On Stability and Contractivity Properties of Semi-Implicit Runge-Kutta Methods

Raymond Legault

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada

March 1987

ISBN 0-315-35508-5

ABSTRACT

On Stability and Contractivity Properties of Semi-Implicit Runge-Kutta Methods

Raymond Legault

Error constants and γ-intervals of stability are computed for Restricted Denominator approximations to e^z of order $p=s$ and $p=(s-1)$, for methods with s stages. Results for $p=(s-1)$ are new.

Stability functions corresponding to improved approximations obtained via Richardson extrapolation and modified Richardson extrapolation are also studied numerically and similar stability intervals presented. Furthermore, an efficient approach is used to find in all these cases the maximum angle α for which we have $A(\alpha)$-stability. To our knowledge, this collected information is also new and should prove very useful in the choice of γ, the stability parameter, for new methods.

Finally, we investigate the contractivity properties of semi-implicit Runge-Kutta methods. It is found that many methods can be contractive for *any* stepsize, provided that $|\frac{L}{\mu}|$ is sufficiently small. A program was developed which numerically estimates the maximum ratio allowed for this and also computes the maximum stepsize for contractivity, if $|\frac{L}{\mu}|$ is larger.

ACKNOWLEDGEMENTS

I would like to acknowledge the assistance of Dr. T. D. Bui, his guidance of the research project and reviewing of this final report. I am also grateful for his financial assistance and the opportunity to participate in the ODE conference in Albuquerque, New Mexico (1986).

My sincere thanks to Dr. P. Kaps (Innsbruck) for his suggestions concerning the topic of this research and many clarifying discussions; also to Nguyen Long, for his careful work in plotting the graphs of the appendices.

This project was supported in part by scholarships from the Natural Sciences and Engineering Research Council of Canada, Concordia University and CRIM (Centre de Recherche Informatique de Montréal).

Last but not least, I would like to thank Joäne, Etienne and Annick for their patience, understanding and support throughout this work: we should now make up for some of the lost time!

CONTENTS

CHAPTER 1

INTRODUCTION

The mathematical modelling of numerous physical, chemical and biological processes in engineering and the natural sciences often requires the solution of systems of first order* ordinary differential equations (o.d.e.) of the form

(1.1) $$\mathbf{y}'(x) = \mathbf{f}(x,\mathbf{y}(x)), \qquad \mathbf{y}(x_0) = \mathbf{y}_0,$$

over some specified interval $[x_0,x_f]$.

The number of equations in the system simply corresponds to the dimensionality of the real vector $\mathbf{y}(x)$ and the real-valued vector function $\mathbf{f}$. Because the conditions fixing the solution are all given at a single point ($\mathbf{y}(x_0) = \mathbf{y}_0$), we speak of an *initial value problem*, as opposed to boundary value problems, where conditions are given for many different points.

* Note that a single ordinary differential equation of order m, written in the form $y^{(m)} = f(x, y^{(0)}, y^{(1)}, \dots y^{(m-1)})$, can easily be converted into a first order system in the form (1.1) above.

The present thesis is concerned with stability and contractivity properties of certain numerical methods for solving stiff problems of the type just outlined.

In this chapter, we introduce some key concepts (including stability and stiffness) for the study of numerical solutions to (1.1). The second chapter then presents the particular methods of interest in this study, along with their stability functions and error estimation techniques suitable for them. Chapters 3 and 4 deal with the accuracy and *linear* stability properties of Rosenbrock and semi-implicit Runge-Kutta methods, as a function of the so-called stability parameter.

Chapter 5 introduces many recently developed concepts to handle *nonlinear* stiff problems. Finally, chapter 6 examines the contractivity of semi-implicit methods when applied to these problems.

1.1 Discrete variable methods

It is probably safe to say that most differential equations cannot be solved analytically, in closed form. Thus scientists and mathematicians have worked on obtaining approximate solutions for the above problem and a great variety of numerical methods were developed (and still are!). For recent surveys, see Bui, Oppenheim, Pratt [8] and Gupta, Sacks-Davis, Tischer [27].

Generally, the methods for approximating the solutions of ordinary differential equations are based on the principle of *discretization*. In this case, an approximate solution to $\mathbf{y}(x)$ is sought only for a set of discrete points x_0, x_1, x_2, ... The spacing, $h = x_{n+1} - x_n$, is called the *stepsize* and may vary as the numerical integration proceeds. We denote by y_0, y_1, y_2, ... the approximations computed for $y(x_0)$, $y(x_1)$, $y(x_2)$, ... If a method only requires y_{n-1} to generate y_n, it is called a *one-step method*; if it requires older approximations (y_{n-2}, y_{n-3}, ...), it is a *multistep method*.

For a good introductory reference to numerical solutions of ordinary differential equations, see Lambert [36].

1.2 Existence of solutions; convergence of numerical methods

For the scalar case, i.e. a *single* equation $y' = f(x,y)$, $y(x_0) = y_0$, we have the following theorem:

Theorem 1.2.1 Consider the initial value problem $y' = f(x,y)$, $y(x_0) = y_0$. Let $f(x,y)$ be defined and continuous for all (x,y) in a region D such that $x_0 \leq x \leq x_f$ and $-\infty < y < \infty$; and let there exist a constant L such that for every (x,y) and $(x,\tilde{y})$ in the above domain we have

(1.2.1) $$|f(x,y) - f(x,\tilde{y})| \leq L|y - \tilde{y}|.$$

Then, for any given number y_0, *there exists a unique solution* $y(x)$ which is continuous and differentiable for all (x,y) in the specified domain.

This existence and uniqueness theorem is proven in Henrici [30], pages 15-25. We will just outline here the method followed.

First of all, we define a uniform spacing $h_p = (x_f - x_0)/2^p$ for successive values of $p = 0, 1, \ldots$ and, for each p, we consider only the points $x_n = x_0 + nh_p$, for $n = 0, 1, \ldots 2^p$. We compute points (x_n, y_n) using Euler's (explicit) method:

$$y_{n+1} = y_n + hf(x_n, y_n). \tag{1.2.2}$$

We join the points with line segments, thus creating a piecewise linear approximate solution $y_p(x)$ for which we can obtain an analytical form as follows:

$$y_p(x_0) = y_0$$

and
$$y_p(x) = y_p(x_{[p]}) + (x - x_{[p]})\, f(x_{[p]}, y_p(x_{[p]})),$$

where $x \in (x_0, x_f]$ and $x_{[p]}$ is defined as $x_0 + ih_p$ for the integer value i such that $x_0 + ih_p < x \le x_0 + (i+1)h_p$.

The proof is then divided in 3 parts:

(a) proving that as $p \to \infty$, $y_p(x)$ converges uniformly on $[a,b]$ to a *continuous* function $y(x)$;

(b) proving that this limit $y(x)$ is a *solution* of the differential equation;

(c) proving that it is the *only* solution.

Condition (1.2.1) is called a *Lipschitz condition* and L is called a *Lipschitz constant.*

The above theorem can be generalized to *systems* as follows. We simply substitute the domain D, defined by $x \in [x_0, x_f]$ and $-\infty < y_i < \infty$, for $i = 1, 2, \ldots m$ (assuming we have m equations). Condition (1.2.1) is replaced by

$$\|\mathbf{f}(x,\mathbf{y}) - \mathbf{f}(x,\tilde{\mathbf{y}})\| \le L\|\mathbf{y} - \tilde{\mathbf{y}}\|. \tag{1.2.3}$$

The proof may be found in Henrici [30].

In the case where each component of the vector function $\mathbf{f}(x,y)$ possesses a continuous derivative with respect to each of the components of the vector $\mathbf{y}$, then we may choose

(1.2.4) $$L = \sup \|\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\|, \text{ over all } (x,y) \in D,$$

where $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ is the Jacobian matrix J with components $J_{ij} = \frac{\partial f_i}{\partial y_j}$, and the matrix norm is subordinate to the vector norm used in (1.2.3).

Definition 1.2.2 A numerical integration method based on discretization is *convergent* if for all initial value problems (1.1) satisfying condition (1.2.3), its approximate solutions $\mathbf{y}_n$ for any given point x in $[x_0,x_f]$ satisfy

(1.2.5) $$\lim_{\substack{h \to 0 \\ nh \to x-x_0}} \|\mathbf{y}_n - \mathbf{y}(x)\| \to 0.$$

Clearly, the approach indicated above to prove theorem 1.2.1 proves by the same token that Euler's explicit method is convergent. Henrici [30] also proves some general convergence results applicable to one-step methods.

Euler's (explicit) rule is very simple. It is a one-step method which is linear in both $\mathbf{y}_n$ and $\mathbf{f}_n$. The Taylor series expansion of the exact solution $\mathbf{y}(x_{n+1})$ about x_n and Euler's rule for the numerical solution will agree only up to the first power of the stepsize h. Thus this method is said to be of order 1. Higher order can be achieved by sacrificing either its one-step or its linear character. In the first case, we have for example the well-known *linear multistep methods*, whereas in the second case we have general one-step methods, of which the most well-known are *Runge-Kutta methods*, presented in chapter 2.

1.3 Errors in numerical integration methods

While convergence results have their theoretical and practical importance, we do not use infinitely small stepsizes in practise and this introduces errors in the numerically computed solutions.

In fact, we can distinguish three types of errors which will contaminate results to varying degrees. First there is the *starting error* if the various parameters for the method used and the initial values of the solution y_0 are not exactly representable in the computer. Secondly, throughout the computation of the solution, the limited accuracy of the machine will cause some *rounding errors*. Finally, the limited *order* of the method used introduces a *truncation error*, which is generally the most important error factor.

We now proceed to define the truncation error for the case of one-step methods like the ones we will later investigate. The *general one-step method* can be written in the following form:

(1.3.1) $$y_{n+1} = y_n + h\Phi(x_n, y_n, y_{n+1}, h).$$

Definition 1.3.1 The *local truncation error* at x_{n+1} of the one-step method is defined as

(1.3.2) $$T_{n+1} = y(x_{n+1}) - \hat{y}_{n+1}, \text{ where}$$

(1.3.3) $$\hat{y}_{n+1} = y(x_n) + h\Phi(x_n, y(x_n), \hat{y}_{n+1}, h).$$

Note that (1.3.3) computes the approximate solution vector at x_{n+1}, using $y(x_n)$ instead of y_n. It is in this sense that T_{n+1} is *local.* It is the error made at step n+1, assuming no errors were made before. Dropping this assumption, $E_{n+1} = y(x_{n+1}) - y_{n+1}$, where y_{n+1} is obtained with (1.3.1), is called the *global truncation error.* Here the difference will depend not only on error due to the current step, but also on the accumulation of truncation errors at all previous steps.

Definition 1.3.2 The one-step method (1.3.1) is of *order* p, if p is the largest integer for which we have

(1.3.4) $$T_{n+1} = O(h^{p+1}).$$

Obviously, the higher the order, the more accurate the method will be.

Clearly, the accuracy of a method for numerical integration depends on the build-up of truncation errors and to a lesser extent of rounding errors. In order to maintain a certain accuracy, we would like to be able to monitor this build-up and take appropriate actions when necessary. Most of the time however, one only monitors the local truncation error. If the method is of order p, we have

(1.3.5) $$T_{n+1} = \Psi(x_n,y(x_n))h^{p+1} + O(h^{p+2}),$$

where the first term is called *principal local truncation error* and $\Psi(x,y)$ is called the *principal error function.*

Typically, methods will obtain *estimates* for the principal local truncation error and the stepsize will be dynamically adjusted as integration proceeds, to maintain some user-specified tolerance (absolute or relative) on the local error. Thus, when the function Ψ has larger values, the stepsize needs to be reduced; but if the estimated error becomes significantly smaller than the prescribed level, the stepsize should be increased, to cut down the computation time and speed up the integration. We will later consider some strategies for estimating the principal local truncation error (see chapter 2).

1.4 Linear first-order systems and linear stability

Definition 1.4.1 The first-order system (1.1), where **y** and **f** are m-dimensional vectors is said to be *linear* if $\mathbf{f}(x,\mathbf{y}) = A(x)\mathbf{y} + \mathbf{g}(x)$, where $A(x)$ is an m by m matrix and $\mathbf{g}(x)$ is an m-dimensional vector. In addition, if $A(x) = A$, a constant matrix, then the system is called *linear with constant coefficients.*

For the time being, we will restrict our attention to linear systems with constant coefficients:

$$(1.4.1) \qquad \mathbf{y}' = A\mathbf{y} + \mathbf{g}(x), \quad \mathbf{y}(x_0) = \mathbf{y}_0.$$

If the matrix A has m *distinct* eigenvalues λ_i, $i = 1, 2, \ldots m$, with corresponding eigenvectors $\mathbf{C}_i$ and if $\mathbf{q}(x)$ is a *particular* solution of $\mathbf{y}' = A\mathbf{y} + \mathbf{g}(x)$, then we can write the *general* solution of the linear problem with constant coefficients as

$$(1.4.2) \qquad \mathbf{y}(x) = \sum_{i=1}^{m} \kappa_i \exp(\lambda_i(x - x_0))\mathbf{C}_i + \mathbf{q}(x),$$

where the κ_i are obtained by expanding the vector $\mathbf{y}_0 - \mathbf{q}(x_0)$ along the fundamental system formed by the eigenvectors $\mathbf{C}_i$ i.e.

$$(1.4.3) \qquad \mathbf{y}_0 - \mathbf{q}(x_0) = \sum_{i=1}^{m} \kappa_i \mathbf{C}_i.$$

The proof is easy and can be found in Lambert [30], pages 5-6.

Next we examine the problem of *stability* of numerical methods when applied to problems like (1.4.1). To illustrate this concept, we will examine the application of 2 elementary methods, Euler's explicit and implicit rules, on the very simple *scalar test equation*

(1.4.4) $$y' = \lambda y, \quad y(x_0) = y_0 \quad (\lambda \text{ real and negative}).$$

Euler's explicit rule was already given as (1.2.2) and its implicit counterpart (also called *backward* Euler) is just

(1.4.5) $$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}).$$

The exact solution is given by $y(x) = y_0 \exp(\lambda(x - x_0))$ and it clearly tends to zero as x tends to infinity. It is only natural to require that this also be the case for the numerical solutions. For the explicit scheme, we obtain

(1.4.6) $$y_n = (1 + h\lambda)^n \, y_0,$$

whereas for the implicit case we have

(1.4.7) $$y_n = \left(\frac{1}{1 - h\lambda}\right)^n y_0.$$

The stability requirement for Euler's explicit rule then becomes

(1.4.8) $$h\lambda \in (-2,0),$$

whereas for backward Euler there is no similar restriction on the stepsize. In fact, any $h\lambda \in (-\infty,0)$ will do fine. This is actually a very significant difference between explicit and implicit Runge-Kutta methods, of which the above methods are the simplest examples.

With the simple test equation which we just considered, we can observe that $\lambda = \dfrac{\partial f}{\partial y}$ and that we have obtained intervals of stability on the negative real axis. If we now consider the system (1.4.1), $\dfrac{\partial f}{\partial y}$

is now the matrix A, whose eigenvalues may be complex and we must now consider regions of stability in the complex plane, not simply intervals on the real axis.

Definition 1.4.2 Consider the application of a numerical integration method (for example linear multistep or general one-step) with fixed positive stepsize h to the scalar test equation

$$y' = \lambda y, \quad y(x_0) = y_0 \qquad (\lambda \in \mathbb{C}). \tag{1.4.9}$$

We define the *absolute stability region* to be that region of the complex $h\lambda$-plane for which all resulting solutions will tend to 0 as n tends to infinity.

Note that the equivalent requirement $|y_{n+1}| < |y_n|$ could have been used in the above definition.

Now it can be shown (see section 2.6 for illustration of this) that for problems of type (1.4.1), if the eigenvalues λ_i of the constant matrix A are such that all $h\lambda_i$ lie in the stability region of a certain method, then this method is stable for this problem. Thus the usefulness of stability regions found with the simple scalar test equation (1.4.9) carries over to linear systems. And we see that methods with relatively small regions of stability will impose corresponding restrictions on the stepsize.

The linear stability requirement that we make, namely that numerical solutions of simple test equations should tend to 0 when the exact solutions do, can be understood in an alternative way. In fact, this is equivalent to requiring that the global truncation error be damped out as the numerical integration proceeds (i.e. for large n, we require that error → 0). Thus it is a requirement that there be no out-of-control accumulation of local truncation errors.

The importance of meeting the stability requirements, if we are to have any trust in the computed solutions, is clear from the explicit Euler rule example: if we use a stepsize which is too large (i.e. such that $h\lambda<-2$), the solution (1.4.6) will diverge for large n instead of converging to 0 as the exact solution does. A method which cannot even solve a simple problem like (1.4.4) adequately should surely not be trusted for more complex problems!

We conclude this section by defining various stability criteria which we will need in the rest of our study.

Definition 1.4.3 (Dahlquist [22]) A numerical method is *A-stable* if its absolute stability region contains the whole of the left-hand half-plane $\mathrm{Re}(h\lambda) < 0$.

Definition 1.4.4 (Widlund [58]) A numerical method is *A(α)-stable*, $\alpha \in (0,\frac{\pi}{2})$, if its absolute stability region contains the infinite wedge $W_\alpha = \{h\lambda \text{ such that } |\arg(-h\lambda)| < \alpha\}$. A method is *A(0)-stable* if it is A(α)-stable for some infinitesimal $\alpha > 0$.

Clearly, $A(\frac{\pi}{2})$-stability is the same as A-stability.

Definition 1.4.5 (Ehle [26]) A one-step numerical method is *L-stable* if it is A-stable and in addition, when applied to the scalar test equation (1.4.9), it yields $y_{n+1} = R(h\lambda)y_n$, where $|R(h\lambda)| \to 0$ as $\mathrm{Re}(h\lambda) \to -\infty$.

1.5 Stiffness in the linear case

A simple scalar example will first be used to illustrate the phenomenom of stiffness. It is a special case of the equation

(1.5.1) $$y' = \lambda(y(x) - F(x)) + F'(x), \qquad y(0) = y_0,$$

which is used by Shampine and Gear [48] to discuss stiffness and for which the exact solution is given by

(1.5.2) $$y(x) = (y_0 - F(0))e^{\lambda x} + F(x).$$

For our present purposes, we will consider the particular case $y_0=10$, $\lambda=-10$ and $F(x)=x$, which yields

(1.5.3) $$y' = f(x,y) = 1 + 10(x - y).$$

The exact solution is simply $y(x) = x + 10e^{-10x}$ and is tabulated to six decimal place accuracy for some values of x in the range [0.0,5.5] (see table 1.5).

Clearly, for $x \geq 1.0$, the solution of our differential equation is hardly distinguishable from the straight line $y=x$. But if we start integrating with Euler's (explicit) rule, from $x=1.0$ onward and using a stepsize of $h=0.5$, we quickly run into very serious instability. This occurs even if we have used the exact solution to six decimal places as our starting value. Why is this?

The *accuracy* of Euler's rule cannot be called into question here. For $x \geq 1.0$, the solution is basically the straight line $y=x$, for which Euler's rule is **exact**, for any stepsize!

x	y(x)	Euler (h = 0.5)	Backward Euler (h = 0.5)
0.0	10.000000	---	---
0.1	3.778795	---	---
0.2	1.553353	---	---
0.3	0.797871	---	---
0.4	0.583156	---	---
0.5	0.567379	---	---
1.0	1.000454	1.000454	1.000454
1.5	1.500003	1.498184	1.500076
2.0	2.000000	2.007264	2.000013
2.5	2.500000	2.470944	2.500002
3.0	3.000000	3.116224	3.000000
3.5	3.500000	3.035104	3.500000
4.0	4.000000	5.859584	4.000000
4.5	4.500000	-2.938336	4.500000
5.0	5.000000	34.753344	5.000000
5.5	5.500000	-113.513380	5.500000

Table 1.5: Instability of Euler's explicit rule

As we have seen in the previous section, stability considerations (i.e. ensuring that local errors are not magnified in subsequent steps but rather dampened) can also impose restrictions on the stepsize. For our problem, $\frac{\partial f}{\partial y}=-10$. Thus the stability condition (1.4.8) would require $h<0.2$. What is surprising here is that it seems that the value of λ imposes restrictions on the stepsize, even beyond $x=1.0$ *when it no longer plays any significant role* in the solution we are trying to approximate.

For this simple case, we can readily identify how such error magnification occurs. The global truncation error, $E_n = y(x_n) - y_n$, obeys the following simple recurrence relation:

(1.5.4) $$E_{n+1} = (1 + h\lambda)E_n - [y(x_n) + hy'(x_n) - y(x_{n+1})].$$

Clearly, if $|1 + h\lambda|>1$, any local truncation error will be magnified in subsequent steps and the error build-up will be quite rapid.

This elementary example has nonetheless pointed to the essential character of stiffness: *for most problems, it is the accuracy of the method which dictates the stepsize; but for stiff problems it is stability considerations which take over.* Our example is only very mildly stiff. We normally speak of stiffness when λ is very large negatively and the corresponding reduction of the stepsize can then be very dramatic.

It is important to point out that the problem is not called stiff in the relatively short initial interval before the transient solution component $(y_0 - F(0))e^{\lambda x}$ has died out. Here we do expect the need for a small stepsize, for accuracy reasons, because the solution varies rapidly. The problem is only called stiff *after* the transient has practically disappeared from the solution, but still dictates the stepsize for stability reasons.

Following Dekker and Verwer [24], we now define stiffness more precisely for the linear problem (1.4.1):

Definition 1.5.1 Problem (1.4.1), where A is an m by m matrix with eigenvalues $\lambda_1, \lambda_2, \dots \lambda_m$ is called *stiff* if the following conditions are satisfied:

1. λ_i exist with $\mathrm{Re}(\lambda_i) << 0$;
2. λ_i exist of moderate size, i.e. "small" when compared to the modulus of the ones in the first condition;
3. no λ_i exist with "large" positive real part;
4. no λ_i exist with a "large" imaginary part, unless we also have $\mathrm{Re}(\lambda_i) << 0$.

As explained before, the problem is called stiff only after the fastest decaying terms of the solution have died out. Also it is assumed that g(x) in (1.4.1) is as smooth as the slowly varying exponentials in the solution.

Now we recall that under certain conditions we can take $L = \|\frac{\partial f}{\partial y}\|$ (see 1.2.4). Since the norm of a matrix is never smaller than its spectral radius (i.e. the largest modulus of any of its eigenvalues) and since stiff problems would have some eigenvalues with very large modulus, we can conclude that stiff problems are characterized as well by *very large Lipschitz constants.*

While the typical scientific problem is non-stiff, we cannot downplay the importance of stiff problems and the need to deal with them adequately. Indeed stiff problems arise when we model chemical reactions with large rate constants; nuclear reactions with species decaying at rates varying widely; electric circuitry involving fast elements etc... For a detailed presentation of the many areas in which stiff problems are of interest, see Aiken [3], chapter 2.

As seen above, classical schemes (explicit ones in particular) are not adequate to deal with stiff problems: extreme restrictions on the stepsize would then result in a tremendous computing cost. And even if we could afford this cost, we would then have to worry about the accumulation of rounding errors also since we would be using such a large number of steps.

Methods to deal appropriately with stiffness would have to be ones with very large stability regions. Obviously, A-stable and L-stable methods would be prime candidates. But in a situation where all large eigenvalues are real, we might be satisfied with using simply some A(0)-stable methods; in the same manner, if all eigenvalues of interest are contained in a wedge W_α as previously defined, then A(α)-stability would be suitable.

Dahlquist [22] has proven the following results which show the shortcomings of linear multistep methods for dealing with stiffness:

1. An explicit linear k-step method cannot be A-stable
2. The order p of an A-stable (thus implicit) linear multistep method cannot exceed 2.

In general, implicit methods have much better stability properties than their explicit counterparts; this is exemplified by the results of Euler's backward rule shown in table 1.5.1, which handled well this simple problem with the same stepsize h=0.5. Here we observe the desired damping of the global truncation error as numerical integration proceeds.

In the next chapter, we will turn to Runge-Kutta methods, of which the Euler rules are the simplest cases. We will show that implicit methods of this type have very good stability properties and hence could be good candidates for solving stiff problems. However, implicitness is also a costly affair and we will see how Rosenbrock and semi-implicit methods can become a good compromise in these circumstances.

CHAPTER 2

RUNGE-KUTTA AND SEMI-IMPLICIT METHODS

2.1 General form of Runge-Kutta methods

An s-stage Runge-Kutta method is given by the following formula:

(2.1.1) $$y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i, \text{ where}$$

$$k_i = hf(x_n + c_i h,\ y_n + \sum_{j=1}^{s'} a_{ij} k_j), \qquad i = 1, 2, \dots s.$$

If $s' = (i-1)$ (and $k_1 = f(x_n, y_n)$), the method is *explicit*; if $s' = i$, the method is *diagonally implicit** and if $s' = s$ the method is *fully implicit*.

The value of the real parameters b_i, c_i and a_{ij} define the method and are often presented as a so-called *Butcher tableau*, as shown in figure

* Originally, the terms 'semi-implicit' and 'semi-explicit' were used but 'diagonally implicit' is more descriptive and is now preferred. Furthermore, 'semi-implicit' is now used for somewhat different methods, as we shall see later.

2.1. A convention generally adopted is to use $c_i = \sum_{j=1}^{s} a_{ij}$, for $1 \leq i \leq s$.

$$\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\
c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\
\cdot & \cdot & \cdot & & \cdot \\
\cdot & \cdot & \cdot & & \cdot \\
\cdot & \cdot & \cdot & & \cdot \\
c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\
\hline
 & b_1 & b_2 & \cdots & b_s
\end{array}$$

Figure 2.1: Butcher tableau

Each k_i can be interpreted as an approximation to the derivative $y'(x)$ and the summation $\sum_{i=1}^{s} b_i k_i$ as a weighted mean of these approximations.

2.2 Explicit Runge-Kutta methods

Explicit Runge-Kutta methods have been used since the beginning of this century. Shown below are results proven by Butcher [15,17] on the minimum number of stages required to attain a given order:

(2.2.1)

Order of method	1	2	3	4	5	6	7	8
Min no of stages	1	2	3	4	6	7	9	11

In Butcher [17], it is also proven that order p>8 is not possible with (p+2) stages or less.

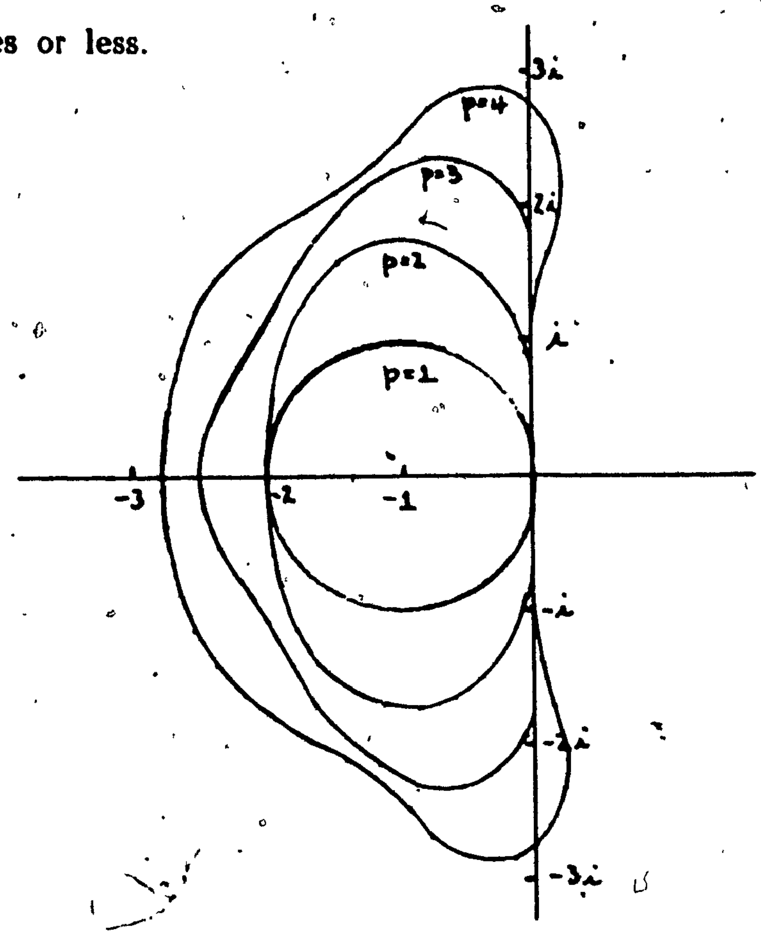


Figure 2.2: Stability regions for explicit methods

Finally, the regions of absolute stability for explicit Runge-Kutta methods of order 1, 2, 3 and 4 and the minimal number of stages are shown in figure 2.2. They were taken from Lambert [36], page 227. It is obvious that such small regions make these methods — and this is true of all explicit Runge-Kutta methods — unsuitable for stiff problems.

2.3 Fully implicit Runge-Kutta methods

The systematic study of implicit Runge-Kutta processes was initiated by Butcher in the early 1960's [13]. In this paper, he lists coefficients related to conditions that the a_{ij}, b_i and c_i of an implicit method must satisfy to reach a certain order, the so-called *order conditions*. Values are given up to order 8, for which there are 108 conditions to satisfy!

Butcher [14] has showed that for any $s \geq 2$, there exists an s-stage implicit method of order 2s. In fact, there is a one-to-one correspondence between these methods and s-point Gauss-Legendre quadrature formulas. Furthermore, these methods have been shown to be A-stable. The first proof of this is due to Ehle [25] but several other simpler proofs were provided later (see Nørsett [37], Crouzeix, Ruamps [21], Wanner Hairer and Nørsett [57]).

Even if we can derive A-stable implicit methods of arbitrarily high order, our troubles with stiffness are not over just yet... We must now deal with the implicitness of the problem and this is not an easy task. For instance, Butcher [14] has proven that the simple iterative scheme

$$(2.3.1) \qquad k_i^{[t+1]} = hf(x_n + c_i h,\ y_n + h\sum_{j=1}^{i-1} a_{ij} k_j^{[t+1]} + h\sum_{j=i}^{s} a_{ij} k_j^{[t]}),$$

for $t = 0, 1, \ldots$, will converge provided that $h < \frac{1}{L(u+v)}$, where u and v are constants related to the absolute value of the (a_{ij}) matrix elements and L is a Lipschitz constant (see 1.2.3) for the problem.

Thus, if we were to use this simple iterative scheme to resolve the implicitness of the problem, it would impose upon us a very severe limitation on the stepsize (remember that L is large for stiff problems) and we would loose the advantage we were seeking of an A-stable method to begin with!

However there are ways out of this. For example, we can use a Newton-Raphson type of iteration scheme instead of the simple one presented above. We can then obtain much less severe restrictions on the stepsize, but the computational complexity of the method would still be very important.

To help on this last point, some linear transformations have been proposed by Butcher [16] and Bickart [5] that significantly reduce the complexity of fully implicit schemes. For instance, the LU-decompositions needed for the Gauss-Legendre fully implicit methods require $O(\frac{sm^3}{3})$ operations for the transformed system as opposed to $O(\frac{s^3m^3}{3})$ for the original one.

2.4 Diagonally and singly implicit Runge-Kutta methods

The situation is somewhat simpler for diagonally implicit Runge-Kutta (DIRK) methods, in which case we obtain a splitting of the ms simultaneous non-linear equations into s distinct sets of m equations (see for example Nørsett [41], Alexander [4]). In fact, the methods studied in the previous 2 references have the additional feature that the elements a_{ii} of the Butcher tableau are all set equal. Such methods are then called singly diagonally implicit Runge-Kutta (SDIRK) methods.

SDIRK methods are actually a subset of so-called singly-implicit Runge-Kutta methods (see Burrage [10], Nørsett [40]) for which the (a_{ij}) matrix is in general **not** lower triangular — as it is for DIRK methods — but nonetheless has only one eigenvalue. If we use Butcher's transformation, these singly-implicit schemes also reduce to s linear systems of dimension m.

We may expect that for a given number of stages the above methods will have a lower attainable order than fully implicit methods. This is indeed the case: s-stage singly-implicit methods (including the SDIRK variety) have a maximum attainable order of (s+1). Proofs can be found in Burrage [10], Nørsett and Wolfbrandt [43], Wolfbrandt [59].

2.5 Rosenbrock and semi-implicit Runge-Kutta methods

When dealing with stiff problems, a Newton-Raphson type of scheme is generally used to solve the systems of equations resulting from any of the above implicit methods. This requires computation of the Jacobian matrix $J = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}$.

Rosenbrock [46] developed new methods which incorporate the Jacobian matrix *directly into* their defining formulas and then only require linear systems of equations to be solved. To save on Jacobian evaluations, it is also possible to use approximate results for J or even an arbitrary matrix A; this is the idea behind semi-implicit

Runge-Kutta methods.

Semi-implicit Runge-Kutta methods for the solution of problem (1.1) are given by the following formula:

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i \mathbf{k}_i, \quad \text{where}$$

(2.5.1) $$(I - \gamma h A)\mathbf{k}_i = h\mathbf{f}(x_n + c_i h,\ \mathbf{u}_i) + hA\sum_{j=1}^{i-1}\gamma_{ij}\mathbf{k}_j, \quad \text{and}$$

$$\mathbf{u}_i = \mathbf{y}_n + \sum_{j=1}^{i-1}\alpha_{ij}\mathbf{k}_j, \qquad i = 1, 2, \dots s.$$

This defines an s-stage method. It is also usual to adopt the additional restriction $c_i = \sum_{j=1}^{i-1}\alpha_{ij}$, $i = 1, 2, \dots s$. Note that the $\mathbf{k}_i$'s can be computed consecutively and that the right-hand side of the second equation above is then readily available for each value of i; thus all we need to do is to solve s linear systems of dimension m with different 'right-hand sides', using the same LU-decomposition of $(I - \gamma hA)$ all along.

In general, A can be an arbitrary m by m matrix. If $A \equiv 0$, we are back to standard explicit Runge-Kutta methods. On the other hand, if we choose $A \equiv J = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}$, the exact Jacobian, and $\gamma_{ij}=0$, we obtain simple Rosenbrock methods; for γ_{ij} non-null, the methods have been designated by various names including 'modified Rosenbrock', 'ROW' (for Rosenbrock-Wanner), 'generalized Runge-Kutta', and 'Kaps-Rentrop' methods. However, the common practise is now to call 'Rosenbrock methods', all schemes sharing the distinctive feature of having the exact Jacobian directly in their defining formulas.

Rosenbrock methods have been studied extensively because of their relatively low computation costs (as compared to fully implicit methods) and yet good overall linear stability properties [6,7,34,35,44,47,60].

If A is not the exact Jacobian, it is usually taken to be some approximation to the Jacobian. As mentioned above, such methods were introduced to reduce the number of Jacobian evaluations substantially (for example we can use the same evaluation for many steps), thus making the methods more efficient. Originally coined *W-methods* by Steihaug and Wolfbrandt [50], they are now known as *semi-implicit Runge-Kutta* methods. Dekker and Verwer [24] call them Runge-Kutta-Rosenbrock methods in chapter 9 of their book.

There exists a very close connection between Rosenbrock and SDIRK methods, which was demonstrated by Steihaug and Wolfbrandt [50]. We will now show this.

For an s-stage SDIRK method with $a_{ii}=\gamma$, i = 1, 2, ... m, we must solve s uncoupled non-linear systems of the form:

$$\text{(2.5.2)} \qquad \mathbf{F}(\mathbf{k}_i) \equiv \mathbf{k}_i - \mathbf{h}\mathbf{f}(x_n+c_ih,\ \mathbf{y}_n + \sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j + \gamma\mathbf{k}_i) = 0.$$

Now if we start with initial guesses $\mathbf{k}_i^{[0]}$ for the unknown $\mathbf{k}_i$'s, and iterate *only one step* using a Newton-like procedure, we obtain

$$\text{(2.5.3)} \qquad (I - \gamma hJ)(\mathbf{k}_i^{[1]} - \mathbf{k}_i^{[0]}) = \mathbf{h}\mathbf{f}(x_n+c_ih,\ \mathbf{y}_n + \sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j^{[1]} + \gamma\mathbf{k}_i^{[0]}) - \mathbf{k}_i^{[0]}.$$

Choosing $\mathbf{k}_i^{[0]} = -\frac{1}{\gamma}\sum_{j=1}^{i-1}\gamma_{ij}\mathbf{k}_j^{[1]}$ as initial guesses and expanding the above formula, we arrive at

$$(2.5.4) \qquad (I - \gamma hJ)\mathbf{k}_i^{[1]} = h\mathbf{f}(x_n+c_ih,\ \mathbf{y}_n + \sum_{j=1}^{i-1}(a_{ij} - \gamma_{ij})\mathbf{k}_j^{[1]}) + hJ\sum_{j=1}^{i-1}\gamma_{ij}\mathbf{k}_j^{[1]}.$$

If we drop the iteration number and set $\alpha_{ij} = a_{ij} - \gamma_{ij}$, we clearly obtain our Rosenbrock method. Replacing J by A, we have (2.5.1).

2.6 Stability functions

We now consider the application of semi-implicit methods (2.5.1) to the scalar test equation (1.4.9). Here the matrix A, which is usually taken as an approximation to the Jacobian, reduces to the scalar λ, which is also $\frac{\partial f}{\partial y}$. Thus the results we are about to obtain will also be valid for Rosenbrock methods. Furthermore, given that method (2.5.1) can be considered as a linearization of an SDIRK method as we have just shown, these results will also be valid for SDIRK schemes.

Using $z = h\lambda$, (2.5.1) applied to (1.4.9) yields

$$(2.6.1) \qquad (1 - \gamma z)k_i = z(u_i + \sum_{j=1}^{i-1}\gamma_{ij}k_j), \qquad i = 1,\ 2,\ \ldots\ s.$$

Substituting for u_i and defining $\beta_{ij} \equiv \alpha_{ij} + \gamma_{ij}$, we then obtain

$$(2.6.2) \qquad k_i = w(y_n + \sum_{j=1}^{i-1}\beta_{ij}k_j), \qquad i = 1,\ 2,\ \ldots\ s,$$

where we have used $w = \frac{z}{1 - \gamma z}$.

At this point, we define the s-vectors $\mathbf{b} = (b_1, b_2, \dots b_s)^T$, as well as $\mathbf{k} = (k_1, k_2, \dots k_s)^T$ and $\mathbf{1} = (1, 1, \dots 1)^T$; we also define the square matrix $B = (\beta_{ij})$. Using this notation, we can rearrange (2.6.2) as

(2.6.3) $$(I - wB)\mathbf{k} = wy_n\mathbf{1} \quad \text{or}$$

(2.6.4) $$\mathbf{k} = (I - wB)^{-1}wy_n\mathbf{1}.$$

Given B has only zero elements on and above its main diagonal, we have $B^s = 0$ and $(I - wB)^{-1} = I + wB + w^2B^2 + \dots w^{s-1}B^{s-1}$. We then have

(2.6.5) $$\mathbf{k} = y_n\sum_{j=1}^{s} w^jB^{j-1}\mathbf{1},$$

which results in

(2.6.6) $$y_{n+1} = (1 + \sum_{j=1}^{s} \mathbf{b}^TB^{j-1}\mathbf{1}w^j)y_n.$$

We now define the *stability function* R as:

(2.6.7) $$R(z) = 1 + \sum_{j=1}^{s} \mathbf{b}^TB^{j-1}\mathbf{1}w^j.$$

Similarly, using the s-vector $\mathbf{a}_i = (\alpha_{i,1}, \alpha_{i,2}, \dots \alpha_{i,i-1}, 0, \dots 0)^T$, we can write u_i as

(2.6.8) $$u_i = (1 + \sum_{j=1}^{i-1} \mathbf{a}_i^TB^{j-1}\mathbf{1}w^j)y_n,$$

from which we define the *internal stability functions*

(2.6.9) $$R_i(z) = 1 + \sum_{j=1}^{i-1} \mathbf{a}_i^TB^{j-1}\mathbf{1}w^j .$$

From (2.6.6) and definition (1.4.2), it immediately follows that the absolute stability region of the method (2.5.1) is then simply that region of the complex z-plane for which we have $|R(z)| < 1$. Also, from (2.6.7) and the definition of w, R(z) must be a rational function such that

(2.6.10) $$R(z)(1 - \gamma z)^s = \sum_{j=0}^{p} l_j z^j, \quad \text{with } p \leq s.$$

Now assume that the method is of exact order p. Since the analytic solution of the test equation is $y(x) = y_0 \exp(\lambda(x - x_0))$, we must have that R(z) is an approximation to e^z of *at least* order p.

At this point we make use of a result from Tricomi [51] page 217, as quoted in Kaps [31]:

(2.6.11) $$e^z(1 - \gamma z)^s = \sum_{j=0}^{\infty} L_j^{s-j}(\tfrac{1}{\gamma})(-\gamma z)^j, \qquad |\gamma z| < 1,$$

where $L_n^\alpha(x)$ is the *generalized Laguerre polynomial* (see [1]) defined as*

(2.6.12) $$L_n^\alpha(x) \equiv \sum_{j=0}^{n} (-1)^j \binom{n+\alpha}{n-j} \frac{x^j}{j!}.$$

Direct comparison of (2.6.10) and (2.6.11) indicates that for the method to be of order p we must have

(2.6.13) $$l_j = (-\gamma)^j L_j^{s-j}(\tfrac{1}{\gamma}), \qquad j = 0, 1, \dots p.$$

Given (2.6.10), this result is also a direct consequence of proposition 6

* A simple relationship exists between generalized Laguerre polynomials and derivatives of the Laguerre polynomials: $L_j^{s-j}(\frac{1}{\gamma}) = (-1)^{s-j} L_s^{(s-j)}(\frac{1}{\gamma})$.

in Nørsett, Wanner [42].

The striking fact about the above result is that the stability function, and hence the absolute stability region of the method, depend on *only one* of the method's parameters, namely γ; for this reason, γ is known as the *stability parameter*. In chapter 4, we shall investigate the linear stability properties of these methods as a function of γ.

To make (2.6.13) more concrete, we now give explicit expressions for the first few polynomials $l_j(\gamma)$:

$$
\begin{aligned}
l_0 &= 1\\
l_1 &= -s\gamma + 1\\
l_2 &= \binom{s}{2}\gamma^2 - s\gamma + \frac{1}{2} \qquad (2.6.14)\\
l_3 &= -\binom{s}{3}\gamma^3 + \binom{s}{2}\gamma^2 - \frac{s}{2}\gamma + \frac{1}{6}\\
l_4 &= \binom{s}{4}\gamma^4 - \binom{s}{3}\gamma^3 + \binom{s}{2}\gamma^2 - \frac{s}{6}\gamma + \frac{1}{24}.
\end{aligned}
$$

Finally we return to a statement made in section 1.4 about the stability of numerical methods when applied to linear problems with constant coefficients.

If we apply method (2.5.1) to the problem $y' = Ay$, where A is a constant matrix, we will obtain a result equivalent to (2.6.6) namely

$$\mathbf{y}_{n+1} = R(hA)\mathbf{y}_n. \qquad (2.6.15)$$

R(hA) is now a matrix which is still equal to the right-hand side of (2.6.7) if we understand w as now being the matrix $hA(I - \gamma hA)^{-1}$. Essentially, we just replace $z = h\lambda$ by $z = hA$ throughout.

Now suppose that λ_i, $i = 1, 2, \ldots m$, are the eigenvalues of A and that they all have negative real parts. From (1.4.2), it follows that the exact solution asymptotically tends to 0 and we would again want the computed solution to exhibit the same behaviour.

Taking norms, (2.6.15) implies $\dfrac{\|y_{n+1}\|_2}{\|y_n\|_2} \leq \|R(hA)\|_2$. If A is a normal* matrix, then so is R(hA); and for normal matrices, the 2-norm is equal to the spectral radius**. Since the eigenvalues of R(hA) are simply $R(h\lambda_i)$, this means

(2.6.16) $$\|R(hA)\| = \max_i |R(h\lambda_i)|.$$

So we see that if $|R(h\lambda_i)| < 1$, for all i (i.e. if $h\lambda_i$ is in the absolute stability region of the method), the approximate solution will also tend to 0 for large x.

2.7 Error estimation: Richardson and modified Richardson extrapolation

As the numerical integration proceeds, one needs a suitable local truncation error *estimate*, for stepsize control. For Runge-Kutta methods, the two most commonly used techniques to compute such an estimate are Richardson extrapolation and the embedding technique.

* A is normal if $A^H A = AA^H$, where A^H is the conjugate transpose of A.

** The spectral radius of A is the maximum modulus of its eigenvalues.

In the *embedding technique*, the idea is to perform the step from x_n to x_{n+1} *twice*, with the integration method of order p and then with *another method* of order (p+1). The difference between the 2 computed values yields an aymptotically correct error estimate. To reduce the computational cost of obtaining this estimate, the p^{th} order method is actually a by-product of the $(p+1)^{th}$ order method, sharing all it's early stages. Thus the error estimate is obtained quite cheaply. On the other hand, such embedded pairs of methods are more difficult to derive, having an increased number of algebraic equations to solve. As examples of such pairs of the .Rosenbrock type, see Kaps, Rentrop [34] and Kaps, Ostermann [32]; for the semi-implicit type, see method W23M1 in Kaps [31].

The other approach, *Richardson extrapolation*, proceeds as follows: we first integrate two steps, from x_n to x_{n+1} and then from x_{n+1} to x_{n+2}, with stepsize h, obtaining an approximate value y_{n+2} for the exact solution $y(x_{n+2})$. Then we integrate one double step of size 2h, directly from x_n to x_{n+2}, thus obtaining another estimate y^+_{n+2} for $y(x_{n+2})$.

The error estimate for a method of order p is then obtained as

$$\text{est}^+ = \frac{y_{n+2} - y^+_{n+2}}{2^p - 1}, \tag{2.7.1}$$

and it has the same degree of accuracy as the embedding technique. Let's show this. For simplicity (and illustration purposes below!) we consider a one-dimensional problem.

First consider integrating 2 steps of size h, for which the process is shown in figure 2.7 below. Assuming that no errors have been made so far, we start integration at $(x_n, y(x_n))$ and obtain the approximate solution point (x_{n+1}, y_{n+1}). From (1.3.5) we have for this step the following truncation error:

(2.7.2) $$T_{n+1} = y(x_{n+1}) - y_{n+1} = \Psi(x_n, y(x_n))h^{p+1} + O(h^{p+2}) .$$

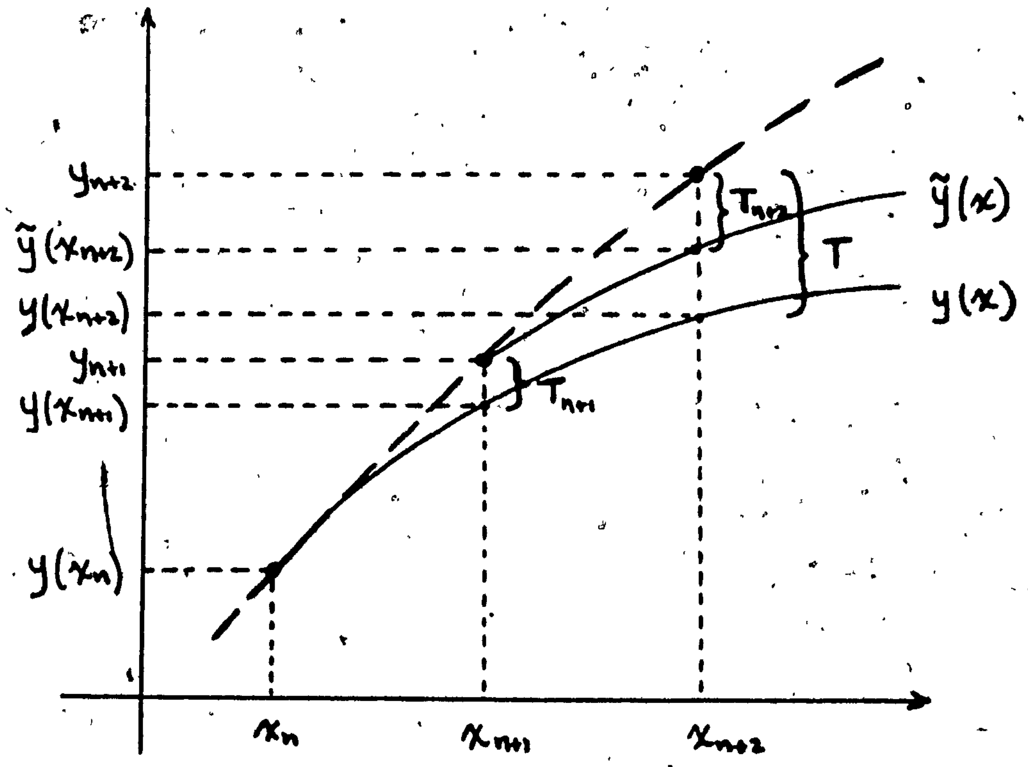


Figure 2.7: Truncation errors for 2 steps of size h

Obviously the point (x_{n+1}, y_{n+1}) is now on another solution branch $\tilde{y}(x)$ and the best we could do in the next step, if we made no error at all, would be to obtain $\tilde{y}(x_{n+2})$. However, there is of course again some truncation error and we instead obtain y_{n+2} with the following truncation error:

(2.7.3) $$T_{n+2} = \tilde{y}(x_{n+2}) - y_{n+2} = \Psi(x_{n+1},y_{n+1})h^{p+1} + O(h^{p+2})$$

What we want however is an estimate for $T = y(x_{n+2}) - y_{n+2}$, the truncation error for the two steps together. We can easily show that $T_{n+1} + T_{n+2}$ is appropriate for this. We have

(2.7.4) $$T_{n+1} + T_{n+2} = y(x_{n+1}) + [\tilde{y}(x_{n+2}) - y_{n+1}] - y_{n+2}.$$

Since $y_{n+1} = \tilde{y}(x_{n+1})$, we can expand the bracketed term above in Taylor series as

$$\tilde{y}(x_{n+2}) - y_{n+1} = \sum_{i=1}^{p+1} \frac{h^i y^{(i)}(x_{n+1},y_{n+1})}{i!} + O(h^{p+2})$$

(2.7.5) $$= \sum_{i=1}^{p+1} \frac{h^i y^{(i)}(x_{n+1},y(x_{n+1}))}{i!} + O(h^{p+2}),$$

where the last step is written using the fact that y_{n+1} is already an order p approximation to $y(x_{n+1})$.

Now (2.7.4) and (2.7.5) yield

(2.7.6) $$y(x_{n+2}) - y_{n+2} = T_{n+1} + T_{n+2} + O(h^{p+2}).$$

Using their respective expressions for T_{n+1} and T_{n+2}, we finally obtain

(2.7.7) $$y(x_{n+2}) - y_{n+2} = 2\Psi(x_n,y(x_n))h^{p+1} + O(h^{p+2}).$$

On the other hand, the double step of size 2h carries with it a truncation error of

(2.7.8) $$y(x_{n+2}) - y^{+}_{n+2} = \Psi(x_n,y(x_n))(2h)^{p+1} + O(h^{p+2}).$$

Substracting (2.7.7) from (2.7.8) we finally have our estimate

(2.7.9) $$\text{est}^{+} = \frac{y_{n+2} - y^{+}_{n+2}}{2^p - 1} = 2\Psi(x_n,y(x_n))h^{p+1} + O(h^{p+2}).$$

We can correct y_{n+2} using est^+ to obtain an improved approximation y_{ex} given by

(2.7.10) $$y_{ex} = y_{n+2} + est^+ = y(x_{n+2}) + O(h^{p+2}).$$

Clearly then, the estimate (2.7.1), like the one obtained with the embedding technique, corresponds to the difference of two approximations of $y(x_{n+2})$, one of order $(p+1)$ and the other of order p.

Richardson extrapolation allows stepsize changes only every second step and will require more work per step than the embedding technique; however, the additional constraints imposed by the latter approach will usually mean that the resulting method will be less efficient. Thus smaller stepsizes will in general result from the embedding technique and Richardson extrapolation could then still be competitive. A detailed study, comparing the 2 error estimation schemes for Rosenbrock methods, has been carried out (see Poon [45] and Kaps, Poon, Bui [33]) and showed that this was indeed the case. It was found that for low tolerances ($\sim 10^{-2}$) embedding was superior while for high tolerances ($< 10^{-5}$) extrapolation was superior; for moderate tolerances ($\sim 10^{-4}$), both approaches were comparable.

It is possible to reduce the extra work required for Richardson extrapolation by modifying its approach. Instead of using the same method for the double step (size 2h), one uses another method with stability parameter $\frac{\gamma}{2}$. Thus no new LU-decomposition is required for this double step. Furthermore, other coefficients of the second method can often be chosen so that some or all of the **k**'s are the same as for the basic integration method, which then results in a very cheap error estimate. This *modified Richardson extrapolation* scheme was first proposed by Cash [18] and further investigated by Bui, Poon [9]. Code W3X1 of Kaps [31] uses modified Richardson extrapolation as error estimate for a semi-implicit method of order 3. See also Kaps, Ostermann [32].

