PHENO–A new concept of hybrid computing elements

by WOLFGANG GILOI
and HEINZ SOMMER
Technical University
Berlin, Germany

INTRODUCTION

PHENOs are based on the well-known fact that a digital-to-analog converter with variable reference (MDAC) can produce the product of an analog and a digital variable.[1] While for multiplication and division this principle can be used directly, it has to be modified for function generation. In order to obtain a system of computing elements in which any input or output variable can exist in analog or digital form, optionally, DACs and ADCs (analog-to-digital converters) are combined. The straight-line-segment approximation of arbitrary functions is done by splitting the (digital) argument of the function in two parts. The first group of r bits defines the nearest preceding breakpoint, while the second group of (n-r) bits is used for linear interpolation (n being the digital word-length). In a second method of function generation, which is particularly suited for multivariable functions, digital table look-up is combined with analog interpolation. On the base of PHENOs, this procedure provides minimum table look-up execution time and avoids stability problems.

The ADCs are key elements with respect to operation speed. Here, adaptive continuous converters, using the up-down-counter principle, show best performance. Constant band width/accuracy ratio is obtained by automatically adapted register length, i.e. for input increments greater than the least significant bit this bit is dropped, resulting in a doubled step size, etc. The conversion rate is further increased by using a subranging technique.

At the time being, the ADCs operate with 6 MC clock frequency. Therewith, a continuous ADC can track a 100 cps sine-wave with 0.01% accuracy and 1 kc sine-wave with 0.1% accuracy. Static errors of multiplication, division and function break points are less than 0.01%.

PHENO-equipped analog computers will essentially not be more expensive than high-precision analog computers containing very precise time-division or quarter-square multipliers, but they will produce a much better bandwidth/accuracy ratio. Additionally, they require no expensive interface when being linked with digital computers, since the variables are partly existing in digital form, anyhow. On the other side, PHENOs can be used to build inexpensive, small-scale, special-purpose computers for military and process control application. For this, additional elements have been developed, for example, units which the the combination of PHENOs with DDA-elements possible (DDA = digital differential analyzer). As opposite to usual analog computing elements, PHENOs permit miniaturization and operation under difficult environmental conditions.

Principle of PHENO

In a ladder network according to Figure 1 the short-circuit transfer conductance is[2]

$$\frac{i_o}{e_{i1}} = \frac{1}{2R} \sum_{k=1}^{n} \frac{S_k}{2^{n-k}}; \qquad (1)$$

$$S_k = \begin{cases} 0 & \text{(switch connected with ground)} \\ 1 & \text{(switch connected with } e_{i1}\text{)} \end{cases}$$

and, therefore, proportional to the digital number $d^* = d_n d_{n-1} \ldots d_2 d_1$ stored in the register. Connecting the output terminals 2-2′ with the input of an operational amplifier with feedback resistor $R_f = R$ results in an amplifier output voltage, which is related to the input voltage by

$$e_o = e_{i1} \cdot d^* \qquad (2)$$

By exchanging the input and output network of the

amplifier we have the inverse relation

$$e_o = \frac{e_{i1}}{d^*} \qquad (3)$$

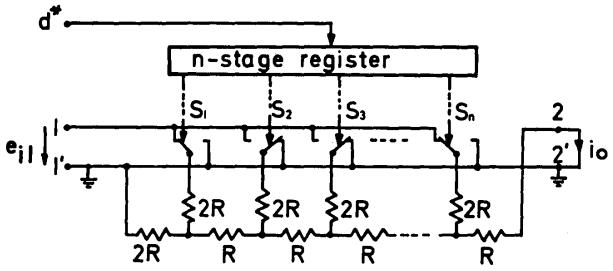


Figure 1—Principle of D/A-conversion

Hence, we obtain in the first case the product of an analog and a digital variable, while in the second case we get the quotient of both variables. In both cases the results are given in analog form. In the following, such a multiplying digital-to-analog converter will be called 'MDAC.'

Most analog-to-digital converters (ADC) are based on the principle that by a certain strategy the register setting d* of a digital-to-analog converter (DAC) has to be found in such a way that the output of the DAC equals the input voltage e_{i2} that has to be converted. When using a MDAC with reference voltage e_{i1}, we can obtain a register setting

$$d^* = K\frac{e_{i2}}{e_{i1}} \qquad (4)$$

i.e., d* gives, in digital form, the result of dividing two analog variables.

It is important to have an element for arbitrary function generation as for multiplication and division. In an analog computer (as well as in digital table look-up programs) the most simple solution of that problem is a straight-line-segment approximation of the desired function. This kind of function generation can be obtained for functions of one (digital) variable by a modified digital-to-analog converter, in which the argument of the function is split into two parts. The first r bits of the digital input of length n define (in binary code) the abscissa at the nearest breakpoint (the function being approximated, for example, by 2^r straight-line segments of equal length), while the remaining (n-r) bits are used for linear interpolation within the actual segment. For that purpose, this second part of the digital input is fed to a MDAC which has the function increment between the two actual breakpoints to reference. Of course, it is just a matter of coding to obtain any other distribution of the breakpoint abscissa. Since this function generating digital-to-analog converter (FGDAC) may operate with variable refrence, ti performs the operation

$$e_o = e_i \cdot f(d^*) \qquad (5)$$

If the output of one FGDAC generating the function $f_1(x^*)$ is used as reference of a second FGDAC generating the function $f_2(y^*)$, we can obtain the expression

$$e_o = e_i\,[f_1(x^*) \cdot f_2(y^*) + a_1 f_1(x^*) + a_2 f_2(y^*)]\ , \qquad (6)$$

which, in many cases, gives a good approximation of functions of two variables f(x,y).

