THE INVESTIGATION OF A NON-LINEAR DIFFERENTIAL EQUATION USING NUMERICAL METHODS

ALAN MORRIS CHRISTIE

SUMMARY OF A THESIS SUBMITTED FOR THE M.Sc. IN COMPUTING

BY ALAN M. CHRISTIE

THE INVESTIGATION OF A NON-LINEAR DIFFERENTIAL EQUATION USING NUMERICAL METHODS

The equation investigated was

$$y'' + ay' + 100y = cy^2$$

the parameters a and c being varied. The boundary conditions imposed upon the equation were

$$y(0) = 0, \quad y(t_m) = 1$$

where t_m was the position of the first maximum after the origin. It was most fully investigated for a 20, this being the region in which the solutions were exponentially decaying.

Although no analytic solution was discovered for the full equation, solutions were found when a = 0. By suitable transformations the solution for $c > 0$ was

$$y = (1-q)\ \mathrm{sn}^2\ (M(t-t_o),\ k) + q$$

where M, t_o, k and q were constants. For $c < 0$ the solution was

$$y = 1 - (1-q)\ \mathrm{sn}^2\ (M(t-t_o),\ k)$$

These, as might be expected were periodic solutions. The four numerical methods used were

(1) Finite Difference (2) Step-by-Step

(3) Picards (4) Perturbation

The first two were purely numeric and the second two, semi-analytic.

The Finite Difference technique was used to find the solution between the boundary values, and the Step-by-Step method then was used to integrate along the

curve until the value of y dropped to 0.01. The initial conditions for this latter method were found from the Finite Difference solution. Picard's Method and Perturbation which were used over the whole region both gave solutions in terms of exponential series. This series was of the form

$$y = \sum_{r=0}^{n} \sum_{s=0}^{r} A_{rs} \exp(-(s\alpha + (r-s)\beta)t$$

where the A_{rs}'s were constant coefficients and α and β were the exponents of t linear solution $y = A_{10}e^{-\alpha t} + A_{01}e^{-\beta t}$.

In all the methods except the Step-by-Step, the maximum had to be iterated onto by some means or another. In the Finite Difference method the secon point was adjusted until this condition had been satisfied. In the two semi-analytic approaches, the coefficients were in effect, altered to suit the condition.

There was good agreement in results between the boundary conditions for all methods, but as might be expected for large values of c, the accuracy out side this region was not good, when the numerical methods were compared with the semi-analytic. This was due to the fact that the semi-analytic solutions were essentially solutions expanded about a point. In comparing the two numeri solutions when the Finite Difference method was used over the whole region, there was good agreement.

In investigating this problem, I would like to thank, besides others, the staff of the Computing Department, and, in particular, Dr. Gilles, whose help has been invaluable to me. I would also thank him for the use of his AITKENROOT procedure which I used in one of my programmes.

CONTENTS

THE INVESTIGATION OF A NON-LINEAR DIFFERENTIAL EQUATION USING NUMERICAL METHODS

Chapter 1 - Initial Investigation

1.1. Introduction

Problems involving non-linear differential equations have been studied for over 200 years now, principally in connection with astronomy. Over the last few decades, however, interest in these equations has increas rapidly due to the fact that problems arising from, for example, hydrodyn elasticity and mechanics often involve non-linear equations. The field has also gained impetus since the war from the building of high-speed computers which are able to solve numerically all classes of differential equations. It was using the KDF 9, an English Electric-Leo-Marconi computer with 16,00 words of storage, that the equation below was investigated. The equation wa

$$\frac{d^2y}{dt^2} + a\frac{dy}{dt} + 100y = cy^2 \qquad (1)$$

the parameters a and c being varied. The boundary conditions were :

$$y(0) = 0 \quad \text{and} \quad y(t_m) = 1 \qquad (2)$$

t_m being the value of the abciesa at the first maximum after the origin. For values of a greater than 20, there was only one maximum and for $0 < a <$ the solution was sinusoidal. No analytic solution was found for the complete non-linear equation, but when a was zero, an analytic solution was obtained. This was expressed, as in the solution of many non-linear equations, in terms of elliptic functions.

The methods of finding solutions for the complete equation were as follows :

(a) Finite Difference (b) Step by Step

(c) Perturbation (d) Picard's

All four methods used the computer to a greater or lesser extent. The first two methods were purely numerical while the latter two were semi-analytic and produced a series approximation to the true solution.

1.2. The Linear Equation $y'' + ay' + 100 = 0$

For values of c which make the equation highly non-linear, the solutions do not differ radically in shape from the solution of the linear equation ($c = 0$). It is therefore instructive to have a look at this equation. We have

$$\frac{d^2y}{dt^2} + a\frac{dy}{dt} + 100y = 0 \qquad (3)$$

For $a > 20$, the solution has the form

$$y = A_1e^{-\alpha t} + B_1e^{-\beta t} \qquad (4)$$

where $-\alpha$ and $-\beta$ are the roots of the equation

$$x^2 + ax + 100 = 0 \qquad (5)$$

For values of a between zero and twenty the solution has the form

$$y = e^{-\gamma t}(A_2\sin \zeta t + B_2\cos \eta t) \qquad (6)$$

where ξ and η are the imaginary parts of the solution of (5) and $-\gamma$ is the real (negative) part.

Combining (4) with the first condition of (2) gives that

$y(0) = A_1 + B_1 = 0$ i.e. $B_1 = -A_1$

From the second condition of (2) we have, since t_m is the position of a turning point, $y'(t_m) = 0$

$$\therefore y'(t_m) = \left[\frac{d}{dt} A_1(e^{-\alpha t} - e^{-\beta t})\right]_{t=t_m} = -A_1(\alpha e^{-\alpha t_m} - \beta e^{-\beta t_m}) = 0$$

$$\therefore \alpha/\beta = \exp[(\alpha-\beta)t_m]$$

$$\therefore \quad t_m = \frac{1}{\alpha-\beta}\ln(\alpha/\beta) \qquad (7)$$

Substituting this value back into the equation $y(t_m) = 1$ will give

$$A_1\left\{\exp\left[\frac{-\alpha}{\alpha-\beta}\ln(\alpha/\beta)\right] - \exp\left[\frac{-\beta}{\alpha-\beta}\ln(\alpha/\beta)\right]\right\} = 1$$

After some algebra this gives

$$A_1 = 1/\left\{k^{k/(k-1)} - k^{1/(k-1)}\right\} \qquad \text{where } k = \alpha/\beta \qquad (8)$$

In a similar way it can be shown that with the same boundary conditions, the sinusoidal case ($0 < a < 20$) has the solution

$$y = A_2 e^{-\gamma t}\sin\omega t \qquad \text{where} \qquad (9)$$

$$A_2 = \exp(\tan^{-1}(k)/k)/\sin(\tan^{-1}k), \quad k = \gamma/\omega = \gamma/\xi$$

The solutions of the two cases are therefore similar to figures 1 and 2.

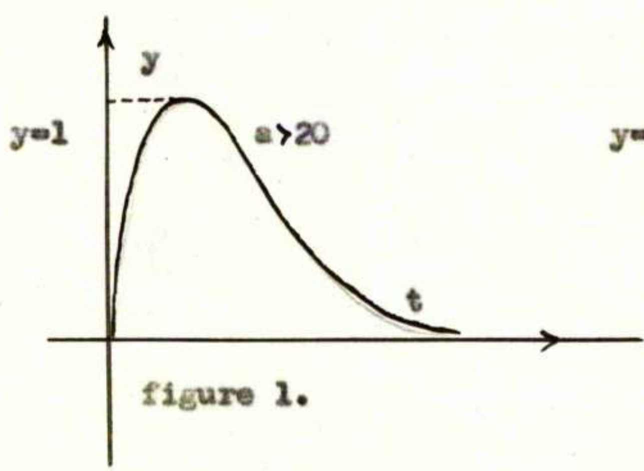


figure 1.

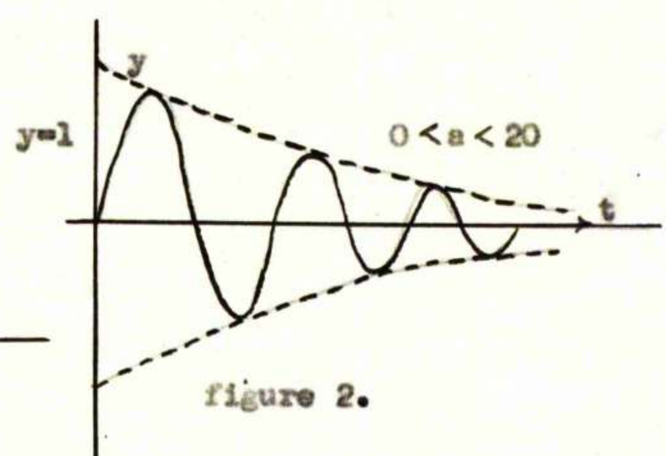


figure 2.

If $-20 < \underline{a} < 0$, then $-\gamma$ would be positive, and if $\underline{a} \leqslant -20$, then $-\alpha$ and $-\beta$ would be positive. Therefore for all cases $\underline{a} < 0$ there are positive exponentials and as $t \to \infty$, so will y. Therefore the cases $\underline{a} < 0$ were not investigated. The coefficient of y was given the value 100 so that for large a i.e. of the order 100, the initial gradient would be substantial (approximately 100 for $\underline{a}$ = 100) and interesting questions of accuracy would arise. For small values of a i.e. $0 < \underline{a} < 20$, a rapidly varying function of the type in figure 2 would result, and similar questions of accuracy would have to be investigated.

Of the methods used, only the numerical methods were applicable for all values of a, the semi-analytic methods only being used for values of $\underline{a} > 20$ but because they involved considerably more algebra for their solution, it was decided only to touch on them.

1.3. An Analytic Solution to $y'' + 100y = cy^2$

In order to obtain some idea of the solution to the non-linear equation it was decided to investigate, analytically, as much as possible of it, and find out something of its nature. Although little was found analytically about the full equation, by making a zero it was found to have, at least for some values of c, an analytic solution. For a=0 in the linear case the solution did not die away i.e. there was no 'frictional' term in the equation, and by investigating the non-linear equation with a=0, it looked as though this was also the case. This can be seen by an investigation of the phase curves i.e. a graph of y against dy/dt.

If we let v = dy/dt then we have

$$\frac{d^2y}{dt^2} = \frac{d}{dt} v = \frac{d}{dy}(v)\frac{dy}{dt} = v\frac{dv}{dy} = \frac{d}{dy}(\tfrac{1}{2}v^2)$$

$$\frac{d^2y}{dt^2} + 100y = \frac{d}{dy}(\tfrac{1}{2}v^2) + 100y = cy^2$$

$$\tfrac{1}{2}v^2 + \frac{100}{2}y^2 = \frac{c}{3}y^3 + \text{constant}$$

or

$$v^2 = \frac{2c}{3}y^3 - 100y^2 + \text{constant}$$

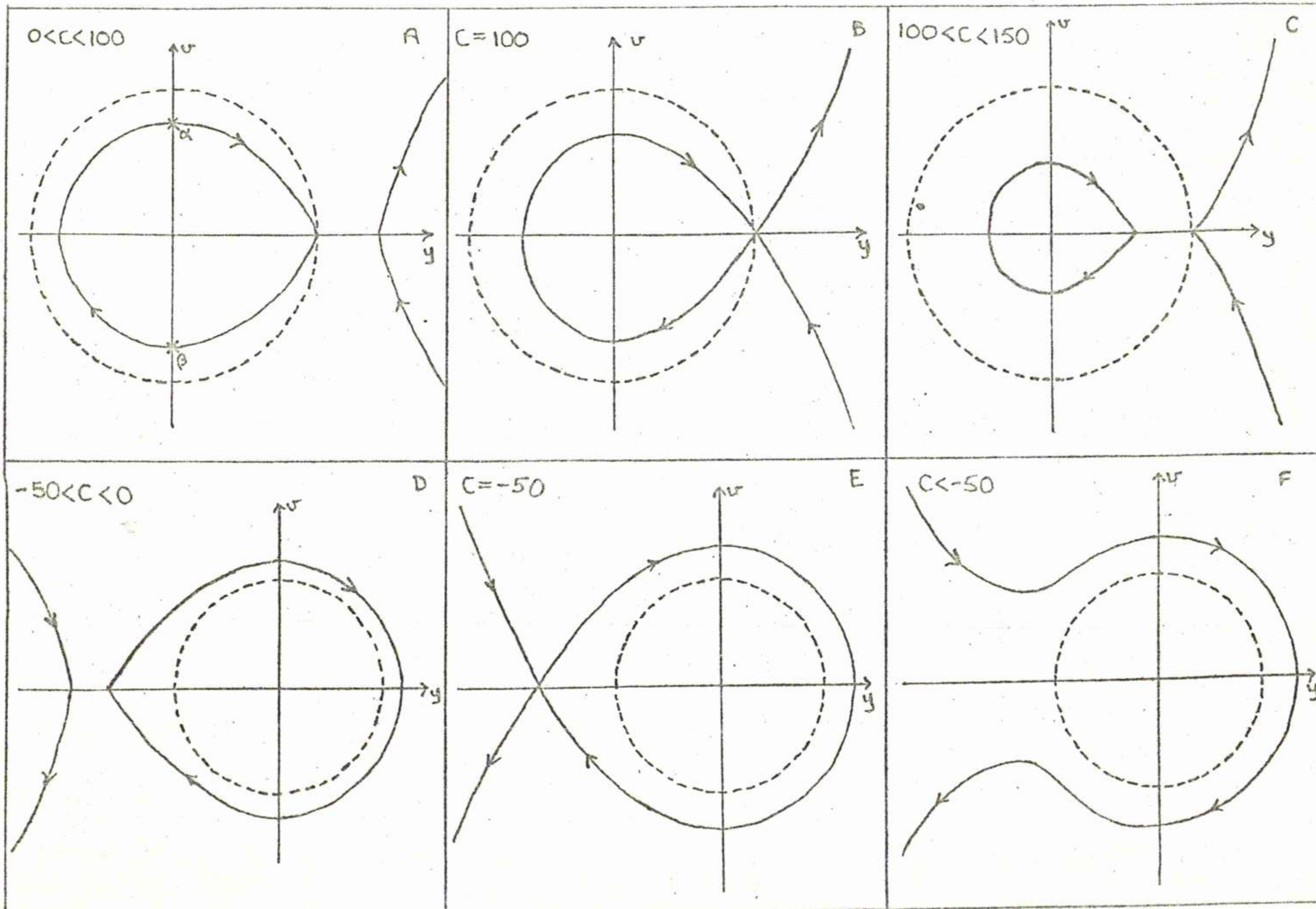
Since v = dy/dt = 0 when y=1, we have

$$\text{constant} = 100 - 2c/3$$

$$v^2 = \frac{2c}{3}(y^3-1) - 100(y^2-1) \qquad (10)$$

An insight can be gained into the solutions by plotting the phase curves for different values of c. Figure 3 shows these curves. In all cases the dotted circles represent the linear case. In actual fact, these are ellipses but the scales have been altered for convenience.

Figure A shows the curve for $0 < c < 100$. As c gets larger, the stable part of the curve, which starts off as an ellipse at $c=0$, becomes distorted, and the unstable part of the curve (on the right hand side) approaches the value y=1 on the y-axis. The values of the function lying between y=1 and the unstable curve were found to be complex and hence there was real solution in this region. On integrating the function to give the full solution, the stable curves were found to be functions of the elliptic sin. No solutions, neither analytic nor numerical, were found for the unstab



0<C<100
A
C=100
B
100<C<150
C
-50<C<0
D
C=-50
E
C<-50
F
v
y
α
β

regions due to the lack of knowledge about their boundary conditions. For these curves, we know their minimum y values but we require another boundary condition to define them uniquely. As this was not forth coming from the elliptic sin solution, there is a good chance that these unstable curves do not have any real existence in the fully integrated equation.

Figure B shows the intermediate case of $\underline{c}=100$. This is the only case for $\underline{c}>0$ where the two curves join. Since when the gradient is positive i.e. $dy/dt>0$, y must be increasing, the directions of this curve, as with the others, can only be as shown.

For values of c between 100 and 150, the stable loop contracts until at the value of 150, it disappears at the origin. The unstable part, however, for $\underline{c}>100$, remains and always goes through the point (1,0). For $\underline{c}>150$, the total integral curve consists solely of this line.

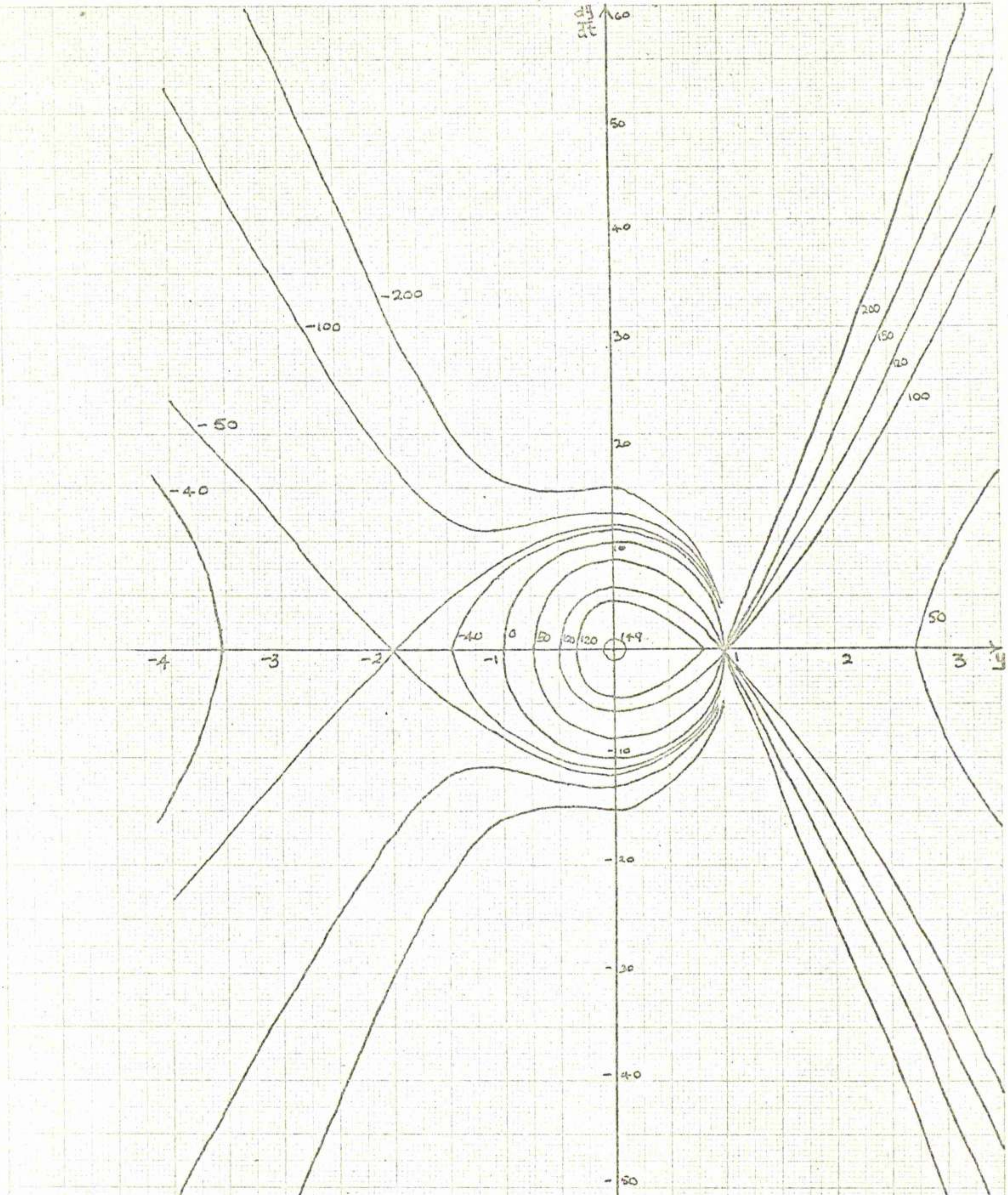
For negative values of c, phase curves similar to those for $c>0$, emerge except that they are in the reverse direction and the stable loops enclose the ellipse rather than being enclosed by it. As can be seen, for $\underline{c}=-50$, the two distinct curves join, but unlike the case for $\underline{c}>100$, they do not separate again, but open out to form an unstable path. The total phase diagram is shown in graph 1.

From equation (10) we have

$$\left(\frac{dy}{dt}\right)^2 = \frac{2c}{3}(y^3-1) - 100(y^2-1)$$

$$= (y-1)\left[\frac{2c}{3}y^2 + \left(\frac{2c}{3}-100\right)y + \left(\frac{2c}{3}-100\right)\right]$$

$$= \frac{2c}{3}(y-1)(y-p)(y-q) \qquad (11)$$



dy/dt
60
50
40
30
20
10
-10
-20
-30
-40
-50
-60
-4
-3
-2
-1
1
2
3
y
-200
-100
-50
-40
0
50
100
120
149
200
150
100
50

Graph 1

where p and q are the roots of the above quadratic i.e.

$$p, q = \frac{1}{2}\left[\left(\frac{150}{c} - 1\right) \pm \frac{3}{c}\sqrt{(50+c)(50-c/3)}\right] \qquad (12)$$

Thus t is given by

$$t = \sqrt{\frac{3}{2c}} \int \frac{dy}{[(y-1)(y-p)(y-q)]^{\frac{1}{2}}} + \text{constant} \qquad (13)$$

This integral can only be evaluated using elliptic functions except for two special cases when c takes the values 100 and -50. In order to integrate the above, some properties of elliptic functions have to be known. The elliptic integral of the first kind is defined as

$$F(x, k) = \int_0^x \frac{dx}{[(1-x^2)(1-k^2x^2)]^{1/2}} \quad , k^2 < 1 \qquad (14)$$

This is the only kind of elliptic integral involved in this problem. It is obvious that $F(x,-k) = F(-x,k) = F(x,k)$.

If $k = 0$ then from the above integral

$$F(x,k) = F(x,0) = \sin^{-1}x$$

Generalising we have that

$$F(x,k) = sn^{-1}(x,k) = u \text{(say)}.$$

Thus $$x = sn(u,k).$$

If we make the substitution $x = \sin\phi$ then (14) above becomes

$$F(\phi, k) = \int_0^\phi \frac{d\phi}{(1-k^2\sin^2\phi)^{1/2}} \qquad (15)$$

From this we can define the other elliptic functions using their trigonometric counterparts

$$sn(u,k) = \sin\phi, \qquad cn(u,k) = \cos\phi,$$

$$\mathrm{dn}(u,k) = (1-k^2 \sin^2 \phi)^{\frac{1}{2}}, \quad \mathrm{am}(u,k) = \phi,$$

$$\mathrm{tn}(u,k) = \mathrm{sn}(u,k)/\mathrm{cn}(u,k) = \tan \phi. \qquad (16)$$

Using suitable transformations, equation (13) can be transformed into equation (14) and hence the elliptic type solution.

For $c > 0$, we use the transformations

$$\left.\begin{aligned} z^2 &= (y-q)/(1-q) \\ k^2 &= (1-q)/(p-q) \\ M^2 &= c(p-q)/6 \end{aligned}\right\} \qquad (17)$$

From the above we have that $2z.dz = y/(1-q)$. Substituting (17) into (13) gives

$$t = \pm \frac{1}{M} \int \frac{dz}{[(1-z^2)(1-k^2z^2)]^{1/2}} + t_0$$

where t_0 = constant.

$$\therefore (t-t_0)M = \pm \int \frac{dz}{[(1-z^2)(1-k^2z^2)]^{1/2}} \qquad (18)$$

From (14) and (16) it can be seen that the solution is therefore

$$z = \mathrm{sn}(\pm M(t-t_0), k) \qquad (19)$$

Substituting y into this equation gives

$$\begin{aligned} y &= (1-q)\,\mathrm{sn}^2(\pm M(t-t_0), k) + q \\ &= (1-q)\,\mathrm{sn}^2(M(t-t_0), k) + q \end{aligned} \qquad (20)$$

since $\mathrm{sn}(-x,k) = -\mathrm{sn}(x,k)$.

To find the value of t_0 we have that $y(0) = 0$.

$$\begin{aligned} \therefore\ 0 &= (1-q)\,\mathrm{sn}^2(Mt_0, k) + q \\ \therefore\ \mathrm{sn}(Mt_0, k) &= \pm\sqrt{\frac{-q}{1-q}} = \pm x \text{ (say)} \end{aligned} \qquad (21)$$

From the fact that $F(x,k) = sn^{-1}(x,k)$, we have

$$Mt_o = \pm F(x,k) \tag{22}$$

The ambiguity in sign can be explained from diagram A of figure 3. The initial conditions stipulate that $y(0) = 0$ but they do not say if the gradient at $y(0)$ is positive or negative i.e. if the curve starts from the point α or β. If t_o is given a negative value then the curve starts at α and vice versa. Therefore t was given the negative value.

Since the transformation (17) is only applicable for positive c we must find the equivalent for $c < 0$. For this case

$$\left(\frac{dy}{dt}\right)^2 = \frac{-2c}{3}(1-y)(y-p)(y-q) \tag{23}$$

and the transformation becomes

$$\left.\begin{aligned} z^2 &= (1-y)/(1-q) \\ k^2 &= (1-q)/(1-p) \\ M^2 &= 6/c\ (1-p) \end{aligned}\right\} \tag{24}$$

In a similar way to $c > 0$ we find that on substituting (24) into (23) gives the solution

$$y = 1 - (1-q)sn^2(M(t-t_o),k) \tag{25}$$

where $$Mt_o = \pm F(x,k)\ , \quad x = \sqrt{\frac{1}{1-q}} \tag{26}$$

If we desire the gradient at the origin to be positive in this case we must take the positive value of t_o. The solutions for c = -40, 0 and +75 are shown in graph 2.

The solutions for c = 100 and -50 were found analytically without using elliptic integrals and it was found, surprisingly, that in these cases, the function took values greater than 1. These values correspond to the unstable arms of the phase diagram (graph 1) for the respective values of c. For c = 100 we have

$$\left(\frac{dy}{dt}\right)^2 = (y-1)\left[\frac{2c}{3}y^2+\left(\frac{2c}{3}-100\right)y-\left(\frac{2c}{3}-100\right)\right]$$

$$= \frac{100}{3}(y-1)^2(2y-1) \tag{27}$$

$$t = \frac{\sqrt{3}}{10}\int\frac{dy}{(y-1)\sqrt{(2y-1)}} + \text{constant} \tag{28}$$

Integrating with the boundary condition y(0) = 0 gives

$$t = \pm\frac{1}{10}\ln\left\{\left|\frac{\sqrt{2y+1}-\sqrt{3}}{\sqrt{2y+1}+\sqrt{3}}\right|\cdot\frac{\sqrt{3}+1}{\sqrt{3}-1}\right\} \tag{29}$$

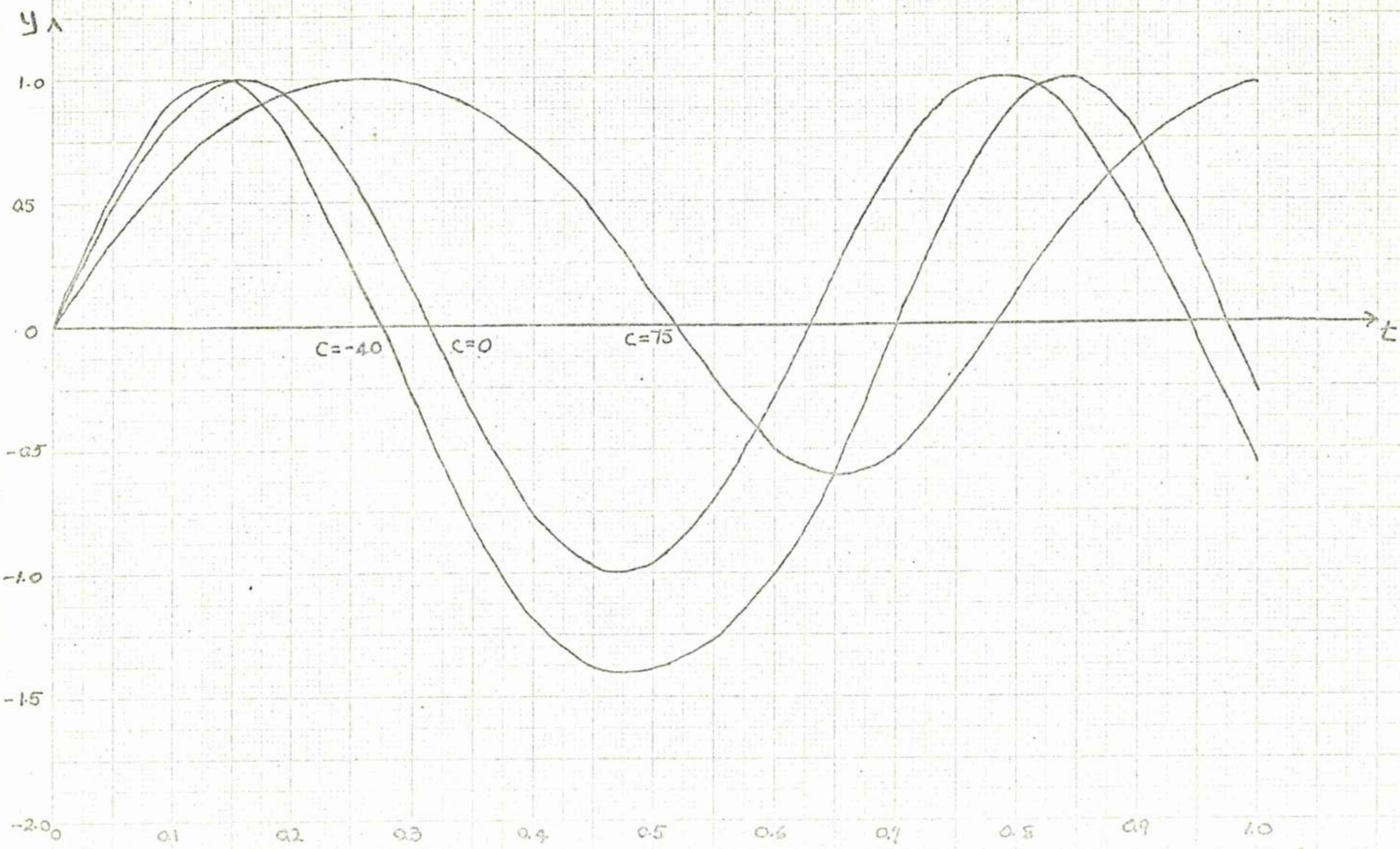
For c = -50 we have

$$\left(\frac{dy}{dt}\right)^2 = \frac{-100}{3}(y-1)(y+2)^2 \tag{30}$$

If we substitute z = -2y, then the above equation becomes the same as for c = 100. Thus the solution of (27) is

$$t = \pm\frac{1}{10}\ln\left\{\left|\frac{\sqrt{1-z}-\sqrt{3}}{\sqrt{1-z}+\sqrt{3}}\right|\cdot\frac{\sqrt{3}+1}{\sqrt{3}-1}\right\} \tag{31}$$

Equation (29) is shown in graph (4). The shape of equation (31) can be



Graph 2

seen by substituting -2t for t in graph (4)

1.4. Nature of Solutions for Different Values of c

It is not possible to find solutions for all the integral curves in graph 1. If p and q are plotted against c, (graph 3), it can be seen that for $-50 \leqslant c \leqslant 150$ p and q have real values but otherwise they are complex.

