On the Numerical Integration of the State Equation

RUBEN CHAER, MEMBER, IEEE.

Abstract- In this paper, some general considerations are made on the stability of first-order integration algorithms for linear systems. An interesting result for the error of an algorithm under constant input is obtained. A method to calculate the critical stability time-step for the Euler algorithm is given. The step-response error for the Euler algorithm for one-dimension and two-dimension systems is also shown.

I. Introduction

We are developing a circuit simulator, called SIMEEP, specially oriented to simulate Power Electronic circuits (PECs). When modelling a PEC, we suppose that it can be represented by a set of switches and a set of linear components. We define the Logic State (LS) of the circuit as a boolean vector that determines the position of the set of switches. For each LS, the PEC has an associated linear circuit, which will have a State Equation (SE) of the form:

$$\dot{X} = A\,X + B\,r(t) \qquad \text{(SE)}$$

where X is a vector of reals that identifies the state of the linear circuit, r(t) is the vector of inputs, and A and B are constant matrices of the adequate dimensions. We will call X "the Real State" (RS).

At any time, the LS determines the linear circuit that represents the PEC, and the RS determines the state of this linear circuit. Therefore, the state of the PEC can be determined by the pair of vectors (LS,RS).

While the LS is constant (no commutations take place), we have a constant SE to integrate. When the LS switches to another value, we must find the new SE before continuing the integration.

This type of simulator, that in fact simulates a chain of linear circuits, must have very robust algorithms, because the probability of a successful simulation is conditioned by the success of the simulation of each link of the chain.

We are talking about circuits which have commutating components (e.g. diodes, thyristors, ...). We have modelled this type of components in such a way that the models themselves determine their logic state, taking into account the signals available at their terminals. The commutation condition is written as a linear combination of signals (at the components' terminals) which are linear combinations of the state variables. In order not to lose commutations, we must perfom the simulation with a time-step shorter than the shortest time-constant of the circuit. This time-step is in the order in which the Euler algorithm is effective. For this reason it was worth studying this simple algorithm.

II. The Discrete System

We have to integrate a SE of the type:

$$\dot{X} = A\,X + B\,r(t) \qquad (SE)$$

Performing the integration on a time-step basis T, we obtain a sequence of X for the instants t = k T of the form:

$$X_0, X_1, \ldots\ldots\ldots, X_k, \ldots\ldots$$

where X_k is the RS for the time t = k T.

Supposing that the input is constant during each integration interval [kT, (k+1)T], we obtain the next equation that we will call Discrete State Equation (DSE):

$$X_{k+1} = e^{AT}\,X_k + \left[e^{AT} - I \right] A^{-1}\,B\,r_k \qquad (DSE)$$

Observe that at first sight we need the inverse of A, but the expression:

$$\left[e^{AT} - I \right] A^{-1} = \left[\sum_{k=}^{\infty} \frac{1}{k!}\,A^k\,T^k - I \right] A^{-1} = \sum_{k=1}^{\infty} \frac{1}{k!}\,A^{k-1}\,T^k$$

is well defined, whether the inverse of A exists or not.

The major problem in using the DSE is calculating e^{AT} and $\left[e^{AT} - I \right] A^{-1} B$. No matter which method is used, we always obtain an approximation to these matrices, that we will call M and N respectively. Then, in order to perform the time-step based integration, we have an Integration Algorithm (IA) of the form:

$$Y_{k+1} = M\,Y_k + N\,r_k \qquad (IA)$$

This is a first-order algorithm, because Y_{k+1} depends only of Y_k and r_k. We choose Y_k to note the IA state, to differenciate it from the DSE state X_k.

