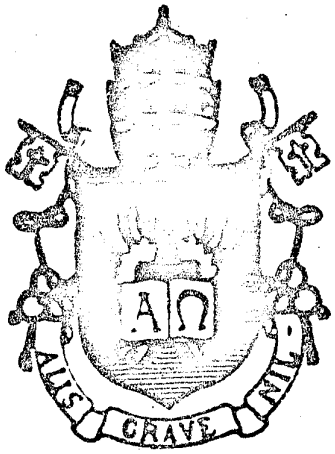

PUC

Series:Monografias em Ciência da Computação,
No. 19/91

ON THE NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS WITH CYCLIC METHODS

Mauricio Kischinhevsky

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP-22453
RIO DE JANEIRO - BRASIL

PUC RIO - DEPARTAMENTO DE INFORMÁTICA

Series: Monografias em Ciência da Computação, No. 19/91
Editor: Carlos J. P. Lucena October , 1991

ON THE NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS WITH CYCLIC METHODS *

Mauricio Kischinhevsky

* This work has been sponsored by Secretaria de Ciência e Tecnologia of Presidência da República Federativa do Brasil.

In charge of publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC Rio - Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453 - Rio de Janeiro, RJ
Brasil

Tel.:(021)529-9386 Telex:31078 Fax:(021)511-5645
E-mail:rosane@inf.puc-rio.br

Resumo:

Este relatório descreve uma implementação eficiente de um resolutor de Equações Diferenciais Ordinárias para métodos cíclicos do tipo Adams. É construído com sucessivas aplicações de estratégias para métodos implícitos, linha a linha do sistema de equações de diferenças correspondente. Alguns preditores do tipo Adams estão disponíveis, bem como o ajuste automático de passo. Discutem-se estratégias para estimativas de erro local e o software gerado é testado para métodos de ordem baixa, com custo computacional reduzido.

Palavras-chave:

Resolutor para equações diferenciais ordináriaas baseado em métodos cíclicos, ajuste de passo, estimativas de erro local.

Abstract:

This is a report on the development of an efficient implementation of a numerical solver for cyclic methods of Adams type. It is built by means of successive applications of strategies for implicit methods, row by row of a systems of difference equations. Several Adams predictors are available and adjustable step size is provided. Strategies for error estimation at each step are also discussed, leading to a computationally inexpensive package, which is tested for some low order methods.

Keywords:

Solver for cyclic methods, adjustable step size, local error estimation.

1 Introduction

The main problem refers to solving an Initial Value Problem (IVP) given by

$$y' = f(x, \mathbf{y}) \ , \ x \in [a, b] \tag{1}$$

with $y(x = a) = y(x_0) = y_0$ making use of an efficient numerical algorithm.

Within classical references for Numerical Solution of Ordinary Differential Equations (ODE) [1][2] one finds discussions that highlights Adams-Moulton implicit methods as those which provide less function evaluations, together with good precision requirements. Such class of multivalue methods offers much less function evaluations than one-step methods such as Runge-Kutta [3] for orders greater than one, and has coefficients smaller than Adams-Bashforth's explicit formulae [1][3], leading to reduced roundoff errors.

Cyclic Methods (CM) are constructed through (p-cyclic requires p equations) k-step linear systems, formed by k-valued linear difference equations, each of them referring to multiple previous values (k in the first equation), whose solution generates an estimate for the solution of the IVP under consideration at some grid point (see,e.g. [4]). These methods may provide fairly good properties of adequacy to power series to high orders, being of interest both theoretically[4] and practically since few iterations can provide low computational cost with good convergence (e.g. Donelson-Hansen's method, 3-equation,3-value, with order of convergence 6).

The solver to be presented finds one new value per row, row by row, leading to p estimates that will be input for the next iteration of the cyclic method. We shall restrict ourselves to the study of $k = p$ methods.