As c tends to zero, p tends to plus or minus infinity depending upon the sign of c and q tends to -1. In this case k tends to zero and, as would be expected from the linear equation, a sinusoidal solution results. Since $0 < k^2 < 1$ we have from (17)

$$0 < (1-q)/(p-q) < 1$$

i.e. $(1-q) < (p-q)$ or $p > 1$

Since $p = \frac{1}{2}\left[\left(\frac{150}{c} - 1\right) + \frac{3}{c}\sqrt{(50+c)(50-\frac{c}{3})}\right] > 1$

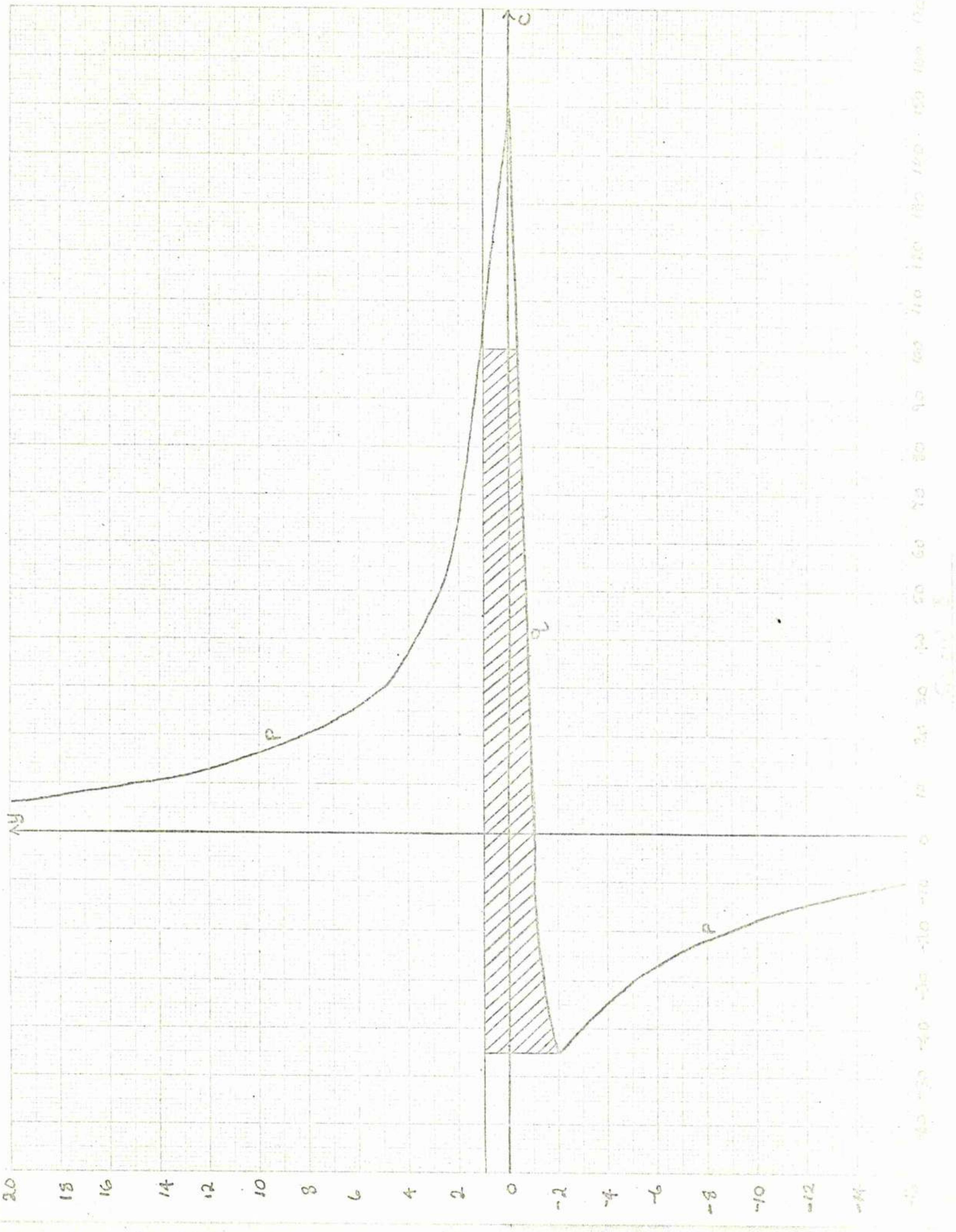
we have $\frac{3}{c}\sqrt{(50+c)(50-\frac{c}{3})} > 3 - \frac{150}{c} = \frac{3}{c}(c-50)$

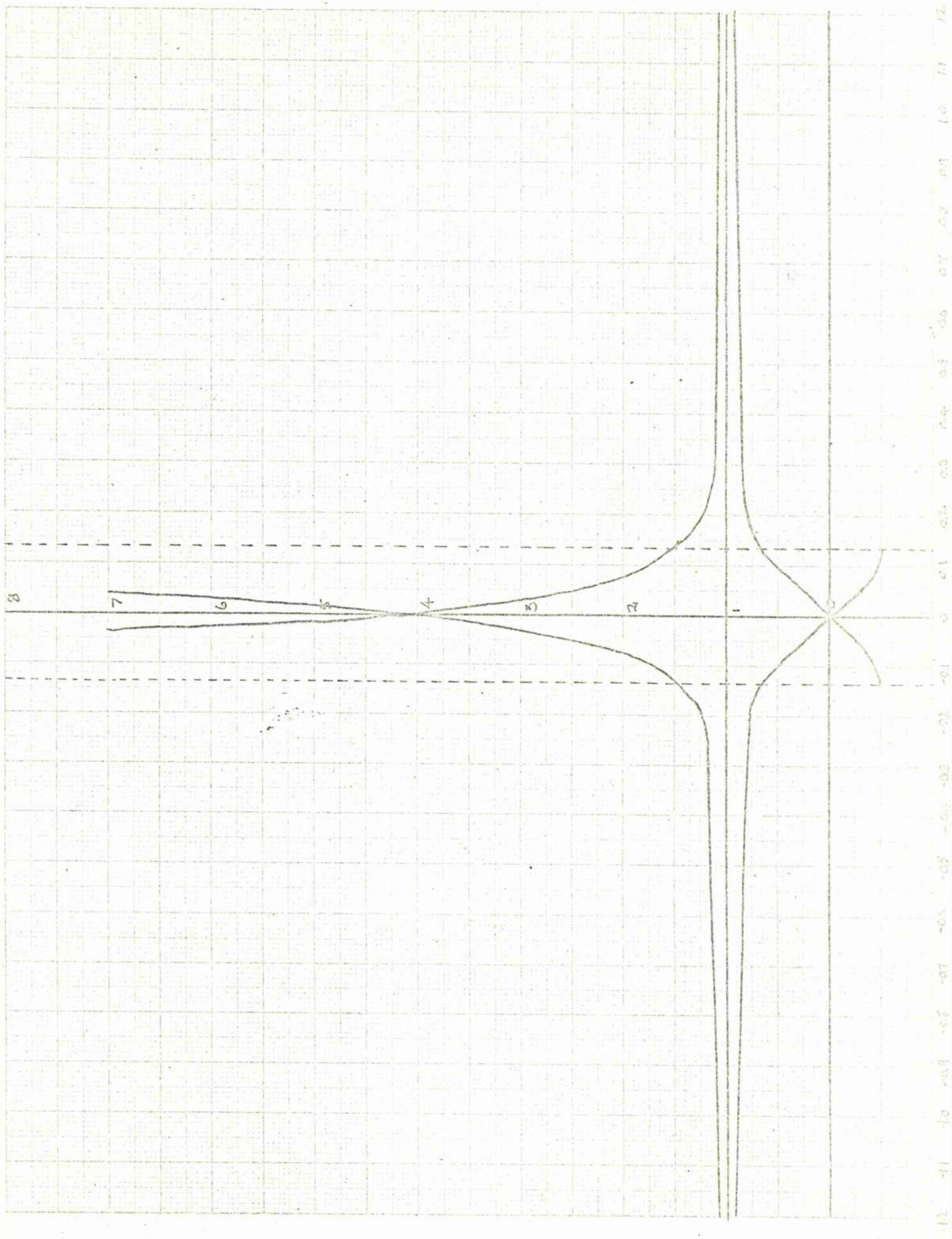
for $c > 50$, we have after squaring both sides

$$(50 + c)(50 - c) > (c - 50)^2$$

$$\frac{4c}{3}(c - 100) < 0 \quad \text{i.e.} \quad c < 100 \qquad (32)$$

It turned out that no solution for $c > 100$ was found. From the nature of the solution no values of y above 1 or below q were found


C
y
P
Q
P
20
18
16
14
12
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-14



except for the exceptional cases $\underline{c} = -50, 100$. The shaded area of graph 3 shows the region investigated most fully. It was found, however, when the coefficient of y' was not zero but relatively large that the equation

$$y'' + 101y' + 160y = -100y^2 \qquad (33)$$

yielded a numerical solution. As might be expected from the parameters on the left hand side of the equation, the solution was exponential in form and the large decay factor probably changed the nature of the solution sufficiently to make it real.

For $\underline{c} = 150$, a solution which turned out to be complex, was found. We have

$$\left(\frac{dy}{dt}\right)^2 = (y-1)\left[\frac{2.150}{3}y^2 + \left(\frac{2.150}{3} - 100\right)y + \left(\frac{2.150}{3} - 100\right)\right]$$

$$= 100y^2\,(y-1)$$

The solution to this is

$$t = \pm\ 2\tan^{-1}(y-1)^{\frac{1}{2}} + \text{constant} \qquad (34)$$

Since the condition $y(0) = 0$ cannot be satisfied in the real plane, it would look as though this is not a real solution for the boundary conditions being used.

The question of solution for the unstable arms in the phase diagram was briefly discussed earlier on and on what evidence there was, there seemed to be not complete solution for these curves.

CHAPTER 2 - NUMERICAL METHODS

2.1. Some Preliminary Methods Investigated

In the initial analysis of the problem, several attempts at a series solution were tried. Attempts at substituting various forms of infinite series with unknown coefficients were investigated but, as shown below, these ran into difficulties due mainly to the boundary conditions imposed upon the equation

(1) The simplest form of solution tried was

$$y = \sum_{r=0}^{\infty} a_r t^r \tag{35}$$

Differentiating gives

$$y' = \sum_{r=0}^{\infty} a_r r t^{r-1}$$

The first condition $y(0) = 0$ is easily satisfied by equation (35), but the second condition namely $y(t_m) = 1$ where t_m is the position of the first maximum is very awkward. For the maximum we have that $y'(t_m) = 0$.

$$y(t_m) = \sum_{r=0}^{\infty} a_r t_m^r = 1$$

$$y'(t_m) = \sum_{r=0}^{\infty} a_r \, r \, t_m^{r-1} = 0$$

Squaring y gives

$$y^2 = \sum_{r=0}^{\infty} a_r t^r \sum_{s=0}^{\infty} a_s t^s = \sum_{r=0}^{\infty} \sum_{s=0}^{r} a_s a_{r-s} t^r$$

Differentiating y gives

$$y' = \sum_{r=0}^{\infty} a_r r t^{r-1}$$

$$y'' = \sum_{r=0}^{\infty} a_r r(r-1)t^{r-2}$$

Substituting these into the equation gives

$$\sum_{r=0}^{\infty} a_r . r(r-1)t^{r-2} + a\sum_{r=0}^{\infty} a_r . r . t^{r-1} + b\sum_{r=0}^{\infty} a_r t^r = \sum_{r=0}^{\infty}\sum_{s=0}^{r} a_s a_{r-s} t^r$$

$$\therefore \sum_{r=0}^{\infty}\left[a_{r+2}(r+2)(r+1) + a a_{r+1}^{(r+1)} + b a_r\right]t^r = \sum_{r=0}^{\infty}\sum_{s=0}^{r} a_s a_{r-s} t^r$$

Equating powers of t gives

$$a_{r+2}(r+2)(r+1) + a.a_{r+1}(r+1) + b a_r = \sum_{s=0}^{r} a_s a_{r-s}$$

$$\therefore a_{r+2} = \frac{\sum a_s a_{r-s} - a.a_{r+1}(r+1) + b a_r}{(r+2)(r+1)} \qquad (36)$$

Since $y(0) = 0$, $a_o = 0$. From the above we therefore have

$$a_2 = \frac{a_o^2 - 2aa_1 + ba_o}{2.1} = -a.a_1$$

It can be seen that since each coefficient is dependent on the two coefficients below it, all the coefficients can eventually be expressed in terms of a_1. Since we have another two equations from the initial conditions involving t_m and the coefficients, we can in theory find a_1. Analytically, this would be very involved, but using numerical techniques, the value of a_1 could be found by iteration.

It was subsequently decided not to elaborate on this method, due to the fact that for large values of t, the error introduced into the corresponding y value would probably be considerable. For the exponentially decaying solution this would be especially noticeable since the function tends asymptotically to the t-axis.

(2) For the cases with the parameter $a > 20$, the linear equation has an exponentially decaying solution (see figure 1). Due to this fact, a solution of the form

$$y = \sum_{r=0}^{\infty} a_r e^{-rt} \tag{37}$$

was attempted. Difficulties with the boundary conditions again arose. We have

$$y(0) = \sum_{r=0}^{\infty} a_r = 0$$

and

$$y(t_m) = \sum_{r=0}^{\infty} a_r e^{-rt_m} = 1, \quad y'(t_m) = -\sum_{r=0}^{\infty} r a_r e^{-rtm} = 0$$

Differentiating equation (37) twice, we get, equating coefficients of e^{-rt}

$$(r^2 - ar + b)a_r = \sum_{s=0}^{r} a_s a_{r-s} \tag{38}$$

For $r = 0$.

$$ba_0 = a_0^2 \qquad\qquad a_0 = b \text{ or } 0$$

Since y tends to zero as t tends to infinity, the coefficient a_0 must be zero.

For $r = 1$

$$(1 - a + b)a_1 = 2a_0 a_1$$

$$\therefore\ a_0 = (1 - a + b)/2 \quad \text{or} \quad a_1 = 0$$

In the first case a_0 is non-zero and since a and b are arbitrary, this is incompatible with the fact that $a_0 = 0$. If a_1 is zero, the rest of the coefficients in turn become zero and the solution y = 0 is incompatible with the non-homogeneous boundary conditions.

A generalisation of the above might be tried i.e.

$$y = \sum_{r=0}^{\infty} a_r e^{-\alpha_r t} \tag{39}$$

This would in theory work, but because of the number of unknowns to be evaluated and the difficulty of fitting the boundary conditions namely,

$$\sum_{r=0}^{\infty} a_r = 0 \quad , \quad \sum_{r=0}^{\infty} a_r \alpha_r e^{-\alpha_r t_m} = 0 \quad , \quad \sum_{r=0}^{\infty} a_r e^{-\alpha_r t_m} = 1 \quad . \tag{40}$$

this method was abondoned.

The solution, as it was actually found, took the form

$$y = \sum_{r=0}^{\infty} \sum_{s=0}^{r} a_{rs} \exp\left[-(s\alpha + (r-s)\beta)t\right] \tag{41}$$

where α and β were the roots of the equivalent linear equation. The methods used, however, did not tackle the problem by straight forward substitution as abovebut were derived from Picard's Method and Perturbation. In the above, apart from the fact that y^2 would be a quadruple summation, the boundary conditions again present difficulties.

2.2. The Madelung Transformation

Using the Madelung transformation, an attempt was made to isolate the exponentially decaying part of the sinusoidally varying solution. For this the function

$$y = F(t) \exp\left[i \int g(t)dt\right] \tag{42}$$

is substituted into the equation, the real and imaginary parts being equated. Thus

$$y' = (F' + ig) \exp\left[i \int g(t)dt\right]$$

$$y'' = (F'' - g^2 + i(g' + F'g)) \exp\left[i \int g(t)dt\right]$$

Substituting these into the equation gives

$$(F'' - g) + i(g' + F'g) + a(F' + ig) + bF = cF^2 \exp\left[i \int g(t)dt\right]$$

i.e. $$(F'' - g + aF' + bF) + i(F'g + g' + ag) = cF^2 \exp\left[i \int g(t)dt\right] \tag{43}$$

In the linear equations, the exponential can be eliminated throughout and two simultaneous equations explicit in F and g obtained. In this case, however, on the right hand side, the exponential cannot be eliminated although it could be expanded in the form

$$\exp\left[i \int g(t)dt\right] = \tfrac{1}{2}\left[\sin\left\{\int g(t)dt\right\} - i\cos\left\{\int g(t)dt\right\}\right]$$

Equating real and imaginary part of this would yield equations which are probably harder to solve than the original equation and therefore this idea was abondoned.

2.3. Methods Used for Solution

Of the four methods tried, two were numerical and two were semi-analytic. The numerical treatment involved a finite difference method and a step-by-step method. The first was used in order to fix the solution at the boundary points, and the step-by-step method, being quicker was then applied, using as its initial conditions two points found from the finite difference method. In figure (4) points A and B represent two points produced from the finite difference method to be used by the step-by-step method.

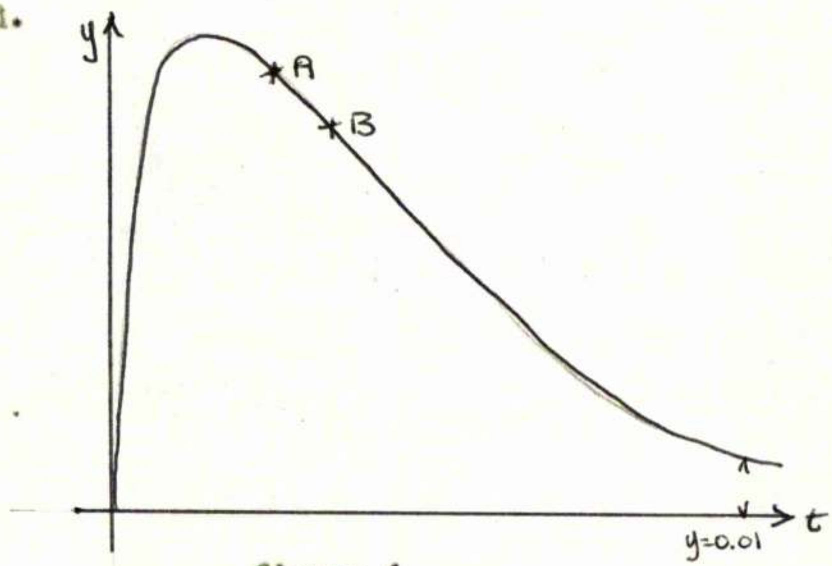


figure 4.

It was decided that a suitable place to stop the integration would be when y was 1% of its maximum value i.e. 0.01. If the finite difference method had been used to integrate the whole curve, a great deal of machine store would have been used. This is due to the fact that great accuracy in the region from zero to B was required because of the considerable gradient at this part of the curve. This accuracy necessitated many points in the section. On the other hand, in the region from A onwards, the gradient was in comparison increasingly gentle, and for the same accuracy, the number of points required per unit length was much less. This difficulty could perhap have been overcome by varying the interval between points, the interval becoming larger for increasing y. Alternatively, the numerical method could have been modified to suit the initial conditions of A and B. This was thought of, but it soon became evident that without too much work, the process could be speeded up considerably by changing it to a non-iterative step-by-step process. It not only has the advantage of speed, but it takes considerably less storage. Because the finite difference method represents the differential equation as a set of simultaneous difference equations, the number of points must be known before the programme is run. In this way, it would be difficulty and certainly very inefficient to use this method in evaluating exactly up to $y = 0.01$. On the other hand, it is extremely easy to do this using the step-by-step process, since each point, as it is evaluated, is tested for the condition $y = 0.01$.

The two semi-analytic methods, based on Picard's Method and Perturbation, were methods in which solutions in terms of a series of exponentials for values of $a > 20$ and in terms of exponentials and trigonometric functions for $a < 20$, could be obtained. In both methods, the initial approximation taken was the solution for the linear case. The first non-linear approximatio in both methods was found without much calculation, but as might be expected, each succeeding new approximation demanded an increase in work and gave a diminishing return.

In the two methods, the higher coefficients were calculated from lower coefficients. In Perturbation this calculation was done by hand, but in Picard's method this was automated and theoretically the solution could be given in terms of any number of exponentials. In the former case, the accuracy was only that of the second non-linear approximation.

2.4. Finite Difference Method

In this method, the differential equation is represented as a finite difference equation. For a second order differential equation there will be two less difference equations than points being calculated, the rest of the information about the solution being given by the boundary conditions. For the equation $y'' + ay + by = 0$, the finite difference equation was derived as follows

$$\frac{dy}{dt} = \frac{\Delta y}{\Delta t} = \frac{y_{r+\frac{1}{2}} - y_{r-\frac{1}{2}}}{h} \tag{44}$$

where $y_{r+\frac{1}{2}}$ and $y_{r-\frac{1}{2}}$ are two consecutive values of the function separated by an interval $\Delta t = h$.

$$\frac{d^2y}{dt^2} = \frac{\Delta}{\Delta t}\left(y_{r+1/2} - y_{r-1/2}\right) \times \frac{1}{h} = \frac{(y_{r+1} - 2y_r + y_{r-1})}{h^2} \tag{45}$$

$$\therefore \frac{(y_{r+1} - 2y_r + y_{r-1})}{h^2} + a\frac{(y_{r+1} - y_{r-1})}{2h} + by_r = 0$$

$$\therefore \left(1 - \frac{h}{2}a\right)y_{r-1} + (bh^2 - 2)y_r + \left(1 + \frac{h}{2}a\right)y_{r+1} = 0 \tag{46}$$

$$\therefore a_1 y_{r-1} + a_2 y_r + a_3 y_{r+1} = 0 \tag{47}$$

where a_1, a_2, a_3 are given from the equation above. For the non-linear equation, the finite difference equation therefore becomes

$$a_1 \, y_{r-1} + a_2 \, y_r + a_3 \, y_{r+1} = c \, y_r^2 \, h^2 \qquad (48)$$

As in most methods of solving non-linear equations, the non-linear element must be linearised. This can be done in several ways. For example, if the non-linear term was u.v where u and v alone are linear functions, this could be linearised by writing

$$u.v = \tfrac{1}{2}(u \, \bar{\bar{v}} + \bar{\bar{u}} \, v)$$

where $\bar{\bar{u}}$ and $\bar{\bar{v}}$ are the previous approximations found from the equation. Similarly y^2 can be linearised by making $y^2 = y \times \bar{y}$ where the same convention holds. Although this is probably faster than letting the approximation be $y^2 = \bar{\bar{y}} \times \bar{\bar{y}}$, it was not thought of at the time the programme was written, and the latter approximation was used.

Initially a value of $\bar{\bar{y}}$ must be guessed. This was not too difficult to do as the non-linear solutions were similar to the linear ones. In actual fact, because it was straightforward and iteration onto the final solution was generally good, a straight line was used as a first approximati to the purely exponential case. However, although this approximation was initially used for the sinusoidal cases, due to instabilities in certain of the equations, the solutions of the equivalent linear equations were subsequently used. Now that we know an approximation to y whether it be a straight line or otherwise, we substitute it into the right hand side of the finite difference equation. We thus obtain from equation (48)

$$
\begin{aligned}
a_1 y_1 + a_2 y_2 + a_3 y_3 &= c\bar{y}_2 h^2 \\
a_1 y_2 + a_2 y_3 + a_3 y_4 &= c\bar{y}_3 h^2 \\
a_1 y_3 + a_2 y_4 + a_3 y_5 &= c\bar{y}_4 h^2 \\
a_1 y_4 + a_2 y_5 + a_3 y_6 &= c\bar{y}_5 h^2 \\
a_1 y_5 + a_2 y_6 + a_3 y_7 &= c\bar{y}_o h^2
\end{aligned}
\qquad (49)
$$

The equations can be represented in matrix form as

$$AY = X$$

where

$$
A = \begin{bmatrix}
a_1 & a_2 & a_3 & & & 0 & \\
 & a_1 & a_2 & a_3 & & & \\
 & & a_1 & a_2 & a_3 & & \\
 & 0 & & a_1 & a_2 & a_3 & \\
 & & & & a_1 & a_2 & a_3
\end{bmatrix}
$$

From the above, and the boundary conditions, the new values of y can be worked out. Each new set of values of the y_r's are tested with the previous values $\bar{y}_r$ and when $(y - \bar{y})$ for all points is less than some predetermined value, e.g. 10^{-4} then the results are obtained.

In most boundary problems, the value of the function is known at two points say, in the above example y_1 and y_7. Eliminating these from the simultaneous equations results in five equations in five unknowns. There is therefore a unique value for each point. Since the boundary conditions were not so simple, another method of tackling the problem had to be found. In essence, the second point y_2 was varied until the first peak hit the value 1. If we let the point y_2 be x, then since we know that $y_1 = 0$, we have

$$\begin{aligned}
a_3y_3 &= c\bar{y}_2^{-2} h^2 - a_2x \\
a_2y_3 + a_3y_4 &= c\bar{y}_3^{-2} h^2 - a_1x \\
a_1y_3 + a_2y_4 + a_3y_5 &= c\bar{y}_4^{-2} h^2 \\
a_1y_4 + a_2y_5 + a_3y_6 &= c\bar{y}_5^{-2} h^2 \\
a_1y_5 + a_2y_6 + a_3y_7 &= c\bar{y}_6^{-2} h^2
\end{aligned}$$

Thus

$$\begin{bmatrix} a_3 & & & & \\ a_2 & a_3 & & 0 & \\ a_1 & a_2 & a_3 & & \\ & a_1 & a_2 & a_3 & \\ 0 & & a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} c\bar{y}_2^{-2} h^2 - a_2x \\ c\bar{y}_3^{-2} h^2 - a_1x \\ c\bar{y}_4^{-2} h^2 \\ c\bar{y}_5^{-2} h^2 \\ c\bar{y}_6^{-2} h^2 \end{bmatrix} \qquad (50)$$

Inverting the matrix A will give the new values of y in terms of the old values which are known, and x which has to be found. We thus obtain

$$
\begin{aligned}
y_3 &= A_3 - B_3\,x \\
y_4 &= A_4 - B_4\,x \\
y_5 &= A_5 - B_5\,x \\
y_6 &= A_6 - B_6\,x \\
y_7 &= A_7 - B_7\,x
\end{aligned}
\tag{51}
$$

where the A's are the elements of the inverse of A times the $c\bar{y}^2h^2$'s. The particular y which fitted the maximum could be found iteratively, but for reasons of speed a different method was used. Each y_r in turn was given the value one until one such y was found say y_p such that when $y_p = 1$, all other y's were less than one. The rest of the y's were of course calculated from the value of x derived from y_p. We have

$$y_r = A_r - B_r x = A_r - \frac{B_r}{B_p}\,(1 - A_p) \tag{52}$$

for all y_r's.

