Application of Bellen's parallel method to ode's with dissipative right-hand side

Philippe Chartier

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1519

Programme 1
Architectures parallèles, Bases de données,
Réseaux et Systèmes distribués

APPLICATION OF BELLEN'S PARALLEL METHOD TO ODE'S WITH DISSIPATIVE RIGHT-HAND SIDE

Philippe CHARTIER

Septembre 1991

RR-1519

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTEMES ALEATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CEDEX FRANCE
Tél. : 99 84 71 00 - Télex : UNIRISA 950 473 F
Télécopie : 99 38 38 32

Application of Bellen's parallel method to ODE's with dissipative right-hand side

Philippe Chartier*
SIMULOG
IRISA-INRIA

Publication interne n° 593 - Juin 1991 - 21 pages

August 7, 1991

Abstract

In this paper, we study different modifications of a class of parallel algorithms, initially designed by A. Bellen and M. Zennaro for difference equations and called "across the steps" methods by their authors, for the purpose of solving initial value problems in ordinary differential equations (ODE's) on a massively parallel computer. Restriction to dissipative problems is discussed which allow these problems to be solved efficiently, as shown by the simulations.

Key-words: Massively parallel, "across the steps" methods, ordinary differential equations, dissipative problems.

* Supported by a grant (Cifre) from SIMULOG.
SIMULOG, 1 rue James Joule, 78183 ST QUENTIN CEDEX

Application de la méthode de Bellen aux équations différentielles dissipatives

Résumé

A. Bellen et M. Zennaro ont développé une classe d'algorithmes parallèles destinés à résoudre les problèmes de valeur initiale pour les équations récurrentes sur des machines massivement parallèles. Nous étudions ici l'application de ces algorithmes à la classe des systèmes d'équations différentielles ordinaires dissipatifs et démontrons leur efficacité sur des simulations.

Mot-clés: Massivement parallèle, méthodes "a travers les pas", équations différentielles ordinaires, problèmes dissipatifs.

1 Introduction

In Bellen and Zennaro [3], the authors present a class of parallel algorithms for initial value problems for difference equations, that result in considerable savings in computing time. These algorithms are directly connected to initial value problems for ordinary differential equations, since the numerical solution of ODE's by a one-step method gives rise to a difference equation. They propose a Steffensen iterative method which transforms the difference equation into a linear recurrence. All computations involved in obtaining the coefficients of the recurrence can be performed in parallel, provided there are enough processors. In a second paper [5] exclusively dedicated to ODE's, they improved their algorithm by introducing step-size control. However, it has been shown that their strategy was not well-suited for message passing machines, owing to the considerable amount of time spent in communication ([4]).

In this paper, we propose a new approach designed for a restricted class of ODE's where the right-hand side function is dissipative. In Sections 2 and 3, we analyse the fixed-point problem arisen from ODE's and give a simplified version of Bellen's algorithm which is shown to be globally convergent. Section 4 presents a possible implementation on a hypercube as well as an evaluation of the performance on that architecture. Finally, numerical simulations are presented in Section 5 that proves our technique is competitive in situations where it is not possible to parallelize "across the system".

2 The fixed-point problem

In this contribution, we are interested in obtaining a numerical solution of

$$\begin{cases} y' = f(x, y(x)) \\ y(x_0) = y_0, \ x \in [x_0, X] \end{cases} \tag{1}$$

where we make the usual assumption that f is continuous and satifies a Lipschitz condition on the region $[x_0, X] \times R^m$. We also need the stronger assumption that f is continuously differentiable on the same region. Now, let $x_0, x_1, \cdots, x_N = X$ be a subdivision of $[x_0, X]$. We define $y(x, x_0, y_0)$ to be the exact solution of (1) at the point x with initial condition $y(x_0) = y_0$.