Using a FGDAC as part of an ADC gives the operation

$$d^* = f^{-1}(e_{i2}/e_{i1}) \qquad (7)$$

(f^{-1} being the inverse of the function f(d*) which is generated by the FGDAC).

By combining the operations according to eqs. (2) through (7) (that means combining digital-to-analog and analog-to-digital conversions) multiplication, division, function generation or combinations of these operations can be arbitrarily performed with analog and/or digital inputs and outputs.

The purpose of this new technique is to exceed the accuracy limits of usual nonlinear analog computing elements. Since any operation of the PHENOs is based on conversion techniques, maximum static accuracy is that of high-precision converters. An analog accuracy of 0.01% (single-scale) and a digital resolution of 14 bits plus sign can be achieved. Hence, the PHENOs are well-matched to the static precision of best linear analog computing elements.

The dynamic errors depend on the conversion speed of MDACs and ADCs. The MDACs are not critical, because all switches in Figure 1 are set simultaneously. Even in the more critical case of variable reference switches, their settling time (to an error smaller than 0.01%) can be kept in the range of 1 μs, approximately.

The conversion time of a successive-approximation ADC is a multiple of that, because complete parallelism in the ADC operation is too expensive. Even when using an expensive subranging procedure, a conversion rate of 200,000 per second is the highest that could be achieved till now. This extremely fast successive-approximation ADC would lead to the same phase error as in linear analog elements with 60 kc 3 dB-frequency. The successive-approximation ADC starts each conversion from zero; i.e. it does not matter whether or not the input signals are continuous. Because the PHENOs are assumed to operate continuously, however, we can take advantage from this fact in order to increase conversion speed. An ADC which operates strictly on continuous signals is called a 'continuous converter.'

The most simple principle for continuous converters is that of using up-down-counters. Here, dynamic errors are almost negligible as long as the condition holds

$$\frac{d}{dt}(e_i) < \frac{e_{ref}}{2^n} \cdot f_c \qquad (8)$$

(n being the digital word length, f_c the counter clock frequency and $2^{-n} \cdot e_{ref}$ the step size).

When the input voltage slope is greater than the counter clock-frequency multiplied by the magnitude of the counter increment, the ADC is 'overrated,' resulting in a time-increasing error. f_c depends on the settling time required by the comparator and the analog switches. It is a function of the magnitude of the least significant increment.

It is difficult to find appropriate semiconductors for building analog switches with variable reference and 10^{-4}-accuracy (circuits with alloy chopper-transistors with high v_{BE} reverse voltage or circuits with field effect transistors have a relatively poor bandwidth). On the other hand very fast variable reference switches with 10^{-3}-accuracy are easy to implement. Therefore, it is of advantage to use a subranging technique by dividing the entire input signal range into k equidistant subranges. The actual subrange in which the input is falling is then expanded by a factor k (e.f., k = 16), so that a conversion accuracy of $k \cdot 10^{-4}$ is solely required. When the input crosses a boundary between two adjacent subranges the next one has to be selected. In this paper we will show a special technique by which that can be done within one clock interval. Of course, the comparator and the analog switches can now work much faster since they have to settle only to an error less than $k.10^{-4}$.

An other important measure in order to increase conversion speed is in using an automatically adapted register length; i.e., when the converter is going to be overrated n will be diminished. For example, when an actual input increment exceeds the magnitude of the least significant bit, this bit is dropped, resulting in a doubled step size, etc. By this means a constant bandwidth/accuracy ratio is obtained.

As we shall show in this paper, by using this advanced technique we designed PHENOs which have an accuracy-bandwidth performance suiting the most precise linear analog computing elements and exceeding the nonlinear ones.

Realization of the AD-continuous converter

Subranging

PHENOs operate in a ± 10 V-range. In order to be able to use fast current-switches of 0.1% accuracy, we divide the ± 10 V-range into 16 subranges. Therefore, each subrange covers 1,25 Volts of the input voltage. A schematic block diagram of a subranging converter is shown in Figure 2. In the subranging techniques one has to select, of all subranges, the one that includes the input voltage, e.g., by subtracting all lower subranges from the input signal. By the same operational amplifier the resulting difference is multiplied by the factor 8. Thus, the following subrange-ADC has to operate with the reduced accuracy of 0.1%. Subtraction, subrange-selecting D/A-conversion, and amplifying must be executed with an accuracy of 0.01% (with respect to the input). One subrange contains 1024 steps, which is the range of operations of the fast subrange-ADC. As long as the input remains within a specific subrange, it operates with the 6 mc clock frequency. When running into the adjacent subrange, a new subrange must be provided with an accuracy of 0.01%. This operation takes much more time (some microseconds) than the clock rate of the subranging converter (0.16 μs) gives. Therefore, considerable errors could arise due to the slowly operating 0.01%-accurate elements. The following section outlines measures that eliminate these errors, although slowly operating elements are used.