As an example for 3-equation, 3 valued method we have

$$\begin{aligned}
\alpha_3^1 \cdot y_{3j} + \cdots + \alpha_0^1 \cdot y_{3j-3} &= h \cdot \{\beta_3^1 \cdot f_{3j} + \cdots + \beta_0^1 \cdot f_{3j-3}\} \\
\alpha_4^2 \cdot y_{3j+1} + \cdots + \alpha_0^2 \cdot y_{3j-3} &= h \cdot \{\beta_4^2 \cdot f_{3j+1} + \cdots + \beta_0^2 \cdot f_{3j-3}\} \\
\alpha_5^3 \cdot y_{3j+2} + \cdots + \alpha_0^3 \cdot y_{3j-3} &= h \cdot \{\beta_5^3 \cdot f_{3j+2} + \cdots + \beta_0^3 \cdot f_{3j-3}\}
\end{aligned} \tag{2}$$

where f_j means $f(x_j, \mathbf{y}_j)$, $\mathbf{y}_j$ is the numerical approximation to $\mathbf{y}(x_i)$, and x_i denote grid points of the discretized domain. Referring to eqs(1), it is to be noticed that the complete step of the cyclic method there expressed will first find an estimate of y_{3j} making use of the first equation. After this an estimate for y_{3j+1} will be searched that uses the recently found y_{3j}. Calculating y_{3j+2} in an analogous way one has the beginning of a process that improves those estimates for y_{3j}, y_{3j+1} and y_{3j+2} through an implicit global procedure built with individual implicit procedures. It is widely accepted that some explicit procedure has to be used when solving for each of the equations, in order to have a preliminar value for f_{3j}, f_{3j+1} and f_{3j+2}, respectively. At such moment Adams-Bashforth formulae will be used. In subsequent iterations of the cyclic method such values will be updated with the help of the recently found values for y_{3j}, y_{3j+1} and y_{3j+2} substituted within f_{3j} , f_{3j+1} and f_{3j+2}, respectively. This is the general working structure of a PECE method, that is, Prediction

{ Evaluation Correction }* Evaluation or P { EC }* . Such structure comes up both in the global method and inside each equation.

In the continuation of the process, the equations will be advanced to other grid points, with superposition of a number of values equal to the number of equations the Cyclic Method shows, towards the grid point where the solution of the IVP is searched, that is, the final value of the independent variable.

A Cyclic Method's iteration will be considered adequately performed when an absolute (or relative) error per step is satisfied in the generated values. But this is enough, since the step size adjustment module that makes the solver an efficient tool will require such error estimate to check for the possibility of increasing the step size, or eventually halving it. Some numerical tests of the Solver are performed, using a class of cyclic methods with $k = p$, in order to describe performances both of the Solver (DECYC) itself and of the criteria proposed for local error estimation.

2 Description of the Solver

2.1 Initialization

To start with the first ordinary step some preliminary calculations must be performed. Once we are solving an Initial Value Problem, where only x_0 and $\mathbf{y}_0$ are available at the first moment, Runge-Kutta formulae of order q are used to generate the $(p-2)$ values of $\mathbf{y}$ ($\mathbf{y}_0$ given) and $(p-1)$ values of $f(x,\mathbf{y})$, necessary to solve the first implicit equation, thus starting the whole sweep through the p equations.

2.2 Steady State Step

The standard step of a Cyclic Method is a sequence of implicit difference equation integrations. In each of these equations we adopted PECE methods of at least k previous values (that is, $k+1$ values are present in first row, including the one to be found). Prediction tasks were performed with Adams-Bashforth formulae of order equal to the order of the correctors, prescribed by the user, while Correction used the implicit formulae generated through the α's and β's specified for that row.

Precision is specified by the user within the parameters passed to the module DECYC, as a tolerance for the residual of the Power Series, of order $q+1$ for an order q method.