If $\overset{*}{y}_{n+2}$ denotes the computed solution value for the double step (with the method which uses $\frac{\gamma}{2}$ instead of γ), we can use the following error estimate for modified Richardson extrapolation:

$$\text{est}^* = \frac{y_{n+2} - \overset{*}{y}_{n+2}}{2^p C^*/C - 1}, \tag{2.7.11}$$

where C^*/C is the ratio of the so-called *error constants* of the 2 methods, using γ and $\frac{\gamma}{2}$ respectively.

Here too we can obtain an improved approximation by correcting $\overset{*}{y}_{n+2}$, namely $y_{mx} = \overset{*}{y}_{n+2} + \text{est}^*$. In this case however it is only for *linear* problems with constant coefficients that y_{mx} will be an order (p+1) approximation to $y(x_{n+2})$. Nonetheless, (2.7.11) can be used as error estimate for non-linear problems also, with good results.

CHAPTER 3

ERROR CONSTANTS AS A FUNCTION OF γ

As seen previously, when applied to the scalar test equation (1.4.9). the s-stage semi-implicit Runge-Kutta method (2.5.1) and many other methods yield the following result:

(3.1) $$y_{n+1} = R(z;\gamma)y_n, \quad z = h\lambda.$$

If the method is of order p, then the stability function will be *at least* an order p approximation to e^z i.e.

(3.2) $$R(z;\gamma) = \frac{P(z;\gamma)}{Q(z;\gamma)} = \frac{P(z;\gamma)}{(1 - \gamma z)^s} = e^z + Cz^{p+1} + O(h^{p+2}).$$

$P(z;\gamma)$ is a polynomial in z whose coefficients are given by (2.6.11). These coefficients can be obtained more easily however, if we just multiply $(1 - \gamma z)^s$ by the series expansion of e^z. Finally, C is called the *error constant* and also depends only on γ.

Such rational approximations to the exponential, called Restricted-Denominator approximations by Nørsett, were studied extensively by Nørsett [38,39,41], Siemieniuch [49] and Wolfbrandt [60].

3.1 The optimal order case

The authors mentioned above focussed their attention on the *optimal* *order* case, which is (p+1) as we have seen previously. In this situation, we have C=0 and the error constant is given by C_2 where $R(z;\gamma) = e^z + C_2 z^{p+2} + O(h^{p+3})$. These optimal order approximations to the exponential are called *Restricted-Padé* approximations.

Furthermore, the cases of interest are p=s and p=(s-1), the latter one potentially allowing L-stability. In both situations, for a given value of s, there are only s distinct values of γ which yield optimal order methods. For the methods with p=s and p=(s-1), we will denote these special values by $\hat{\gamma}_i$ and $\bar{\gamma}_i$ respectively, i = 1, 2, ... s. More precisely (this will become clear later), $\frac{1}{\hat{\gamma}_i}$ is the ith smallest zero of $L_s^1(x)$, while $\frac{1}{\bar{\gamma}_i}$ is the ith smallest zero of $L_s(x)$*.

Table 3.1.1 shows the only methods for the case p=(s-1) which are L-stable, giving their order (s), choice of γ and absolute error constants. Table 3.1.2 shows the only A-stable methods for the case p=s, which are of order (s+1). The information was drawn from Wolfbrandt [60] and Nørsett [39] respectively.

First we note the absence of an L-stable method of order 7 (with 7 stages) and of an A-stable method of order 5 (with 4 stages). Of course, there are methods which are nearly A-stable (i.e. A(α)-stable, for α close to $\frac{\pi}{2}$); Wolfbrandt [60] lists the maximum stability angles

* $L_s(x) = L_s^0(x)$, is the classical Laguerre polynomial.

s	γ	$\|C_2\|$
1	$\bar{\gamma}_1 \approx 1.00000$	5.0E-1
2	$\bar{\gamma}_1 \approx 1.70711$	1.4E-0
2	$\bar{\gamma}_2 \approx 0.29289$	4.0E-2
3	$\bar{\gamma}_2 \approx 0.43587$	2.6E-2
4	$\bar{\gamma}_2 \approx 0.57282$	2.7E-2
5	$\bar{\gamma}_3 \approx 0.27805$	5.3E-4
6	$\bar{\gamma}_3 \approx 0.33414$	3.4E-4
8	$\bar{\gamma}_4 \approx 0.23437$	2.7E-6

Table 3.1.1: L-stable, optimal order methods (p = s-1)

and error constants for all optimal order methods of order up to 15, for the case p=(s-1).

Another observation is that by optimizing the order of the method, we sometimes end up with relatively large error constants. And even when the error constant is acceptable, there are further difficulties because we also need some means of estimating the local truncation error. If we use a second method for this purpose, as would be the case for the embedding technique or modified Richardson extrapolation, we would like it to have equally good accuracy (and stability) features as the basic integration method and this is clearly

s	γ	$\lvert C_2\rvert$
1	$\hat{\gamma}_1 \approx 0.50000$	8.3E-2
2	$\hat{\gamma}_1 \approx 0.78868$	9.0E-2
3	$\hat{\gamma}_1 \approx 1.06858$	1.6E-1
5	$\hat{\gamma}_2 \approx 0.47327$	1.3E-3

Table 3.1.2: A-stable, optimal order methods (p = s)

not possible, if we require maximal order and L- or A-stability.

For example, in trying to optimize all these factors for numerical testing of optimal order 3 schemes, Wolfbrandt ended up discarding the $\gamma = \bar{\gamma}_2$, L-stable method and chose instead $\gamma = \bar{\gamma}_3$ for which $|C_2| = 3.9\text{x}10^{-3}$ (better than $2.6\text{x}10^{-2}$) and A(75.57°)-stability is attained. This also enabled a more reliable error estimate. For maximal order 4, things were nicer and he used $\gamma = \bar{\gamma}_3$, with corresponding $|C_2| = 1.1\text{x}10^{-3}$ (better than $2.7\text{x}10^{-2}$) and A(89.55°)-stability attainable.

In this and the following chapter, we will investigate order p methods. We thus sacrifice maximal order to see whether we can achieve better overall optimization of the accuracy and stability features of the integration and error estimation schemes.

3.2 Continuation with the higher order approximation

As seen in section 2.7, error estimation often involves (or can lead to) 2 approximations, one of order p and the other of order (p+1). While it is customary to use the lower order method as the basic integration scheme, it has also been tried to continue the integration with the higher order approximation (for example, for Richardson extrapolation, we would use y_{ex} or y_{mx}, instead of y_{n+2}, as our computed solution to continue the integration process further).

Examples of this for the embedding technique and Richardson extrapolation can be found in Kaps, Rentrop [34] and Kaps [31], for Rosenbrock and semi-implicit methods respectively.

If we plan to use y_{ex} or y_{mx} for continuation of the integration, we must clearly make sure that they also have the accuracy and stability features required for the problem. We now establish their stability functions.

Again using the scalar test equation (1.4.9), we define the stability function for the corrected approximation y_{ex} with the following relation $y_{ex} = y_{n+2} + est^{+} \equiv R_{ex}(2z;\gamma)y_n$; similarly, we can use the relation $y_{mx} = y_{n+2} + est^{*} \equiv R_{mx}(2z;\gamma)y_n$ to define the stability function for y_{mx}. Using (2.7.1) and (2.7.11), and also the fact that $y_{n+2} = R^{2}(z;\gamma)y_n$, we easily obtain

$$(3.2.1) \qquad R_{ex}(2z;\gamma) = \frac{2^{p}R^{2}(z;\gamma) - R(2z;\gamma)}{2^{p} - 1} \equiv \frac{P_{ex}(2z;\gamma)}{Q_{ex}(2z;\gamma)}$$

$$(3.2.2) \qquad R_{mx}(2z;\gamma) = \frac{2^{p\frac{C^*}{C}}R^{2}(z;\gamma) - R(2z;\frac{\gamma}{2})}{2^{p\frac{C^*}{C}} - 1} \equiv \frac{P_{mx}(2z;\gamma)}{Q_{mx}(2z;\gamma)}$$

where the (*) indicates the use of $\frac{\gamma}{2}$ instead of γ.

While R, R_{ex} and R_{mx} are all rational functions in z, the degrees of their numerator and denominator polynomials differ. They are p and s respectively for P and Q, (2s+p) and 3s for P_{ex} and Q_{ex}, and (s+p) and 2s for P_{mx} and Q_{mx}. The degrees for modified Richardson extrapolation are smaller because $Q(2z,\frac{\gamma}{2}) = Q(z;\gamma)$ and this allows a simpler common denominator in this case.

The solution of the scalar test equation yields $y(x_{n+2}) = e^{2z}y(x_n)$, and since y_{ex} and y_{mx} (in the linear case) are order (p+1) approximations to $y(x_{n+2})$, it follows that R_{ex} and R_{mx} must be *at least* order (p+1) approximations to e^{2z}.

So we end this section by defining the *error constants* C, C_{ex} and C_{mx} respectively for the basic method, Richardson extrapolation and modified Richardson extrapolation:

$$(3.2.3) \qquad R(z;\gamma) = e^{z} + Cz^{p+1} + C_2z^{p+2} + O(h^{p+3})$$

$$(3.2.4) \qquad R_{ex}\text{[illegible]} = e^{2z} + C_{ex}(2z)^{p+2} + O(h^{p+3})$$

$$(3.2.5) \qquad R_{mx}(2z;\gamma) = e^{2z} + C_{mx}(2z)^{p+2} + O(h^{p+3})$$

3.3 Error constants for order p methods

We need to derive expressions for C, C_{ex} and C_{mx}. First we will show that C_{ex} and C_{mx} can both be expressed in terms of C and C_2 of the basic method; then we will obtain formulas for C and C_2.

From (3.2.3) we have

(3.3.1) $$R^2(z;\gamma) = e^{2z} + 2Cz^{p+1} + 2(C + C_2)z^{p+2} + O(h^{p+3}).$$

Now from (3.2.1), using the result just obtained and (3.2.3) with $2z$ instead of z, we arrive at

(3.3.2) $$R_{ex}(2z;\gamma) = e^{2z} + \frac{C - C_2}{2(2^p - 1)}(2z)^{p+2} + O(h^{p+3}).$$

By comparison with (3.2.4) we see that

(3.3.3) $$C_{ex} = \frac{C - C_2}{2(2^p - 1)}.$$

We do likewise for the modified Richardson extrapolation case. First, if we use $\frac{\gamma}{2}$ instead of γ and $2z$ instead of z in (3.2.3) and indicate this by a (*) for the error constants, we obtain:

(3.3.4) $$R(2z;\tfrac{\gamma}{2}) = e^{2z} + C^*(2z)^{p+1} + C_2^*(2z)^{p+2} + O(h^{p+3}).$$

From (3.2.2), using (3.3.1) and (3.3.4), a similar result is achieved, namely

(3.3.5) $$R_{mx}(2z;\gamma) = e^{2z} + \frac{C^*(C + C_2) - 2CC_2^*}{2(2^pC^* - C)}(2z)^{p+2} + O(h^{p+3}).$$

And by comparison with (3.2.5), we finally get

(3.3.6) $$C_{mx} = \frac{C^*(C + C_2) - 2CC_2^*}{2(2^pC^* - C)}.$$

So the problem is now simply to find expressions for C and C_2 of the basic method. Looking back at the end of section 2.6, we can obtain the following useful relation:

$$(3.3.7) \qquad e^z - R(z;\gamma) = \frac{\sum_{j=p+1}^{\infty} l_j(\gamma)z^j}{(1 - \gamma z)^s},$$

where $l_j(\gamma)$ is defined as in (2.6.11), but this time with no restriction on j.

Using the binomial expansion for $(1 - \gamma z)^{-s}$ gives us

$$(3.3.8) \qquad e^z - R(z;\gamma) = l_{p+1}(\gamma)z^{p+1} + (\gamma s l_{p+1}(\gamma) + l_{p+2}(\gamma))z^{p+2} + O(h^{p+3}),$$

from which we obtain the needed information:

$$(3.3.9) \qquad \begin{aligned} -C &= l_{p+1}(\gamma) \\ -C_2 &= \gamma s l_{p+1}(\gamma) + l_{p+2}(\gamma) \\ -C^* &= l_{p+1}(\tfrac{\gamma}{2}) \\ -C_2^* &= \frac{\gamma s}{2}\, l_{p+1}(\tfrac{\gamma}{2}) + l_{p+2}(\tfrac{\gamma}{2}) \end{aligned}$$

3.4 Computation and results

From (2.6.12) and (2.6.13) or directly from $e^z(1 - \gamma z)^s = \sum_{j=0}^{\infty} l_j(\gamma)z^j$, we obtain

$$(3.4.1) \qquad l_j(\gamma) = \sum_{i=0}^{\min(j,s)} \binom{s}{i}\frac{(-\gamma)^i}{(j - i)!}.$$

Since we are interested only in j=(p+1) and j=(p+2) for the error constants evaluation, and we are dealing with the cases p=s and p=(s-1), we will always have the upper limit of the summation equal to s,

At this point, we remark that using (3.4.1) one can show that $l_{p+1}(\gamma)$ is proportional to $L_s^1(\frac{1}{\gamma})$ for the case p=s, and proportional to $L_s(\frac{1}{\gamma})$ for the case p=(s-1). This justifies the special sets of values $\hat{\gamma}_i$ and $\bar{\gamma}_i$ yielding optimal order (p+1), as mentioned in section 3.1.

In order to obtain the values of the error constants, we can proceed as follows. Defining $T_i = \binom{s}{i}\frac{(-\gamma)^i}{(p+1-i)!}$, we first compute

$$l_{p+1}(\gamma) = \sum_{i=0}^{s} T_i \,,$$

(3.4.2)
$$l_{p+1}(\tfrac{\gamma}{2}) = \sum_{i=0}^{s} \frac{T_i}{2^i} \,,$$

$$l_{p+2}(\gamma) = \sum_{i=0}^{s} \frac{T_i}{(p+2-i)} \,,$$

$$l_{p+2}(\tfrac{\gamma}{2}) = \sum_{i=0}^{s} \frac{T_i}{2^i(p+2-i)} \,.$$

Then, from (3.3.3), (3.3.6) and (3.3.9) we arrive at

$$C = -l_{p+1}(\gamma),$$

(3.4.3)
$$C_{ex} = \frac{l_{p+2}(\gamma) - C(\gamma s - 1)}{2(2^p - 1)},$$

$$C_{mx} = \frac{(C - l_{p+2}(\gamma))\, l_{p+1}(\frac{\gamma}{2}) - 2C l_{p+2}(\frac{\gamma}{2})}{2(C + 2^p l_{p+1}(\frac{\gamma}{2}))}.$$

We see that C and C_{ex} are polynomials in γ, of degrees s and (s+1) respectively; on the other hand, C_{mx} is a rational function in γ, with a numerator polynomial of degree 2s and a denominator polynomial of degree s.

The above formulas were used to compute the *logarithm* (base 10) of the absolute error constants $|C|$, $|C_{ex}^{*}|$, and $|C_{mx}|$. In order to clearly locate discontinuities on these graphs, the zeroes of all 4 polynomials in γ were obtained.

This presented no problems for C and C_{ex}, but required some caution for the 2 polynomials of C_{mx}. For the denominator, the coefficient of γ^{i} cancels out for $i=p$. Consequently, for $p=s$, the denominator polynomial is only of order $(s-1)$, while for $p=(s-1)$ it is of order s but has no term in γ^{s-1}.

Similarly, for the numerator of C_{mx}, the coefficient of γ^{2s} is given by $2^{-s}((p+1-s)!)^{-2}(\frac{1}{p+2-s} - 1)$ which cancels for $p=(s-1)$. Thus, for $p=(s-1)$, the numerator will be of degree $(2s-1)$ instead of $2s$.

Finally, there is one more peculiarity of C_{mx} for $s=2$: in this case, both polynomials have a common zero. More precisely, we have for $s=p=2$

$$(3.4.4) \qquad C_{mx} = \frac{(\gamma - 0.5)(6\gamma^{3} - 21\gamma^{2} + 14\gamma - 2)}{96(\gamma - 0.5)},$$

and for $s=(p+1)= 2$, we have

$$(3.4.5) \qquad C_{mx} = \frac{(\gamma - 1)(-9\gamma^{2} + 10\gamma - 2)}{12(\gamma - 1)(\gamma + 1)}.$$

In the first case, for $\gamma=0.5$, we simply obtained the limiting value $\lim_{\gamma\to0.5} \log_{10}|C_{mx}| \approx -2.283$; in the second case, for $\gamma=1$, we chose $\lim_{\gamma\to1.0} \log_{10}|C_{mx}| \approx -1.380$.

The computed results are plotted in Appendix 1 as a function of γ, for $\gamma \in [0.0,2.0]$. Plots are given for the two cases $p=s$ and $p=(s-1)$, for $s = 1, 2, \ldots 8$. For each graph, the *continuous line* represents $\log_{10}|C|$, the *dashed line* represents $\log_{10}|C_{ex}|$ and the *dotted line* represents $\log_{10}|C_{mx}|$.

For $p \geq 4$, it is difficult to distinguish clearly the behaviour of the individual curves for small values of γ; therefore, a second graph is provided, showing an "enlargement" of the "small γ"-region of the main graph.

The values computed appear to match the results published by Wanner [56], which however deal only with the basic method, only with the case $p=s$ and only with $\gamma \in [0.0,1.0]$.

CHAPTER 4

LINEAR STABILITY AS A FUNCTION OF γ

In this chapter, we will investigate the linear stability properties of methods characterized by the stability function (3.2). We consider stability at infinity, A(0)-, A-, and A(α)-stability, for the cases $p=s$ and $p=(s-1)$. Note that the first 2 properties are preconditions of the next one and that A-stability is strengthened to L-stability for the case $p=(s-1)$.

4.1 γ-regions for stability at infinity

Definition 4.1.1 A numerical integration method is *stable at infinity* if its stability function satisfies

(4.1.1) $$\lim_{z\to\infty} |R(z;\gamma)| \leq 1.$$

We will investigate whether $R(z;\gamma)$, $R_{ex}(2z;\gamma)$ and $R_{mx}(2z;\gamma)$ satisfy this criteria. First, it is obvious that for the case $p=(s-1)$ it does, because then the above limit is 0 for all 3 stability functions. What about the case $p=s$?

For the *basic method*, using (2.6.8) and (2.6.11), (4.1.1) then becomes

(4.1.2) $$|L_s(\tfrac{1}{\gamma})| \leq 1.$$

The γ-regions for stability at infinity are then simply found by locating the intercepts of the Laguerre polynomial $L_s(x)$ with $y(x)=\pm 1$.

For *Richardson extrapolation*, using (3.2.1) and a little algebra reveals that the condition $|R_{ex}(\infty)| \leq 1$ is equivalent to

(4.1.3) $$-1 + 2^{-p} \leq L_s(\tfrac{1}{\gamma}) \leq 1.$$

Comparing (4.1.2) and (4.1.3), we expect intervals of stability at infinity for Richardson extrapolation to each have one boundary in common with an interval for the basic method and to be a bit narrower; but as p increases, they should overlap more and more exactly.

For *modified Richardson extrapolation*, using (3.2.2), the stability at infinity condition is

(4.1.4) $$\left|\frac{2^s C^*[L_s(\frac{1}{\gamma})]^2 - CL_s(\frac{2}{\gamma})}{2^s C^* - C}\right| \leq 1.$$

With (3.4.1) we can show that $C = \frac{(-\gamma)^s}{s+1}L_s^1(\frac{1}{\gamma})$ and then (4.1.4) yields

(4.1.5) $$-1 \leq \frac{(-1)^s\{L_s^1(\frac{2}{\gamma})[L_s(\frac{1}{\gamma})]^2 - L_s^1(\frac{1}{\gamma})L_s(\frac{2}{\gamma})\}}{(-1)^s\{L_s^1(\frac{2}{\gamma}) - L_s^1(\frac{1}{\gamma})\}} \leq 1,$$

where $(-1)^s$ was added to the numerator and denominator so that they both are positive as $\gamma \to 0^+$, which simplifies the interpretation of results below.

We must then have that the following 3 conditions hold *simultaneously* or else that the 3 reverse inequalities hold *simultaneously*:

$$(-1)^s\{L_s^1(\frac{2}{\gamma}) - L_s^1(\frac{1}{\gamma})\} \geq 0,$$

$$(4.1.6) \qquad (-1)^s\{L_s^1(\frac{2}{\gamma})[(L_s(\frac{1}{\gamma}))^2 - 1] - L_s^1(\frac{1}{\gamma})[L_s(\frac{2}{\gamma}) - 1]\} \leq 0,$$

$$(-1)^s\{L_s^1(\frac{2}{\gamma})[(L_s(\frac{1}{\gamma}))^2 + 1] - L_s^1(\frac{1}{\gamma})[L_s(\frac{2}{\gamma}) + 1]\} \geq 0.$$

The left-hand side of each condition above is a polynomial in $\frac{1}{\gamma}$ for which we can easily find the positive real roots and then find the γ-regions where each condition holds (or where the reverse holds). Then the problem is simply to find the intersection of these 3 sets of γ-regions where the conditions above hold together and the intersection of the 3 sets of γ-regions where together they do not hold.

For the numerical computation of the roots of the 3 polynomials of (4.1.6), a few more details must be looked at. In the first case, while formally it is of degree s, it turns out that its constant term is zero; thus we should "factor out" $\frac{1}{\gamma}$ and look for the roots of a polynomial of degree (s-1). The same holds true of the third polynomial, formally of degree 3s but in reality of degree (3s-1). Finally, the polynomial of the second condition, also formally of degree 3s, has $(\frac{1}{\gamma})^2$ as its lowest power; so in reality we will look for the positive real roots of a polynomial of degree (3s-2).

As an example, conditions (4.1.6) for p=s=2 are as follows:

$$-3y + 1.5y^2 \geq 0,$$

$$(4.1.7) \qquad 21y^2 - 36y^3 + 21.75y^4 - 5.5y^5 + 0.5y^6 \leq 0,$$

$$-6y + 24y^2 - 36y^3 + 21.75y^4 - 5.5y^5 + 0.5y^6 \geq 0,$$

where we have substituted $y = \frac{1}{\gamma}$. And in fact we would actually look for the zeroes of the following polynomials of course

$$-3 + 1.5y,$$

(4.1.8) $$21 - 36y + 21.75y^2 - 5.5y^3 + 0.5y^4,$$

$$-6 + 24y - 36y^2 + 21.75y^3 - 5.5y^4 + 0.5y^5.$$

With these details taken care of, the computation of the zeroes of all these polynomials yielded consistent results for the intervals of stability at infinity, with one exception which occurred for p=s=3 in the modified Richardson extrapolation approach. In this case, a very small interval which was not supposed to be stable at infinity turned out to be A(26.52°)-stable in later investigation! The issue was resolved by realizing that the 2 (numerically computed) very close roots which had "created" this supposedly non-stable interval were in fact a single double root.

All the γ-intervals for stability at infinity are shown in Appendix 2. For each interval, the bounds are given up to the tenth digit after the decimal point. The left bound was systematically rounded *up* and the right bound *truncated*, so that the bounds of each given interval are indeed part of the stable (at infinity) region.

Consistent with conditions (4.1.2) and (4.1.3) we see that the intervals of stability for Richardson extrapolation are more and more indistinguishable from those for the basic method, as p increases.

One odd result is obtained for modified Richardson extrapolation in the case p=s=2: we have an isolated point $\gamma=0.5$ for which we have stability at infinity (i.e. an interval of width 0).

4.2 General treatment for $A(\alpha)$-stability

Instead of requiring that the modulus of the stability functions be less than 1, we can define the absolute stability regions of the 3 methods by the equivalent conditions

$$(4.2.1) \qquad |Q(z;\gamma)|^2 - |P(z;\gamma)|^2 > 0\ ,$$

$$(4.2.2) \qquad |Q_{ex}(2z;\gamma)|^2 - |P_{ex}(2z;\gamma)|^2 > 0\ ,$$

$$(4.2.3) \qquad |Q_{mx}(2z;\gamma)|^2 - |P_{mx}(2z;\gamma)|^2 > 0\ .$$

Making use of (3.2.1) and (3.2.2), we can express the last 2 criteria only in terms of P and Q for the basic method. We obtain

$$(4.2.4) \quad (2^p - 1)^2 |Q^2(z)Q(2z)|^2 - |2^p P^2(z)Q(2z) - P(2z)Q^2(z)|^2 > 0\ ,$$

$$(4.2.5) \qquad (2^p\frac{C^*}{C} - 1)^2 |Q^2(z)|^2 - |2^p\frac{C^*}{C}P^2(z) - P^*(2z)Q(z)|^2 > 0\ ,$$

where we have dropped the dependancy on γ to shorten the notation and indicated the use of $\frac{\gamma}{2}$ instead of γ by a (*).

Given the definitions of $P(z;\gamma)$ and $Q(z;\gamma)$, the left-hand sides of (4.2.1), (4.2.4) and (4.2.5) are polynomials in z of degrees 2s, 6s and 4s respectively, whose coefficients are themselves polynomials in γ. Thus what is needed for computational purposes are routines for multiplying and subtracting polynomials as well as a routine to compute the square of the modulus of a polynomial (in a complex variable, with real coefficients). We now define a formula for this purpose.

Let $A(z) = \sum_{k=0}^{n} a_k z^k$, $a_k \in \mathbb{R}$, $z \in \mathbb{C}$, be a polynomial of degree n. Let $z = re^{i\varphi}$. We then obtain

$$(4.2.6) \qquad |A(z)|^2 = \sum_{k=0}^{2n} \left(\sum_{j=0}^{k} a_j a_{k-j} e^{i(2j-k)\varphi} \right) r^k ,$$

with the understanding that $a_i \equiv 0$, for $i > n$.

Now for $t = 0, 1, 2, \ldots \lfloor \frac{k-1}{2} \rfloor$, the terms for $j=t$ and $j=k-t$ in the internal summation can be combined together resulting in $2a_t a_{k-t} \cos(2t-k)\varphi$. And if k is even, there is an additional unpaired term which is $a_{k/2}^{\ 2}$. Thus we obtain

$$(4.2.7) \qquad \sum_{j=0}^{k} a_j a_{k-j} e^{i(2j-k)\varphi} = a_{k/2}^{\ 2} \delta^*(k) + 2 \sum_{j=0}^{\lfloor \frac{k-1}{2} \rfloor} a_j a_{k-j} \cos(2j-k)\varphi ,$$

where $\delta^*(k)$ is equal to 1 if k is even and is null otherwise, and where we still have $a_i \equiv 0$ for $i > n$.

This last provision can be built into the summation itself by changing its lower limit to $\max(0,k-n)$.

Finally, when investigating $A(\alpha)$-stability, we are interested in values of φ in $(\pi-\alpha, \pi+\alpha)$ for $\alpha \in [0, \frac{\pi}{2}]$. Replacing φ by $\pi \pm \alpha$ in (4.2.6) and (4.2.7) leads to the needed result:

(4.2.8)

$$|A(z)|^2 = \sum_{k=0}^{2n} \left(a_{k/2}^{\ 2} \delta^*(k) + 2(-1)^k \sum_{j=\max(0,k-n)}^{\lfloor \frac{k-1}{2} \rfloor} a_j a_{k-j} \cos(k-2j)\alpha \right) r^k ,$$

where the term for $k=0$ is simply $a_0^{\ 2}$.

Now back to our concern with A(α)-stability. Once γ is set for a particular method, how do we make sure that for a given α, the stability conditions (4.2.1), (4.2.4) and (4.2.5) hold for all $r \geq 0$ and for any complex z such that $|\arg(-z)| < \alpha$?

To help with this task, we can make use of the maximum modulus theorem for analytic functions (see page 134 of Ahlfors [2]) which states that if f(z) is analytic and non-constant in an (open) region, then |f(z)| has no maximum in this region. Conversely, if f(z) is analytic on a *closed* region, then |f(z)| attains its maximum *on the boundary* of the region.

Since R, R_{ex} and R_{mx} are analytic for $Re(z) \leq 0$, we conclude that we need only check if the stability condition is satisfied *on the boundary* of W_α to make sure that it is everywhere *in* W_α. Furthermore, since A(α)-stability in fact does not require the stability conditions to be met for $|\arg(-z)| = \alpha$ but we are verifying it for this case anyway, it follows that we can relax the conditions (4.2.1), (4.2.4) and (4.2.5); we can replace the strictly positive (>) requirement by a non-negative ($\geq$) requirement.

Now a close scrutiny of these stability conditions, making use of (4.2.8), reveals that these polynomials in r do not have a constant term (i.e. the coefficient for the term in r^0 is null). Also, their term in r^1 has coefficient $2\cos(\alpha)$ for (4.2.1), $4(2^p - 1)^2\cos(\alpha)$ for (4.2.4) and finally $4(2^p\frac{C^*}{C} - 1)^2\cos(\alpha)$ for (4.2.5).

Thus the conditions are satisfied for r=0, and for r>0 we can divide the inequalities by r. We shall call these reduced polynomials the ***stability polynomials*** for their corresponding method and denote them with S's. Taking into account what has been explained above, we will have $A(\alpha)$-stability if we satisfy the conditions:

(4.2.9) $$S(r,\alpha;\gamma) \geq 0$$

(4.2.10) $$S_{ex}(r,\alpha;\gamma) \geq 0$$

(4.2.11) $$S_{mx}(r,\alpha;\gamma) \geq 0 \ ,$$

where S, S_{ex} and S_{mx} are polynomials in r of degrees (2s-1), (6s-1) and (4s-1) respectively. They are the result of dividing our stability conditions by r and using (4.2.8) for the squares of moduli in these formulas.

As a concrete example, for p=s=2, the $A(\alpha)$-stability requirement for the basic method (4.2.9) becomes:

(4.2.12) $$(4\gamma^3 - 5\gamma^2 + 2\gamma - 0.25)r^3 + (10\gamma^2 - 6\gamma + 1)\cos(\alpha)r^2 + (4\gamma - 1)(1 + \cos(2\alpha))r + 2\cos(\alpha) \geq 0 \ .$$

For a given γ and α, the conditions to satisfy are just that certain polynomials in r be non-negative. How do we verify this?

Assuming simple roots, a necessary and sufficient condition is that the polynomials should be non-negative for r=0, and that there be no positive real roots. The first part is guaranteed for any $0 \leq \alpha < \frac{\pi}{2}$, given the already discussed values of the constant terms of S, S_{ex} and S_{mx}.

We can verify the second part by finding the *number* of positive real roots based on Sturm's theorem (see pages 261-263 of Cohn [19]). For this purpose, our programs made calls to subroutine "PA02B" of the Harwell library.

4.3 γ-regions for A(0)-stability

Intervals of A(0)-stability are computed by setting $\alpha=0$. For the case $p=s$, we investigate only inside the already computed intervals of stability at infinity, since this is a condition for A(0)-stability. For $p=(s-1)$, we must examine all of $(0.0,2.0]$ because we have stability at infinity for any $\gamma \in (0.0,\infty)$.

The numerical testing of (4.2.9), (4.2.10) and (4.2.11) inside the intervals of interest was carried out at equally spaced knots; while the distance between consecutive knots was often smaller, it was never more than 0.001. Thus no intervals of A(0)-stability of at least this width could be missed.

Bisection was used to find the boundaries of our A(0)-stable γ-regions with an accuracy of 10 digits after the decimal point. Computations were generally carried out in single precision, but many checks were done in double precision (mainly for the Richardson extrapolation method, where we have polynomials of higher degrees), with consistent results.

The computed intervals for all 3 methods can be found in Appendix 3. Note that when the upper bound for a γ-interval is 2.0, it does not imply lack of A(0)-stability for $\gamma>2.0$; simply, we only studied the interval $[0.0,2.0]$. Once again, we see that for $p\geq3$, the γ-intervals for Richardson extrapolation agree more and more closely with the ones for the basic method.

4.4 γ-regions for A-stability

For the special case $\alpha=\frac{\pi}{2}$ (90°), we can see that the constant terms of (4.2.9), (4.2.10) and (4.2.11), which we have described previously, are null. In fact, many other coefficients cancel out and this must be taken into account in the numerical computation.

From (4.2.8), we have that for $\alpha=\frac{\pi}{2}$, the square of the modulus of a polynomial A(z) will contain only even powers of r. Thus the left-hand sides of (4.2.1), (4.2.4) and (4.2.5) are polynomials in r with only even powers. But there is more.

From (3.2) it follows that

(4.4.1) $$P(z;\gamma) = e^{z}Q(z;\gamma) + O(z^{p+1}) .$$

Using the result $|B + C|^2 = |B|^2 + |C|^2 + 2\mathrm{Re}(B\overline{C})$, and $z=iy$ for the case $\alpha=\frac{\pi}{2}$, we arrive at

(4.4.2) $$|Q(iy;\gamma)|^2 - |P(iy;\gamma)|^2 = O(y^{p+1}) .$$

Thus we see that all powers of y which are less than (p+1) have null coefficients.

Given the above, the left-hand side polynomial of (4.4.2), sometimes called an E(y)-polynomial (see Nørsett [37]), can be written as:

(4.4.3) $$E(y) = \sum_{k=0}^{s} c_{2k}y^{2k}, \text{ with } e_{2k} = 0 \text{ for } 2k \leq p .$$

For Richardson extrapolation and modified Richardson extrapolation, the equivalent polynomials would be

(4.4.4) $$E_{ex}(y) = \sum_{k=0}^{3s} e'_{2k}y^{2k} ,$$

(4.4.5) $$E_{mx}(y) = \sum_{k=0}^{2s} e''_{2k} y^{2k} ,$$

with e'_{2k}, $e''_{2k} \geq 0$, for $2k \leq (p+1)$.

Evidently, these E-polynomials must be reduced as much as possible, and their null coefficients should be *set* to 0 if we want the numerical computation to proceed efficiently and stably. Then it is a matter of verifying the condition for $y=0$, and of making sure that there are no positive real roots.

Intervals of A-stability were computed, with the already established A(0)-stable intervals as the starting domain of investigation. The boundary values for the new intervals were obtained with the accurary already mentioned, again using a bisection scheme.

From the results, which are presented in Appendix 4, we see that Richardson extrapolation has no intervals of A-stability in [0.0,2.0], for $p = 2$, 4, 6 and 8 (where applicable) in both cases $p=s$ and $p=(s-1)$. The same holds for $p=s=7$, in Richardson extrapolation and the basic method; and for $p=s=8$, in the modified Richardson extrapolation method.

This is not as terrible as it seems however, because there are appreciable intervals where we nearly have A-stability. For example, for Richardson extrapolation in the case $p=s=4$, we have at least A(89.99°)-stability for $\gamma \in [0.40,1.15]$ (the boundaries of the interval are approximate). More examples are given in the next section.

Another interesting fact, if we require A- or L-stability, is that in some cases the modified Richardson extrapolation scheme satisfies this requirement when the basic scheme does not. As examples, we have the intervals $\gamma\in(1.068,2.000)$ for $p=s=3$, $\gamma\in(0.474,1.123)$ for $p=s=5$, and $\gamma\in(0.365,0.591)$ for $p=s=7$. Note that in this last case the basic method was nowhere A-stable in [0.0,2.0]. For $p=(s-1)$, we can find other examples for all values of p except $p=(s-1)=2$! This means that for linear problems, we could gain both in accuracy and in stability for certain values of γ, if we use the modified Richardson extrapolation method for continuation instead of the basic method.

In general, the γ-intervals obtained for A(0)- and A-stability agree with those published by Wanner [56], for the basic method, the case $p=s$ and γ in the interval [0.0.1.0]. But there are slight differences.

For $p=s=4$, the left bound of the A-stable interval is given by Wanner as 0.394339, whereas we would have 0.394338 to six decimals. Similarly, for $p=s=5$, Wanner gives the following A-stable regions: [0.246506, 0.361801] and [0.420785, 0.47328]. To the same accuracy, our own intervals are: [0.246506, 0.361803] and [0.420785, 0.47327].

Other differences exist for $p=s=6$ and $p=s=8$. In the first case, our values are [0.284065, 0.54091] and Wanner's [0.284065, 0.54090]. Finally, for $p=s=8$, the interval is given as [0.217005, 0.264716], whereas our own would be [0.217050, 0.264714]. In this last case, note that Wanner's A-stable interval agrees exactly with our second A(0)-stable interval; but we have found that while this interval is almost completely A-stable, it is not so in very small regions at the beginning and end of the interval.

In any case, these differences appear to be quite minor.

4.5 Numerical results for A(α)-stability

The approach concerning the computation of A(α)-stability was already presented in section 4.2.

The computation proceeds as follows: for each γ-region where we have successfully established A(0)-stability but not A-stability, we use a bisection algorithm (starting with the bracket $[0^\circ, 90^\circ]$) to find the *maximum* α for which we have A(α)-stability. The bisection is stopped when the bracket has been narrowed to a width of less than 0.01°. The largest value of α for which A(α)-stability holds, say α_0, is then truncated to 2 digits after the decimal point, resulting in a final value $\overline{\alpha}_0$.

Based on the method just described, we can be confident (neglecting rounding errors) that ~~the maximum α for~~ which we have A(α)-stability is such that

$$\alpha \in [\overline{\alpha}_0, \overline{\alpha}_0 + \epsilon(\overline{\alpha}_0)] \ . \tag{4.5.1}$$

In other words, we are certain to underestimate the maximum α, by an amount at most equal to $\epsilon(\overline{\alpha}_0)$.

To reach the accuracy mentioned above requires 14 bisection steps. Furthermore, it is easy to see that

$$\max \epsilon(\overline{\alpha}_0) \approx 0.0155 \ , \tag{4.5.2}$$

and this maximum was indeed seen to occur for $\overline{\alpha}_0 = 71.11^\circ$. More numerical computation also shows that the average value of $\epsilon(\overline{\alpha}_0)$ is:

$$\mathrm{Avg}\ [\epsilon(\overline{\alpha}_0)] \approx 0.0105 \ . \tag{4.5.3}$$

In this sense, the angles obtained are accurate within 0.01°.

The results of extensive computation for A(α)-stability are provided in Appendices 5 and 6, for the cases p=s and p=(s-1) respectively. For each value of p, a series of graphs are provided, plotting maximum α (in degrees) as a function of γ. In these plots, the *solid line* represents results for the basic method, the *dashed line* for Richardson extrapolation and the *dotted line* for modified Richardson extrapolation.

Each series of graphs starts with an overall picture of the results for $\gamma \in [0.0, 2.0]$ and ends with an enlargement of the top part of this first graph, where α is close to 90°. The other graphs are enlargements of particular (at least) A(0)-stable subregions, so that more fine detail can be captured visually.

From these plots, we see once again how the results for the basic method and for Richardson extrapolation are closer to one another, as p increases, until they are hardly distinguishable. Also the last graph of each series provides a good illustration of what we had already noted around A-stability: if we are aiming at maximum possible α, for a given value of γ, the modified Richardson extrapolation method can sometimes surpass the basic method. But we see that the gain is not very important.

We are now in a position to see that there are sometimes fairly large intervals which are almost A-stable, but not quite... In some cases, we even have A(89.99°)-stability and this does not seem to be due to accuracy problems in the computation. For instance, we obtain the same results in double precision as we do in single precision for many checks done on those intervals. The cause of the lack of A-stability is often the presence of a negative constant term in the deflated E(y)-polynomial; and this constant is not small enough (in absolute value) to blame it on limited accuracy of the numerical computation. Furthermore, we obtain the same negative constant term when we check in double precision.

p	Method	Interval	α_{max}
4	Richardson	(0.40, 1.15)	89.99
4	Modified	(0.98, 2.00)	89.98
6	Richardson	(0.285, 0.550)	89.99
6	Modified	(0.545, 1.35)	89.99
7	Basic	(0.32, 0.60)	89.99
7	Richardson	(0.32, 0.60)	89.99
7	Modified	(0.625, 1.572)	89.98
3	Richardson	(0.21715, 0.264)	89.99
8	Modified	(0.41. 0.675)	89.99

Table 4.5.1: Some nearly A-stable intervals (p=s)

Finally, to complete the information found in the various graphs, Tables 4.5.1 and 4.5.2 list certain intervals where we are very close to A-stability. The method and corresponding values of p are also given. However the boundaries of these intervals remain approximate as no effort was made to make them very sharp.

p	Method	Interval	α_{max}
2	Modified	(0.315, 0.333)	89.99
4	Richardson	(0.26, 0.65)	89.98
5	Modified	(0.57, 1.39)	89.97
6	Richardson	(0.21, 0.40)	89.98
6	Modified	(0.370, 0.504)	89.97
6	Modified	(0.66, 0.86)	89.95
7	Basic	(0.235, 0.45)	89.98
7	Richardson	(0.196, 0.211)	89.89
7	Richardson	(0.225, 0.40)	89.99
7	Modified	(0.277, 0.355)	89.99
7	Modified	(0.43, 0.69)	89.99

Table 4.5.2: Some nearly A-stable intervals (p=s-1)

CHAPTER 5

THE NONLINEAR PROBLEM

The criteria introduced in the first chapter, on the basis of the very simple scalar test equation (1.4.9), are appropriate to predict the stability behaviour of numerical integration methods when applied to stiff *linear* problems with *constant* coefficients.

Even if they have been used to predict the stability behaviour when dealing with general nonlinear systems, this is not without problems. Over the past decades, these difficulties have sparked intensive research into new model problems and stability criteria more suitable for the nonlinear case.

Our aim in the next chapter will be to investigate the stability of *semi-implicit methods* in particular. This chapter shall present a brief overview of new concepts — and related stability criteria — which represent some of the recent theoretical developments and will serve as background material for chapter 6.

5.1 Stiffness in the nonlinear case

In chapter 1, stiffness was defined for linear systems with constant coefficients (see definition 1.5.1), in terms of the eigenvalues of the constant matrix A, which determine the exact solution.

By analogy, for nonlinear systems and for linear systems with varying coefficients, stiffness is loosely described in terms of the eigenvalues of the Jacobian matrix

$$(5.1.1) \qquad f'(x,y) = \frac{\partial f(x,y)}{\partial y} .$$

If the eigenvalues of $f'(\tilde{x},y)$ satisfy the conditions of definition 1.5.1, we would then say that the problem is stiff at the point $x=\tilde{x}$; and if these eigenvalues multiplied by the stepsize lie inside the *linear* stability region of the method, as already defined, we would expect stable computation.

However there are serious problems with this definition extension. More precisely, it turns out that the eigenvalues of the "frozen" Jacobian give no valid information about the actual behaviour of the exact solution, not even for x arbitrarily close to $\tilde{x}$. An often quoted example (due to Vinograd; see Dekker and Verwer [24]) will illustrate this.

We consider the linear problem with varying coefficients* $y'(x) = A(x)y$, where A(x) is given by

$$(5.1.2) \quad A(x) = \begin{bmatrix} -1-9\cos^2(6x)+6\sin(12x) & 12\cos^2(6x)+\frac{9}{2}\sin(12x) \\ -12\sin^2(6x)+\frac{9}{2}\sin(12x) & -1-9\sin^2(6x)-6\sin(12x) \end{bmatrix} .$$

The 2 eigenvalues of this matrix are independant of x and have the

* From the presentation so far, it should be clear that linear problems with varying coefficients present difficulties similar to nonlinear problems.

values -1 and -10. On the other hand, the exact solution is

$$(5.1.3)\quad y(x) = C_1 e^{2x} \begin{bmatrix} \cos(6x)+2\sin(2x) \\ 2\cos(6x)-\sin(6x) \end{bmatrix} + C_2 e^{-13x} \begin{bmatrix} \sin(6x)-2\cos(6x) \\ 2\sin(6x)+\cos(6x) \end{bmatrix},$$

with C_1 and C_2 being arbitrary constants.

Clearly the exponentials e^{-x} and e^{-10x}, which are inferred from the "frozen" Jacobian for *any* value of x, are not present in the solution at all and do not capture its local behaviour anywhere! Therefore, outside of the class of linear problems with constant coefficients, the spectrum of the Jacobian matrix does not provide reliable information about stability and error propagation.

We have seen that an analogous extension of definition 1.5.1 is not satisfactory to define stiffness for nonlinear systems or even for linear systems with varying coefficients. However, this difficulty in finding a suitable mathematical definition does not mean that stiff problems are difficult to recognize *in practise*. For instance, problem (5.1.2) is mildly stiff as can be seen from its solution (5.1.3). Of course, an analytical expression for the solution is not usually available! But the stiff character of a problem will manifest itself by a dramatic inefficiency when we apply classical explicit methods to its solution.

5.2 One-sided Lipschitz constants and the logarithmic norm

Assume we perform numerical integration starting from the initial value $y(x_0) = y_0$. After one step, the computed solution will be y_1, which is only an approximation to the exact solution $y(x_0+h)$, because of the truncation error associated with the method used.

However, there is a solution branch of (1.1) which goes through the point y_1 and corresponds instead to a different starting value, say $y(x_0) = \tilde{y}_0$. From this it follows that one way of looking at the effect of the limited accuracy of a numerical integration method is to consider that the approximate solution is taking us from one exact solution branch to another "neighbouring" one, from step to step. Thus we are led to investigate what is the behaviour of these "neighbouring" solution branches: we want to learn how far away they could actually take us from the exact solution.

An important tool in this perturbation analysis of stiff nonlinear initial value problems is the *one-sided Lipschitz constant* which we will now introduce as in Dekker and Verwer [24], section 1.2.

We consider problems (1.1) whose right-hand side function satisfies the condition

$$(5.2.1) \qquad <f(x,y_1)-f(x,y_2),\ y_1-y_2> \ \leq \ \nu(x)\|y_1-y_2\|^2 ,$$

where $y_1,\ y_2 \in \mathbb{R}^m$ (assuming a system of m equations) and where $\|\ .\ \|$ is the norm corresponding to the inner product $<.\ ,\ .>$ defined on $\mathbb{R}^m$. Note that it is not necessary for (5.2.1) to hold for all possible $y_1,\ y_2$ in $\mathbb{R}^m$, as long as it is satisfied in a neighbourhood of the exact solution where the numerical solution will lie.

Now we look at the function

$$(5.2.2) \qquad \phi(x) = \|\tilde{y}(x) - y(x)\|^2 , \qquad x\in[x_0,x_f].$$

Here $y(x)$ and $\tilde{y}(x)$ are exact solutions of (1.1) corresponding to initial values y_0 and $\tilde{y}_0$ respectively.

Differentiating $\phi(x)$ and using (5.2.1), we obtain

(5.2.3) $$\phi'(x) = 2<\tilde{y}'(x)-y'(x), \tilde{y}(x)-y(x)> \leq 2\nu(x)\phi(x).$$

Multiplying both sides by $\exp(-2\int_0^x \nu(z)dz)$ leads to the inequality

(5.2.4) $$\frac{d}{dx}\left(\phi(x)\exp(-2\int_0^x \nu(z)dz)\right) \leq 0.$$

The function in brackets being monotonically decreasing, we finally can write, for $x_0 \leq x_1 \leq x_2 \leq x_f$

(5.2.5) $$\|\tilde{y}(x_2) - y(x_2)\| \leq \|\tilde{y}(x_1) - y(x_1)\| \exp(\int_{x_1}^{x_2} \nu(z)dz).$$

Here we see the importance of the function $\nu(x)$ in condition (5.2.1) in that it determines the evolution of the "distance" between 2 solution branches of (1.1). This function is called the *one-sided Lipschitz constant* for $\mathbf{f}(x,y)$ and (5.2.1) is a *one-sided Lipschitz condition.* Note that $\nu(x)$ depends on the particular inner product chosen.

It is easy to show that if $\mathbf{f}(x,y)$ satisfies a classical Lipschitz condition (1.2.3) then the classical Lipschitz constant L is also a one-sided Lipschitz constant. Using it, (5.2.5) would become

(5.2.6) $$\|\tilde{y}(x_2) - y(x_2)\| \leq \|\tilde{y}(x_1) - y(x_1)\| \exp(L(x_2 - x_1)).$$

Remembering that, for stiff problems, L is very large, this is definitely not a very informative (or reassuring!) bound on the evolution of the difference between 2 solutions.

However, for many problems, one-sided Lipschitz constants can be found which are much smaller than L. Many stiff problems even have $\nu(x) \leq 0$. In this case, (5.2.5) results in

(5.2.7) $$\|\tilde{y}(x_2) - y(x_2)\| \leq \|\tilde{y}(x_1) - y(x_1)\| .$$

Such problems and their associated vector function $\mathbf{f}(x,y)$ are then called *dissipative.*

Loosely speaking, (5.2.7) means that the "distance" between 2 solution branches with different starting values remains bounded by their initial "distance". Thus the solution branches do not get further and further apart as x increases. This is important for the accuracy of the numerical approximation due to truncation errors. If $\nu(x)<0$, the various solution branches* get closer and closer as x increases.

Clearly we are interested in the smallest possible Lipschitz constant for (5.2.5) to provide a tighter bound on the difference of 2 solutions.

For linear problems where $f(x,y) = Ay$, with A a constant matrix, it can be shown that the smallest possible Lipschitz constant $\mu[A]$ is given by

(5.2.8) $$\mu[A] = \lim_{\delta \to 0^+} \frac{\|I + \delta A\| - 1}{\delta} .$$

The right-hand side limit is called the *logarithmic matrix norm* of A.

So far we have only considered inner product norms. However Dahlquist has proven that (5.2.5) holds for any norm provided that $\mu\left(\frac{\partial f(x,y)}{\partial y}\right) \le \nu(x)$.

We end this section with some concrete expressions for the logarithmic norm $\mu[A]$ of a matrix $A = (a_{ij})$ in the usual norms $\|\cdot\|_1$, $\|\cdot\|_2$ and $\|\cdot\|_\infty$:

$$\mu_1[A] = \max_j \left(a_{jj} + \sum_{i \neq j} |a_{ij}|\right),$$

* As mentioned previously, this is possibly restricted to a certain region of the m-dimensional solution space, around the exact solution sought.

(5.2.9)
$$\mu_2[A] = \lambda_{\max}\left[\frac{A + A^T}{2}\right],$$
$$\mu_\infty[A] = \max_i \left(a_{ii} + \sum_{j\neq i} |a_{ij}|\right).$$

5.3 Contractivity and A-stability

We have seen in the previous section that some systems of stiff ordinary differential equations are *dissipative*, a condition which should be favorable to numerical integration, given the inevitable truncation error.

An important question in the study of numerical methods for ordinary differential equations is to find out whether and in what conditions these methods yield results (computed solutions) which exhibit a similar contractive behaviour. The reason is simple: in practise, from step to step, the computed solution carries some rounding error in addition to the truncation error; thus we are not integrating along a single numerical solution branch either and we have to worry about the perturbation sensitivity of the numerical formulas as well.

We now proceed to give some definitions in this regard.

Definition 5.3.1 The one-step method (1.3.1) is called *contractive* on a class $\mathcal{F}$ of stiff initial value problems if for all problems in $\mathcal{F}$ and *any* stepsize $h\in(0,h_0]$, we have

(5.3.1) $$\|\tilde{y}_{n+1} - y_{n+1}\| \leq \kappa\|\tilde{y}_n - y_n\|, \quad 0<\kappa\leq 1,$$

where κ and h_0 are independant of the stiffness of problems in $\mathcal{F}$. If $\kappa<1$, we have *strict contractivity.*

Definition 5.3.2 The one-step method (1.3.1) is called *unconditionnally contractive* on $\mathcal{F}$ if (5.3.1) holds for all problems in $\mathcal{F}$ and any positive stepsize, with κ independant of the stiffness of problems in $\mathcal{F}$.

A few comments are in order. First, we note the key importance of κ and h_0 being independant of stiffness. We want methods which will offer favorable conditions for stable integration of stiff problems, without imposing severe restrictions on the stepsizes to be used. If, for example, we have contractivity for $h \in (0,h_0]$, where h_0 is inversely proportional to the classical Lipschitz constant of the problem, this is of no practical use for integrating stiff problems!

As presented by Dekker and Verwer [24], a class $\mathcal{F}$ is specified in 3 ways: first, by the *type of vector function* $f(x,y)$ considered (linear with constant coefficients, with varying coefficients etc...); secondly, by a particular upper bound function for the logarithmic norm of this vector function (i.e. $\mu[f'(x,y)] \leq \nu(x)$); and lastly by the particular norm used to compute the logarithmic norm. In all cases, $\nu(x)$ is assumed to be of reasonable size, whereas the classical Lipschitz constants of problems in $\mathcal{F}$ may be arbitrarily large.