Definition 1 *For $i = 1, \cdots, N$, let $\varphi_i(u)$ represent the value of $y(x_i, x_{i-1}, u)$. φ_i is the map:*

$$\begin{aligned} \varphi_i \ : R^m &\longrightarrow R^m \\ u &\longmapsto \varphi_i(u) = y(x_i, x_{i-1}, u) \end{aligned}$$

Remark 1 *If (1) is autonomous and the grid regular, then all the φ_i's are equal.*

Definition 2 *ϕ is defined to be:*

$$\begin{aligned} \phi \ : R^{m(N+1)} &\longrightarrow R^{m(N+1)} \\ U = (u_0^T, u_1^T, \cdots, u_N^T)^T &\longmapsto \phi(U) = (y_0^T, \varphi_1(u_0)^T, \cdots, \varphi_N(u_{N-1})^T)^T \end{aligned}$$

The above definition of ϕ shows that it is possible to formulate the problem (1) as a fixed-point problem. As a matter of fact, finding the exact solution of (1) is equivalent to finding a fixed-point of ϕ. Existence and uniqueness of such a point U^* follows from the fact that $\phi^N(U) = U^*$ for any U. Hence, we shall now deal with obtaining a solution of $\phi(U^*) = U^*$ by an iterative method.

3 Solving the fixed-point problem

It is easily seen that we get an exact value for an additional component with each new application of ϕ. It leads us to the naive Algorithm:

Algorithm 1

- $U^0 = (y_0^T, y_0^T, \cdots, y_0^T)^T$
- **repeat** $(U^{k+1} = \phi(U^k))$
  $u_0^{k+1} = y_0$
  for $i = 1 \cdots N$,
  $u_i^{k+1} = \varphi_i(u_{i-1}^k)$
  end
  until $\|U^{k+1} - U^k\| \leq \epsilon$

where ϵ is a parameter that should be defined by the user. U^k will converge to the exact solution of (1). However, in order to achieve any speed-up in solving (1), it is necessary to reach a global convergence in many fewer than N iterations.

For the convenience of the reader, we recall two classical results of the theory of ordinary differential equations and the definition of the logarithmic norm (see [1]):

Theorem 1 *Suppose that v is an approximate solution of the system of differential equations (1), where f is L-Lipschitz, satisfying:*

1. $\forall x \in [x_0, X], \|v(x_0) - y(x_0)\| \leq \rho$
2. $\forall x \in [x_0, X], \|v'(x) - f(x, v(x))\| \leq \epsilon$

Then, for $x \geq x_0$ we have the error estimate:

$$\|y(x) - v(x)\| \leq \rho e^{L(x-x_0)} + \frac{\epsilon}{L}(e^{L(x-x_0)} - 1)$$

Definition 3 *Let Q be a $n \times n$ matrix and let $\|.\|$ be a norm on $R^{n \times n}$. We call:*

$$\mu(Q) = \lim_{h \to 0} \frac{\|I + hQ\| - 1}{h}$$

the logarithmic norm of Q.

Theorem 2 *Suppose that we have the estimates:*

$$\mu(\frac{\partial f}{\partial y}(x,\eta)) \leq l(x) \text{ for } \eta \in [y(x), v(x)] \text{ and}$$
$$\|v'(x) - f(x, v(x)\| \leq \delta(x), \|v(x_0) - y(x_0)\| \leq \rho.$$

Then for $x \geq x_0$ we have:

$$\|y(x) - v(x)\| \leq e^{L(x)}(\rho + \int_{x_0}^{x} e^{-L(s)}\delta(s)ds$$

with $L(x) = \int_{x_0}^{x} l(s)ds$.

Returning to our problem, we now introduce a new norm in which convergence results can be obtained.

Definition 4 *Let us assume that there exists $l \in \mathcal{L}^1([x_0, X])$ such that:*

$$\forall x \in [x_0, X], \forall y \in R^m, \mu(\frac{\partial f}{\partial y}(x,y)) \leq l(x)$$

Let $q_i = e^{\int_{x_{i-1}}^{x_i} l(x)dx}$, $i = 1, \cdots, N$ and let D denote the block-diagonal matrix:

$$D = \begin{pmatrix} d_0 1_m & 0 & \cdots & \cdots & 0 \\ 0 & d_1 1_m & & & \vdots \\ \vdots & & d_2 1_m & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & d_N 1_m \end{pmatrix}$$

where d_0 is a positive real number and $d_i = \frac{\lambda d_{i-1}}{q_i}, i = 1, \cdots, N, \lambda \in [0, 1[$. Then we define the weighted norm $\|.\|_D$ to be one of the following:

1. $\|U\|_D = \|D.U\|_1 = \sum_{i=0}^{N} d_i \|u_i\|_1$
2. $\|U\|_D = \|D.U\|_2 = \sqrt{\sum_{i=0}^{N} d_i^2 \|u_i\|_2^2}$
3. $\|U\|_D = \|D.U\|_\infty = \max_{0 \leq i \leq N} d_i \|u_i\|_\infty$

depending on the norm defining μ.