To find the values of the A_r's and B_r's, it was not, in practice, necessary to invert the matrix A. As can be seen from equations (50) and (51) we can derive the following :

$$y_3 = (c\bar{y}_2^2h^2 - a_2x)/a_3 = A_3 - B_3x$$

$$
\begin{aligned}
y_4 &= (c\bar{y}_3h^2 - a_1x)/a_3 - a_2y_3/a_3 \\
&= (c\bar{y}_3^2h^2 - a_1x)/a_3 - a_2(c\bar{y}_2^2h^2 - a_2x)/a_3^2 \\
&= c\left(\bar{y}_3^2 - \frac{a_2}{a_3}\bar{y}_2^2\right)\frac{h^2}{a_3} - \left(\frac{a_1}{a_3} - \frac{a_2^2}{a_3^2}\right)x \\
&= A_4 - B_4x
\end{aligned}
$$

For the rth equation

$$a_1 y_{r-1} + a_2 y_r + a_3 y_{r+1} = c\bar{y}_r^2 h$$

$$\begin{aligned} \therefore y_{r+1} &= (c\bar{y}_r^2 h^2 - a_1 y_{r-1} - a_2 y_r)/a_3 \\ &= [(c\bar{y}_r^2 h^2 - a_1(A_{r-1} - B_{r-1}x) - a_2(A_r - B_r x)]/a_3 \\ &= (c\bar{y}_r^2 h^2 - a_1 A_{r-1} - a_2 A_r)/a_3 + x(a_1 B_{r-1} + a_2 B_r)/a_3 \\ &= A_{r+1} - B_{r+1} x. \end{aligned} \tag{53}$$

A_5 and B_5 can be found now in terms of A_4, B_4 and A_3, B_3. Similarly, each new constant can be found in terms of the two preceding ones.

Once the solution has been found to the required degree of accuracy in terms of previous approximations, a still more accurate result can be achieved by means of a difference correction. Using operators we have :

$$\begin{aligned} h^2 y_r'' &= (hD)^2 y_r = (\delta^2 - \tfrac{1}{12}\delta^4 - \tfrac{1}{90}\delta^6 - \dots) y_r \\ h y_r' &= (hD) y_r = (\mu\delta - \tfrac{1}{6}\mu\delta^3 + \tfrac{1}{30}\mu\delta^5 - \dots) y_r \end{aligned} \tag{54}$$

where $\delta y_r = (y_{r+1/2} - y_{r-1/2})/h$ and $\mu y_r = \frac{1}{2}(y_{r-1/2} + y_{r+1/2})$

The first term in each expansion gives respectively

$$\begin{aligned} \delta^2 y_r &= (y_{r-1} - 2y_r + y_{r+1})/2h \\ \mu\delta y_r &= \mu(y_{r+1/2} - y_{r-1/2})/h = (y_{r+1} - y_{r-1})/2h \end{aligned} \tag{55}$$

and these are the terms which have previously been used to find the appropriate solution. A better solution is therefore given by the equation

$$a_1 y_{r-1} + a_2 y_r + a_3 y_{r+1} = c\bar{y}_r^2 h^2 + \tfrac{1}{6}(\mu\delta^3 + \tfrac{1}{2}\delta^4 - \tfrac{1}{6}\mu\delta^5 - \tfrac{1}{15}\delta^6 + \dots)\bar{y}_r \tag{56}$$

The tabulated values of y are therefore differenced six times, the difference correction calculated from the above, and then added to the previous approximation of the right hand side.

In the program, it was found that for certain values of the interval, a_1 or a_2 might become zero, in which case the program would fail. In this case, the interval was altered and the calculation restarted.

The flow diagram, figure (5), shows how the programme for the finite difference method works. The figure n represents the number of different equations to be solved by varying the parameters a and c. The diagram follows more or less the theory which has just been explained. Initially q = 0, but when a sufficient degree of accuracy has been attained, the difference correction is calculated, and q becomes 1. With the new values of the right hand side of the difference equations, the final y values are calculated and the values of y output. The test for $x > 1$ was inserted because it was found that for values of a close to zero, the solution diverged. More will be said about this in the final chapter.

2.5. Step-by-Step Method

One ce the solution between the boundary conditions had been found the rest of the y values were calculated numerically to y = 0.1 using the step-by-step process. The problem was now essentially one of the initial value type, the two initial values being taken from the solution of the bounda type problem. If the equation is

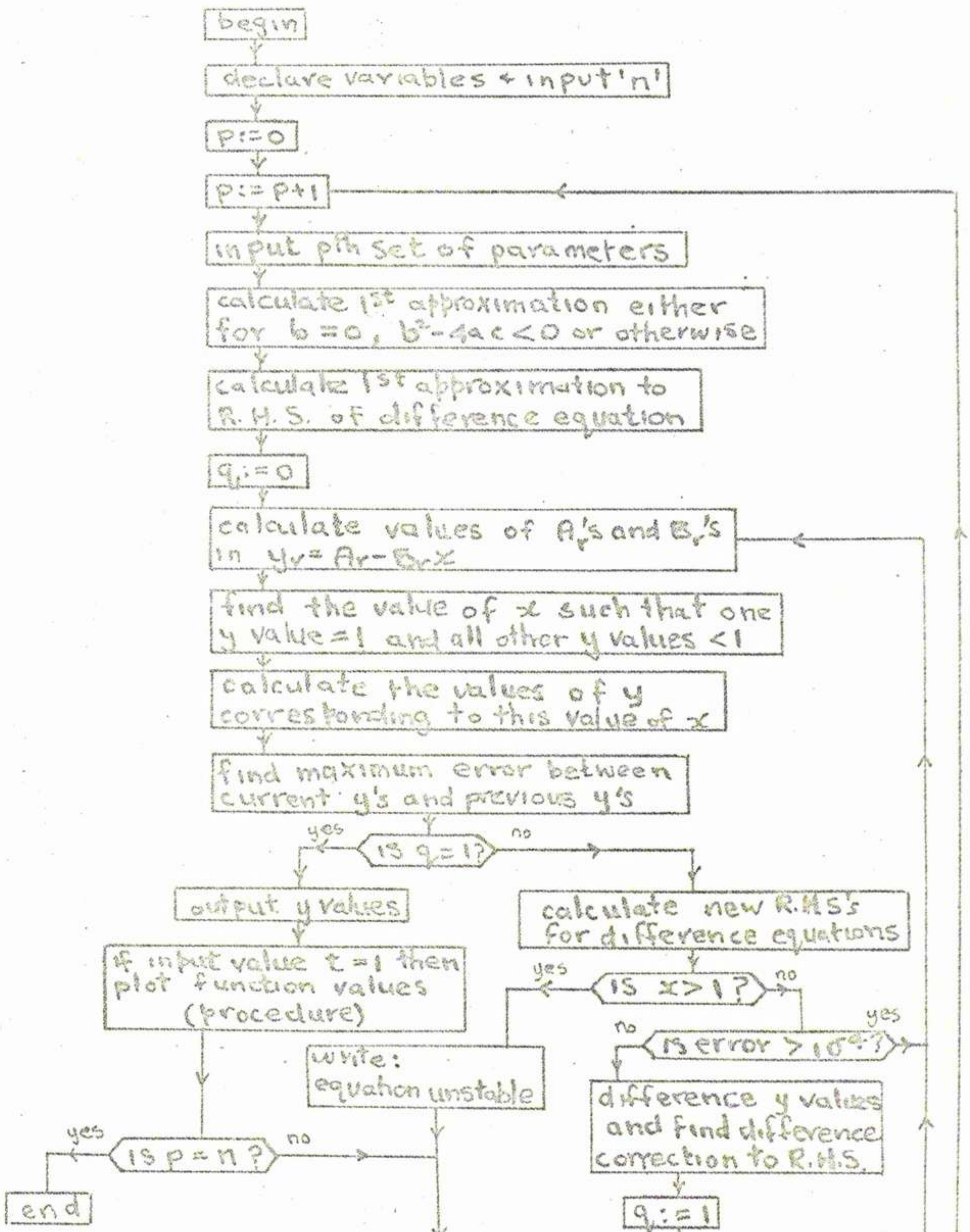
$$y'' + ay' + by = cy^2$$

then as before, the corresponding difference equation is

$$a_1 y_{r-1} + a_2 y_r + a_3 y_{r+1} = c y_r^2 h^2$$

Flow Diagram for Finite Difference Method

Figure 5



Knowing y_1 and y_2, we can calculate y_3 and by resubstitution of each successive point and the one previous into the equation, the next point in turn can be found. For the exponential-type solution, each new value was tested until y = 0.01 was found. For the sinusoidal solution, each maximum was found and the function value at these points was tested for 0.01 Once found, the values of y were output. The flow diagram for the calculation is shown in figure (6).

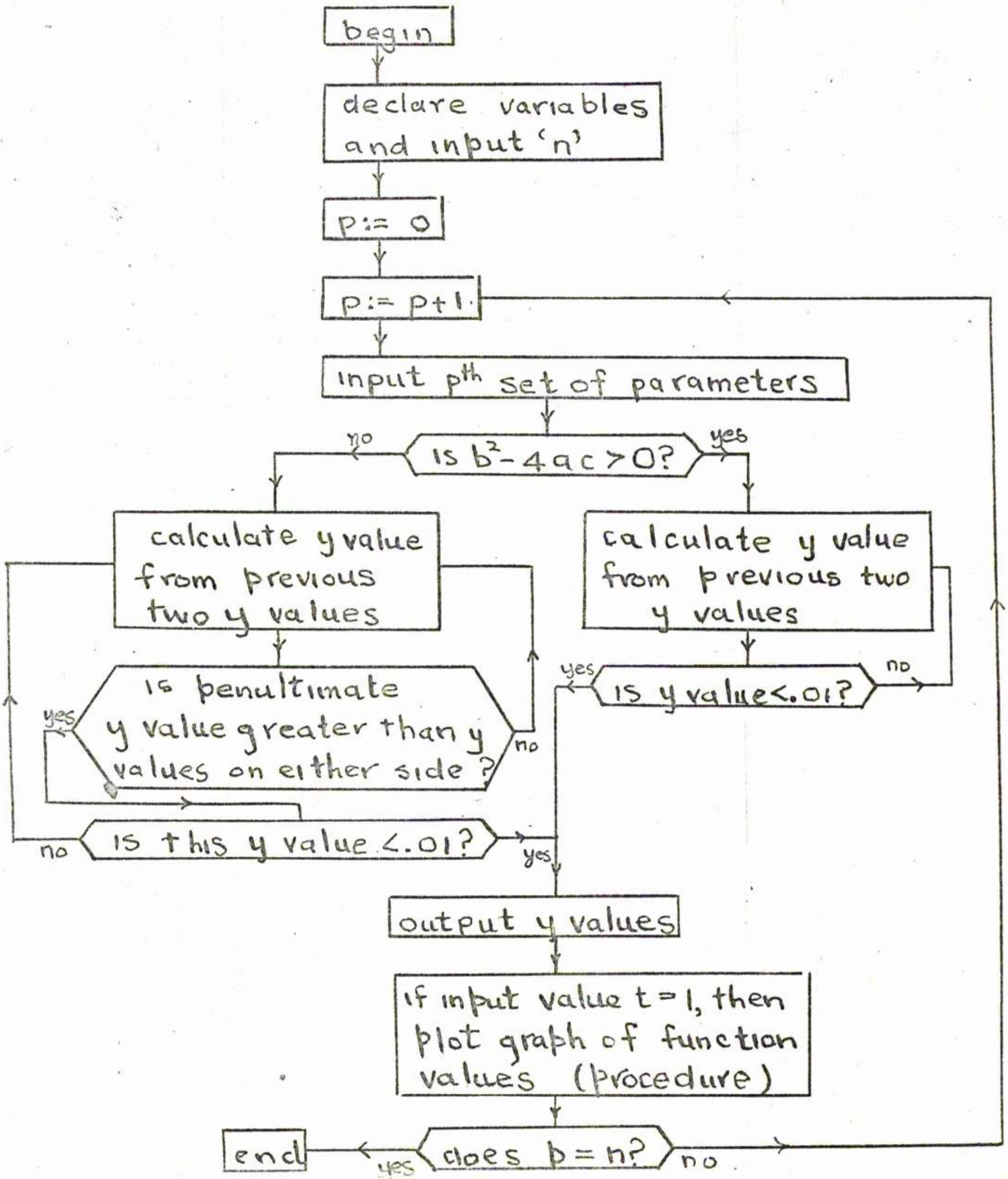
As in the previous flow diagram, n represents the number of sets of parameters input. In the actual programme, the amount of storage used was much more than was absolutely essential. The reason for this was that both in this programme and the finite difference programme, an optional graph plotter in the form of a procedure was used. This meant that y and t values had to be stored in two vectors and could not be output immediately after the y values had been calculated. For the most economical use of storage, only the two previous values of y need be kept. The graph plotter is described below.

2.6. The Graph Plotter

This graph plotter gave a scaled digitalised form of graph suitable for output on a line printer of flexowriter. It plotted out points, but because these could only be printed at integer places, it was not particularly accurate. The points were scaled so that the difference between the maximum and minimum point across the page was 100 spaces. The length of

Flow Diagram for Step-by-Step Method

Figure 6



the graph down the page was left to be scaled by the user. The method was as follows.

If the coordinates are represented by an X vector across the page and Y vector down the page, then the maximum and minimum values in the X array are found and the X values are all scaled by 100 over the difference of these two. This,in effect, brought all the X values within the breadth of the page. A similar technique scaled the Y array down the page by inputing the number of carriage returns desired. In order to make all the X values positive, the lowest number in the X array was subtracted from all other X values. This prevented meaningless negative spaces from being calculated.

Since there can only be an integer number of spaces across the page and carriage returns down the page, the values of the adjusted points had to be rounded off to the nearest whole number. The elements of the Y array then had to be arranged in decreasing order of magnitude so that as the page moved up, the highest values in Y were at the top of the page (see figure (7)). The corresponding X values were at the same time arranged in the same order as the new Y array.

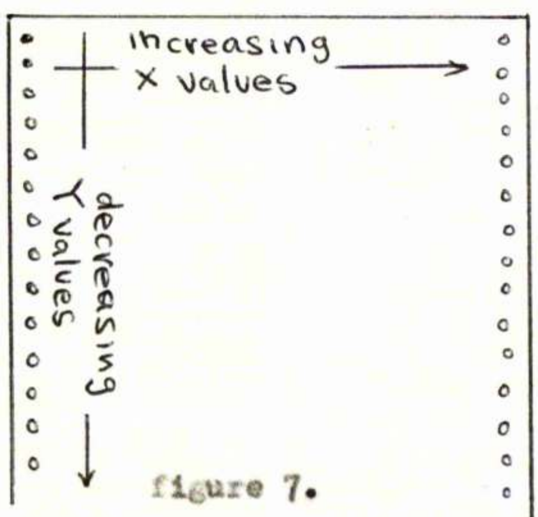


figure 7.

If it happened that two or more 'integerised' Y values were the same, then the corresponding X values had to be rearranged in increasing order of magnitude. If it also happened that two of these newly rearranged X values were the same, then obviously one had to be eliminated. For this case, a counter was set up which, after the graph had been drawn, output the number of points ignored.

In the initial version of the programme, the appropriate coordinates for X and Y were output across and down the page respectively, but since the graph was only rough, and the accurate points were being output before the curve, it was decided to eliminate them. The flow diagram for the programme is shown in figure (8) and an example of the output is shown in graph (21) at the end of the book.

2.7. Perturbation Method

For this method, a general solution in the form of a series of real or complex exponentials with real or complex coefficients, was found. However for $a < 20$, the region in which complex constants arise, it would be a long and arduous task to extract the real sinusoidal solution. It was therefore decided to investigate the two cases, $a < 20$ and $a > 20$, separately.

For the equation $y'' + ay' + by = cy^2$, we assume that we can find a series expansion of the form

$$y = y_o + cy_1 + c^2y_2 + \dots \qquad (57)$$

where y_o, y_1, y_2, etc. are twice differential functions of t to be determined

Flow Diagram for Graph Plotting Procedure.

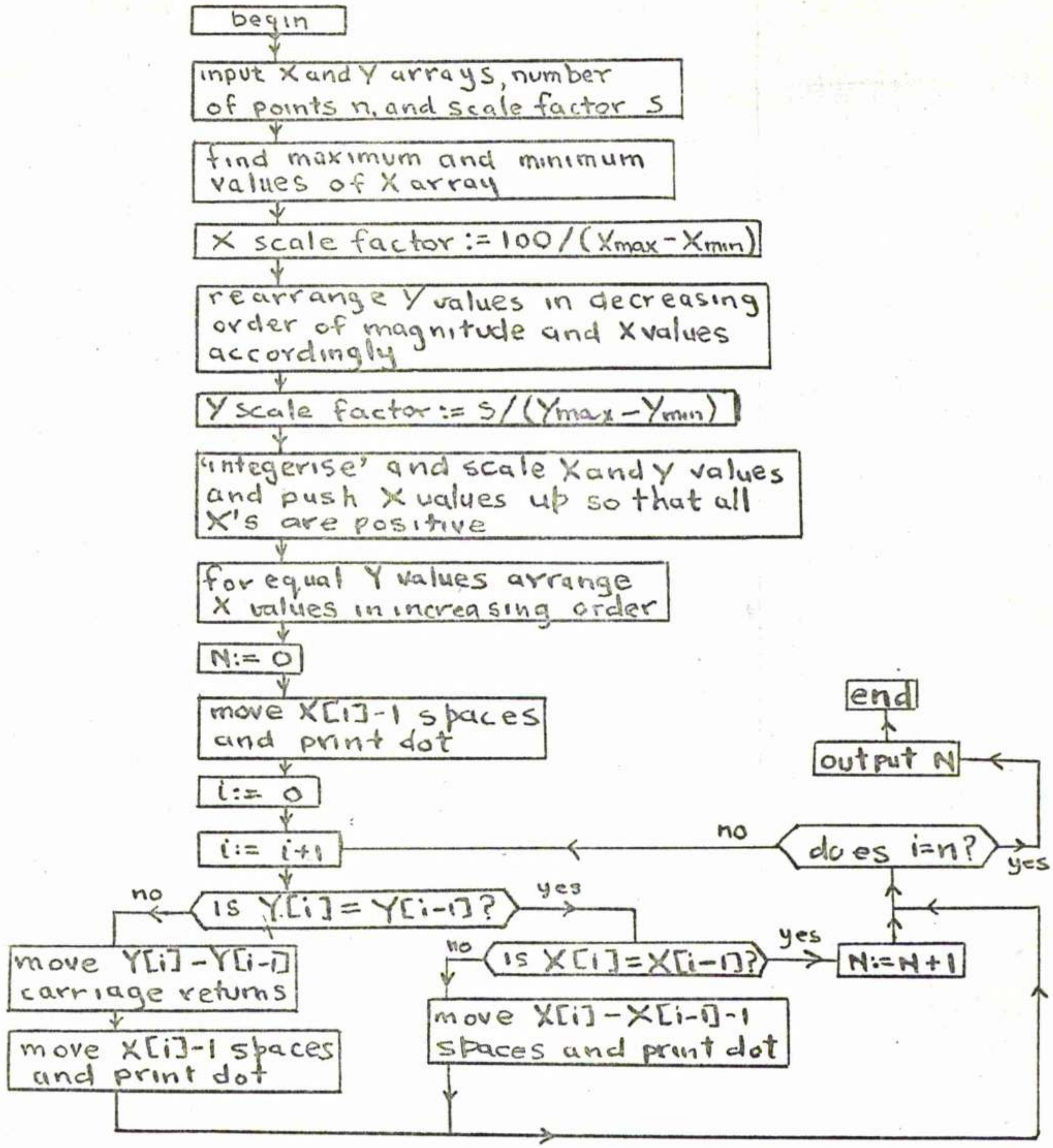


Figure 8

Initially it was thought that the method would only apply for small values of c (i.e. c less than unity), but it was found in comparing the results of this method with the numerical methods, that to the degree of accuracy the solution was taken, reasonably satisfactory results were obtained at least between the boundary conditions, for larger values of c. This suggests that the values of higher order y terms fall off quicker than the coefficients in c increase.

We also let

$$b = b_o + cb_1 + c^2b_2 + \dots \tag{58}$$

where b_o, b_1, b_2 etc. are also to be determined. Since $y(0) = 0$ we let

$$y_o(0) = y_1(0) = y_2(0) = 0 \tag{59}$$

For the second boundary condition we have $y(t_m) = 1$. As it stands, this condition could not easily be incorporated into the calculation. We therefore have to convert the problem into an initial value one by adjusting the gradient by iterative means until $y(t_m)$ hits the value 1. For this condition we take

$$y_o'(0) = y'(0), \quad y_1'(0) = y_2'(0) = \dots = 0 \tag{60}$$

By squaring y, we have

$$y^2 = (y_o + cy_1 + c^2y_2 + \cdots) = y_o^2 + c(2y_oy_1) + c^2(2y_oy_2 + y_1^2) + \cdots$$

Differentiating and substituting into equation (1) we have

$$\begin{aligned}(y_o'' + cy_1'' + c^2y_2'' + \cdots) + a(y_o' + cy_1' + c^2y_2' + \cdots)&\\ +(b_o + cb_1 + c^2b_2 + \cdots)(y_o + cy_1 + c^2y_2 + \cdots)&\\ = cy_o^2 + c^2(2y_oy_1) + c^3(2y_oy_2 + y_1^2) + \cdots&\end{aligned} \tag{61}$$

We then equate powers of c. This gives the first approximation

$$y_0'' + ay_0' + b_0 y_0 = 0$$

This is just the linear equation, the solution of which is known i.e.

$$y_0 = A e^{-\alpha t} + B e^{-\beta t} = A(e^{-\alpha t} - e^{-\beta t}) \qquad \text{(since } y(0) = 0\text{)}$$

where $\alpha, \beta = (a \mp \sqrt{a^2 - 4b_0})/2$ If we let $A_0 = A$ then

$$y_0'(0) = A_0(\beta - \alpha)$$

For the second approximation, equating c we have

$$y_1'' + ay_1' + b_0 y_1 + b_1 y_0 = y_0^2$$

$$\therefore\ y_1'' + ay_1' + b_0 y_1 = y_0^2 - b_1 y_0$$

$$= A_0^2(e^{-2\alpha t} - 2e^{-(\alpha+\beta)t} + e^{-2\beta t}) - b_1 A_0 (e^{-\alpha t} - e^{-\beta t}) \qquad (62)$$

Since $e^{-\alpha t}$ and $e^{-\beta t}$ are both solutions of the complementary function and this equation is linear, it follows that b_1 must be zero. For the particular integral, we therefore let

$$y_1 = B_0 e^{-2\alpha t} + C_0 e^{-(\alpha+\beta)t} + D_0 e^{-2\beta t} \qquad (63)$$

Differentiating this twice, substituting it into the left hand side of (62), and equating coefficients of the exponentials gives

$$\begin{aligned} B_0(4\alpha^2 - 2a\alpha + b_0) &= A_0^2 \\ C_0((\alpha+\beta)^2 - a(\alpha+\beta) + b_0) &= -2A_0^2 \\ D_0(4\beta^2 - 2a\beta + b_0) &= A_0^2 \end{aligned} \qquad (64)$$

Thus B_0, C_0, and D_0 can be found in terms of A_0. The full solution is thus of the form

$$y_1 = A_1 e^{-\alpha t} + B_1 e^{-\beta t} + B_0 e^{-2\alpha t} + C_0 e^{-(\alpha+\beta)t} + D_0 e^{-2\beta t})$$

Since $y_1(0) = 0$ we have

$$A_1 + B_1 + B_0 + C_0 + D_0 = 0$$

since $y_1'(0) = 0$ we have

$$-(\alpha A_1 + \beta B_1 + 2\alpha B_0 + (\alpha+\beta) C_0 + 2\beta D_0) = 0$$

The only coefficients not known in terms of A_0 are A_1 and B_1. From the above equations we therefore obtain

$$A_1 = -[(B_0 + C_0 + D_0)\beta - 2B_0\alpha - C_0(\alpha+\beta) - 2D_0\beta]/(\beta-\alpha)$$

$$B_1 = [(B_0 + C_0 + D_0)\alpha - 2B_0\alpha - C_0(\alpha+\beta) - 2D_0\beta]/(\beta-\alpha) \qquad (65)$$

For the third, and in this discussion final, approximation we have by equating the coefficients of c^2

$$y_2'' + ay_2' + (b_0\ y_2 + b_1 y_1 + b_2 y_0) = 2y_0 y_1 \qquad (66)$$

$$y_2'' + ay_2' + b_0 y_2 = 2y_0 y_1 - b_2 y_0 \quad \text{since } b_1 = 0$$

Using a similar argument, it can be shown that b_2 will be zero. Hence

$$y_2'' + ay_2' + b_0 y_2 = 2A_0(e^{-\alpha t} - e^{-\beta t})(A_1 e^{-\alpha t} + B_1 e^{-\alpha t} + B_0 e^{-2\alpha t} + C_0 e^{-(\alpha+\beta)t} + D_0 e^{-2\beta t})$$

Using the same method as in the second approximation, we find that the values of the coefficients of the exponentials

$$e^{-2\alpha t},\ e^{-(\alpha+\beta)t},\ e^{-2\beta t},\ e^{-3\alpha t},\ e^{-(2\alpha+\beta)t},\ e^{-(\alpha+2\beta)t},\ e^{-3\beta t}$$

are respectively given by the identities

$$
\begin{aligned}
&A_2(\,4\alpha^2 - 2a\alpha + b_0) = 2A_0A_1 \\
&B_2(\,(\alpha+\beta)^2 - a(\alpha+\beta) + b_0) = 2A_0(B_1-A_1) \\
&C_2(\,4\beta^2 - 2a\beta + b_0) = -2A_0B_1 \\
&D_2(\,9\alpha^2 - 3a\alpha + b_0) = 2A_0B_0 \\
&E_2(\,(2\alpha+\beta)^2 - a(2\alpha+\beta) + b_0) = 2A_0(C_0-B_0) \\
&F_2(\,(\alpha+2\beta)^2 - a(\alpha+2\beta) + b_0) = 2A_0(D_0-C_0) \\
&G_2(\,9\beta^2 - 3a\beta + b_0) = -2A_0D_0
\end{aligned}
\tag{67}
$$

Thus the total solution will be

$$
\begin{aligned}
y_2 = {} & A_3 e^{-\alpha t} + B_3 e^{-\beta t} + A_2 e^{-2\alpha t} + B_2 e^{-(\alpha+\beta)t} + C_2 e^{-2\beta t} \\
& + D_2 e^{-3\alpha t} + E_2 e^{-(2\alpha+\beta)t} + F_2 e^{-(\alpha+2\beta)t} + G_2 e^{-3\beta t}
\end{aligned}
\tag{68}
$$

Again using the initial conditions $y_2(0) = 0$ and $y_2'(0) = 0$ we can find the values of A_3 and B_3 in terms of the other coefficients. These turn out to be

$$
\begin{aligned}
A_3 = {} & -[\beta(A_2+B_2+C_2+D_2+E_2+F_2+G_2) - 2\alpha A_2 - (\alpha+\beta)B_2 \\
& - 2\beta C_2 - 3\alpha D_2 - (2\alpha+\beta)E_2 - (\alpha+2\beta)F_2 - 3\beta G_2]/(\beta-\alpha) \\
B_3 = {} & [\alpha(A_2+B_2+C_2+D_2+E_2+F_2+G_2) - 2\alpha A_2 - (\alpha+\beta)B_2 \\
& - 2\beta C_2 - 3\alpha D_2 - (2\alpha+\beta)E_2 - (\alpha+2\beta)F_2 - 3\beta G_2]/(\beta-\alpha)
\end{aligned}
\tag{69}
$$

For this degree of approximation we find that the total solution is given by

$$
\begin{aligned}
y &= y_0 + c y_1 + c^2 y_2 \\
&= A_0(e^{-\alpha t} - e^{-\beta t}) + c(A_1 e^{-\alpha t} + B_1 e^{-\beta t} + B_0 e^{-2\alpha t} + C_0 e^{-(\alpha+\beta)t} \\
&\quad + D_0 e^{-2\beta t} + c^2(A_3 e^{-\alpha t} + B_3 e^{-\beta t} + A_2 e^{-2\alpha t} + B_2 e^{-(\alpha+\beta)t} + C_2 e^{-2\beta t} \\
&\quad + D_2 e^{-3\alpha t} + E_2 e^{-(2\alpha+\beta)t} + F_2 e^{-(\alpha+2\beta)t} + G_2 e^{-3\beta t}) \\
&= (A_0 + cA_1 + c^2 A_3) e^{-\alpha t} + (-A_0 + cB_1 + c^2 B_3) e^{-\beta t} + c(B_0 + cA_2) e^{-2\alpha t} \\
&\quad + c(C_0 + cB_2) e^{-(\alpha+\beta)t} + c(D_0 + cC_2) e^{-2\alpha t} + c^2 D_2 e^{-3\alpha t} \\
&\quad + c^2 E_2 e^{-(2\alpha+\beta)t} + c^2 F_2 e^{-(\alpha+2\beta)t} + c^2 G_2 e^{-3\beta t} \qquad (70)
\end{aligned}
$$

Thus the non-linear equation is each time being reduced to a linear equation with the boundary conditions always being satisfied.