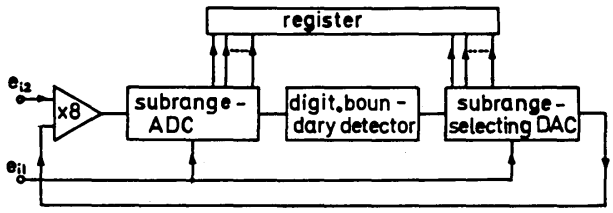


Figure 2—Schematic block diagram of a traditional ADC with subranging

Principles of subranging continuous-converters

When the input signal is crossing the boundary of one subrange, the converter has to switch over to the adjacent one. If the next subrange is prepared by a second subrange-selecting DAC, the required signal for the subranged-ADC is available at any instant. For a special reason (which shall be explained later), the subrange-ADC converts positive and negative input voltages. As long as the voltage is positive the next higher subrange is prepared. If it is negative, the adjacent lower subrange will be prepared.

Preparing of a new subrange is started when the input of the subrange-ADC crosses zero. Therefore, elements with 0.01 %-accuracy have a time interval available for reaching the steady state which is 1024 times the clock-interval of the subrange-ADC. Hence, the subrange-ADC operates over the full scale within subranges which are always in a steady-state, so that no

additional errors can occur. The price for this is in providing a second subrange-selecting DAC. Figure 3 shows the block diagram of the complete continuous converter.

The subrange-selecting DAC 'SSDAC I' is active, when even-numbered subranges are used, and 'SSDAC II' operates on odd-numbered ones. Whether or not a subrange is even-numbered is determined by flipflop FF 1, which switches the subrange-ADC to SSDAC I or SSDAC II. Furthermore, FF 1 marks which system is actually used and which one is in the preparing state. Usually, subranging ADCs, which are known from the literature,[2] have only one subrange-selecting DAC. When the input reaches the upper boundary of a subrange, the subrange-selecting DAC is set to the next higher subrange, and the ADC is reset to the lower boundary of that subrange. If the input voltage decreases or if there is an overflow in the DAC-system, the ADC must be switched back to the subrange just left. This may cause an instable operation, in cases when the input fluctuates around a subrange boundary.

This can be avoided by expanding the range of operation of the subrange-ADC to negative values. Nevertheless, the converter is set back to zero (and not to the lower boundary) when the input crosses the upper boundary. If the input is now decreasing, after just having left a subrange, the converter can use the negative part of the new range. Due to this principle the converter will always remain in the new subrange after a subrange-change and, therefore, stability at the transition-levels is secured.

If for example the subrange-ADC in Figure 4 reaches the value 16, it is reset to zero and 16 is added to the register of the subrange-selecting DAC. If the input decreases immediately after a subrange-change, the subrange-ADC continues to operate in the new subrange. Two special outputs ("Δe" and "i") are provided for a particular application described in section 4. Note that with "△e" we have in analog form the equivalent of the last 10 bits of the digital output, while at terminal "i" we shall get an impulse every time the input e_{in} is going to cross a subrange boundary.

Automatic step size adaption of the subrange-ADC

Formula 8 of chapter 1 reads:

$$\frac{d}{dt}(e_i) < \frac{e_{ref}}{2^n} \cdot f_c$$

In order to reduce the time-increasing errors of the

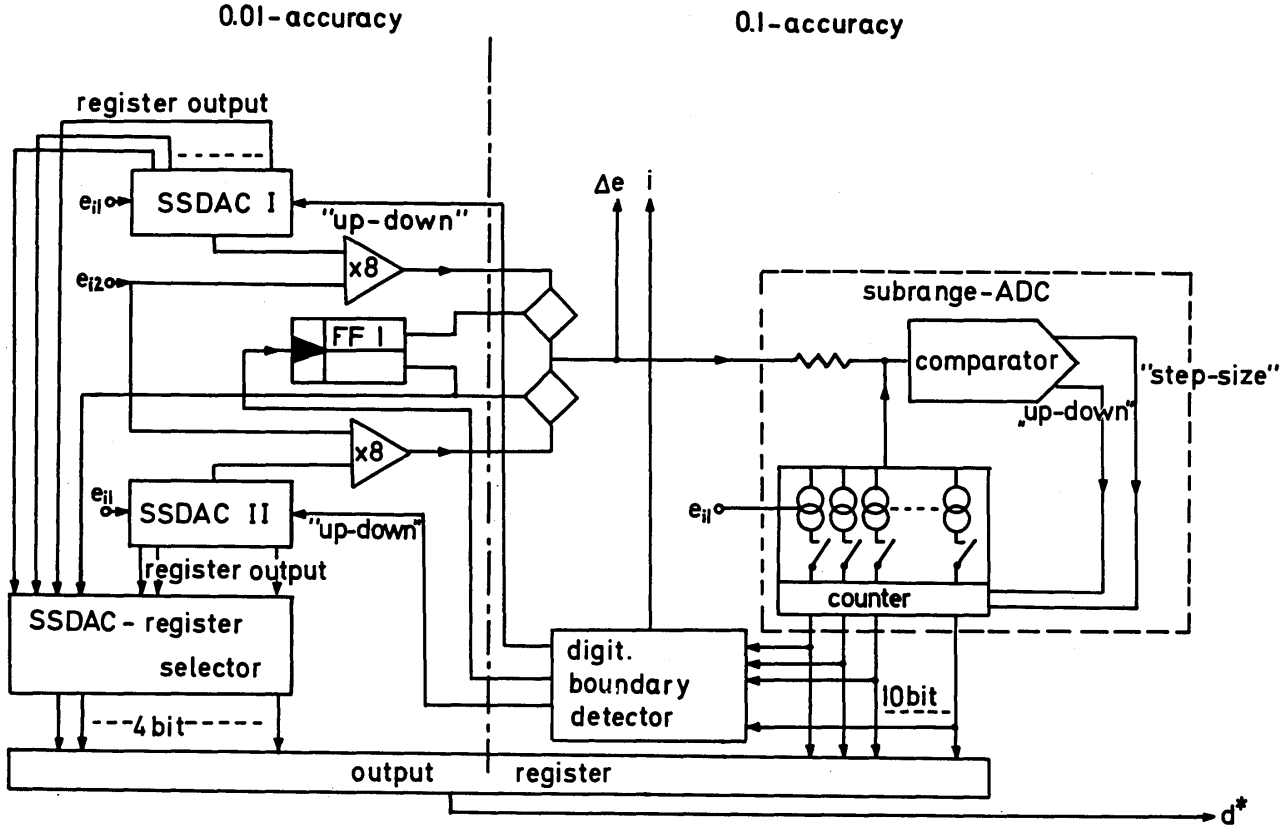


Figure 3—Block diagram of a PHENO-ADC

usual up-down-counter ADC we can either choose a higher clock frequency or a larger step size.

The transient function of most analog switches are almost exponential. Thus, the transient time for 0.01% accuracy is equal to 9 time-constants and the transient time for 0.1%-accuracy is equal to 5 time-constants. By doubling the clock-frequency the error would increase by the factor 100. On the other hand, errors increase only linearily with the step size, so that a constant error bandwidth ratio can be obtained.

The step size $\frac{e_{ref}}{2^n}$ of a counter-ADC depends on the magnitude of the reference voltage and the digital word length (register length) n. The static error of PHENO is less than 1mV. Assuming a 10V-reference voltage, we obtain a relative error of 0.01%. With decreasing reference voltage the step size decreases, too. While errors of the comparators and switches are constant, the maximal slope of the input voltage is reduced .

By an automatically adapted register-length the required step size can be matched to the slope of the input-voltage. The criteria for changing the step size can be obtained by measuring the summing-junction offset of the subrange-ADC. Traditional counter-ADCs detect whether the summing-junction has a positive or negative offset, and thus the counter is set in the 'up' or 'down' mode. Additionally, in our case the amplitude of the summing-junction offset is measured by two additional comparators. According to the used step size, the threshold voltages of these comparators have to be varied. A converter-'overrating' may occur under different circumstances. E.g.: if there is a positive step and

(i) a positive overflow, the stepsize has to be halved
(ii) no overflow, the stepsize has to be retained
(iii) a negative overflow, the stepsize has to be doubled.

For negative steps similar considerations will hold.

Hence, the sign of the last step has to be stored, and, in order to decide how to change the step size, that information has to be combined logically with the outputs of the overrate-detecting comparators.

Our AD-continuous converter has 8 possible step sizes, starting with 1mV. The other step sizes are given by successive doubling. The actual step size is stored in a 3-flipflop counter, which determine the register length and the threshold of the comparators.

Implementation of PHENOS

The elements

In table I existing standard types of PHENOs are listed. In addition to elements for AD- and DA- conversion, multiplication and division as well as function generation, special elements such as incrementing ADCs and accumulating DACs have been developed. They give the possibility of combining PHENOs with digital differential analyzer elements (DDAs). By combining them, new systems can be created, e.g., hybrid integrators, etc.

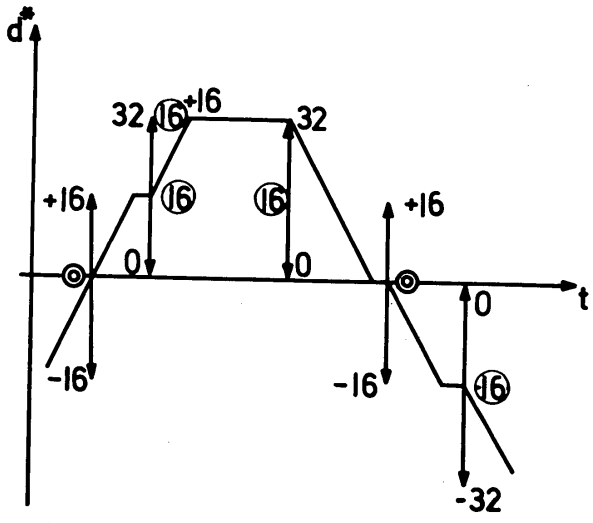


Figure 4—Scheme of subrange-transition
NOTE: The numbers at the arrows indicate the range of operation of the various subranges, the encircled numbers give the setting of the actual SSDAC

It should be emphasized that the possibility of "slaving" several elements also exists. For example, one ADC can be connected with two or more MDACs or FGDACs, giving the possibility of "multichannel" multiplication or function generation.