2.3 Order Control

One of the facilities of current versions of Numerical Solvers for IVP Ordinary Differential Equations is to alter the order of the step, thus offering increased orders when convergence is not adequately achieved and avoiding computing efforts (by lowering the order) when precision does not decrease substantially.

In the implementation considered here this was not possible, since the coefficients and the order (fixed) are provided by the user in a list of parameters only once, thus remaining constant throughout the process of solving the ODE.

Consequently the only degree of freedom left was the Step Size.

2.4 Adjustable Step Size

The automatic control of step size is performed through an indirect estimate of the coefficient for the error term (in the Power Series Expansion) of the CM being performed.

Estimating local errors is the means the software has, in order to adjust the step size for minimizing global errors without expending too much computer time (and avoiding round-off errors related to it). The usual way to obtain such estimate is [1][2] to calculate periodically(e.g. at each 10 steps[1]) the residual of the Taylor expansion that relates the grid points at the difference equation. It can be shown [1] that the change in the last component of a vector a which contains finite differences of y, that is,

$$\mathbf{a} = \left[y, h \cdot y', \ldots, \frac{h^q \cdot y^{(q)}}{q!} \right]$$

is an estimate, at each step, of

$$\frac{h^{q+1} \cdot y^{(q+1)}}{q!}$$

So the strategy consists of forming the quantities to verify

$$C_{q+1} \cdot q! \cdot \nabla a_q \leq \varepsilon$$

where

- q - order of the method
- C_{q+1} - coefficient of Power Series Residual, of order $q+1$
- ε - precision, specified by the user, for maximum local error
- a_q - stores the scaled derivatives of y
- $\nabla \mathbf{a_q}$ - backward finite difference

Unfortunately, however, the coefficient of the first non-exact term within the order consistency analysis of the cyclic method under consideration, that is C_{q+1}, is not explicitly available. This led us to look for alternative ways of controlling step sizes. Next we discuss some aspects concerning the various trials performed.

2.4.1 Higher Order Runge-Kutta methods

In order to estimate the truncation error appearing when a value is obtained for a certain y_j, such strategy compares it with the result that would be obtained for the same x_j, starting from the last accepted value, that is, y_{j-p} for a p-stage (or p-row) method. Here a $(q+1)^{th}$ order Runge-Kutta method would be performed to provide a more accurate value in which we have stronger confidence.

Despite the fact that such suggestion is frequently brought up for its pedagogical interest (to ilustrate equivalences between Multivalue and RK methods), it did not become a good choice, and conceptually it would mean a disaster since a common sense solution would then be to discard such cyclic method, using the RK one. Other important aspect to consider was that the overhead associated would be relevant, since p RK steps (with at least $q+1$ evaluations of f in each) would be needed.

2.4.2 Internal Control by Doubling

Once the possibility of halving and doubling is devised as a means of controlling step sizes(see,e.g.,[5]), a very simple and efficient strategy is to control the difference between the values y_j would have if h and $2h$ (or $\frac{h}{2}$, alternatively) are used to obtain it.

One should notice that this strategy is quite empirical and calculates, at each step size's validation,

$$ERRD = \|y_j^h - y_j^{2h}\|$$

and/or

$$ERRH = \|y_j^h - y_j^{\frac{h}{2}}\|$$

where tests for modification of step size are performed making use of $ERRD$, $ERRH$ or both.

The strategies proposed were:

- (a) Doubling ($ERRH$ is not formed); if $ERRD$ is identified as in $I_1 = [0, \eta \cdot \varepsilon)$, $I_2 = [\eta \cdot \varepsilon, \theta \cdot \varepsilon)$ or $I_3 = [\theta \cdot \varepsilon, \infty)$ it causes the program to double, preserve or halve, respectively, the step size.
- (b) Halving ($ERRD$ is not formed); analogously it selects the correct interval and chooses the next step size.
- (c) Mixed (both $ERRD$ and $ERRH$ are formed); in case $ERRD$ is lower than $\gamma \cdot \varepsilon$, step size is doubled while when $ERRH$ is greater than $\delta \cdot \varepsilon$ causes step size to be halved. Otherwise step size is preserved.