The above solution is the general one discussed previously. For exponential-type solutions, it stands as it is, but for sinusoidal solutions it must be broken down into its real and imaginary parts. For calculating the real parts of the appropriate constants on a computer with complex facility, this might be a reasonable proposition, but by hand the problem would be rather involved. It was also found to be a much more time consuming problem to work out a reasonably accurate solution for the sinusoidal $y = Ae^{-\gamma t} \sin\omega t$. Since this was also the case for the other semi-analytic method investigated, and as in neither method was there a significant difference in the theory it was decided to investigate the purely exponential solutions only, and leave the sinusoidal.

Now that we know the solution in terms of A_o which is a function of the gradient, we can work out, iteratively, its value. First of all, an initial guess was obtained for A_o by using its linear value. Using this the maximum value of y was found and if this was greater than unity, the gradient was proportionately reduced.

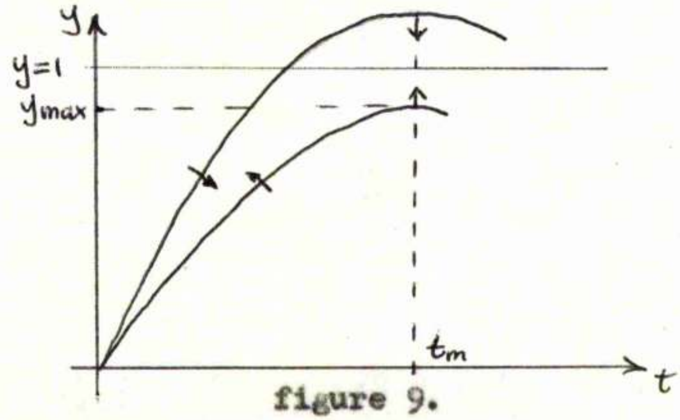


figure 9.

The reverse was done if the maximum was greater. This is shown in figure (9). In effect, the gradient was changed by (1 - diff) where 'diff' was the difference between the maximum and one.

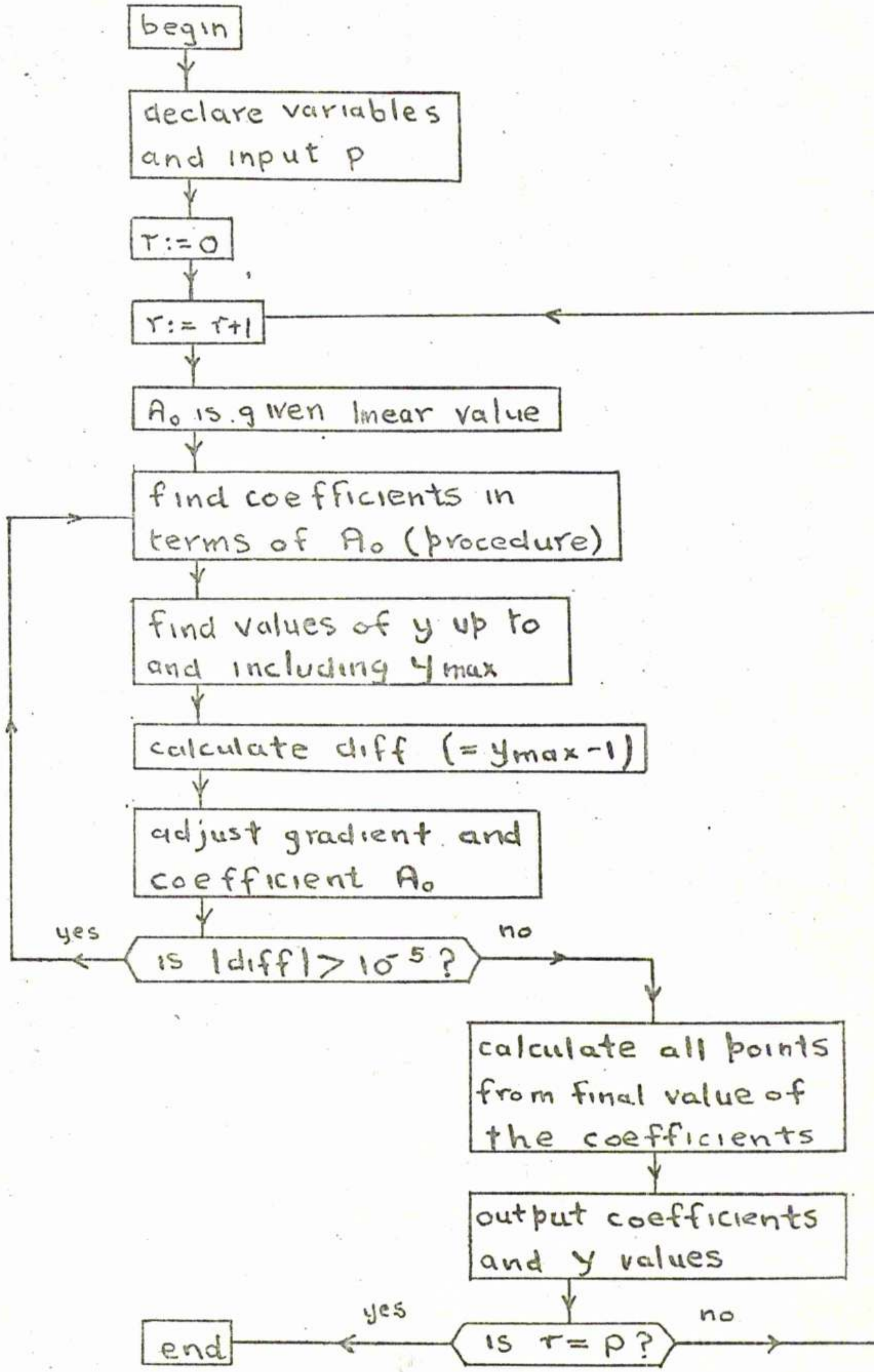
The programme was such that for each iteration, only those points up to and including the maximum were calculated. This means that all points after the maximum were ignored until the required degree of accuracy had been reached when every point was calculated with the final value of the coefficien The flow diagram is shown in figure (10).

2.8. Picard's Method

This method is similar to Perturbation in that it produces an exponential series, but each successive approximation gives a greater number of terms to the expansion than in Perturbation. Essentially, an approximate solution was found from the linear equation, say $f_o(t)$. This solution is substituted into the non-linear part of the equation giving

$$y'' + ay' + by = cf_o^2(t)$$

Figure 10


begin
declare variables and input p
r:=0
r:= r+1
A₀ is given linear value
find coefficients in terms of A₀ (procedure)
find values of y up to and including y max
calculate diff (= ymax-1)
adjust gradient and coefficient A₀
yes
is |diff| > 10⁻⁵ ?
no
calculate all points from final value of the coefficients
output coefficients and y values
yes
is r = p ?
no
end

This equation is a better linear approximation to the non-linear equation being investigated, and we solve it with the boundary conditions to find the new solution $f_1(t)$. This is substituted into the equation as before and the process is repeated until the required degree of approximation is reached.

We have that the solution of the linear equation

$$y'' + ay' + by = 0 \qquad \text{is} \qquad y = A_{10}e^{-\alpha t} + A_{01}e^{-\beta t}$$

Squaring this and substituting it into the right hand side of the equation gives

$$y'' + ay' + by = c\left(A_{10}^2 e^{-2\alpha t} + 2A_{10}A_{01}e^{-(\alpha+\beta)t} + A_{01}^2 e^{-2\beta t}\right) \qquad (71)$$

For the particular integral, we let the solution be

$$y = A_{20}e^{-2\alpha t} + A_{11}e^{-(\alpha+\beta)t} + A_{02}e^{-2\beta t} \qquad (72)$$

where a_{20}, A_{11} and A_{02} are to be determined. As can be seen, the suffices of the coefficients correspond to the appropriate powers in α and β. Differentiating equation (72) twice and substituting the resultant values of y" and y' into the left hand side of equation (71) gives, by equating the coefficients of the exponentials, the values of A_{20}, A_{11} and A_{02} in terms of A_{10} and A_{01}. Once these have been found, we have that the full solution to the first approximate equation is

$$y = A'_{10}e^{-\alpha t} + A_{01}e^{-\beta t} + A_{20}e^{-2\alpha t} + A_{11}e^{-(\alpha+\beta)t} + A_{02}e^{-2\beta t} \qquad (73)$$

$A'_{10}e^{-\alpha t} + A'_{01}e^{-\beta t}$ being the solution of the complementary function.

From the boundary conditions A'_{10} and A'_{01} can be found. From the condition $y(0) = 0$, we have

$$A'_{10} + A'_{01} + A_{20} + A_{11} + A_{02} = 0 \qquad (74)$$

Thus A'_{01} can be found in terms of A'_{10}. The other condition, $y(t_m) = 1$ uniquely determines the value of A'_{10}, but as in the previous method, A_{10} can only be found with relative ease using an iterative technique.

Once this approximate solution has been found, we start off again knowing the coefficients from A'_{10} to A_{20} and square equation (73). The same process is repeated and the coefficients of the exponentials from $e^{-\alpha t}$ to $e^{-2\beta}$ are re-evaluated and the coefficients of $e^{-3\alpha t}$ to $e^{-4\beta t}$ are evaluted.

This can be repeated to give the solution to any number of exponentials. The problem of iteration for the second boundary condition turned out to be more difficult than in Perturbation. Initially, the same method was used, i.e. if the maximum value of y turned out to be below 1 for a particular set of coefficients, the gradient was increased, the coefficients re-evaluated and the maximum value of y again found. This was done, as previously, through modifying the gradient by (1-diff), 'diff' being the difference between y_{max} and 1. It worked quite satisfactorily for positive values of the non-linear coefficient to the order of 75, but for values about -40, the process did not converge. In general it was found that it took longer to find the solution for negative values of c than for their equivalent positive values. This maybe due to the following. For the value of y at t_m, $y' = 0$, and approximately

$$y'' = -(100 - cy)y \qquad (y \simeq 1) \qquad (75)$$

When c is negative, the coefficient of y is much larger than when c has the equivalent positive value. Thus the value of y at the maximum is much more rapidly varying for this case, and accuracy will consequently be more difficult to achieve. This fact can be seen from the shape of any graph with $c = -40$. The peak for this is much sharper than for positive c. For the cases with negative c which did work, it was found that it took more iterations than was thought necessary (sometimes more than 20). It was found, for values of c about -40, that although the solution did not converge, neither did it diverge rapidly. The maximum value of y seemed to fluctuate between 0.73 and 1.2 for a large number of iterations. To reduce the amount of 'feedback' from the difference to the gradient, various functions similar to $\tan^{-1}$ (diff) we tried. This, in effect, would reduce the magnitude of the gradient correcting factor for large values of diff, but leave smaller values relatively unchanged ($\tan^{-1} x \simeq x$ for $x \ll 1$).

Ultimately an empirical formula was found which gave the solution for $c = -40$, but the number of iterations required made the method very impracticable. Hence another method of iteration was looked for. Using Aitken's inverse interpolation, the process was found to converge much faster.

AITKENROOT Procedure

Interpolation was carried out between diff and the coefficient of the first exponential A_{10} and the value of A_{10} was found such that diff was close enough to zero and that the boundary condition $y(0) = 0$ was satisfied. This did not involve calculating the gradient for each iteration as was done in Perturbation. When diff was less than zero, the value of the coefficient was increased by as many increments as was necessary to make diff positive. When the sign change occurred, Aitken's interpolation was used to find the value of A_{10} such that diff $\simeq 0$. If diff was originally greater than zero, A_{10} was decreased. In figure (11) a_1 represents the value of diff for a particular value of A_{10}. If we decrease the value of A_{10} enough, eventually diff will go negati

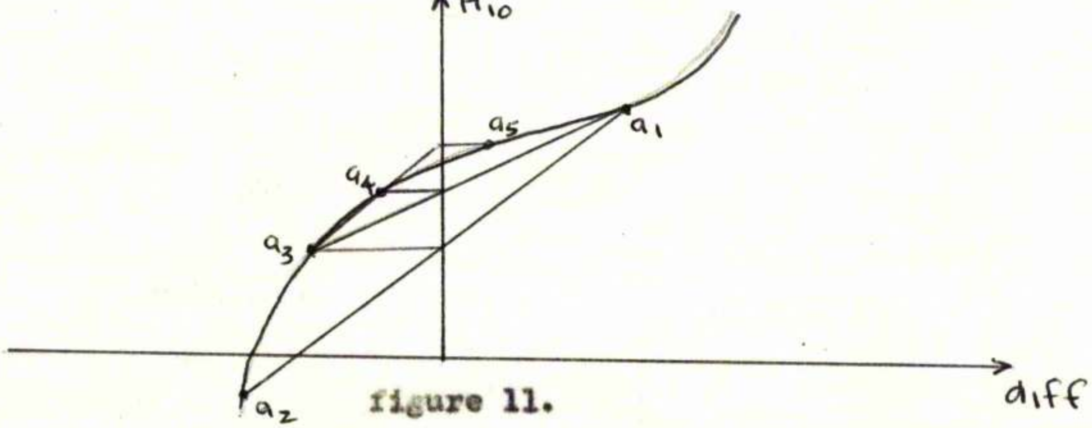


figure 11.

Interpolating linearly between those two points gives a more accurate value to A_{10}. With this new value, A_{01} is recalculated and $A_{01}e^{-\beta t}$, with the rest of the terms of the series, gives the new value of diff corresponding to the new A_{10}. This diff now corresponds to the point a_3. Linearly interpolating between a_1 and a_3 gives a_4, and applying the same to a_3 and a_4 produces, in stable cases, the most accurate approximation so far $-a_5$. Generally, if the interpolation is ill-behaved, any instabilities which occur, will do so when the point a_5 is being calculated. Although in this method, interpolation is always being carri out just between two points, in effect, n^{th} order interpolation is achieved by successive interpolation between the correct points.

Thus finding a_3 is equivalent to first degree interpolation and finding a_5 is equivalent to second degree. In figure (12), the logical sequence of events as it is programmed, is shown

Current Value of A_{10} ($\equiv f$)	Derivation
f_1	initial value
$f_2, f_2 := f_{12}$	f_{12} from $f_1 + f_2$
$f_3, f_3 := f_{13} := f_{23}$	f_{13} from $f_1 + f_3$ f_{23} from $f_2 + f_3$
$f_4, f_4 := f_{14}$ $f_4 := f_{24}, f_4 := f_{34}$	f_{14} from $f_1 + f_4$ f_{24} from $f_2 + f_4$ f_{34} from $f_3 + f_4$

figure 12

From the two points f_1 and f_2 ($\equiv a_1$ and a_2) a third point f_{12} ($\equiv a_3$), is calculated. This is the new f_2 and it replaces the old value of f_2. It is also the first approximation to the calculation of f_3. The second approximat to f_3 is calculated from f_1 and f_3. This new value f_{13} replaces the old value of f_3 to become the next approximation. Interpolation between this and the point f_2 gives f_{23} which in turn becomes the final f_3. Using this as a first approximation to f_4, the same sequence of events is carried out to find the

the most accurate to 4th degree interpolation. This is continued until two successive approximations f_{r-1} and f_r are close enough such that the error between them can be neglected. Thus the final value of A_{10} is found and the solution to the required degree of accuracy is obtained.

Since only current values of f_1, onwards are stored, intermediate points being overwritten by new approximations, the amount of storage for this application of Aitken's interpolation is small compared with more conventional approaches.

Calculation of Coefficients

In Picard's Method, it was decided to calculate the coefficients inside a procedure rather than by hand as is Perturbation. The procedure (CONST) evaluated the coefficients for the new terms and re-calculated the old coefficients from A_{20} up to the start of the new terms.

For the first non-linear approximation, A_{20}, A_{11} and A_{02} were calculated from A_{10} and A_{01}. Since each new approximation is found by squaring the previous solution, the number of terms rises very rapidly. For the first, second and third approximations we have respectively 5, 14 and 44 terms in the series.

From the first approximation, we have, squaring equation (73)

$$y^2 = (A_{10}' e^{-\alpha t} + A_{10}' e^{-\beta t} + A_{20}e^{-2\alpha t} + A_{11}e^{-(\alpha+\beta)t} + A_{02}e^{-2\beta t})^2 \tag{76}$$

When multiplied out this gives exponential terms from $e^{-2\alpha t}$ through all variations up to $e^{-4\beta t}$. For the right hand side of the equation we substitute

$$y = A'_{20}\, e^{-2\alpha t} + \ldots\ldots\; A'_{04} e^{-4\beta t} \qquad (77)$$

Differentiating twice and substituting the value of y", y' and y into the left hand side of the equation will give an exponential series from $e^{-2\alpha t}$ to $e^{-4\beta t}$ with respective coefficients from A'_{20} to A'_{04}. Equating coefficients will give A'_{20} to A'_{04} in terms of A'_{10} to A_{62} and α and β. The main part of the procedure squares the series which was the previous approximation (see equation (76)). If we have that a term of the new series of equation (77) is $A'_{rs} d^{-(r\alpha + s\beta)t}$ then for this we have that

$$y'' + ay' + by = A'_{rs}\,((r\alpha + s\beta)^2 - a(r\alpha + s\beta) + b)\bar{e}^{(r\alpha + s\beta)t}$$

Thus to find the new coefficient A'_{rs}, we must divide the coefficient of $\bar{e}^{(r\alpha + s\beta)t}$ from equation (76) by $((r\alpha + s\beta)^2 - a(r\alpha + s\beta) + b)/c$. The latter part of the procedure does this. The diagram below shows how the process was programmed.

$A_{10}\ A_{01}$ ________________________________

$A_{20}\ A_{11}\ A_{02}$ — 1st non-linear coefficients

____

$A_{30}\ A_{21}\ A_{12}\ A_{63}$ — 2nd non-linear coefficients

$A_{40}\ A_{31}\ A_{22}\ A_{13}\ A_{04}$ ________________________________

$A_{50}\ A_{41}\ A_{32}\ A_{23}\ A_{14}\ A_{05}$

$A_{60}\ A_{51}\ A_{42}\ A_{33}\ A_{24}\ A_{15}\ A_{06}$ — 3rd non-linear coefficients

$A_{70}\ A_{61}\ A_{52}\ A_{43}\ A_{34}\ A_{25}\ A_{16}\ A_{07}$

$A_{80}\ A_{71}\ A_{62}\ A_{53}\ A_{44}\ A\ 35\ A_{26}\ A_{17}\ A_{08}$ ________________________________

figure (13).

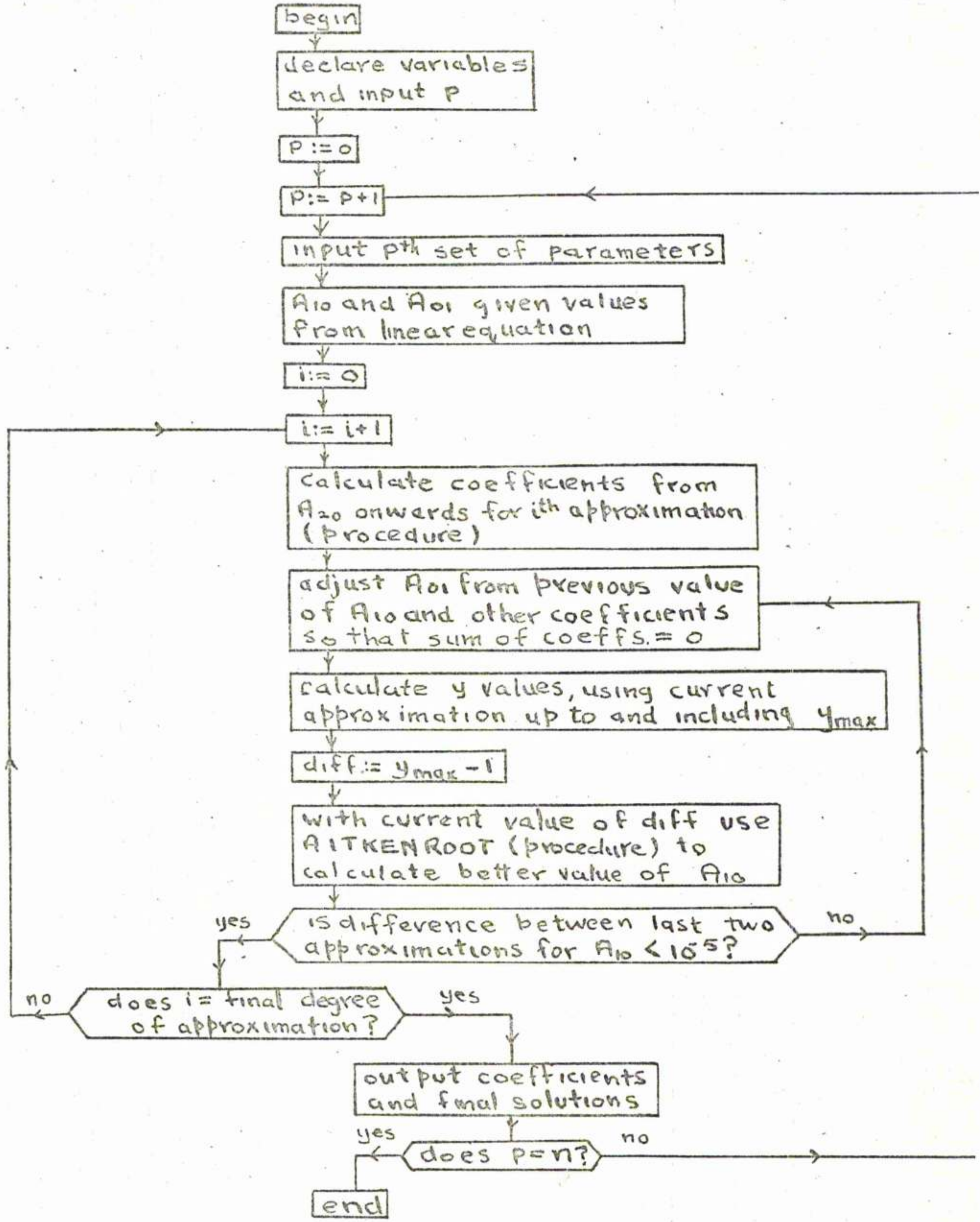
The coefficients in, for example, the third group of the above are calculated solely from the first two groups and the coefficients A_{10} and A_{01}. For example A_{34} is found as below

$$A_{34} = 2A_{30}\ A_{04} + 2A_{21}\ A_{13} + 2A_{31}\ A_{03} + 2A_{12}\ A_{22}$$

Thus the coefficient A_{34} is formed from allcombinations of coefficients whose suffices add up to 3 and 4 from the two previous groups. This process lends itself quite well to being programmed.

Figure (14) shows the flow diagram for the whole programme.

Figure 14


begin
declare variables and input P
p := 0
p := p+1
input pth set of parameters
A10 and A01 given values from linear equation
i := 0
i := i+1
Calculate coefficients from A20 onwards for ith approximation (procedure)
adjust A01 from previous value of A10 and other coefficients so that sum of coeffs. = 0
calculate y values, using current approximation up to and including ymax
diff := ymax - 1
with current value of diff use AITKENROOT (procedure) to calculate better value of A10
is difference between last two approximations for A10 < 10^-5?
yes
no
does i = final degree of approximation?
no
yes
output coefficients and final solutions
does p = n?
yes
no
end

CHAPTER 3 - ANALYSIS OF RESULTS

3.1. Method of Obtaining Numerical Results

Because of the boundary conditions, and the fact that the integration was to be carried along until y was less than 0.01, the programmes were used in a definite order. To find the approximate position of the maximum so that the Finite Difference programme could be used, a modification was made in the Perturbation programme such that it output the approximate value of t_m and nothing else. Once this position had been found, the Finite Difference programme was used to find the solution between the boundary conditions. Next, the Step-by-Step programme, with initial conditions given by the previous programme, found the solution to the point y = 0.01. Having found the upper limit to the integration, the two semi-analytic methods were used.

Due to the fact that the gradient before the maximum was much greater than that after it, it was decided in Picard's method and Perturbation to output a greater density of points for the first part of the curve than the second part. In both programmes a facility was available for outputing every nth point (where n could be varied) after the maximum rather than every point.

In order to compare the results of the numerical methods with these of the semi-analytic, exponential solutions were obtained for variations of the parameters a and c. These are as below.

Values of a : 25, 50, 75, 101

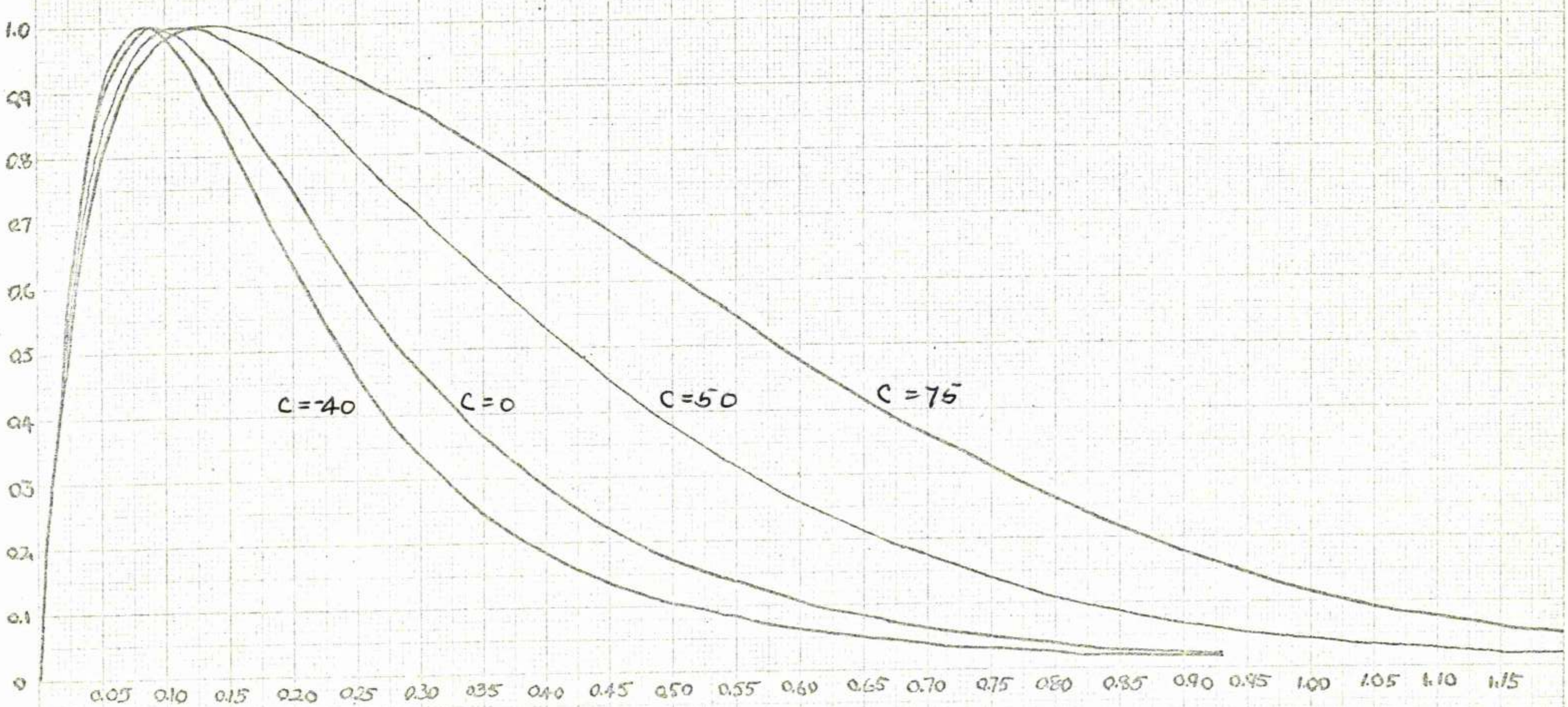
Values of c : -40, 0, 50, 75

The parameter c was given the value -20 as well, but because the solution for -40 was relatively close to the linear case, it was not analysed fully.

