DLTHIC)

where the initial condition parameter is given a name in case we want to study the effect of closing the loop at different times: With this initial condition the error can be set to drive the loop in the wrong direction. In actual fact, and contrary to some claims, the initial phase error has little effect on lock-up time.

The voltage from the phase detector V_1 is obtained by

$$V_1 = RSW(LINEAR, E, SIN(E))$$

The RSW or real switch operator says to choose the second argument if the first argument is .TRUE., else the third. Thus if LINEAR is .FALSE., the sine function is applied to the phase error. Other functions can be used at this point to model triangle or saw tooth phase detectors.

The lead over an integrator for the type 2 loop is next modelled using the LEDLAG operator. It is done this way to make it easy to see the effect of finite amplifier gains. In the base-line model, the lag time constant is made large (100 sec) so its actually a very good integrator. Since T_{LAG} is very much greater than unity, the gain is written in terms of T_{LAG} so that the terms will cancel i.e.

$$LEDLAG(V_2 = TLED, TLAG, TLAG * K_F * V_1, V_{1c})$$

represents

$$\frac{V_2}{V_1} = \frac{T_{LAG} K_F (T_{LED} s + 1)}{T_{LAG} s + 1} \approx \frac{K_F (T_{LED} s + 1)}{s}$$

The loop is closed by specifying the output frequency of the VCO as a function of the input voltage - defined previously as a non-linear function via the TABLE operator

$$FO = VCOF(V_2)$$

In order to excite the system, the switching transient is defined as a change from a feedback ratio of K_{HZ} to that of $K_{HZ} + 1$. Nominally set to go from 20 to 21 at a switching time T_{SWIT} - preset to 0.2 msec.

$$KN = LSW(T.GE.TSWIT, KNZ + 1KN, KNZ)$$

where LSW is similar to RSW described previously but is used for logical or integer quantities.

The last statement of the program is the TERMT operator which stops the simulation run when the argument becomes true.

$$TERMT(T.GE.TSTP)$$

Since TSTP has been preset to 3.9 msec, the model will run for a little longer than this. Any number of TERMT operators may be included in the simulation and the arguments may be logical expressions of any complexity.

SIMULATION TEST RESULTS

The model definition of figure 5 was typed into a permanent file (PLL) on a CDC CYBER computer, NOS operating system. The ACSL processor consists of a translator that translates the model definition code to Fortran and an extensive run-time library that provides all the interactive input/output and plot features. With the IOS operating system, a procedure file is used so that the user doesn't specifically have to invoke each processor in turn - translator, compiler, loader - all the control cards being compressed into two. At an interactive terminal, after logging on, the procedure file RUNACSL is retrieved and executed with a CALL statement

```
...
GET,RUNACSL/UN=ACLSYS
CALL,RUNACSL(MODEL=PLL,PLTDEV=1)
```

The model file is indicated as PLL which is retrieved from the users workspace. The plot device (PLTDEV) is indicated by numeric pointer so that test within the procedure can retrieve the appropriate device libraries. With PLTDEV=1, this means to use the Tektronix PLOT10 driver for interactive plotting. Other devices that may be connected include Calcomp,

SC-4020, Zeta, Gould, Xynetics and Houston Instruments. Interface is through a simple device driver subroutine that initializes and moves with pen up or pen down so implementation is relatively straight forward to any other hardware.

When the translation, compilation and load sequence is complete, the execution sequence expects what are called run-time drive cards to exercise the model. A prompt of a question mark (?) is generated by NOS to tell the user that input is expected so that he can enter the appropriate command. The following list gives most of the run time commands available to the user in running the simulation, the standard sequence being SET, START, DISPLY and PLOT - repeated over and over again as different design changes are analyzed.