Since the values that are required are stored to be at the program's disposal, no extra calculation is needed to start the Cyclic Method step which will generate y_j with h doubled, y_j^{2h}. For the Cyclic Step to be performed with h halved(leading to $y_j^{\frac{h}{2}}$), it was necessary to generate some values for y which were not stored since the x's were not part of the grid

for h but would be for $\frac{h}{2}$, thus making use of an interpolation procedure such as Newton's backward difference formula or one step of single-valued method such as Runge-Kutta's(of order q).

It should be pointed out that the criterion of Step Size Adjustment based on $ERRD$ appeared to be slightly more convenient than the ones that used $ERRH$, alone or in the mixed strategy.

2.4.3 Internal Control with Extrapolation Procedure

After trying to explicitly have the value of C_{q+1}, that could be done as soon as two steps of the cyclic method (with different step sizes, of course) had been performed, a slightly different procedure was adopted.

The coefficient C_{q+1} and the finite difference which approximates $y^{(q+1)}$ at a certain value of x can be obtained, exact to order q, by comparing the various y in the following expressions. These specify that, neglecting previous errors, we may have errors only due to higher order innaccuracy, after a CM step is performed, leading to

$$\begin{aligned} y_j^h &= y(x_j) + C_{q+1} \cdot \frac{h^{q+1}}{(q+1)!} \cdot y^{(q+1)}(\xi_h) \\ y_j^{2h} &= y(x_j) + C_{q+1} \cdot \frac{(2h)^{q+1}}{(q+1)!} \cdot y^{(q+1)}(\xi_{2h}) \\ y_j^{h/2} &= y(x_j) + C_{q+1} \cdot \frac{(\frac{h}{2})^{q+1}}{(q+1)!} \cdot y^{(q+1)}(\xi_{\frac{h}{2}}) \end{aligned} \tag{3}$$

Here the notation y_j^h means the value obtained as approximation for $y(x_j)$ making use of a step size h while $y^{(q+1)}(\xi_h)$ is the $(q+1)^{th}$ order derivative of function $y(x)$ in some point of the interval $[x_{j-q}^h, x_j^h]$, thus defined by **h**.

With such estimates, and making use of the usual assumption [1] that the immediately higher order derivative varies smoothly in the integration domain, we can build the general Richardson's Extrapolation Procedure, which leads to

$$C_{q+1} \cdot \frac{h^{q+1}}{(q+1)!} \cdot y^{(q+1)}(\xi) = \frac{y_j^h - y_j^{2h}}{1 - 2^{q+1}}$$

and

$$C_{q+1} \cdot \frac{h^{q+1}}{(q+1)!} \cdot y^{(q+1)}(\xi) = \frac{y_j^h - y_j^{\frac{h}{2}}}{1 - 2^{-(q+1)}}$$

with first correction term being of order $q+2$.

The previous equations give rise to the local error estimation, used to control step size adequacy. Namely, the tests for maintenance, doubling or halving of step size are performed by checking if

$$LOCAL\_ERR = \| C_{q+1} \cdot \frac{h^{q+1}}{(q+1)!} \cdot y^{(q+1)}(\xi) \|$$

is placed in the interval $I_1 = [0, \eta \cdot \varepsilon]$, $I_2 = (\eta \cdot \varepsilon, \theta \cdot \varepsilon]$ or $I_3 = (\theta \cdot \varepsilon, \infty)$, where ε is the precision required at each step by the user. Here, the values of η and θ are chosen to allow low computational cost (that is, large step sizes), together with satisfactory precision requirements. In practice the values $\eta = 0.7$ and $\theta = 1.4$ were found to generate good performance, in agreement with previous works [1][2].