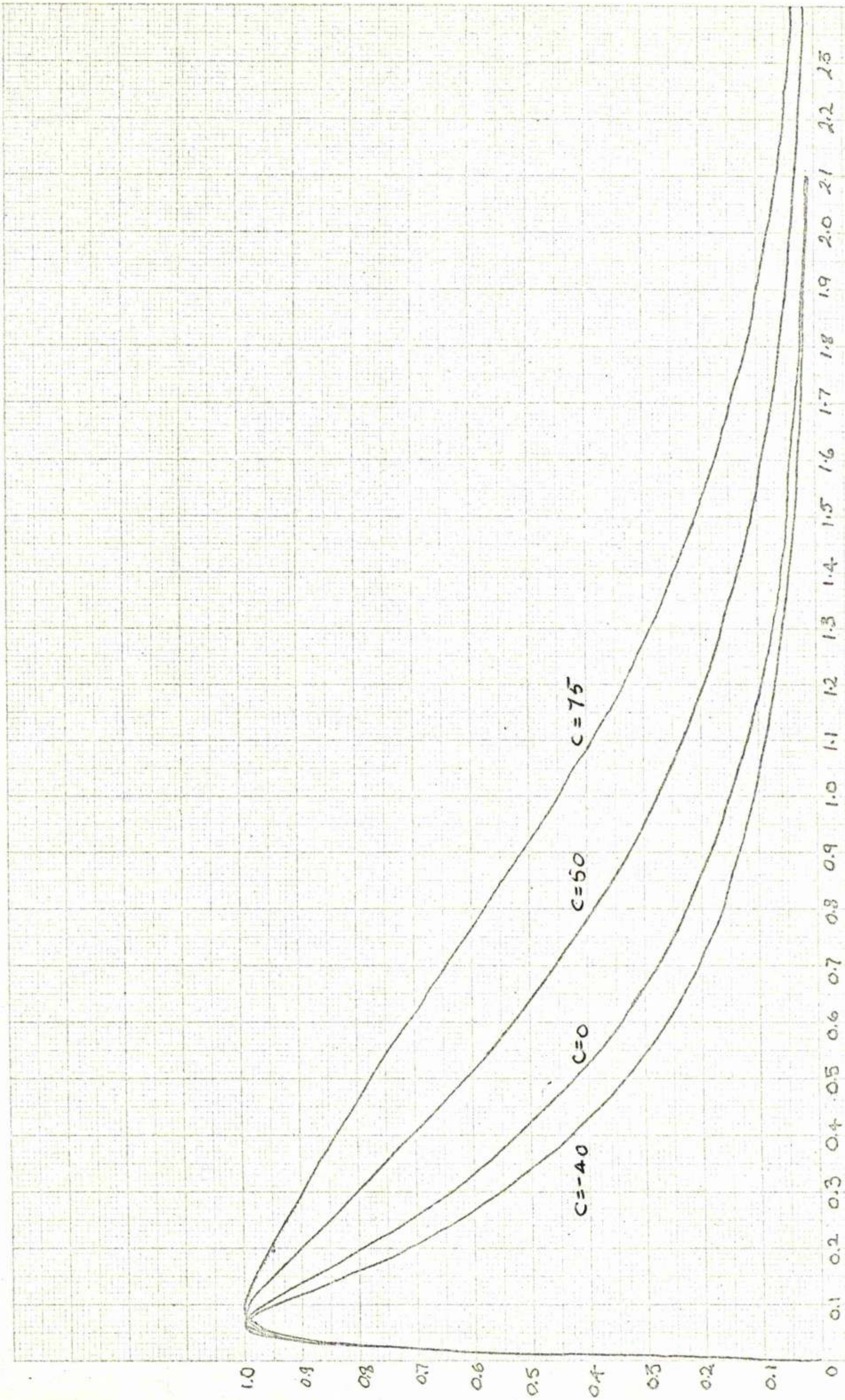
3.2. Comparison of Solutions

The results for a combination of the two numerical methods are shown on graphs 5 to 8. They are arranged so that variations in the values of c appear for constant values of a on the same graph. For negative values of c, the function rises quicker and falls quicker than for the linear case, the opposite being true for values of $c > 0$. As might be expected from a comparison of the linear solutions in which a is varied, the larger the value of a, the steeper the rise and fall of the function. Because of the very fast rise of most of the solutions, any function values taken from the graph would be highly inaccurate.

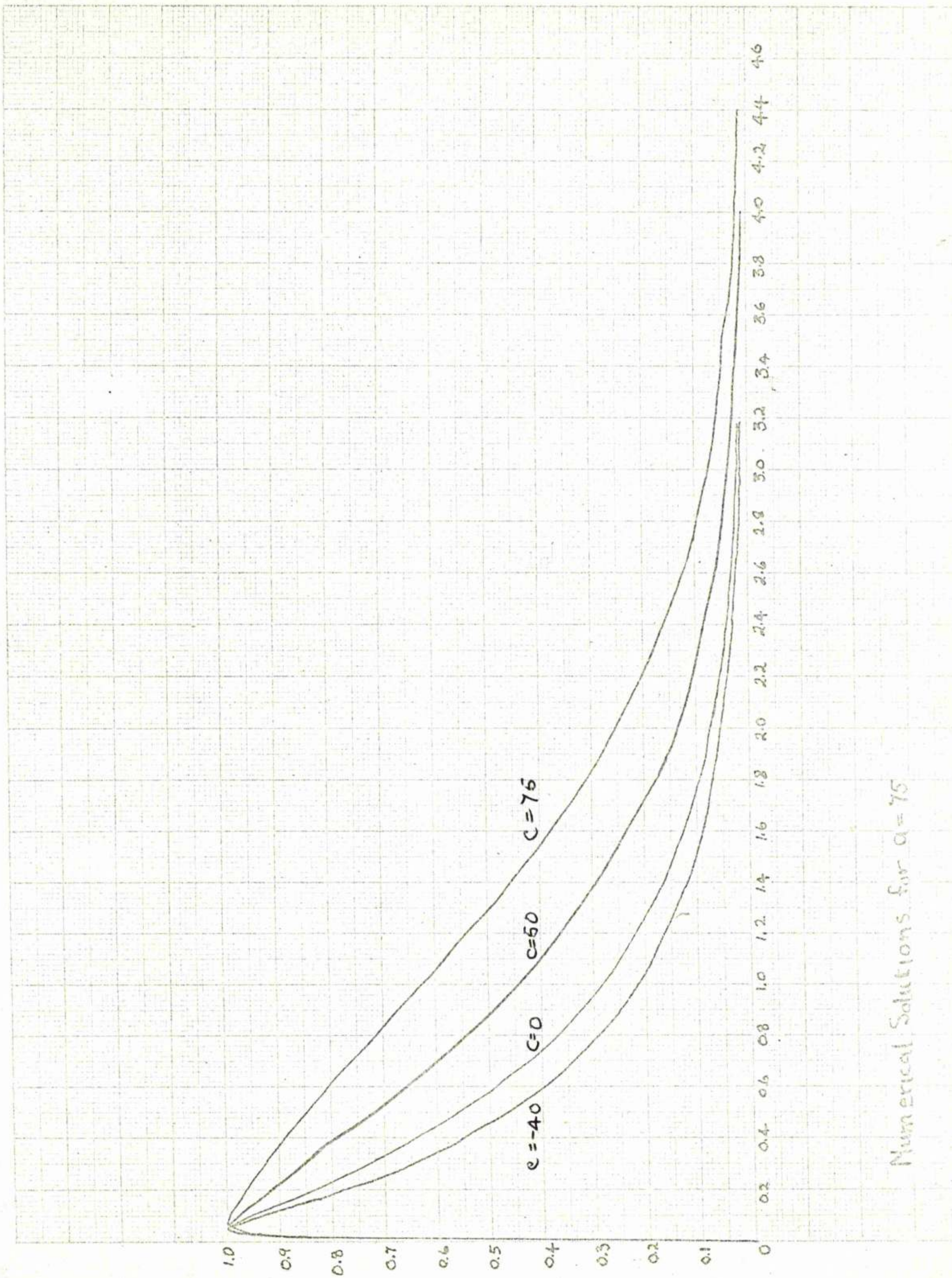
It was found, rather unfortunately, that the Finite Difference method failed for values of a close to zero. This meant that no direct comparison could be made between the numerical solutions and the analytic. Solutions were found for a greater than or approximately equal to 5 for some values of c (as can be seen in the example of the graph plotter procedure, graph (21), a = 5, c = -40), but with this amount of damping, the solution was altered to such an extent that it was felt that no accurate comparison could be made between this and the equivalent undamped case. However, both in the analytic solutions and the numerical, the order in which the maxima came for increasing c values was the same.



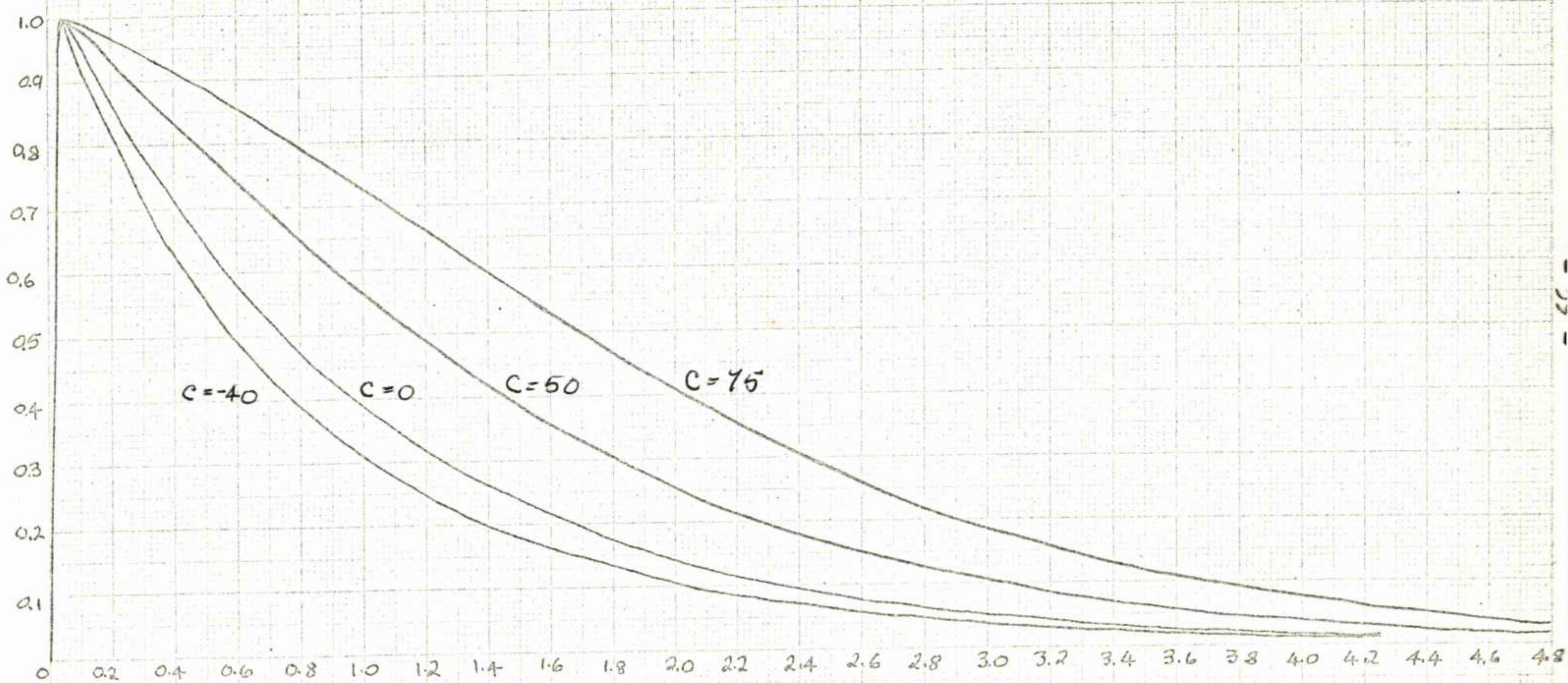
Numerical Solutions for $a = 25$



Numerical Solutions for a = .50



Numerical Solutions for a = 75



Numerical Solutions for a = 101

The reason for the instability was not found, although one possibility might be that of higher frequency terms dominating the solution. In the reasonably damped solutions these terms would probably be coupled to large exponentials which would effectively eliminate them. For this theor a crude interval was tried but with no success. A very small interval was also attempted, but with the same result.

To test the relative accuracy of the two numerical methods, solutio using the Finite Difference method were found for two sets of parameters for the whole length of the curve. These were compared with the appropriate solutions found from the Step-by-Step programme, and the difference between the two was never greater than one point out in the 4th Decimal place. This could be improved either by using a smaller interval or by adding a differenc correction to the Step process.

Because the position of the points after the maxima in the numerica methods, often did not correspond to the position of the points for the other two methods, it was difficult in this region to compare errors. One way of solving the problem would have been to iterate between the values of the semi-analytic methods. However, it was thought faster to write a short programme which evaluated the desired function values by inputing the coefficients of the exponentials. The corresponding function values from the Step-by-Step process were also input and the deviation and percentage deviation between the numerical methods agains both Picard's method and Perturbation were worked out.

The deviation and percentage deviation were as below :

$$\text{deviation} = y_{\text{numerical}} - y_{\text{semi-analytic}}$$

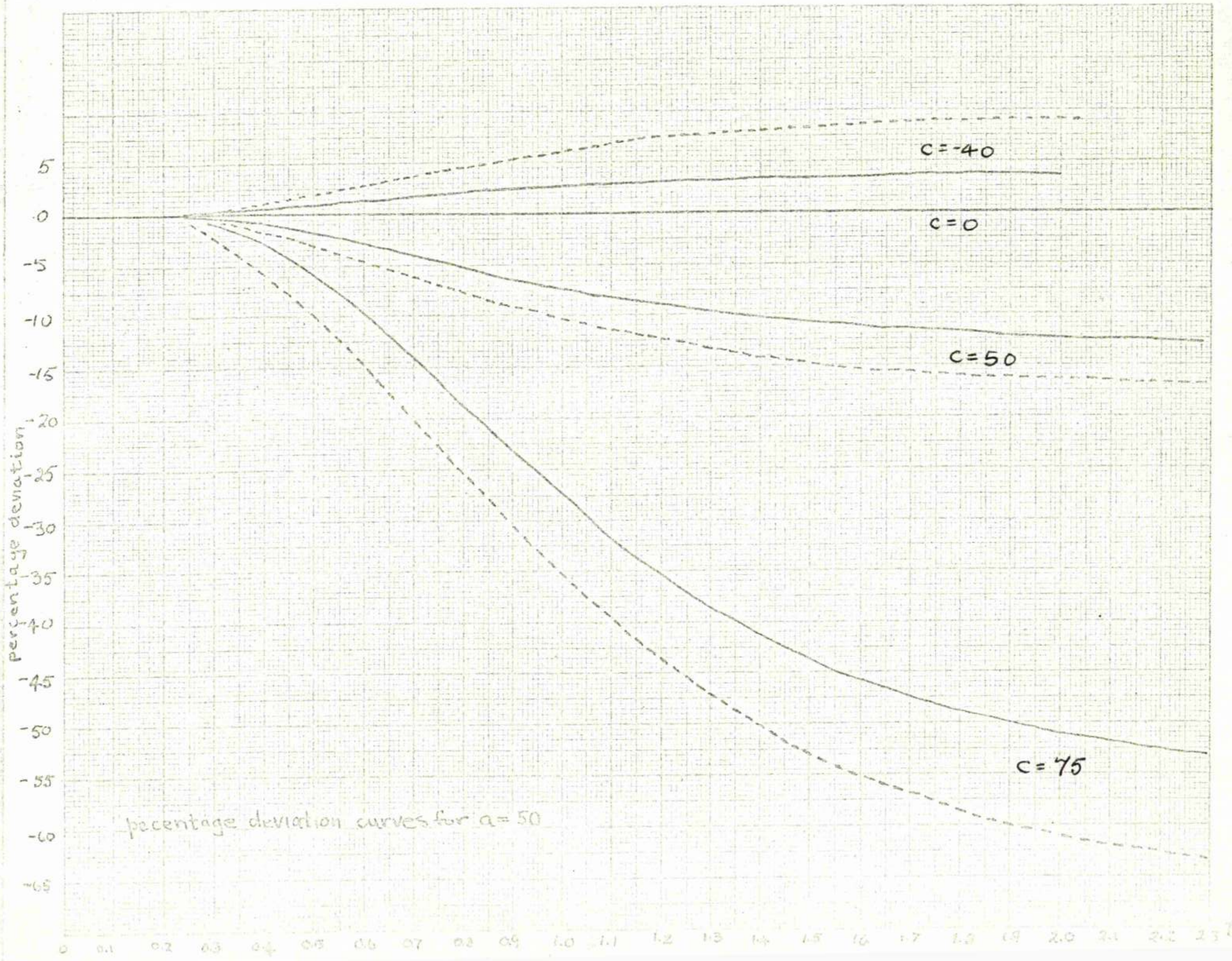
$$\text{percentage deviation} = \left(\frac{y_{\text{numerical}} - y_{\text{semi-analytic}}}{y_{\text{numerical}} + y_{\text{semi-analytic}}}\right) \times 200$$

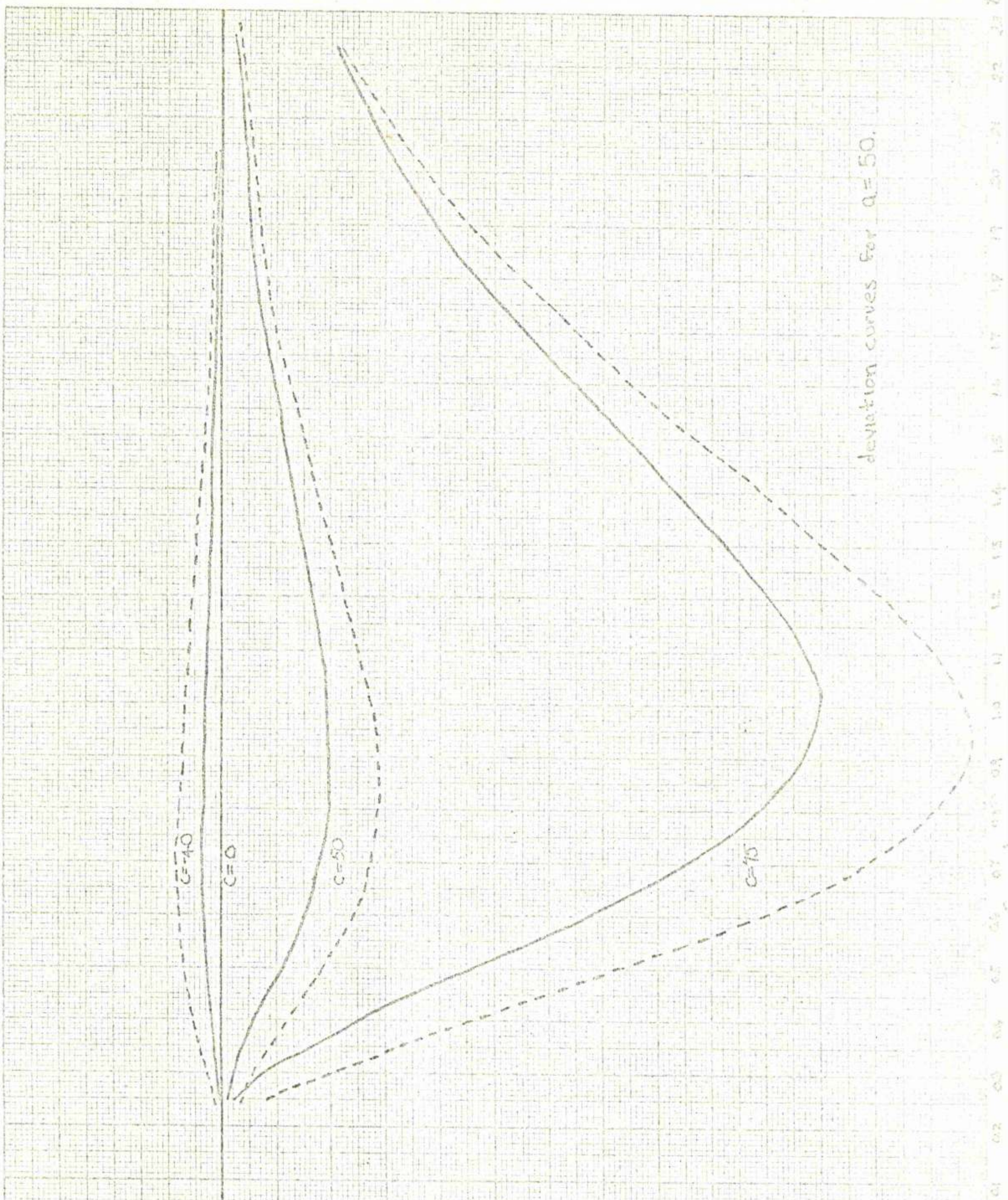
The errors obtained are shown in graphs 9 to 14, the grouping of the parameters being the same as in results graphs. Graphs 9, 11 and 13 show the percentage deviation after the maximum while graphs 10, 12 and 14 show the true deviation. Only the errors for a = 50 were shown for Perturbation (the dotted curves in graphs 9 and 10) but for particular values of c the deviation and percentage deviation in these cases were the worst found. Because Perturbation and Picard's method produce similar exponential solutions, as can be seen from graphs 9 and 10, were not plotted. Since the Perturbation solutions approximated to the true solution with less exponenti it might be expected that its solution is less accurate. Assuming the numerical methods produce the closest approximation, this appeared to be the case for all parameters.

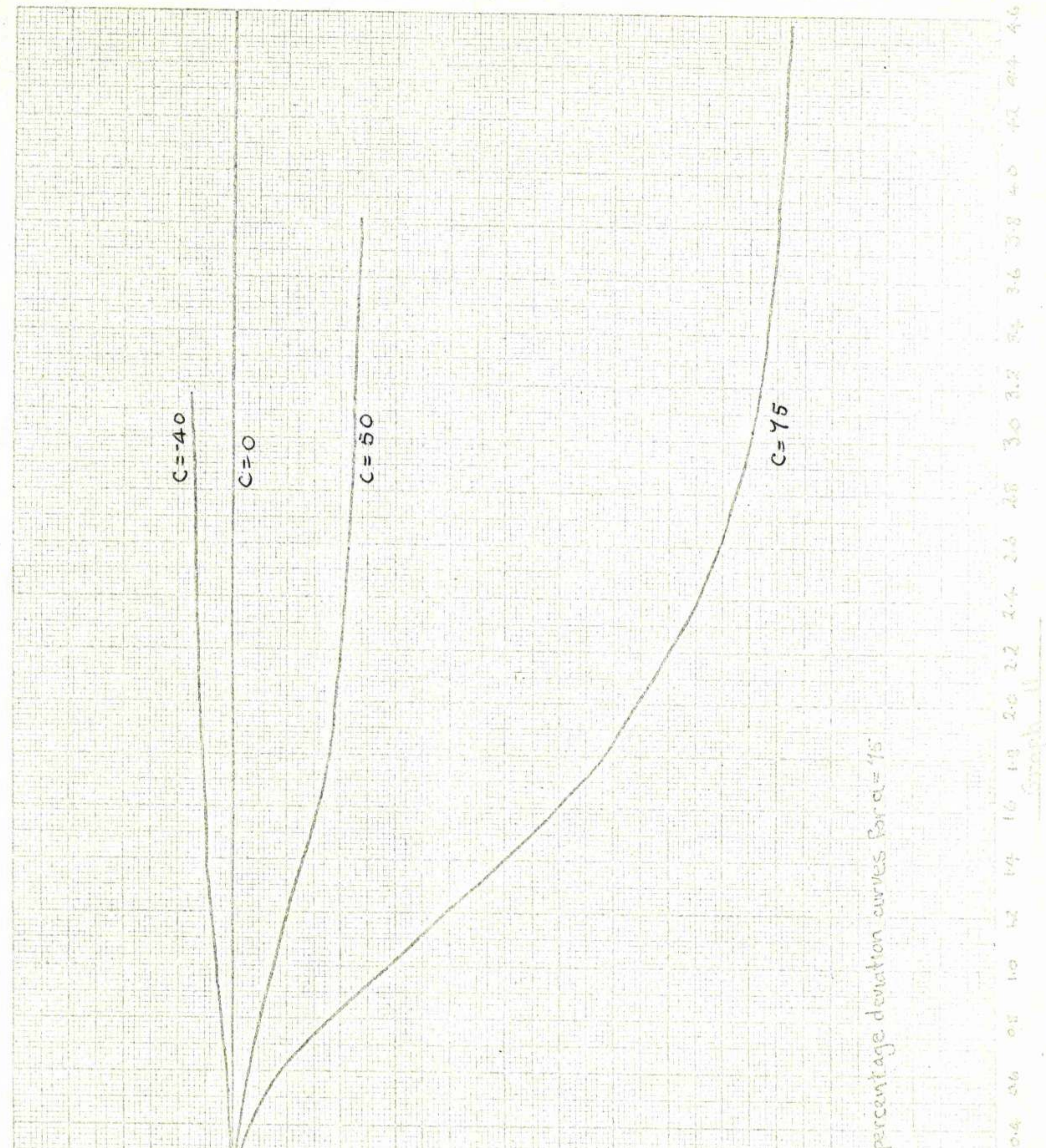
In the curves of the solutions, it will be seen that there are graphs for a = 25 but no error curves. It turned out that in Picard's Method, one of the coefficients became infinite for this value of a and hence no solution could be found. For this parameter the values of α and β are respectively 5 and 20. In the last part of the procedure CONST of the pertinent programme we have that the coefficient

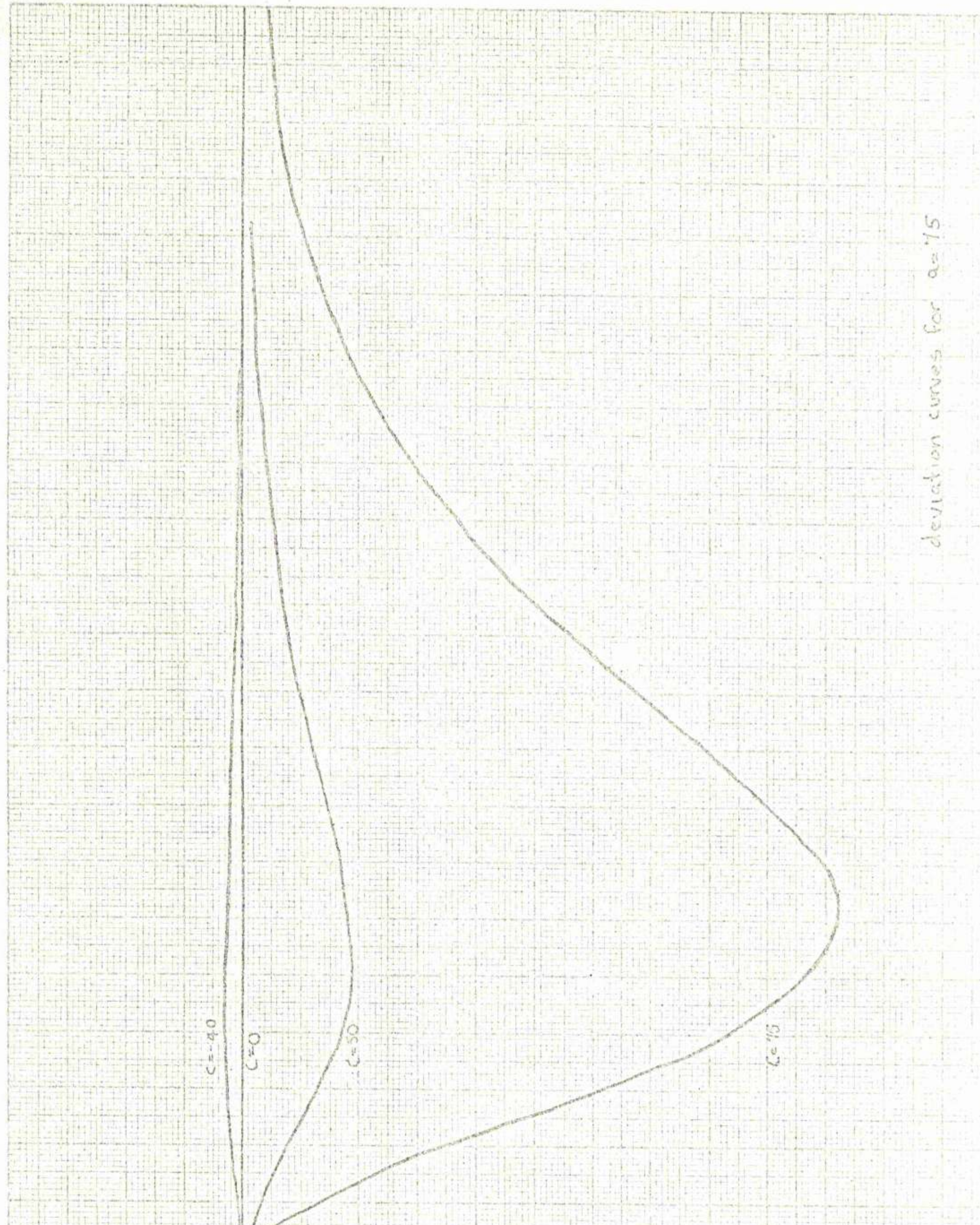
$$A[i,j] := B[i,j] \times d \,/\, (a \times ((i-j+1) \times f + (j-1) \times g)^2 - b \times ((i-j+1) \times f + (j-1) \times g) + c) \qquad (78)$$