- a) DISPLY A, B, ... list current values of any named variable or array
- b) OUTPUT A, B, ... list values during subsequent runs at communication interval
- c) PLOT A, B, ... both printer and/or line plots from previous run
- d) PREPAR A, B, ... save values during subsequent runs at communication interval
- e) PRINT A, B, ... print values saved from previous run
- f) PROCED name ... established command collection as 'name'
- g) END ... terminate PROCEDURE definition command collection
- h) REINIT ... change initial condition to current state
- i) SAVE file ... save current variable values on named file
- j) RESTOR file ... restore current variable values to those of previous SAVE
- k) SET K9 = 5.6, ... change value of any named variable
- l) SPARE ... links to user supplied subroutine
- m) START ... initiate simulation run
- n) STOP ... terminate simulation study

Since the base simulation in the model definition section has all its parameters defined, ready to run, the only things to do is to define what variables are to be OUTPUT during the run and what saved for plotting or column printing (PREPAR). In response to the question mark enter

```
?OUTPUT T,E,FO,'NCIOUT'=50
```

which says we want to see the values of time (T), phase error (E), and output frequency (FO) listed at every 50 communication intervals while the simulation executes: Since the communication interval was defined to be 20 usec, this means that we'll see some numbers every millisecond. At the next question mark, enter

```
?PREPAR T,E,FO,V1,V2
```

which says to save the values of the named variables at every communication interval while the run is in progress. No qualifier is used since all data points are needed for plotting.

Having established the output lists, the command START actually exercises the model which will run for the 3.9 msec specified in the TERMT operator. The variables on the OUTPUT list will be listed every millisecond or five lines of output (0,1,2,3,3,9) so that progress can be monitored. If its not satisfactory, the break key will interrupt the simulation, returning control to the ACSL executive so that conditions can be modified and the run reinitiated.

Before we plot any pictures, we have to define the medium to the ACSL executive. The default action is to obtain printer plots and no line plots. Either or both can be obtained at each plot by setting system flags PRNPLT (printer plot) and/or CALPLT (line or Calcomp plots). Since we want to see pictures on the Tektronix screen, we enter

```
?SET PRNPLT=.FALSE.,CALPLT=.TRUE.
```

The variable CALPLT is a hold over from the days of interface

to a Calcomp plotter. Now this variable is used to signify the drawing of line plots - as opposed to character or printer plots - on the device selected at load time - in this case the Tektronix by PLTDEV=1. Now we are ready to draw the picture so enter

```
?PLOT FO,E
```

which produces the picture of figure 6. Remember this shows the response of the linearized system. Frequency jumps from 2.0 to 2.1 Mhz, the transient starting at 0.2 msec and is complete at 1.2 msec. Note, however, that the error E peaks at about 2 radians which should indicate that the linear criterion isn't really satisfied for this particular transient. If we had a saw-tooth phase comparator that covers the range -- to + then the linear assumption would be in order.

To see the non-linear model perform, enter

```
?SET LINEAR=.FALSE. $ START
```

which changes the flag so that the RSW operator will pick SII(E) and run the simulation once more.

```
?PLOT FO,E
```

now produces the picture of figure 7. The frequency oscillates at the beat frequency (of 5 khz) but the average value drifts slowly up to 2.1 Mhz, the beat frequency becoming slower until it is in range to lock-up which occurs at about 2.5 msec. Cycle slipping has occurred so that the cumulative phase error E comes to 10 or five complete cycles.

A more severe test comes when we try to switch the feedback divider over two units - 20 to 22 - or change the output frequency from 2.0 to 2.2 Mhz. To initiate this transient

```
?SET IKN=2 $ START
```

which will add two to the value of KNZ (=20) at the switch time. This was the reason we gave the increment a name so that it could be changed: If we'd written the switch action as

```
KN = LSW(T.GE.TSWIT,KNZ+1,KNZ)
```

or even worse

```
KN = LSW(T.GE.TSWIT,21,20)
```

we would have had to change the code in the model in order to change the increment - necessitating the overhead of a retranslation, compilation and load. As a general rule, all constants should be referred to symbolically since it is difficult to decide what quantities it would be nice to change at run-time. It is surprising how many switches can be avoided given the ability to change the value of any constant in the problem.