Lemma 1 *For any chosen norm in R^m, we have: $\forall i \in [1, N], \forall (u, v) \in R^m, \|\varphi_i(u) - \varphi_i(v)\| \leq q_i \|u - v\|$*

Proof: From Theorem 2, we have:

$$\|y(x_i, x_{i-1}, u) - y(x_i, x_{i-1}, v)\| \leq e^{\int_{x_{i-1}}^{x_i} l(x)dx} \|u - v\|$$

which is exactly the desired result. □

Theorem 3 *Let us assume that there exists* $l \in \mathcal{L}^1([x_0, X])$ *such that:*

$$\forall x \in [x_0, X], \forall y \in R^m, \mu(\frac{\partial f}{\partial y}(x, y)) \leq l(x)$$

then ϕ *is a contraction map with respect to the weighted norm.*

Proof: Consider the case where the 1-norm is used. Then we have:

$$\begin{aligned} \|\phi(U) - \phi(V)\|_D &= \max_{i=1,\cdots,N} d_i \|\varphi_i(u_{i-1}) - \varphi_i(v_{i-1})\|_{1,R^m} && (2) \\ &\leq \max_{i=1,\cdots,N} d_i q_i \|u_{i-1} - v_{i-1}\|_{1,R^m} && (3) \\ &\leq \lambda \max_{i=0,\cdots,N-1} d_i \|u_i - v_i\|_{1,R^m} \\ &\leq \lambda \|U - V\| \end{aligned}$$

(3) is induced by (2) as a simple application of the previous lemma. □

The above theorem shows that global convergence can be achieved using a non-uniform norm. It guarantees that the Algorithm 1 is robust, but does not insure computational efficiency. In order to accelerate the convergence, it therefore seems natural to use Newton's Algorithm. Let us define the frame of Newton's method:

Theorem 4 ϕ *is continuously differentiable on* $R^{m^{(N+1)}}$ *and* ϕ' *is a* $(N+1) \times (N+1)$ *block-matrix with block size* m *of the form:*

$$\forall U \in R^{m^{N+1}}, \phi'(U) = \begin{pmatrix} 0 & & & \\ \varphi_1'(u_0) & 0 & & \\ 0 & \varphi_2'(u_1) & & \\ \vdots & \ddots & \ddots & \\ 0 & \cdots & \varphi_N'(u_{N-1}) & 0 \end{pmatrix}$$

Proof : Since the partial derivative of f with respect to y exists and is continuous, the solution $y(x, x_{i-1}, u)$ of (1) on $[x_{i-1}, x_i]$ is differentiable with respect to u (see, for example, [1] p.97). Hence, by definition of φ_i, it is easily seen that $\varphi_i'(u) = \frac{\partial y(x_i, x_{i-1}, u)}{\partial u}$. □

We may now write down Newton's Algorithm for our problem:

Algorithm 2

- $U^0 = (y_0^T, y_0^T, \cdots, y_0^T)^T$
- **repeat** solve $U^{k+1} = U^k - (I - \phi'(U^k))^{-1}(U^k - \phi(U^k)$
  $u_0^{k+1} = y_0$
  for $i = 1 \cdots N$,
  $u_i^{k+1} = \varphi_i(u_{i-1}^k) + \varphi_i'(u_{i-1}^k)(u_{i-1}^{k+1} - u_{i-1}^k)$
  end
  until $\|U^{k+1} - U^k\| \leq \epsilon$

According to the theory of Newton's method, we have the convergence result given below in Theorem 5, whose proof can be found in [2] and relies on the following lemma:

Lemma 2 *Let us further assume that $\frac{\partial f}{\partial y}$ satisfies a Lipschitz condition with Lipschitz constant L_∂. Then there exists a real $\alpha > 0$ such that:*

$$\forall U \in R^{m^{N+1}}, \|\phi'(U) - \phi'(U^*)\| \leq \alpha \|U - U^*\|$$

where U^* denote the fixed-point of ϕ.