Analog switches and variable function generator circuitry

The operating speed of MDACs and ADCs depends mainly on the transient time of the analog switches. In our subranging continuous converter two types of analog switches are required: (i) a very fast one with 0.1%-accuracy and (ii) a very accurate switch (0.01%), which is uncritical in terms of speed.

Newly-developed, inexpensive variable reference current switches are meeting these requirements. The transient time of the simple, 0.1%-accurate switch plus comparator (μA 710) is 150 ns. The transient time of 0.01%-accurate switch is about 500 ns.

The principle of function generation has been explained in section 1. The most significant r bits of a digital number with length n define the breakpoints. The remaining (n-r) bits are used for linear interpolation.

The function-ordinates at the breakpoints are adjusted by potentiometers, which are sources with inductive output impedance. The switches must provide that the currents flowing in the potentiometers are not affected by switching (Figure 5).

As Figure 5 shows, the settings of potentiometers $P_1 \ldots P_r$ are not affecting one another. According to the digital number, a decoding matrix causes one pair of FETs, e.g., F_1 and F'_1, to be switched on and the transistors T_1 and T_2 to be switched off. All other switches are used the opposite way. If F_1 is switched on, the voltage $-V_1$ appears at the output of amplifier A_1. At the same time a current (proportional to the difference $(V_2 - V_1)$) flows through the FET'$_2$, resulting in the voltage $-(V_2 - V_1)$ at the output of amplifier A_2. $-(V_2 - V_1)$ acts as reference voltage of a MDAC which linearily interpolates between breakpoints.

The FET-switches 1 through r operate with an accuracy of 0.01%; their settling time for that accuracy is about 2μs. By inserting them in the feedback loop of an operational amplifier, errors due to their temperature-sensitive on-resistance are avoided. The FETs 1' through r' switch the difference voltages of two adjacent breakpoints to the summing-junction of the amplifier A_2. Here, the temperature-variable on-resistances of the FETs are part of the resistors which define the slopes of the straight-line segments. This is possible since the maximum function increments between adjacent breakpoints are limited and only small error propagation can occur.

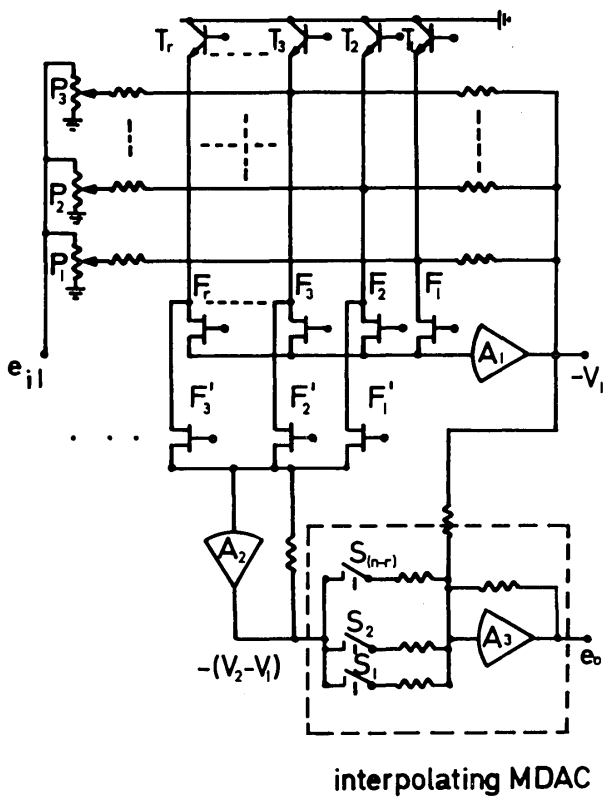


Figure 5—Circuitry of a FGDAC (schematic)

The transistors T_1 through T_r are in on-state when the related FET-switches are switched off. By that measure, discontinuities of the currents flowing in the potentiometers are avoided.

Incrementizer for linking hybrid and DDA sub-systems

DDA-elements are processing increments of digitally represented variables. Generally, these increments only have the values $+1$, -1, or 0, corresponding to the least-significant bit of the digital word.

When applying that DDA-technique to analog signals, increments smaller than the digital aperture may occur if the rate of change of the analog signal is too small (compared with the clock interval). Thus, the DDA may not give any response at all, though the analog signal can slowly vary over the entire scale. This difficulty can be avoided by accumulating consecutive increments and continuing that until the sum is going to exceed the threshold which corresponds to the least-significant bit. In our counting-ADC this procedure is performed by comparing the sum of all preceding output increments with the actual input. Thus, a counting-ADC can be used for linking analog and DDA-subsystems.