After running this new condition, we find that lock-up doesn't occur in 3.9 msec so we have to extend the stop time and run again

```
?SET TSTP=0.0099 $ START
```

As before, we set the stop time to be a little less than a round number since the simulation will always exceed this time when stopped - the equality of real numbers occurs with very low probability. If we'd set it to stop at 10 msec, then the actual stop time would have been 10.01 msec, one calculation interval or MAXT beyond. If we were to PLOT in this condition the rounding action would cause the x-axis to extend from 0-20 msec wasting half the picture. An alternate is to key the x-axis scale to the stop time set by

```
?PLOT 'XHI' = TSTP, E, FO
```

which forces the high value of the x-axis scale to the contents of TSTP while plotting the y-axis variables E and FO. We'd prefer to stop the problem a little before the rounding values of 1, 2, 4 or 5 and let the automatic scale calculator do the work.

To return to the run-time drive card sequence, having just run the model for a little over 9.9 msec, the command

```
?PLOT FO, E
```

now produces the picture of figure 8, which shows 53 cycles of slippage or over 100 accumulated phase error before lock is achieved. The wavy envelope on the frequency plot, FO, is not due to a change in amplitude but to the beat of the oscillation with the data recording action at every communication interval CIHT(= 20 usec). The initial cycle slipping or difference frequency is at a 10 khz rate or about 16 usec/rad so the sampling is rather coarse - 5 points per cycle. As the output frequency drifts towards lock, the beat frequency becomes slower and slower and we record more and more points per cycle.

Compare the two lock-up times: When the frequency offset is 5 khz, the last cycle starts at 1.6 msec into the transient: When the offset is 10 khz, the last cycle starts at about 9 msec. Theoretical lock-up time is proportional to the square of the frequency difference and Viterbi (3) gives the following approximate formula for the pull-in time, T_p , for a type 2 loop

$$T_p = \frac{(\Delta\omega)^2}{2\zeta\omega_n^3}$$

where $\Delta\omega$ is the initial frequency offset. The parabolic trajectory of FO can be clearly seen in figure 8. For this case ($KH=20$) we have $\zeta = 0.98$ and $\omega_n = 5500$ rad/sec so pull-in time should be 3 msec for $\Delta\omega = 31,420$ rad/sec (5 khz) and 12 msec for $\Delta\omega = 125,700$ rad/sec (10khz) - in reasonable agreement with the results considering that the formula is only recommended for use at mid-range frequency offsets. The simulation program will give the exact answer within the limits of accuracy of the integration routine.

To consider the original design problem, it now seems not feasible to expect the system to lock-up within 1 msec unless a considerably higher loop gain is achieved to keep the phase error below 90 degrees or unless the linear region of the phase comparator can be extended. It's usual in this situation to help the loop by switching in nominal voltages to the VCO so that the integrator only has to accumulate enough charge to overcome errors - temperature effects, resistor inaccuracies etc.

GENERIC EQUATION, TYPE 2 LOOP

Viterbi (3) has reduced the standard type 2 loop to a one parameter non-linear differential equation by time scaling. Figure 9 shows the block diagram for the standard loop where the loop gain is collected into one constant K. From this figure, we can write

$$\begin{aligned} \dot{E} &= X - Y \\ \dot{Y} &= K \left(T_{LED} \frac{d}{dt} \sin(E) + K \sin(E) \right) \\ &= K \left(T_{LED} \cos(E) \dot{E} + K \sin(E) \right) \end{aligned}$$