Proof: If $U^* = (y_0^T, \cdots, y_N^T)^T$, then we have according to the Theorem 14.3 in [1]:

$$\frac{\partial y}{\partial u_{i-1}}(x, x_{i-1}, u_{i-1}) = R(x, x_{i-1}, u_{i-1}),$$

where R is the solution of the system:

$$\left\{ \begin{array}{lcl} \frac{\partial R}{\partial x}(x, x_{i-1}, u_{i-1}) & = & \frac{\partial f}{\partial y}(x, y(x, x_{i-1}, u_{i-1}))R(x, x_{i-1}, u_{i-1}) \\ R(x_{i-1}, x_{i-1}, u_{i-1}) & = & I \end{array} \right. \tag{4}$$

Similarly,

$$\frac{\partial y}{\partial u_{i-1}}(x, x_{i-1}, y_{i-1}) = R(x, x_{i-1}, y_{i-1}),$$

where:

$$\left\{ \begin{array}{lcl} \frac{\partial R}{\partial x}(x, x_{i-1}, y_{i-1}) & = & \frac{\partial f}{\partial y}(x, y(x, x_{i-1}, y_{i-1}))R(x, x_{i-1}, y_{i-1}) \\ R(x_{i-1}, x_{i-1}, y_{i-1}) & = & I \end{array} \right. \tag{5}$$

We may then consider $R(x, x_{i-1}, y_{i-1})$ as an approximate solution of (4). Thus,

$$\begin{aligned}
&\|\frac{\partial R}{\partial x}(x, x_{i-1}, y_{i-1}) - \frac{\partial f}{\partial y}(x, y(x, x_{i-1}, u_{i-1}))R(x, x_{i-1}, y_{i-1})\| = \\
&\quad \|\frac{\partial R}{\partial x}(x, x_{i-1}, y_{i-1}) - \frac{\partial f}{\partial y}(x, y(x, x_{i-1}, y_{i-1}))R(x, x_{i-1}, y_{i-1}) - \\
&\quad [\frac{\partial f}{\partial y}(x, y(x, x_{i-1}, u_{i-1})) - \frac{\partial f}{\partial y}(x, y(x, x_{i-1}, y_{i-1}))]R(x, x_{i-1}, y_{i-1})\| \\
&\quad \leq \|\frac{\partial f}{\partial y}(x, y(x, x_{i-1}, u_{i-1})) - \frac{\partial f}{\partial y}(x, y(x, x_{i-1}, y_{i-1}))\|.\|R(x, x_{i-1}, y_{i-1})\| \\
&\quad \leq L_\partial \|y(x, x_{i-1}, u_{i-1})) - y(x, x_{i-1}, y_{i-1}))\|.\|R(x, x_{i-1}, y_{i-1})\|
\end{aligned}$$

By using the Theorem 2 and by considering $y(x, x_{i-1}, u_{i-1})$ as an approximate solution of (1), $y(x, x_{i-1}, y_{i-1}) = y_{i-1}$, we finally obtain:

$$\begin{aligned}
&\|\frac{\partial R}{\partial x}(x, x_{i-1}, y_{i-1}) - \frac{\partial f}{\partial y}(x, y(x, x_{i-1}, u_{i-1}))R(x, x_{i-1}, y_{i-1})\| \\
&\quad \leq L_\partial q_i(\sup_{x \in [x_{i-1}, x_i]} \|R(x, x_{i-1}, y_{i-1})\|).\|u_{i-1} - y_{i-1}\| \\
&\quad \leq L_i \|u_{i-1} - y_{i-1}\|.
\end{aligned}$$

where $L_i = L_\partial q_i(\sup_{x \in [x_{i-1}, x_i]} \|R(x, x_{i-1}, y_{i-1})\|)$. Now that we have estimated the defect, Theorem 1 implies:

$$\|R(x, x_{i-1}, y_{i-1}) - R(x, x_{i-1}, u_{i-1})\| \leq \frac{(L_i.\|u_{i-1} - y_{i-1}\|)}{L_\partial} e^{L_\partial(x - x_{i-1})}$$