2.5 Termination

Termination is performed as soon as the Solver achieves a region which contains the end point of the interval, namely b. The interpolation procedure uses a Newton's backward finite difference formula, written as a Horner's algorithm, that leaves the termination costless.

That is, it is implemented through

$$\begin{aligned} y_{x_f} &= p_{q+1}(x_f) \\ &= y_{-[q-p+2]} + (x_f - x_{-[q-p+2]}) \cdot \{y[x_{-[q-p+2]}, x_{-[q-p+1]}] \\ &\quad + (x_f - x_{-[q-p+1]}) \cdot \{y[x_{-[q-p+2]}, x_{-[q-p+1]}, x_{-[q-p]} \\ &\quad + (x_f - x_{-q+p}) \cdots (x_f - x_{p-2}) \cdot \{y[x_{-q+p-2}, \cdots, x_{p-1}]\} \cdots\}\} \end{aligned} \tag{4}$$

where

$$y[x_{-q+p-2}, \cdots, x_{p-1}] = \frac{\Delta^{(q+1)} y(x_{-q+p-2})}{h^{q+1} \cdot (q+1)!}$$

2.6 Schematic Description

The Solver module for IVP's, named DECYC is now described through a diagram:

- DECYC
  - INITIAL
    - VALIDATION
    - INIT
  - LOOP {C1}
    - CYSTEP
      - ADAMS-BASHFORTH
      - CYCLIC-CORRECTION
    - STEP-CHECK {C2}
      - HALVE
      - DOUBLE
      - STEPS
    - RFRESH

- FINISH {C3}
  - * ABORT
  - * INTERP

In the diagram, the logical conditions C1,C2 and C3 mean:

- C1 - verifies if limit number of steps(provided by the user) and/or if the region which contains last point of integration were achieved
- C2 - tests if it is time to perform step size adjustment (period is chosen as $3 \times k$)
- C3 - checks if number of steps exceeded(performing ABORT) or normal end is to be processed(call INTERP)

As a brief description we mention the modules and their functions

- VALIDATION - verifies if the parameters provided by the user allow execution(e.g. if maximum number of steps permitted greater than one)
- INIT - prepares the environment for performing Cyclic steps, filling y's and f's, via Runge-Kutta calculations and function f calls
- CYSTEP - performs a cycle of the method, with the coefficients given by the user, through an order q prediction followed by one corrective step
- HALVE - generates the value $y_j^{\frac{h}{2}}$ with the execution of a CYSTEP with step-size halved and auxiliary values for y and f at grid points of the "halved grid" which were not part of the present one
- DOUBLE - generates the value y_j^{2h} with the execution of a CYSTEP with step-size doubled and the use of some stored y and f values obtained in previous steps
- STEPS - checks for the need of halving or doubling step-size by performing the Extrapolation or Relative Error Tests
- RFRESH - put the values of y and f in the right places inside data structure(arrays) and sets new beginning of Cyclic Interval(x_0)
- ABORT - ends the whole procedure giving control back to calling program
- INTERP - performs Newton's backward finite difference interpolation with Horner's formula to give the approximate final value of the IVP(x_f)

3 Numerical Tests

3.1 Preliminaries

In this section we list several experimental results obtained with the Solver DECYC. The environment in which such runs were performed is VS-FORTRAN in an IBM 4341 machine. We point out that the coefficients used give rise to Cyclic Methods of orders 3 and 4, namely:

- Order 4 3-cyclic 3-valued method

$\alpha_3^1 = \alpha_4^2 = \alpha_5^3 = 1$
$\alpha_3^2 = \alpha_3^3 = \alpha_4^3 = 0$
$\alpha_2^1 = \alpha_2^2 = \alpha_2^3 = -1.3668442334$
$\alpha_1^1 = \alpha_1^2 = \alpha_1^3 = 0.4243045877$
$\alpha_0^1 = \alpha_0^2 = \alpha_0^3 = -0.0574603543$
$\beta_3^1 = \beta_4^2 = \beta_5^3 = 0.535$
$\beta_2^1 = 0.5809884664; \beta_1^1 = -0.3615511793$
$\beta_0^1 = 0.0784994759$
$\beta_3^2 = 0.5443459196; \beta_2^2 = 1.3326552905$
$\beta_1^2 = -0.9598846175; \beta_0^2 = 0.2384995021$
$\beta_4^3 = 0.8889666672; \beta_3^3 = 0.4868124281$
$\beta_2^3 = 1.9481223276;$
$\beta_1^3 = -1.6090849082; \beta_0^3 = 0.4407995814$