If we differentiate the first equation and then substitute for \dot{Y} , then

$$\ddot{E} + K T_{LED} \cos(E) \dot{E} + K \sin(E) = \dot{X}$$

Using the damping factor ζ and natural frequency for the equivalent linear circuit, we have

$$\ddot{E} + 2\zeta\omega_n \cos(E) \dot{E} + \omega_n^2 \sin(E) = \dot{X}$$

where

$$\begin{aligned} \omega_n^2 &= K \\ 2\zeta\omega_n &= K T_{LED} \end{aligned}$$

Now change the time scale to τ such that

$$\tau = 2\zeta\omega_n t$$

Then

$$\dot{E} = 2\zeta\omega_n \frac{dE}{dt} = 2\zeta\omega_n E'$$

and

$$\ddot{E} = (2\zeta\omega_n)^2 E''$$

and the characteristic equation becomes

$$E'' + \cos(E) E' + \frac{1}{4\zeta^2} \sin(E) = 0$$

Viterbi uses 'a' for the parameter $1/4\zeta^2$, the coefficient of $\sin(E)$ in the above equation. For the previous loop, $\zeta = 1$, so $a = 1/4$ and the initial condition on phase rate E' is 5 kHz time scaled by $2\zeta\omega_n$ or about 11000 i.e. $E'(0) = 2.9$ rad/sec.

SIMULATION PROGRAM FOR PHASE PLANE ANALYSIS

Rather than plot the frequency and phase as a function of time, it is instructive to plot the phase plane trajectory of E versus E' . Due to cycle slipping, this doesn't look like the conventional spiral since phase rate E' retains the same sign until lock-up occurs. In order to plot the trajectory, the phase error is calculated modulo 2π and when plotted, each cycle slipped draws a line from one side of the plot to the other, slowly moving towards the x-axis as lock is approached. When the error rate comes within the lock-up range, the trajectory quickly spirals towards the origin.

Viterbi obtained pictures of this effect using an analog computer and fairly complicated logic to hold the model, lift the pen and move it from one side to the other before repeating the cycle. We show here how the same action can be performed in an all-digital environment. The logic is still relatively complex but is implemented using the standard ACSL plotting features.

The model definition code to implement this simulation is shown in figure 10. The differential equations are straight forward, being expressed by

$$THD = \text{INTEG}(-(\text{COS}(\text{TH}) * \text{THD} + \text{SIN}(\text{TH})), \text{THDIC})$$

$$\text{TH} = \text{INTEG}(\text{THD}, \text{THIC})$$

The tricky part of the simulation is to integrate across a complete cycle (at the slip or difference frequency) and ensure that a data point is recorded as close to $\theta = +\pi$ and $\theta = -\pi$ as possible - after placing the actual angle θ within the circle by a modulo operation. i.e.

$$\theta' = \text{mod}(\theta + \pi, 2\pi) - \pi$$

In addition, on a subsequent PLOT command of $\dot{\theta}$ versus θ , the pen should be lifted as θ completes the cycle going from $+\pi$ to $-\pi$.

ACSL is able to lift the pen by using a system variable FTSPLT - flyback trace suppression for plots. When this variable is true, the pen is lifted when drawing a line between two points on the saved data file if the first variable on the PREPAR list is more negative than the previous time. It's designed basically for plotting multiple runs where time T successively sweeps from zero to some end value. The reset action when time goes back to zero triggers the flyback trace suppression action so that parametric curves can be plotted on the same piece of paper. We'll use this feature to plot multiple cycles in the same run by generating a variable FTS that holds constant during the cycle, then steps negative at the point when the pen is to be lifted - its the inverse of the cycle counter - delayed by one communication interval. In order to record data at the end points of the cycle, we adjust the communication interval dynamically by recalculating it in the DYNAMIC region - this is the region executed at each communication or data recording interval. The estimated time to reach the end of the current cycle, assuming $\dot{\theta}$ is positive is given by

$$\Delta = \frac{(2n_{CY} + 1)\pi - \theta}{\dot{\theta}}$$

where n_{CY} is the cycle count, initialised originally to zero. Note n_{CY} is a real variable in the program.