Hence, φ_i' satisfies a Lipschitz condition at the point y_{i-1} with constant $L_i' = L_i(e^{L_\partial(x_i - x_{i-1})} - 1)/L_\partial$. The result is now straightforward for ϕ' when evaluated at the point U^* and where $\alpha = \max_{i=1,\ldots,N} L_i'$.□

Finally we sum up the hypothesis and get the promised result:

Theorem 5 *Let us assume that f is continuously differentiable on $\mathcal{R} = [x_0, X] \times R^m$ and satisfies a Lipschitz condition on $\mathcal{R}$, and that $\frac{\partial f}{\partial y}$ satisfies a Lipschitz condition on the same region $\mathcal{R}$. Then, U^* is a point of attraction of Algorithm 2 and the iteration is locally quadratically convergent.*

Proof: It is easily seen that $F = I - \phi$ satisfies the hypothesis of the "Newton Attraction Theorem" (see [2] p.312): F is continuously differentiable on $R^{m^{(N+1)}}$ and $F'(U^*)$ nonsingular, $F(U^*) = U^*$, and for all $U \in R^{m^{(N+1)}}$, $\|(I - \phi')(U) - (I - \phi')(U^*)\| \leq (\alpha + 1)\|U - U^*\|$.□

Now it is well known that Newton's iteration is highly efficient in a neighbourhood of the solution, but behaves very badly elsewhere. The smaller the first and second derivative of F

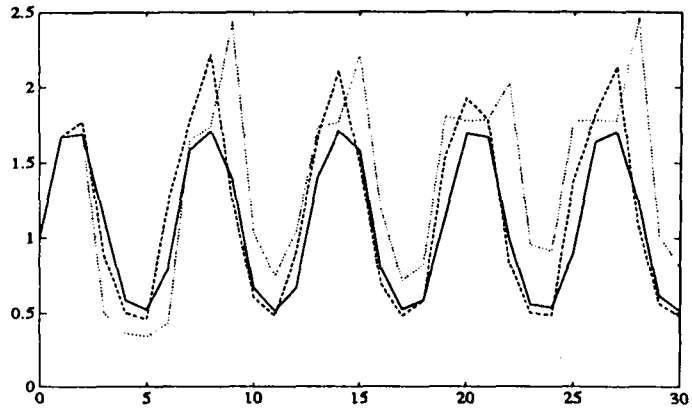

Figure 1: Algorithm 2' applied to a non-dissipative problem

(when they exist), the larger the neighbourhood and the faster the convergence. However, these quantities are known as soon as f is given. Since we do not have any information on the exact solution (except the initial condition), it is difficult to get a better estimation than the constant solution over the whole integration interval. One way to overcome this difficulty is to choose a grid fine enough to make the constant solution a good approximation, at least for the first elements of the grid (see [3]). Nevertheless this technique was shown to be inefficient when implemented on a hypercube (see [4]) due to very poor load-balancing . Moreover, step-size control is by no means obvious, which makes this technique too complex for a message-passing machine. In fig. 1 we present the first two iterations of the algorithm when it is applied to the following problem: $y' = cos(x)sin(y^2), y(x_0) = 1,\ x \in [0, 30]$ with a coarse grid. The exact solution is ploted in solid line, the numerical solution after one iteration in dashed line and after two iterations in dotted line. We can observe the disastrous second iteration that completely overshadows the first one, from which we might have expected fast convergence. This phenomenon should be attributed to the well-known instability of Newton's algorithm, when the initial guess does not belong to a neighbourhood of the solution. Therefore, it does not seem possible to handle all problems with this algorithm.

Incidentally, it seems natural to restrict ourselves to ODE problems which have a dissipative function f. We have indeed the following global convergence result:

Theorem 6 *Let us assume that there exists* $l \in \mathcal{L}^1([x_0, X])$ *such that :*

$$\forall x \in [x_0, X], \forall y \in R^m, \mu(\frac{\partial f}{\partial y}(x, y)) \leq l(x)$$

If we have either $\lambda \leq 1/3$ *or* $(N+1) < \frac{\ln(3\lambda-1)-\ln(1+\lambda)}{\ln(\lambda)}$ *then the map defined by the iteration of Algorithm 2 is a contraction with respect to the weighted norm.*

We will need first the following lemma:

Lemma 3 *For any chosen norm in R^m, we have:* $\forall i \in [1,N], \forall u \in R^m, \|\varphi_i'(u)\| \leq q_i$