Hybrid integrators

Analog computers perform integration with respect to time. The generalized integration of arbitrary functions can be performed with a hybrid integrator. For this purpose integration is approximated by Euler's formula

$$z(x) = K \int_{x_o}^{x} y\,dx = Z_n(x) \qquad \begin{matrix} x = x(t) \\ y = y(t) \end{matrix} \qquad (9)$$

$$Z_n(x) = K \cdot \sum_{k=o}^{n} y_k \cdot \Delta x_k, \quad \Delta x_k = +1 \text{ or } -1 \qquad (10)$$

Referring to Figure 6, the hybrid integrator consists of two ADCs and one DAC. ADC I produces the increments +1 oder —1 (with respect to the clock period T^*_1). The analog switches S_1 and S_2 generate the product $(y_n \Delta x_n)$ which is added to the sum

$$K \cdot \sum_{k=o}^{n-1} y_k \Delta x_k$$

obtained before. The result is converted by ADC II which works with a higher clock rate. The output of ADC II is stored in DAC I (in digital form) and its analog equivalent is fed back to summer S. The resulting $Z_n(x)$ is available in digital as well as in analog form.

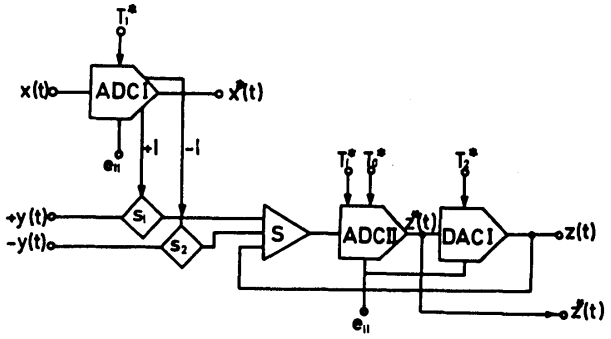


Figure 6—Hybrid integrator, programmed with PHENOs
NOTE: Asterisks denote digital variables

PHENOS in hybrid computer systems

For being linked with a digital computer, PHENO-equipped analog computers only require an inexpensive control interface but no data interface. Since all required analog-to-digital conversions are simultaneously performed by independent continuius converters, no errors due to the time shift of analog multiplexing or to the skewing time of sample-and-hold circuits can occur.[4] Moreover, PHENO-equipped analog computers as part of a combined hybrid computer system offer particular possibilities which do not exist in conventional hybrid systems.

As an example of great practical importance, we consider the problem of multivariable function generation. On the base of PHENOs, the well-known digital table look-up routine can be combined in a simple way with an analog interpolation, resulting in minimum execution time. Because of a direct analog path between input and output signals, the stability problems of pure digital function generation are avoided.

The generation of arbitrary functions is executed by straight-line-segment approximation. For example, in the case of a function of one variable we have the interpolation formula

$$f(x) \approx f_i + \frac{f_{i+1} - f_i}{x_{i+1} - x_i}(x - x_i)$$

$$\approx f_i + m_i(x - x_i) \quad , \qquad (13)$$

$$\text{for } x_i \leq x \leq x_{i+1} \; ; \; f_i = f(x_i) \; .$$

The breakpoint values f_i ($i = 0,1,2 \ldots,B\text{-}1$), B being the total number of breakpoints (e.g., $B=2^4=16$,) and the average slopes m_i are stored in the memory of the digital computer. The necessary multiplication of (m_i and $x—x_i$) now can be performed by a MDAC. Note that now, contrary to the FGDAC which was described in section 3.2, the slope is given in digital form while the function argument is analog. Hence, for the digital table look-up routine, the argument x first of all has to be converted by an ADC.

Because of the special subrange technique used in the PHENO-ADCs, the last 10 bits of the 14-bit output word are available in analog form. If the remaining 4 most-significant bits are used in order to define the function breakpoint (corresponding with $B = 2^4 = 16$), the last 10 bits or their analog equivalence, respectively, are identical to the increment $(x - x_i)$. Thus, we can use the very simple setup outlined in Figure 7.

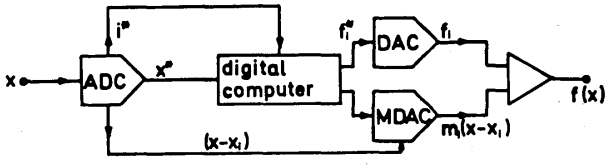


Figure 7—Hybrid generation of functions of one variable on the base of PHENOs

Here, the function argument x is fed into a PHENO-ADC and converted in digital form. Simultaneously,

the ADC gives out the analog signal $(x - x_i)$ which is fed into a MDAC in order to build the product $m^*_i (x - x_i)$.

The digital representation x^* of the function argument is transferred into the digital computer, but only the 4 most significant bits are required for the table look-up routine. By appropriate indexing these four bits can directly be used to give the (relative) addresses of the memory cells in which the corresponding breakpoint abscissa f_i and the average slopes m^*_i can be found.[3] Therefore, the digital program has to execute the following routine:

(i) Load x^* into accumulator;
(ii) Shift x^* to the right so that the most significant 4 bits give the relative address of the memory cell in which f^*_i is stored;
(iii) Load f^*_i into DAC;
(iv) Add B to the address. This modification gives the relative address of the memory cell in which m^*_i is stored.
(v) Load m^*_i into MDAC.