- Order 3 3-cyclic 3-valued method

$\alpha_3^1 = \alpha_4^2 = \alpha_5^3 = 1$
$\alpha_0^2 = \alpha_0^3 = \alpha_1^3 = \alpha_2^3 = 0$
$\alpha_2^1 = \alpha_3^2 = -18.0/11.0$
$\alpha_0^1 = \alpha_1^2 = -2.0/11.0$
$\alpha_1^1 = \alpha_2^2 = 9.0/11.0$
$\alpha_3^3 = 3.0; \alpha_4^3 = -4.0$
$\beta_3^1 = \beta_4^2 = 6.0/11.0$
$\beta_2^1 = \beta_1^1 = \beta_0^1 = 0.0;$
$\beta_0^2 = \beta_1^2 = \beta_2^2 = \beta_3^2 = 0.0$
$\beta_4^3 = \beta_3^3 = -4.0/3.0$
$\beta_5^3 = 2.0/3.0$
$\beta_0^3 = \beta_1^3 = \beta_2^3 = 0.0$

In what follows we respectively name the above methods 4^{th} **order** and 3^{rd} **order** . We also make use of a pair of results to confirm the use of only one sweep through the

cyclic method [1], since an Implicit Multivalued method of order q in whose predictive step an Explicit formula of order r has been used needs $q - r + 1$ corrective steps to provide a stable result in terms of error estimation(see [2]).

The values *RATIOD* and *RATIOH* refer to the multiplicative factors that serve to force halving or doubling of step size, respectively(or α and β). Moreover, the *MAXITE* is used as an upper bound for cyclic steps to be performed and serve to calculate the first step size estimate.

About criteria used it is necessary to state that the options were *DOUBLE*, *HALVE* and *MIXED*, all performed with Extrapolation Procedure or pure Relative Error calculation.

The measure of Computational Complexity is, as usual[1][2], the number of function $f(x,y)$ calls(NCALLS), while FINAL ERROR is measured as the relative error to the exact value. That is, FINAL ERROR carries contributions from the amount of roundoff generated after satisfying the local requirements (Precision).

3.2 Results for $f(x,y) = 1 - y$

In this section we report some results obtained for $y(x) = 1 - e^{-x}$ in $x_f = 20$, through an integration starting from $x_0 = 0$. The parameter MAXITE serves as an implicit specification of the first step size to be used within the method, that is, it will provide the step size which would generate exactly MAXITE iterations if there were no step size controls. As such facility is active, step size will initially be doubled several times (for a smooth integrand) until a region of values is reached where step sizes do not change abruptly.

3.2.1 Order 3-Halve Criterion-Extrapolation

RATIOH = 1.4-*RATIOD* = 0.7-Precision:10^{-4}

MAXITE	NCALLS	FINAL ERROR
200	311	0.1693×10^{-6}
250	329	0.7990×10^{-9}
300	347	0.2470×10^{-7}
400	373	0.9464×10^{-7}
500	391	0.1286×10^{-8}
1000	459	0.1467×10^{-8}

[1] We made use of Adams-Bashforth's formulae of order q within Prediction.