At this point, we consider the relationship between A-stability and unconditional contractivity. First, we note that the scalar test equation (1.4.9) is *dissipative* for any complex λ with non-positive real part since it obeys

(5.3.2) $\|\tilde{y}(x_2) - y(x_2)\| = \|\tilde{y}(x_1) - y(x_1)\| \exp(\mathrm{Re}(\lambda)(x_2-x_1)).$

Furthermore, as we saw in section 1.4, A-stability is characterized by the numerical method yielding $|y_{n+1}| < |y_n|$ for all λ in the left half of the complex plane. Thus we could define A-stability as a requirement of *unconditional contractivity* when methods are applied to *dissipative* scalar test equations (1.4.9). In fact, one can generalize this characterization as in the following theorem.

Theorem 5.3.3 A Runge-Kutta method is A-stable iff the method is unconditionally contractive in the Euclidian norm on the class of dissipative, constant coefficient problems $y' = Ay$, with normal matrix A.

That unconditional contractivity on the specified class of problems is a necessary condition was in essence already demonstrated at the end of section 2.6. That it is a sufficient condition can be seen by forming, for any $\lambda \in \mathbb{C}$ with $\mathrm{Re}(\lambda) \leq 0$, the following matrix

$$(5.3.3) \qquad A = \begin{bmatrix} \mathrm{Re}(\lambda) & -\mathrm{Im}(\lambda) \\ \mathrm{Im}(\lambda) & \mathrm{Re}(\lambda) \end{bmatrix}.$$

This matrix is normal, with eigenvalues λ and $\bar{\lambda}$, and its associated problem $y' = Ay$ is dissipative. Since

$$(5.3.4) \qquad \|\tilde{y}_{n+1} - y_{n+1}\|_2 \leq \|R(hA)\|_2 \|\tilde{y}_n - y_n\|_2 ,$$

unconditional contractivity requires that $\|R(hA)\|_2 \leq 1$. But R(hA) is also normal with eigenvalues $R(h\lambda)$ and $R(h\bar{\lambda}) = \overline{R(h\lambda)}$. Thus $\|R(hA)\|_2 = |R(z)|$ and we have A-stability.

Equation (5.3.4) is actually valid in any norm for a Runge-Kutta method applied to $y' = Ay$. In the investigation of numerical contractivity, we are thus interested in finding some good upper bound for $\|R(hA)\|$ in an arbitrary norm. To this end, the following theorem (see Hairer, Bader, Lubich [29]) is very useful*.

Theorem 5.3.4 Let $\| \cdot \|$ be any inner product norm and A a given matrix. Let R(z) be a given rational function. Then for all positive h we have

$$\|R(hA)\| \leq \varphi_R(h\mu[A]),$$

where $\mu[A]$ is the logarithmic norm of A and φ_R is defined as

$$\varphi_R(x) = \sup_{\mathrm{Re}(z) \leq x} |R(z)|.$$

From this theorem it immediately follows that for an A-stable method and a problem with $\mu[A]<0$, we have $\|R(hA)\|<1$ for any inner product norm. This result is broader than that of theorem 5.3.3 as it is not limited to normal matrices nor to the Euclidian norm. The following property of the function φ_R, also proven by Hairer, Bader and Lubich, will be needed later.

Theorem 5.3.5 Let R(z) be an A-stable rational approximation to e^z of order at least 1 and let $|R(iy)|<1$ for $y\neq0$ and $|R(\infty)|<1$. Then we have $\varphi_R(x) = 1 + x + o(x)$, for $x\to0$.

5.4 Some nonlinear stability criteria for Runge-Kutta methods

In this section, we focus our attention on Runge-Kutta methods when applied to dissipative initial value problems (1.1), i.e. problems which satisfy

* In most cases, norms considered in practise will be inner product norms.

(5.4.1) $$<f(x,y_1)-f(x,y_2),\ y_1-y_2> \ \leq 0, \quad x\in[x_0,x_f],$$

for some real inner product.

Definition 5.4.1 (Butcher [12]) A Runge-Kutta method (2.1.1) is called *B-stable* if for all autonomous* problems satisfying (5.4.1) and for any positive stepsize h we have

(5.4.2) $$\|\tilde{y}_{n+1} - y_{n+1}\| \leq \|\tilde{y}_n - y_n\|.$$

Definition 5.4.2 (Burrage, Butcher [11]; Crouzeix [20]) A Runge-Kutta method (2.1.1) is called *BN-stable*, if for all problems satisfying (5.4.1) and any positive stepsize h, (5.4.2) holds.

Notice that these definitions mean *unconditional* contractivity for dissipative problems in some inner product norm. Since the class of problems covered by definition (5.4.2) encompasses the class for the preceding definition, it follows that BN-stability $\Rightarrow$ B-stability. Furthermore, both these properties imply A-stability as the latter requires unconditional contractivity on a narrower class of problems than the one needed for B-stability.

Burrage and Butcher [11] and Crouzeix [20] independantly discovered a suitable characterization of BN-stable methods which involves the matrix $M = BA + A^TB - bb^T$, where $A = (a_{ij})$ and $b^T = (b_1, b_2, \dots b_s)$ are taken from the Butcher tableau and $B = \mathrm{diag}(b_1, b_2, \dots b_s)$. Their results are summarized as follows:

* The problem (1.1) is autonomous if $f(x,y) = f(y)$, i.e. f does not explicitly depend on the independant variable.

Definition 5.4.3 If a Runge-Kutta method is such that B and M are positive semi-definite, the method is said to be ***algebraically stable.***

Theorem 5.4.4 If a method is algebraically stable, then it is BN-stable.

Note that for B to be positive semi-definite, we must have $b_i \geq 0$ for each i = 1, 2, ... s. And for irreducible methods, we must have strictly $b_i > 0$. We now mention some algebraic stability results regarding methods already introduced in chapter 2.

Butcher's [14] s-stage methods of order 2s (corresponding to Gauss-Legendre quadrature formulas) as well as some methods of order (2s-1) and (2s-2) are algebraically stable.

The 2-stage SDIRK methods with Butcher tableau given as

$$(5.4.3)\qquad \begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & 1-2\gamma & \gamma \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

are algebraically stable for any $\gamma \geq \frac{1}{4}$ (Note that for $\gamma < \frac{1}{4}$ we have seen that we do not even have stability at infinity). This includes the optimum order 3 method (corresponding to $\gamma = \hat{\gamma}_1 \approx 0.78868$ as given in table 3.1.2).

Also, the 3-stage SDIRK method of optimal order 4 is algebraically stable for $\gamma = \hat{\gamma}_1 \approx 1.06858$. In fact, Hairer [28] has proven the more general result that the highest possible order for an algebraically stable DIRK method is 4.

Verwer [55] has shown that the simple Rosenbrock scheme corresponding to (2.5.1) with $\gamma_{ij}=0$ and $A = J(x_n+c_1h,y_n)$ is not algebraically stable. In this paper, Verwer also makes the important point that methods having the BN-stability property may in fact lose it as soon as they are implemented in practise. He demonstrates this with SDIRK methods. The reason for this paradox lies in the schemes used to resolve the implicitness of these methods.

When BN-stability properties are established, they are based on the assumption that the numerical solution is the *exact* solution of the implicit sets of equations to be solved at each step. In practise however, we must use some iterative procedure to solve these equations *approximately*. Is the *practical* solution thus obtained unconditionally contractive when applied to any dissipative problem? Or else are limitations on the stepsize required to ensure convergence of the iterative scheme for at least some problems of the class considered?

Verwer focuses his attention on the modified Newton procedure (2.5.3) with precisely 1 Jacobian evaluation per step at the point (x_n+c_1h,y_n) and provides counter-examples to show that the property of BN-stability is then lost in practise.

Even if one wishes to consider 1 Jacobian evaluation per stage or even per Newton iteration — a situation which would make the method highly inefficient — Verwer conjectures that other counter-examples could be found.

Despite this result which clearly invites caution, one should not be too pessimistic because these counter-examples are not necessarily typical of results obtained in practise. Experience reveals that usually Newton-type iterative schemes, used with implicit methods for solving stiff nonlinear problems, show a satisfactory convergence behaviour, with a reasonable stepsize and within a few iterations.

5.5 D-stability

The concept of D-stability is not a stability property like A-, B- or BN-stability. It is not concerned with the propagation of error and stepwise stability but rather with the *boundedness* of the numerical solution over a *single step* when applied to a certain class of problems which can be of arbitrary stiffness (infinitely stiff, in the limit).

D-stability is concerned with linear variable coefficient problems of the form

(5.5.1) $y'(x) = A(x)y(x), \qquad x\in[x_0,x_f], \quad y(x_0) = y_0.$

When applied to this problem, one-step integration methods yield

(5.5.2) $$y_{n+1} = M(x_n,h)y_n.$$

Definition 5.5.1 (van Veldhuisen [52]) An integration method resulting in (5.5.2) when applied to a particular class $\mathcal{F}$ of problems of type (5.5.1) is called *D($\mathcal{F}$)-stable* if for all problems in $\mathcal{F}$ and all $x_n\in[x_0,x_f]$, we have $\|M(x_n,h)\|\leq\tilde{M}<\infty$ for any stepsize $h\in(0,h_0]$.

Note that h_0 and $\tilde{M}$ must exist independant of stiffness of the problems in $\mathcal{F}$ and that $\tilde{M}$ is also independant of stepsize. As we are only concerned with boundedness, the choice of the norm does not matter and is free.

Next we present the particular problem class S considered by Van Veldhuizen when he introduced his D-stability concept.

Definition 5.5.2 The problem class S includes all 2-dimensional linear systems (5.5.1) with real solutions, and such that matrix $A(x) = E(x)D(x)E^{-1}(x)$ with $D(x) = \text{diag}\left(\frac{d_1(x)}{\epsilon}\ ,\ d_2(x)\right)$ where $\epsilon\in(0,\epsilon_0)$ is a small parameter and $\text{Re}(d_1(x))\le\tilde{d}_1<0$ for $x\in[x_0,x_f]$. Furthermore, d_1, d_2, E and E^{-1} must depend smoothly on the independant variable and possibly on ϵ too.

Any problem in S can be written as

(5.5.3) $$y'(x) = \epsilon^{-1}\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} y(x),$$

where $a_{ij} = a_{ij}(x;\epsilon)$ depends smoothly on x and ϵ.

Now let v(x) be defined by the transformation

(5.5.4) $$v(x) = E^{-1}(x)y(x).$$

Differentiating $y(x) = E(x)v(x)$ and multiplying by $E^{-1}(x)$ from the left, we obtain:

(5.5.5) $$v'(x) = E^{-1}(x)y'(x) - E^{-1}(x)E'(x)v(x).$$

Now replacing $y'(x)$ by its defining expression for class S and using (5.5.4) again we end up with

(5.5.6) $$v'(x) = (D(x) - C(x))v(x),$$

where $C(x) = E^{-1}(x)E'(x)$.

If $C(x)$ is also a diagonal matrix, the system has been uncoupled by the chosen transformation, the first component of $\mathbf{v}(x)$ being the *transient* (or stiff) component while the other one is the *smooth* component. Otherwise, a coupling exists between the 2 components.

From the above, we can see that S has some desirable properties to serve as a model class to investigate the application of numerical methods to stiff systems. It allows the coexistence of stiff and smooth components, while the parameter ϵ can be used to make the problem arbitrarily stiff; furthermore, problems in S have a variable eigensystem.

It can be shown that for any problem in S, the *exact* solution $\mathbf{y}(x)$ satisfies

(5.5.7) $$\|\mathbf{y}(x)\|_\infty \leq g(x;\epsilon), \qquad x\in[x_0,x_f],$$

where g in continuous and *bounded* for $\epsilon\in(0,\epsilon_0]$. Thus D($S$)-stability can be interpreted as a requirement that stepsizes $h\in(0,h_0]$ exist such that the *numerical* solution satisfy a similar boundedness property.

Now consider the matrix $C = (c_{ij})$ of equation (5.5.6), where $c_{ij} = c_{ij}(x;\epsilon)$. The off-diagonal elements c_{12} and c_{21} represent respectively the coupling from the smooth component to the transient (stiff) component and the coupling from the transient component to the smooth component. If these coefficients behave as $O(\epsilon)$ as $\epsilon\rightarrow 0$, we say that the coupling is *weak*; otherwise, we have *strong* coupling.

Definition 5.5.3 We define $\mathcal{W}_{st}$ and $\mathcal{W}_{ts}$ as 2 subclasses of $\mathcal{S}$ consisting respectively of all problems of $\mathcal{S}$ with a weak coupling from the smooth to the transient component, and all problems of $\mathcal{S}$ with a weak coupling from the transient to the smooth component.

We note that the kind of coupling between components is invariant under transformations which are ϵ- and x-independant; but for x-(and possibly ϵ-) dependant transformations, the nature of the coupling betwen components can change. We now present some results about D-stability of methods (2.5.1) where $A = J(x_n,y_n)$, i.e. Rosenbrock-type methods.

The first order, one-stage method of this kind is D($\mathcal{S}$)-stable. However, for two-stage, second order methods of the same type with positive γ, one can only prove that it is D($\mathcal{W}_{ts}$)-stable; that it is D($\mathcal{W}_{st}$)-stable iff $R_2(\infty)=0$, where R_2 is an internal stability function (see section 2.6); that it cannot be D($\mathcal{S}$)-stable. The combined proof of these results is fairly long and can be found in Dekker and Verwer [24], pages 239-244.

Verwer [54] investigated a property similar to D-stability which he called *ϵ-boundedness*, defined on the basis on a different model class but where ϵ is also a parameter allowing transition to arbitrary stiffness. In this paper, he gives some D-stability results as well.

Trying to gain insight into the performance of Rosenbrock-type methods when applied to *nonlinear* stiff problems, he concludes that to cope with time-varying eigensystems, we should have $R(\infty)=0$ and $R_1(\infty)=0$. Despite this, if both couplings are strong, he proves that one can only guarantee D($\mathcal{S}$)-stability if the Rosenbrock methods re-evaluate the Jacobian at every stage*. The property is lost if we evaluate J once per *step* only, as is the case for (2.5.1) with A = $J(x_n,y_n)$ and in most implementations, for obvious efficiency reasons.

To conclude this section, we note that it is difficult to relate D-stability to stepwise stability properties. While D-stability is concerned with stable behaviour over a single step in the transition to infinite stiffness, stepwise stability properties are concerned with accumulation of errors over many steps. Furthermore, the classes of problems which D-stability and BN-stability, for example, deal with are very different and no easy relationship can be shown among these model classes.

In a specific context — that of perturbed collocation methods — Van Veldhuizen [53] has shown that A-stability combined with D-stability is a slightly weaker property than BN-stability.

* The methods originally proposed by H. H. Rosenbrock did this.

CHAPTER 6

CONTRACTIVITY OF SEMI-IMPLICIT METHODS

The material of the first 2 sections of this chapter is drawn from Hairer, Bader, Lubich [29].

6.1 Applying semi-implicit methods to nonlinear problems

For the stability analysis of semi-implicit methods, we will rewrite the nonlinear initial value problem as

(6.1.1) $$y' = f(x,y) = Ay + g(x,y) ,$$

where A is the arbitrary matrix of method (2.5.1), which will be chosen as an approximation to the Jacobian of f at the point from which we want to advance integration.

In this chapter, we make the assumption that the nonlinear function g satisfies a Lipschitz condition

(6.1.2) $$\|g(x,y) - g(x,\tilde{y})\| \le L\|y - \tilde{y}\|,$$

where the Lipschitz constant L is *small*, in the sense that it does not depend on the stiffness of the problem. This assumption is reasonable given the choice of A mentioned above.

A series expansion of $f(x,y)$ about the point (x_n,y_n) yields

(6.1.3) $$f(x,y) = f(x_n,y_n) + \frac{\partial f(x_n,y_n)}{\partial x}(x-x_n) + \frac{\partial f(x_n,y_n)}{\partial y}(y-y_n) + \ldots$$

Choosing $A = \dfrac{\partial f(x_n,y_n)}{\partial y}$, we have that $\dfrac{\partial g(x_n,y_n)}{\partial y} = 0$. Since we can choose $L = \sup \|\frac{\partial g}{\partial y}\|$ over the domain on which we want (6.1.2) to hold, it is then reasonable to assume that L will be small if we remain sufficiently close to (x_n,y_n). If $A \approx \dfrac{\partial f(x_n,y_n)}{\partial y}$, again we require that L not be too large; if it is not small enough, the computed approximation will be bad which should be indicated by a large truncation error estimate and a new A would be computed.

Theorem 6.1.1 If we apply the semi-implicit method (2.5.1) to the differential equation (6.1.1) we obtain

(6.1.4) $$\begin{aligned} y_{n+1} &= R(hA)y_n + hG(hA,x_n,y_n,h) \\ u_i &= R_i(hA)y_n + hG_i(hA,x_n,y_n,h) , \end{aligned}$$

where $R(z)$ and $R_i(z)$ are the stability function and internal stability functions defined in section 2.6. Also G and G_i are given by

(6.1.5) $$\begin{aligned} G(hA,x_n,y_n,h) &= S\sum_{j=1}^{s}\left(\sum_{l=j}^{s} b_l v_{lj}(hA)\right)g_j \\ G_i(hA,x_n,y_n,h) &= S\sum_{j=1}^{i-1}\left(\sum_{l=j}^{i-1} \alpha_{il} v_{lj}(hA)\right)g_j , \end{aligned}$$

where we have used $S = (I - \gamma hA)^{-1}$ and $g_j = g(x_n+c_jh, u_j)$ and the functions v_{lj} are defined recursively (using $\beta_{lj} = \alpha_{lj} + \gamma_{lj}$) as

(6.1.6) $$\begin{aligned} v_{jj}(z) &= 1 , \qquad 1\le j\le s, \\ v_{lj}(z) &= \sum_{s=j}^{l-1}\beta_{ls}\left(\frac{z}{1-\gamma z}\right)v_{sj}(z) , \qquad 1\le j<l\le s. \end{aligned}$$

We now prove this theorem.

Applying (2.5.1) to problem (6.1.1) and defining $W = hSA$, we obtain:

$$k_i = W\left(y_n + \sum_{j=1}^{i-1}\beta_{ij}k_j\right) + hSg_i. \tag{6.1.7}$$

Note the similarity of formula (2.6.2) to the first term making up $\mathbf{k}_i$. Clearly, the solution of $\mathbf{y}' = A\mathbf{y}$ would yield only this term and it accounts for the presence of $R(hA)$ and $R_i(hA)$ in $\mathbf{y}_{n+1}$ and $\mathbf{u}_i$ respectively.

If we use (6.1.7) to find the first few $\mathbf{k}_i$'s, we obtain

$$\begin{aligned} k_1 &= Wy_n + hSg_1 \\ k_2 &= W(I + \beta_{21}W)y_n + hS(\beta_{21}Wg_1 + g_2) \\ k_3 &= W(I + (\beta_{31}+\beta_{32})W + \beta_{32}\beta_{21}W^2)y_n + \\ &\qquad hS((\beta_{31}+\beta_{32}\beta_{21}W)Wg_1 + \beta_{32}Wg_2 + g_3). \end{aligned} \tag{6.1.8}$$

We next define recursively the functions $v_{lj}(z)$ as in (6.1.6) and the functions $r_{lj}(z)$ as follows:

$$\begin{aligned} r_{jj}(z) &= \frac{z}{1-\gamma z}, \qquad 1\le j\le s \\ r_{lj}(z) &= \sum_{s=j}^{l-1}\beta_{ls}\left(\frac{z}{1-\gamma z}\right)r_{sj}(z), \qquad 1\le j<l\le s. \end{aligned} \tag{6.1.9}$$

Using these definitions, one can easily show by induction that

$$k_l = \left(\sum_{j=1}^{l} r_{lj}(hA)\right)y_n + hS\sum_{j=1}^{l} v_{lj}(hA)g_j. \tag{6.1.10}$$

Incorporating (6.1.10) into the definition of $\mathbf{u}_i$ in (2.5.1), we obtain our first result:

$$\begin{aligned} u_i &= \left(1+\sum_{l=1}^{i-1}\sum_{j=1}^{l}\alpha_{il}r_{lj}(hA)\right)y_n + hS\sum_{j=1}^{i-1}\sum_{l=j}^{i-1}\alpha_{il}v_{lj}(hA)g_j \\ &= R_i(hA)y_n + hG_i(hA,x_n,y_n,h). \end{aligned} \tag{6.1.11}$$

Likewise, we obtain for y_{n+1}

$$y_{n+1} = \left(1+\sum_{l=1}^{s} b_l \sum_{j=1}^{l} r_{lj}(hA)\right) y_n + hS \sum_{j=1}^{s} \sum_{l=j}^{s} b_l v_{lj}(hA) g_j$$

(6.1.12) $$= R(hA) + hG(hA, x_n, y_n, h).$$

6.2 Conditions for contractivity

We consider problems (6.1.1) where A and g satisfy the following conditions

(6.2.1) $$<Ay,y> \leq \mu\|y\|^2 \quad \text{and}$$
$$\|g(x,\tilde{y}) - g(x,y)\| \leq L\|\tilde{y} - y\|,$$

for all $x \in [x_0, x_f]$ and all y, $\tilde{y} \in \mathbb{R}^m$.

Then, if $y(x)$ and $\tilde{y}(x)$ are 2 *exact* solutions corresponding to different starting values, we can prove the following:

(6.2.2) $$\|\tilde{y}(x+h) - y(x+h)\| \leq \|\tilde{y}(x) - y(x)\| e^{(\mu+L)h}.$$

The proof is along the same lines as the one already given in section 5.2 to obtain a similar bound result.

Clearly, the problem is dissipative for $\mu+L \leq 0$. Now we are interested in discovering under what conditions the *numerical* solution will share this contractive behaviour. To this end, assume that there exists a constant $C>0$ such that for every problem (6.1.1) satisfying (6.2.1) with $\mu+\omega L \leq 0$ — for a certain real value ω — we can use any stepsize $0 \leq h \leq \frac{C}{L}$ and obtain a contractive numerical solution by the semi-implicit method in the sense that

(6.2.3) $$\|\tilde{y}_{n+1} - y_{n+1}\| \leq \|\tilde{y}_n - y_n\|.$$

Thus we are aiming for some *conditional* contractivity criteria and we see the importance of our initial assumption that L should be of reasonable size and not depend on the stiffness of the problem.

Definition 6.2.1 We define S_ω as the set of all real ω for which conditional contractivity as just described can be established. Also, we define $\omega^* = \inf S_\omega$.

If $\hat{\omega} \in S_\omega$, then the same holds for all $\omega > \hat{\omega}$ (the same constant C would clearly do the job). Thus S_ω is either empty or contains an infinite real interval. Obviously, we would like ω^* to be as small as possible, since then for a given μ we can tolerate a larger L and thus can possibly use a larger stepsize.

Hairer, Bader and Lubich [29] proved that any semi-implicit method satisfying $S_\omega \neq \emptyset$ is necessarily A-stable and has $\omega^* \geq 1$. Thus A-stability is a minimum requirement for obtaining conditional contractivity as previously defined. The same paper also states that for methods where the $c_i = \sum_{j=1}^{i-1} \alpha_{ij}$ are all distinct, we must have $|R(\infty)| < 1$ to obtain $S_\omega \neq \emptyset$.

Using $G \equiv G(hA, x_n, y_n, h)$ and $\tilde{G} \equiv G(hA, x_n, \tilde{y}_n, h)$, we now make use of theorem (6.1.1) to obtain

$$\begin{aligned} \|\tilde{y}_{n+1} - y_{n+1}\| &\leq \|R(hA)\| \|\tilde{y}_n - y_n\| + h\|\tilde{G} - G\|, \\ &\leq \kappa \|\tilde{y}_n - y_n\|, \end{aligned} \tag{6.2.4}$$

where we have defined $\kappa \equiv \|R(hA)\| + hL_G$, L_G being a classical Lipschitz constant for G with respect to y.

Furthermore, for convenient presentation of future results, we adopt the following additional definitions and short notations:

$$B_j(z) = \sum_{l=j}^{s} \frac{b_l}{1-\gamma z} v_{lj}(z),$$

(6.2.5)
$$A_{ij}(z) = \sum_{l=j}^{i-1} \frac{\alpha_{il}}{1-\gamma z} v_{lj}(z),$$

$$\tilde{G}_i = G_i(hA,x_n,\tilde{y}_n,h),$$

$$G_i = G_i(hA,x_n,y_n,h).$$

Following Hairer, Bader and Lubich, we now proceed to obtain a computationnally available estimate for ω^*.

Theorem 6.2.2 For all problems (6.1.1) satisfying (6.2.1), we have

(6.2.6)
$$\|\tilde{G} - G\| \leq L\omega(h\mu,hL)\|\tilde{y}_n - y_n\| \quad \text{and}$$

(6.2.7)
$$\|\tilde{G}_i - G_i\| \leq L\omega_i(h\mu,hL)\|\tilde{y}_n - y_n\|,$$

where

$$\omega_1(h\mu,hL) = 0,$$

$$\omega_i(h\mu,hL) = \sum_{j=1}^{i-1} \overline{A}_{ij}(h\mu)W_j(h\mu,hL),$$

$$\omega(h\mu,hL) = \sum_{j=1}^{s} \overline{B}_j(h\mu)W_j(h\mu,hL),$$

and

$$W_j(h\mu,hL) = \varphi_j(h\mu) + hL\omega_j(h\mu,hL),$$

$$\overline{A}_{ij}(x) = \sup_{Re(z)\leq x} |A_{ij}(z)|,$$

$$\overline{B}_j(x) = \sup_{Re(z)\leq x} |B_j(z)|,$$

$$\varphi_j(x) = \sup_{Re(z)\leq x} |R_j(z)|.$$

In order to prove this, we use (6.2.5) to write

$$(6.2.8) \qquad \|\tilde{G}_i - G_i\| \le \sum_{j=1}^{i-1} \|A_{ij}(hA)\| \|g_j(x_n+c_jh,\tilde{u}_j) - g_j(x_n+c_jh,u_j)\|.$$

Now using the Lipschitz condition on g and (6.1.4) for u_j, we have:

$$\|G_j(\tilde{u}_j)-G_j(u_j)\| \le L\|R_j(hA)(\tilde{y}_n - y_n) + h(\tilde{G}_j - G_j)\|,$$

$$(6.2.9) \qquad \le L(\varphi_j(h\mu)\|\tilde{y}_n - y_n\| + h\|\tilde{G}_j - G_j\|),$$

where we have used theorem 5.3.4 to obtain the last inequality.

Using (6.2.9) and $\|A_{ij}(hA)\| \le \overline{A}_{ij}(h\mu)$ — again a consequence of theorem 5.3.4 — , (6.2.8) becomes

$$(6.2.10) \quad \|\tilde{G}_i - G_i\| \le L\sum_{j=1}^{i-1} \overline{A}_{ij}(h\mu)(\varphi_j(h\mu)\|\tilde{y}_n - y_n\| + h\|\tilde{G}_j-G_j\|).$$

Now (6.2.7) obviously holds for i=1; then using recursion (6.2.10), we can easily prove by induction that (6.2.7) is valid for any i>1 also. And we can prove (6.2.6) in exactly the same manner.

Theorem 6.2.3 Let R(z) be the stability function of method (2.5.1) and assume that $\varphi_R(x) = 1 + x + o(x)$, for $x \to 0^-$ (for sufficient conditions, see theorem 5.3.5), then $S_\omega \neq \emptyset$ and

$$(6.2.11) \qquad \omega^* \le \omega_0 = \omega(0,0) = \sum_{j=1}^{s} \overline{B}_j(0)\varphi_j(0).$$

We can prove this by first using theorems 5.3.4 and 6.2.2 to rewrite equation (6.2.4) as

$$(6.2.12) \qquad \|\tilde{y}_{n+1} - y_{n+1}\| \le (\varphi_R(h\mu) + hL\omega(h\mu,hL))\|\tilde{y}_n - y_n\|.$$

Now we assume that $\mu+(\omega_0+\epsilon)L \le 0$ and prove that $\omega_0+\epsilon \in S_\omega$ for all positive ϵ. By the maximum modulus theorem, we know that φ_R is monotonic increasing, as well as $\overline{B}_j(h\mu)$ and $\varphi_j(h\mu)$; thus the

$\omega_j(h\mu,hL)$ and finally $\omega(h\mu,hL)$ are monotonic functions of their first argument. Then it follows that

(6.2.13) $\quad \varphi_R(h\mu) \leq \varphi_R(-(\omega_0+\epsilon)hL) = 1 - (\omega_0+\epsilon)hL + o(hL),$

(6.2.14) $\quad \omega(h\mu,hL) \leq \omega(-(\omega_0+\epsilon)hL,hL) = \omega_0 + o(1).$

Consequently, we have

(6.2.15) $\quad \varphi_R(h\mu) + hL\omega(h\mu,hL) \leq 1 - \epsilon hL + o(hL).$

This guarantees the existence of $C>0$ (which may depend on ϵ) such that (6.2.3) holds for all $h\in(0,\frac{C}{L})$.

Finally, Hairer, Bader and Lubich establish the following result concerning the conditions required for a method (2.5.1) to have the optimal value $\omega^*=1$. We give this without proof.

Theorem 6.2.4 Let the semi-implicit method (2.5.1) satisfy the conditions of theorem 5.3.5. Then $\omega_0=1$ if, for $j = 1, 2, \ldots, s$, we have

a) $b_j\geq 0$ and $|B_j(z)|\leq b_j$ for $Re(z)\leq 0$,

b) if $B_j(z)\neq 0$ then $|R_j(z)|\leq 1$ for $Re(z)\leq 0$.

Furthermore, if the method has distinct coefficients $c_i = \sum_{j=1}^{i-1}\alpha_{ij}$ and satisfies the conditions of theorem 5.3.5, then $\omega^*=1$ implies $b_j\geq 0$ for all j.

6.3 A simple case: the one-stage method

The one-stage method is presented as an example in Hairer, Bader and Lubich but their treatment is not quite complete. This method is given by

(6.3.1) $$(I - \gamma hA)k_1 = hf(x_n, y_n),$$
$$y_{n+1} = y_n + k_1.$$

We know that it is A-stable for $\gamma \geq \frac{1}{2}$; however, for nonlinear contractivity, it is necessary to have $|R(\infty)| = \frac{|1-\gamma|}{\gamma} < 1$. For this, we must have $\gamma > \frac{1}{2}$.

Since $B_1(z) = \frac{b_1}{1-\gamma z}$, we easily arrive at the following result, using $b_1=1$, $\varphi_1(h\mu)=1$ and the fact that $\mu \leq 0$:

(6.3.2) $$\omega(h\mu, hL) = \overline{B}_1(h\mu) = \frac{1}{1-\gamma h\mu}.$$

Now we must find $\varphi_R(h\mu)$. By definition, $\varphi_R(x) = \sup_{Re(z)\leq x} |R(z)|$. Since $h\mu<0$ and $R(z)$ is analytic in the left-half of the complex plane, we can once again invoke the maximum modulus theorem to write $\varphi_R(x) = \sup_{Re(z)=x} |R(z)|$.

The stability function for this simple method is

(6.3.3) $$R(z) = \frac{1 + (1-\gamma)z}{1-\gamma z}.$$

With $z = x + iy$, we have

(6.3.4) $$|R(z)|^2 = \frac{(1 + (1-\gamma)x)^2 + (1-\gamma)^2 y^2}{(1-\gamma x)^2 + \gamma^2 y^2}.$$

This is of the form $\frac{a+by^2}{c+dy^2}$, with $a,b,c,d > 0$.

Differentiating this form with respect to y, we obtain $\frac{2(bc-ad)y}{(c+dy^2)^2}$. We now examine all possible cases.

Case1: bc-ad$\leq$0.

Here $|R(z)|^2$ is monotone decreasing and thus maximum for $y=0$. We obtain

(6.3.5) $$\varphi_R(h\mu) = \frac{|1 + (1-\gamma)h\mu|}{1-\gamma h\mu}.$$

And the condition bc-ad$\leq$0 boils down to

(6.3.6) $$h\mu(\gamma-1) \leq \frac{2\gamma - 1}{2\gamma}.$$

This is always satisfied for $\gamma\geq1$ and *any* positive stepsize h; otherwise, we must have

(6.3.7) $$h\mu \geq \frac{2\gamma - 1}{2\gamma(\gamma - 1)}.$$

It is easily verified that when (6.3.6) holds, the absolute value in (6.3.5) is redundant. For this case, (6.3.2) and (6.3.5) finally yield:

(6.3.8) $$\|\tilde{y}_{n+1} - y_{n+1}\| \leq \left(\frac{1 + (1-\gamma)h\mu + hL}{1-\gamma h\mu}\right)\|\tilde{y}_n - y_n\|.$$

This holds for the case $\gamma\geq1$, and also for $\frac{1}{2}<\gamma<1$, with additional condition (6.3.7). And contractivity then follows simply if $\mu+L \leq 0$.

Case 2: bc-ad>0.

Now $|R(z)|^2$ is monotome increasing. And obviously, this can only happen for $\frac{1}{2}<\gamma<1$ and $h\mu < \frac{2\gamma - 1}{2\gamma(\gamma - 1)}$. In this case,

(6.3.9) $$\varphi_R(h\mu) = \lim_{y\to\infty} |R(h\mu+iy)| = \frac{1-\gamma}{\gamma},$$

so that we arrive at

(6.3.10) $$\|\tilde{y}_{n+1} - y_{n+1}\| \leq \left(\frac{1-\gamma}{\gamma} + \frac{hL}{1-\gamma h\mu}\right)\|\tilde{y}_n - y_n\|.$$

Thus we will have contractivity for $\frac{hL}{1-\gamma h\mu} \leq \frac{2\gamma-1}{\gamma}$, which can be rewritten as

(6.3.11) $$h(\mu + (\frac{1}{2\gamma-1})L) \leq \frac{1}{\gamma}.$$

Here we note that if μ is negative enough, i.e. $\mu + (\frac{1}{2\gamma-1})L \leq 0$, (6.3.11) will again hold *for any positive stepsize h*. Otherwise, this inequality will impose an upper bound on the stepsize.

We can summarize the analysis of contractivity for this simple method in the following 3 points:

1. For $\gamma \geq 1$, we have contractivity for any positive stepsize, if $\mu + L \leq 0$;
2. For $\frac{1}{2} < \gamma < 1$, we have contractivity for any positive stepsize, if $\mu + (\frac{1}{2\gamma-1})L \leq 0$;
3. For $\frac{1}{2} < \gamma < 1$, and $\mu + (\frac{1}{2\gamma-1})L > 0$, we have contractivity for $h \leq \frac{1}{\gamma(\mu + (\frac{1}{2\gamma-1})L)}$, if $\mu + L \leq 0$.

We see that $\mu + L \leq 0$ is a necessary condition in all cases.

These results are somewhat tighter that those presented by Hairer, Bader and Lubich. While they mention the first point above, they do not mention the second point at all. Furthermore, their conclusion for the case $\frac{1}{2} < \gamma \leq 1$ is that we have contractivity whenever $\mu + L \leq 0$ and $hL \leq \frac{2\gamma - 1}{2\gamma(1 - \gamma)}$. This is certainly true; and for the special case $\mu + L = 0$, it identical with our third point. However for $\mu + L < 0$ and $\mu + (\frac{1}{2\gamma-1})L > 0$, it unnecessarily imposes a more stringent restriction on the stepsize than the one we have above. In other words, it is a sufficient condition for our third point to hold but not a necessary one.

It is important to realize that the first 2 summary points above do not amount to BN-stability, despite the "for any positive stepsize"—statement. The reason is simple: for many problems, the conditions $\mu+L \le 0$ or $\mu + (\frac{1}{2\gamma-1})L \le 0$ will not be satisfied if the stepsize is too large. This point was explained at the beginning of section 6.1.

6.4 Estimating ω^* for higher order semi-implicit methods

A program has been developed that computes ω_0, an estimate for ω^*, based on theorem 6.2.3. It is *assumed* that methods whose defining coefficients are provided as input do satisfy the conditions of this theorem: no verification of this is performed.

The program (see Appendix 7) applies to methods with up to 4 stages. For this purpose, the following expressions were obtained, in which s is the number of stages and $\beta_{ij} \equiv \alpha_{ij} + \gamma_{ij}$:

(6.4.1) $$B_s(z) = \frac{b_s}{1-\gamma z} ;$$

(6.4.2) $$B_{s-1}(z) = \frac{b_{s-1} + (b_s\beta_{s,s-1}-\gamma b_{s-1})z}{(1-\gamma z)^2} ;$$

(6.4.3) $$B_{s-2}(z) = (1-\gamma z)^{-3}(A + (B-2\gamma A)z + (A\gamma^2-B\gamma+C)z^2), \quad \text{where}$$
$$A \equiv b_{s-2} ,$$
$$B \equiv b_{s-1}\beta_{s-1,s-2} + b_s\beta_{s,s-2} ,$$
$$C \equiv b_s\beta_{s,s-1}\beta_{s-1,s-2} ;$$

(6.4.4) $$B_{s-3}(z) = (1-\gamma z)^{-4}(A + (B-3A\gamma)z + (C-2B\gamma+3A\gamma^2)z^2 + (D-C\gamma+B\gamma^2-A\gamma^3)z^3), \quad \text{where}$$

$$A \equiv b_{s-3},$$
$$B \equiv b_{s-2}\beta_{s-2,s-3} + b_{s-1}\beta_{s-1,s-3} + b_s\beta_{s,s-3},$$
$$C \equiv b_{s-1}\beta_{s-1,s-2}\beta_{s-2,s-3} + b_s(\beta_{s,s-1}\beta_{s-1,s-3} + \beta_{s,s-2}\beta_{s-2,s-3}),$$
$$D \equiv b_s\beta_{s,s-1}\beta_{s-1,s-2}\beta_{s-2,s-3}.$$

These expressions were derived from (6.2.5) and the recursive definition (6.1.6) for the v_{lj}'s. In fact, some closed form formulas can be found which are useful in obtaining the previous results. More particularly, we have:

(6.4.5) $$v_{lj}(z) = \sum_{i=1}^{l-j} \left(\frac{z}{1-\gamma z}\right)^i \sum_{l>k_1>k_2>\ldots>k_{i-1}>j} \beta_{lk_1}\beta_{k_1k_2}\cdots\beta_{k_{i-1}j},$$

for $1\leq j<l\leq s$,

(6.4.6) $$B_j(z) = \frac{1}{1-\gamma z}\left(b_j + \sum_{i=1}^{s-j}\left(\frac{z}{1-\gamma z}\right)^i \sum_{k_1>k_2>\ldots>k_i>j} b_{k_1}\beta_{k_1k_2}\beta_{k_2k_3}\cdots\beta_{k_ij}\right),$$

for $j = 1, 2, \ldots s$.

Note that in the internal summations of (6.4.5) and (6.4.6), each term contains exactly "i" β—factors.

In order to compute ω_0, we also require expressions for the internal stability functions. These are obtained from (2.6.9):

(6.4.7) $$R_1(z) = 1,$$

(6.4.8) $$R_2(z) = \frac{1 + (c_2-\gamma)z}{1-\gamma z},$$

(6.4.9) $$R_3(z) = \frac{1 + (c_3-2\gamma)z + (\alpha_{32}\beta_2-c_3\gamma+\gamma^2)z^2}{(1-\gamma z)^2},$$

(6.4.10) $$R_4(z) = \frac{1 + (c_4-3\gamma)z + Az^2 + Bz^3}{(1-\gamma z)^3},$$ where

$$A \equiv (\alpha_{42}\beta_2+\alpha_{43}\beta_3) - 2c_4\gamma + 3\gamma^2,$$

$$B \equiv \alpha_{43}\beta_{32}\beta_2 - (\alpha_{42}\beta_2+\alpha_{43}\beta_3)\gamma + c_4\gamma^2 - \gamma^3.$$

Here, we have used $c_i = \sum_{j=1}^{i-1}\alpha_{ij}$ and also $\beta_i \equiv \sum_{j=1}^{i-1}\beta_{ij}$.

From theorem 6.2.3, we have $\omega_0 = \sum_{j=1}^{s}\overline{B}_j(0)\varphi_j(0)$. By the maximum modulus theorem, we can write

(6.4.11) $\quad \overline{B}_j(0) = \sup_{Re(z)\leq 0} |B_j(z)| = \sup_{y\geq 0} |B_j(iy)|$ and

(6.4.12) $\quad \varphi_j(0) = \sup_{Re(z)\leq 0} |R_j(z)| = \sup_{y\geq 0} |R_y(z)|$.

From (6.4.1) through (6.4.4), expressions were obtained for $|B_j(iy)|^2$ and likewise for $|R_j(iy)|^2$ from (6.4.7) through (6.4.10). The suprema can then be found by differentiating these expressions with respect to y, setting the derivatives to zero and finding the roots*.

The program was used to estimate ω^* for 4 semi-implicit methods. First, a second order, two-stage method from Steihaug and Wolfbrandt [50]. In their paper, they use it as their basic integration method in a so-called (2,4)-W embedded pair of formulas. We shall denote this method W2. The other three methods are third order, four-stage methods from Kaps [31], also used as basic integration methods. They are called W32M1V1, W32M1V3 and W3X1. All four methods are L-stable and meet the requirements for theorem 6.2.3 to apply.

* In the program, roots of the resulting polynomials are found by calls to the subroutine "ZPOLR" from the IMSL library.

	W2	W3X1	W32M1V3	W32M1V1
γ:	$1 - \frac{\sqrt{2}}{2}$	0.55185066	0.56	$\frac{1}{2}$
b_1:	0.25	0.25	0.25	$-\frac{1}{12}$
b_2:	0.75	0.00	0.00	$\frac{1}{6}$
b_3:	—	0.00	0.00	$\frac{2}{3}$
b_4:	—	0.75	0.75	$\frac{1}{4}$
α_{21}:	$\frac{2}{3}$	0.55185066	0.56	$\frac{1}{2}$
α_{31}:	—	0.9374015826	0.9500454133	0
α_{32}:	—	0.0625984174	0.0499545867	$\frac{1}{4}$
α_{41}:	—	0.4536780949	0.4579242536	$\frac{1}{4}$
α_{42}:	—	-0.0200039587	-0.0306359298	$\frac{7}{12}$
α_{43}:	—	0.2335925304	0.2393783429	$\frac{1}{6}$
γ_{21}:	$-\frac{4\gamma}{3}$	3.3989123070	4.3724800000	-1
γ_{31}:	—	-1.0875681763	-0.8500454133	$\frac{5}{4}$
γ_{32}:	—	-0.1875984174	-0.1499545867	$\frac{1}{4}$
γ_{41}:	—	-0.3610096565	-0.3773297003	$-\frac{29}{12}$
γ_{42}:	—	-0.0153760876	-0.0090826521	$-\frac{29}{12}$
γ_{43}:	—	-0.3504151359	-0.3784196185	$-\frac{1}{2}$

Table 6.4.1: Coefficients for the 4 methods

In table 6.4.1, we give all the defining coefficients for these methods. In Table 6.4.2, we give the program estimates for ω^* for all 4 methods, as well as the values of $\varphi_j(0)$ and $\overline{B}_j(0)$ in each case. We

	W2	W3X1	W32M1V3	W32M1V1
ω_0:	1.2301086827	1.1487694492	1.1451488320	1.4583621068
$\varphi_1(0)$:	1.0	1.0	1.0	1.0
$\varphi_2(0)$:	1.2761423749	1.0	1.0	1.0
$\varphi_3(0)$:	1.0	1.0	1.0	1.0062305899
$\varphi_1(0)$:	1.0	1.0	1.0	1.0
$\overline{B}_1(0)$:	0.2730019015	0.2767134467	0.2767715315	0.3111753049
$\overline{B}_2(0)$:	0.75	0.0365555574	0.0252693034	0.2263664086
$\overline{B}_3(0)$:	----	0.0855004451	0.0931079971	$\frac{2}{3}$
$\overline{B}_4(0)$:	----	0.75	0.75	0.25

Table 6.4.2: Computed values for ω_0, $\varphi_j(0)$ and $\overline{B}_j(0)$

notice that while none of the 4 methods enjoys the optimal value $\omega_0=1$, the estimates obtained are fairly small, especially for W32M1V3 and W3X1 which are both *internally* L-stable.

Inspection of both tables against the sufficient conditions for $\omega_0=1$, as presented in theorem 6.2.4, reveals where each method departs from the optimal case. In particular, we see that the presence of a negative $b_1 = -\frac{1}{12}$ for W32M1V1 is costly: this method obtains the largest ω_0 as a direct consequence of this, despite the fact that 3 of its 4 internal stability functions are L-stable (and $R_3(z)$ does not miss by much either!).

6.5 Maximum stepsize for contractivity

In the previous section, we found estimates for ω^*, the smallest ω such that the methods investigated are *contractive* when applied to problems (6.1.1) satisfying conditions (6.2.1) with $\mu+\omega L \leq 0$.

While this tells us that stepsizes do exist for which these methods are contractive, it does not tell us what these stepsizes are nor if they are reasonably large. In this section, we try to answer these questions for the 4 particular methods already presented.

We must find for what range of stepsizes we have

(6.5.1) $$\|\tilde{y}_{n+1} - y_{n+1}\| \leq \kappa\|\tilde{y}_n - y_n\|,$$

where $0<\kappa\leq 1$. From (6.2.12), we have a readily available estimate for κ namely

(6.5.2) $$\kappa = \varphi_R(h\mu) + hL\omega(h\mu,hL).$$

The program developed computes values for κ and finds — for given μ and L — the maximum stepsize h for which $\kappa\leq 1$. Formulas (6.4.1) through (6.4.4) and (6.4.7) through (6.4.10) are of course important ingredients in the making of this program. Also, we must have formulas for the stability functions and for the functions $A_{ij}(z)$. The stability functions of order 1, 2, 3 and 4 respectively are obtained using (2.6.10) and (2.6.14):

(6.5.3) $$R_{11}(z) = \frac{1 + (1-\gamma)z}{1-\gamma z},$$

(6.5.4) $$R_{22}(z) = \frac{1 + (1-2\gamma)z + (\frac{1}{2} - 2\gamma+\gamma^2)z^2}{(1-\gamma z)^2},$$

(6.5.5) $$R_{33}(z) = (1-\gamma z)^{-3}(1 + (1-3\gamma)z + (\tfrac{1}{2} - 3\gamma+3\gamma^2)z^2 + (\tfrac{1}{6} - \tfrac{3}{2}\gamma+3\gamma^2-\gamma^3)z^3,$$

(6.5.6) $$R_{44}(z) = (1-\gamma z)^{-4}(1 + (1-4\gamma)z + (\tfrac{1}{2} - 4\gamma+6\gamma^2)z^2 + (\tfrac{1}{6} - 2\gamma+6\gamma^2-4\gamma^3)z^3 + (\tfrac{1}{24} - \tfrac{2}{3}\gamma+3\gamma^2-4\gamma^3+\gamma^4)z^4 .$$

Also formulas for $A_{ij}(z)$ which are sufficient for our purposes are as follows:

(6.5.7) $$A_{i,i-1}(z) = \frac{\alpha_{i,i-1}}{1-\gamma z},$$

(6.5.8) $$A_{i,i-2}(z) = \frac{\alpha_{i,i-2} + (\alpha_{i,i-1}\beta_{i-1,i-2} - \alpha_{i,i-2}\gamma)z}{(1-\gamma z)^2},$$

(6.5.9) $$A_{i,i-3}(z) = \frac{\alpha_{i,i-3} + (C-2\alpha_{i,i-3}\gamma)z + (D-C\gamma+\alpha_{i,i-3}\gamma^2)z^2}{(1-\gamma z)^3},$$

where

$$C = \alpha_{i,i-2}\beta_{i-2,i-3} + \alpha_{i,i-1}\beta_{i-1,i-3}$$
$$D = \alpha_{i,i-1}\beta_{i-1,i-2}\beta_{i-2,i-3} .$$

The following closed form formula was useful in deriving the previous expressions:

(6.5.10) $$A_{ij}(z) = \frac{1}{1-\gamma z}\left[\alpha_{ij} + \sum_{l=1}^{i-j-1}\left(\frac{z}{1-\gamma z}\right)^l \sum_{i>k_1>\ldots>k_l>j} \alpha_{ik_1}\beta_{k_1k_2}\beta_{k_2k_3}\cdots\beta_{k_lj}\right],$$

where each term of the internal summation has exactly one α—factor and "l" β—factors.

We note that all stability functions and internal stability functions, as well as all A- and B-functions are rational functions with their denominator being some power of $(1-\gamma z)$. Now the program must be able to compute the supremum of the modulus of these complex functions along a line $\mathrm{Re}(z)=x$ (here we implicitly make use of the maximum modulus theorem, since all these functions are analytic in the domain of interest). Thus an important tool needed is a formula to compute the norm of a polynomial in a complex variable with real coefficients. We now derive such a formula.

Let $p(z) = \sum_{k=0}^{n} a_k z^k$, $a_k \in \mathbb{R}$, $z \in \mathbb{C}$, be a polynomial of degree n. Using $z^k = (x+iy)^k$, we have:

(6.5.11) $$p(z) = \sum_{k=0}^{n} a_k \sum_{j=0}^{k} \binom{k}{j} x^j (iy)^{k-j}.$$

Now let $k-j=l$ and note that $\binom{k}{j} = \binom{k}{k-j} = \binom{k}{l}$. We then obtain

(6.5.12) $$p(z) = \sum_{k=0}^{n} a_k \sum_{l=0}^{k} \binom{k}{l} x^{k-l} (iy)^l.$$

The next step is to split the complex p(z) into a real part and an imaginary part. Then it will be easy to compute $|p(z)|^2$. After some manipulations, we end up with the following expression:

(6.5.13) $$p(z) = \sum_{j=0}^{\lfloor\frac{n}{2}\rfloor}\left((-1)^j\sum_{k=2j}^{n}a_k\binom{k}{2j}x^{k-2j}\right)(y^2)^j + iy\sum_{j=0}^{\lfloor\frac{n-1}{2}\rfloor}\left((-1)^j\sum_{k=2j+1}^{n}a_k\binom{k}{2j+1}x^{k-2j-1}\right)(y^2)^j.$$

Since all functions of interest are of the form $f(z) = \frac{p(z)}{(1-\gamma z)^q}$, we will have

(6.5.14) $$|f(z)|^2 = \frac{|p(z)|^2}{((1-\gamma x)^2 + \gamma^2 y^2)^q}.$$

If $p(z)$ is of degree n, then $|p(z)|^2$ will be a polynomial in y^2 of degree n also. Thus, for any given x, we can easily find the values of y at which $|f(z)|$ reaches its extrema.

From (6.5.2) and theorem 6.2.2, we see that for a certain value of the ratio $(-L/\mu)$, κ depends only on the product $h\mu$. Consequently, instead of computing the maximum stepsize as such, the program will compute the maximum value of the product $(-h\mu)$ allowed for contractivity. This is investigated for values of $(-h\mu)$ in the range $[10^{-8},10^{8}]$. A first bisection determines the appropriate power of 10 and a second bisection determines 3 significant digits for the maximum $(-h\mu)$ allowed.

The numerical investigation was carried out for equally spaced values of the ratio $(-L/\mu)$, using an increment of 0.02, up to the maximum of $\frac{1}{\omega_0}$. Beyond this value, we know that contractivity is not possible.

For each of the 4 methods described previously, it turned out that for "small" values of the ratio $(-L/\mu)$, we obtained contractivity $(\kappa \le 1)$ even for the largest $h\mu = 10^8$. We therefore conclude that, as was the case for the simple one-stage method in section 6.3, there exists a smallest value ω_{inf} such that, if $\mu + \omega_{inf} L \le 0$ is satisfied, we obtain contractivity for *any* stepsize*. Of course, $\omega_{inf} \ge \omega_0$.

For the one-stage method, we can summarize our results from section 6.3 as follows

(6.5.15) $$\omega_{inf} = \max\left(1, \frac{1}{2\gamma-1}\right), \qquad \gamma > \frac{1}{2}.$$

	W2	W3X1	W32M1V3	W32M1V1
ω_0:	1.23011	1.14877	1.14515	1.45830
ω_{inf}:	5.82843	1.90250	1.68271	2.75453

Table 6.5.1: Computed ω_0 and ω_{inf}

* Remember however that $\mu + \omega_{inf} L \le 0$ may often impose limitations on the stepsize. What is meant is that there is no additional restriction on the stepsize.

To take the above results into account, the final version of the program (see Appendix 8) first estimates ω_0 and ω_{inf} and then computes maximum values of $(-h\mu)$ for contractivity, for equally spaced values of the ratio $(-L/\mu)$ in the interval $(\frac{1}{\omega_{inf}}, \frac{1}{\omega_0})$.

The results are presented in Tables 6.5.1 and 6.5.2.