(In the case of the digital computer SDS 930 for which the above routine was programmed, the entire instruction sequence requires about 16 memory cycles $= 28\ \mu s$).

That digital table look-up routine is not necessarily part of the intrinsic hybrid computation. Since the ADC provides an interrupt signal ("i") when the input 'x is going to cross a subrange boundary (which is identical with crossing a function breakpoint), the hybrid computation can be interrupted. While the analog computer remains in the 'hold' mode, the actual values f_i^* and m_i^* can be replaced by f^*_{i+1} and m^*_{i+1}, and then, the hybrid computation may continue. During the real-time operation of the combined system there is an analog path between input and output of the function generation procedure. Hence, the stability problems are avoided which can arise when a digital computer is inserted in analog loops.[4]

We have to emphasize that this procedure is similar to the one previously proposed by A. I. Rubin.[5] But when PHENOs exist, our approach is simpler and less expensive.

The above principle can easily be expanded to the task of generating functions of more than one variable. If, for example, a function of two variables f(x,y) has to be generated, we use the interpolation formula*

$$f(x,y) = f_{i,k} + m^x_i (x - x_i) + m^y_k (y - y_k) \quad , \quad \text{for } x_i \leq x \leq x_{i+1}\ ,\ y_k \leq y \leq y_{k+1} \quad , \quad (14)$$

$$f_{i,k} = f(x_i, y_k)$$

which leads to the implementation given in Figure 8.

Now, for any breakpoint of the two-dimentional grid the values $f^*_{i,k}$, $m^x_i{}^*$, and $m^y_k{}^*$ have to be stored. An appropriate modification of the above outlined special indexing technique leads to a very efficient table look-up routine by which those values can be loaded into the DAC and the two MDACs (requiring now 38 μs on the SDS 930). Note that only one additional pair of ADC and MDAC is necessary for each additional input variable.

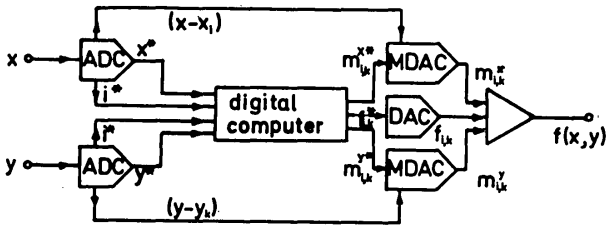


Figure 8—Hybrid generation of functions of two variables

Hybrid special purpose computers

PHENO offers a particular flexibility in building precise, inexpensive, miniaturized, hybrid special purpose computers. If only 'static' operations have to be performed (such as summing and subtraction, multiplication and division, function generation), a combination of PHENOs and analog summing amplifiers will be sufficient. The accuracy is strictly dependent on resistors, electronic switches, and operational amplifiers. Since all these components can be realized with small temperature drift, operation under extreme environmental conditions is possible. A high degree of integration and, therewith, miniaturization can be obtained.

In the case of 'dynamic' problems, i.e., problems which include integration, either the hybrid integrator of section 3.4 may be used or a combination of PHENOs with DDA-elements. The first approach is advisable if only a few integrations are required. For the second approach we shall give a typical example.

In case of a special purpose computer for solving a proportional navigation problem, the solution of the following equations is required

$$d\gamma_M = K \cdot d\sigma \quad (15)$$
$$K = K_o f(e) \quad (16)$$
$$\gamma_M = \Sigma d\gamma \quad (17)$$
$$du_1 = -u_2 d\gamma_M \quad (18)$$
$$du_2 = u_1 d\gamma_M \quad (19)$$
$$dV_y = V_M du_1 + u_1 dv_M \quad (20)$$
$$dV_x = V_M du_2 + u_2 dV_M \quad (21)$$

*Rubin has used a more sophisticated interpolation formula which gives a slightly better approximation but requires a more expensive hardware.

We assume that the input variables $d\sigma$ and dV_M are given as digital increments, the input e and the output γ_M are digital parallel words, and the output variables V_x and V_y are required as analog signals. Therefore, we may rewrite eqs. (16)-(21) as follows, denoting digital parallel information by asterisks.

	Components
$d\gamma_M = K^* \cdot d\sigma$	DDA-integrator
$K = K_o \cdot f(e^*)$	FGDAC
$K^* = K \cdot 1$	ADC
$\gamma^*_M = \Sigma d\gamma_M$	counter
$du_1 = -u_2 d\gamma_M$	DDA-integrator
$du_2 = u_1 d\gamma_M$	DDA-integrator
$dV_y = V_M du_1 + u_1 dV_M$	DDA-integrator
$dV_x = V_M du_2 + u_2 dV_M$	DDA-integrator
$V_y = \Sigma dV_y$	accumulating-DAC
$V_x = \Sigma dV_x$	accumulating-DAC

Figure 9 shows the resulting computer block diagram on the base of PHENOs according to Table 1 and DDA-components.