3.2.2 Order 3-Halve Criterion-Extrapolation

$RATIOH = 1.4$-$RATIOD = 0.7$-Precision:10^{-6}

MAXITE	NCALLS	FINAL ERROR
300	527	0.6954×10^{-8}
400	565	0.1127×10^{-8}
500	521	0.3269×10^{-8}
700	607	0.6986×10^{-8}
1000	589	0.3182×10^{-8}

3.2.3 Order 3-Halve Criterion-Extrapolation

$RATIOH = 1.4$-$RATIOD = 0.7$-Precision:10^{-8}

MAXITE	NCALLS	FINAL ERROR
500	1381	0.1057×10^{-8}
700	1461	0.1214×10^{-8}
1000	1381	0.1186×10^{-8}
5000	1585	0.7406×10^{-9}
10000	1653	0.7386×10^{-9}
15000	1633	0.1164×10^{-8}

3.2.4 Order 3-Double Criterion-Extrapolation

$RATIOH = 1.4$-$RATIOD = 0.7$-Precision:10^{-8}

MAXITE	NCALLS	FINAL ERROR
500	1047	0.1833×10^{-8}
700	941	0.4591×10^{-8}
1000	1047	0.1665×10^{-8}
5000	1127	0.3049×10^{-9}

3.2.5 Order 4-Mixed Criterion-Extrapolation

$RATIOH = 10.0$-$RATIOD = 0.3$-Precision:10^{-6}

MAXITE	NCALLS	FINAL ERROR
1000	459	0.3062×10^{-7}
1500	503	0.8521×10^{-8}
2000	527	0.4547×10^{-8}
5000	627	0.3569×10^{-8}
6000	639	0.1750×10^{-8}
8000	663	0.5608×10^{-7}

Summarizing these results we must emphasize some aspects as, for example, that comparing the values obtained for the various tables 3.2.1, 3.2.2 and 3.2.3 global precision remains relatively stable while computational cost increases with the lowering of local error allowed (see,e.g., the cases when MAXITE = 1000). The stability of global precision is probably due to the smoothness of the function $\mathbf{y}(x)$ under consideration.

By comparing tables 3.2.3 and 3.2.4 one will certainly consider that the Doubling criterion is more effective than the Halving one. For similar global results (both not reaching maximum local requirements) the computational cost is 40% higher in the latter.

For all five tables above it is a remarkable property the fact that NCALLS remains stable in each table. This means that the step size control module is working adequately since changes of orders of magnitude in initial step size affect only slightly the number of function calls. That is, the global cost is nearly independent of initial step size, for a certain local precision.

3.3 Results for $f(x,\mathbf{y}) = -y$, $x_0 = 0$ and $x_f = 10$

$RATIOH = 1.4, RATIOD = 0.7$

3.3.1 Order 3-Double Criterion-Extrapolation- Precision:10^{-6}

MAXITE	IMAX	FINAL ERROR
100	3	0.190×10^{-1}
200	3	0.126
100	2	0.180×10^{-1}
200	2	0.120

Table 3.3.1 ilustrates that it is useless to increase the number of corrections of the implicit method to the improvement of the global result, provided that the condition discussed in 3.1 is satisfied.

3.3.2 Order 4-Double Criterion-Extrapolation- Precision:10^{-5}

MAXITE	FINAL ERROR
500	0.5330×10^{-2}
550	0.3732×10^{-2}
600	0.2713×10^{-2}
700	0.1603×10^{-1}
800	0.1904×10^{-1}

3.3.3 Order 3-Double Criterion-Relative Error

PRECISION(10^{-p})	MAXITE	FINAL ERROR	NCALLS
4	100000	0.5900×10^{-2}	1553
5	10000	0.1268×10^{-2}	1887
5	100	0.1277×10^{-2}	1863
6	1000	0.2956×10^{-3}	2579
7	10000	0.1806×10^{-4}	6455

3.3.4 Order 3-Halve Criterion-Relative Error

PRECISION(10^{-p})	MAXITE	FINAL ERROR	NCALLS
4	10000	0.1162×10^{-1}	1177
5	100	0.5693×10^{-2}	985
5	10000	0.1268×10^{-2}	1887
6	1000	0.2956×10^{-3}	2579
7	10000	0.1496×10^{-3}	3383