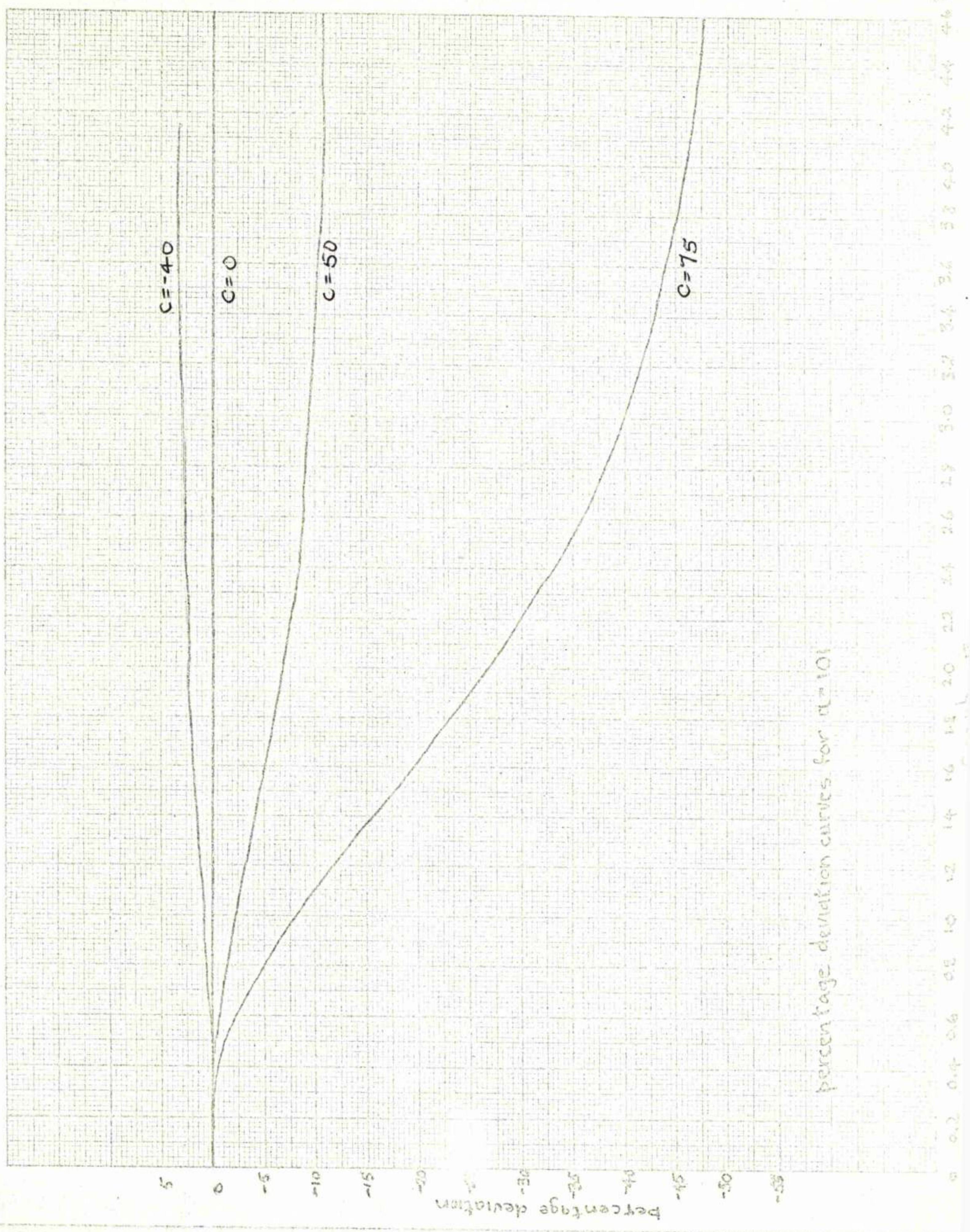
$$(f \equiv \alpha,\ g \equiv \beta,\ d \equiv c,\ c \equiv b,\ b \equiv a,\ a = 1)$$


c=-40
c=0
c=50
c=75
pecentage deviation curves for a= 50
percentage deviation
5
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
-55
-60
-65
t

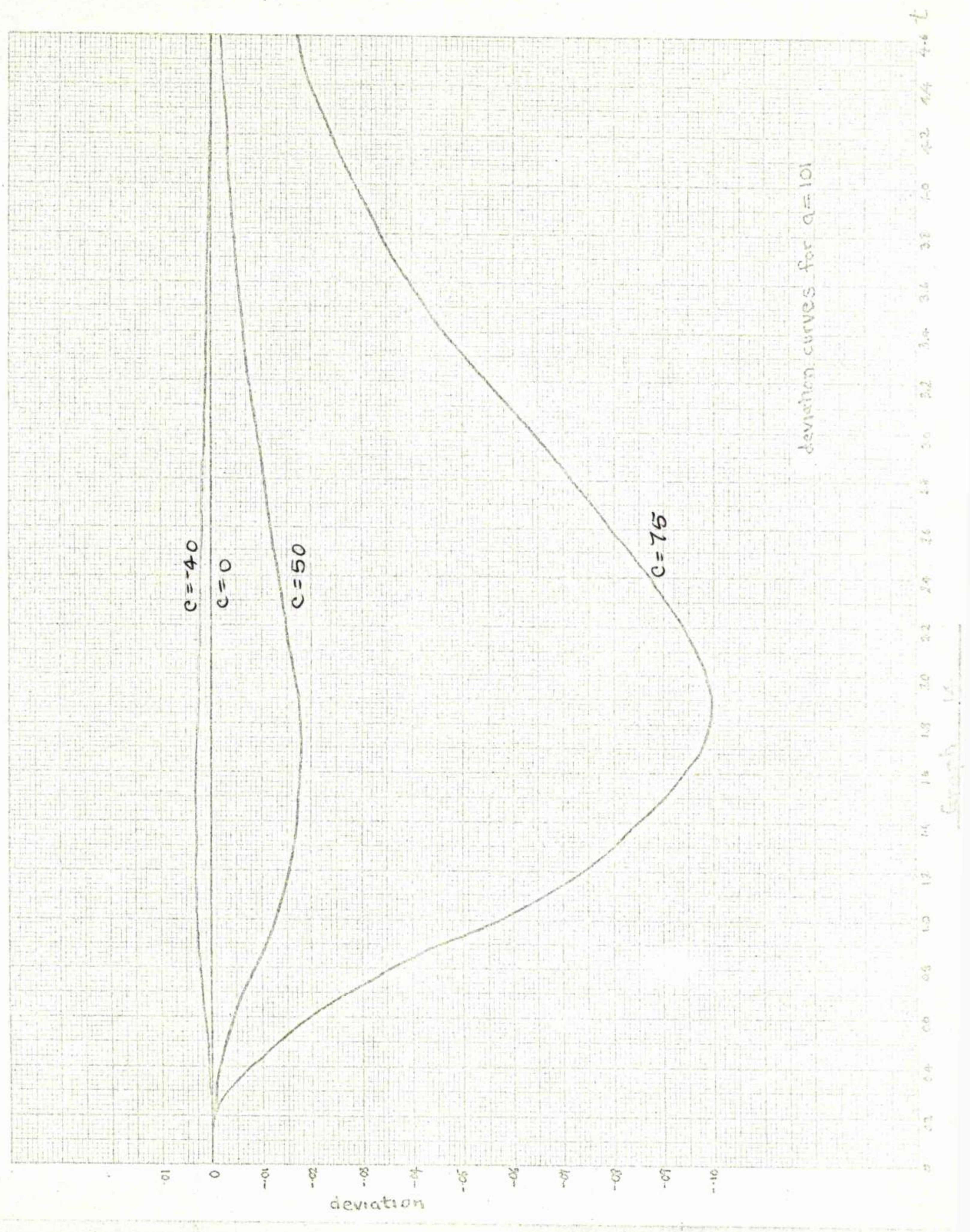

deviation curves for a = 50
c=40
c=0
c=50
c=75
deviation


C=-40
C=0
C=50
C=75
percentage deviation curves for a=15
Graph 11
percentage deviation
5
0
-5
-10
-18
-20
-25
-30
-35
-40
-45
-50
-55
0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
4.2
4.4
4.6


deviation curves for a=·75
c=-·40
c=0
c=·50
c=·75
·01
0.0
-·01
-·02
-·03
-·04
-·05
-·06
-·07
-·08
-·09
-·10
deviation



percentage deviation curves for a=101


c = -40
c = 0
c = 50
c = 75
deviation curves for q = 101
deviation
t

The denominator of this equation is zero when f=5, g=20, i=4 and j=1. Thus, for this particular parameter and using this method, the solution can only be worked out to the first non-linear approximation (i.e. when i only goes up to the value 2).

By looking at the percentage deviation curves it is obvious that in the region beyond the maximum, the numerical and semi-analytic methods do not agree. This is presumably because, as in all series solutions expanded about a point, the solution will only be accurate within a certain region. Some idea of how the error behaves is found from an approximate analytic solution to the differential equation outside upper boundary value. If we assume that y" is zero for the exponential decay, we have

$$ay' + by = cy^2$$

$$\text{i.e. } t = a\int \frac{dy}{y(cy-b)} + K' = \frac{a}{b} \ln \frac{cy-b}{Ky} \qquad K, K' = \text{constants}$$

$$\therefore \quad y = -\frac{b}{K} e^{-bt/a} \Big/ \left(1 - \frac{c}{K} e^{-bt/a}\right) \tag{79}$$

Thus if y is to have a valid exponential expansion

$$\frac{c}{k} e^{-bt/a} < 1 \tag{80}$$

K is a constant which, since we are dealing with a region of the curve outside the boundary conditions, must be found from elsewhere. In an approximate way, the numerical solutions can satisfy this demand. Comparing K found from various numerical points, it is found that it stays as constant as can be expected since y" = 0. See figure (14)

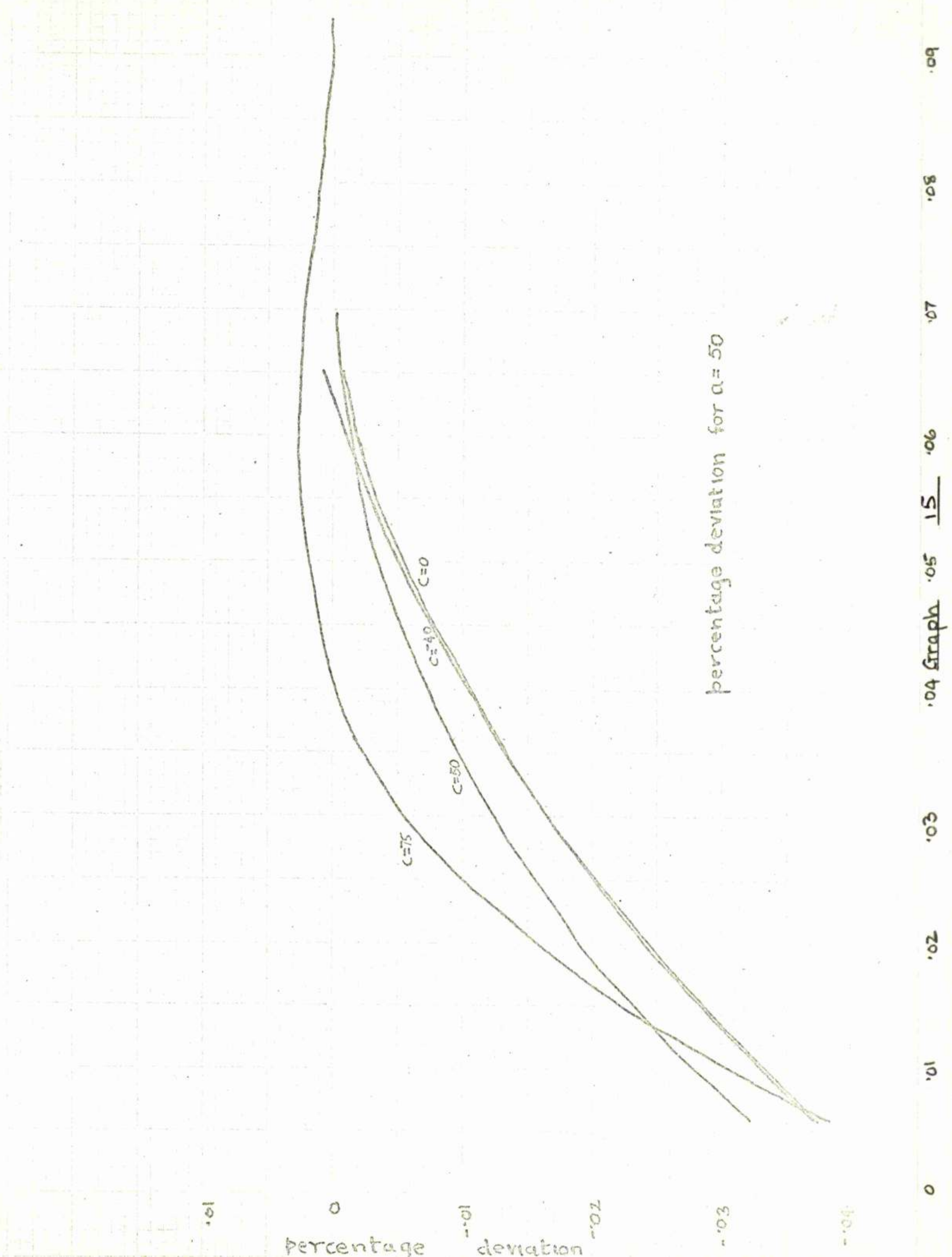
t	1.55	2.75	3.95
K	47.22	47.64	48.14

for $\begin{cases} a = 101 \\ c = 50 \end{cases}$

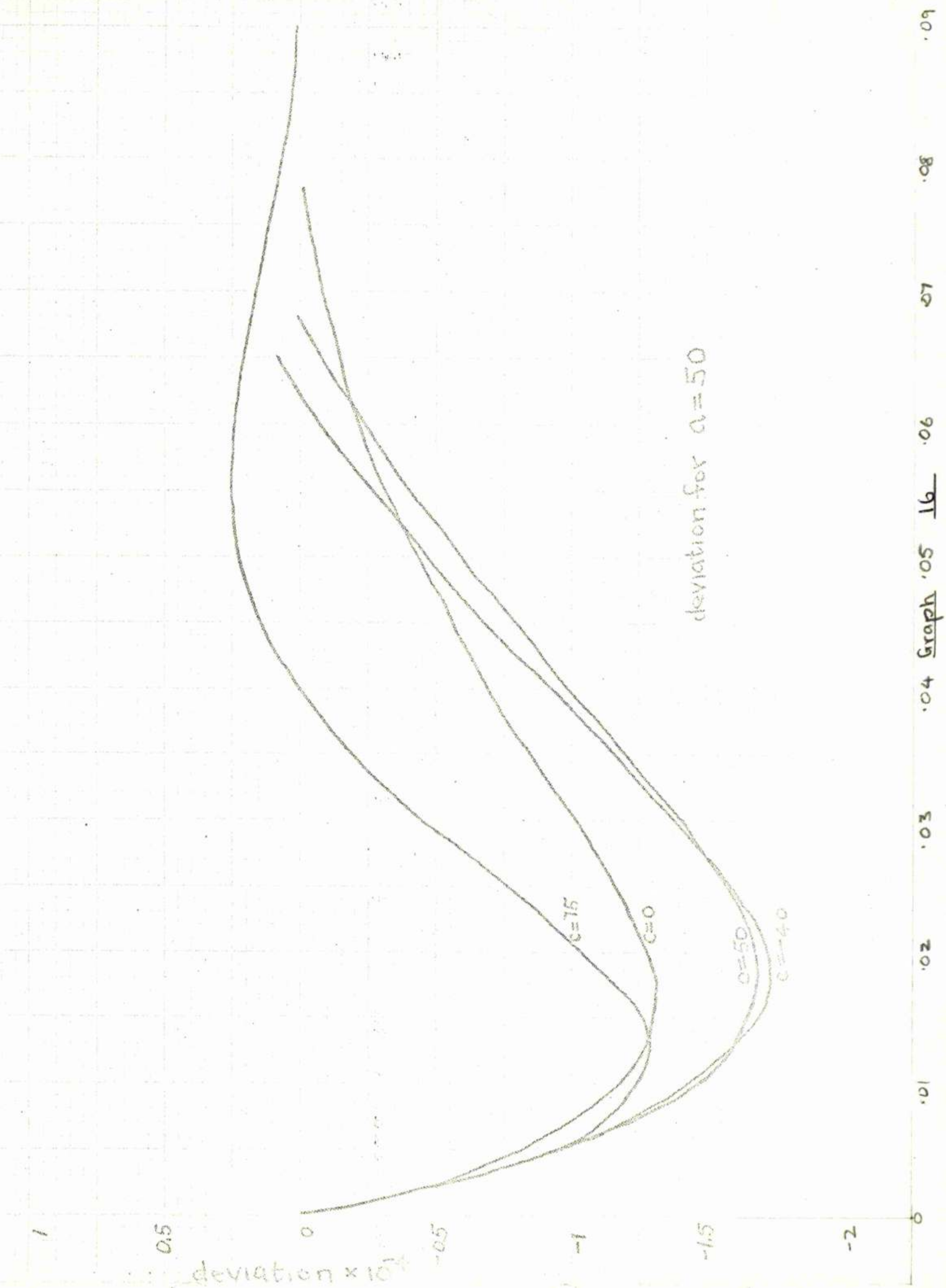
figure (14)

Using an approximate value of K, the inequality (80) should determine the minimum value of t such that the expansion is valid. The fact that this is not the case is probably because the expansion, although of exponential form, is not the same as was derived for the solution. There is, however, a qualitative agreement between the error found and size of $ce^{-bt/a}/K$. If c is increased, an increase in the error is found as would be expected in the above. Similarly if a is decreased, the error increases. This is not at first sight obvious since in decreasing a, the value of above exponential decreases, but it is found that K becomes smaller fast enough to swamp this effect.

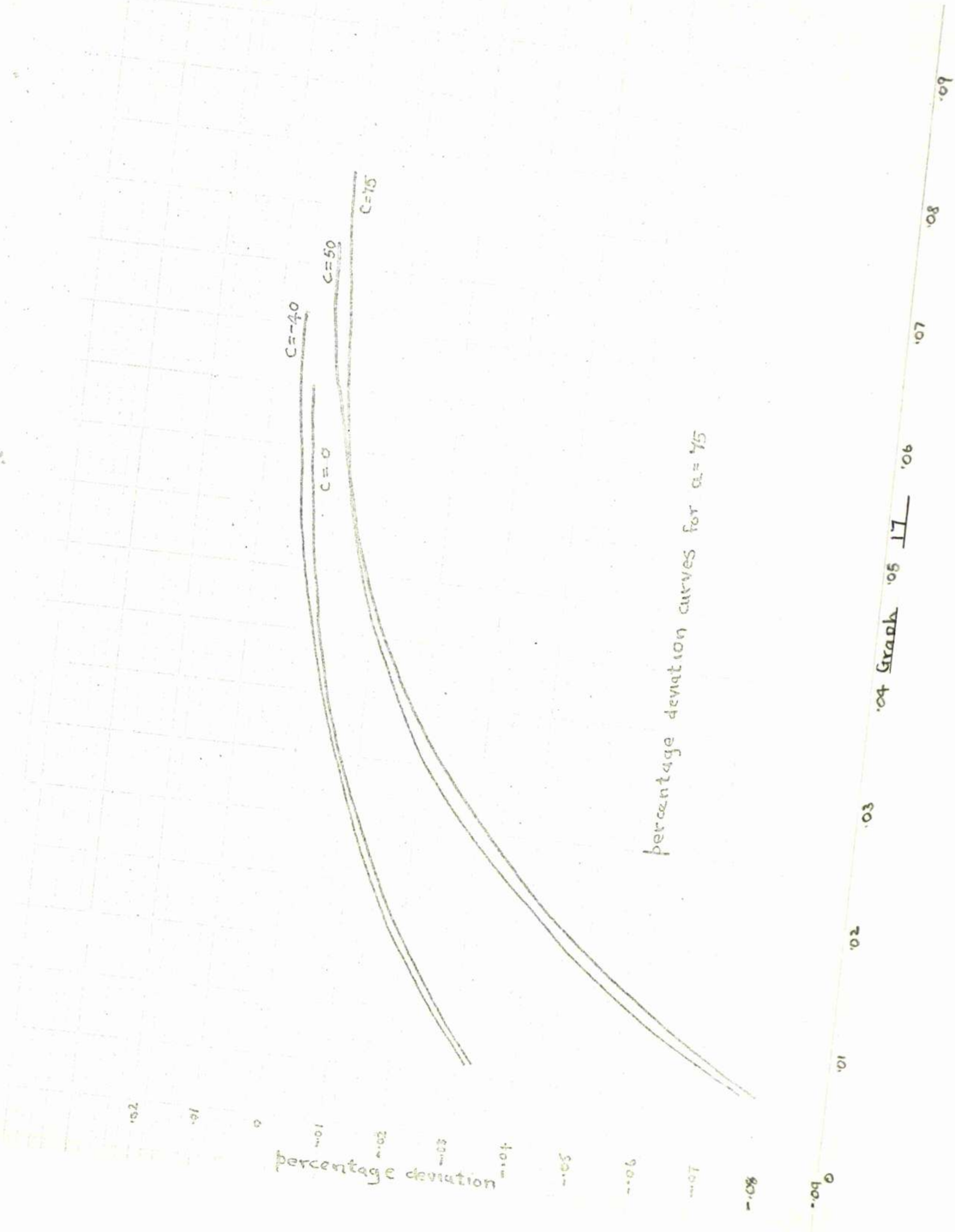
Because the percentage deviations after the maximum were on average several orders of magnitude greater than the errors between zero and the maximum, the two regions could not be plotted on the same graph. The percentage deviations and deviations for the region between the boundary conditions are therefore shown on graphs (15) - (20). It is obvious comparing the deviation before and after the maximum that in the former case there is quite good agreement between the different techniques but not in the latter. To find if there was analytic basis for the errors found, the test for convergence for Picard's method was investigated. The convergence condition is easily found for the equation $y'' = f(t,y)$ and it therefore had to be



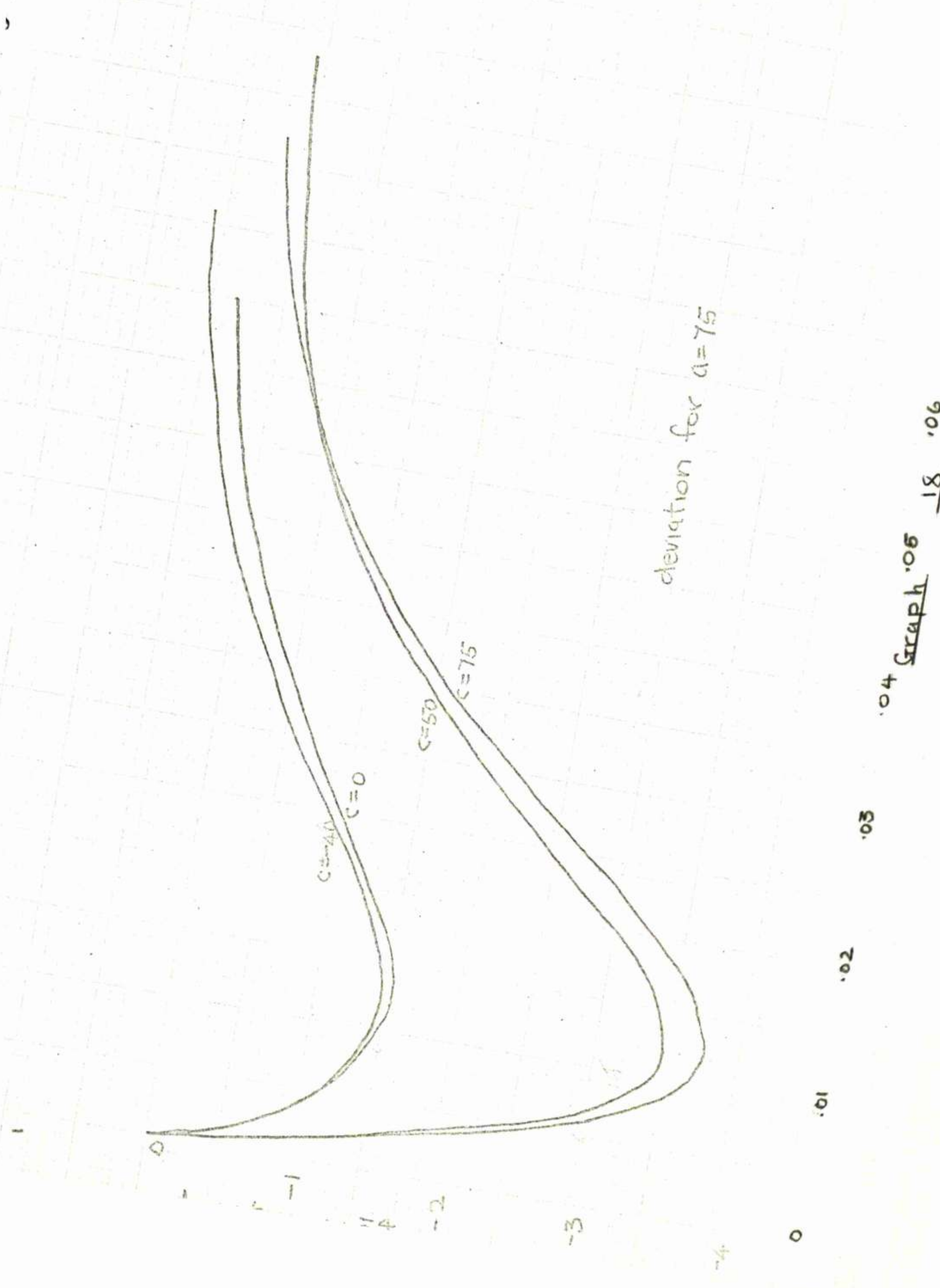
Graph 15 percentage deviation for a = 50


deviation for a=50
c=75
c=0
c=50
c=-40
deviation ×10⁴
1
0.5
0
-0.5
-1
-1.5
-2
0
.01
.02
.03
.04
.05
.06
.07
.08
.09