We are going to consider only positive initial conditions on $\dot{\theta}$, so if $\dot{\theta}$ is zero or negative we are close to lock and no more cycle slippage will occur. The expression thus used for the time is

$$\Delta t = \text{RSH}(\theta > 0, \frac{(2n_{CY} + 1)\pi - \theta}{\dot{\theta}}, 10^{30})$$

which says to choose an infinitely large number when $\dot{\theta}$ is negative. We can now change the communication interval to aim at the cycle end - using Δt if its smaller than a base communication interval CINTZ (= 50 msec) but not allowing it to go too small (<0.001) i.e.

$$\text{CINT} = \text{AMIN1}(\text{CINTZ}, \text{AMAX1}(\text{CINTMN}, \Delta t))$$

Once the communication interval has arrived at the minimum CINTMN (= 1msec), then we want to increase the cycle count which will make Δt open up next time, putting the communication interval back to its base level of CINTZ (= 50 msec)

$$\text{IF}(\text{CINT} .EQ. \text{CINTMN}) \text{NCY} = \text{NCY} + 1$$

Note that this is one of the few times that real numbers can be tested for equality. Since Δt is monotonic downwards, eventually it will become smaller than CINTMN and then the AMIN/AMAX operation will guarantee that CINT is exactly equal to CINTMN.

The flyback trace suppression flag FTS is made the negative of the cycle counter though physically placed before the line where n_{CY} gets bumped. (Code in the INITIAL, DYNAMIC or TERMINAL section is not sorted). This means that FTS changes by one - more negative - at the data point that terminates the 1 msec communication interval so that the pen is lifted just for this period, while the cycle restarts from the left.

The variable to be plotted is θ placed within the range $-\pi$ to $+\pi$. This is accomplished by defining a scaled θ , θ_S , calculated by subtracting 2π from θ for each cycle slipped.

$$\theta_S = \theta - 2n_{CY}\pi$$

The stopping condition is when the radial distance from the origin in $\theta_S, \dot{\theta}$ space is less than some given value (10^{-3}) when lock up is said to have occurred i.e.

$$\text{TERMT}(\theta_S^2 + \dot{\theta}^2 < \epsilon_{TH})$$

It is also a good idea to specify a stopping condition on time since from experience it is relatively easy to specify a termination condition that never occurs - the first run of this model had θ instead of θ_S in the above TERMT operator!

As before, to run the program we place the model definition on a permanent file and invoke the procedure file RUNACSL to process it. The key to the operation is the PREPAR statement which is entered as

$$\text{PREPAR FTS, T, THS, THD}$$

which keys the variable FTS to the flyback trace suppression action. After a START which runs the simulation

$$\text{SET FTSPLT} = \text{.TRUE.}$$

$$\text{PLOT 'XAXIS'} = \text{THS}, 'XLO' = -3.142, 'XHI' = 3.142, \text{THD}$$

will plot $\dot{\theta}$ (THD) against an x-axis of θ_S (THS) which runs from -3.142 on the left to 3.142 on the right. If we didn't specify the x-axis variable, then the first variable on the PREPAR list FTS would have been used - not really suitable for drawing pictures. If we hadn't specified the scale factors, ACSL would have chosen -5 to +5, wasting nearly 40% of the plot area.

Figure 11 shows the phase plan trajectory for $a=0.25$ and an initial rate $\theta_{IC} = 3.142$ rad/sec. The bunching together

shows the slow drift when the system is far from lock-in becoming progressively faster as lock approaches. The time domain plot of this run is shown next - figure 12.

The rest of the figures show the trajectories for various combinations of parameter $a (= 1/2\zeta^2)$ and initial phase rate $\dot{\theta}_{IC}$.

CONCLUSIONS

An interactive model of a phase locked loop has been developed and it has been shown that non-linear effects must be considered in the system design. These models can be extended to include receiver amplifier characteristics such as AGC circuits and IF amplifiers though as the band-width of the signal increases the running cost goes up due to smaller and smaller step sizes required.