From Table 6.5.1, we see that W3X1 and W32M1V3 obtain the best results for ω_{inf}, with values smaller than 2. While W32M1V1 was the worst method in terms of ω_0, it does much better than W2 for its ω_{inf} value, obtaining 2.75453 against 5.82843.

Table 6.5.2 presents the maximum value of $(-h\mu)$ allowed for contractivity for selected values of the ratio $(-L/\mu)$. For example, if $\mu = -1$, the leftmost column then gives the values of L and other entries in the table provide the corresponding maximum *stepsize* for contractivity.

The respective effect of ω_0 and ω_{inf} are quite visible in this table. We also see that methods W32M1V3 and W3X1 generally allow maximum stepsizes much larger than W32M1V1 and W2. In the range $(-L/\mu) \in [0.54, 0.70]$, W32M1V3 is even significantly better than than W3X1, but for larger values of the ratio, the difference between both methods is less significant. Finally, W32M1V1 surpasses W2 in the whole range $(-L/\mu \in (0, 0.68)$, but beyond this point only W2 is contractive.

$(-L/\mu)$	W2	W3X1	W32M1V3	W32M1V1
0.12	.	.	.	.
0.14	.	.	.	.
0.16	>.100E+0	>.100E+9	>.100E+9	>.100E+9
0.18	.249E+2	.	.	.
0.20	.111E+2	.	.	.
0.22	.694E+1	.	.	.
0.24	.485E+1	.	.	.
0.26	.365E+1	.	.	.
0.28	.287E+1	.	.	.
0.30	.232E+1	.	.	.
0.32	.193E+1	.	.	.
0.34	.162E+1	.	.	.
0.36	.138E+1	.	.	>.100E+9
0.38	.110E+1	.	.	.958E+1
0.40	.103E+1	.	.	.595E+1
0.42	.905E+0	.	.	.428E+1
0.44	.793E+0	.	.	.325E+1
0.46	.695E+0	.	.	.256E+1
0.48	.612E+0	.	.	.209E+1
0.50	.538E+0	.	.	.176E+1
0.52	.473E+0	>.100E+9	.	.150E+1
0.54	.415E+0	.653E+2	.	.129E+1
0.56	.364E+0	.243E+2	.	.112E+1
0.58	.317E+0	.159E+2	>.100E+9	.971E+0
0.60	.275E+0	.114E+2	.148E+3	.836E+0
0.62	.237E+0	.848E+1	.347E+2	.709E+0
0.64	.203E+0	.648E+1	.207E+2	.582E+0
0.66	.172E+0	.515E+1	.145E+2	.440E+0
0.68	.143E+0	.423E+1	.105E+2	.229E+0
0.70	.110E+0	.356E+1	.774E+1	<.100E-7

Table 6.5.2: Maximum $(-h\mu)$ for given $(-L/\mu)$

$-\frac{L}{\mu}$	W2	W3X1	W32M1V3	W32M1V1
0.72	.927E-1	.306E+1	.548E+1	<.100E-7
0.74	.690E-1	.266E+1	.442E+1	.
0.76	.489E-1	.236E+1	.347E+1	.
0.78	.293E-1	.210E+1	.277E+1	.
0.80	.111E-1	.188E+1	.226E+1	.
0.82	<.100E-7	.165E+1	.187E+1	.
0.84	.	.141E+1	.155E+1	.
0.86	.	.120E+1	.125E+1	.
0.88	.	<.100E-7	<.100E-7	.
0.90	.	.	.	.
0.92	.	.	.	.
	.	.	.	.

Table 6.5.2 (cont.): Maximum $(-h\mu)$ for given $(-L/\mu)$

REFERENCES

1. Abramovitz, M., Stegun, I.A., *Handbook of mathematical functions.* New-York: Dover Publications Inc., (1970).
2. Ahlfors, L.V., *Complex Analysis.* New York: McGraw-Hill Book Company, 2nd edition, (1966).
3. Aiken, R.C. (ed.), *Stiff Computation.* New-York: Oxford University Press, (1985).
4. Alexander, R., *Diagonally implicit Runge-Kutta methods for stiff ODE's.* SIAM Journal of Numerical Analysis, 14, 1006-1021, (1977).
5. Bickart, T.A., *An efficient solution process for implicit Runge-Kutta methods.* SIAM Journal of Numerical Analysis, 14, 1022-1027, (1977).
6. Bui, T.D., *A note on the Rosenbrock procedure.* Mathematics of Computation, 33, 971-975, (1979).
7. Bui, T.D., *Some A-stable and L-stable methods for the numerical integration of stiff ordinary differential equations.* Journal of the ACM, 26, 483-493, (1979).
8. Bui, T.D., Oppenheim, A.K., Pratt, D.T., *Recent advances in methods for numerical solutions of O.D.E. initial value problems.* Journal of Computational and Applied Mathematics, 11, 283-296, (1984).

9. Bui, T.D., Poon, S.W.H., *On the computational aspects of Rosenbrock procedures with built-in error estimates for stiff systems.* BIT, 21, 168-174, (1981).

10. Burrage, K., *A special family of Runge-Kutta methods for solving stiff differential equations.* BIT, 18, 22-41, (1978).

11. Burrage, K., Butcher, J.C., *Stability criteria for implicit Runge-Kutta methods.* SIAM Journal of Numerical Analysis, 16, 46-57, (1979).

12. Butcher, J.C., *A stability property of implicit Runge-Kutta methods.* BIT, 15, 358-361, (1975).

13. Butcher, J.C., *Coefficients for the study of Runge-Kutta integration processes.* Journal of the Australian Mathematical Society, 3, 185-201, (1963).

14. Butcher, J.C., *Implicit Runge-Kutta processes.* Mathematics of Computation, 18, 50-64, (1964).

15. Butcher, J.C., *On the attainable order of Runge-Kutta methods.* Mathematics of Computation, 19, 408-417, (1965).

16. Butcher, J.C., *On the implementation of implicit Runge-Kutta methods.* BIT, 16, 237-240, (1976).

17. Butcher, J.C., *The non-existence of ten stage eighth order explicit Runge-Kutta methods.* BIT, 25, 521-540, (1985).

18. Cash, J.R., *Semi-implicit Runge-Kutta procedures with error estimates for the numerical integration of stiff systems of ordinary differential equations.* Journal of the ACM, 23, 455-460, (1976).

19. Cohn, P.M., *Algebra (vol. 2).* New-York: John Wiley & Sons, (1974).

20. Crouzeix, M., *Sur la B-stabilité des méthodes de Runge-Kutta.* Numerische Mathematik, 32, 75-82, (1979).

21. Crouzeix, M., Ruamps, F., *On rational approximations to the exponential.* R.A.I.R.O., Analyse numérique, 11, 241-243, (1977).

22. Dahlquist, G.G., *A special stability problem for linear multistep methods.* BIT, 3, 27-43, (1963).

23. Dahlquist, G.G., *Convergence and stability in the numerical integration of ordinary differential equations.* Mathematica Scandinavica, 4, 33-53, (1956).

24. Dekker, K., Verwer, J.G., *Stability of Runge-Kutta methods for stiff nonlinear differential equations.* Amsterdam: North Holland, (1984).

25. Ehle, B.L., *A-stable methods and Padé approximations to the exponential.* SIAM Journal of Mathematical Analysis, 4, 671-680, (1973).

26. Ehle, B.L., *On Padé approximations to the exponential function and A-stable methods for the numerical solution of initial value prolems.* University of Waterloo Dept. of Applied Analysis and Computer Science, Research Report No. CSRR2010, (1969).

27. Gupta, G.K., Sacks-Davis, R., Tischer, P.E., *A review of recent developments in solving ODEs.* ACM Computing Surveys, 17, 5-47, (1985).

28. Hairer, E., *Highest possible order of algebraically stable diagonally implicit Runge-Kutta methods.* BIT, 20, 254-256, (1980).

29. Hairer, E., Bader, G., Lubich, Ch., *On the stability of semi-implicit methods for ordinary differential equations.* BIT, 22, 211-232, (1982).

30. Henrici, P., *Discrete variable methods in ordinary differential equations.* New-York: John Wiley & Sons, (1962).

31. Kaps, P., *Semi-implicit Runge-Kutta methods of order 3 with stepsize control.* Institutsnotiz No. 2, Institut fuer Mathematik und Geometrie, Innsbruck, Austria, (1985).

32. Kaps, P., Ostermann, A., *Rosenbrock methods using few L-U decompositions.* Institutsnotiz No. 3, Institut fuer Mathematik und Geometrie, Innsbruck, Austria, (1985).

33. Kaps, P., Poon, S.W.H., Bui, T.D., *Rosenbrock methods for stiff ODEs: a comparison of Richardson extrapolation and the embedding technique.* Computing, 34, 17-40, (1985).

34. Kaps, P., Rentrop, P., *Generalized Runge-Kutta methods of order 4 with stepsize control for stiff ordinary differential equations.* Numerische Mathematik, 33, 55-68, (1979).

35. Kaps, P., Wanner, G., *A study of Rosenbrock-type methods of high order.* Numerische Mathematik, 38, 279-298, (1981).

36. Lambert, J.D., *Computational methods in ordinary differential equations.* New-York: John Wiley & Sons, (1976).

37. Nørsett, S.P., *C-polynomials for rational approximations to the exponential function.* Numerische Mathematik, 25, 39-56, (1975).

38. Nørsett, S.P., *One-step methods of Hermite type for numerical integration of stiff systems.* BIT, 14, 63-77, (1974).

39. Nørsett, S.P., *Restricted Padé approximations to the exponential function.* SIAM Journal of Numerical Analysis, 15, 1008-1029, (1978).

40. Nørsett, S.P., *Runge-Kutta methods with a multiple real eigenvalue only.* BIT, 16, 388-393, (1976).

41. Nørsett, S.P., *Semi-explicit Runge-Kutta methods.* Mathematics and Computation No. 6/74, University of Trondheim, Norway, (1974).

42. Nørsett, S.P., Wanner, G., *The real-pole sandwich for rational approximations and oscillation equations.* BIT, 19, 79-94, (1979).

43. Nørsett, S.P., Wolfbrandt, A., *Attainable order of rational approximations to the exponential function with only real poles.* BIT, 17, 200-208, (1977).

44. Nørsett, S.P., Wolfbrandt, A., *Order conditions for Rosenbrock-type methods.* Numerische Mathematik, 32, 1-15, (1979).

45. Poon, S., *Rosenbrock-type methods and error estimates for the numerical integration of stiff systems of ODEs.* Master's Thesis, Dept. of Computer Science, Concordia University, (1981).

46. Rosenbrock, H.H., *Some general implicit processes for the numerical solution of differential equations.* The Computer Journal, 5, 329-330, (1963).

47. Shampine, L.F., *Implementation of Rosenbrock methods.* ACM Transactions on Mathematical Software, 8, 93-113, (1982).

48. Shampine, L.F., Gear, C.W., *A user's view of solving stiff ordinary differential equations.* SIAM Review, 21, 1-17, (1979).

49. Siemieniuch, J.L., *Properties of certain rational approximations to e^{-x}.* BIT, 16, 172-191, (1976).

50. Steihaug, T., Wolfbrandt, A., *An attempt to avoid exact Jacobian and nonlinear equations in the numerical solution of stiff differential equations.* Mathematics of Computation, 33, 521-534, (1979).

51. Tricomi, F.G., *Vorlesungen ueber Orthogonalreihen.* Berlin: Springer, (1970).

52. Veldhuisen, M. van, *D-stability.* SIAM Journal of Numerical Analysis, 18, 45-64, (1981).

53. Veldhuisen, M. van, *On D-stability and B-stability.* Numerische Mathematik, 42, 349-359, (1983).

54. Verwer, J.G., *An analysis of Rosenbrock methods for nonlinear stiff initial value problems.* SIAM Journal of Numerical Analysis, 19, 155-170, (1982).

55. Verwer, J.G., *On the practical value of the notion of BN-stability.* BIT, 21, 355-361, (1981).

56. Wanner, G., *On the choice of γ for singly-implicit Runge-Kutta or Rosenbrock methods.* BIT, 20, 102-106, (1980).

57. Wanner, G., Hairer, E., Nørsett, S.P., *Order stars and stability theorems.* BIT, 18, 475-489, (1978).

58. Widlund, O.B., *A note on unconditionally stable linear multistep methods.* BIT, 7, 65-70, (1967).

59. Wolfbrandt, A., *A note on a recent result of rational approximations to the exponential function.* BIT, 17, 367-368, (1977).

60. Wolfbrandt, A., *A study of Rosenbrock processes with respect to order conditions and stiff stability.* Research Report No. 77.01, Dept. of Computer Science, Chalmers University of Technology and the University of Göteborg, Göteborg, Sweden, (1977).

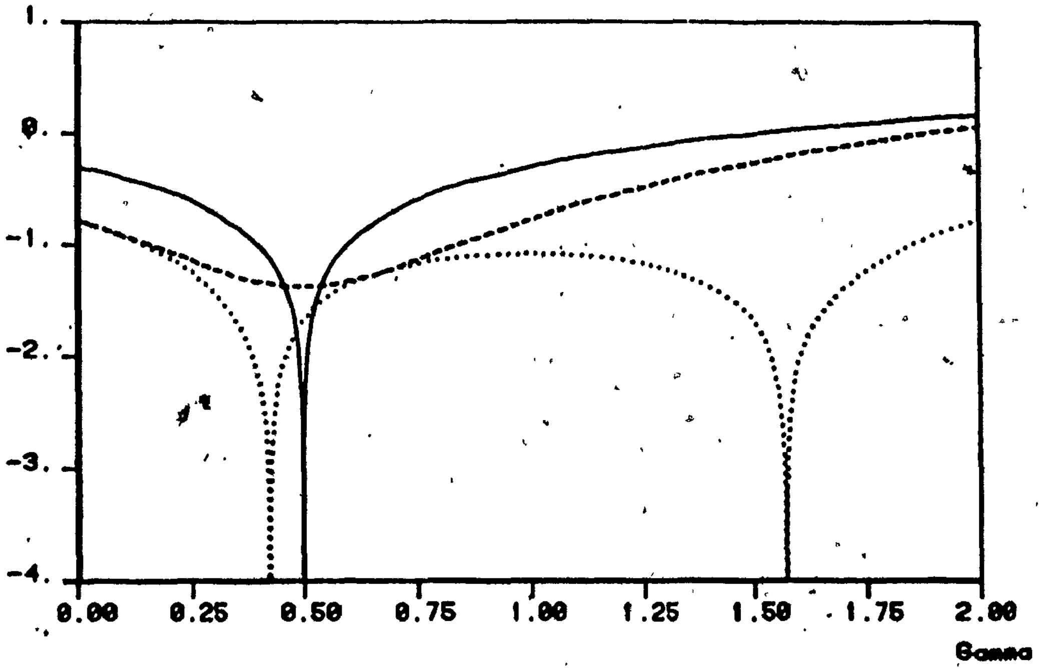
APPENDIX 1

Logarithm of Absolute Error Constants

This appendix presents plots of the logarithm (base 10) of the absolute error constants for the basic method (solid line), Richardson extrapolation (dashed line), and modified Richardson extrapolation (dotted line).

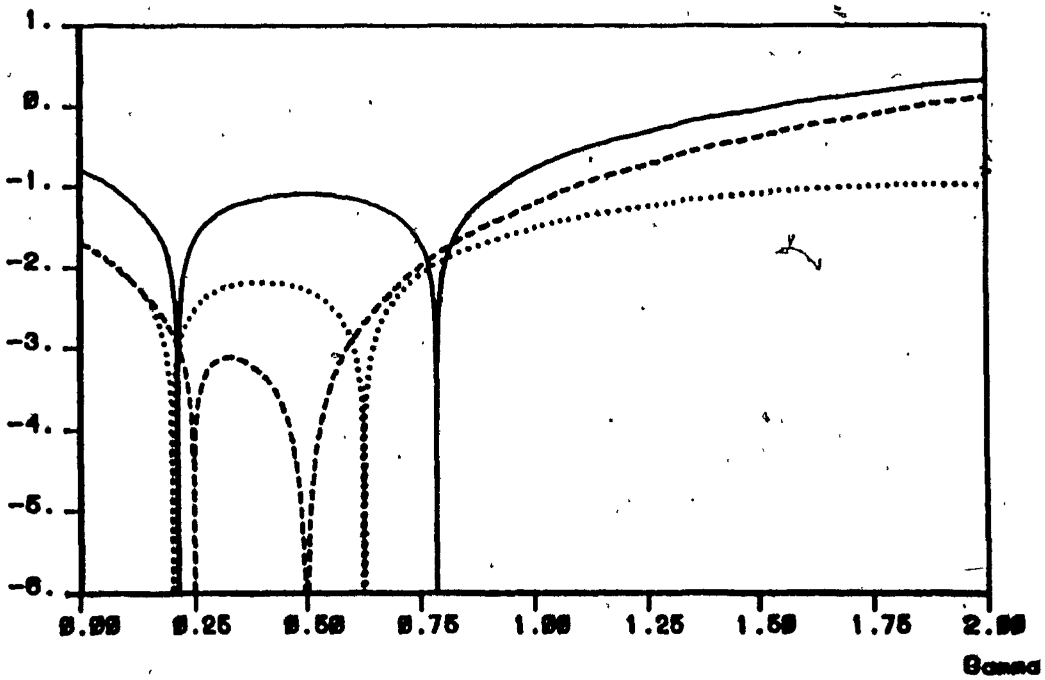
Plots are given for $\gamma \in [0.0,2.0]$, for the cases p=s and p=(s-1), with s = 1, 2, ... 8. For $p \geq 4$, two graphs are provided for each value of p, the second being an enlargement of the "small γ"-region of the first graph.

Log(base 10) of absolute error constant vs Gamma

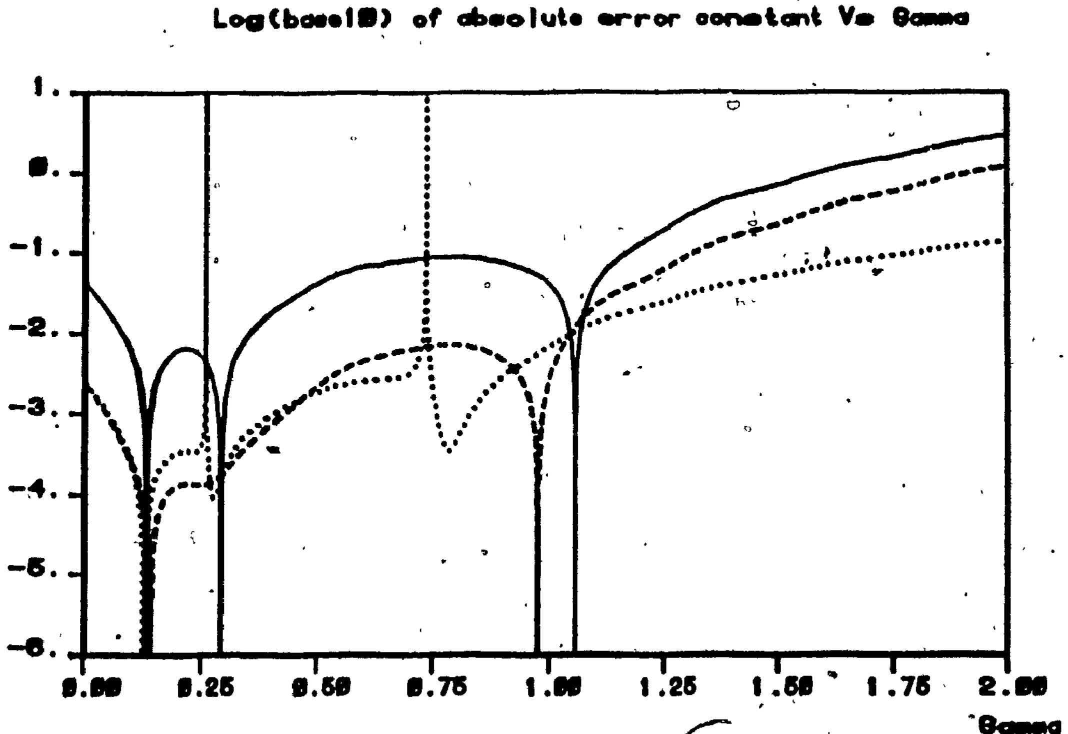


S = P = 1

Log(base10) of absolute error constant Vs Gamma

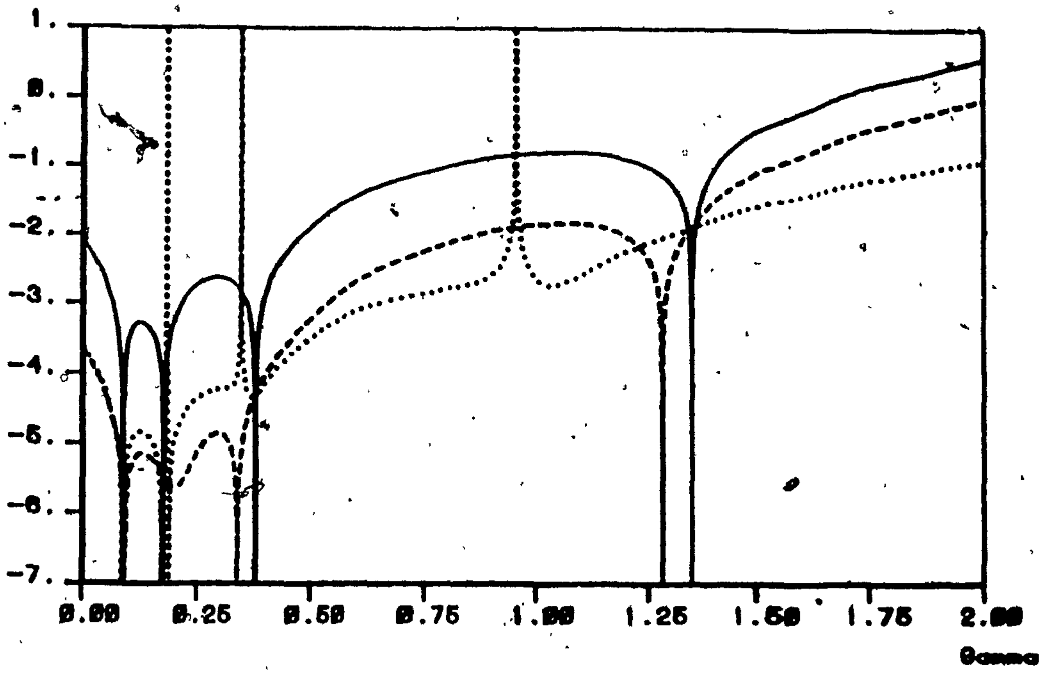


S = P = 2


Log(base10) of absolute error constant Vs Gamma
1.
0.
-1.
-2.
-3.
-4.
-5.
-6.
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Gamma

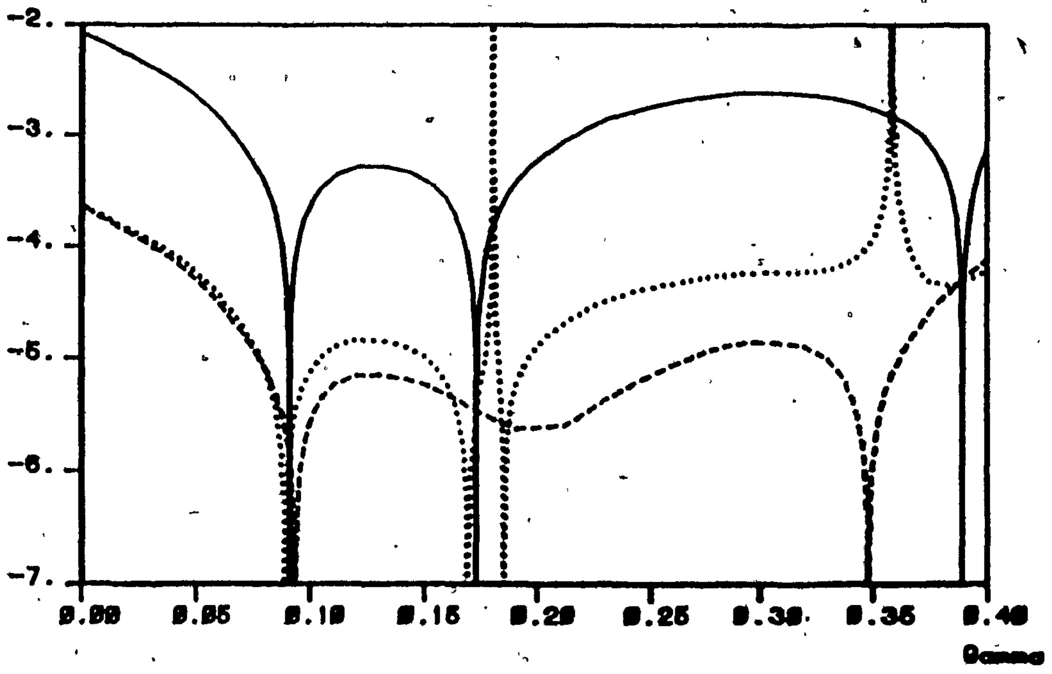
S = P = 3

Log(base10) of absolute error constant Vs Gamma



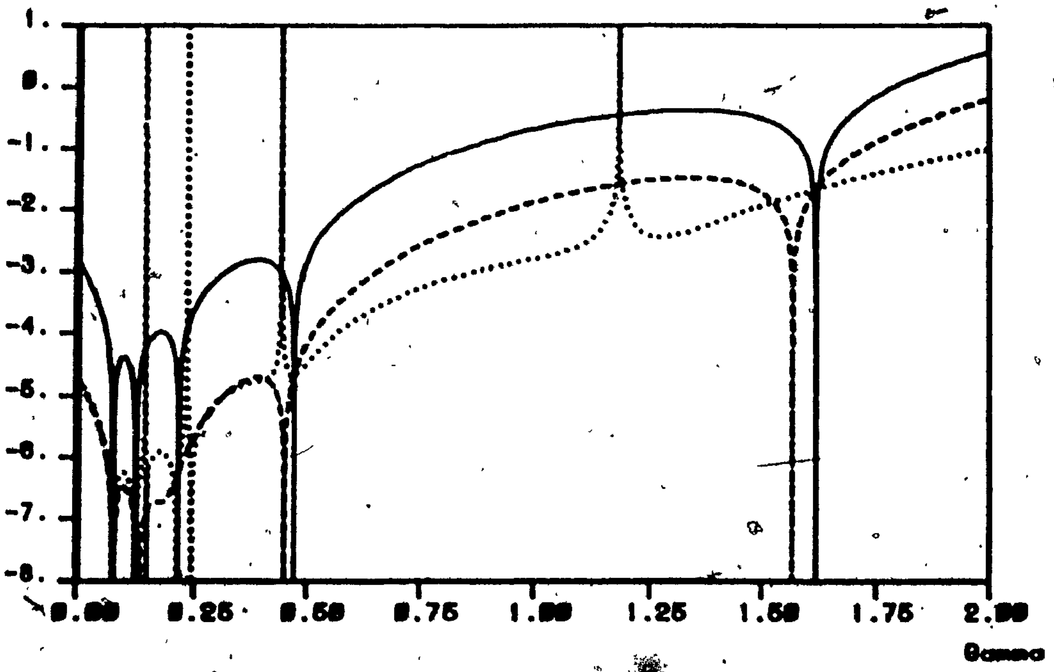
S = P = 4

Log(base10) of absolute error constant Vs Gamma



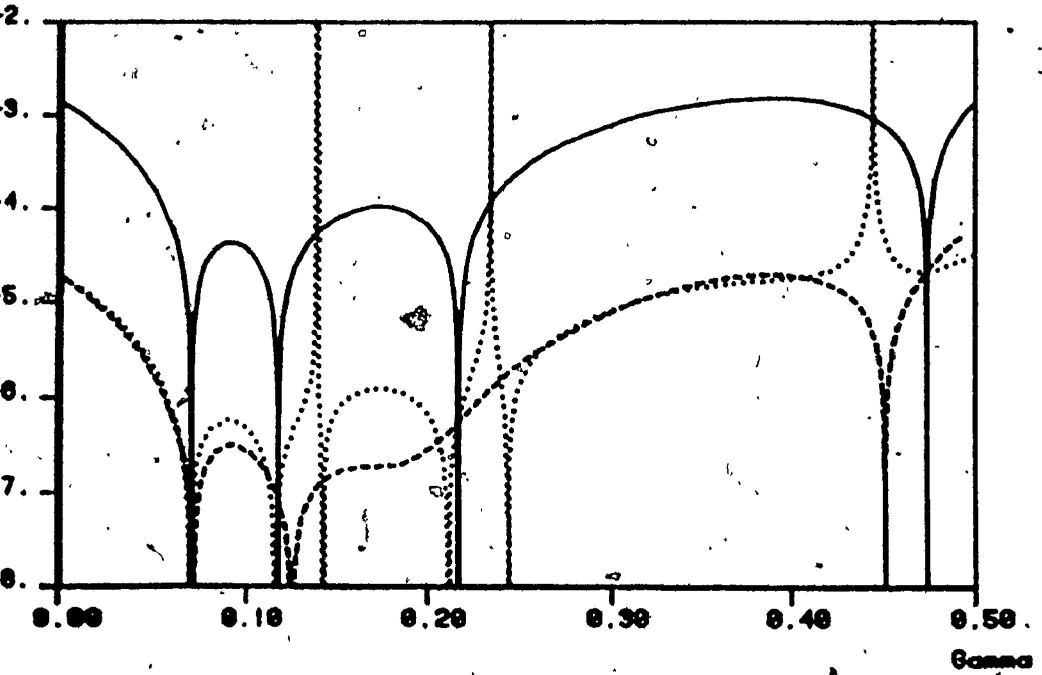
S = P = 4

Log(base10) of absolute error constant Vs Gamma



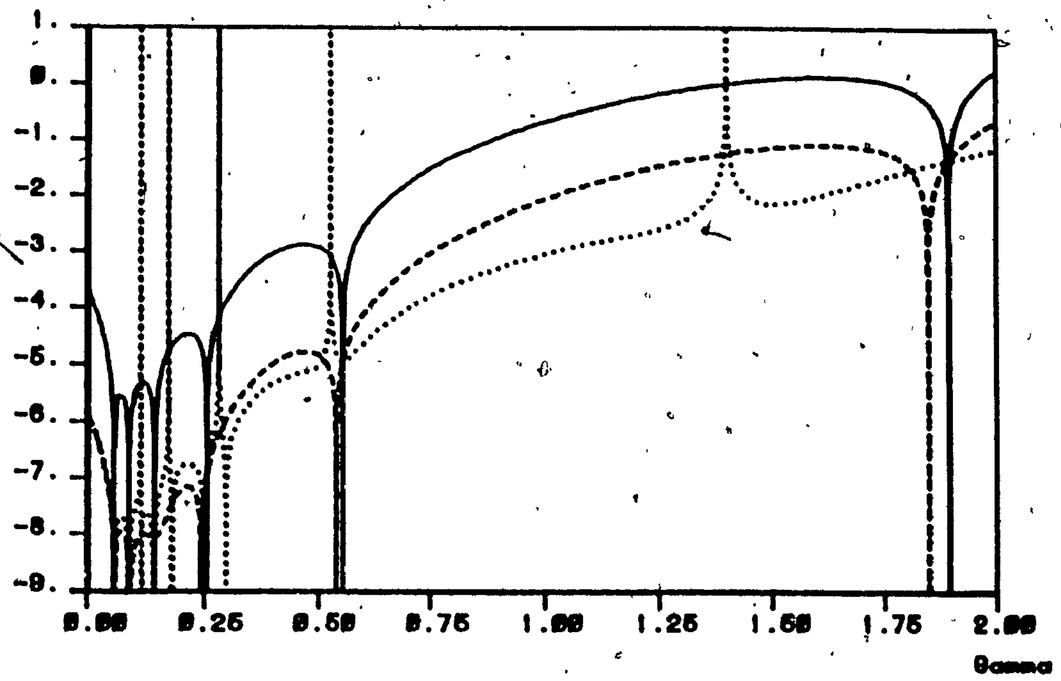
S = P = 5

Log(base 10) of absolute error constant vs Gamma



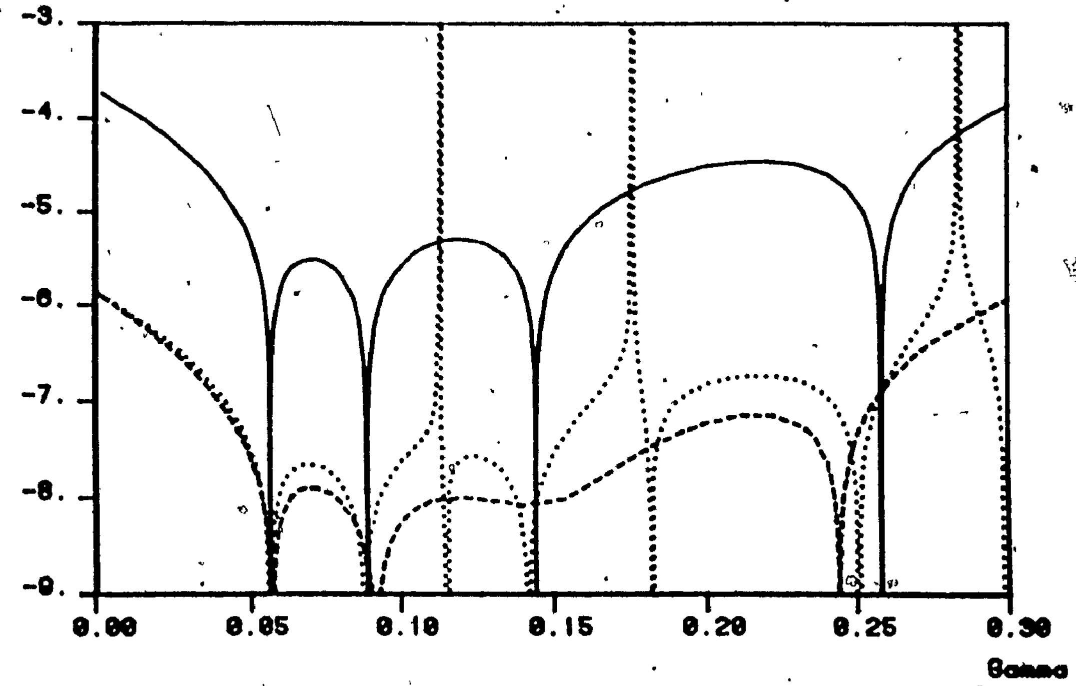
S = P = 5

Log(base10) of absolute error constant Vs Gamma



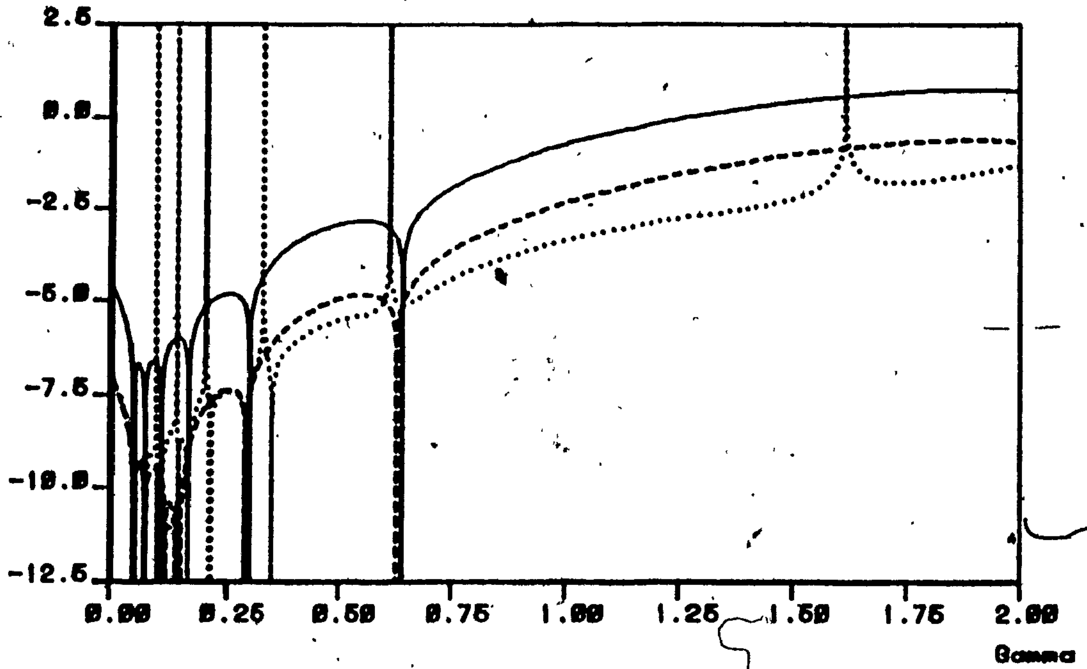
S = P = 6

Log(base 10) of absolute error constant vs Gamma



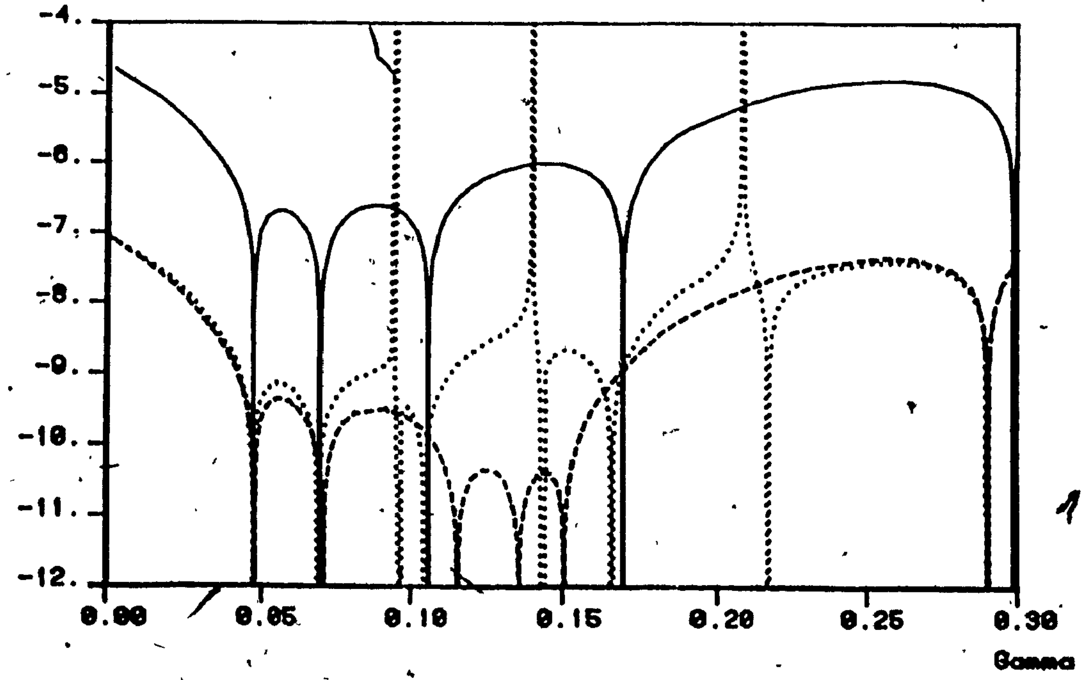
S = P = 6

Log(base10) of absolute error constant Vs Gamma



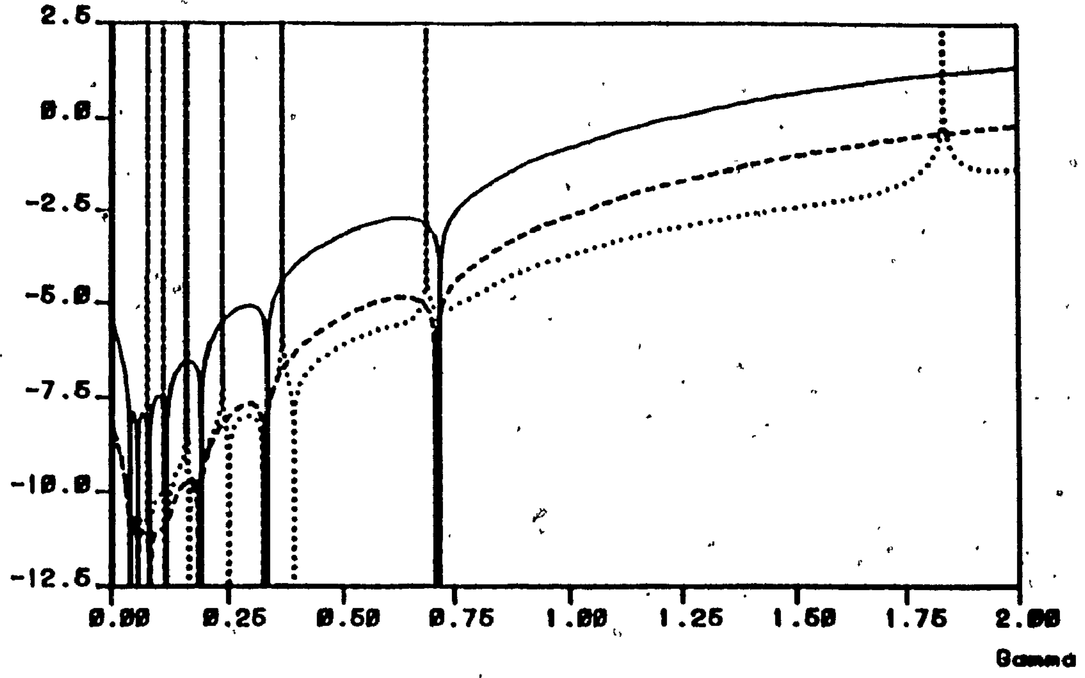
S = P = 7

Log(base 10) of absolute error constant vs Gamma



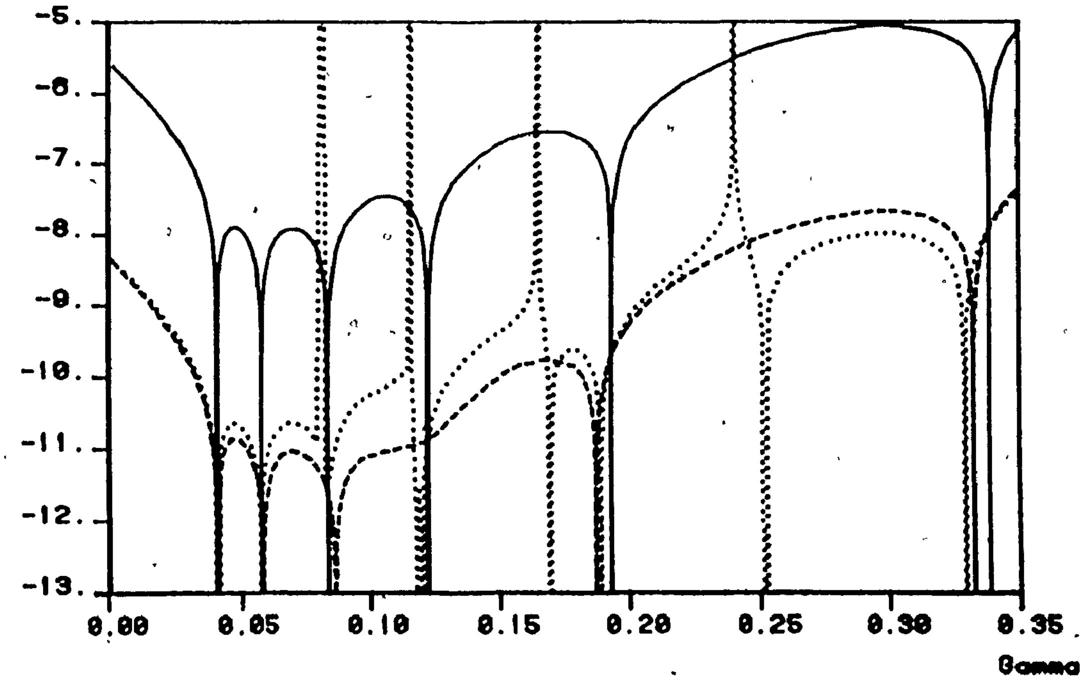
S = P = 7

Log(base10) of absolute error constant Vs Gamma



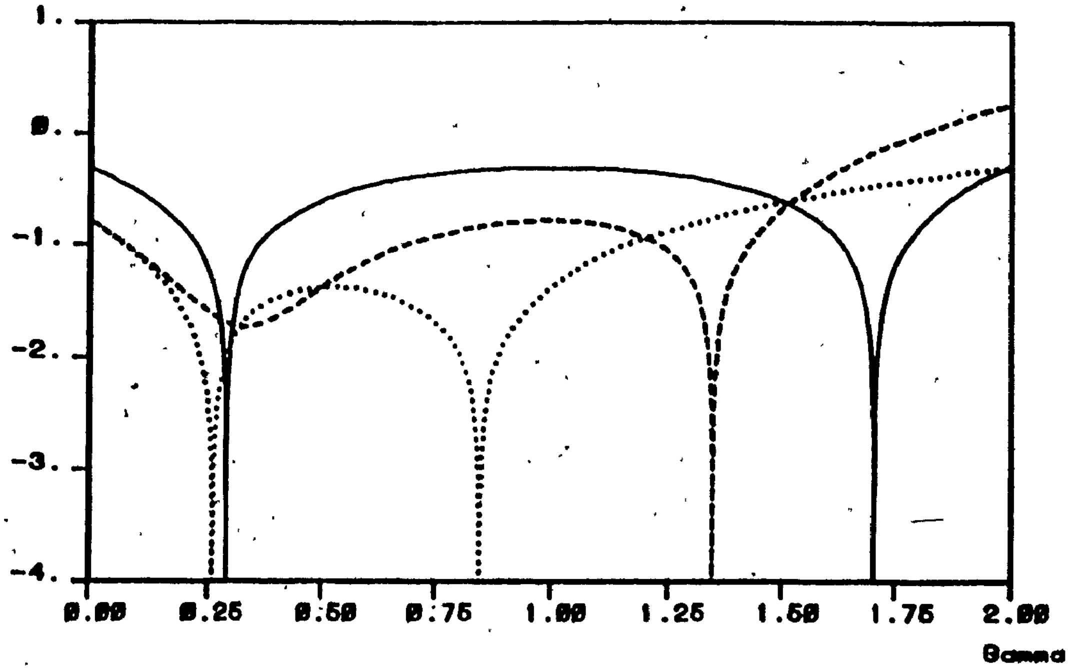
S = P = 8

Log(base 10) of absolute error constant vs Gamma



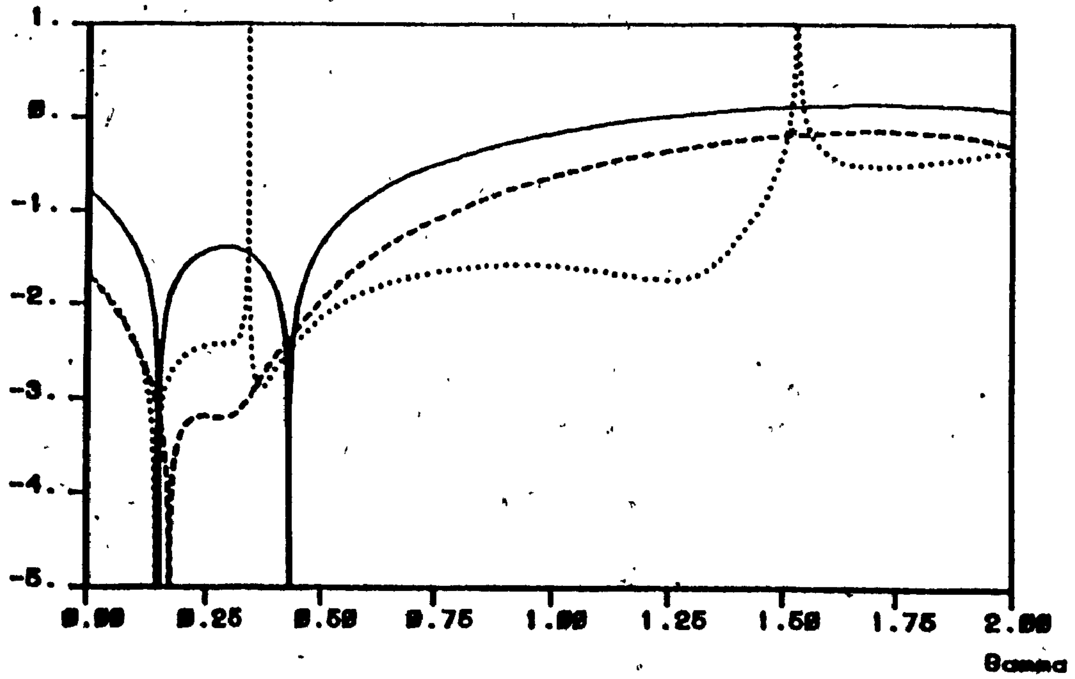
S = P = 8

Log(base10) of absolute error constant Vs Gamma

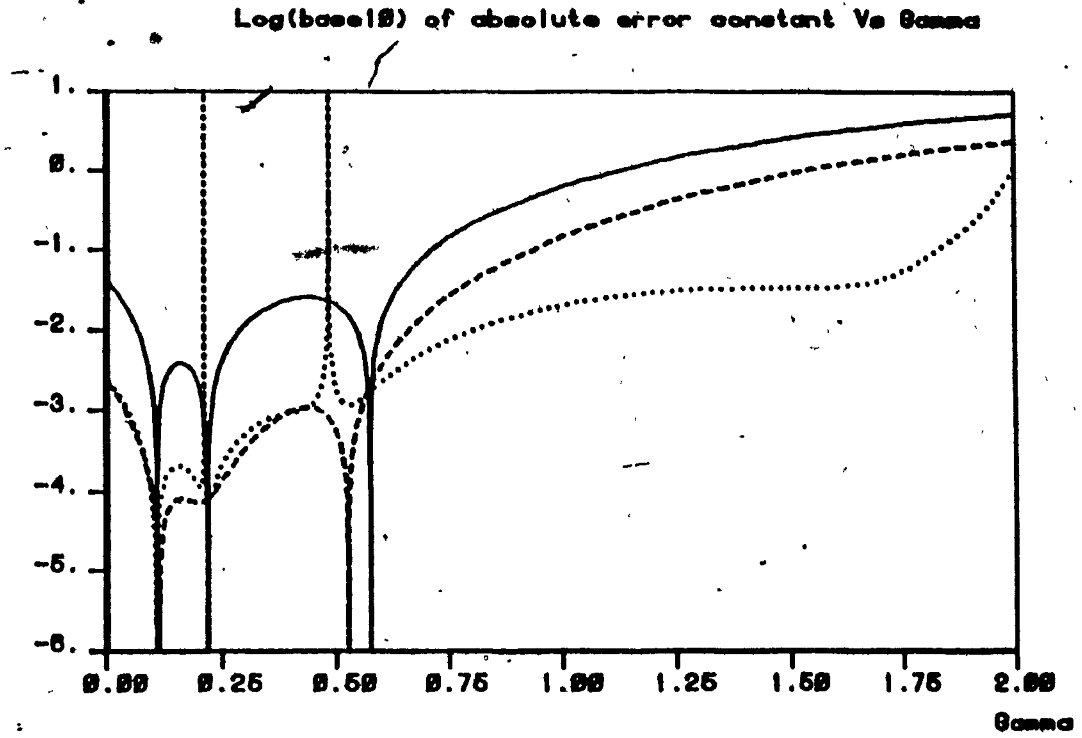


S = P + 1 = 2

Log(base10) of absolute error constant Vs Gamma

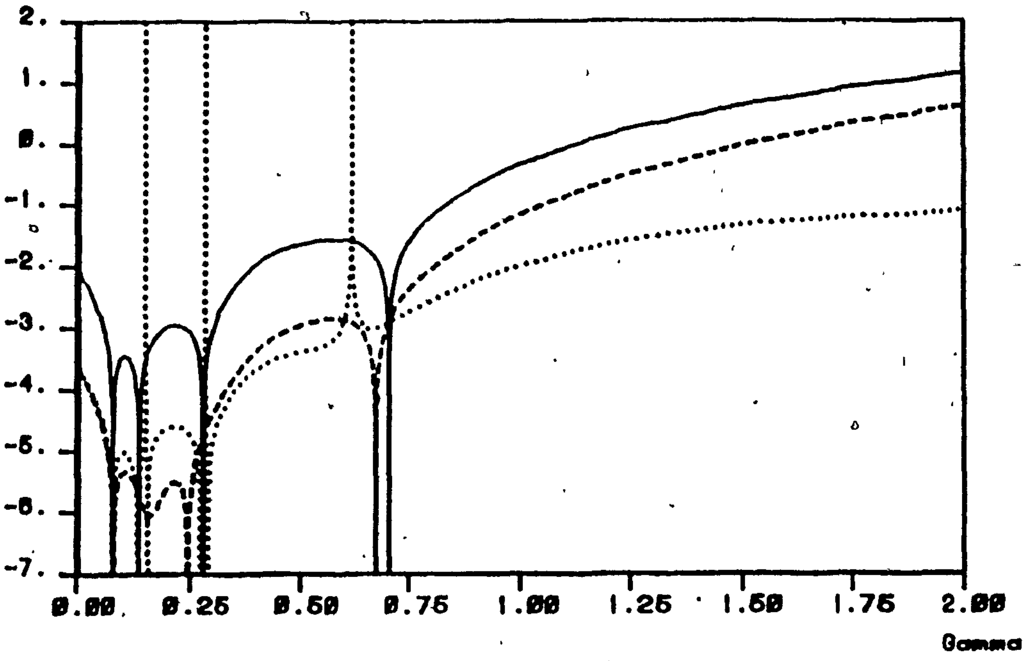


S = P + 1 = 3


Log(base10) of absolute error constant Vs Gamma
1.
0.
-1.
-2.
-3.
-4.
-5.
-6.
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Gamma