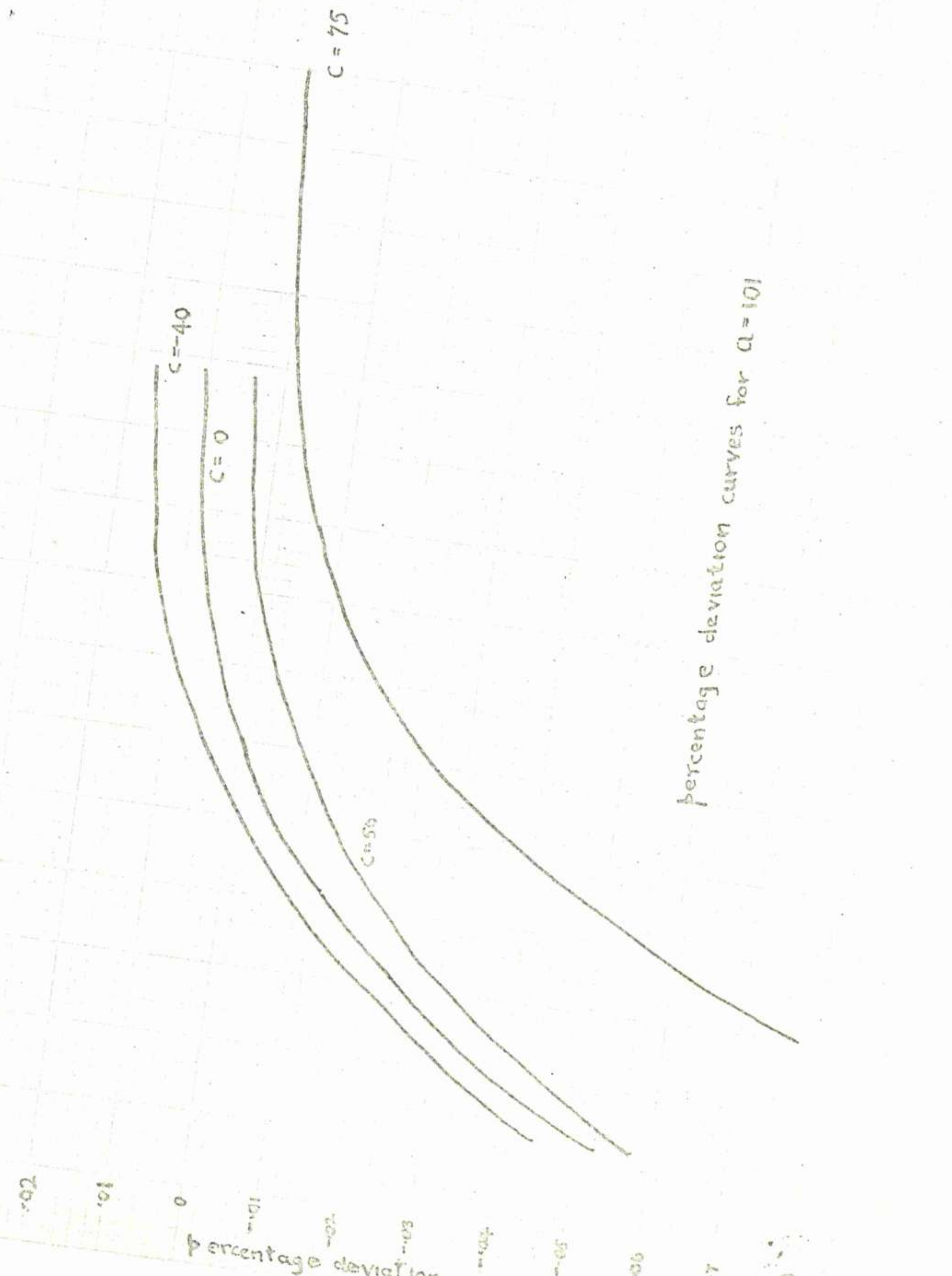
Graph 16

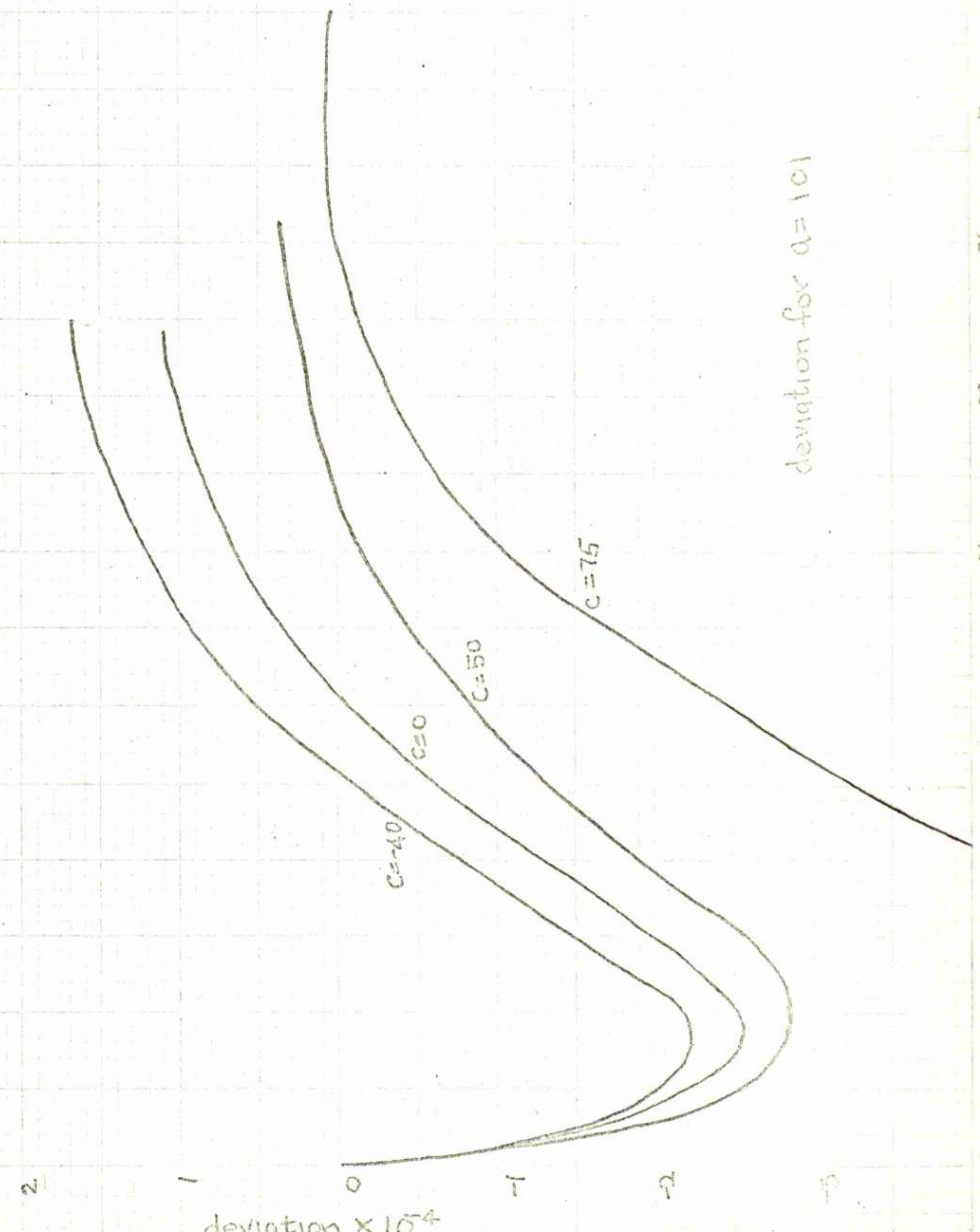

percentage deviation curves for a = ·75
C=-40
C=0
C=50
C=75
percentage deviation
·01
·02
·03
·04
·05
·06
·07
·08
·09

Graph 17


deviation for a=75
c=0
c=40
c=50
c=75
0
.01
.02
.03
.04
.05
.06
.07
.08
.09
0
-1
-2
-3
-4

Graph 18


C = 75
C = -40
C = 0
percentage deviation curves for a = 101
percentage deviation
.02
.01
0
-.01
-.02
-.03
-.04
-.05
-.06
-.07
-.08


deviation for a=101
c=75
c=50
c=0
c=-40
2
1
0
-1
-2
-3
deviation × 10⁻⁴

generalised for the equation $y'' = f(t,y,y')$. The condition turned out to be

$$\left(M + \frac{N}{\sqrt[n]{e_1}}\frac{de_1}{dt}\right)R^2 < 1 \qquad (81)$$

where $M = \frac{\partial f}{\partial y}$ $\qquad N = \frac{\partial f}{\partial y'}$

$e_1 = y_1 - y$ $\qquad$ n = number of approximations

y_1 = first approximation to the solution

y = the analytic solution

R = upper limit - lower limit

When y" is not a function of y', the condition is very easy to apply, but no way was found of deriving any value for the function de_1/dt. The term $1/\sqrt[n]{e}$ probably could however be ignored if n was large enough. In the equation being investigated $\frac{\partial f}{\partial y'}$, has the value a. Since this must be greater than 20, it is doubtful if the second term of the inequality (81) could be left out.

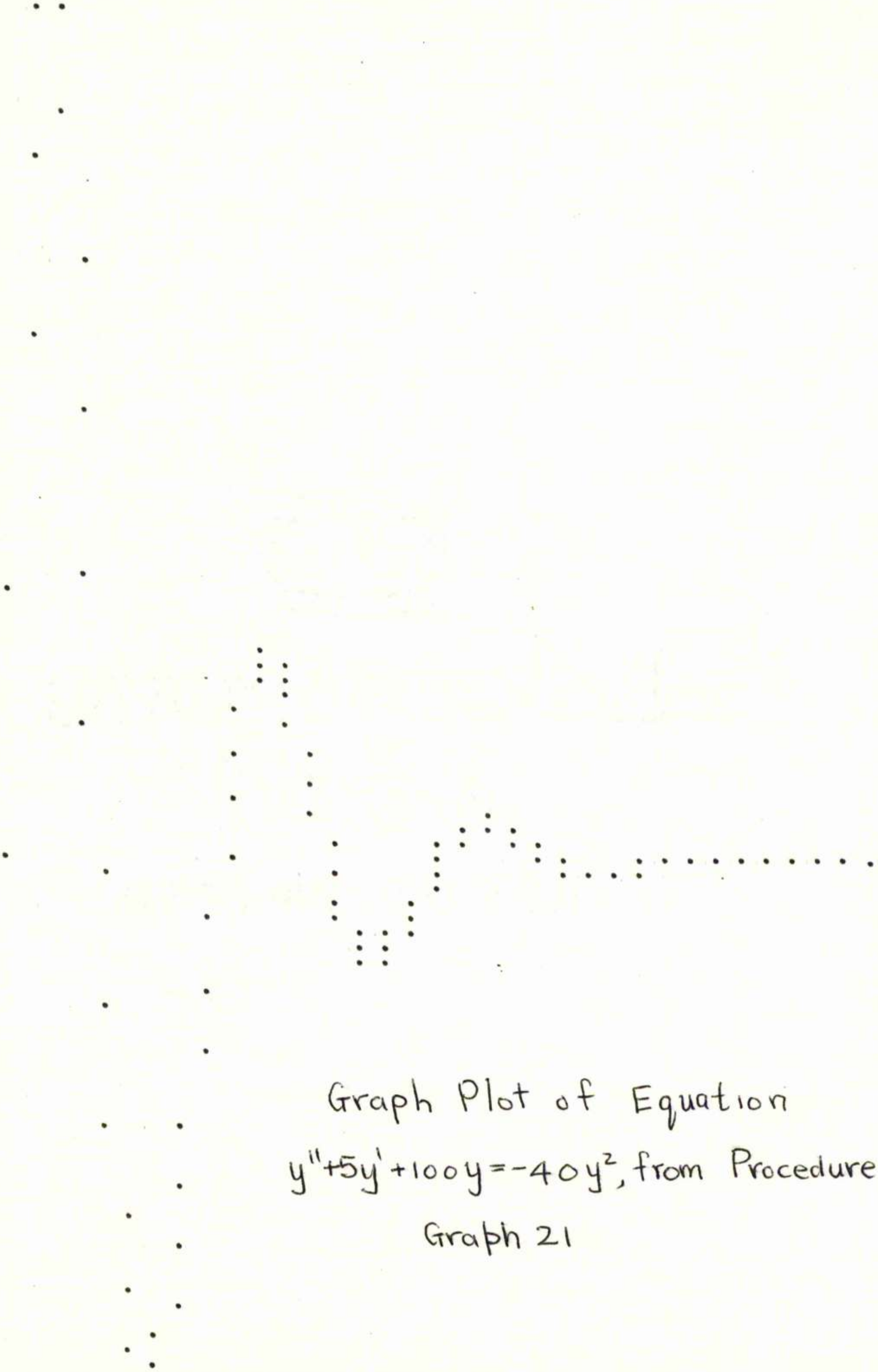
Since it would have taken up a very large amount of space to include the numerical results for all the parameters, it was decided only to include, for each method, one set of results. For all methods the parameters included are for a=50, c=-40. These are shown in Appendix 1.

3.3. Conclusion

In dealing with this problem, it has become obvious that, in fitting an exponential series, such as taken here, to the differential equation, the solution will only be accurate within the boundary values. Since the solution is still exponential in form it is possible that an adaption of Collocation or Least Squares could give an exponential series solution with different coefficients and exponents. This would probably be much more accurate outside the upper boundary condition.

It is unfortunate that the numerical method was unstable when a was close to zero, as the two results would have acted as a check on each other. An investigation of the difference between the analytic linear solutions and the programmed methods would, however, give some idea of the absolute error involved in computing those latter results. Although this was not analysed, in calculating a few points analytically, the exponential solutions in the linear cases seemed to be slightly more accurate than the purely numerical solutions (for the region before the maximum).

One of the main difficulties in solving this problem has certainly been fitting the upper boundary condition. Both in the numerical and semi-analytic approaches, the condition $y(t_m) = 1$ had to be iterated onto and, because of this, the solutions took longer to compute than they would have taken under more normal conditions.



Graph Plot of Equation
$y''+5y'+100y=-40y^2$, from Procedure.
Graph 21

APPENDIX 1

Numerical Results for Equation $y''+50y'+100y=-40y^2$

for Methods:

(1) Finite Difference

(2) Step-by-Step

(3) Picard

(4) Perturbation

input parameters dy/dt,y×y and interval
+50.0; -40.0; +0.0015;

X and Y values

+0.0000;	+0.0000 0000;	+0.0015;	+0.0808 6767;	+0.0030;	+0.1571 830
+0.0045;	+0.2273 1404;	+0.0060;	+0.2923 2158;	+0.0075;	+0.3525 589
+0.0090;	+0.4083 5462;	+0.0105;	+0.4600 1382;	+0.012[illegible];	+0.5078 2 50
+0.0135;	+0.5520 3877;	+0.0150;	+0.5929 1439;	+0.0165;	+0.6306 760
+0.0180;	+0.6655 3670;	+0.0195;	+0.6976 9454;	+0.0210;	+0.7273 341
+0.0225;	+0.7546 2757;	+0.0240;	+0.7797 3489;	+0.0255;	+0.8028 053
+0.0270;	+0.8239 7811;	+0.0285;	+0.8433 8270;	+0.0300;	+0.8611 399
+0.0315;	+0.8773 6244;	+0.0330;	+0.8921 5519;	+0.0345;	+0.9056 160
+0.0360;	+0.9178 3630;	+0.0375;	+0.9289 0103;	+0.0390;	+0.9388 896
+0.0405;	+0.9478 7602;	+0.042[illegible];	+0.9559 2932;	+0.0435;	+0.9631 138
+0.0450;	+0.9694 8973;	+0.0465;	+0.9751 1289;	+0.0480;	+0.9800 356
+0.0495;	+0.9843 0661;	+0.0510;	+0.9879 7136;	+0.0525;	+0.9910 722
+0.0540;	+0.9936 4885;	+0.0555;	+0.9957 3804;	+0.0570;	+0.9973 742
+0.0585;	+0.9985 8961;	+0.0600;	+0.9994 1404;	+0.0615;	+0.9998 754
+0.0630;	+1.0000 0000;	+0.0645;	+0.9998 1190;	+0.0660;	+0.9993 338
+0.0675;	+0.9985 8703;	+0.0690;	+0.9975 9112;	+0.0705;	+0.9963 645
+0.0720;	+0.9949 2441;	+0.0735;	+0.9932 8676;	+0.0750;	+0.9914 664
+0.0765;	+0.9894 7752;	+0.0780;	+0.9873 3282;	+0.0795;	+0.9850 444
+0.0810;	+0.9826 2375;	+0.0825;	+0.9800 8116;	+0.0840;	+0.9774 265
+0.0855;	+0.9746 6890;	+0.0870;	+0.9718 1686;	+0.0885;	+0.9688 783
+0.0900;	+0.9658 6064;	+0.0915;	+0.9627 7072;	+0.0930;	+0.9596 149
+0.0945;	+0.9563 9935;	+0.0960;	+0.9531 2943;	+0.0975;	+0.9498 103
+0.0990;	+0.9464 4702;	+0.1005;	+0.9430 4384;	+0.1020;	+0.9396 050
+0.1035;	+0.9361 3440;	+0.1050;	+0.9326 3566;	+0.1065;	+0.9291 121
+0.1080;	+0.9255 6691;	+0.1095;	+0.9220 0295;	+0.1110;	+0.9184 229

+0.1125;	+0.9148 2935;	+0.1140;	+0.9112 2456;	+0.1155;	+0.9076 1071
+0.1170;	+0.9039 8982;	+0.1185;	+0.9003 6374;	+0.1200;	+0.8967 342
+0.1215;	+0.8931 0281;	+0.1230;	+0.8894 7106;	+0.1245;	+0.8858 4031
+0.1260;	+0.8822 1185;	+0.1275;	+0.8785 8684;	+0.1290;	+0.8749 6639
+0.1305;	+0.8713 5149;	+0.1320;	+0.8677 4308;	+0.1335;	+0.8641 42
+0.1350;	+0.8605 4909;	+0.1365;	+0.8569 6503;	+0.1380;	+0.8533 9050
+0.1395;	+0.8498 2612;	+0.1410;	+0.8462 7246;	+0.1425;	+0.8427 3004
+0.1440;	+0.8391 9932;	+0.1455;	+0.8356 8076;	+0.1470;	+0.8321 7473
+0.1485;	+0.8286 8161;	+0.1500;	+0.8252 0171;	+0.1515;	+0.8217 3535
+0.1530;	+0.8182 8278;	+0.1545;	+0.8148 4425;	+0.1560;	+0.8114 1998
+0.1575;	+0.8080 1016;	+0.1590;	+0.8046 1497;	+0.1605;	+0.8012 3456
+0.1620;	+0.7978 6906;	+0.1635;	+0.7945 1860;	+0.1650;	+0.7911 8328
+0.1665;	+0.7878 6318;	+0.1680;	+0.7845 5838;	+0.1695;	+0.7812 6894
+0.1710;	+0.7779 9491;	+0.1725;	+0.7747 3633;	+0.1740;	+0.7714 9323
+0.1755;	+0.7682 6563;	+0.1770;	+0.7650 5353;	+0.1785;	+0.7618 5694
+0.1800;	+0.7586 7587;	+0.1815;	+0.7555 1028;	+0.1830;	+0.7523 6018
+0.1845;	+0.7492 2552;	+0.1860;	+0.7461 0629;	+0.1875;	+0.7430 0246
+0.1890;	+0.7399 1397;	+0.1905;	+0.7368 4080;	+0.1920;	+0.7337 8288
+0.1935;	+0.7307 4018;	+0.1950;	+0.7277 1264;	+0.1965;	+0.7247 0020
+0.1980;	+0.7217 0281;	+0.1995;	+0.7187 2040;	+0.2010;	+0.7157 5291
+0.2025;	+0.7128 0027;	+0.2 40;	+0.7098 6242;	+0.2055;	+0.7069 3929
+0.2070;	+0.7040 3080;	+0.2085;	+0.7011 3689;	+0.2100;	+0.6982 5747
+0.2115;	+0.6953 9249;	+0.2130;	+0.6925 4186;	+0.2145;	+0.6897 0550
+0.2160;	+0.6868 8334;	+0.2175;	+0.6840 7530;	+0.2190;	+0.6812 8131
+0.2205;	+0.6785 0128;	+0.2220;	+0.6757 3514;	+0.2235;	+0.6729 8280
+0.2250;	+0.6702 4420;				

input parameters dy/dt,yxy,interval and limit
+50.0; -40.0; +0.0150; +1.9200;

X and Y values

+0.2100;	+0.6983;	+0.2250;	+0.6702;	+0.2400;	+0.6436;
+0.2550;	+0.6183;	+0.2700;	+0.5941;	+0.2850;	+0.5711;
+0.3000;	+0.5492;	+0.3150;	+0.5282;	+0.3300;	+0.5082;
+0.3450;	+0.4892;	+0.3600;	+0.4709;	+0.3750;	+0.4535;
+0.3900;	+0.4368;	+0.4050;	+0.4208;	+0.4200;	+0.4055;
+0.4350;	+0.3908;	+0.4500;	+0.3767;	+0.4650;	+0.3632;
+0.4800;	+0.3503;	+0.4950;	+0.3379;	+0.5100;	+0.3260;
+0.5250;	+0.3145;	+0.5400;	+0.3035;	+0.5550;	+0.2930;
+0.5700;	+0.2828;	+0.5850;	+0.2730;	+0.6000;	+0.2636;
+0.6150;	+0.2546;	+0.6300;	+0.2459;	+0.6450;	+0.2375;
+0.6600;	+0.2295;	+0.6750;	+0.2217;	+0.6900;	+0.2142;
+0.7050;	+0.2070;	+0.7200;	+0.2001;	+0.7350;	+0.1934;
+0.7500;	+0.1869;	+0.7650;	+0.1807;	+0.7800;	+0.1747;
+0.7950;	+0.1689;	+0.8100;	+0.1633;	+0.8250;	+0.1580;
+0.8400;	+0.1528;	+0.8550;	+0.1477;	+0.8700;	+0.1429;
+0.8850;	+0.1382;	+0.9000;	+0.1337;	+0.9150;	+0.1294;
+0.9300;	+0.1252;	+0.9450;	+0.1211;	+0.9600;	+0.1172;
+0.9750;	+0.1134;	+0.9900;	+0.1097;	+1.0050;	+0.1062;

+1.0200;	+0.1028;	+1.0350;	+0.0995;	+1.0500;	+0.0963;
+1.0650;	+0.0932;	+1.0800;	+0.0902;	+1.0950;	+0.0873;
+1.1100;	+0.0845;	+1.1250;	+0.0818;	+1.1400;	+0.0792;
+1.1550;	+0.0767;	+1.1700;	+0.0742;	+1.1850;	+0.0719;
+1.2000;	+0.0696;	+1.2150;	+0.0674;	+1.2300;	+0.0653;
+1.2450;	+0.0632;	+1.2600;	+0.0612;	+1.2750;	+0.0593;
+1.2900;	+0.0574;	+1.3050;	+0.0556;	+1.3200;	+0.0538;
+1.3350;	+0.0521;	+1.3500;	+0.0505;	+1.3650;	+0.0489;
+1.3800;	+0.0474;	+1.3950;	+0.0459;	+1.4100;	+0.0444;
+1.4250;	+0.0430;	+1.4400;	+0.0417;	+1.4550;	+0.0404;
+1.4700;	+0.0391;	+1.4850;	+0.0379;	+1.5000;	+0.0367;
+1.5150;	+0.0355;	+1.5300;	+0.0344;	+1.5450;	+0.0334;
+1.5600;	+0.0323;	+1.5750;	+0.0313;	+1.5900;	+0.0303;
+1.6050;	+0.0294;	+1.62 ;	+0.0285;	+1.6350;	+0.0276;
+1.6500;	+0.0267;	+1.6650;	+0.0259;	+1.6800;	+0.0251;
+1.6950;	+0.0243;	+1.7100;	+0.0235;	+1.7250;	+0.0228;
+1.7400;	+0.0221;	+1.7550;	+0.0214;	+1.770 ;	+0.0207;
+1.7850;	+0.0201;	+1.8000;	+0.0195;	+1.8150;	+0.0189;
+1.8300;	+0.0183;	+1.8450;	+0.0177;	+1.8600;	+0.0172;
+1.8750;	+0.0166;	+1.8900;	+0.0161;	+1.9050;	+0.0156;
+1.9200;	+0.0151;	+1.9350;	+0.0147;	+1.9500;	+0.0142;
+1.9650;	+0.0138;	+1.9800;	+0.0133;	+1.9950;	+0.0129;
+2.0100;	+0.0125;	+2.0250;	+0.0121;	+2.0400;	+0.0118;
+2.0550;	+0.0114;	+2.0700;	+0.0110;	+2.0850;	+0.0107;
+2.1000;	+0.0104;	+2.1150;	+0.0101;	+2.1300;	+0.0097;

successive number of iterations 4; 10;

input parameters dy/dt, yxy, upper limit, interval and approximation

+50.0; -40.0; +2.2[illegible]00; +0.0015; 2;

coefficients

$+8.6090_{10}$ -1;	-2.7134_{10} +0;			
$+1.9174_{10}$ -1;	$+1.2961_{10}$ +0;	-5.3424_{10} -2;		
$+1.9396_{10}$ -1;	$+3.0323_{10}$ -1;	$+4.7721_{10}$ -2;	-1.8696_{10} -4;	
$+6.5566_{10}$ -2;	-1.8186_{10} -1;	-1.0487_{10} -2;	$+8.5655_{10}$ -5;	-2.4627_{10} -7;

X and Y values

+0.0000;	+0.0000 0000;	+0.0015;	+0.0815 1365;	+0.0030;	+0.1571 1907;
+0.0045;	+0.2272 2524;	+0.0060;	+0.2922 1199;	+0.0075;	+0.3524 3223;
+0.0090;	+0.4082 1396;	+0.0105;	+0.4598 6214;	+0.0120;	+0.5076 6036;
+0.0135;	+0.5518 7246;	+0.0150;	+0.5927 4391;	+0.0165;	+0.6305 032[illegible];
+0.0180;	+0.6653 6300;	+0.0195;	+0.6975 2137;	+0.0210;	+0.7271 6274;
+0.0225;	+0.7544 5888;	+0.0240;	+0.7795 6985;	+0.0255;	+0.8026 4475;
+0.0270;	+0.8238 2253;	+0.0285;	+0.8432 3271;	+0.0300;	+0.8609 9600;
+0.0315;	+0.8772 2491;	+0.0330;	+0.8920 2436;	+0.0345;	+0.9054 9214;
+0.0360;	+0.9177 1948;	+0.0375;	+0.9287 9141;	+0.0390;	+0.9387 8724;
+0.0405;	+0.9477 8094;	+0.042 ;	+0.9558 4151;	+0.0435;	+0.9630 3328;
+0.0450;	+0.9694 1628;	+0.0465;	+0.9750 4648;	+0.0480;	+0.9799 7612;
+0.0495;	+0.9842 5390;	+0.0510;	+0.9879 2526;	+0.0525;	+0.9910 3260;
+0.0540;	+0.9936 1546;	+0.0555;	+0.9957 1074;	+0.0570;	+0.9973 5283;
+0.0585;	+0.9985 7385;	+0.0600;	+0.9994 0374;	+0.0615;	+0.9998 7044;
+0.0630;	+1.0000 0000;	+0.0645;	+0.9998 1673;	+0.0660;	+0.9993 4331;
+0.0675;	+0.9986 0088;	+0.0690;	+0.9976 0916;	+0.0705;	+0.9963 8656;
+0.0720;	+0.9949 5023;	+0.0735;	+0.9933 1617;	+0.0750;	+0.9914 9931;
+0.0765;	+0.9895 1355;	+0.0780;	+0.9873 7188;		

+0.0795;	+0.9850 8638;	+0.1095;	+0.9220 7577;	+0.1395;	+0.8499 1868;
+0.1695;	+0.7814 1365;	+0.1995;	+0.7189 7120;	+0.2295;	+0.6625 2449;
+0.2595;	+0.6114 9240;	+0.2895;	+0.5652 5179;	+0.3195;	+0.5232 4028;
+0.3495;	+0.4849 6973;	+0.3795;	+0.4500 1978;	+0.4095;	+0.4180 2813;
+0.4395;	+0.3886 8158;	+0.4695;	+0.3617 0832;	+0.4995;	+0.3368 7154;
+0.5295;	+0.3139 6404;	+0.5595;	+0.2928 0385;	+0.5895;	+0.2732 3056;
+0.6195;	+0.2551 0215;	+0.6495;	+0.2382 9251;	+0.6795;	+0.2226 892 ;
+0.7095;	+0.2081 9166;	+0.7395;	+0.1947 0966;	+0.7695;	+0.1821 6195;
+0.7995;	+0.1704 7523;	+0.8295;	+0.1595 8310;	+0.8595;	+0.1494 2532;
+0.8895;	+0.1399 4707;	+0.9195;	+0.1310 9836;	+0.9495;	+0.1228 3350;
+0.9795;	+0.1151 1066;	+1.0095;	+0.1078 9145;	+1.0395;	+0.1011 4059;
+1.0695;	+0.0948 2561;	+1.0995;	+0.0889 1657;	+1.1295;	+0.0833 8582;
+1.1595;	+0.0782 0783;	+1.1895;	+0.0733 5894;	+1.2195;	+0.0688 1725;
+1.2495;	+0.0645 6244;	+1.2795;	+0.0605 7566;	+1.3095;	+0.0568 3937;
+1.3395;	+0.0533 3728;	+1.3695;	+0.0500 5424;	+1.3995;	+0.0469 7613;
+1.4295;	+0.0440 8980;	+1.4595;	+0.0413 8297;	+1.4895;	+0.0388 4422;
+1.5195;	+0.0364 6287;	+1.5495;	+0.0342 2896;	+1.5795;	+0.0321 3317;
+1.6095;	+0.0301 6681;	+1.6395;	+0.0283 2175;	+1.6695;	+0.0265 9039;
+1.6995;	+0.0249 6561;	+1.7295;	+0.0234 4076;	+1.7595;	+0.0220 0962;
+1.7895;	+0.0206 6635;	+1.8195;	+0.0194 0551;	+1.8495;	+0.0182 2198;
+1.8795;	+0.0171 1096;	+1.9095;	+0.0160 6799;	+1.9395;	+0.0150 8885;
+1.9695;	+0.0141 6960;	+1.9995;	+0.0133 0656;	+2.0295;	+0.0124 9627;
+2.0595;	+0.0117 3547;	+2.0895;	+0.0110 2113;	+2.1195;	+0.0103 5039;
+2.1495;	+0.0097 2058;	+2.1795;	+0.0091 2918;		

input parameters dy/dt,y×y,upper limit,interval and iterations

+50.0; -40.0; +2.2000; 0.0030; 3;

coefficients

+9.1167$_{10}$ -1;	-2.9167$_{10}$ +0;
+2.4684$_{10}$ -2;	-4.1340$_{10}$ -2;
+1.8792$_{10}$ +0;	+3.7604$_{10}$ -1;
-9.7899$_{10}$ -5;	+2.5191$_{10}$ -2;
-2.5867$_{10}$ -1;	

X and Y values

+0.0000;	+0.0000 0000;	+0.0030;	+0.1571 2068;	+0.0060;	+0.2922 1496;
+0.0090;	+0.4082 1808;	+0.0120;	+0.5076 6541;	+0.0150;	+0.5927 4965;
+0.0180;	+0.6653 6922;	+0.0210;	+0.7271 6921;	+0.0240;	+0.7795 7636;
+0.0270;	+0.8238 2889;	+0.0300;	+0.8610 0202;	+0.0330;	+0.892 2988;
+0.0360;	+0.9177 2437;	+0.0390;	+0.9387 9141;	+0.0420;	+0.9558 4490;
+0.0450;	+0.9694 1887;	+0.0480;	+0.9799 7792;	+0.0510;	+0.9879 2635;
+0.0540;	+0.9936 1594;	+0.0570;	+0.9973 5284;	+0.0600;	+0.9994 0349;
+0.0630;	+0.9999 9972;	+0.0660;	+0.9993 4328;	+0.0690;	+0.9976 0971;
+0.0720;	+0.9949 5172;	+0.0750;	+0.9915 0214;	+0.0780;	+0.9873 7650;
+0.0810;	+0.9826 7523;	+0.0840;	+0.9774 8564;	+0.0870;	+0.9718 8359;
+0.0900;	+0.9659 3500;	+0.0930;	+0.9596 9708;		
+0.0960;	+0.9532 1950;	+0.1260;	+0.8824 0963;	+0.1560;	+0.8118 1696;
+0.1860;	+0.7468 2295;	+0.2160;	+0.6880 3954;	+0.2460;	+0.6349 9779;
+0.2760;	+0.5870 5322;	+0.3060;	+0.5435 9628;	+0.3360;	+0.5040 9198;
+0.3660;	+0.4680 7910;	+0.3960;	+0.4351 6115;	+0.4260;	+0.4049 9683;
+0.4560;	+0.3772 9154;	+0.4860;	+0.3517 9014;	+0.5160;	+0.3282 7091;
+0.5460;	+0.3065 4049;	+0.5760;	+0.2864 2958;	+0.6060;	+0.2677 8944;
+0.6360;	+0.2504 8887;	+0.6660;	+0.2344 1171;	+0.6960;	+0.2194 5472;
+0.7260;	+0.2055 2581;	+0.7560;	+0.1925 4254;	+0.7860;	+0.1804 3082;

+0.8160;	+0.1691 2388;	+0.8460;	+0.1585 6132;	+0.8760;	+0.1486 8835;
+0.9060;	+0.1394 5509;	+0.9360;	+0.1308 1604;	+0.9660;	+0.1227 2959;
+0.9960;	+0.1151 5754;	+1.0260;	+0.1080 6483;	+1.0560;	+0.1014 1912;
+1.0860;	+0.0951 9059;	+1.1160;	+0.0893 5169;	+1.1460;	+0.0838 7689;
+1.1760;	+0.0787 4251;	+1.2060;	+0.0739 2659;	+1.2360;	+0.0694 0870;
+1.2660;	+0.0651 6984;	+1.2960;	+0.0611 9230;	+1.3260;	+0.0574 5959;
+1.3560;	+0.0539 5629;	+1.3860;	+0.0506 6805;	+1.4160;	+0.0475 8142;
+1.4460;	+0.0446 8384;	+1.4760;	+0.0419 6359;	+1.5060;	+0.0394 0967;
+1.5360;	+0.0370 1179;	+1.5660;	+0.0347 6033;	+1.5960;	+0.0326 4627;
+1.6260;	+0.0306 6114;	+1.6560;	+0.0287 9704;	+1.6860;	+0.0270 4654;
+1.7160;	+0.0254 0267;	+1.7460;	+0.0238 5890;	+1.7760;	+0.0224 0911;
+1.8060;	+0.0210 4755;	+1.8360;	+0.0197 6884;	+1.8660;	+0.0185 6791;
+1.8960;	+0.0174 4002;	+1.9260;	+0.0163 8072;	+1.9560;	+0.0153 8582;
+1.9860;	+0.0144 5140;	+2.0160;	+0.0135 7377;	+2.0460;	+0.0127 4948;
+2.0760;	+0.0119 7528;	+2.1060;	+0.0112 4812;	+2.1360;	+0.0105 6514;
+2.1660;	+0.0099 2365;	+2.1960;	+0.0093 2112;		

APPENDIX 2 -REFERENCES

(1) Hartree, D.R.: Numerical Analysis. Oxford University Press.