The ACSL system allows relatively simple model definitions to be coded but then automatically handles all input/output operation and plotting. For some statistics on program overhead, the times in CP seconds (CDC 6600) are given for various segments of the simulation. Both models had virtually the same preparation cost, 4 - 4.5 CP seconds to complete the load, broken down as follows:

ACSL translation	1.42 sec
FTN compilation	0.52 sec
Load or Link-edit	2.23 sec
Total (CPC sec, CDC 6600)	4.17 sec

Execution was about 1000 RK4 integration steps per CP second. Tests with other models coded in Fortran show that ACSL models equal or exceed (because of readily changeable step sizes and integration algorithms) the solution rate of such programs.

REFERENCES

- 1) *Phase Locked Loop Application Book*, Signetics Inc., 811 East Arques Avenue, Sunnyvale, Ca.
- 2) Floyd M. Gardner, *Phase Lock Techniques*, Wiley, 1966.
- 3) Andrew J. Viterbi, *Principles of Coherent Communication*, McGraw Hill, 1966.
- 4) E. E. L. Mitchell and J. S. Gauthier, "Advanced Continuous Simulation Language (ACSL)", *Simulation*, March 1976, pp 72-78.

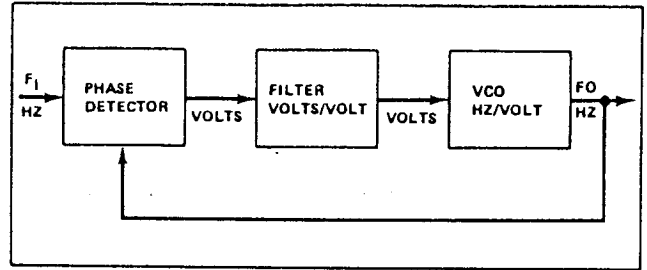


FIGURE 1: PHASE LOCKED LOOP CHARACTERISTIC BLOCK DIAGRAM

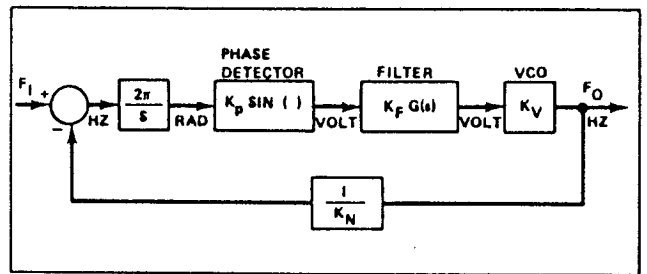


FIGURE 2: DETAILS OF PHASE LOCKED LOOP FREQUENCY SYNTHESIZER

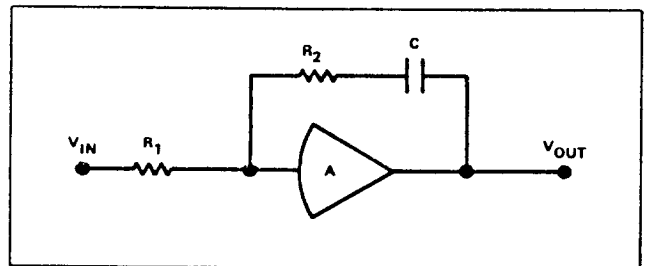


FIGURE 3: ACTIVE NETWORK FOR LEAD OVER INTEGRATOR FILTER

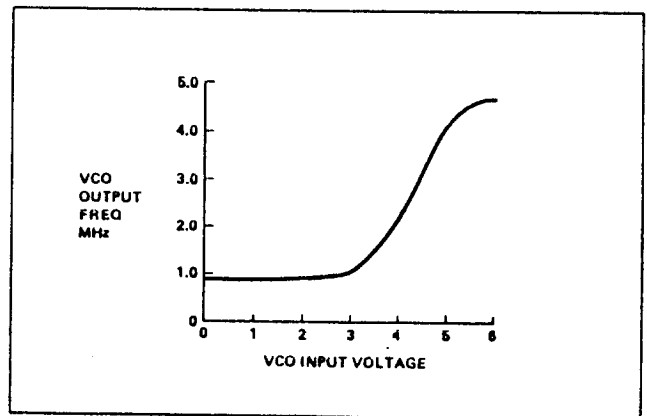


FIGURE 4: VCO OUTPUT FREQUENCY VERSUS INPUT VOLTAGE

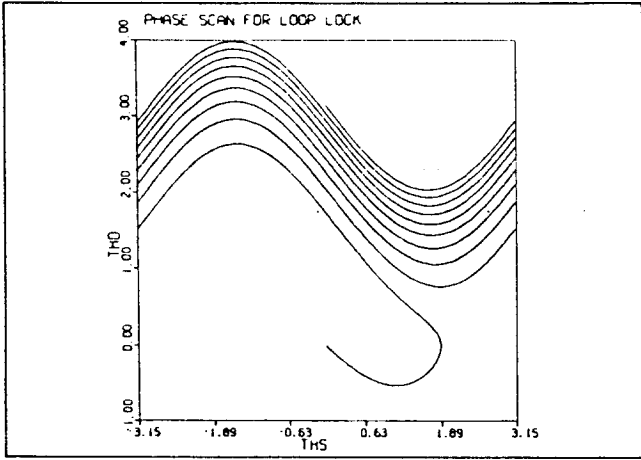