III. The Error System

We define the error of the algorithm as the difference between the algorithm state and the discrete system state.

$$Ek = Yk - Xk \qquad \text{(Algorithm's Error)}$$

Substracting term by term the DSE to the IA, we obtain the Error State Equation (ESE), as follows:

$$Ek+1 = M\,Ek + \left(M - e^{AT} \right)Xk + \left[N - \left(e^{AT} - I \right)A^{-1}B \right] rk$$

Defining the vector:

$$uk = \left(M - e^{AT} \right) Xk + \left[N - \left(e^{AT} - I \right)A^{-1}B \right] rk$$

as the ESE input, we have:

$$Ek+1 = M\,Ek + uk \qquad \text{(ESE)}$$

From this equation we can conclude that, in order to obtain a bounded error for bounded values of Xk and rk, we must guarantee that all the eigenvalues of M have modulus less than one.

Observe that by conditioning M in this way, the stability of the IA is also guaranteed.

IV. Steady State Response of the ESE under Constant Input

Suppose that the SE's input rk is constant, i.e., $rk = r_\infty$ for $k = 0,1,2,.....$

Considering that the SE is stable, we can calculate the value of X_∞ when k tends to infinite as: $X_\infty = - A^{-1} B\, r_\infty$

The input of the ESE, uk, tends to:

$$u_\infty = \left[N - \left(M - I \right) A^{-1} B \right] r_\infty$$

In order to have $E_\infty = \emptyset$, we need that $u_\infty = \emptyset$, and this can be reached by making:

$$N - \left(M - I \right) A^{-1} B = \emptyset$$

Then, choosing:

$$N = \left(M - I \right) A^{-1} B$$

we have a null steady error under constant input:

$$E_\infty = \emptyset$$

From this result, we always choose $N = (M-I)A^{-1}B$, and then, our algorithms are only determined by M.

V. Stability

We have just shown that in order to have a stable algorithm (and a stable Error), we must ensure that all the eigenvalues of M have modulus less than one. When M is a polynomial of the matrix A, this stability condition can be written as a condition on the eigenvalues of A.

Suppose that:

$$M = p(A) = \sum b_k A^k$$

If λ is an eigenvalue of A, then, by definition,

$$\exists X \neq \emptyset \ / \ A X = \lambda X$$

It is easy to see that:

$$M X = p(A) X = p(\lambda) X$$

and therefore, $p(\lambda)$ is an eigenvalue of M.

The transformation $z = p(s)$, transforms a region $\mathbb{S}$ of the s-complex plane into the unit circle of the z-complex plane.

Let $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n$ be the set of eigenvalues of A, and $\zeta_1, \zeta_2, \zeta_3, \ldots, \zeta_n$ the set of eigenvalues of M.

Adequately ordering these sets, we can write:

$$\zeta_k = p(\lambda_k)$$

Then, all the eigenvalues of M are in the unit circle if all the eigenvalues of A are in the $\mathbb{S}$ region. Therefore, the stability of the algorithm is guaranteed if all the eigenvalues of A are inside the $\mathbb{S}$ region corresponding to the algorithm.

VI. The Euler Algorithm

In the Euler algorithm, $M = I + A\,T$, where T is the time-step.

The corresponding value for N is: $N = B\,T$

The matrix M can be written as a polynomial of the matrix A, in the form: $M = p(A)$, where the polynomial $p(s)$ is :

$$p(s) = 1 + s\,T\ .$$

The region $\mathbb{S}$ is the set of s for which modulus of $p(s)$ is less than one, i.e.,

$$\mathbb{S} \equiv \left\{ s \;/\; \left| 1 + T\,s \right| < 1 \right\}$$

The region $\mathbb{S}$ for a given time-step T is a circle with center in $s = -1 / T$ and radius $1 / T$, as it is shown in Fig.1.

Fig. 1. Region $\mathbb{S}(T)$

The region $\mathbb{S}$, is shown in reference [1] for the Euler and some other algorithms.

It can be demonstrated the following:

$$\text{if } T_1 < T_2 \text{ then } \mathbb{S}(T_2) \subset \mathbb{S}(T_1)$$

where $\mathbb{S}(T_1)$ is the corresponding $\mathbb{S}$ region for the algorithm $M_1 = I + A\,T_1$, and $\mathbb{S}(T_2)$ is the corresponding one for $M_2 = I + A\,T_2$.