3.3.5 Order 4-Halve Criterion-Relative Error

PRECISION(10^{-p})	MAXITE	FINAL ERROR	NCALLS
4	100000	0.3382×10^{-1}	973
5	10000	0.4447×10^{-2}	849
6	1000	0.5727×10^{-3}	811
7	100	0.9518×10^{-4}	985

3.3.6 Order 4-Double Criterion-Relative Error

PRECISION(10^{-p})	MAXITE	FINAL ERROR	NCALLS
4	100000	0.1624×10^{-2}	1145
5	10000	0.2339×10^{-3}	1177
6	1000	0.4336×10^{-4}	1355
7	100	0.3163×10^{-4}	1863

3.4 Some Remarks

Through an inspection of previous tables we can select pairs of results which adequately ilustrate the facts that:

- It is convenient to adopt the number of corrections of the implicit method as one, since the limitation is at the method's order, and would not be overcome by increasing the number of corrective steps, once predictor and corrector have the same order (table 3.3.1).

- The results obtained through extrapolation are very poor in relation to those obtained with pure relative error evaluation, when working in a problem which deals with small absolute values(e^{-x}), although they both work well with $y = 1 - e^{-x}$
- It is clear that in some cases the halving criterion behaves badly in comparison with the doubling one. Among the reasons for this may be that it reuses values stored from a step that is being checked, and specially that the interpolation which calculates the intermediate necessary values is of the same order q of the global method.
- Important to notice that the value of $MAXITE$ affects in a smooth way the number $NCALLS$, that is, computational effort.

4 Conclusion

The implementation of a Solver for Cyclic Methods appeared to be an important environment for the proposal of mechanisms of step size control in IVP solvers.

Some of the results, when compared with others in the literature [1], bring evidence to the fact that we should apply cyclic methods of higher order to have a chance to compete against conventional Adams formulae, since the Computational cost per successful cyclic step is quite high for the global order we adopted.

A brief overview of the Tables of results will highlight the fact that the best overall performance is achieved when the Solver is set for use with the criterion of doubling alone, and moreover, with error control being made by considering pure relative error analysis.

A possible reason for the failure of the halving criterion is that the one-step method used for generating the interpolation to new grid points was of order q. This, together with the order of consistency of each equation being at most q, and the restriction made that only one cycle should be performed(that required the Solver to reuse values obtained in the step to be checked) generates errors of order at least $q+1$(eventually lower). Unnecessary to say that the failure of the halving criterion led to innadequacy also for the mixed method, though some combinations of ratios for doubling and halving stepsizes performed reasonably.

Although the validity of the scheme given by the Extrapolation procedure became clear, its practicity is still dependent of the relative distance to the values being measured, that is, it has to be considered as a first attempt to achieve a complete and robust criterion for Step Size Control. We suggest that this direction may be a source for forhthcoming studies and will lead to a stronger tool for the solution of IVP's.

List of References

1- Gear,C.W. ; "Numerical Initial Value Problems in Ordinary Differential Equations" ; Prentice-Hall (1971);

2- Shampine,L.F.& Gordon,M.K. ;"Computer Solution of Ordinary Differential Equations - The Initial Value Problem" ; W.H.Freeman and Company (1975)

3- Albrecht,P. ;"Análise Numérica-Um Curso Moderno" ed. LTC (1973)

4- Chaves,T. "Métodos Cíclicos com Extensa Região de Estabilidade" Tese de Doutorado, PUC/RJ (1979)

5- Shampine,L.F.; "Local Error Estimation by Doubling"- Computing v.34,pp. 179-190 (1985)

6- Shampine,L.F.; "Some Practical Runge-Kutta Formulas"- Mathematics of Computation v.46, no 173,pp.135-150 (1986)