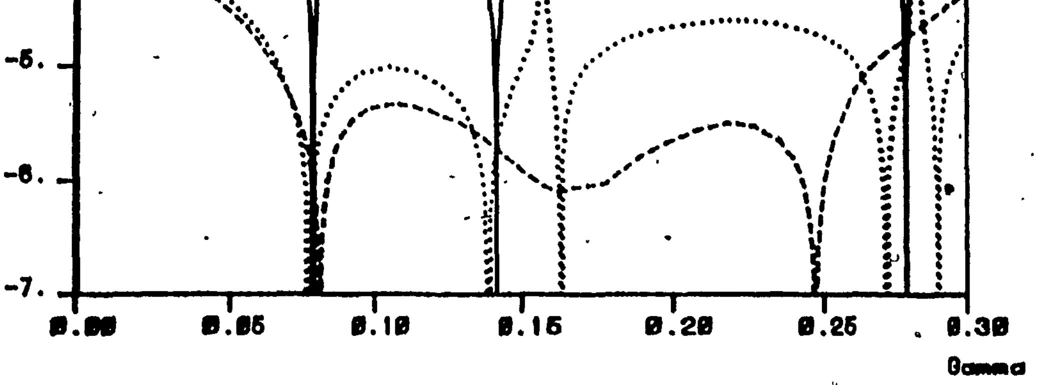
S = P + 1 = 4

Log(base10) of absolute error constant Vs Gamma



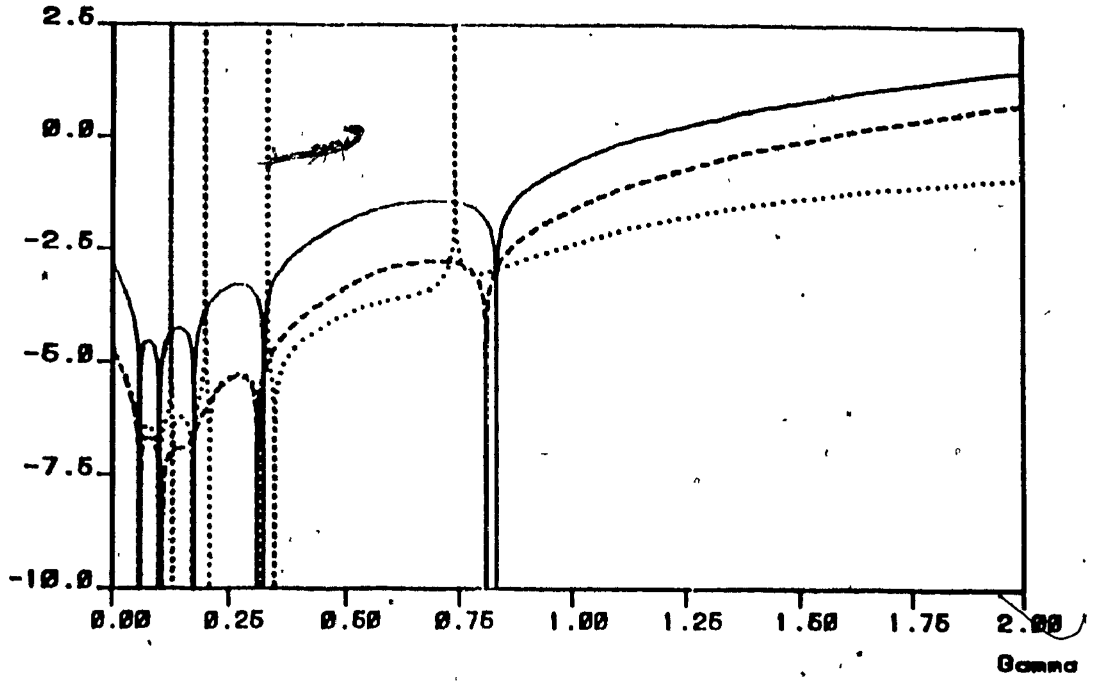
S = P + 1 = 5

Log(base10) of absolute error constant Vs Gamma



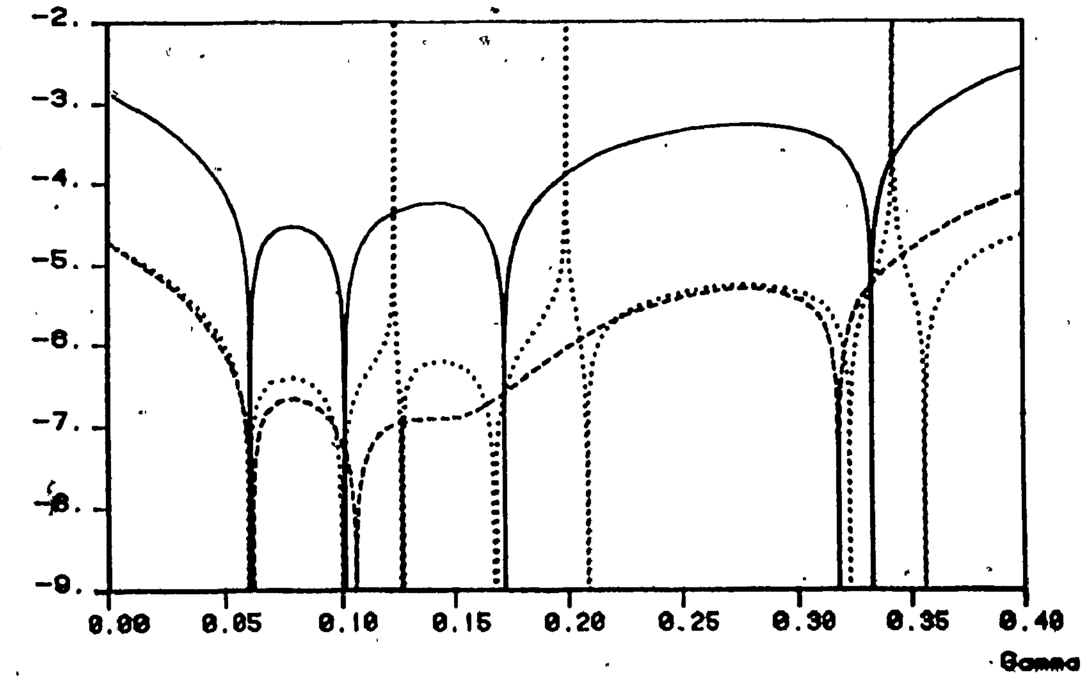
S = P + 1 = 5

Log(base10) of absolute error constant Vs Gamma

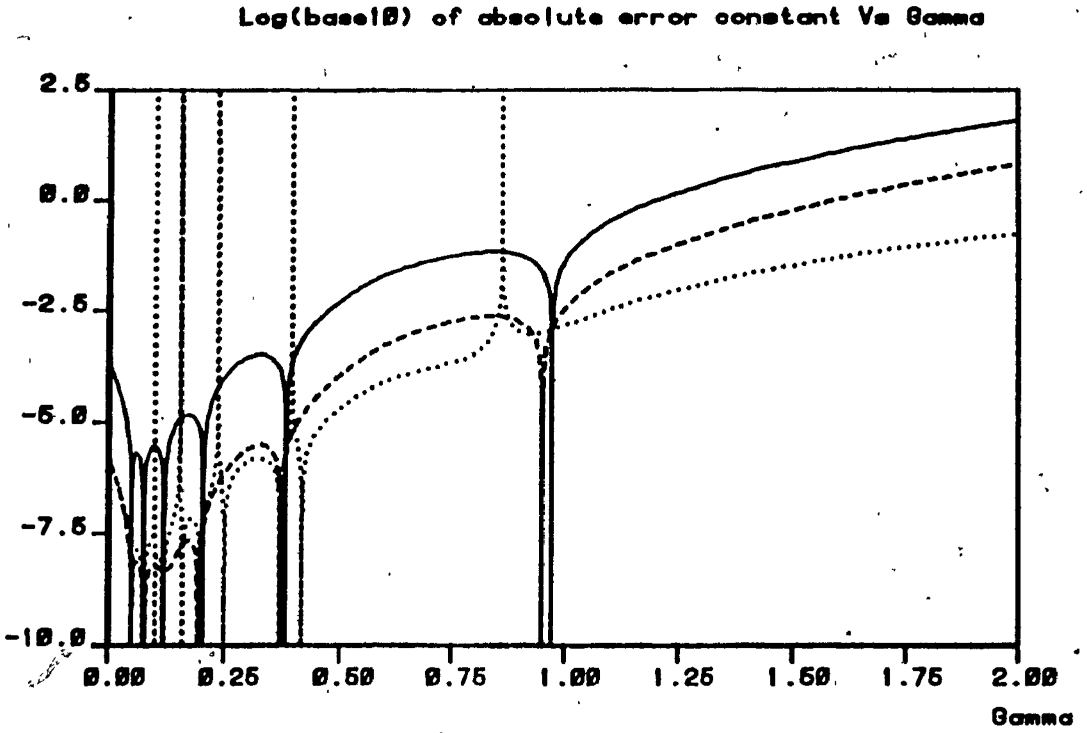


S = P + 1 = 6

Log(base 10) of absolute error constant vs Gamma

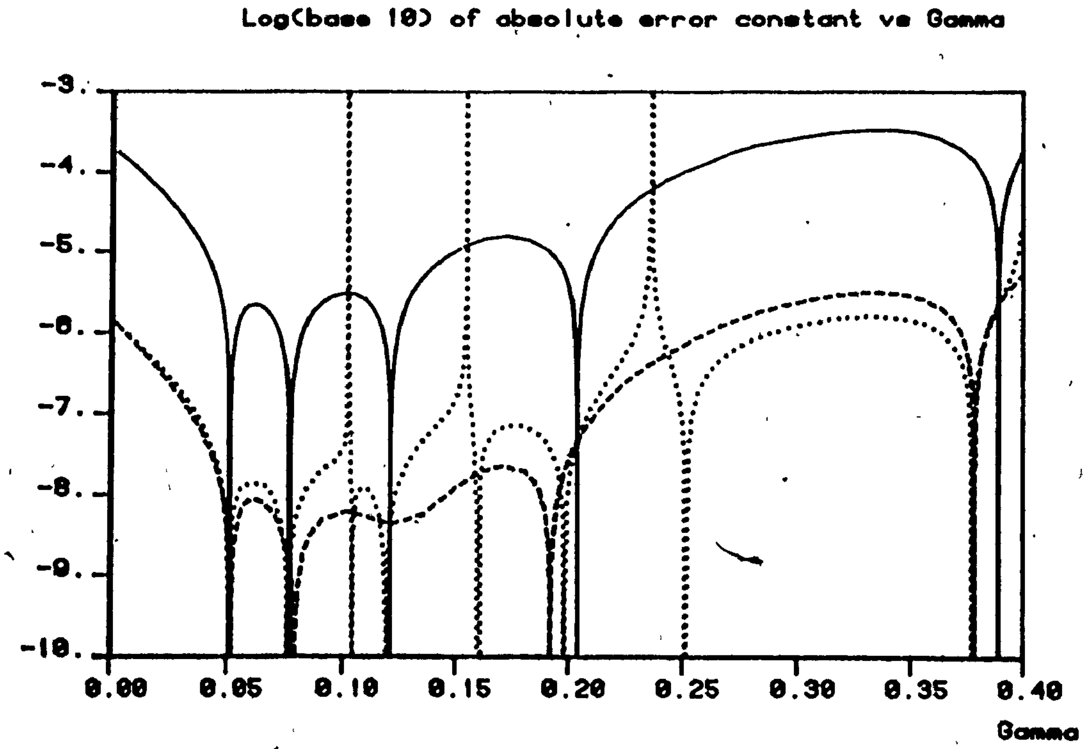


S = P + 1 = 6



Log(base10) of absolute error constant Vs Gamma
2.5
0.0
-2.5
-5.0
-7.5
-10.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Gamma

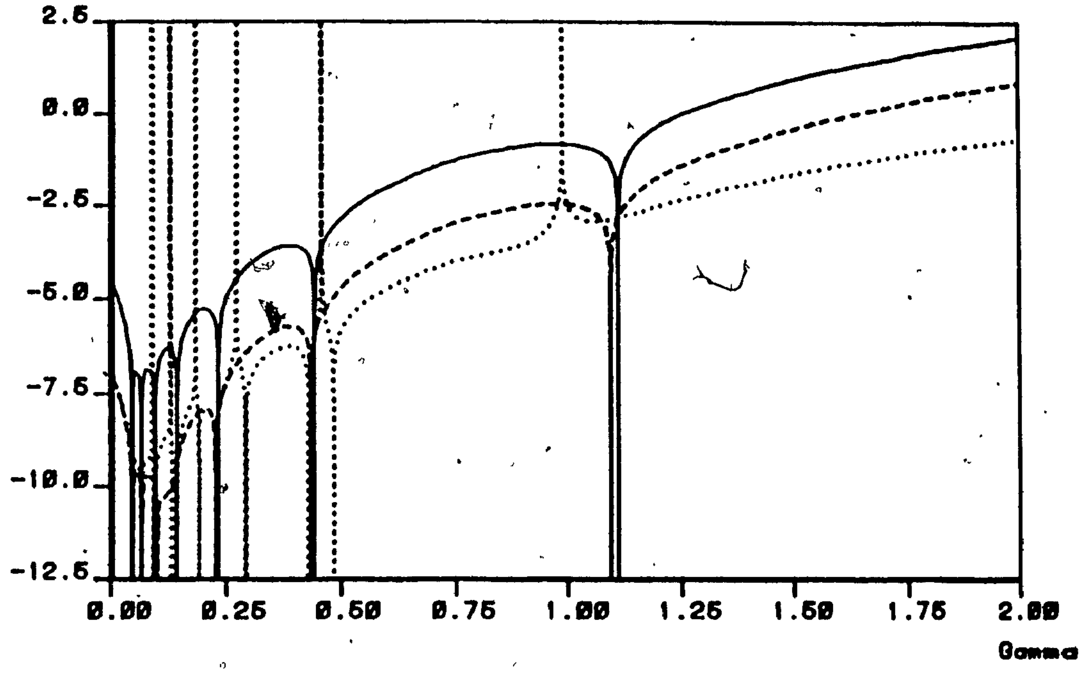
S = P + 1 = 7



Log(base 10) of absolute error constant vs Gamma
-3.
-4.
-5.
-6.
-7.
-8.
-9.
-10.
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Gamma

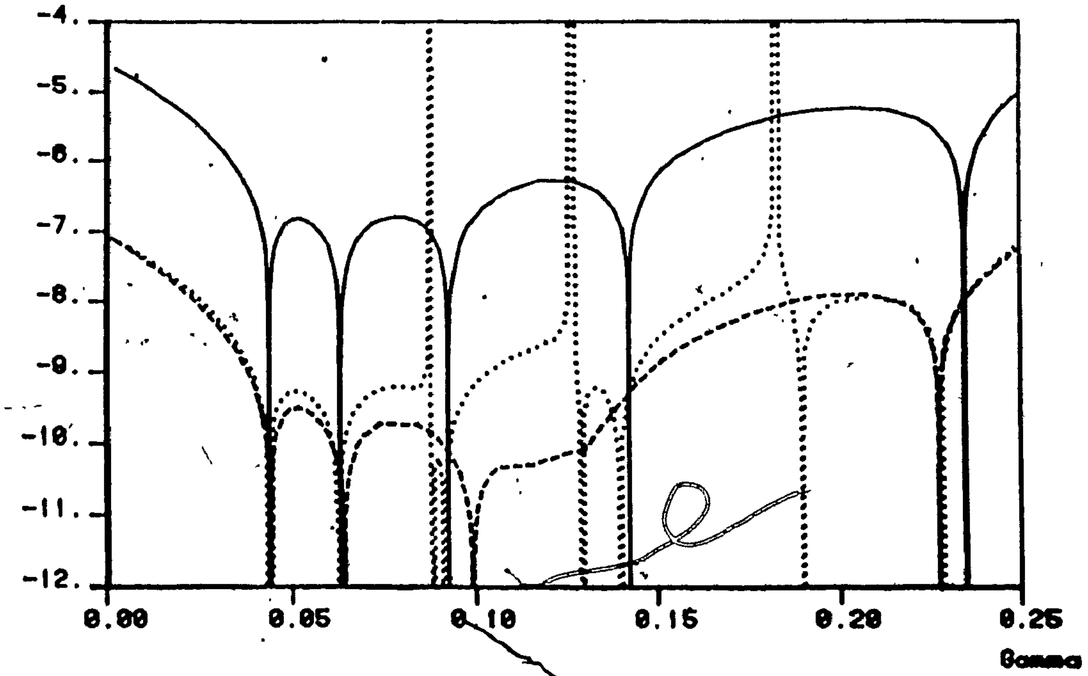
S = P + 1 = 7

Log(base10) of absolute error constant Vs Gamma



S = P + 1 = 8

Log(base 10) of absolute error constant vs Gamma



S = P + 1 = 8

APPENDIX 2

γ-Intervals for Stability at Infinity

For the case p=s, the intervals of stability at infinity are provided within [0.0,2.0], with an accuracy of 10 digits after the decimal point, for the basic method, Richardson extrapolation, and modified Richardson extrapolation.

For the case p=(s-1), all three methods are stable at infinity for any $\gamma \in (0.0, \infty)$.

	Basic method	Richardson extrapolation	Modified Richardson
p=s=1	[0.5000000000, ∞)	[0.6666666667, ∞)	[0.5000000000, ∞)
p=s=2	[0.2500000000, ∞)	[0.2500000000, 0.3693980625] [0.7734590804, ∞)	[0.5000000000, 0.5000000000] [0.6980648682, ∞)
p=s=3	[0.1533193629, 0.1666666666] [0.3333333334, ∞)	[0.1539530258, 0.1666666666] [0.3333333334, ∞)	[0.1472474769, 0.1488985597] [0.1859451800, 0.2401216698] [0.3333333334, 0.4527525231] [0.7387814874, ∞)
p=s=4	[0.1056624328, 0.1072789059] [0.2038425245, 0.2500000000] [0.3943375673, ∞)	[0.1056624328, 0.1072243112] [0.2047106042, 0.2500000000] [0.3943375673, ∞)	[0.1802001018, 0.1804359286] [0.2074492815, 0.3265792356] [0.3910324283, 0.7415011197] [0.9657091306, ∞)
p=s=5	[0.0790023320, 0.0792174872] [0.1381966012, 0.1446533596]	[0.0790056332, 0.0792174872] [0.1381966012, 0.1445311400]	[0.0782277965, 0.0782382478] [0.0801543774, 0.0801689687] [0.1506028771, 0.1516802234]

Basic method	Richardson extrapolation	Modified Richardson
p=s=5(continued)		[0.2271235366, 0.2295310167]
[0.2465051932, 0.3618033988]	[0.2473110122, 0.3618033988]	[0.2646179064, 0.4175383364]
[0.4207825128, ∞)	[0.4207825128, ∞)	[0.4655249471, 1.1229891433]
		[1.1880784728, ∞)
p=s=6		
[0.0625521650, 0.0625818351]	[0.0625521650, 0.0625816024]	[0.0987126723, 0.0987673792]
[0.1011425802, 0.1021910733]	[0.1011503359, 0.1021910733]	
[0.1666666667, 0.1819126665]	[0.1666666667, 0.1817466118]	[0.1909351920, 0.1944122176]
		[0.2684201679, 0.2749637698]
[0.2840646381, ∞)	[0.2847137610, ∞)	[0.3162186552, 0.5060748389]
		[0.5446935958, 1.3540580640]
		[1.4081748255, ∞)
p=s=7		
[0.0515556619, 0.0515598307]	[0.0515556782, 0.0515598307]	[0.0514742706, 0.0514743200]
		[0.0516442231, 0.0516442759]
[0.0784418185, 0.0786171416]	[0.0784418185, 0.0786164462]	[0.0950054662, 0.0950057706]
[0.1208848250, 0.1236982326]	[0.1208947490, 0.1236982326]	[0.1178485598, 0.1181011604]
[0.1927306812, 0.2213231260]	[0.1927306812, 0.2211370449]	[0.2278192382, 0.2364098033]
[0.3171124189, ∞)	[0.3176100364, ∞)	[0.3065568208, 0.3192800500]
		[0.3642179032, 0.5916421525]
		[0.6249242871, 1.5738784811]
		[1.6269664610, ∞)

Basic method	Richardson extrapolation	Modified Richardson
p=s=8		
[0.0437382439, 0.0437388360]	[0.0437382439, 0.0437388349]	[0.0632274578, 0.0632280935]
[0.0635149404, 0.0635444417]	[0.0635149978, 0.0635444417]	[0.0638613659, 0.0638621172]
		[0.0813305724, 0.0813305760]
[0.0926841283, 0.0932243458]	[0.0926841283, 0.0932232536]	[0.1162689718, 0.1162725317]
[0.1392988724, 0.1450176578]	[0.1393084539, 0.1450176578]	[0.1361707938, 0.1369236276]
[0.2170048191, 0.2647163947]	[0.2170048191, 0.2645144053]	[0.2624909880, 0.2800247144]
[0.3449341067, ∞)	[0.3453262687, ∞)	[0.3413937543, 0.3630059834]
		[0.4098604046, 0.6752671126]
		[0.7053760619, 1.7904300724]
		[1.8449376710, ∞)

APPENDIX 3

γ-Intervals for A(0)-Stability

For both cases p=s and p=(s-1), all the A(0)-stable intervals in [0.0,2.0] are provided with an accuracy of 10 digits after the decimal point, for the basic method, Richardson extrapolation, and modified Richardson extrapolation.

γ-INTERVALS FOR A(0)-STABILITY (p=s)

Basic method	Richardson extrapolation	Modified Richardson
p=s=1		
[0.5000000000, ∞)	[0.6666666667, 2.0000000000]	[0.5000000000, 2.0000000000]
p=s=2		
[0.2500000000, ∞)	[0.2500000000, 0.3693980625]	[0.6980648682, 2.0000000000]
	[0.7734590804, 2.0000000000]	
p=s=3		
[0.1533193629, 0.1666666666]	[0.1539530258, 0.1666666666]	[0.1931007995, 0.2291012453]
[0.3333333334, 2.0000000000]	[0.3333333334, 2.0000000000]	[0.3333333334, 0.4527525231]
		[0.7387814874, 2.0000000000]
p=s=4		
[0.1056624328, 0.1060555535]	[0.1056624328, 0.1059021605]	
[0.2038425244, 0.2500000000]	[0.2047106042, 0.2499999999]	[0.2074992815, 0.3265792356]
[0.3943375673, 2.0000000000]	[0.3943375673, 2.0000000000]	[0.3910324283, 0.7415011197]
		[0.9657091306, 2.0000000000]

Basic method	Richardson extrapolation	Modified Richardson
p=s=5		
[0.1381966012, 0.1446463307]	[0.1381966012, 0.1445136962]	[0.1506028771, 0.1516802234]
		[0.2271235366, 0.2281959908]
[0.2465051932, 0.3618033988]	[0.2473110122, 0.3618033988]	[0.2646179064, 0.4175383364]
[0.4207825128, 2.0000000000]	[0.4207825128, 2.0000000000]	[0.4655249471, 1.1229891433]
		[1.1880784728, 2.0000000000]
p=s=6		
[0.1011425302, 0.1014643414]	[0.1011503359, 0.1014615305]	
[0.1666666667, 0.1819126666]	[0.1666666667, 0.1817466118]	[0.1909351920, 0.1944122176]
		[0.2684201679, 0.2742452861]
[0.2840646381, 2.0000000000]	[0.2847137609, 2.0000000000]	[0.3162186552, 0.5060748389]
		[0.5446935958, 1.3540580640]
		[1.4081748255, 2.0000000000]
p=s=7		
[0.1208848250, 0.1236019498]	[0.1208947489, 0.1236018335]	
[0.1927306812, 0.2213231260]	[0.1927306812, 0.2211370449]	[0.2278192382, 0.2364098033]
		[0.3065568208, 0.3189764522]
[0.3171124189, 2.0000000000]	[0.3176100364, 2.0000000000]	[0.3642179032, 0.5916421525]
		[0.6249242871, 1.5738784811]
		[1.6269664610, 2.0000000000]

Basic method	Richardson extrapolation	Modified Richardson
p=s=8		
[0.1392988724, 0.1450176579]	[0.1393084539, 0.1450176578]	
[0.2170048191, 0.2647163948]	[0.2170048191, 0.2645144053]	[0.2624909880, 0.2800247144]
		[0.3413937543, 0.3629102335]
[0.3449341067, 2.0000000000]	[0.3453262687, 2.0000000000]	[0.4098604046, 0.6752671126]
		[0.7053760619, 1.7904300724]
		[1.8449376710, 2.0000000000]

γ-INTERVALS FOR A(0)-STABILITY (p=s-1)

Basic method	Richardson extrapolation	Modified Richardson
p=s-1=1		
[0.1464466095, 2.0000000000]	[0.2075752790, 2.0000000000]	[0.1721919166, 0.5344647580]
		[0.6050012514, 2.0000000000]
p=s-1=2		
[0.1339745963, 2.0000000000]	[0.1346852273, 2.0000000000]	[0.1368013322, 0.2537590819]
		[0.2759971258, 0.8098908074]
		[0.8597887838, 2.0000000000]
p=s-1=3		
[0.1079916182, 2.0000000000]	[0.1092963823, 2.0000000000]	[0.1039220719, 0.1137904590]
		[0.1357386229, 0.1785176624]
		[0.1870957518, 0.3596243757]
		[0.3665233546, 1.0610974740]
		[1.1057017740, 2.0000000000]

Basic method	Richardson extrapolation	Modified Richardson
p=s-1=4		
[0.0356871212, 0.0987902280]	[0.0857059871, 0.0976386082]	[0.1185196066, 0.1289733224]
[0.1172053136, 2.0000000000]	[0.1196468596, 2.0000000000]	[0.1538961788, 0.2297819393]
		[0.2378605867, 0.4487090066]
		[0.4525060673, 1.3051765321]
		[1.3483035539, 2.0000000000]
p=s-1=5		
[0.0686631737, 0.0698210577]	[0.0686891880, 0.0698118852]	[0.1245513653, 0.1657238343]
[0.1027715610, 2.0000000000]	[0.1027971449, 2.0000000000]	[0.1854095487, 0.2770572492]
		[0.2827667651, 0.5338839819]
		[0.5364529926, 1.5460225891]
		[1.5892411854, 2.0000000000]
p=s-1=6		
[0.0841924774, 0.0962235184]	[0.0842378166, 0.0961966849]	[0.1074479867, 0.1222149929]
[0.1086585950, 2.0000000000]	[0.1087033589, 2.0000000000]	[0.1485992916, 0.1993701324]
		[0.2128164335, 0.3219163627]
		[0.3259344935, 0.6173134421]
		[0.6192675487, 1.7851028772]
		[1.8292109970, 2.0000000000]

Basic method	Richardson extrapolation	Modified Richardson
p=s-1=7		
[0.0690600020, 0.0706650650]	[0.0690606503, 0.0706544164]	
[0.0954956591, 2.0000000000]	[0.0955461446, 2.0000000000]	[0.0949925474, 0.0997717631]
		[0.1196659003, 0.1546820934]
		[0.1700328478, 0.2299871115]
		[0.2389738500, 0.3653473042]
		[0.3682338329, 0.6997826840]
		[0.7013840419, 2.0000000000]

APPENDIX 4

γ-Intervals for A-Stability

For both cases $p=s$ and $p=(s-1)$, all the A-stable intervals in [0.0,2.0] are provided with an accuracy of 10 digits after the decimal point, for the basic method, Richardson extrapolation, and modified Richardson extrapolation.

Basic method	Richardson extrapolation	Modified Richardson
p=s=1		
[0.5000000000, ∞)	[0.6666666667, 2.0000000000]	[0.5000000000, 2.0000000000]
p=s=2		
[0.2500000000, ∞)	----------	[0.6980648682, 2.0000000000]
p=s=3		
[0.3333333334, 1.0685790[illegible]	[0.3427625989, 0.9387415149]	[0.3333333334, 0.3885757263]
		[0.7387814874, 2.0000000000]
p=s=4		
[0.3943375673, 1.2805797612	----------	[0.3910324283, 0.7415011197]
p=s=5		
[0.2465051932, 0.3618033988]	[0.2473110122, 0.3617251799]	
[0.4207825128, 0.4732683912]	[0.4207825128, 0.4420798615]	[0.4655249471, 1.1229891433]
p=s=6		
[0.2840646381, 0.5409068780]	----------	[0.3162186552, 0.4634493247]
p=s=7		
----------	----------	[0.3642179032, 0.5914300543]
p=s=8		
[0.2170497431, 0.2647142465]	----------	----------

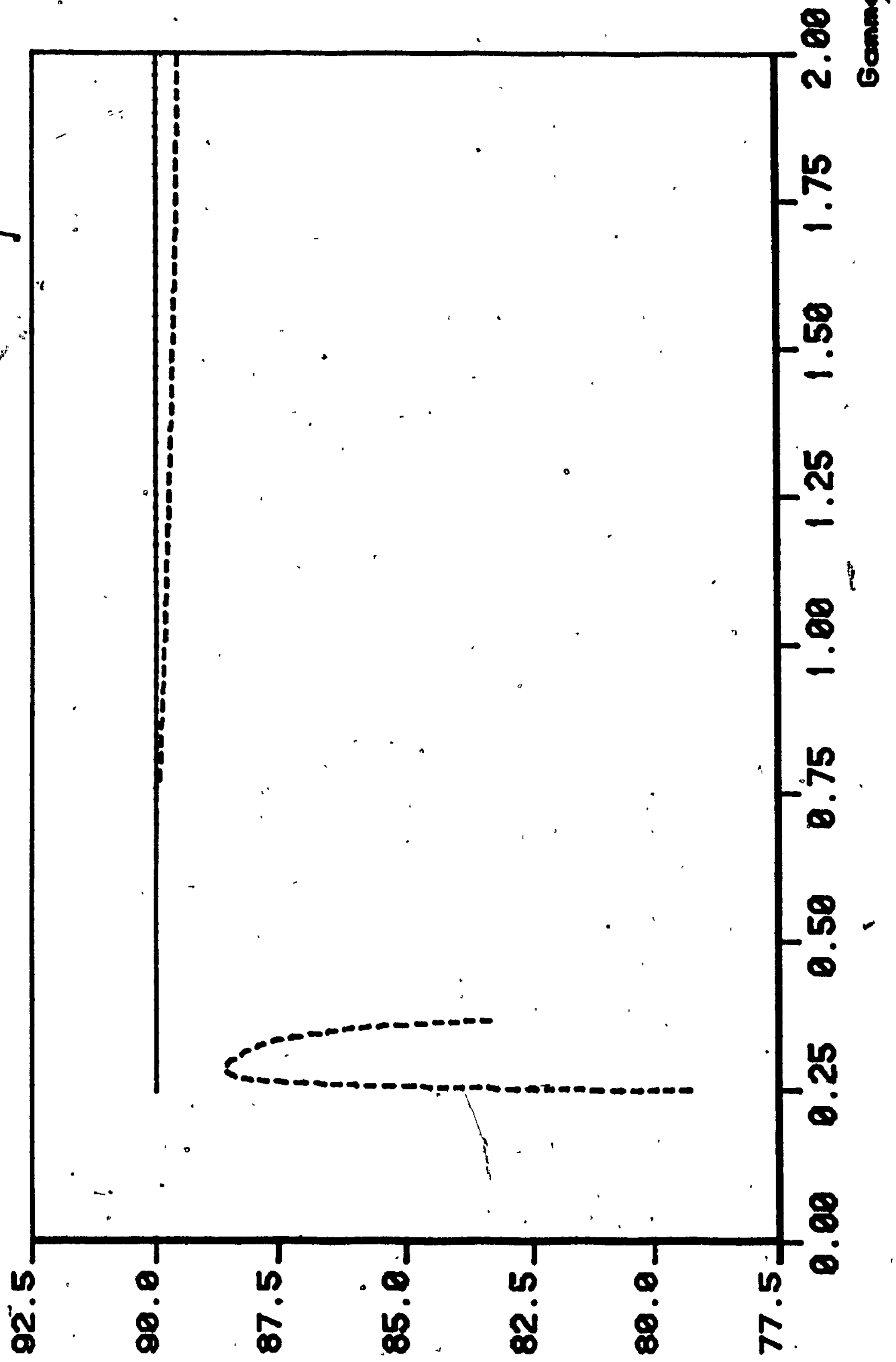
	Basic method	Richardson extrapolation	Modified Richardson
p=s-1=1	[0.2928932189, 1.7071067811]	[0.3376421306, 1.2561986596]	[0.6274321174, 2.0000000000]
p=s-1=2	[0.1804253065, 2.0000000000]	----------	[0.2820333303, 0.3147773843] [0.3333333334, 0.6403619441] [0.8896675429, 2.0000000000]
p=s-1=3	[0.2236478010, 0.5728160624]	[0.2349924674, 0.5105312760]	[0.3712082323, 0.9240383665]
p=s-1=4	[0.2479940363, 0.6760423932]	----------	[0.2939630288, 0.4284439357] [0.4564318474, 1.1677219666]
p=s-1=5	[0.1839146537, 0.3341423670]	[0.1861482462, 0.3135617892]	[0.2147400303, 0.2683100089] [0.3489150709, 0.5241555202]
p=s-1=6	[0.2040834518, 0.3788648944]	----------	[0.2444269424, 0.3142884225] [0.5043137607, 0.6096131725]
p=s-1=7	[0.1566585994, 0.2029348608] [0.2051941720, 0.2343731596]	[0.1571034631, 0.1960854393] [0.2116743808, 0.2249416856]	[0.1902090835, 0.1963255535] [0.3845588477, 0.4760938015]

APPENDIX 5

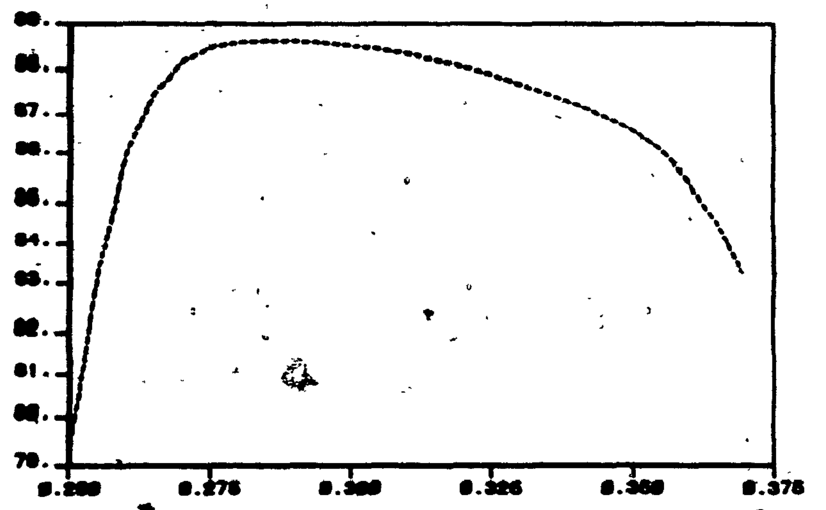
Maximum A(α)-Stability (p=s)

For the case p=s, this appendix presents plots of the maximum angle α (in degrees) for which we have A(α)-stability, as a function of γ for $\gamma \in [0.0,2.0]$. The *solid line* provides the basic method results; the *dashed line* and the *dotted line* are for Richardson extrapolation and modified Richardson extrapolation results.

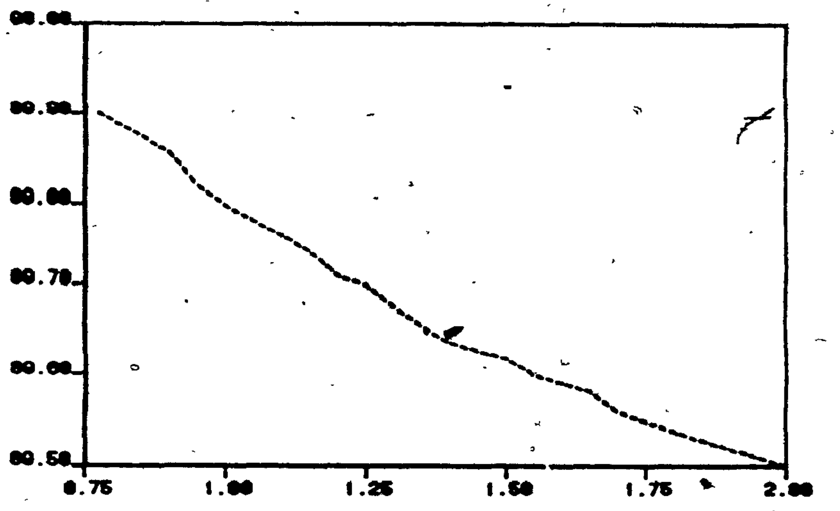
For each value of p, a series of graphs is provided. Generally, the first one presents the overall information. It is followed by many more plots, showing enlargements of certain subregions, so that more accurate information can be visually obtained if these subregions are of particular interest. The last plot of each series is an enlargement of the near—90° region.



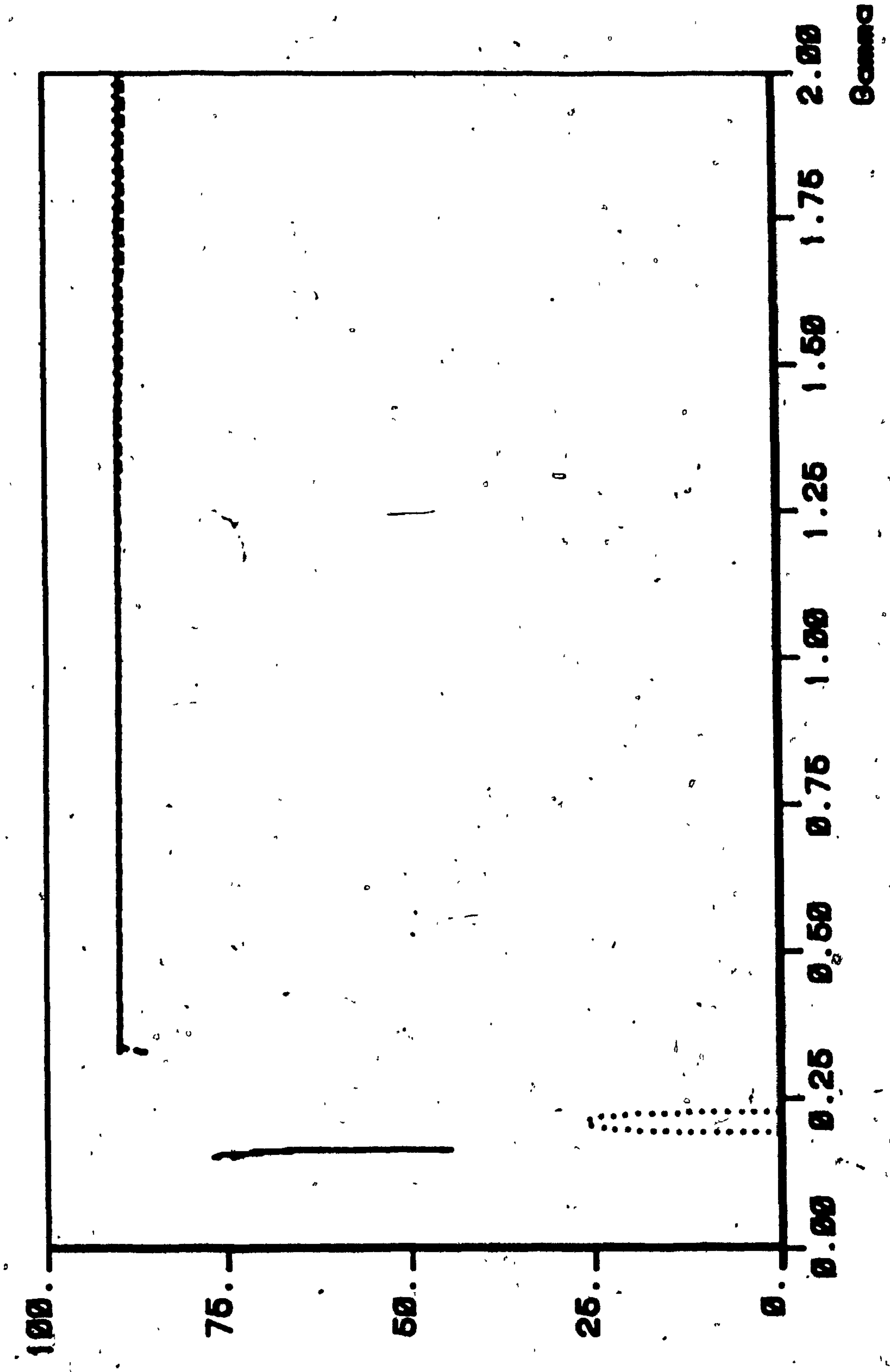
A(alpha)-STABILITY for S = P = 2



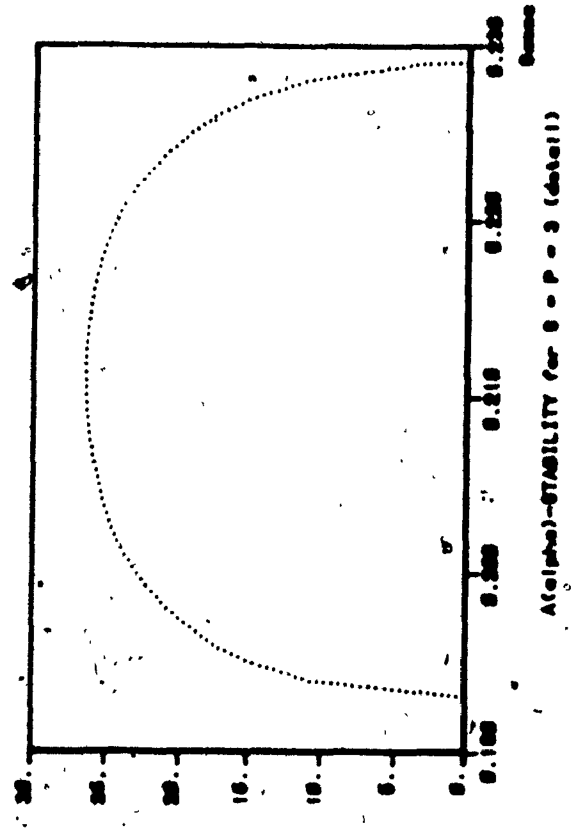
A(alpha)-STABILITY for S = P = 2 (detail)

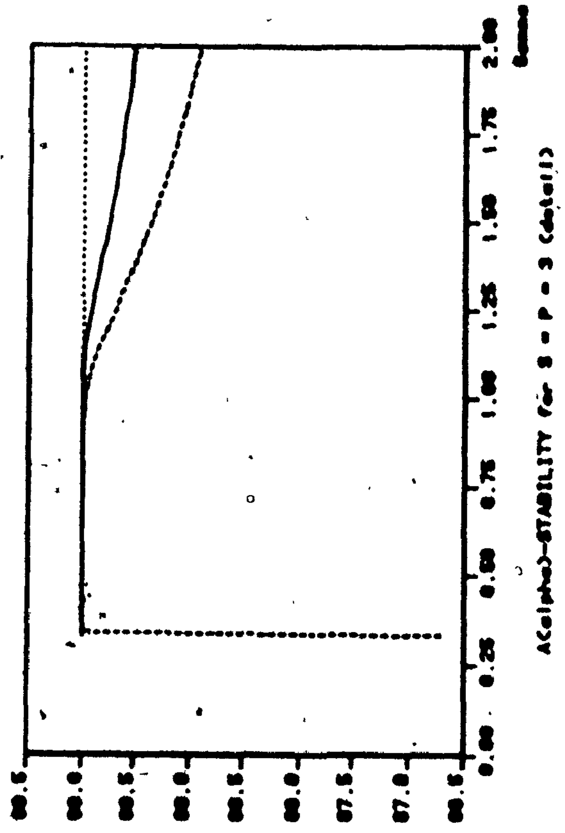


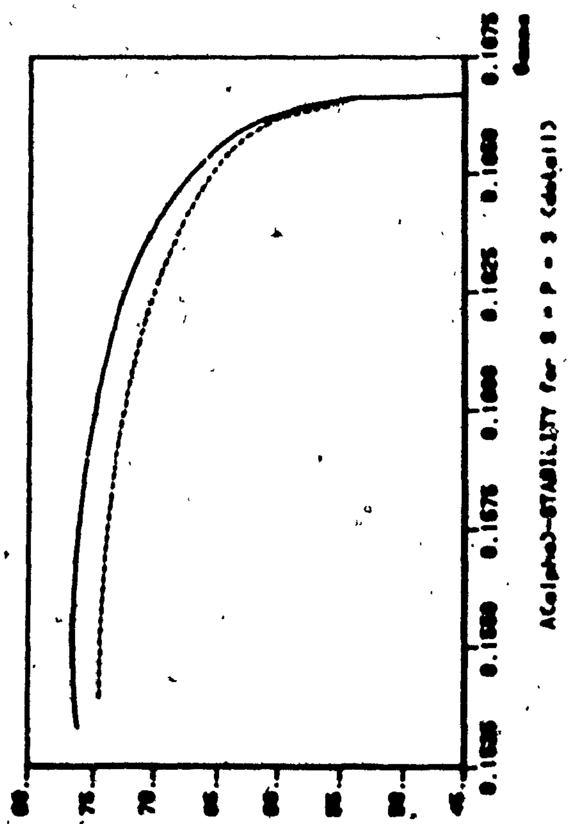
A(alpha)-STABILITY for S = P = 2 (detail)

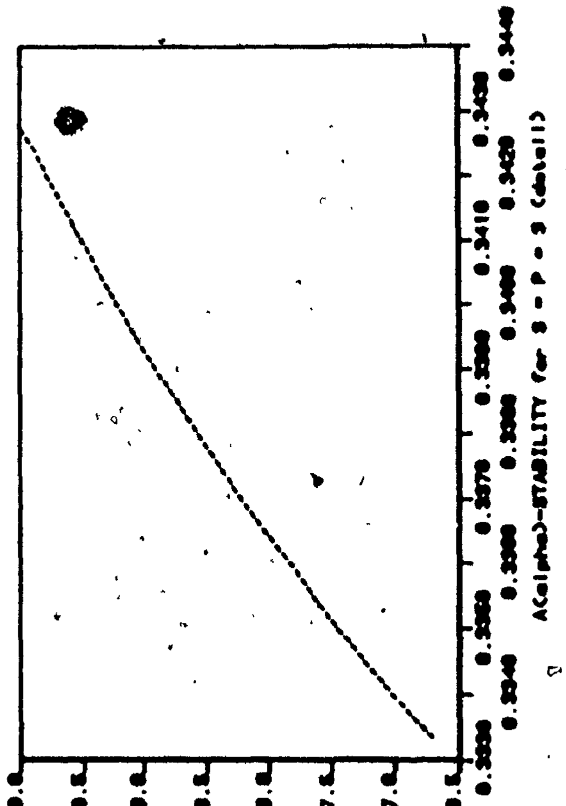


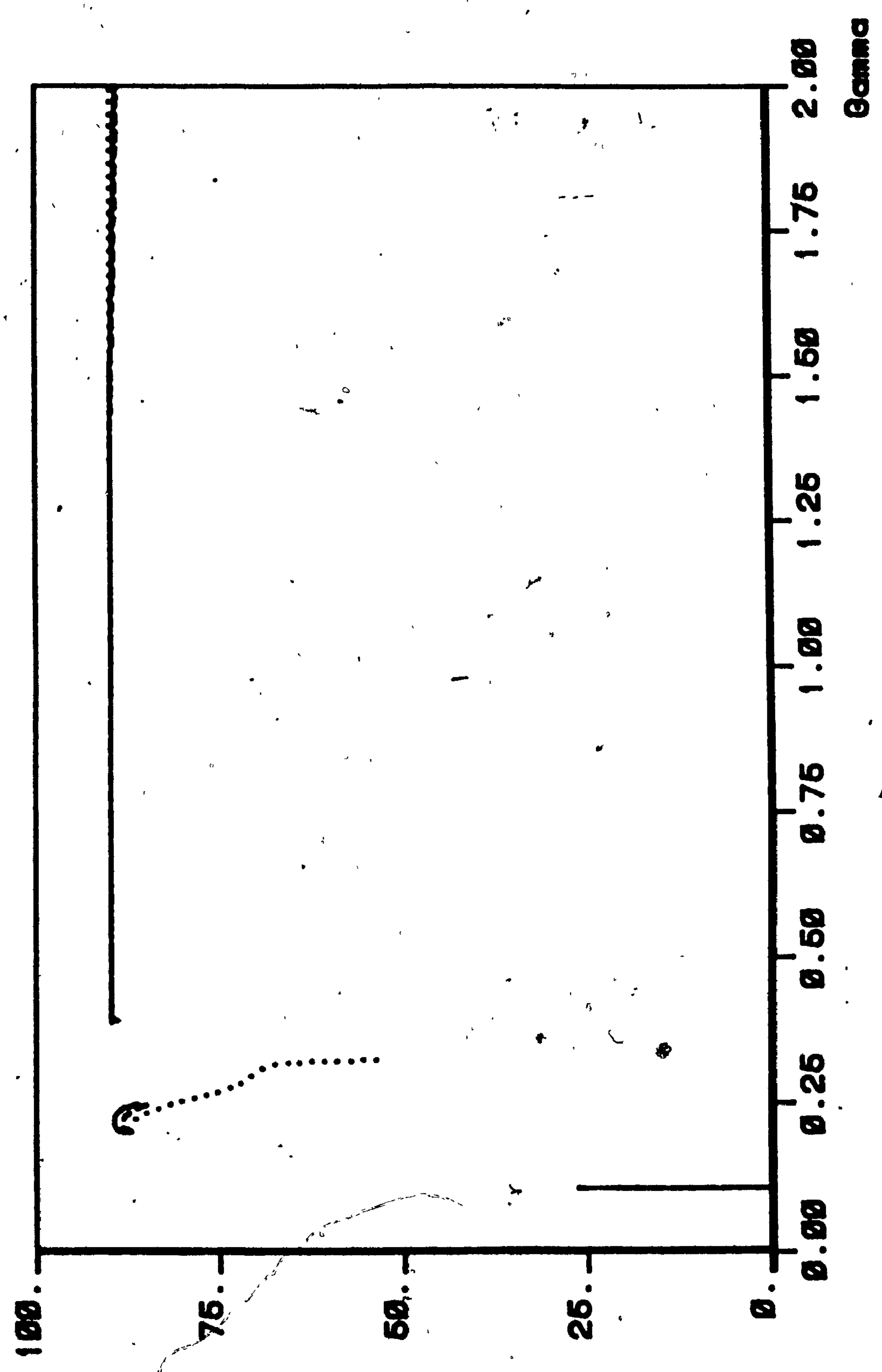
A(alpha)-STABILITY for S = P = 3


A(alpha)-STABILITY for S = P = 3 (detail)
Gamma

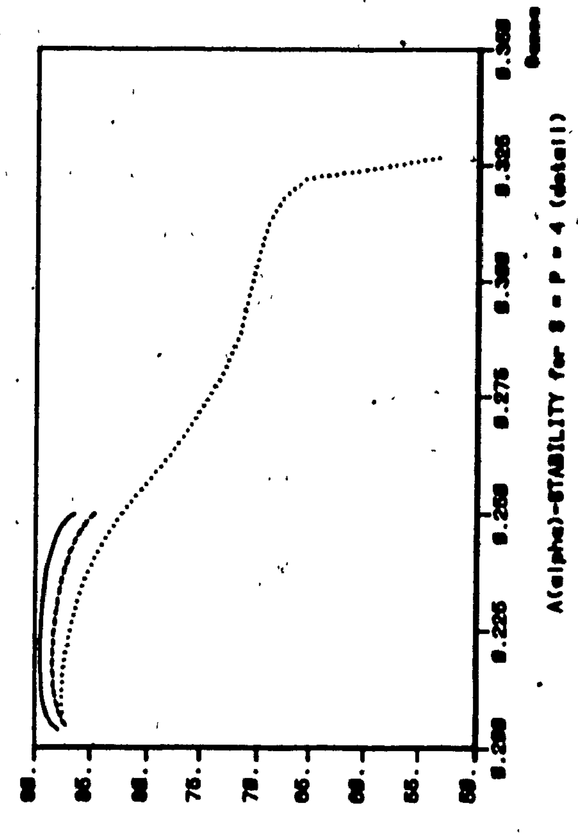

A(alpha)-STABILITY for S = P = 3 (detail)
Gamma
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