Therefore, if the algorithm M_2 is stable, then the algorithm M_1 is also stable.

Besides, if the SE is stable (all the eigenvalues of A have negative real part), taking T small enough, the corresponding $\mathbb{S}$ region will include all the eigenvalues of A, and then the algorithm $M = I + A\,T$ will be stable.

In this way, we can find a critical stability time-step (T_{crit}) for which the algorithm:

$$M = I + A\,T \quad \text{is} \quad \begin{cases} \text{stable, when } 0 < T < T_{crit} \\ \text{unstable, when } T > T_{crit} \end{cases}$$

Therefore, building a function "f(T)" that only changes its sign in $T = T_{crit}$, it is easy to find T_{crit} using conventional methods (as the Bisection method)[2].

In order to build f(T), we need the characteristic polynomial of A, which we calculate using the Souriau (or Fadeev - Frame)[3] method, and which we call q(s).

In order to evaluate f(T), we make the following change of variable: $h(z) = q((z-1) / T)$. If all the roots of h(z) are in the unit circle, then all the roots of q(s) are in the region $\mathbb{S}$,

and we take f(T) = 1. On the other hand, if any of the roots of h(z) are outside the unit circle, then we take f(T) = -1. We apply the Jury criterion to the polynomial h(z) in order to determine how many roots h(z) has outside the unit circle.

The following is a possible Pascal definition of the function "f(T)" and the function "FindTcrit", included just to help understanding.

```
function f( T: real): real;
var h: Polynomial;
begin
     ChangeVar( h, q, T );      { h(z) = q( (z-1) / T) }
     if Jury( h ) > 0 then f:= - 1
     else f:= + 1
end;   { f }

function FindTcrit( A: matrix; Tmax: real ):real;
var  q: Polynomial;
begin
     q := CaracteristicPolynomial( A );
     FindTcrit := Bisection( 0, Tmax, f );
end;  { FindTcrit }
```

VII. Response of the Error System to the Step Input

A. *One-dimension system.*

Suppose the following stable system:

$$\dot{x} = -a\,x + b\,r(t) \qquad \text{" System "}$$

and the corresponding algorithm:

$$y_{k+1} = (1 - aT)\,y_k + bT\,r_k \qquad \text{" Algorithm "}$$

Instead of applying a positive step in r(t) , we suppose that the input has been constant enough time to reach the steady state, (remember that the steady state error is zero), and in t = ∅ we apply a negative step in r(t) driving it to zero.

$$r(t) = \begin{cases} r_c, & t < \emptyset \\ \emptyset, & t >= \emptyset \end{cases}$$

Initial conditions:

$$x(\emptyset) = x_o = a^{-1} b\, r_c$$

$$y(\emptyset) = y_o = x_o$$

The state trajectories are:

$x(t) = e^{-at} x_o$ to the system, and

$y_k = (1 - aT)^k\, x_o$ to the algorithm.

In the stability algorithm condition ($|\,1 - aT\,| < 1$), we can differenciate two cases:

Case 1: $\emptyset < 1 - aT < 1$

$$\emptyset < 1 - aT < 1 \;\leftrightarrow\; \emptyset < aT < 1$$

if $\emptyset < 1 - aT < 1$ then, $\exists\ \beta /\ e^{-\beta T} = (1 - aT)$, and therefore, the algorithm response is similar to the system response, with a time constant β instead of a.

The biggest error $\varepsilon_k = x(kT) - y_k$ takes place for k = 1 and its value is:.

$$\varepsilon_1 = x(T) - y_1 = (e^{-aT} - 1 + aT) x_o$$

We define the *relative local error* as:

$$\rho(aT) = \frac{\varepsilon_1(aT)}{x_o} = (e^{-aT} - 1 + aT) \quad \text{"Relative local error"}$$

For example, if we want a local error lower than 5%, we must take $aT < 0.334$.