Proof: By definition of φ_i, we have:

$$\varphi_i'(u) = \frac{\partial y(x_i, x_{i-1}, u)}{\partial u} = R(x_i, x_{i-1}, u)$$

where R is the solution of the differential system:

$$\begin{cases} \frac{\partial R}{\partial x}(x, x_{i-1}, u) &= \frac{\partial f}{\partial y}(x, y(x, x_{i-1}, u))R(x, x_{i-1}, u) \\ R(x_{i-1}, x_{i-1}, u) &= I \end{cases}$$

Since $R \equiv 0$ is also a solution, it follows as a simple consequence of Theorem 2 that:

$$\|R(x, x_{i-1}, u)\| \leq e^{\int_{x_{i-1}}^{x} l(s)ds} \|I\|$$

i.e. $\|\varphi_i'(u)\| \leq q_i$. □

Proof of Theorem 6: For $k \in N$ we have $U^{k+1} = U^k - (I - \phi'(U^k))^{-1}(U^k - \phi(U^k))$. Let us consider the matrix $L_k = (I - \phi'(U^k))^{-1}$. We have:

$$U^{k+1} - U^* = U^k - U^* - L_k(U^k - U^*) + L_k(\phi(U^k) - \phi(U^*))$$

Multiplying by D and taking the norm, we get:

$$\|D(U^{k+1} - U^*)\| \leq \|D(I - L_k)D^{-1}\|.\|D(U^k - U^*)\| + \|DL_kD^{-1}\|.\|D(\phi(U^k) - \phi(U^*))\|$$

We have:

$$DL_kD^{-1} = D(I - \phi')^{-1}D^{-1} = [I - D\phi'D^{-1}]^{-1}.$$

Now, let $T = [D\phi'D^{-1}]$. Since ϕ' is nilpotent we can write:

$$[I - T]^{-1} = I + \sum_{n=1}^{n=N} T^n$$

so that:

$$\|DL_kD^{-1}\| \leq 1 + \sum_{n=1}^{n=N} \|T\|^n.$$

We can compute T explicitely and obtain:

$$T = \begin{pmatrix} 0_m & \cdots & \cdots & \cdots & 0_m \\ \frac{d_1}{d_0}\varphi_1' & 0_m & & & \vdots \\ 0_m & \frac{d_2}{d_1}\varphi_2' & 0_m & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0_m & \cdots & \cdots & \frac{d_N}{d_{N-1}}\varphi_N' & 0_m \end{pmatrix}$$

According to the previous lemma, $\|\varphi'_n\| \leq q_n$, and it is easily seen that $\|T\|_1 \leq \lambda$ and that $\|T\|_\infty \leq \lambda$. As for the euclidian norm, we have $\|T\|_2 = \sqrt{\rho(TT^*)}$ where:

$$TT^* = \begin{pmatrix} \frac{d_1}{d_0}^2 \varphi'_1 \varphi'^*_1 & 0_m & \cdots & \cdots & 0_m \\ 0_m & \frac{d_2}{d_1}^2 \varphi'_2 \varphi'^*_2 & & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \frac{d_N}{d_{N-1}}^2 \varphi'_N \varphi'^*_N & 0_m \\ 0_m & \cdots & \cdots & 0_m & 0_m \end{pmatrix}$$

Hence, it follows that:

$$\rho(TT^*) = \max_{i=1,\cdots,N} \rho[(\frac{\lambda}{q_n})^2 \varphi'_n \varphi'^*_n] \leq \lambda^2 \|\frac{\varphi'_n}{q_n}\|_2^2 \|\frac{(\varphi'_n)^*}{q_n}\|_2^2 \leq \lambda^2$$

and we get the same relation $\|T\|_2 \leq \lambda$ as for the 1 and ∞ norms. In all cases, this leads us to the estimation:

$$\|DL_k D^{-1}\| \leq \frac{1-\lambda^{N+1}}{1-\lambda}$$

Similarly, we have:

$$\|I - DL_k D^{-1}\| \leq \lambda \frac{1-\lambda^N}{1-\lambda}$$

Furthermore, according to Theorem 3 we can write:

$$\|D(\phi(U^k) - \phi(U^*))\| \leq \lambda . \|D(U^k - U^*)\|$$

This finally gives us the estimation:

$$\|D(U^{k+1} - U^*)\| \leq \beta . \|D(U^k - U^*)\|$$

where:

$$\beta = \lambda \frac{1-\lambda^N}{1-\lambda} + \lambda \frac{1-\lambda^{N+1}}{1-\lambda}$$