FIGURE 11: PHASE PLANE PLOT SHOWING APPROACH TO LOCK: INITIAL RATE = 3.14 rad/sec, PARAMETER A = 0.25

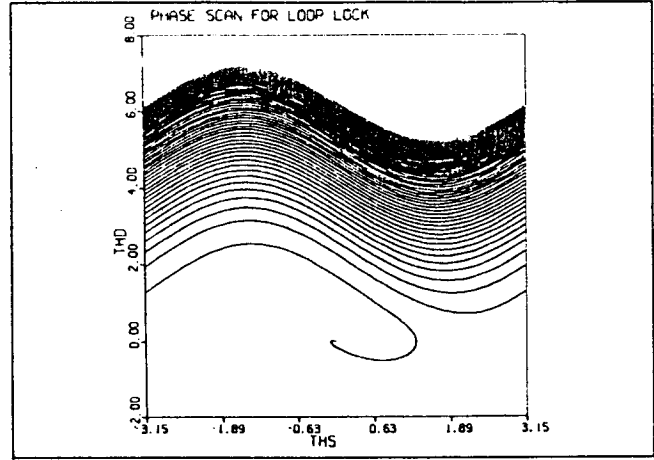


FIGURE 14: PHASE PLANE PLOT SHOWING APPROACH TO LOCK: INITIAL RATE = 6.28 rad/sec, PARAMETER A = 0.5

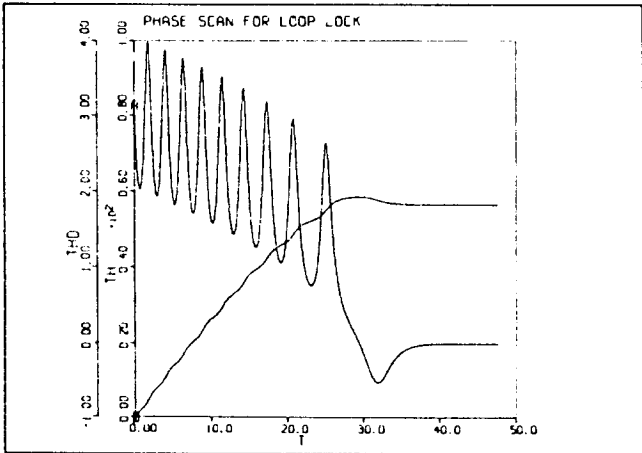


FIGURE 12: TIME HISTORY OF RATE AND ANGLE UP TO LOCK: INITIAL RATE = 3.14 rad/sec, PARAMETER A = 0.25

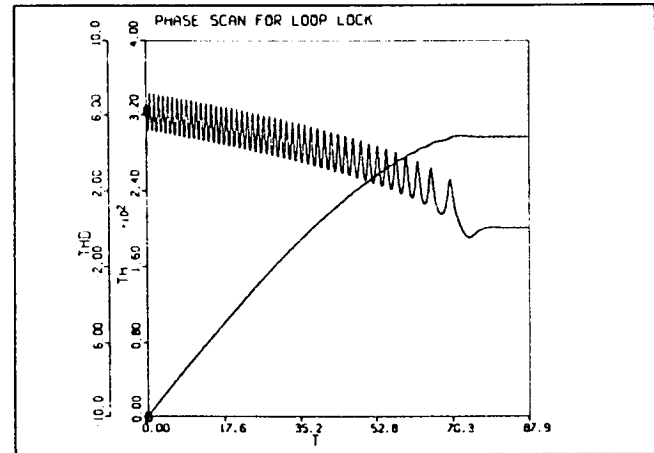


FIGURE 15: TIME HISTORY OF RATE AND ANGLE UP TO LOCK: INITIAL RATE = 6.28 rad/sec, PARAMETER A = 0.5

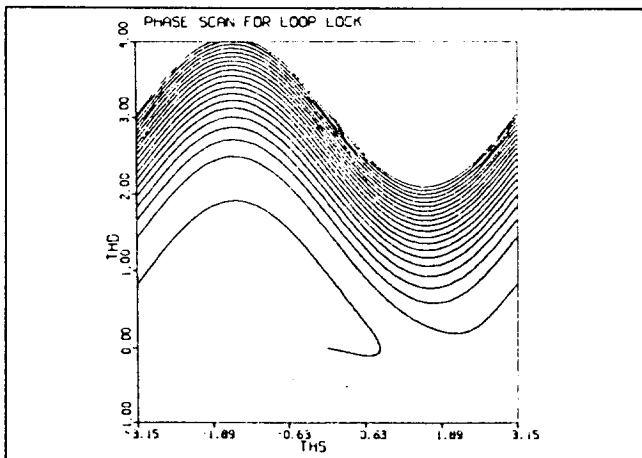


FIGURE 13: PHASE PLANE PLOT SHOWING APPROACH TO LOCK: INITIAL RATE = 3.14 rad/sec, PARAMETER A = 0.125

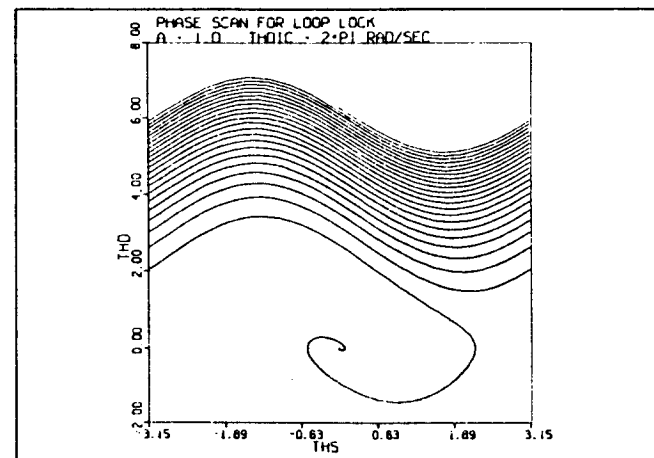


FIGURE 16: PHASE PLANE PLOT SHOWING APPROACH TO LOCK: INITIAL RATE = 6.28 rad/sec, PARAMETER A = 1.0