Case 2: $-1 < 1 - aT < 0$

In the same way as in case 1, we define the local error:

$$\rho(aT) = \frac{\varepsilon_1(aT)}{x_o} = (e^{-aT} - 1 + aT) \quad \text{" Relative local error"}$$

Now $aT \in (1,2)$, and therefore, the relative error in this case is bigger than the one in case 1.

Added to the big relative errors (> 40%), exists an important difference between case 1 and case 2. In both cases, the error is exponentially decreased, but in case 2 it has alternatively sign + and sign - , whereas in case 1, the error has only one sign.

The sign inversion that takes place in case 2, may cause spurious commutations of the circuit switches, and must be avoided.

Fig. 2. shows the trajectories of the algorithm's state for T = Tcrit / Kdiv, taking Kdiv = 1, 3, 5, 7 and 9.

For Kdiv = 1, T = Tcrit and the algorithm has a non-damped trajectory. As Kdiv increases, the trajectory tends to the damped exponential.

Observe that the trajectories for Kdiv = 7 and Kdiv = 9 are

almost the same.

Fig. 2. One dimension system. Kdiv = 1, 3, 5, 7, 9

B. Two-dimension system

In this section, we analyse the transient response of a second-order system with a pair of complex conjugated eigenvalues.

Making the necessary coordinated transformation, the system can be written as:

$$\dot{X} = \begin{bmatrix} -\alpha & -\omega \\ \omega & -\alpha \end{bmatrix} X + B\, r(t) \ ,$$

where $-\alpha \pm j\,\omega$ are the two eigenvalues.

$$e^{\begin{bmatrix} -\alpha & -\omega \\ \omega & -\alpha \end{bmatrix} t} = e^{-\alpha t} \begin{bmatrix} \cos \omega t & -\text{sen } \omega t \\ \text{sen } \omega t & \cos \omega t \end{bmatrix}$$

The corresponding algorithm is:

$$Y_{k+1} = \begin{bmatrix} 1 - \alpha T & -\omega T \\ \omega T & 1 - \alpha T \end{bmatrix} Y_k = M\, Y_k$$

M can be written as a rotation-contraction, where the contracting constant is:

$$e^{-\alpha' T} = \sqrt{(1 - \alpha T)^2 + (\omega T)^2} \ ,$$

and the rotation matrix: $\begin{bmatrix} \cos \omega' T & -\text{sen } \omega' T \\ \text{sen } \omega' T & \cos \omega' T \end{bmatrix}$,

where: $$\cos \omega'T = \frac{1 - \alpha T}{\sqrt{(1 - \alpha T)^2 + (\omega T)^2}}$$, and

$$\text{sen } \omega'T = \frac{\omega T}{\sqrt{(1 - \alpha T)^2 + (\omega T)^2}}$$

As in the one-dimension system, we apply a negative step after a long-time constant input.

We define two local errors:

1-Contraction local error:

$$e^{-\alpha T} - \sqrt{(1 - \alpha T)^2 + (\omega T)^2}$$

2-Rotation local error:

$$\sqrt{(\cos \omega T - \cos \omega'T)^2 + (\text{sen } \omega T - \text{sen } \omega'T)^2}$$

To have both errors in small values, it is necessary that:

$$(\alpha T)^2 + (\omega T)^2 << 1.$$

In order to ensure the stability of the algorithm and have small errors, we can find T_{crit} and take as integration step $T = T_{crit} / Kdiv$, where Kdiv is a constant that we determine in an empirical way in order to solve the error-speed compromise. A good value is Kdiv = 10

In Figs 3, 4 and 5, we show the trajectories of a second

dimension system, taking as initial state $X_o^* = [\ 2,\ 2\ ]$ for $\alpha << \omega$ (Fig.3.), $\alpha = \omega$ (Fig.4.) and $\alpha > \omega$ (Fig.5.). As in the case of the one-dimension system the multiple traces correspond to Kdiv = 1, 3, 5, 7 and 9.