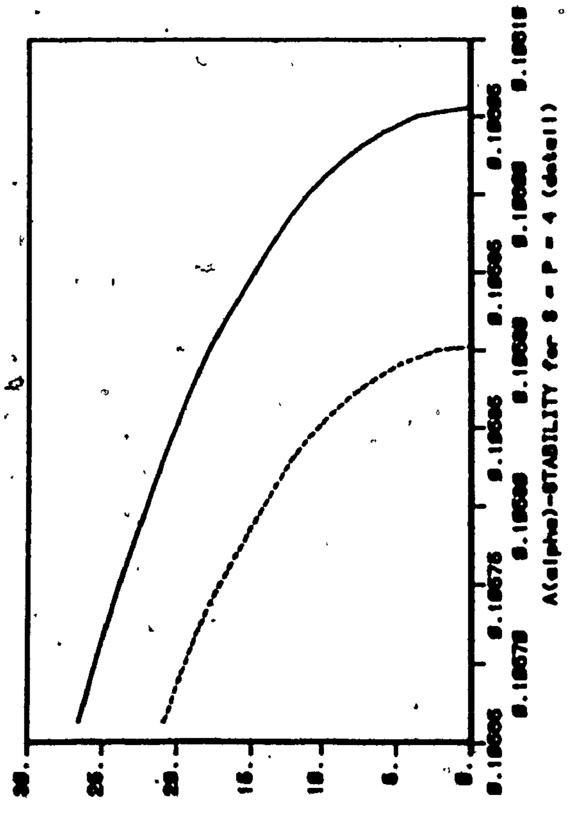

A(alpha)-STABILITY for S = P = 3 (detail)
Gamma
80
75
70
65
60
55
50
45

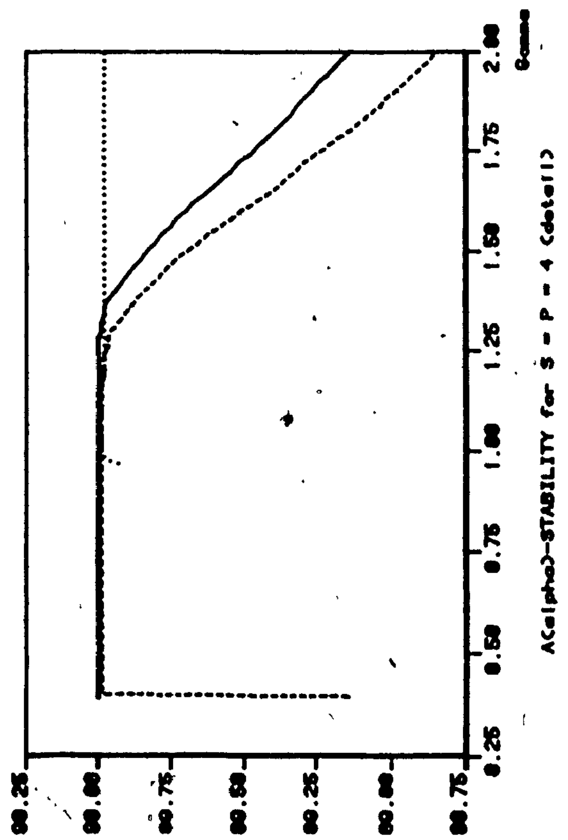

A(alpha)-STABILITY for S = P = 3 (detail)
0.3440

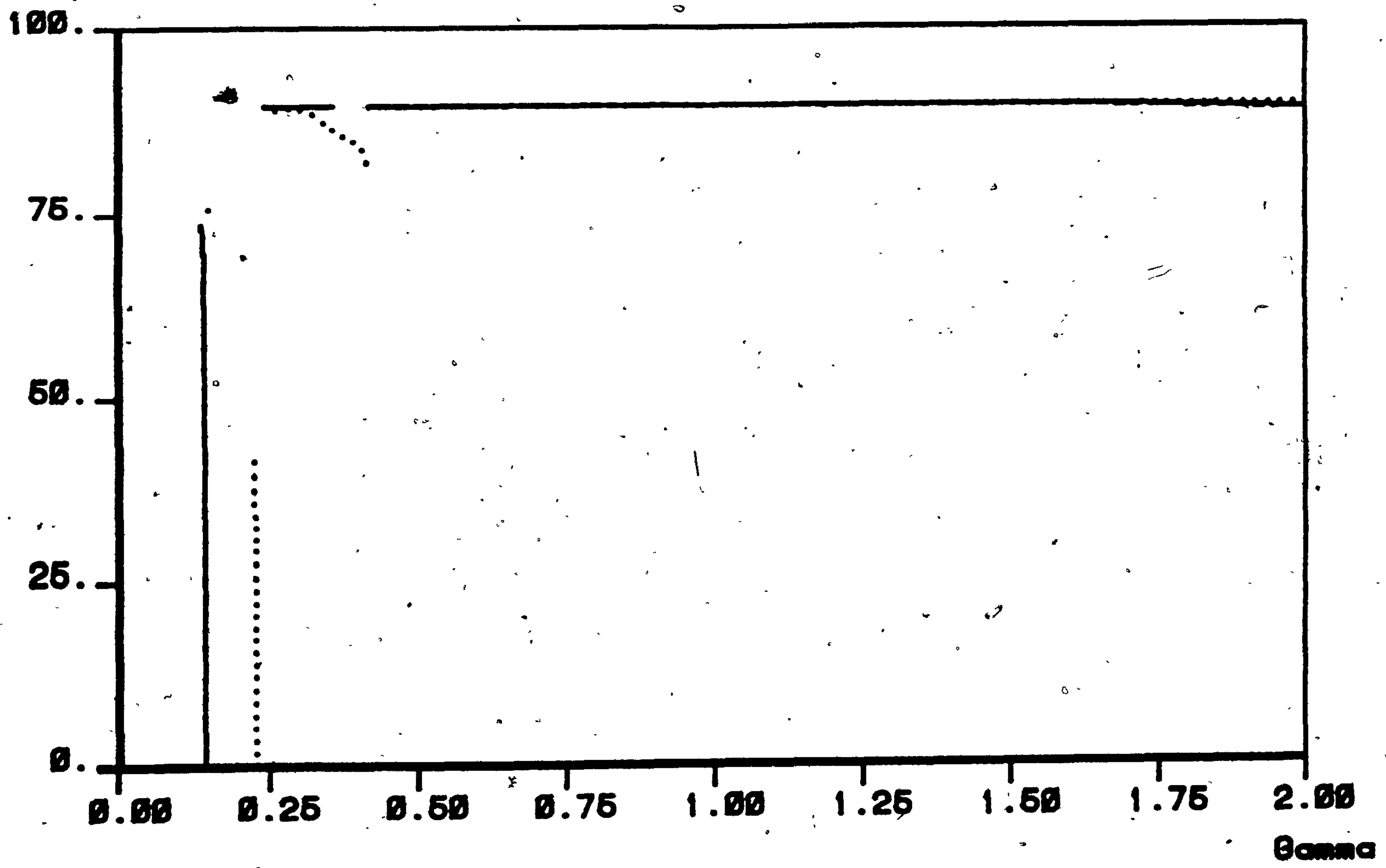


A(alpha)-STABILITY for S = P = 4

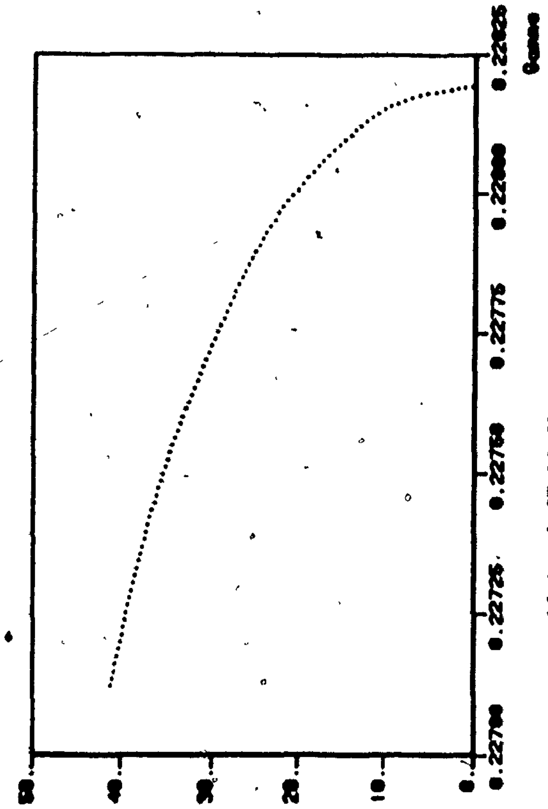

A(alpha)-STABILITY for S = P = 4 (detail)
Gamma
0.200
0.225
0.250
0.275
0.300
0.325
0.350
90
85
80
75
70
65
60
55
50

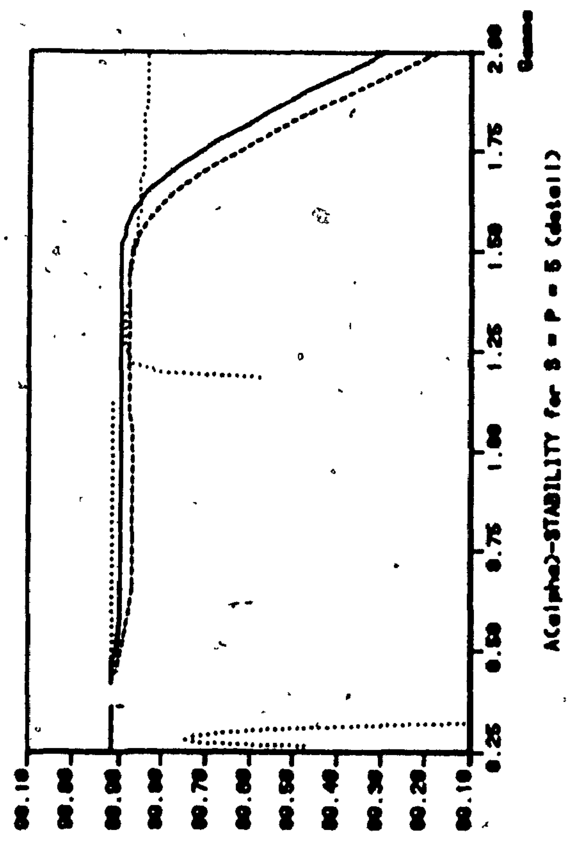

A(alpha)-STABILITY for S = P = 4 (detail)
30
25
20
15
10
5
0

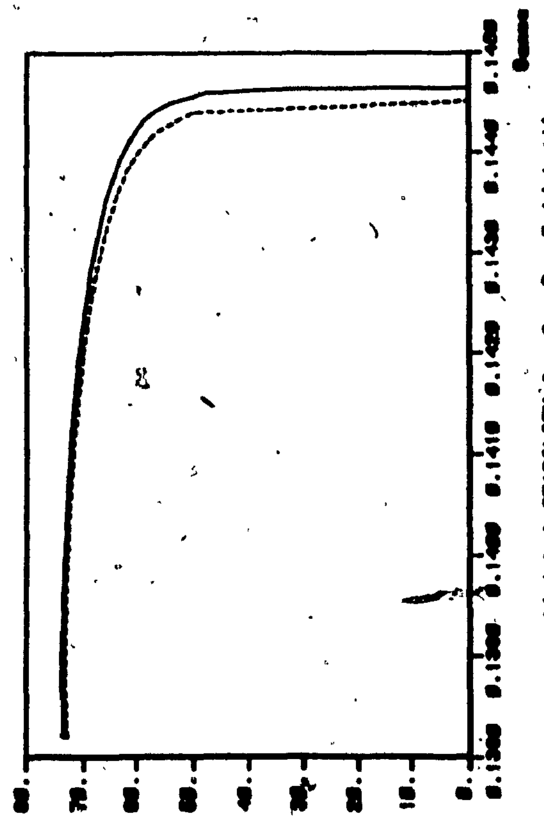

A(alpha)-STABILITY for S = P = 4 (detail)
Gamma
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

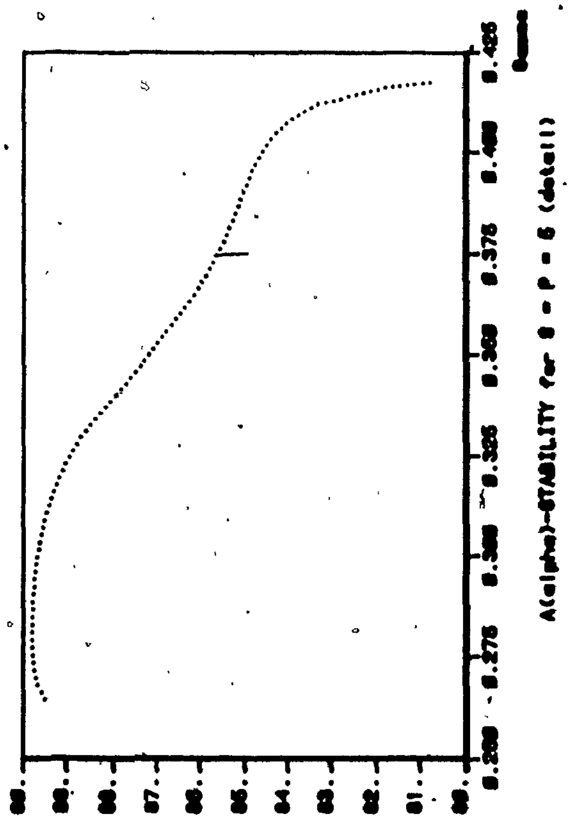


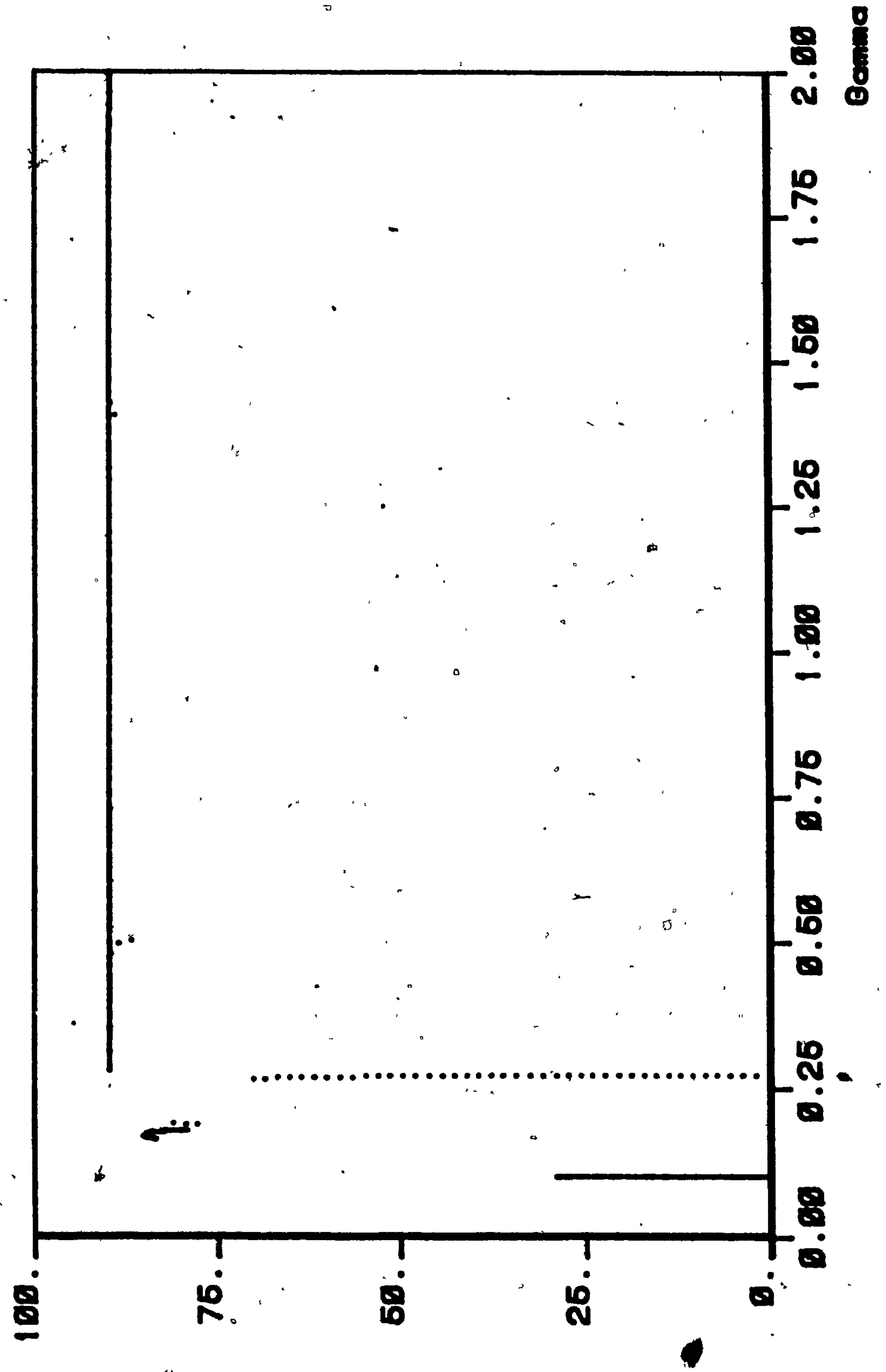
A(alpha)-STABILITY for S = P = 5


A(alpha)-STABILITY for S = P = 5 (detail)
Gamma
0.22700
0.22725
0.22750
0.22775
0.22800
0.22825
50.
40.
30.
20.
10.
0.

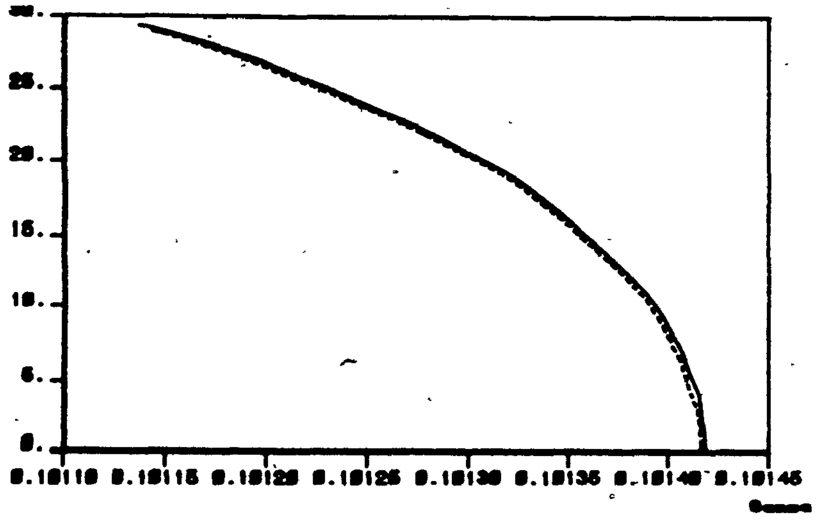

A(alpha)-STABILITY for S = P = 5 (detail)
Gamma
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
89.10
89.00
88.90
88.80
88.70
88.60
88.50
88.40
88.30
88.20
88.10


A(alpha)-STABILITY for S = P = 5 (detail)
Gamma
0.1380
0.1390
0.1400
0.1410
0.1420
0.1430
0.1440
0.1450
90.
75.
60.
45.
30.
15.
0.

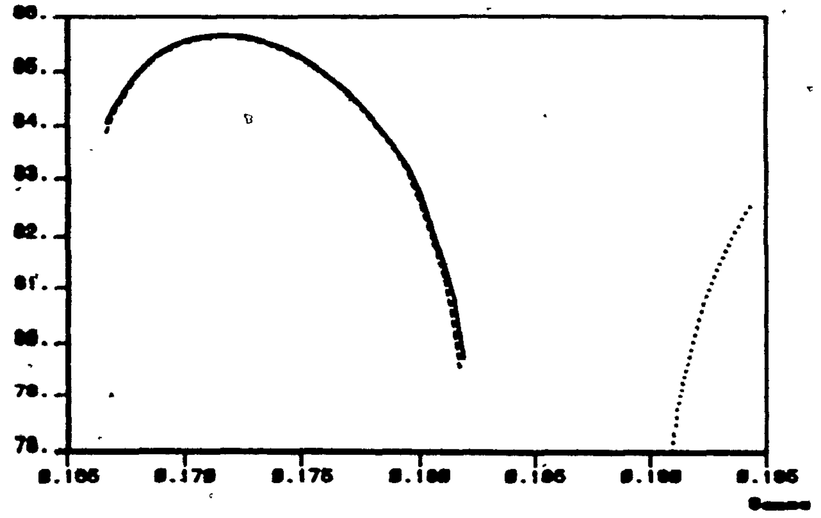

A(alpha)-STABILITY for S = P = 5 (detail)
Gamma
0.250
0.275
0.300
0.325
0.350
0.375
0.400
0.425
90.
89.
88.
87.
86.
85.
84.
83.
82.
81.
80.



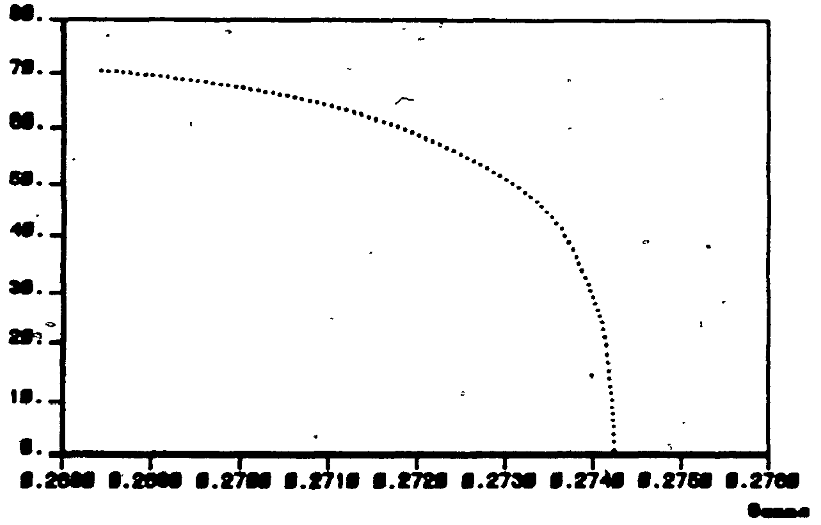
A(alpha)-STABILITY for S = P = 6



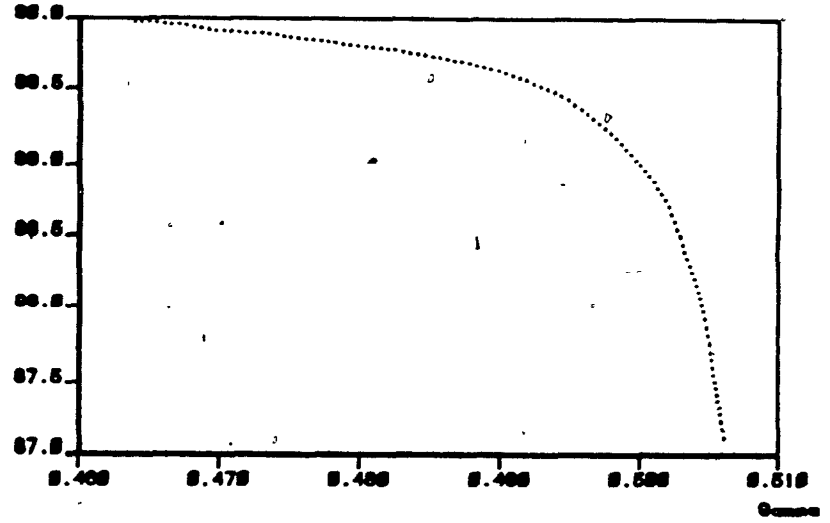
A(alpha)-STABILITY for S = P = 6 (detail)



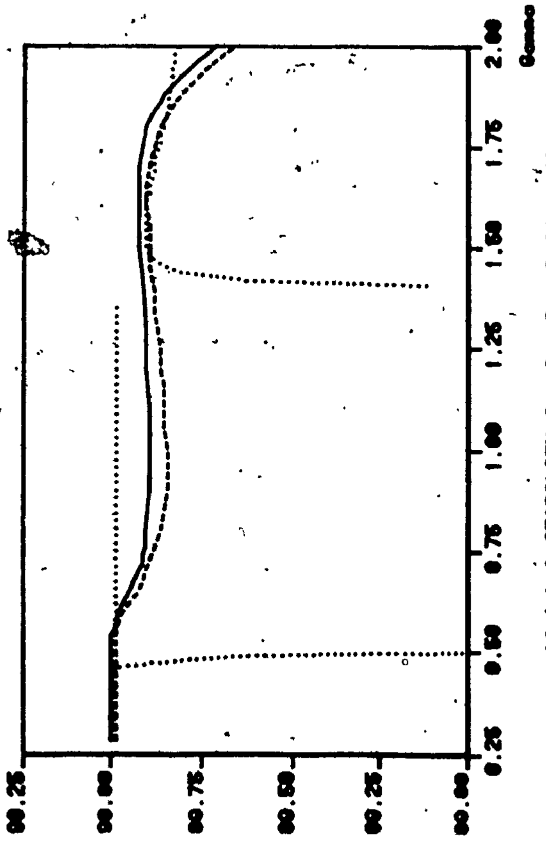
A(alpha)-STABILITY for S = P = 6 (detail)



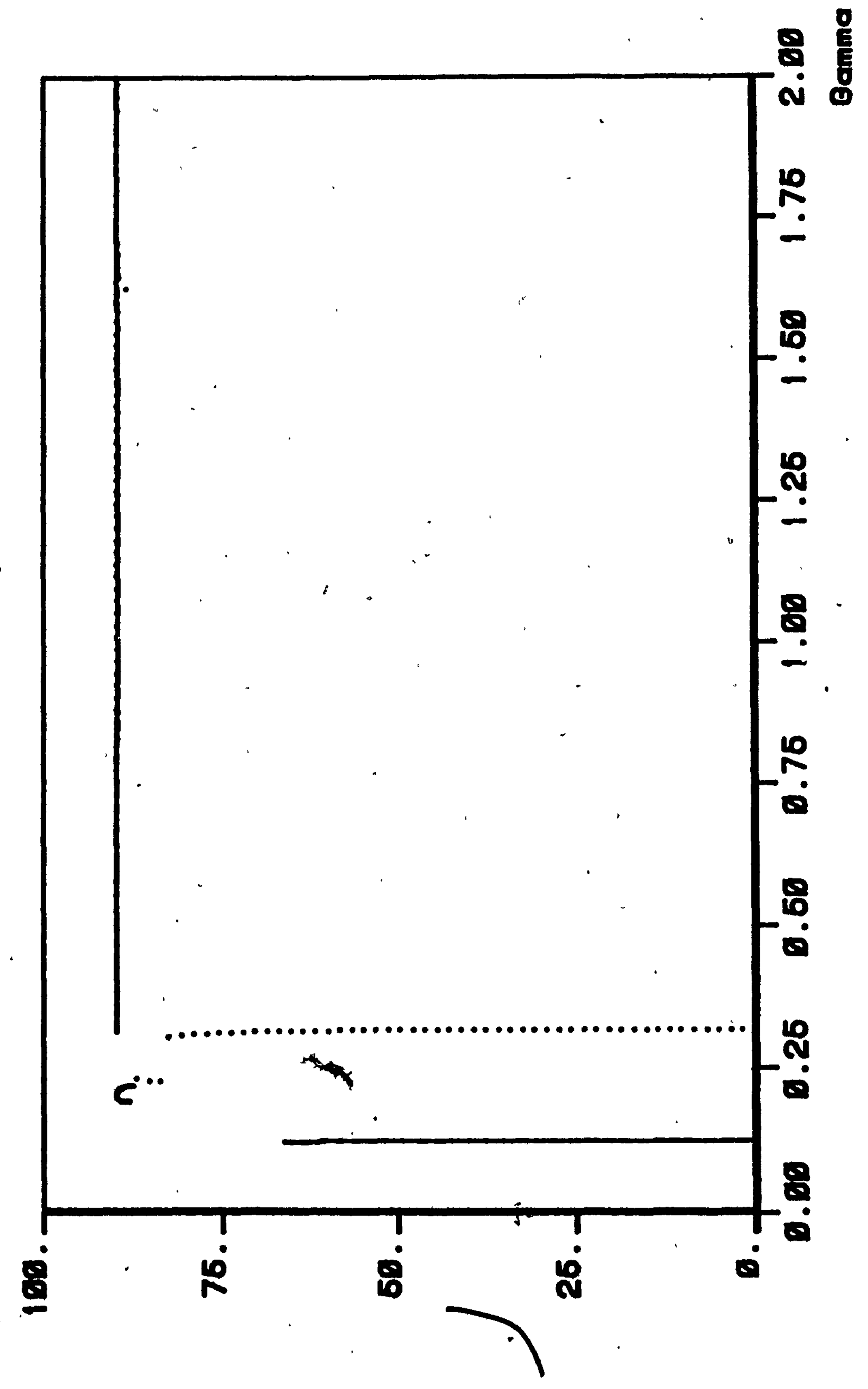
A(alpha)-STABILITY for S = P = 6 (detail)



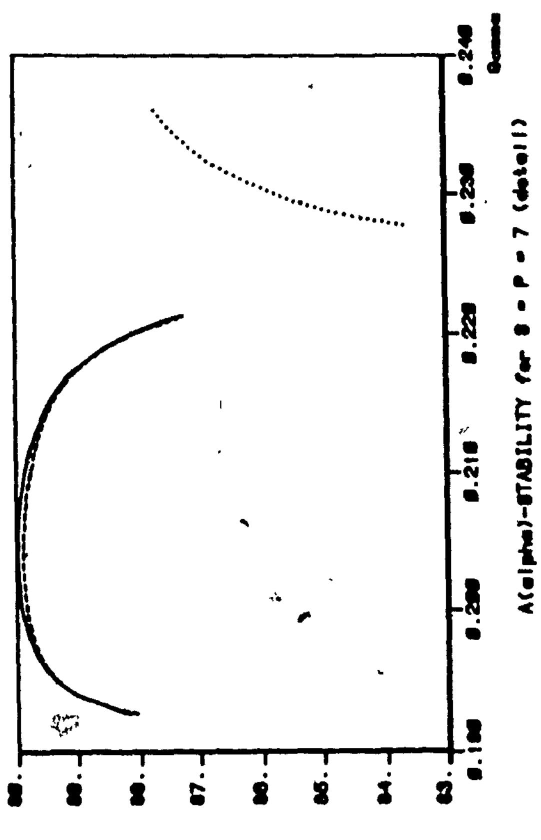
A(alpha)-STABILITY for S = P = 6 (detail)

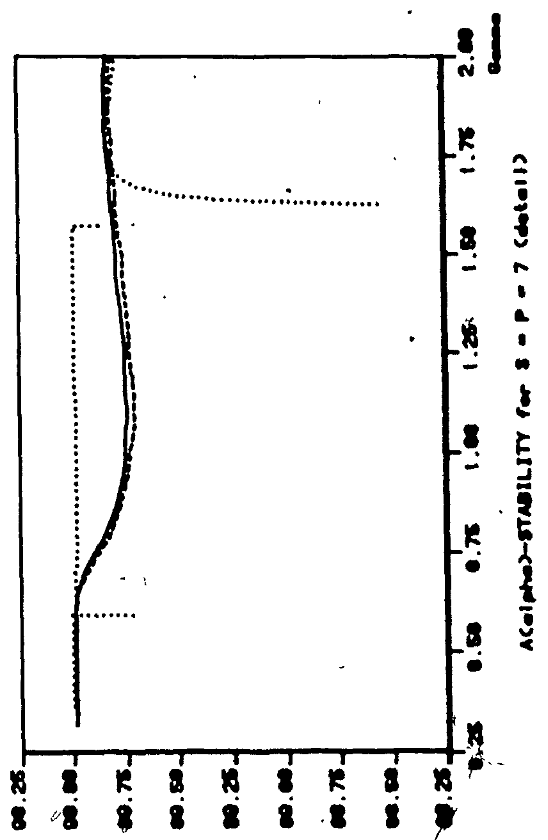


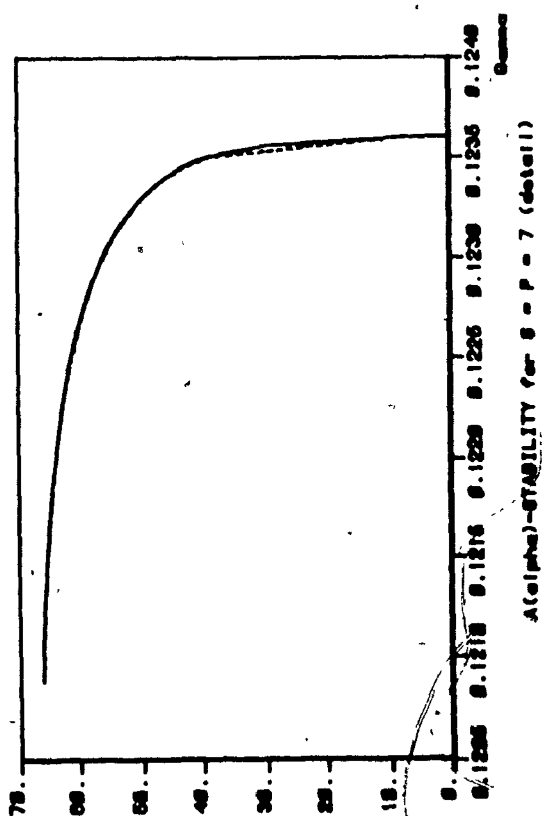
A(alpha)-STABILITY for S = P = 0 (detail)

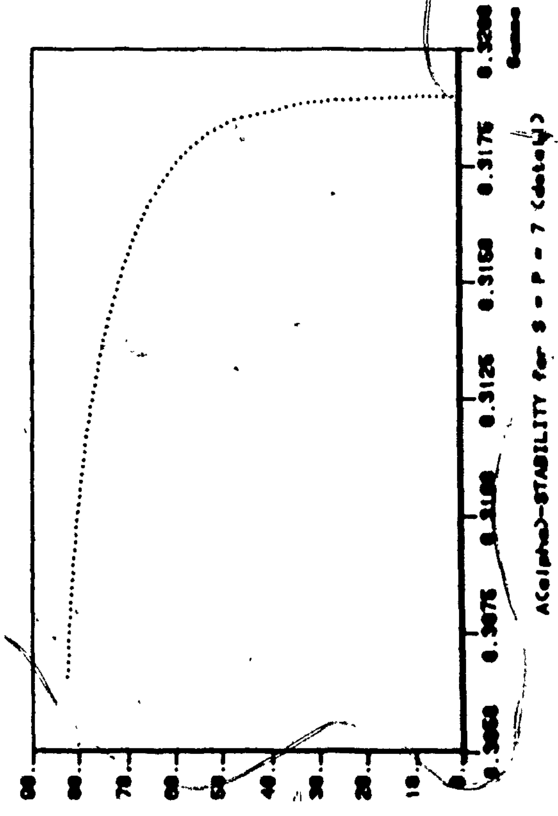


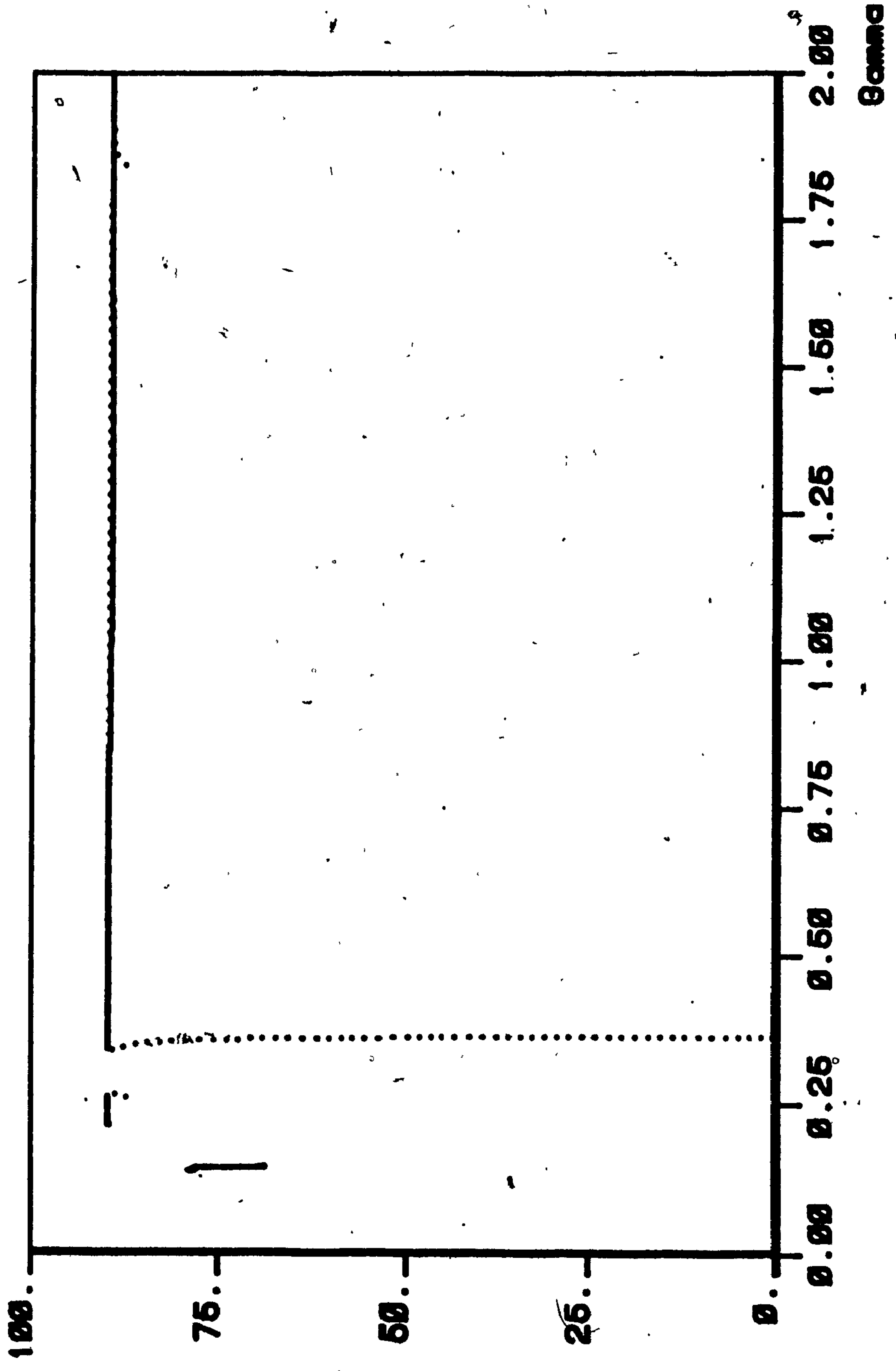
A(alpha)-STABILITY for S = P = 7


A(alpha)-STABILITY for S = P = 7 (detail)
Gamma

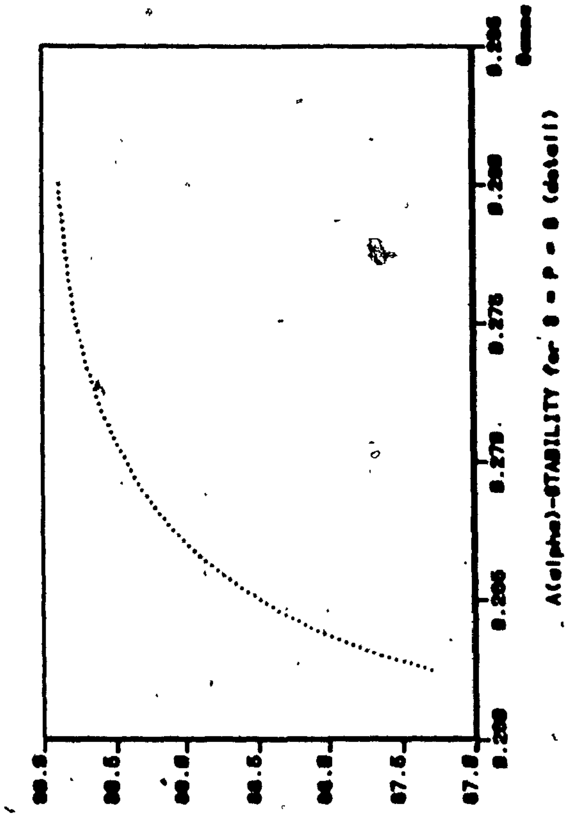

A(alpha)-STABILITY for S = P = 7 (detail)
Gamma

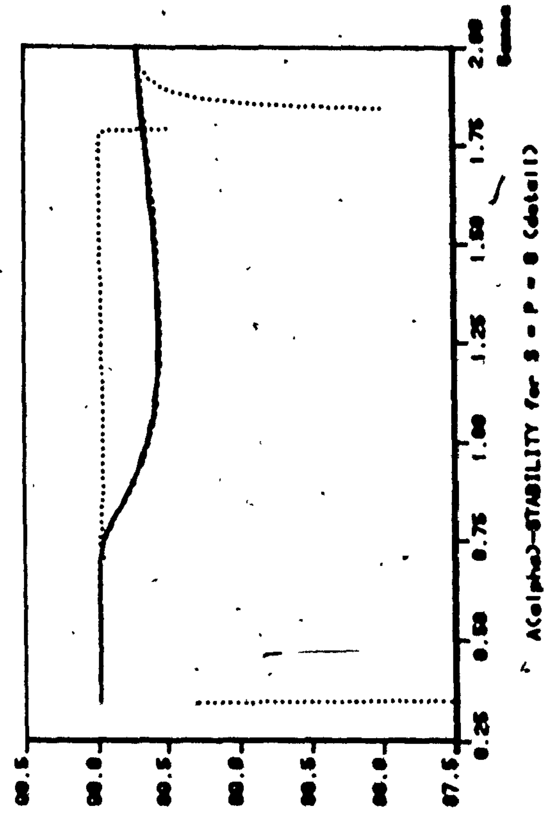

A(alpha)-STABILITY for S = P = 7 (detail)
Gamma

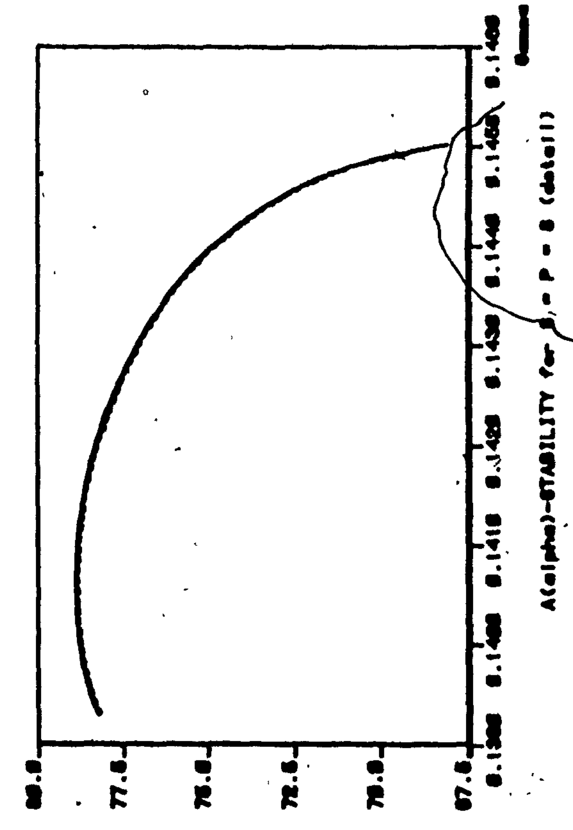

A(alpha)-STABILITY for S = P = 7 (detail)
Gamma

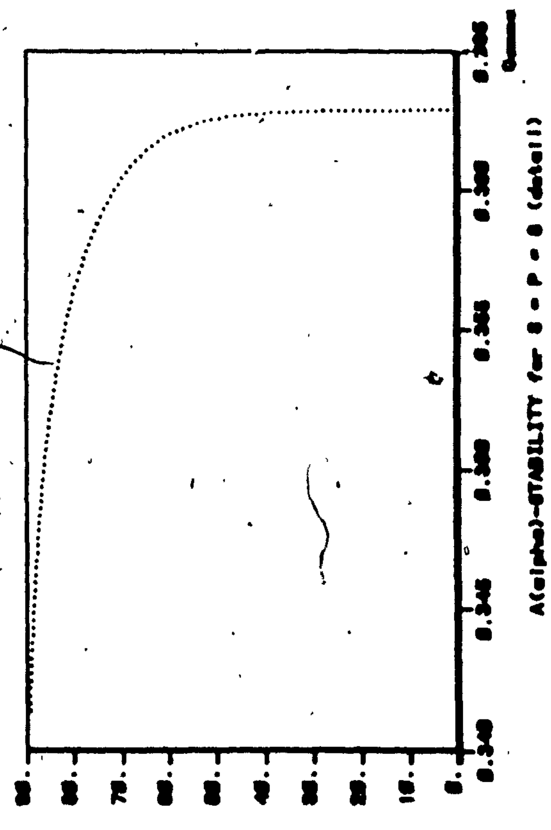


A(alpha)-STABILITY for S = P = 8


A(alpha)-STABILITY for S = P = 8 (detail)
Gamma


A(alpha)-STABILITY for S = P = 8 (detail)
Gamma
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
90.5
90.0
89.5
89.0
88.5
88.0
87.5


A(alpha)-STABILITY for S = P = 8 (detail)
Gamma
80.0
77.5
75.0
72.5
70.0
67.5

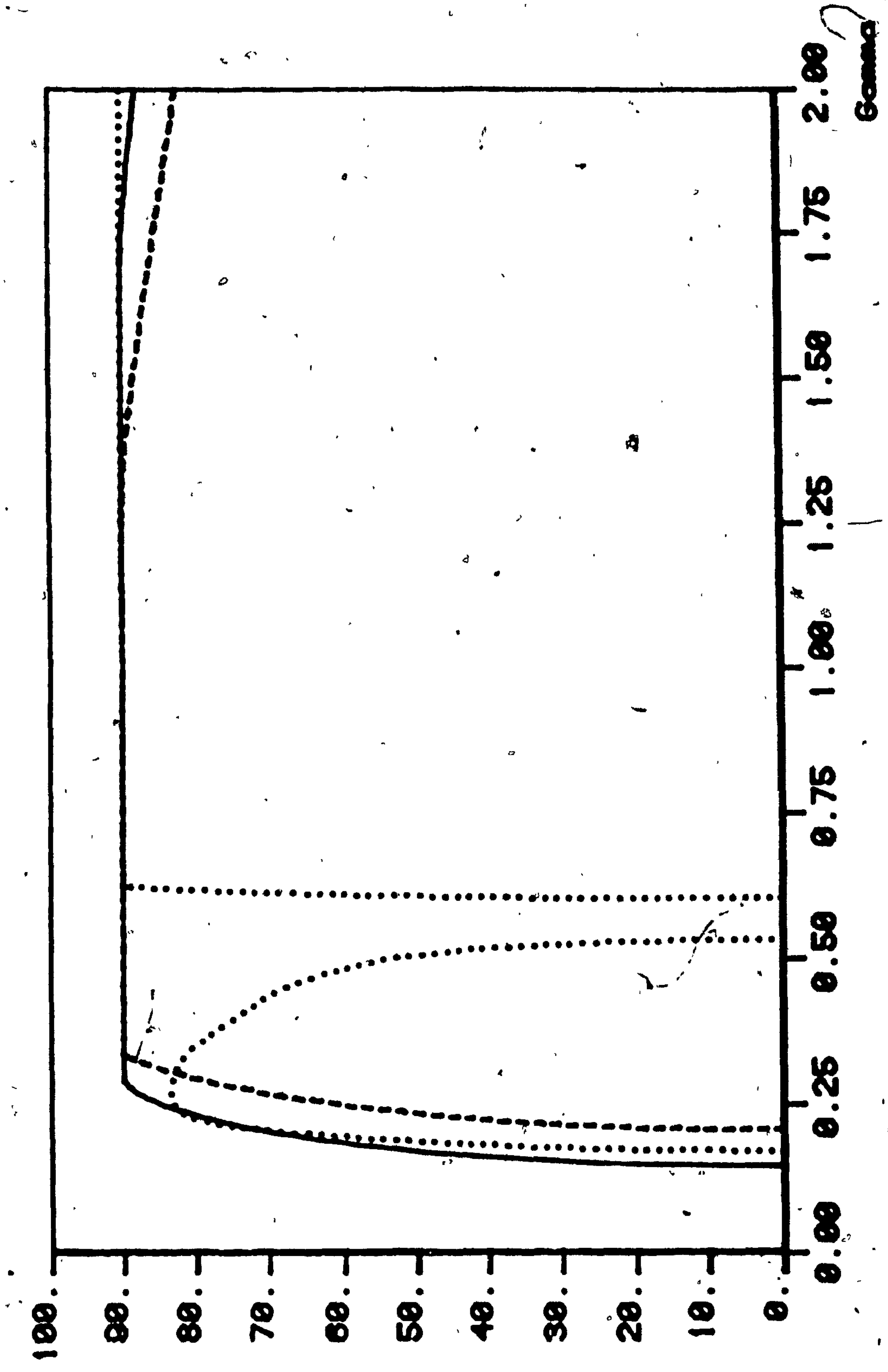

A(alpha)-STABILITY for S = P = 8 (detail)
Gamma

APPENDIX 6

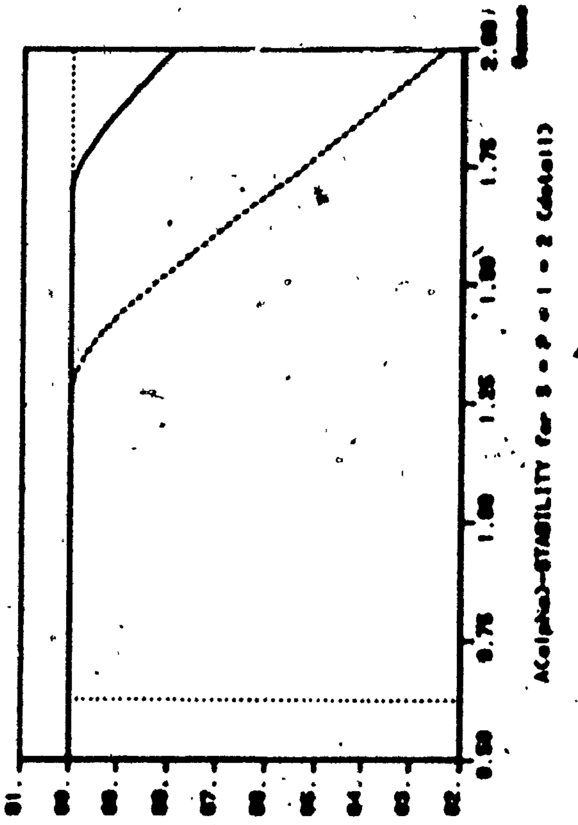
Maximum A(α)-Stability (p=s-1)

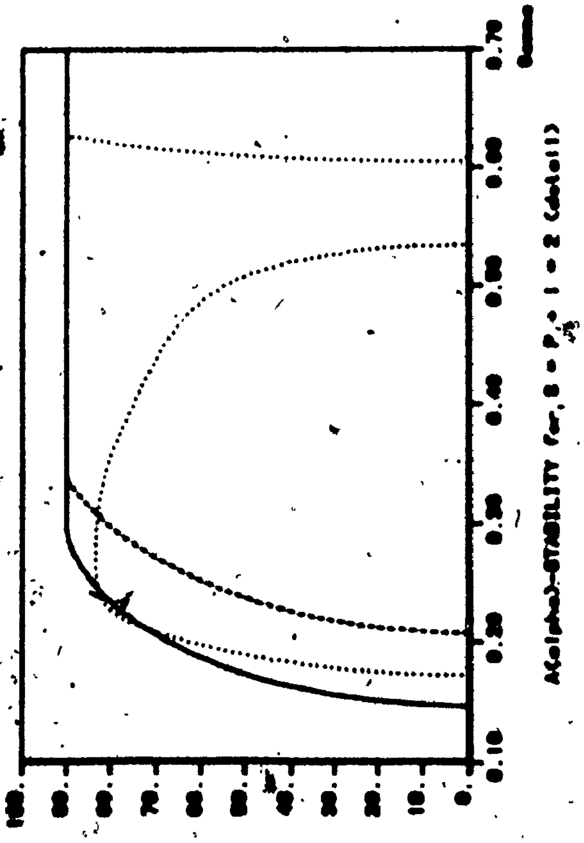
For the case p=(s-1), this appendix presents plots of the maximum angle α (in degrees) for which we have A(α)-stability, as a function of γ for $\gamma \in [0.0,2.0]$. The *solid line* provides the basic method results; the *dashed line* and the *dotted line* are for Richardson extrapolation and modified Richardson extrapolation results.