(2) Davis, H.T.: Introduction to Nonlinear Differential and Integral Equations. Dover Publications Inc.

(3) Spenceley, G.W. and R.M.: Smithsonian Elliptic Functions Tables. Smithsonian Institution.

(4) McLauchan, N.W.: Ordinary Non-linear Differential Equations. Oxford University Press.

(5) Fox, L.: The Numerical Solution of Two-Point Boundary Problems. Oxford University Press.

APPENDIX 3 - PROGRAMMES

(1) Finite Difference

(2) Step-by-Step

(3) Perturbation

(4) Picard

INPUT PARAMETERS FOR FINITE DIFFERENCE PROGRAMME

a,b,c,d: coefficients of y'', y', y and y^2 respectively

h: interval between points

m: number of points to be calculated

t: if graph plot is required then t:=1 else 0

f: number of carriage returns required for graph plotter

```
begin library A0,A6,A12;
      procedure  GP(X,Y,n,s); value X,Y,n,s;
      integer n,s;
      real array X,Y;
      begin real sfX, sfY, XMIN,YMAX,XINC,YINC,CX1,CY1,X1,X2,LL,UL;
      integer i,j,k,m,N;
      XMIN:=X[1];
      X1:=X[1]; X2:=X[1];
      for i:=2 step 1 until n do
      begin if X[i]>X1 then X1:=X[i];
            if X[i]<X2 then X2:=X[i];
      end;
      XMIN:=X2;
      XINC:=(X1-X2)/10;
      sfX:=100/(X1-X2);
      for i:= 1 step 1 until n-1 do
      for j:=i+1 step 1 until ndo
      begin ifY[j]>Y[i] then
            begin CY1:=Y[i]; Y[i]:=Y[j]; Y[j]:=CY1;
                  CX1:=X[i]; X[i]:=X[j]; X[j]:=CX1;
            end;
      end;
      sfY:=s/(Y[1]-Y[n]);
      YMAX:=Y[1]; YINC:=(Y[1]-Y[n])/s;
      for i:=1 step 1 until n do
      begin X[i]:=entier((X[i]-XMIN)xsfX+0.5);
            Y[i]:=entier(Y[i]xsfY+0.5);
      end; i:=0;
    D:i:=i+1; if Y[i+1]=Y[i] then
      begin LL:=i; j:=i;
         if j=n-1 then begin UL:=n; goto F; end;
         E:j:=j+1; if Y[j+1]=Y[j] and j<n-1 then goto E else if Y[j+1]=Y[i]
            and j=n-1 then UL:=n else UL:=j;
         F:for k:=LL step 1 until UL-1 do
            for m:=k+1 step 1 until UL do
            begin if X[m]<X[k] then
                  begin CX1:=X[k]; X[k]:=X[m]; X[m]:=CX1;
                  end;
            end; i:=UL;
      end;  if i<n-1 then goto D;
      if X[1]>1 then space(10,X[1]-1);
      write text (10,[.]);
      N:=0; for i:=2 step 1 until n do
      begin m:=Y[i-1]-Y[i];
            if m=0 then
            begin ifX[i]-X[i-1]=0 then begin N:=N+1; goto A; end else
                  if X[i]-X[i-1]=1  then gotoB else space(10,X[i]-X[i-1]-1);
```

```
                B:write text(10,[.]);
            end else
            begin  new line(10,m); if X[i]>1 then space(10,X[i]-1);
                write text (10,[.]);
            end;
        A:end;
        new line(10,2);
        write text(10,[[c] Number*of*points*ignored*=]);
        write(10,format([-nddd;ccc]),N);
        end;

begin real h,x,e,a,b,c,d,f,g,t,v,a1,a2,a3,k1,k2,w;
        integer i,m,n,k,p,q,r,j;
        open(10); open(20); n:=read(20);
        for p:=1 step 1 until n do
        begin a:=read(20); b:=read(20); c:=read(20);d:=read(20); h:=read(20);
            m:=read(20); t:=read(20); f:=read(20);
            write text(10,[input*parameters*dy/dt,yXy*and*interval[c]]);
        write(10,format([+nddd.d;]),b); write(10,format([sss+nddd.d;]),d);
        write(10,format([sss+d.dddd;ccc]),h);
            begin real array A,B,W,Z[1:m-1],D[1:7,1:m+1],X,Y[1:m+1];
          T:Y[1]:=0; X[1]:=0;
            if b=0 then begin w:=sqrt(c/a); for i :=2 step 1 until m+1 do
                begin X[i]:=X[i-1]+h;
                Y[i]:=sin(wXX[i]);end;
            end else
            if bXb-4XaXc<0 then
            begin k1:=(2XaXsqrt(4XaXc-bXb))/b;
                k2:=exp(arctan(k1)/k1)/sin(arctan(k1));
                w:=sqrt(4XaXc-bXb)/(2Xa);  k1:=-b/(2Xa);
                for i:=2 step 1 until m+1 do
                begin X[i]:=X[i-1]+h;
                    Y[i]:=k2Xexp(k1XX[i])Xsin(wXX[i]);
                end ;
            end else
            for i:= 2 step 1 until m+1 do
            begin Y[i]:=0.1; X[i]:=X[i-1]+h;
            end;
            for i:= 1 step 1 until m-1 do
            Z[i]:=Y[i+1]XY[i+2]XhXhXd; q:=0;
            a1:=a-bXh/2; a2:=cXhXh-2Xa; a3:=a+bXh/2;
            k1:=a1/a3; k2:=a2/a3;
      P:A[1]:=Z[1]/a3; B[1]:=k2;
        A[2]:=Z[2]/a3-k2XA[1]; B[2]:=k1-k2XB[1];
        for r:=3 step 1 until m-1 do
        begin A[r]:=Z[r]/a3-k2XA[r-1]-k1XA[r-2];
            B[r]:=-(k2XB[r-1]+k1XB[r-2]);
        end;
```

```
      r:=0;
    Q:r:=r+1;j:=r;
      if abs(B[r])>⏨-4 then
      x:=(A[r]-1)/B[r] else begin h:=10×h/9 ; write text(10,[interval*altered[ccc]]); goto T;
                            end;
    R:j:=j+1;  if A[j]-B[j]×x>1 then goto Q;
      if j≠m-1 then goto R;
      e:=⏨-4; Y[1]:=0;  Y[2]:=x;
      for i:=1 step 1 until m-1 do
      begin W[i]:=A[i]-B[i]×x;
            v:=abs(W[i]-Y[i+2]);  if v>e then e:=v;
      end;
      if q=1 then begin r:=0; goto exit1; end;
      for i:=1 step 1 untl m-1 do
      begin Y[i+2]:=W[i]; Z[i]:=Y[i+1]×Y[i+1]×h×h×d;
      end;
      if abs(x)>1 then begin write text(10,[equation*unstable[ccc]]); goto exit2; end;
      if e>⏨-4 then goto P else
      for i:=1 step 1 until m+1 do D[1,i]:=Y[i];
      l:=0; k:=1;
    S:l:=l+2;
      for j:=l/2 step 1 until m-l/2+1 do
      D[l,j]:=D[l-1,j+1]-D[l-1,j];
      k:=k+2;
      for j:=(k+1)/2 step 1 until m-(k-3)/2 do
      D[k,j]:=D[k-1,j]-D[k-1,j-1];
      if k<6 then goto S;
      for i:= 4 step 1 until m-2 do
      begin D[1,i]:=(D[4,i-1]+D[4,i])/2;
            D[2,i]:=(D[6,i-1]+D[6,i])/2;
      end;
      for i:=4 step 1 until m-2 do
      Z[i]:=Z[i]+(D[1,i]+D[5,i]/2-D[2,i]/6-D[7,i]/15)/6;
      q:=1; goto P ;
 exit1:write text(10,[X*and*Y*values[c]]);
      for i:=1 step 1 until m+1 do
      begin write(10,format([6s+ndd.dddd;]),X[i]);
            write(10,format([sss+d.ddddsdddd;]),Y[i]);
            r:=r+1; if r/3=entier(r/3) then new line(10,1);
      end; new line(10,5);  if t=1 then begin for i:=1 step 1 until m+1 do X[i]:=-X[i]; GP(Y,X,m+1,f); end;
end;
exit2:
end; close(10);close(20);
end;
end→
```

INPUT PARAMETERS FOR STEP-BY-STEP PROGRAMME

a,b,c,d: coefficients of y'', y', y and y^2 respectively

h: interval between points

t: if graph plot is required then t:=1 else 0

p1,p2: initial function values from Finite Difference results

f: number of carriage returns required for graph plotter

w: initial t value corresponding to p1

```
begin real h,a,b,c,d,f,t,a1,a2,a3,k1,k2,k3,p1,p2,W;
      integer i,m,n,p,r;
      open(10); open(20); n:=read(20);
      for p:=1 step 1 until n do
      begin a:=read(20); b:=read(20); c:=read(20);d:=read(20); h:=read(20);
            t:=read(20); p1:=read(20); p2:=read(20); f:=read(20); W:=read(20);
            begin real array X,Y[1:1000];
                  Y[1]:=p1; Y[2]:=p2; r:=0;
                  a1:=a-b×h/2; a2:=c×h×h-2×a; a3:=a+b×h/2;
                  k1:=a1/a3; k2:=a2/a3; k3:=h×h×d/a3;
                  if b×b-4×a×c>0 then
                  begin for i:=3,i+1 while Y[i-1]>.01 do
                        Y[i]:=Y[i-1]×Y[i-1]×k3-Y[i-1]×k2-Y[i-2]×k1; m:=i-2;
                  end else
                  begin i:=2;
                      R:i:=i+1;
                  Y[i]:=Y[i-1]×Y[i-1]×k3-Y[i-1]×k2-Y[i-2]×k1;
                        if Y[i-1]>Y[i-2] and Y[i-1]>Y[i] then
                        begin if Y[i-1]>.01 then gotoR else goto S;
                        end else goto R;
                     S:m:=i-1;
                  end;
                  write text(10,[input*parameters*dy/dt,yxy,interval*and*limit[c]]);
                  write(10,format([+nddd.d;]),b); write(10,format([sss+nddd.d;]),d);
                  write(10,format([sss+d.dddd;]),h); write(10,format([sss+d.dddd;ccc]),m×h);
                  X[1]:=W; for i:=2 step 1 until m+1 do
                  X[i]:=X[i-1]+h;
                  write text(10,[X*and*Y*values[c]]);
                  for i:=1 step 1 until m+1 do
                 begin write(10,format([6s+ndd.dddd;]),X[i]);
                        write(10,format([sss+d.dddd;]),Y[i]);
                        r:=r+1; if r/3=entier(r/3) then new line(10,1);
                  end; new line(10,5);
                  if t=1 then
                  begin for i:=1 step 1 until m+1 do X[i]:=-X[i]; GP(Y,X,m+1,f); new line(10,5);
                  end;
            end;
      end; close(10); close(20);
end;
end→
```

INPUT PARAMETERS FOR PICARD PROGRAMME

a,b,c,d:	coefficients of y'', y', y and y^2 respectively
L1:	upper limit of integration
q:	interval between points
l:	degree of appoximation required
sp:	the ammount by which the point density after the maximum has to be decreased i.e. if every 10th point is required then sp=10

```
begin library A0,A6,A12;
    procedure AITKENROOT(r,x,y,eps);
              value eps;
              real array x,y;
              real eps;
              integer r;
              begin integer i;
                    x[0]:= 1;
                    if x[r]=0 then x[0]:= 0
                              else if r=1 then begin y[2]:= y[1]+y[0];
                                                     r:= 2
                                               end
                              else if r=2 and sign(x[2])= sign(x[1]) then begin x[1]:= x[2];
                                                                                y[1]:= y[2];
                                                                                y[2]:= y[1]+y[0]
                                                                          end
                               else begin for i:= 1 step 1 until r-1 do
                                          y[r]:= (y[i]×x[r]-y[r]×x[i])/(x[r]-x[i]);
                                          if abs(y[r]-y[r-1])<eps then x[0]:= 0
                                                                  else begin y[r+1]:= y[r];
                                                                             r:= r+1
                                                                       end
                                    end;
              end AITKENROOT;

procedure CONST(const,f,g,n,a,b,c,d,A,B); value f,g,n,a,b,c,d;
     real const,f,g,a,b,c,d; integer n; real array A,B;
     begin integer i,j,p,k,t; real sum;
           for i:=2 step 1 until n do
           begin sum:=0; t:=2↑(entier(ln(i)/ln(2)-0.0000001));
                 for j:=1 step 1 until i-1 do
                 begin ifi-j<t and j<t then
                       sum:=sum+A[j,T]×A[i-j,1];
                 end; B[i,1]:=sum;
           end;
           for k:=2 step 1 until n do
           begin t:=2↑(entier(ln(k)/ln(2)-0.0000001));
                 for i:=2 step 2 until k+1 do
                 begin sum:=0;
                       for j:=1 step 1 until i/2 do
                       for p:=1 step 1 until k-i+j do
                       begin if k-p<tand p<t then
                             sum:=sum+A[k-p,i-j+1]×A[p,j]×2;
                       end; B[k,i]:=sum;
                 end;
```

```
                    for i:=3 step 2 until k+1 do
                    begin sum:=0;
                          for j:=1 step 1 until (i-1)/2 do
                          for p:=1 step 1 until k-i+j do
                          begin if k-p<t and p<t then
                                sum:=sum+A[k-p,i-j+1]×A[p,j]×2;
                          end;
                          for p:=(i-1)/2 step 1 untilk-(i-1)/2 do
                          begin if k-p<t and p<t then
                                sum:=sum+A[p,(i+1)/2]×A[k-p,(i+1)/2];
                    end; B[k,i]:=sum;
              end;
              end; const:=0;
              for i:=2 step 1 until n do
              for j:=1 step 1 until i+1 do
              begin A[i,j]:=d×B[i,j]/(a×((i-j+1)×f+(j-1)×g)↑2-b×((i-j+1)×f+(j-1)×g)+c);
                    const:=const+A[i,j];
              end;
        end;

begin real a,b,c,d,f,g,q,L1,YM,AO,M,eps,const,k;
      integer m,n,p,r,s,N,l,u,i,j,w,sp,inc,h;
      open(10); open(20); p:=read(20);
      for r:=1 step 1 until p do
      begin a:=read(20); b:=read(20); c:=read(20); d:=read(20);
      L1:=read(20); q:=read(20); l:=read(20);sp:=read(20);
      write text(10,[successive*number*of*iterations]);
    P:h:=b×b-4×a×c; n:=entier(L1/q); s:=2000; N:=0;
      f:=(b-sqrt(h))/(2×a); g:=(b+sqrt(h))/(2×a);
      k:=f/g; AO:=1/(k↑(k/(1-k))-k↑(1/(1-k))); eps:=₁₀-5; m:=2↑l;
      begin real array A,B[1:m,1:m+1],X,Y[1:2000],x,y[0:30];
      A[1,1]:=AO; A[1,2]:=-AO;
      X[1]:=0; for h:=2 step 1 until s do X[h]:=X[h-1]+q;
      for inc:=1 step 1 until l do
      begin w:=1; u:=2; y[1]:=A[1,1]; M:=2↑inc;
          R:CONST(const,f,g,M,a,b,c,d,A,B);
          Q:A[1,2]:=-A[1,1]-const; N:=N+1;
            for h:=1 step 1 until 3 do
            begin Y[h]:=0;
                  for i:=1 step 1 until M do
                  for j:=1 step 1 until i+1 do
                  Y[h]:=Y[h]+A[i,j]×exp(-((i-j+1)×f+(j-1)×g)×X[h]);
            end;
            for h:=3,h+1 while Y[h-2]<Y[h-1] and Y[h]>Y[h-1] do
```

```
          begin Y[h+1]:=0;
                for i:=1 step 1 until M do
                for j:=1 step 1 until i+1 do
                Y[h+1]:=Y[h+1]+A[i,j]×exp(-((i-j+1)×f+(j-1)×g)×X[h+1]);
          end; YM:=Y[h-1]; s:=h;
          x[w]:=YM-1;
          if u=2 then begin if YM-1>0 then y[0]:=-A[1,1]/(5×inc) else y[0]:=A[1,1]/(5×inc); u:=1;
                      end;
          AITKENROOT(w,x,y,eps); A[1,1]:=y[w];
          if x[0]≠0 then goto Q;
          write(10,format([sss-ndd;]),N-1);
    end;
    for h:=1 step 1 until n+1 do Y[h]:=0;
    for h:=1 step 1 until n+1 do
    begin for i:=1 step 1 untilMdo
          for j:=1 step 1 until i+1 do
          Y[h]:=Y[h]+A[i,j]×exp(-((i-j+1)×f+(j-1)×g)×X[h]);

    end; new line(10,3);
    write text(10,[input*parameters*dy/dt,yxy,upper*limit,interval*and*approximation[c]]);
    write(10,format([+nddd.d;]),b); write(10,format([sss+nddd.d;]),d);
    write(10,format([sss+d.dddd;]),L1); write(10,format([sss+d.dddd;]),q);
     write(10,format([sss-ndd;ccc]),l); write text(10,[coefficients[c]]);
   fori:=1 step 1 until 2↑l do
   for j:=1step 1 until i+1 do
    begin write(10,format([sss+d.dddd₁₀+nd;]),A[i,j]);
          if j=i+1 then new line(10,1);
    end; new line(10,3);
    write text(10,[X*and*Y*values[c]]);
    for m :=1 step 1 until s+9 do
    begin write(10,format([6s+d.dddd;]),X[m]);
          write(10,format([3s+d.dddds dddd;]),Y[m]);
          if m/3=entier(m/3) then new line(10,1);
    end;
    new line(10,2); h:=0;
    for m:=s+10 step sp until n+1 do
    begin write(10,format([6s+d.dddd;]),X[m]);
          write(10,format([3s+d.dddds dddd;]),Y[m]);
          h:=h+1; if h/3=entier(h/3) then new line(10,1);
    end; new line(10,3);
    end; end; close(10); close(20);
end;
end→
```

INPUT PARAMETERS FOR PERTURBATION PROGRAMME

a,b,c,d: coefficients of y'', y', y and y^2 respectively

L1: upper limit of integration

q: interval between points

sp: the ammount by which the point density after the maximum has to be decreased i.e. if every 10th point is required then sp=10

```
begin library A0,A6,A12;
      procedure CONST(a,b,c,d,f,g,A0,Z); value a,b,c,d,f,g,A0;
      real a,b,c,d,f,g,A0; real array Z;
      begin Z[1]:=A0;
            Z[5]:=A0↑2/(4×a×f↑2-2×b×f+c);
            Z[9]:=A0↑2/(4×a×g↑2-2×b×g+c);
            Z[11]:=-2×A0↑2/(a×(f+g)↑2-b×(f+g)+c);
            Z[2]:=(g×(Z[5]+Z[9]+Z[11])-2×f×Z[5]-2×g×Z[9]-(f+g)×Z[11])/(f-g);
            Z[6]:=(f×(Z[5]+Z[9]+Z[11])-2×f×Z[5]-2×g×Z[9]-(f+g)×Z[11])/(g-f);
            Z[3]:=2×A0×Z[2]/(4×a×f↑2-2×b×f+c);
            Z[7]:=-2×A0×Z[6]/(4×a×g↑2-2×b×g+c);
            Z[10]:=2×A0×(Z[6]-Z[2])/(a×(f+g)↑2-b×(f+g)+c);
            Z[12]:=2×A0×Z[5]/(9×a×f↑2-3×b×f+c);
            Z[13]:=-2×A0×Z[9]/(9×a×g↑2-3×b×g+c);
            Z[14]:=2×A0×(Z[9]-Z[11])/(a×(f+2×g)↑2-b×(f+2×g)+c);
            Z[15]:=2×A0×(Z[11]-Z[5])/(a×(2×f+g)↑2-b×(2×f+g)+c);
            Z[4]:=(g×(Z[3]+Z[7]+Z[10]+Z[12]+Z[13]+Z[14]+Z[15])-2×f×Z[3]-2×g×Z[7]
                  -(f+g)×Z[10]-3×f×Z[12]-3×Z[13]×g-(f+2×g)×Z[14]-(2×f+g)×Z[15])/(f-g);
            Z[8]:=(f×(Z[3]+Z[7]+Z[12]+Z[10]+Z[13]+Z[14]+Z[15])-2×f×Z[3]-2×g×Z[7]
                  -(f+g)×Z[10]-3×f×Z[12]-3×g×Z[13]-(f+2×g)×Z[14]-(2×f+g)×Z[15])/(g-f);
            Z[16]:=Z[1]+d×Z[2]+d×d×Z[4];
            Z[17]:=-Z[1]+d×Z[6]+d×d×Z[8];
            Z[18]:=d×(Z[5]+d×Z[3]);
            Z[19]:=d×(Z[9]+d×Z[7]);
            Z[20]:=d×(Z[11]+d×Z[10]);
            Z[21]:=d×d×Z[12];
            Z[22]:=d×d×Z[13];
            Z[23]:=d×d×Z[14];
            Z[24]:=d×d×Z[15];
      end;

begin real a,b,c,d,f,g,j,k,q,diff,L1,YM,A0;
      integer n,sp,p,r,s,t,h,N;
      open(10); open(20); p:=read(20);
      for r:=1 step 1 until p do
      begin a:=read(20); b:=read(20); c:=read(20); d:=read(20);
            L1:=read(20); q:=read(20); sp:=read(20);
      k:=b×b-4×a×c; n:=entier(L1/q); s:=n+1; N:=-1; t:=0;
      begin real array Z[1:25],X,Y[1:2000];
      f:=(b-sqrt(k))/(2×a); g:=(b+sqrt(k))/(2×a);
      k:=f/g; A0:=1/(k↑(k/(1-k))-k↑(1/(1-k))); Z[25]:=A0×(g-f);
      X[1]:=0; for h:=2 step 1 until s do X[h]:=X[h-1]+q;
    R:CONST(a,b,c,d,f,g,A0,Z); N:=N+1;
```

```
        for h:=1 step 1 until 3 do
        begin j:=X[h];
              Y[h]:=Z[16]×exp(-f×j)+Z[17]×exp(-g×j)+Z[18]×exp(-2×f×j)
                        +Z[19]×exp(-2×g×j)+Z[20]×exp(-(f+g)×j)+Z[21]×exp(-3×f×j)
                        +Z[22]×exp(-3×g×j)+Z[23]×exp(-(f+2×g)×j)+Z[24]×exp(-(2×f+g)×j);
        end;
        for h:=3,h+1 while Y[h-2]<Y[h-1] and Y[h]>Y[h-1] do
        begin j:=X[h+1];
              Y[h+1]:=Z[16]×exp(-f×j)+Z[17]×exp(-g×j)+Z[18]×exp(-2×f×j)
                        +Z[19]×exp(-2×g×j)+Z[20]×exp(-(f+g)×j)+Z[21]×exp(-3×f×j)
                        +Z[22]×exp(-3×g×j)+Z[23]×exp(-(f+2×g)×j)+Z[24]×exp(-(2×f+g)×j);
        end;
        YM:=Y[h-1]; s:=h;
        diff:=YM-1;
        Z[25]:=Z[25]×(1-diff);
        A0:=Z[25]/(g-f);
        if abs(diff)>₁₀-5 then goto R;
        for h:=1 step 1 until n+1 do
        begin j:=X[h];
              Y[h]:=Z[16]×exp(-f×j)+Z[17]×exp(-g×j)+Z[18]×exp(-2×f×j)
                        +Z[19]×exp(-2×g×j)+Z[20]×exp(-(f+g)×j)+Z[21]×exp(-3×f×j)
                        +Z[22]×exp(-3×g×j)+Z[23]×exp(-(f+2×g)×j)+Z[24]×exp(-(2×f+g)×j);
        end;
        write text(10,[input*parameters*dy/dt,y×y,upper*limit,interval*and*iterations[c]]);
        write(10,format([+nddd.d;]),b); write(10,format([sss+nddd.d;]),d);
        write(10,format([sss+d.dddd;]),L1); write(10,format([sssd.dddd;]),q);
        write(10,format([sssndd;ccc]),N); write text(10,[coefficients[c]]);
        for h:=16 step 1 until24 do
        begin write(10,format([sss+d.dddd₁₀+nd;]),Z[h]);
              if h/2≠entier(h/2) then new line(10,1);
        end; new line(10,3);
        write text(10,[X*and*Y*values[c]]);
        for h :=1 step 1 until s+9 do
        begin write(10,format([6s+d.dddd;]),X[h]);
              write(10,format([3s+d.ddddsdddd;]),Y[h]);
              if h/3=entier(h/3) then new line(10,1);
        end; new line(10,2); t:=0;
        for h:=s+10 step sp until n+1 do
        begin write(10,format([6s+d.dddd;]),X[h]);
              write(10,format([3s+d.ddddsdddd;]),Y[h]);
              t:=t+1; if t/3=entier(t/3) then new line(10,1);
        end; new line(10,6);
        end;
        end; close(10); close(20);
end;
end→
```