In the three figures, the non-damped trajectory corresponds to Kdiv = 1. Observe that here, too, the trajectories for Kdiv = 7 and Kdiv = 9 are almost the same.

Fig.3. $\omega = 1$, $\alpha = 0.1$

Fig.4. $\omega = 1$, $\alpha = 1$

Fig.5. $\omega = 1$, $\alpha = 2$

VIII. Conclusion

The application of the proposed method (automatically finding an integration time-step for the Euler algorithm for each logic state during the simulation) to the simulation of many circuits, has shown an important decrease of the total time spent in the simulations, compared with simulations performed with a constant time-step.

Due to the characteristics of the switched circuits, it is difficult to think about performing a filtering of the SE in order to be able to set a longer time-step for the simulation, because we run the risk of losing a commutation and consequently having qualitative differences between the simulator's output and the circuit behavior.

As a future work we are thinking in the possibility of performing changes of the state variables in order to uncouple the

SE in sets of equations that need different time-steps, and then perform the integration with different time-step for each set. However, it will be necessary to evaluate, (depending on the number of inputs, outputs, and state variables), how much time is really saved, because the outputs and some of the state variables will still have to be calculated with the shortest time-step.

References:

[1] DONALD A. CALAHAN, "Numerical Solution of Linear System with Widely Separated Time Constants", *proc. IEEE*, Nov. 1967, vol. 55 p.2016.

[2] L.V. ATKINSON, P.J. HARLEY, *"An Introduction to Numerical Methods with Pascal"*, Addison-Wesley. 1983.

[3] NOËL GASTINEL, *"Análisis Numérico Lineal"*, Reverté 1975.