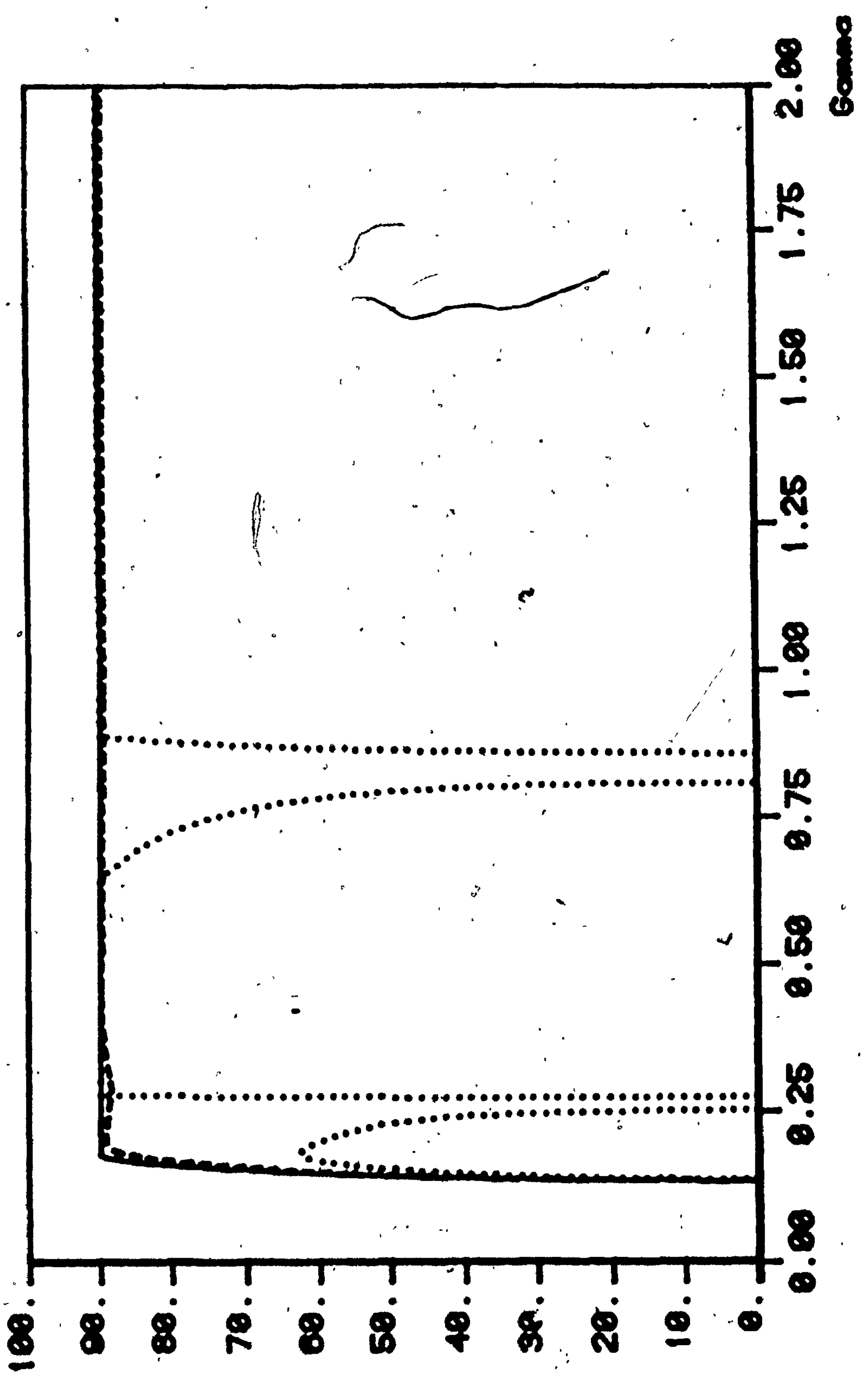
For each value of p, a series of graphs is provided. Generally, the first one presents the overall information. It is followed by many more plots, showing enlargements of certain subregions, so that more accurate information can be visually obtained if these subregions are of particular interest. The last plot of each series is an enlargement of the near—90° region.



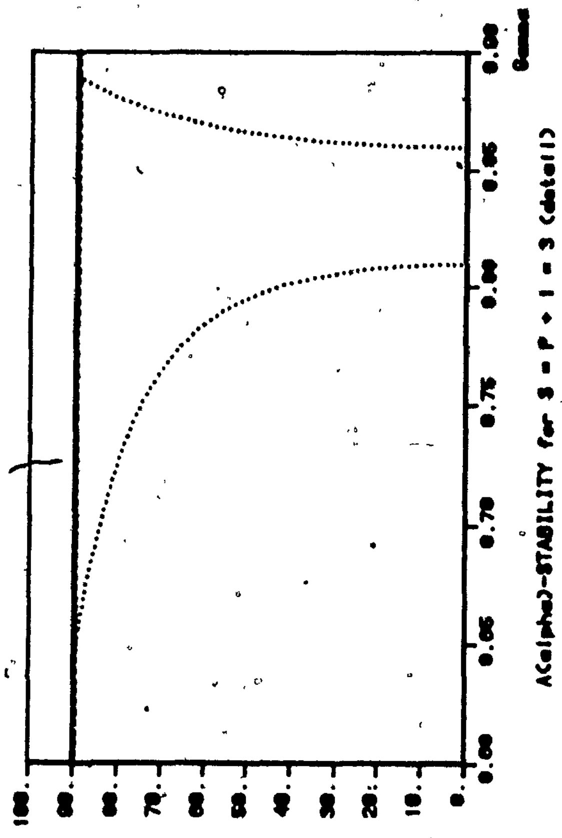
A(alpha)-STABILITY for S = P + 1 = 2

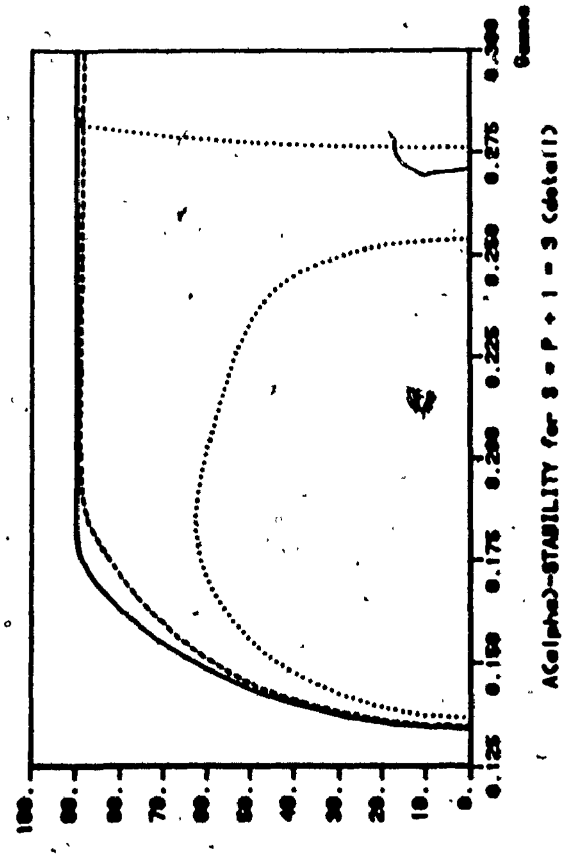

A(alpha)-STABILITY for S = P = I = 2 (detail)
Gamma
0.50
0.75
1.00
1.25
1.50
1.75
2.00
91
90
89
88
87
86
85
84
83
82

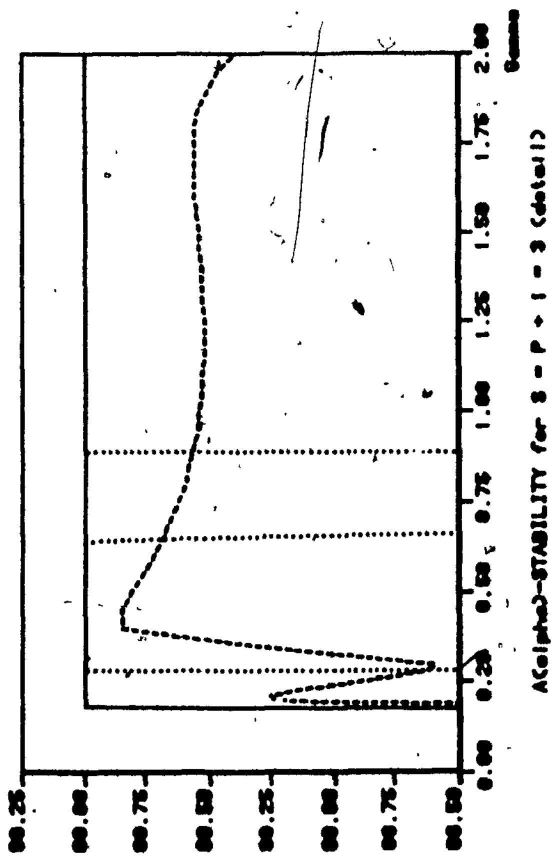

A(alpha)-STABILITY for S = P = I = 2 (detail)
Gamma
0.10
0.20
0.30
0.40
0.50
0.60
0.70
100
90
80
70
60
50
40
30
20
10
0

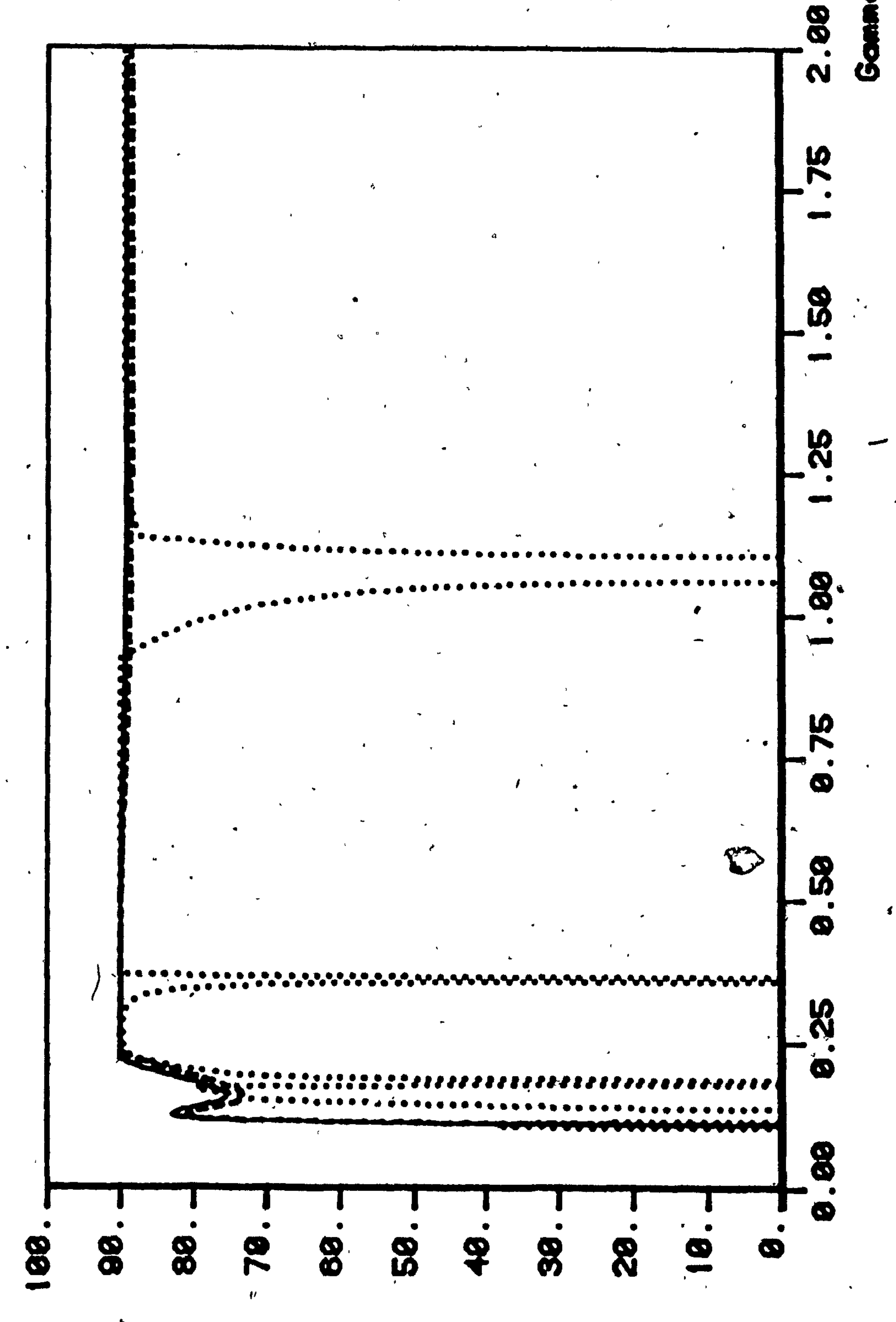


A(alpha)-STABILITY for S = P + I = 3

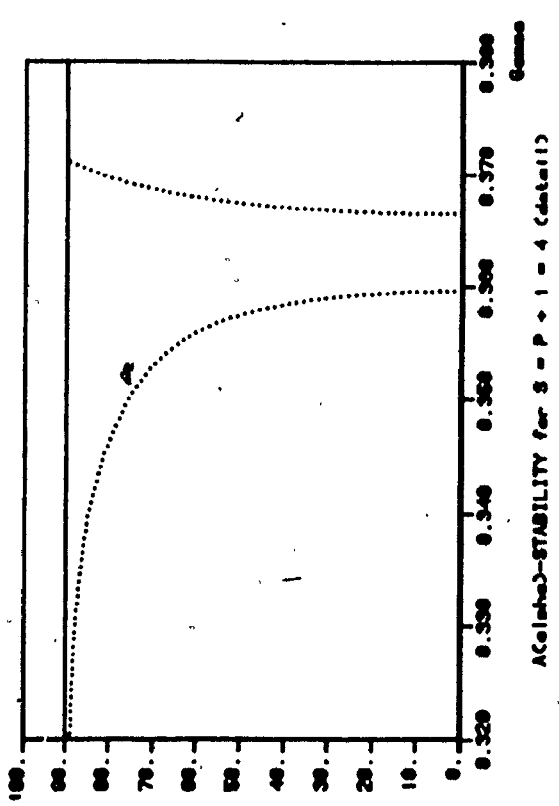

A(alpha)-STABILITY for S = P + 1 = 3 (detail)
Gamma
0.60
0.65
0.70
0.75
0.80
0.85
0.90
100.
90.
80.
70.
60.
50.
40.
30.
20.
10.
0.


A(alpha)-STABILITY for S = P + 1 = 3 (detail)
Gamma
0.125
0.150
0.175
0.200
0.225
0.250
0.275
0.300
100.
90.
80.
70.
60.
50.
40.
30.
20.
10.
0.

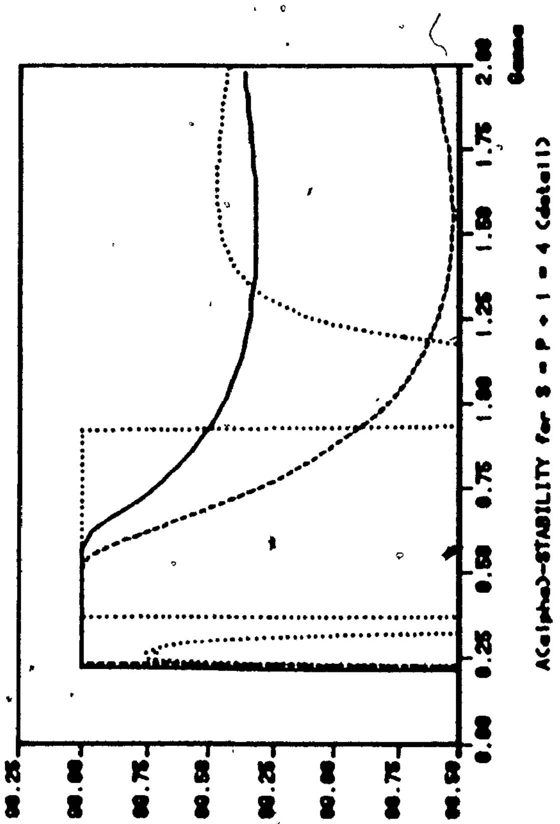

A(alpha)-STABILITY for S = P + 1 = 3 (detail)
Gamma
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
90.25
90.00
89.75
89.50
89.25
89.00
88.75
88.50



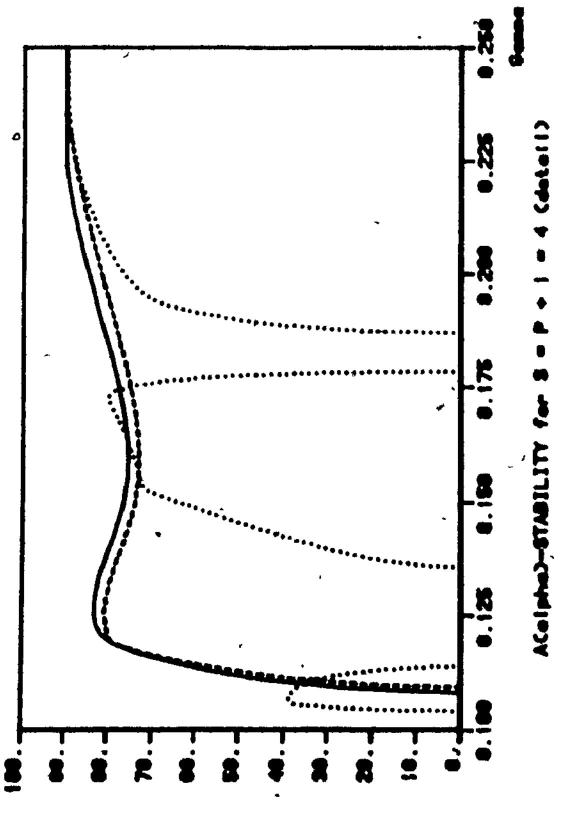
A(alpha)-STABILITY for S = P + 1 = 4



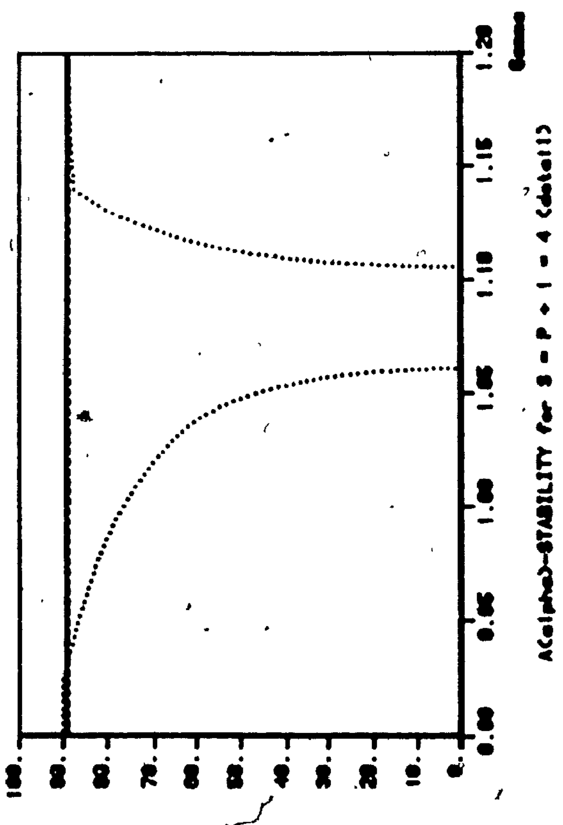
A(alpha)-STABILITY for S = P + 1 = 4 (detail)
Gamma
0.320
0.330
0.340
0.350
0.360
0.370
0.380



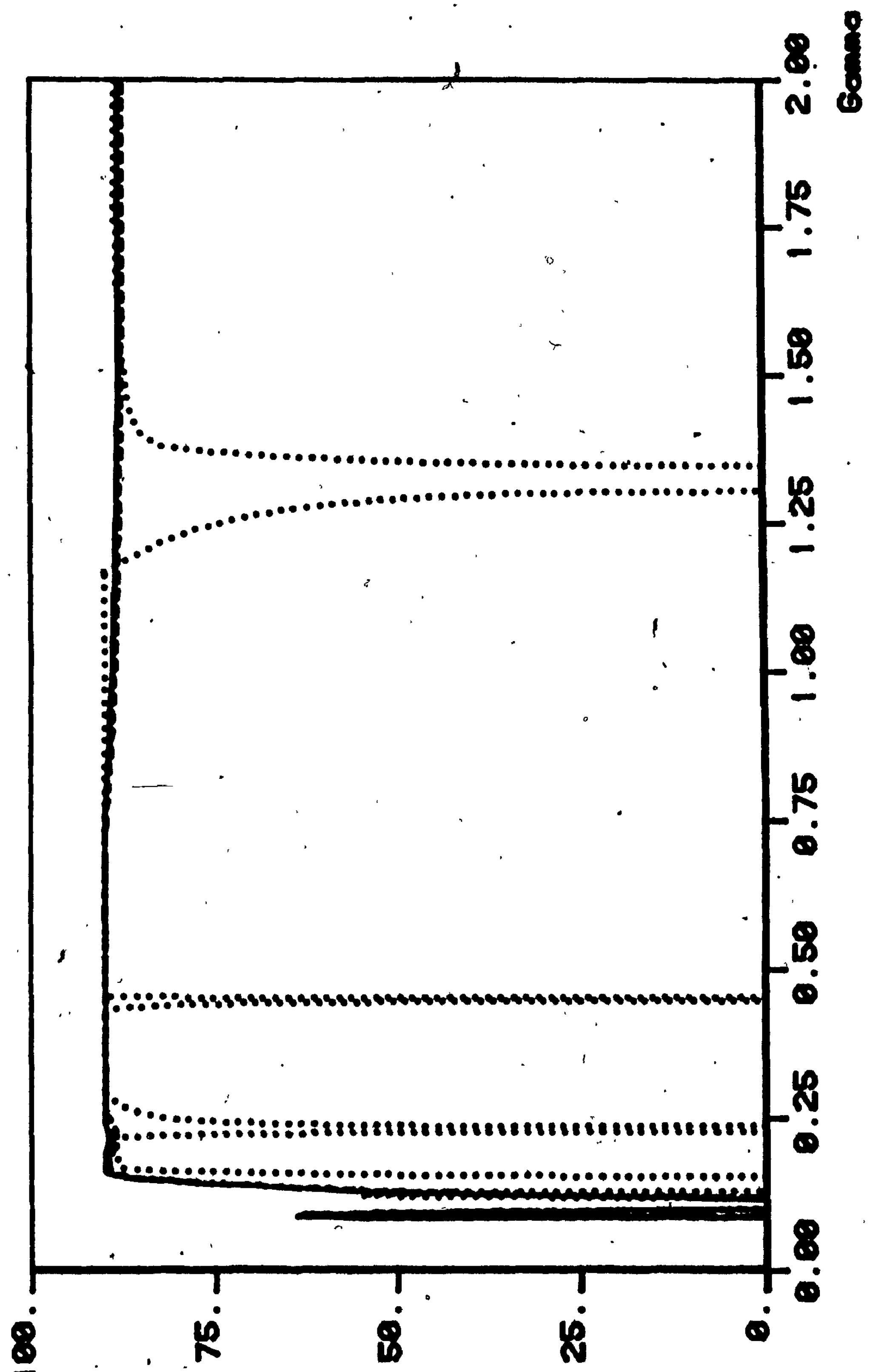
A(alpha)-STABILITY for S = P + 1 = 4 (detail)
Gamma
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00



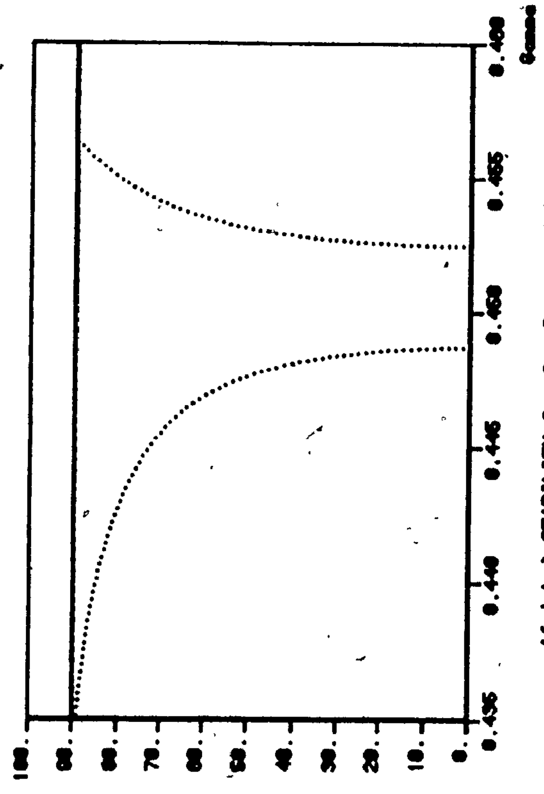
A(alpha)-STABILITY for S = P + 1 = 4 (detail)
Gamma
0.100
0.125
0.150
0.175
0.200
0.225
0.250

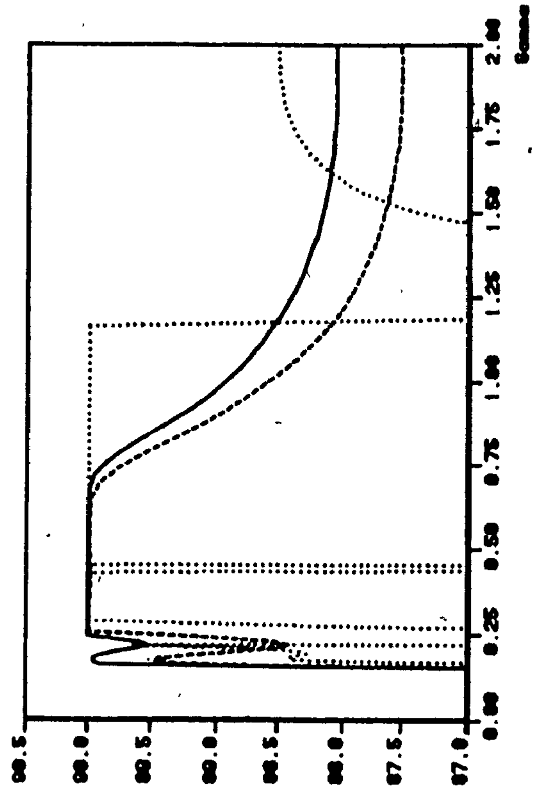


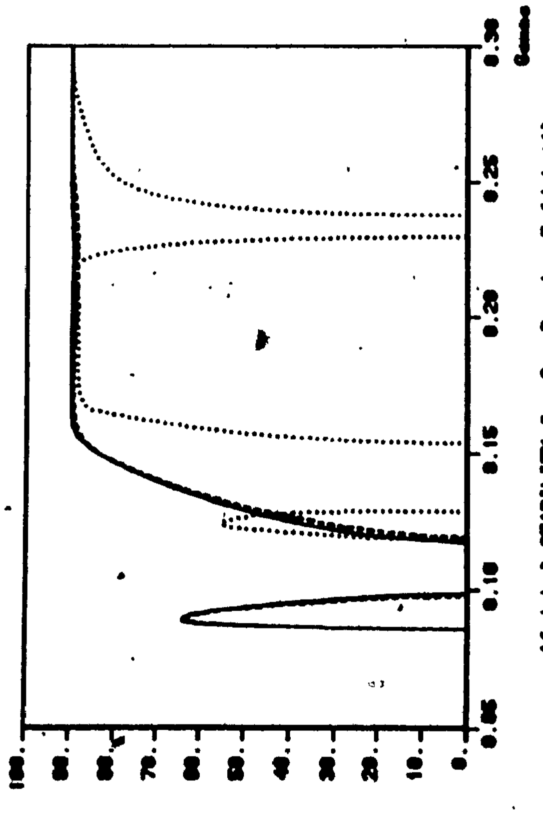
A(alpha)-STABILITY for S = P + 1 = 4 (detail)
Gamma
0.90
0.95
1.00
1.05
1.10
1.15
1.20

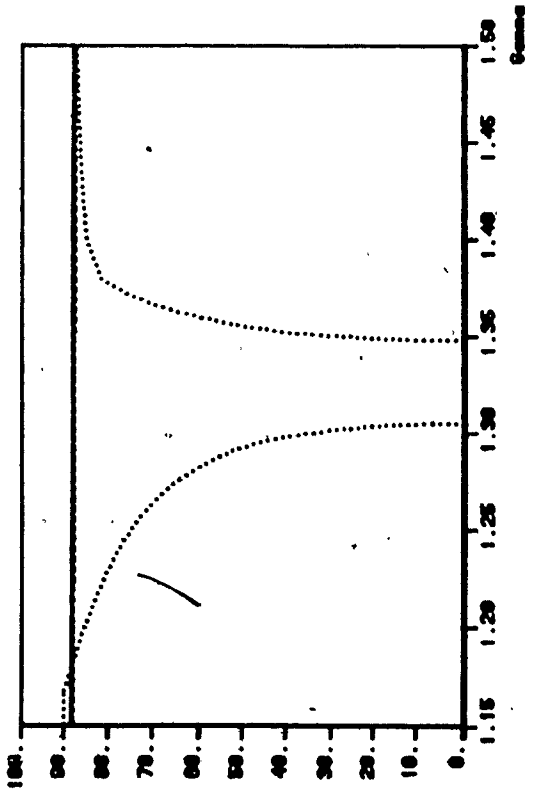


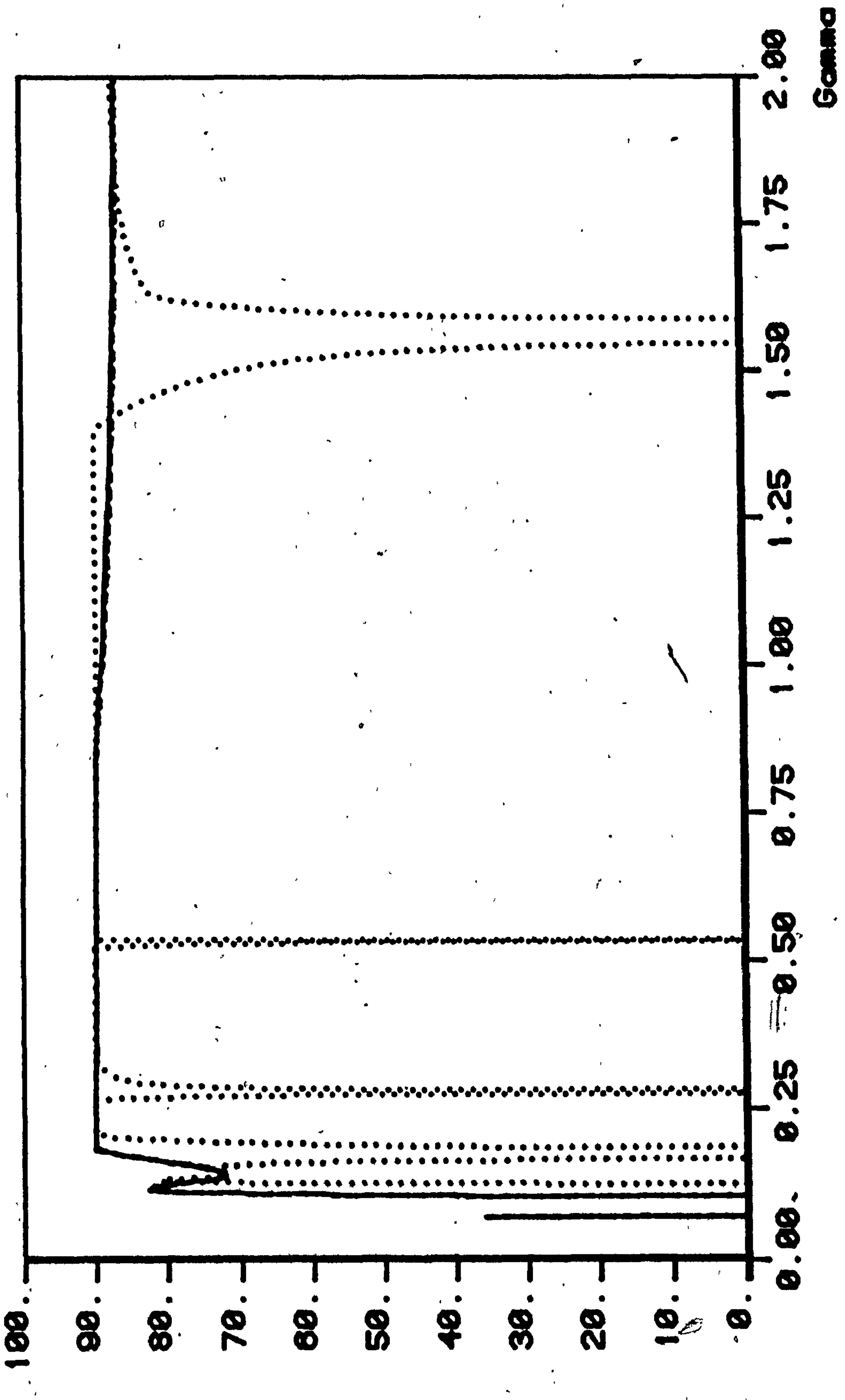
A(alpha)-STABILITY for S = P + 1 = 5


A(alpha)-STABILITY for S = P + 1 = 5 (detail)
Gamma
0.435
0.440
0.445
0.450
0.455
0.460
0.
10.
20.
30.
40.
50.
60.
70.
80.
90.
100.

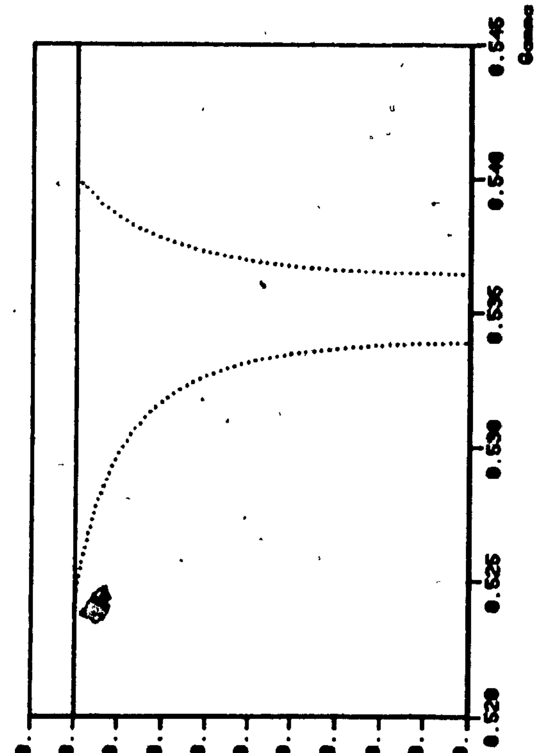

A(alpha)-STABILITY for S = P + 1 = 5 (detail)
Gamma
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
97.0
97.5
98.0
98.5
99.0
99.5
99.0
99.5


A(alpha)-STABILITY for S = P + 1 = 5 (detail)
Gamma
0.05
0.10
0.15
0.20
0.25
0.30
0.
10.
20.
30.
40.
50.
60.
70.
80.
90.
100.

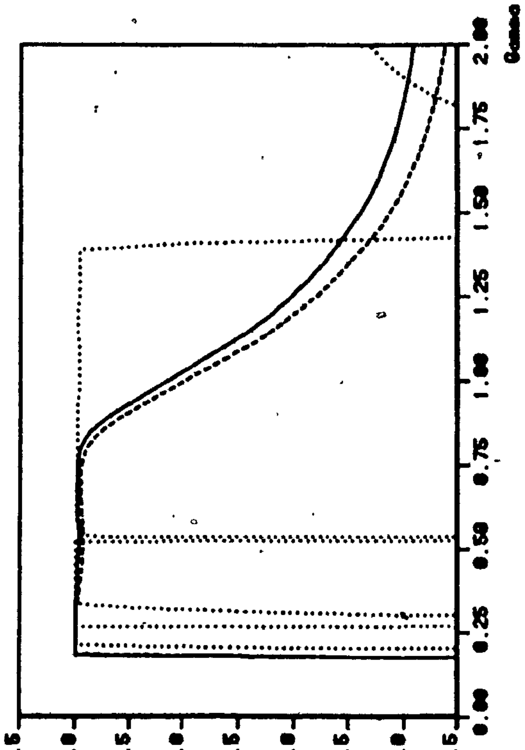

A(alpha)-STABILITY for S = P + 1 = 5 (detail)
Gamma
1.15
1.20
1.25
1.30
1.35
1.40
1.45
1.50
0.
10.
20.
30.
40.
50.
60.
70.
80.
90.
100.



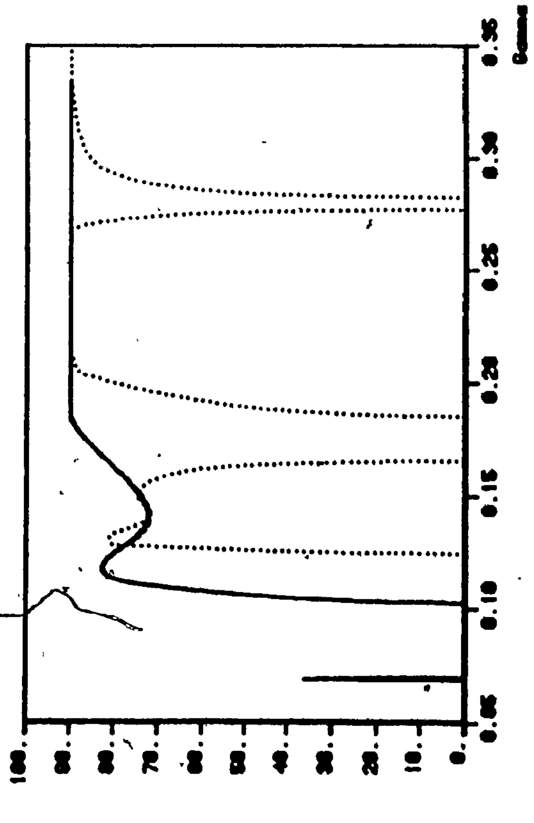
A(alpha)-STABILITY for S = P + I = 6



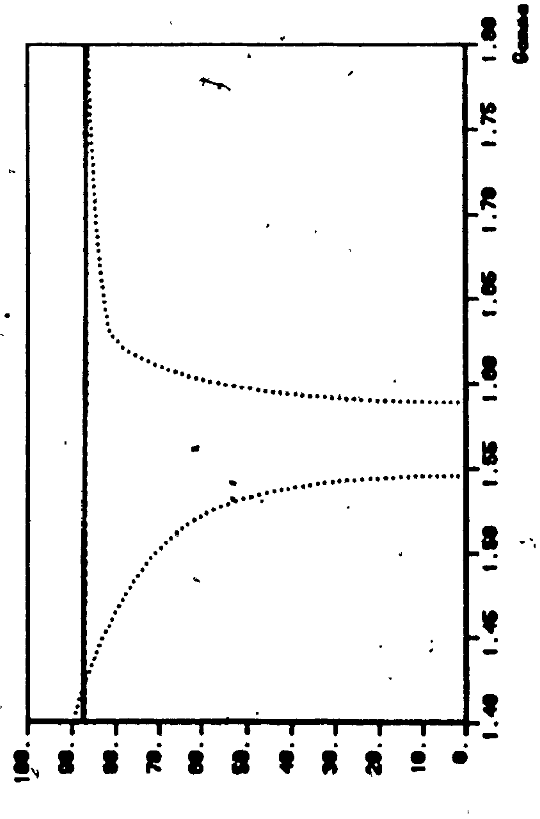
A(alpha)-STABILITY for S = P + 1 = 6 (detail)



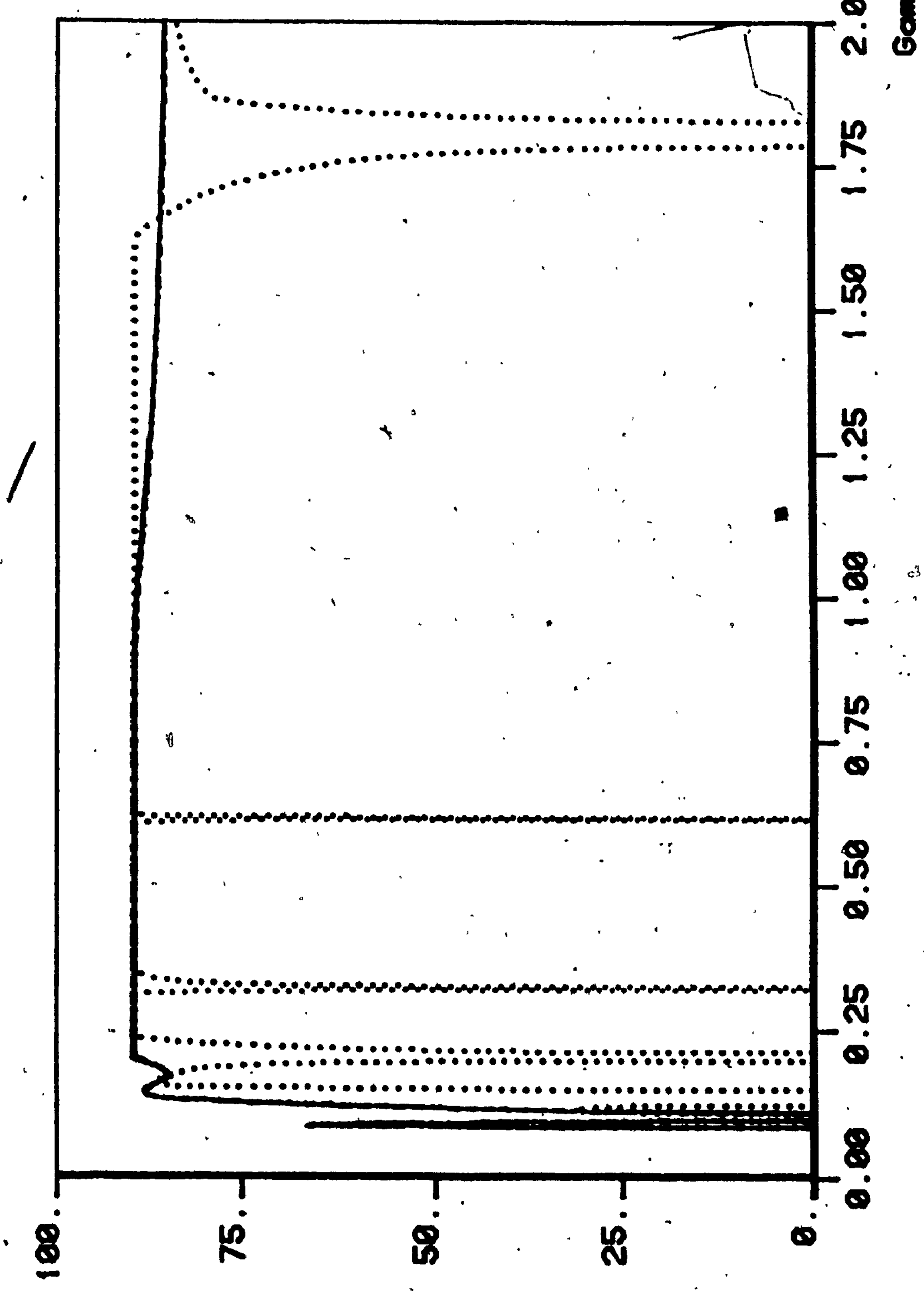
A(alpha)-STABILITY for S = P + 1 = 6 (detail)



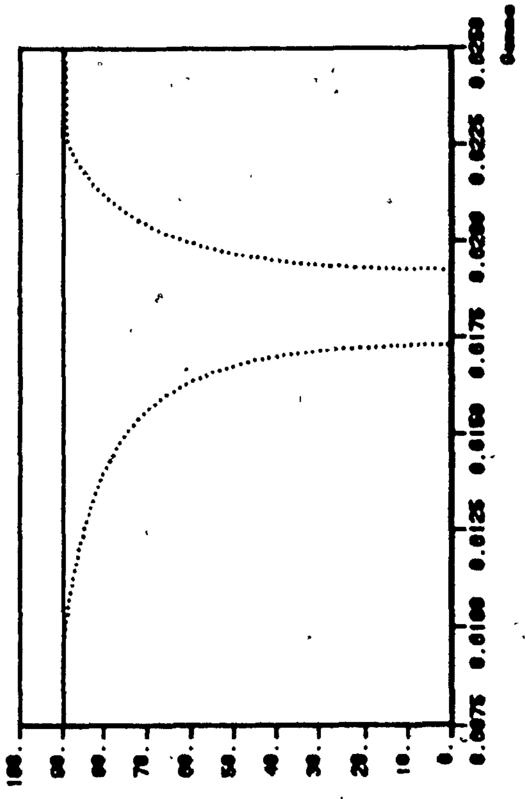
A(alpha)-STABILITY for S = P + 1 = 6 (detail)



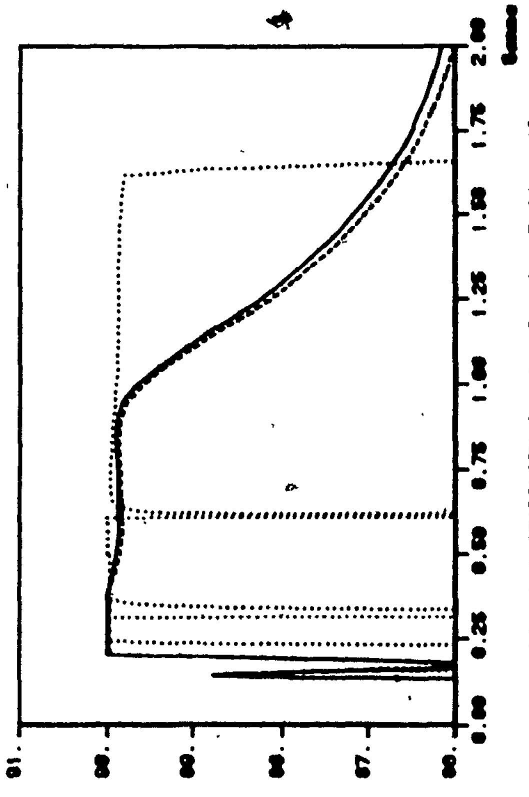
A(alpha)-STABILITY for S = P + 1 = 6 (detail)



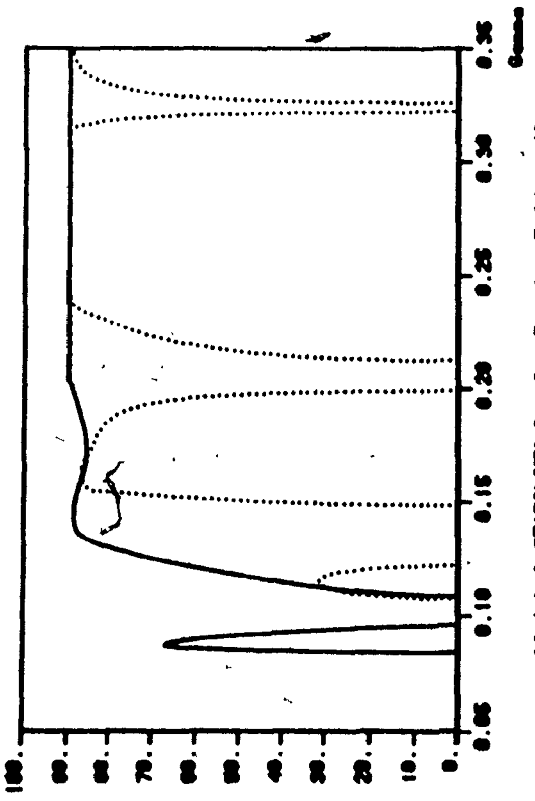
A(alpha)-STABILITY for S = P + 1 = 7



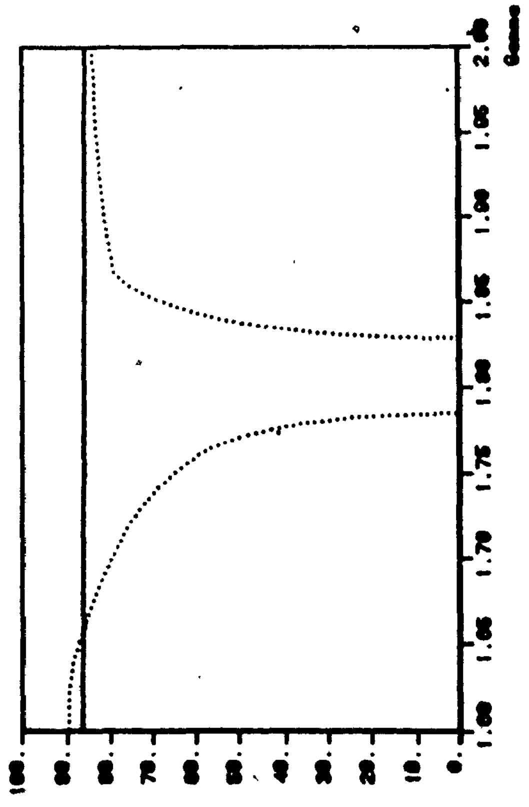
A(alpha)-STABILITY for S = P + 1 = 7 (detail)



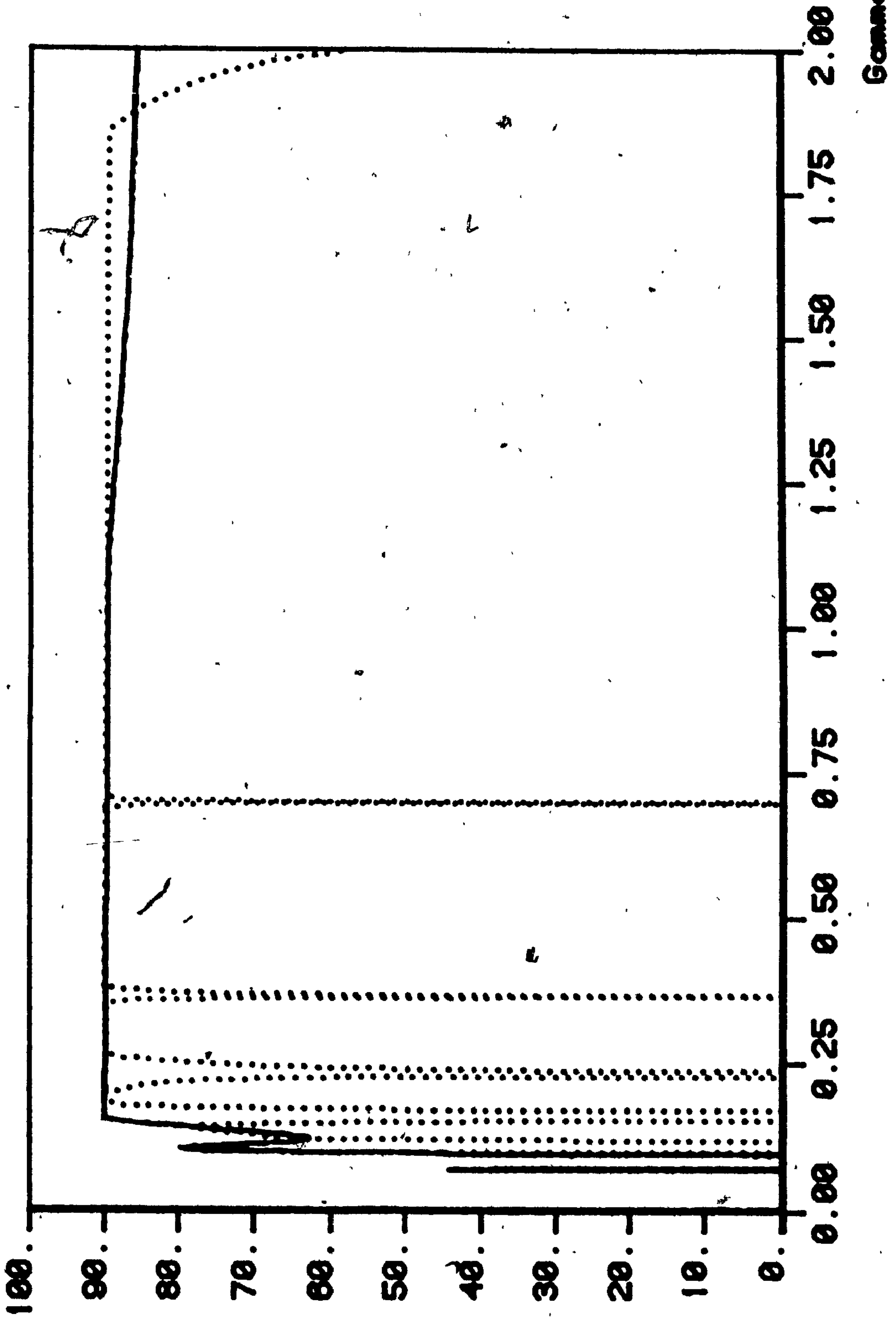
A(alpha)-STABILITY for S = P + 1 = 7 (detail)



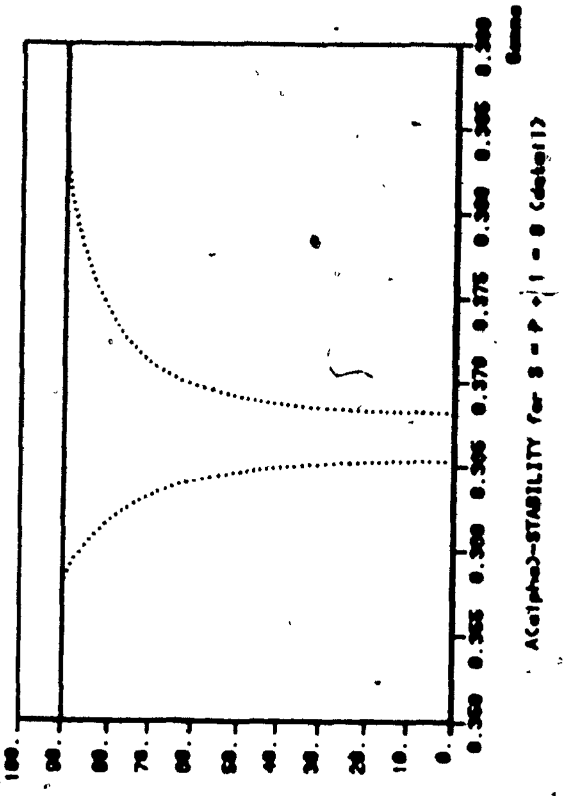
A(alpha)-STABILITY for S = P + 1 = 7 (detail)

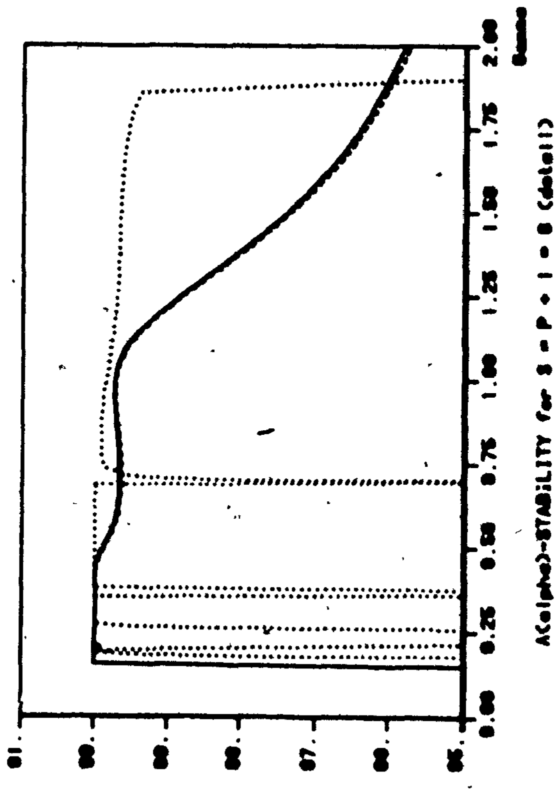


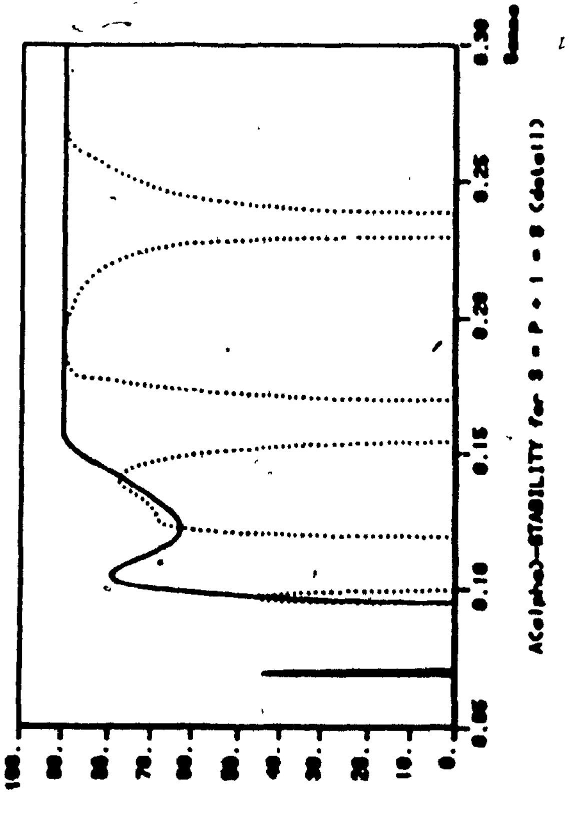
A(alpha)-STABILITY for S = P + 1 = 7 (detail)

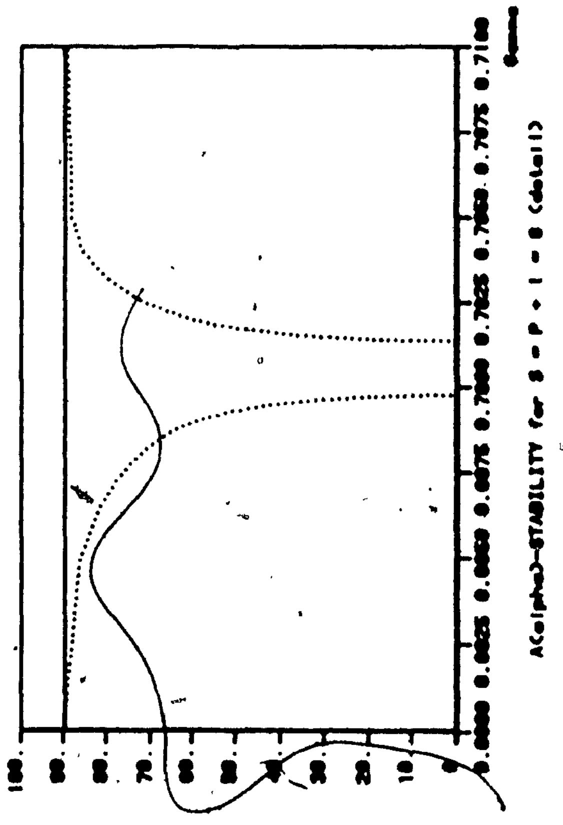


A(alpha)-STABILITY for S = P + 1 = 8


A(alpha)-STABILITY for S = P + I = 8 (detail)
Gamma
0.350 0.355 0.360 0.365 0.370 0.375 0.380 0.385 0.390
100. 90. 80. 70. 60. 50. 40. 30. 20. 10. 0.