REFERENCES

1 T G HAGAN
AMBILOG computers: hybrid machines for measurement-system calculation tasks
Proc. 17th Annual ISA conf New York Oct 1962

2 DIGITAL EQUIPMENT CORPORATION
Analog/digital conversion handbook
Maynard Massachusetts 1964

3 G SCHWEIZER H SEELMANN
The application of hybrid simulation for VTOL-aircraft and certain reentry-problems
Proc. 4th Internat. Analogue Computation Meetings Brighton 1964 pp. 18-29

4 W GILOI
Error-corrected operation of hybrid computer systems
Paper presented at the 5th Internat. Analogue Computation Meeting Lausanne 1967

5 A I RUBIN
Hybrid techniques for generation of arbitrary functions
SIMULATION vol 7 no 6 Dec 1966 pp. 293-308

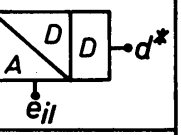
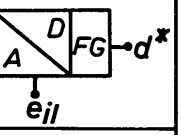
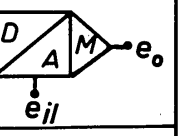
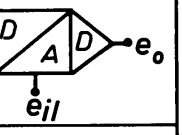
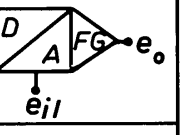
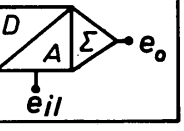
element	formula	symbol
ADC	$d^* = \frac{e_{i2}}{e_{il}}$	
FGADC	$d^* = f(\frac{e_{i2}}{e_{il}})$	
MDAC	$e_o = e_{il} \cdot d^*$	
DAC	$e = \frac{e_{il}}{d^*}$	
FGDAC	$e_o = e_{il} \cdot f(d^*)$	
Accumulat.-DAC	$e_o = e_{il} \cdot \sum_{K=1}^{n} \Delta d_K^*$	

Table I

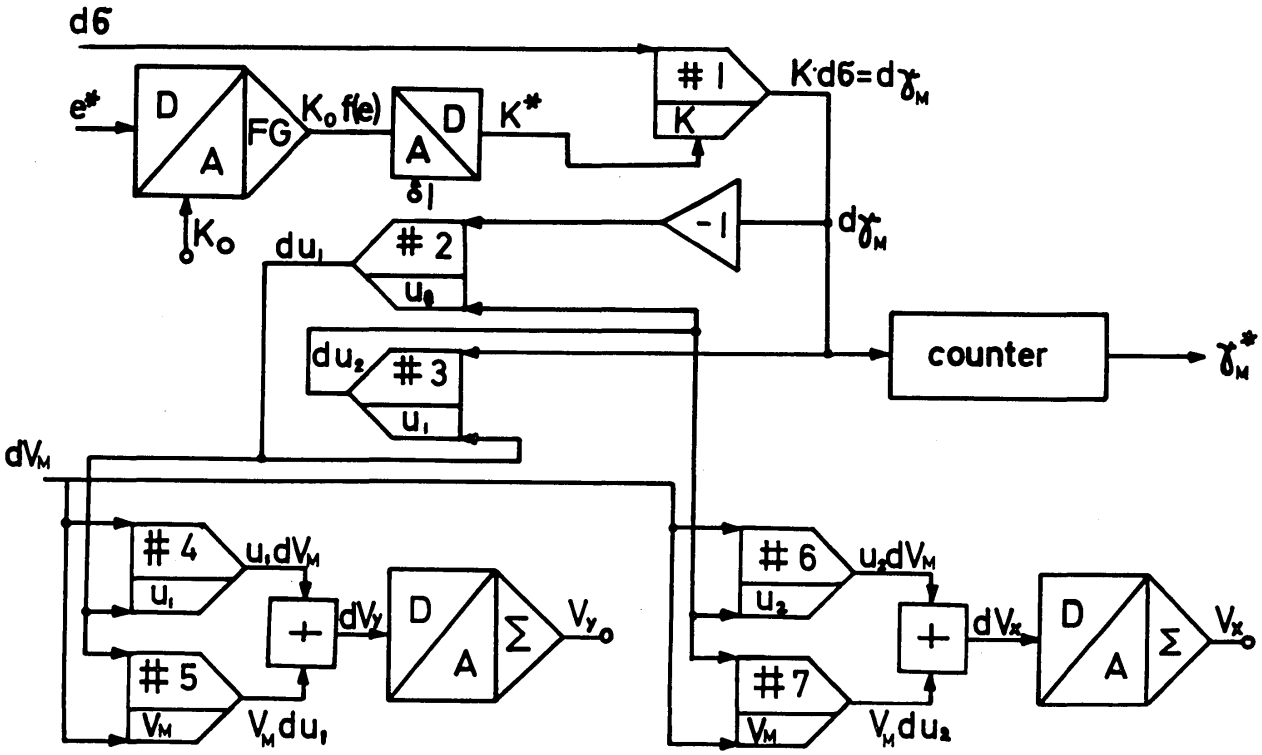


Figure 9—Block diagram of a special purpose computer, solving eqs. (13)-(19). The computer consists of PHENOs and DDA-elements