Author: Ruben Chaer

Address: Av. Brasil 2995 / ap. 402

Montevideo. C.P. 11300 URUGUAY

Tel. #: 77 03 04

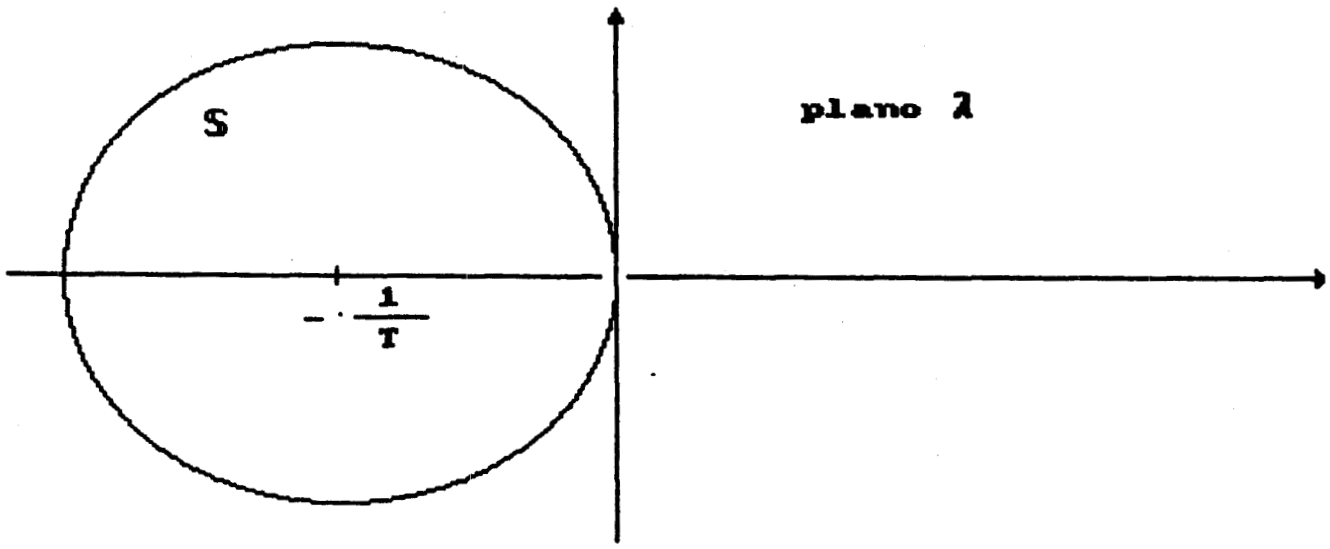

Fig. 1. Region $S_{(T)}$