A(alpha)-STABILITY for S = P + I = 8 (detail)
Gamma
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00


A(alpha)-STABILITY for S = P + I = 8 (detail)
Gamma
0.05 0.10 0.15 0.20 0.25 0.30
100. 90. 80. 70. 60. 50. 40. 30. 20. 10. 0.


A(alpha)-STABILITY for S = P + I = 8 (detail)
Gamma
0.6900 0.6925 0.6950 0.6975 0.7000 0.7025 0.7050 0.7075 0.7100
100. 90. 80. 70. 60. 50. 40. 30. 20. 10. 0.

APPENDIX 7

Program to Compute ω_0

For problems (6.1.1) satisfying (6.2.1), this program computes ω_0, an estimate of the *smallest* ω such that if $\mu+\omega L \leq 0$, the semi-implicit method will be contractive.

```
C OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO

      PROGRAM OMEGAS (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)

C OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
C
C CONSIDER THE NON-LINEAR SYSTEM  Y' = AY + G(T,Y), WHERE THE MATRIX A AND
C THE FUNCTION G SATISFY THE FOLLOWING:
C     <X,AX> <= M*(NORM(X))**2   AND  NORM(G(T,Y)-G(T,Z)) <= L*NORM(Y-Z).
C
C FOR SUCH PROBLEMS, ONE CAN SHOW THAT SEMI-IMPLICIT METHODS HAVE INTERESTING
C CONTRACTIVITY PROPERTIES, PROVIDED THEY ARE A-STABLE AND THAT THEIR STABI-
C LITY FUNCTION HAS MODULUS LESS THAN 1 AT INFINITY AND ON THE IMAGINARY AXIS.
C MORE PRECISELY, FOR A SET OF VALUES "OMEGA", WHEN THE SYSTEMS SATISFY THE
C CONDITION  M + OMEGA*L <= 0, WE CAN FIND A RANGE OF STEPSIZES (0,C/L), WHERE
C C IS A CONSTANT, FOR WHICH THE NUMERICAL SOLUTION IS CONTRACTIVE (I.E. WE
C HAVE   NORM(Y1-Z1) <= NORM(Y0-Z0) ).
C
C THIS PROGRAM COMPUTES  "OMEGA0", AN ESTIMATE OF THE LOWEST VALUE OF "OMEGA"
C FOR WHICH WE HAVE SUCH CONDITIONAL CONTRACTIVITY.
C
C REFERENCES: HAIRER, BADER AND LUBICH: ON THE STABILITY OF SEMI-IMPLICIT
C ----------  METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS, BIT (22), 1982,
C             PP. 211-232.
C             THIS PAPER WILL BE REFERRED TO BELOW AS HBL.
C
C THE PROGRAM WORKS FOR SEMI-IMPLICIT METHODS OF UP TO 4 STAGES. IT USES THE
C IMSLIBS SUBROUTINE "ZPOLR" TO COMPUTE ALL THE ROOTS OF POLYNOMIALS OF DEGREE
C 2 AND 3, WHEN NEEDED.
C
C
C     INPUT :  ON CONSECUTIVE LINES,
C     -----    --------------------
C                           - SOME IDENTIFICATION FOR THE METHOD (10 CHARS)
C                           - THE NUMBER OF STAGES
C                           - THE VALUE OF GAMMA
C                           - THE "B(I)" COEFFICIENTS ON SEPARATE LINES
C                           - ON EACH OF THE FOLLOWING LINES, A PAIR OF
C                             VALUES "ALPHA(I,J)" AND "BETA(I,J)" WITH
C                             I>J, IN LEXICOGRAPHICAL ORDER (FIRST: (2,1)
C                             THEN (3,1), THEN (3,2), THEN (4,1), THEN
C                             (4,2) AND FINALLY (4,3)).
C
C     OUTPUT : THE VALUE OF THE ESTIMATED "OMEGA0", AS WELL AS THE MAXIMUM
C     ------   MODULUS OF EACH BETA-FUNCTION (NOTED "BJ" IN THE PROGRAM) AND
C              EACH INTERNAL STABILITY FUNCTION (NOTED "RJ" IN THE PROGRAM)
C              ON THE IMAGINARY AXIS (SEE HBL, THEOREM 13).
```

```
      INTEGER NDEG,NPRR,I,IER,J,MAXS,NBCHAR,S
      PARAMETER (MAXS = 4, NBCHAR=10)

      REAL GAMMA,GAMMA2,OMEGA,TEMPMAX,T1,T2,T3,T4,T5,T6,T7,T8

      REAL ALPHA(MAXS),B(MAXS),BETA(MAXS),BJ(MAXS),POLY(MAXS),
     1     PRROOT(MAXS),RJ(MAXS)

      REAL ALPH(MAXS,MAXS),BET(MAXS,MAXS)

      COMPLEX Z(MAXS)

      CHARACTER*(NBCHAR)  IDENT
      CHARACTER*50 OZZ, TITLE

      DATA  OZZ /'00000000000000000000000000000000000000000000000000'/
      DATA TITLE/'  COMPUTATION OF CONTRACTIVITY PARAMETER OMEGAD.   '/

C ---------------------------------------------------------------
C  THIS FIRST PART READS IN THE COEFFICIENTS OF THE METHOD AND COMPUTES
C  SOME USEFUL PARTIAL SUMS ON THEM.
C ---------------------------------------------------------------

      READ (5,500) IDENT
500   FORMAT (A10)
      READ*, S
      READ*, GAMMA
      DO 10 I=1,S
10       READ*, B(I)
      DO 20 I=2,S
         BETA(I) = 0.0
         ALPHA(I) = 0.0
         DO 20 J=1,(I-1)
            READ*, ALPH(I,J),BET(I,J)
            BETA(I) = BETA(I) + BET(I,J)
            ALPHA(I) = ALPHA(I) + ALPH(I,J)
20    CONTINUE
```

```
C ------------------------------------------------------------------
C  NOW WE COMPUTE THE MAXIMUM MODULUS OF THE BETA-FUNCTIONS ON THE
C  IMAGINARY AXIS (SEE HBL, PAGE 221) AND THE SAME FOR THE INTERNAL
C  STABILITY FUNCTIONS. THE FORMER ARE COMPUTED IN DESCENDING ORDER
C  (I.E. FIRST BETA(S), THEN BETA(S-1) ETC...) WHILE THE FORMER ARE
C  COMPUTED IN ASCENDING ORDER (I.E. FIRST R(1), THEN R(2) ETC...)
C ------------------------------------------------------------------

      BJ(S) = ABS(B(S))
      RJ(1) = 1.0
C ------------------------------------------------------------------
      IF (S.EQ.1) GO TO 999
C ------------------------------------------------------------------

      T1 = (B(S)*BET(S,S-1) - GAMMA*B(S-1))**2
      T2 = (GAMMA*B(S-1))**2
      IF ((T1-2.0*T2).LE.0.0) THEN
         BJ(S-1) = ABS(B(S-1))
      ELSE
         BJ(S-1) = T1/(2.0*GAMMA*SQRT(T1-T2))
      ENDIF

      T1 = ABS(ALPHA(2) - GAMMA)/GAMMA
      IF (T1.GT.1.0) THEN
         RJ(2) = T1
      ELSE
         RJ(2) = 1.0
      ENDIF

C ------------------------------------------------------------------
      IF (S.EQ.2) GO TO 999
C ------------------------------------------------------------------

      GAMMA2 = GAMMA**2
      T1 = B(S-1)*BET(S-1,S-2) + B(S)*BET(S,S-2)
      T2 = B(S-2)
      T3 = GAMMA*(T1-GAMMA*T2) - B(S)*BET(S,S-1)*BET(S-1,S-2)
      T4 = (T1 - 2.0*GAMMA*T2)**2
      T5 = T4 + 2.0*T2*T3
      POLY(1) = -(GAMMA*T3)**2
      POLY(2) = 2.0*(T3**2 - T5*GAMMA2)
      POLY(3) = T5 - 3.0*(T2*GAMMA)**2

      NDEG = 2
      CALL ZPOLR(POLY,NDEG,Z,IER)
      CALL GETREAL(NDEG,Z,NPRR,PRROOT)
      TEMPMAX = ABS(B(S-2))
      DO 50 I=1,NPRR
         T1 = PRROOT(I)
         T5 = SQRT(((T2+T3*T1)**2 + T4*T1)/(1.0+T1*GAMMA2)**3)
         IF (T5.GT.TEMPMAX) TEMPMAX = T5
50    CONTINUE

      BJ(S-2) = TEMPMAX

      T1 = ALPHA(3) - 2.0*GAMMA
      T2 = ALPH(3,2)*BETA(2) + GAMMA*(GAMMA-ALPHA(3))
      TEMPMAX = ABS(T2)/GAMMA2
      IF (TEMPMAX.LT.1.0) TEMPMAX = 1.0
      T3 = T1**2 - 2.0*T2
      T4 = (T3 - 2.0*GAMMA2)/(T3*GAMMA2-2.0*T2**2)
      IF (T4.GT.0.0) THEN
         T5 = SQRT((1.0+T3*T4 +(T2*T4)**2)/(1.0+T4*GAMMA2)**2)
         IF (T5.GT.TEMPMAX) TEMPMAX = T5
      ENDIF
      RJ(3) = TEMPMAX

C ---------------------------------------------------------------
      IF (S.EQ.3) GO TO 999
C ---------------------------------------------------------------

      T1 = B(S-3)
      T2 = B(S-2)*BET(S-2,S-3)+B(S-1)*BET(S-1,S-3)+B(S)*BET(S,S-3)
      T3 = BET(S-1,S-2)*BET(S-2,S-3)
      T4 = B(S-1)*T3 + B(S)*(BET(S,S-1)*BET(S-1,S-3)+
     1                       BET(S,S-2)*BET(S-2,S-3))
      T5 = B(S)*BET(S,S-1)*T3
      T6 = -(T4 + GAMMA*(3.0*T1*GAMMA - 2.0*T2))
      T7 = T2 - 3.0*T1*GAMMA
      T8 = -(T5 + GAMMA*(-T4 + GAMMA*(T2 - T1*GAMMA)))
      T2 = T7**2 + 2.0*T1*T6
      T3 = 2.0*(T6**2 + 2.0*T7*T8)

      NDEG = 3
      POLY(1) = -(GAMMA*T8)**2
      POLY(2) = 3.0*T8**2 - GAMMA2*T3
      POLY(3) = T3 - 3.0*GAMMA2*T2
      POLY(4) = T2 - 4.0*(T1*GAMMA)**2
      CALL ZPOLR (POLY,NDEG,Z,IER)
      CALL GETREAL (NDEG,Z,NPRR,PRROOT)
      TEMPMAX = ABS(B(S-3))
      DO 100 I=1,NPRR
         T4 = PRROOT(I)
         T5 = SQRT(((T1+T6*T4)**2+T4*(T7+T8*T4)**2)/
     1             (1.0+GAMMA2*T4)**4)
         IF (T5.GT.TEMPMAX) TEMPMAX = T5
100   CONTINUE

      BJ(S-3) = TEMPMAX

      T1 = ALPH(4,2)*BETA(2) + ALPH(4,3)*BETA(3)
      T2 = ALPHA(4) - 3.0*GAMMA
      T3 = T1 + GAMMA*(3.0*GAMMA - 2.0*ALPHA(4))
      T4 = ALPH(4,3)*BET(3,2)*BETA(2)-GAMMA*(GAMMA*(GAMMA-ALPHA(4))+T1)
      T5 = T2**2 - 2.0*T3
      T6 = T3**2 - 2.0*T2*T4
      POLY(1) = 3.0*T4**2 - GAMMA2*T6
      POLY(2) = 2.0*(T6 - GAMMA2*T5)
      POLY(3) = T5 - 3.0*GAMMA2

      NDEG = 2
      CALL ZPOLR(POLY,NDEG,Z,IER)
      CALL GETREAL (NDEG,Z,NPRR,PRROOT)
      TEMPMAX = ABS(T4/GAMMA**3)
      IF (TEMPMAX.LT.1.0) TEMPMAX = 1.0
      DO 150 I=1,NPRR
         T1 = PRROOT(I)
         T5 = SQRT(((1.0-T1*T3)**2+T1*(T2-T4*T1)**2)/(1.0+GAMMA2*T1)**3)
         IF (T5.GT.TEMPMAX) TEMPMAX = T5
150   CONTINUE

      RJ(4) = TEMPMAX

C -----------------------------------------------------------------------
C  NOW THAT ALL MAXIMUM MODULUS OF THE BETA- AND INTERNAL STABILITY
C  FUNCTIONS ON THE IMAGINARY AXIS HAVE BEEN COMPUTED, WE ARE READY
C  TO COMPUTE THE VALUE OF THE METHOD-DEPENDANT PARAMETER "OMEGAO"
C -----------------------------------------------------------------------

999   OMEGA = 0.0
      DO 200 I=1,S
200      OMEGA = OMEGA + BJ(I)*RJ(I)
```

```
C ---------------------------------------------------------------------
C  NOW WE PREPARE THE OUTPUT FOR PRINTING
C ---------------------------------------------------------------------

      WRITE (6,1000) OZZ,TITLE,OZZ
1000  FORMAT('1'/1X,A50//1X,A50//1X,A50////)
      PRINT*, '         METHOD :   ', IDENT
      PRINT*, '         ********************'
      WRITE (6,2000)
2000  FORMAT (///15X,'BETA-FUNCTION',14X,'INTERNAL STABILITY FUNCTION'/
     1           14X,'-----------------------------------------',
     1                                  '---------------------')
      DO 300 I=1,S
300      WRITE(6,3000) I,BJ(I),RJ(I)
3000  FORMAT ('  (',I1,') :',5X,E20.14,13X,E20.14)

      WRITE (6,4000) OMEGA
4000  FORMAT (///'     THE VALUE OF OMEGA0 IS:     ', E20.14/
     1               '*********************************************')

      STOP
      END
```

```
C 0000000000000000000000000000000000000000000000000000000000000000000000
C
      SUBROUTINE GETREAL (N,Z,NPRR,PRROOT)
C
C 0000000000000000000000000000000000000000000000000000000000000000000000
C
C  THIS SUBROUTINE TAKES AS ITS INPUT A VECTOR "Z" OF "N" COMPLEX ROOTS
C  AND RETURNS A VECTOR "PRROOT" CONSISTING OF ONLY THE POSITIVE REAL
C  ROOTS AND THEIR NUMBER "NPRR".

      INTEGER N,NPRR,I
      REAL PRROOT(N),TOL
      COMPLEX Z(N)
      PARAMETER (TOL=1.0E-8)

      NPRR = 0
      DO 10 I=1,N
         IF((ABS(AIMAG(Z(I))).LT.TOL).AND.(REAL(Z(I)).GT.0.0)) THEN
            NPRR = NPRR + 1
            PRROOT(NPRR) = REAL(Z(I))
         ENDIF
10    CONTINUE
      RETURN
      END
```

APPENDIX 8

Program to Compute Maximum h for Contractivity

We consider problems (6.1.1), satisfying (6.2.1). The following program computes ω_0 as well as $\omega_{\inf}$, the smallest value of ω such that for $\mu+\omega L \leq 0$, the semi-implicit method is *unconditionnally* contractive.

For equally spaced values of $(-L/\mu)$ in the range $(\frac{1}{\omega_{\inf}}, \frac{1}{\omega_0})$, the program computes the maximum value of $(-h\mu)$ such that the method is contractive.

```
C 000000000000000000000000000000000000000000000000000000000000000
C
      PROGRAM KAPPAS (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
C 000000000000000000000000000000000000000000000000000000000000000
C
C  SEMI-IMPLICIT METHODS HAVE INTERESTING CONTRACTIVITY PROPERTIES
C  WHEN APPLIED TO NONLINEAR SYSTEMS CF DIFFERENTIAL EQUATIONS
C  Y' = AY + G(T,Y), WHERE THE MATRIX A AND THE FUNCTION G SATISFY
C  THE FOLLOWING CONDITIONS:
C              --  <X,AX> <= M*(NORM[X])**2
C              --  NORM[G(T,Y)-G(T,Z)] <= L*NORM[Y-Z]
C
C  FOR PROBLEMS SATISFYING M + OMEGA*L  <=  0  (OMEGAO>=1), WE CAN
C  ALWAYS FIND C SUCH THAT THE METHOD IS CONTRACTIVE FOR STEPSIZES IN
C  (0,C/L).
C
C  LET "OMEGAO" BE THE SMALLEST OMEGA FOR WHICH THE ABOVE HOLDS FOR FINITE
C  CONSTANT C; ALSO, LET "OMEGAINF" BE THE SMALLEST OMEGA FOR WHICH THE ABOVE
C  INEQUALITY HOLDS FOR INFINITE C.
C
C  THIS PROGRAM FIRST COMPUTES "OMEGAO" AND THEN "OMEGAINF". THEN, FOR
C  EQUALLY SPACED VALUES OF THE RATIO (-L/M) --THE INCREMENT USED IS SPECIFIED
C  IN THE PARAMETER STATEMENT BELOW (INC = )-- THE PROGRAM COMPUTES THE MAXIMUM
C  VALUE OF (-H*M) FOR WHICH THE METHOD IS CONTRACTIVE. VALUES OF (-L/M) MUST
C  OF COURSE BE IN THE INTERVAL (1/OMEGAINF, 1/OMEGAO).
C
C  TO COMPUTE THE MAXIMUM (-H*M), A FIRST BISECTION DETERMINES THE ORDER OF
C  MAGNITUDE I.E. THE APPROPRIATE POWER OF 10 IN THE RANGE (10**-8, 10**8);
C  THEN A SECOND BISECTION DETERMINES THE VALUE TO THREE SIGNIFICANT DIGITS.
C
C
C  REFERENCES:  -- HAIRER, BADER AND LUBICH: ON THE STABILITY OF SEMI-IMPLICIT
C  ----------      METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS, BIT (22), 1982,
C                  PP. 211-232.
C
C               -- LEGAULT: ON THE STABILITY AND CONTRACTIVITY PROPERTIES OF
C                  SEMI-IMPLICIT RUNGE-KUTTA METHODS, MASTER'S THESIS,
C                  CONCORDIA UNIVERSITY, MONTREAL, SECTION 6.5.
C
```

```
C  INPUT (ON CONSECUTIVE LINES):
C  --------------------------
C                    - SOME IIDENTIFICATION FOR THE METHOD (10 CHARS)
C                    - THE DEGREES OF THE NUMERATOR AND DENOMINATOR OF THE STABILITY
C                    - FUNCTION OF THE METHOD
C                    - THE VALUE OF GAMMA
C                    - THE "B(I)" COEFFICIENTS ON SEPARATE LINES
C                    - ON EACH OF THE FOLLOWING LINES, A PAIR OF VALUES
C                      "ALPHA(I,J)" AND "BETA(I,J)" WITH I>J, IN LEXICOGRAPHICAL
C                      ORDER (I.E. FIRST (2,1), THEN (3,1), THEN (3,2), THEN
C                      (4,1), (4,2) AND FINALLY (4,3)).
C
C  OUTPUT :
C  ------            - THE ESTIMATED "OMEGA0"
C                    - THE ESTIMATED "OMEGAINF"
C                    - A TABLE GIVING THE MAXIMUM (-H*M) AS A FUNCTION OF DIFFERENT
C                      VALUES OF THE RATIO (-L/M)
C
C

      INTEGER I,J,LEFT,MAXS,NBCHAR,P,RIGHT,S,TRY
      PARAMETER (MAXS = 4, NBCHAR = 10)

      REAL APPROX

      REAL GAMMA,GOOD,INC,KAPPA,KAPPAF,MAXRAT,MIDDLE,NOGOOD,OMEGA,
     1     OMEGINF,RATIO,X,Y
      PARAMETER (INC = 0.02)

      REAL ALPHA(MAXS),B(MAXS),BETA(MAXS),TEN(0:16)
      REAL ALPH(MAXS,MAXS),BET(MAXS,MAXS),COMBI(0:MAXS,0:MAXS)
      CHARACTER*(NBCHAR) IDENT
      CHARACTER*50 OZZ,TITLE

      COMMON /COM1/ COMBI
      COMMON /COM2/ ALPH,ALPHA,B,BET,BETA
      COMMON /COM3/ OMEGA

      DATA OZZ /'00000000000000000000000000000000000000000000000000'/
      DATA TITLE/'  MAXIMUM ALLOWABLE STEPSIZE FOR CONTRACTIVITY.   '/
```

```
C ---------------------------------------------------------------
C  THIS FIRST PART READS IN THE COEFFICIENTS OF THE METHOD AND COMPUTES
C  SOME USEFUL PARTIAL SUMS ON THEM.
C ---------------------------------------------------------------

      READ (5,500) IDENT
500   FORMAT (A10)
      READ*, P, S
      READ*, GAMMA
      DO 10 I=1,S
10       READ*, B(I)
      DO 20 I=2,S
         BETA(I) = 0.0
         ALPHA(I) = 0.0
         DO 20 J=1,(I-1)
            READ*, ALPH(I,J),BET(I,J)
            BETA(I) = BETA(I) + BET(I,J)
            ALPHA(I) = ALPHA(I) + ALPH(I,J)
20    CONTINUE

      COMBI(0,0) = 1.0
      DO 30 I=1,S
         COMBI(I,0) = 1.0
         COMBI(I,I) = 1.0
30    CONTINUE

      DO 40 I=2,S
         DO 40 J=1,I-1
            COMBI(I,J) = COMBI(I-1,J-1) + COMBI(I-1,J)
40    CONTINUE

      TEN(0) = 1.0E8
      DO 50 I=1,16
50       TEN(I) = TEN(I-1)/10.0

C ----------------------------------------------------------------------
C  NOW WE PREPARE THE OUTPUT AND COMPUTE OMEGA*
C ----------------------------------------------------------------------

      WRITE (6,1000) OZZ,TITLE,OZZ
1000  FORMAT('1'/1X,A50//1X,A50//1X,A50///)
      PRINT*, '    METHOD :   ',IDENT
      PRINT*, '    ********************'
      X = 0.0
      Y = 0.0
      KAPPA = KAPPAF(P,S,GAMMA,X,Y)
      WRITE (6,2000) OMEGA
2000  FORMAT (//4X,'COMPUTED "OMEGA*"  :',F14.10)
```

```
C -----------------------------------------------------------------
C  NOW WE COMPUTE "OMEGAINF"
C -----------------------------------------------------------------

      MAXRAT = 1.0/OMEGA
      NOGOOD = MAXRAT + 1.0E8
      X = - TEN(0)
      Y = 0
      KAPPA = KAPPAF (P,S,GAMMA,X,Y)
      IF (KAPPA.GT.1.0) THEN
         WRITE (6,3000)
3000     FORMAT (4X,'COMPUTED "OMEGINF":  INFINITY')
         RATIO = 0.0
         GO TO 200
      ENDIF
      GOOD = 0.0

100   MIDDLE = (GOOD+NOGOOD)/2.0
      Y = -MIDDLE*X
      KAPPA = KAPPAF (P,S,GAMMA,X,Y)
      IF (KAPPA.LE.1.0) THEN
         GOOD = MIDDLE
      ELSE
         NOGOOD = MIDDLE
      ENDIF
      IF ((NOGOOD - GOOD).GT.1.0E-6) GO TO 100

      OMEGINF = 1.0/GOOD
      WRITE (6,4000) OMEGINF
4000  FORMAT (4X,'COMPUTED "OMEGAINF": ',F8.5//)

      PRINT*,'      -L/MU       MAX(-H*MU)'
      PRINT*,'      -----       ----------'
      WRITE (6,4500)
4500  FORMAT (/)

      RATIO = INC*(IFIX(GOOD/INC) - 1)
```

```
C ----------------------------------------------------------------------
C  NOW, FOR VALUES OF (-L/MU) IN THE ACCEPTABLE RANGE (1/OMEGINF,
C   1/OMEGA*) WE COMPUTE THE MAXIMUM (H*MU) ALLOWED FOR CONTRACTIVITY
C ----------------------------------------------------------------------

200    RATIO = RATIO + INC
       X = -TEN(0)
       Y = -RATIO*X
       KAPPA = KAPPAF(P,S,GAMMA,X,Y)

       IF (KAPPA.LE.1.0) THEN
          WRITE (6,5000) RATIO
5000      FORMAT (6X,F3.2,8X,'>.100E+09')
          GO TO 230
       ENDIF

       X = - TEN(16)
       Y = -RATIO*X
       KAPPA = KAPPAF (P,S,GAMMA,X,Y)
       IF (KAPPA.GT.1.0) THEN
          WRITE (6,6000) RATIO
6000      FORMAT (6X,F3.2,8X,'<.100E-07')
          GO TO 230
       ENDIF

       LEFT = 0
       RIGHT = 16
210    TRY = (LEFT+RIGHT)/2
       X = - TEN(TRY)
       Y = -RATIO*X
       KAPPA = KAPPAF(P,S,GAMMA,X,Y)

       IF (KAPPA.LE.1.0) THEN
          RIGHT = TRY
       ELSE
          LEFT = TRY
       ENDIF
       IF ((RIGHT-LEFT).GT.1) GO TO 210

       GOOD = 1.0
       NOGOOD = 10.0
```

```
  220   MIDDLE = (GOOD+NOGOOD)/2.0
        X = - MIDDLE*TEN(RIGHT)
        Y = -RATIO*X
        KAPPA = KAPPAF(P,S,GAMMA,X,Y)
        IF (KAPPA.LE.1.0) THEN
           GOOD = MIDDLE
        ELSE
           NOGOOD = MIDDLE
        ENDIF
        IF ((NOGOOD-GOOD).GT.0.01) GO TO 220

        APPROX = IFIX(GOOD*100.0)*TEN(RIGHT)/100.0
        WRITE (6,7000) RATIO,APPROX
 7000   FORMAT(6X,F3.2,9X,E8.3)

  230   IF(RATIO.LT.MAXRAT) GO TO 200

        STOP
        END
```

```
C 0000000000000000000000000000000000000000000000000000000000000000000
C
      REAL FUNCTION KAPPAF (P,S,GAMMA,X,Y)
C
C 0000000000000000000000000000000000000000000000000000000000000000000
C
C     THIS FUNCTION EVALUATES THE AMPLIFICATION FACTOR FOR A GIVEN
C     SEMI-IMPLICIT METHOD WHOSE STABILITY FUNCTION HAS A NUMERATOR
C     POLYNOMIAL OF DEGREE "P" AND A DENOMINATOR POLYNOMIAL OF DEGREE
C     "S" (HENCE A METHOD WITH S STAGES). THE PARAMETER "X" IS THE
C     PRODUCT OF THE STEPSIZE AND THE LOGARITHMIC NORM OF THE PROBLEM
C     MATRIX A; AND "Y" IS THE PRODUCT OF THE STEPSIZE AND THE LIPSCHITZ
C     CONSTANT FOR THE PROBLEM VECTOR FUNCTION G.. MANY CALLS
C     ARE MADE TO THE SUBROUTINE "SUP", DESCRIBED BELOW.


      INTEGER DEGD,DEGN,I,MAXS,P,S
      PARAMETER (MAXS = 4)

      REAL GAMMA,OMEGA,SUP,SUPR,T1,T2,T3,T4,T5,TOL,X,Y
      PARAMETER (TOL = 1.0E-10)
C "TOL" IS A TOLERANCE TO CHECK FOR NULL LEADING COEFFICIENT IN POLYNOMIALS

      REAL ALPHA(MAXS),B(MAXS),BETA(MAXS),BF(MAXS),
     1                 NUM(0:MAXS),OMEG(3:MAXS),PHI(2:MAXS),W(2:MAXS)

      REAL AF(2:MAXS,1:MAXS),ALPH(MAXS,MAXS),
     1                 BET(MAXS,MAXS),COMBI(0:MAXS,0:MAXS)

      COMMON /COM1/ COMBI
      COMMON /COM2/ ALPH,ALPHA,B,BET,BETA
      COMMON /COM3/ OMEGA


      DEGN = 0
      DEGD = 1
      NUM(0) = B(S)
      BF(S) = SUP (DEGN,NUM,DEGD,GAMMA,X)

C ---------------------------------------------------------------------
      IF (S.EQ.1) GO TO 100
C ---------------------------------------------------------------------

      DO 10 I=2,S
         NUM(0) = ALPH(I,I-1)
         AF(I,I-1) = SUP (DEGN,NUM,DEGD,GAMMA,X)
10    CONTINUE
```

```
      NUM(0) = 1.0
      NUM(1) = ALPHA(2) - GAMMA
      IF (ABS(NUM(1)).GT.TOL) DEGN = 1
      PHI(2) = SUP (DEGN,NUM,DEGD,GAMMA,X)
      W(2) = PHI(2) + Y*AF(2,1)

      DEGN = 1
      DEGD = 2
      NUM(0) = B(S-1)
      NUM(1) = B(S)*BET(S,S-1) - GAMMA*B(S-1)
      BF(S-1) = SUP (DEGN,NUM,DEGD,GAMMA,X)
C ------------------------------------------------------------------
      IF (S.EQ.2) GO TO 100
C ------------------------------------------------------------------

      DO 20 I=3,S
         NUM(0) = ALPH(I,I-2)
         NUM(1) = ALPH(I,I-1)*BET(I-1,I-2) - GAMMA*ALPH(I,I-2)
         AF(I,I-2) = SUP (DEGN,NUM,DEGD,GAMMA,X)
20    CONTINUE

      OMEG(3) = AF(3,1) + AF(3,2)*W(2)

      NUM(0) = 1.0
      NUM(1) = ALPHA(3) - 2.0*GAMMA
      NUM(2) = ALPH(3,2)*BETA(2) + GAMMA*(GAMMA - ALPHA(3))
      IF (ABS(NUM(2)).GT.TOL) DEGN = 2
      PHI(3) = SUP (DEGN,NUM,DEGD,GAMMA,X)
      W(3) = PHI(3) + Y*OMEG(3)

      T1 = B(S-1)*BET(S-1,S-2) + B(S)*BET(S,S-2)
      T2 = B(S-2)
      DEGN = 2
      DEGD = 3
      NUM(0) = T2
      NUM(1) = T1 - 2.0*GAMMA*T2
      NUM(2) = GAMMA*(GAMMA*T2 - T1) + B(S)*BET(S,S-1)*BET(S-1,S-2)
      BF(S-2) = SUP (DEGN,NUM,DEGD,GAMMA,X)
```

```
C ---------------------------------------------------------------------
      IF (S.EQ.3) GO TO 100
C ---------------------------------------------------------------------

      DO 30 I=4,S
         T1 = ALPH(I,I-2)*BET(I-2,I-3) + ALPH(I,I-1)*BET(I-1,I-3)
         T2 = ALPH(I,I-1)*BET(I-1,I-2)*BET(I-2,I-3)
         T3 = ALPH(I,I-3)
         NUM(0) = T3
         NUM(1) = T1 - 2.0*GAMMA*T3
         NUM(2) = T2 - GAMMA*T1 + T3*(GAMMA**2)
         AF(I,I-3) = SUP (DEGN,NUM,DEGD,GAMMA,X)
30    CONTINUE

      OMEG(4) = AF(4,1) + AF(4,2)*W(2) + AF(4,3)*W(3)

      T1 = ALPH(4,2)*BETA(2) + ALPH(4,3)*BETA(3)
      NUM(0) = 1.0
      NUM(1) = ALPHA(4) - 3.0*GAMMA
      NUM(2) = T1 + GAMMA*(3.0*GAMMA - 2.0*ALPHA(4))
      NUM(3) = ALPH(4,3)*BET(3,2)*BETA(2)
     1       - GAMMA*(GAMMA*(GAMMA - ALPHA(4)) + T1)
      IF (ABS(NUM(3)).GT.TOL) DEGN = 3
      PHI(4) = SUP (DEGN,NUM,DEGD,GAMMA,X)

      W(4) = PHI(4) + Y*OMEG(4)

      T1 = B(S-3)
      T2 = B(S-2)*BET(S-2,S-3)+B(S-1)*BET(S-1,S-3)+B(S)*BET(S,S-3)
      T3 = BET(S-1,S-2)*BET(S-2,S-3)
      T4 = B(S-1)*T3 + B(S)*(BET(S,S-1)*BET(S-1,S-3)+
     1                       BET(S,S-2)*BET(S-2,S-3))
      T5 = B(S)*BET(S,S-1)*T3

      DEGN = 3
      DEGD = 4
      NUM(0) =-T1
      NUM(1) = T2 - 3.0*T1*GAMMA
      NUM(2) = T4 + GAMMA*(3.0*GAMMA*T1 - 2.0*T2)
      NUM(3) = T5 - GAMMA*(T4 + GAMMA*(GAMMA*T1 - T2))
      BF(S-3) = SUP (DEGN,NUM,DEGD,GAMMA,X)
```

```
C ----------------------------------------------------------------------
C WE ARE NOW READY TO COMPUTE THE VALUE OF "OMEGA" AND THE SUPREMUM "SUPR"
C OF THE STABILITY FUNCTION, R(Z), FOR THE SPECIFIED ABCISSA "X".
C ----------------------------------------------------------------------

100   OMEGA = BF(1)
      DO 110 I=2,S
110      OMEGA = OMEGA + BF(I)*W(I)

      DEGD = S
      DEGN = P
      NUM(0) = 1.0
      IF (P.GE.1)  NUM(1) = 1.0 - S*GAMMA
      IF (P.GE.2)  NUM(2) = 0.5 + GAMMA*(COMBI(S,2)*GAMMA - S)
      IF (P.GE.3)  NUM(3) = 1.0/6.0 - GAMMA*(S/2.0+GAMMA*(GAMMA*
     1                                   COMBI(S,3) - COMBI(S,2)))
      IF (P.GE.4)  NUM(4) = 1.0/24.0 + GAMMA*(-S/6.0 + GAMMA*
     1         (0.5*COMBI(S,2)+GAMMA*(COMBI(S,4)*GAMMA-COMBI(S,3))))

      SUPR = SUP (DEGN,NUM,DEGD,GAMMA,X)
C -------------------------------------------------------------------
C FINALLY, WE EVALUATE THE AMPLIFICATION FACTOR
C -------------------------------------------------------------------

      KAPPAF = SUPR + Y*OMEGA

      RETURN
      END
```

```
C 000000000000000000000000000000000000000000000000000000000000000000
C
      REAL FUNCTION SUP (DEGN,NUM,DEGD,GAMMA,X)
C
C 000000000000000000000000000000000000000000000000000000000000000000
C
C     THIS FUNCTION RETURNS THE MAXIMUM MODULUS OF A RATIONAL FUNCTION
C     IN A COMPLEX VARIABLE, ON A SPECIFIED VERTICAL LINE IN THE COMPLEX
C     PLANE (SPECIFIED BY GIVEN ABCISSA "X")
C
C     THE FUNCTION IS SPECIFIED AS 2 POLYNOMIALS (NUMERATOR AND DENOMINATOR)
C     IN THE COMPLEX VARIABLE . NOTE THAT THESE POLYNOMIALS HAVE REAL COEFFI-
C     CIENTS. FURTHERMORE, THE DENOMINATOR IS ALWAYS AN INTEGER POWER OF
C     (1 - GAMMA*Z); SO WE NEED ONLY SPECIFY IT WITH THE VALUE OF "GAMMA"
C     AND ITS DEGREE, "DEGD".
C
C     THE NUMERATOR POLYNOMIAL HAS SPECIFIED DEGREE "DEGN" AND ITS COEFFI-
C     CIENTS ARE STORED IN ARRAY "NUM" IN INCREASING ORDER OF THEIR CORRES-
C     PONDING POWER OF Z (I.E. THE POLYNOMIAL IS:    NUM(0) + NUM(1)*Z +
C     NUM(2)*Z**2 + ... + NUM(DEGN)*Z**DEGN)
C
C     OF COURSE, WE MUST HAVE DEGN <= DEGD, OTHERWISE THE ANSWER IS INFINITE.
C             ------------------------
C
C     CALLS ARE MADE TO OTHER SUBROUTINES DESCRIBED BELOW, NAMELY TO
C     "NORM2", "PMULT" AND "PEVAL". ALSO "ZPOLR" IS USED FROM IMSL
C     (IMSLIB5), TO COMPUTE THE ROOTS OF POLYNOMIALS.


      INTEGER DEG1,DEG2,DEGD,DEGN,I,IER,MAXS,MAXSP1,NPRR
      PARAMETER (MAXS=4,MAXSP1=5)

      REAL G2,GAMMA,MAXNOW,PEVAL,TEMP,X
      REAL DEN(0:1),NUM(0:MAXS),NUMY2(0:MAXS)
      REAL P1(0:MAXS),P2(0:MAXS),P3(MAXSP1),PRROOT(MAXS)
      REAL COMBI(0:MAXS,0:MAXS)

      COMPLEX Z(MAXS)

      COMMON /COM1/ COMBI
      IF (DEGN.EQ.0) THEN
         SUP = ABS(NUM(0))/(1.0-GAMMA*X)**DEGD
         GO TO 50
      ENDIF
      CALL NORM2 (DEGN,NUM,X,NUMY2)
      G2 = GAMMA**2
      DEN(0) = (1.0-GAMMA*X)**2
      DEN(1) = G2

      DO 10 I=1,DEGN
10       P2(I-1) = I*NUMY2(I)

      DEG1 = 1
      DEG2 = DEGN - 1
      CALL PMULT (DEG1,DEN,DEG2,P2,DEGN,P1)

      TEMP = - DEGD*G2
      DO 20 I=0,DEGN
20       P1(I) = P1(I) + TEMP*NUMY2(I)
      MAXNOW = SQRT (NUMY2(0))/(1.0-GAMMA*X)**DEGD
      IF (DEGN.EQ.DEGD) THEN
         TEMP = ABS(NUM(DEGN))/(GAMMA**DEGD)
         IF (TEMP.GT.MAXNOW) MAXNOW = TEMP
         DEG1 = DEGN
      ELSE
         DEG1 = DEGN + 1
      ENDIF

      IF (DEG1.GT.1) THEN
         DO 30 I=1,DEG1
30          P3(I) = P1(DEG1-I)
         DEG2 = DEG1 - 1
         CALL ZPOLR (P3,DEG2,Z,IER)
         CALL GETREAL (DEG2,Z,NPRR,PRROOT)

         DO 40 I=1,NPRR
            TEMP = SQRT(PEVAL(DEGN,NUMY2,PRROOT(I))/
     1              (DEN(0) + G2*PRROOT(I))**DEGD)
            IF (TEMP.GT.MAXNOW) MAXNOW = TEMP
40       CONTINUE
      ENDIF

      SUP = MAXNOW

50    RETURN
      END
```

```
C 000000000000000000000000000000000000000000000000000000000000000000000000
C
      SUBROUTINE NORM2 (DEG,V,X,PY2)
C
C 000000000000000000000000000000000000000000000000000000000000000000000000
C
C     THIS SUBROUTINE TAKES AS ITS INPUT A POLYNOMIAL IN A COMPLEX VARIABLE
C     OF DEGREE "DEG", WHOSE COEFFICIENTS ARE STORED IN ARRAY "V" IN INCREA-
C     ORDER OF THEIR CORRESPONDING POWER FROM 0 TO DEG.
C
C     THE 'NORM' (I.E. THE SQUARE OF THE MODULUS) OF THIS POLYNOMIAL ON ANY
C     VERTICAL LINE IN THE COMPLEX (X,Y)-PLANE CAN BE EXPRESSED AS A POLYNO-
C     MIAL OF DEGREE "DEG" IN THE VARIABLE Y**2, WHOSE COEFFICIENTS ARE RE-
C     TURNED IN ARRAY "PY2", IN THE SAME INCREASING ORDER OF THEIR CORRESPON-
C     DING POWER OF Y**2.
C
C     THE COEFFICIENTS IN "PY2" ARE COMPUTED FOR THE SPECIFIED ABCISSA VALUE
C     "X", IN THE COMPLEX PLANE.


      INTEGER DEG,DEGP,DEGX,J,J2,K,MAXS
      PARAMETER (MAXS = 4)

      REAL PEVAL,X
      REAL IP(0:MAXS),IP2(0:MAXS),PXR(0:MAXS),PXI(0:MAXS),
     1                 PY2(0:MAXS),RP(0:MAXS),RP2(0:MAXS),V(0:MAXS)
      REAL COMBI(0:MAXS,0:MAXS)

      COMMON /COM1/ COMBI


      DO 10 J=0,(DEG-1)/2
         J2 = 2*J
         PXR(0) = V(J2)
         DO 20 K=J2+1,DEG
            PXR(K-J2) = V(K)*COMBI(K,J2)
            PXI(K-J2-1) = V(K)*COMBI(K,J2+1)
20       CONTINUE
         DEGX = DEG - J2
         RP(J) = PEVAL(DEGX,PXR,X)
         DEGX = DEGX - 1
         IP(J) = PEVAL (DEGX,PXI,X)
         IF (MOD(J,2).EQ.1) THEN
            RP(J) = - RP(J)
            IP(J) = - IP(J)
         ENDIF
10    CONTINUE
```

```
      DEGX = DEG/2
      IF (MOD(DEG,2).EQ.0) THEN
         RP(DEGX) = V(DEG)
         IF (MOD(DEGX,2).EQ.1)   RP(DEGX) = - RP(DEGX)
      ENDIF

      DEGP = 2*DEGX
      CALL PMULT (DEGX,RP,DEGX,RP,DEGP,RP2)
      DEGX = (DEG-1)/2
      DEGP = 2*DEGX
      CALL PMULT (DEGX,IP,DEGX,IP,DEGP,IP2)

      PY2(0) = RP2(0)
      IF (MOD(DEG,2).EQ.0) THEN
         PY2(DEG) = RP2(DEG)
      ELSE
         PY2(DEG) = IP2(DEG-1)
      ENDIF

      DO 30 I=1,DEG-1
30       PY2(I) = RP2(I) + IP2(I-1)

      RETURN
      END
```

```
C 00000000000000000000000000000000000000000000000000000000000000000000
C
      REAL FUNCTION PEVAL (DEG,POL,X)
C
C 00000000000000000000000000000000000000000000000000000000000000000000
C
C     THIS FUNCTION EVALUATES A POLYNOMIAL OF DEGREE "DEG" AT THE
C     SPECIFIED VARIABLE VALUE "X". THE REAL COEFFICIENTS OF THE
C     POLYNOMIAL ARE STORED IN ARRAY "POL", IN INCREASING ORDER OF THEIR
C     CORRESPONDING POWER FROM 0 TO DEG. THE METHOD USED IS HORNER'S RULE.

      INTEGER DEG,I
      REAL POL(0:DEG),TEMP,X

      TEMP = 0.0
      DO 10 I=DEG,0,-1
10       TEMP = TEMP*X + POL(I)

      PEVAL = TEMP

      RETURN
      END
```

```
C 000000000000000000000000000000000000000000000000000000000000000000000
C
      SUBROUTINE PMULT (DEG1,P1,DEG2,P2,DEGP,PROD)
C
C 000000000000000000000000000000000000000000000000000000000000000000000
C
C     THIS FUNCTION MULTIPLIES 2 POLYNOMIALS "P1" AND "P2" OF DEGREE
C     "DEG1" AND "DEG2" RESPECTIVELY AND RETURNS THE PRODUCT IN "PROD".
C     "DEGP" SHOULD BE SET TO (DEG1+DEG2) IN THE CALLING PROGRAM. FOR
C     ALL THE POLYNOMIALS, THE COEFFICIENTS ARE GIVEN IN INCREASING ORDER
C     OF THEIR CORRESPONDING POWERS, FROM 0 TO THEIR RESPECTIVE DEGREE.

      INTEGER DEG1,DEG2,DEGP,FIRST,I,K,LAST
      REAL P1(0:DEG1),P2(0:DEG2),PROD(0:DEGP)

      DO 10 K=0,DEGP

         IF (K.GT.DEG2) THEN
            FIRST = K - DEG2
         ELSE
            FIRST = 0
         ENDIF

         IF (K.LT.DEG1) THEN
            LAST = K
         ELSE
            LAST = DEG1
         ENDIF

         PROD(K) = 0.0
         DO 20 I=FIRST,LAST
20          PROD(K) = PROD(K) + P1(I)*P2(K-I)

10    CONTINUE

      RETURN
      END
```

```
C 000000000000000000000000000000000000000000000000000000000000000000
C
      SUBROUTINE GETREAL (N,Z,NPRR,PRROOT)
C
C 000000000000000000000000000000000000000000000000000000000000000000
C  THIS SUBROUTINE TAKES AS ITS INPUT A VECTOR "Z" OF "N" COMPLEX ROOTS
C  AND RETURNS A VECTOR "PRROOT" CONSISTING OF ONLY THE POSITIVE REAL
C  ROOTS AND THEIR NUMBER "NPRR". A ROOT IS CONSIDERED REAL IF ITS IMAGI-
C  NARY PART IS LESS (IN ABSOLUTE VALUE) THAN "TOL", SET IN THE
C  PARAMETER STATEMENT BELOW.

      INTEGER N,NPRR,I
      REAL PRROOT(N), TOL
      COMPLEX Z(N)
      PARAMETER (TOL=1.0E-6)

      NPRR = 0
      DO 10 I=1,N
         IF((ABS(AIMAG(Z(I))).LT.TOL).AND.(REAL(Z(I)).GT.0.0)) THEN
            NPRR = NPRR + 1
            PRROOT(NPRR) = REAL(Z(I))
         ENDIF
10    CONTINUE
      RETURN
      END
```