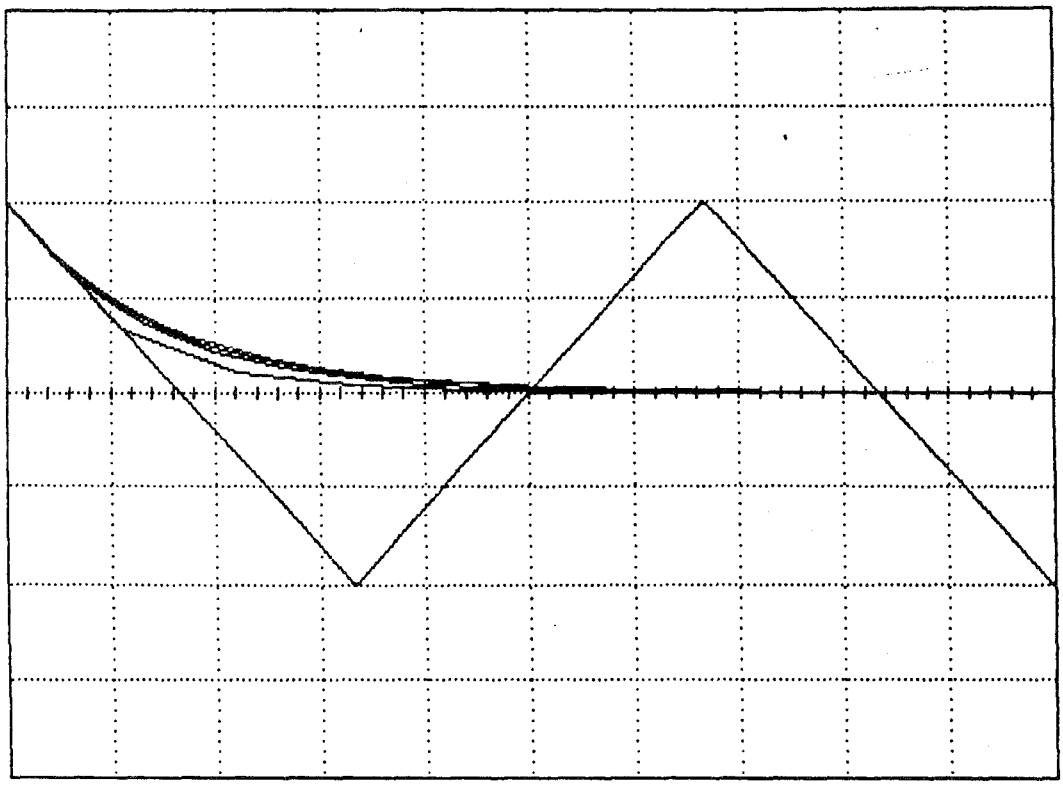

Fig. 2. One dimension system. Kdiv = 1, 3, 5, 7, 9

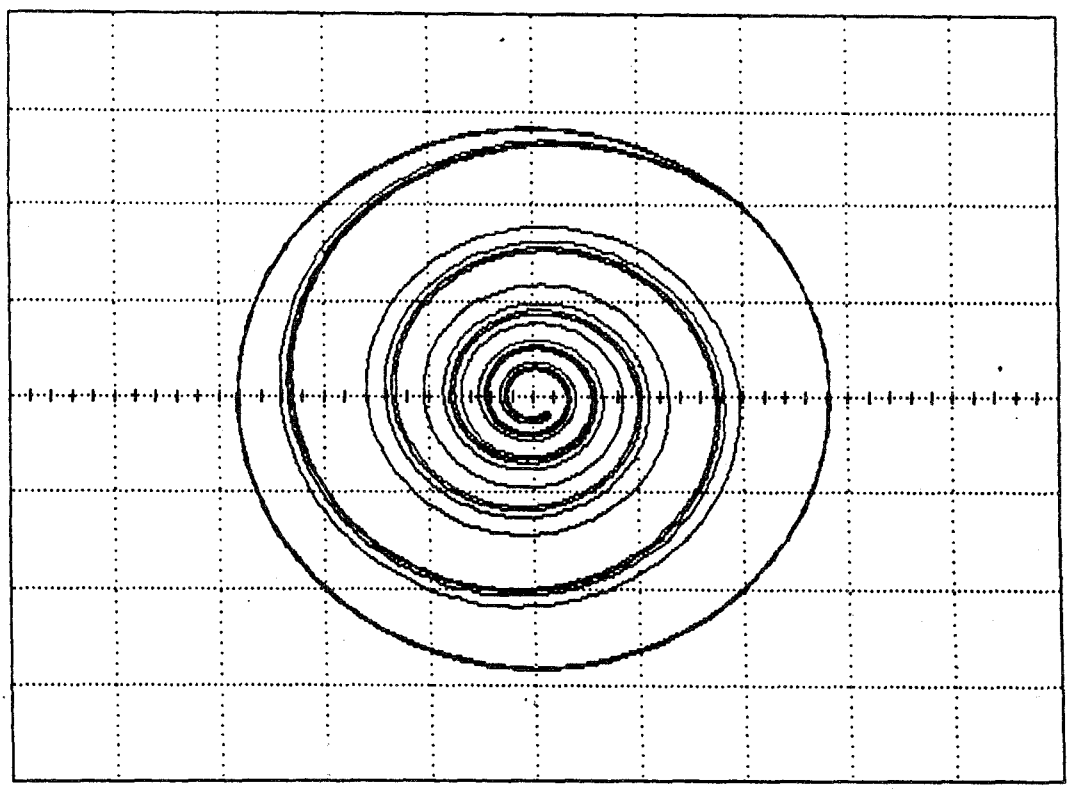

Fig. 3. $\omega = 1$, $\alpha = 0.1$

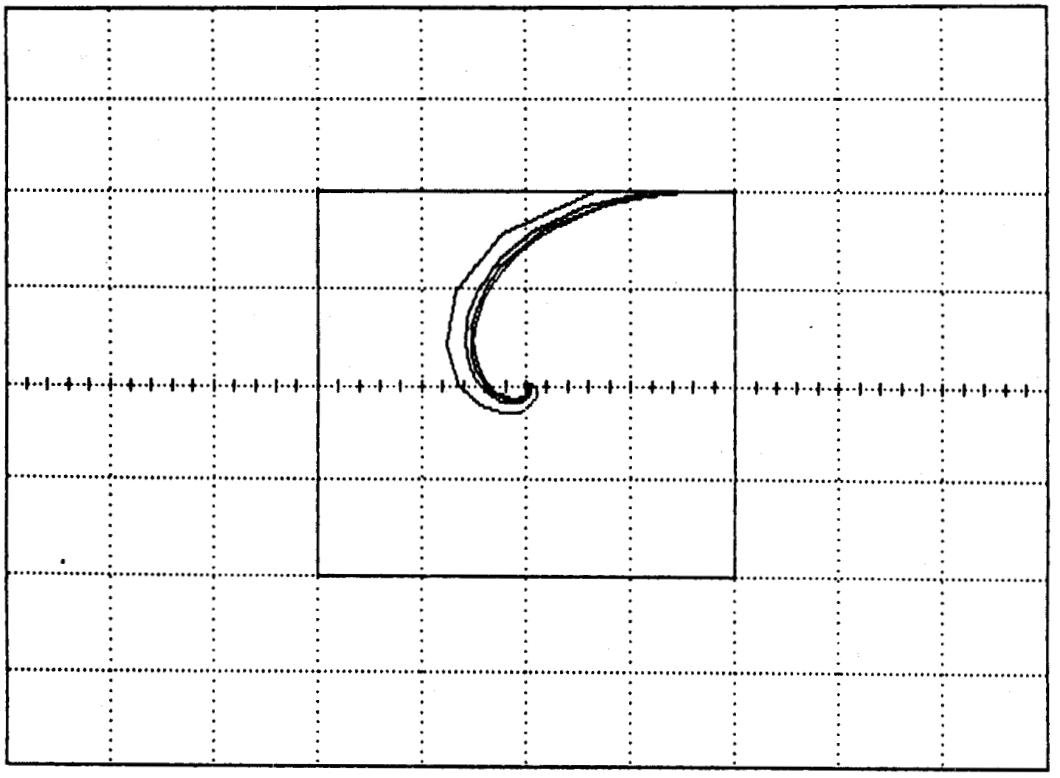

Fig. 4. $\omega = 1$, $\alpha = 1$

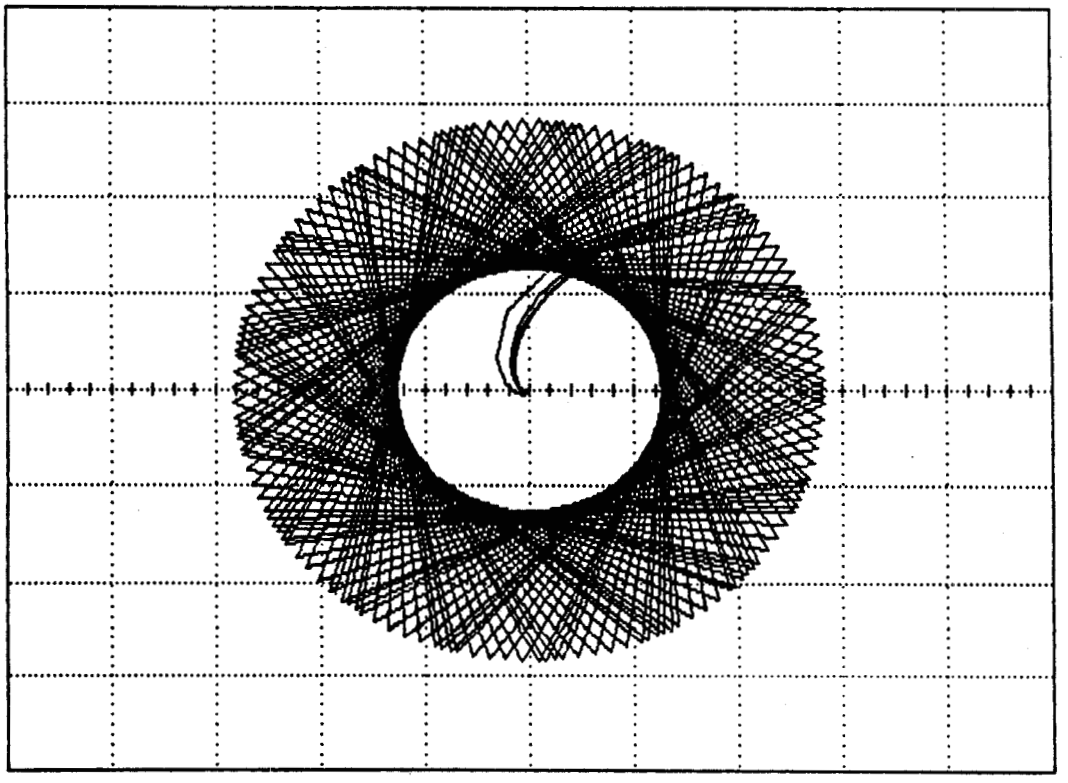

Fig. 5. $\omega = 1$, $\alpha = 2$