Hence a sufficient condition for convergence is $\beta < 1$, which leads to the result.□

For the dissipative functions f, we have the following more convenient result:

Corollary 1 *Let us assume that there exists $l \in \mathcal{L}^1([x_0, X])$ such that :*

$$\forall x \in [x_0, X], \forall y \in R^m, \mu(\frac{\partial f}{\partial y}(x,y)) \leq l(x) < 0$$

Then the conclusion of Theorem 6 remains exact for $D = I$, i.e. with the norms:

1. $\|U\| = \|U\|_1$
2. $\|U\| = \|U\|_2$
3. $\|U\| = \|U\|_\infty$

Proof: The proof is just as in Theorem 6, except that we must set:

$$\lambda = q = \max_{i=1,\cdots,N} e^{\int_{x_{i-1}}^{x_i} l(x)dx} < 1$$

and set all the q_i's equal to q.□

Remark 2 *It is worth noting that in the context of the above corollary, lengthening the intervals* $[x_{i-1}, x_i]$, $i = 1, \cdots, N$ *favours both convergenve and load-balancing. This is the reason why we consider dissipative functions.*

4 Practical implementation

We now propose a slightly modified and somewhat simplified version of Bellen's Algorithm dedicated to problems where f is dissipative. We emphasize the following differences: on the one hand, we choose Newton's method instead of Steffensen's because the latter involves two sequential computations of ϕ_i compared to one for Newton, which offers an acceleration up to two. On the other hand, we gave up the part of the error control process that was aimed at reinitialising those values that are not sufficiently accurate to be reiterated, since Theorem 6 ensures reasonable behaviour of U given reasonable conditions on N. As for the accepted values, we adopt the same strategy as in [3] based on the following theorem:

Theorem 7 *Let us assume that there exists* $l \in \mathcal{L}^1([x_0, X])$ *such that :*

$$\forall x \in [x_0, X], \forall y \in R^m, \mu(\frac{\partial f}{\partial y}(x, y)) \leq l(x) < 0$$

and let us define the local error to be: $\tau^k = \phi(U^k) - U^k$. *Then, we have the following bound on* $E^k = U^* - U^k$*:*

$$\|E^k\| \leq \frac{1}{1-q}\|\tau^k\|$$

where $q = \max_{i=1,\cdots,N} e^{\int_{x_{i-1}}^{x_i} l(x)dx} < 1$.

Proof: The proof is obvious since ϕ is a contraction in this case.□

So as to reduce the size of the recurrence involved in Algorithm 2, at each iteration we shall accept as good approximations the n first components of U that pass the test $\max_{j \leq n} \|\tau_j\| \leq \epsilon$,

where ϵ is to be defined by the user and represents the tolerance on the local error. Finally, we sketch the Algorithm below and label each stage according whether it is carried out in parallel (P) or sequentially (S):

Algorithm 2'

1. set n=0, set k=0
   for $i = n+1, \cdots, N$, (P)
   set $u_i^0 = u_0^0$
   end

2. **for** $i = n+1, \cdots, N$, (P)
   compute $v_i^{k+1} = \varphi_i(u_{i-1}^k)$
   for $j = 1, \cdots, m$, (P)
   compute $w_{i,j}^{k+1} = \varphi_i(u_{i-1}^k + \eta e_j)$
   end
   end

3. **for** $i = n+1, \cdots, N$, (P)
   compute $\tau_i^{k+1} = v_i^{k+1} - u_i^k$
   compute $\|\tau_i^{k+1}\|$
   end

4. **for** $i = n+1, \cdots, N$, (P)
   for $j = 1, \cdots, m$, (P)
   compute $C_{i,j}^{k+1} = \frac{w_{i,j}^{k+1} - v_i^{k+1}}{\eta}$
   end
   end

5. **for** $i = n+1, \cdots, N$, (P)
   compute $\Delta_i^{k+1} = [C_{i,1}^{k+1}, \cdots, C_{i,m}^{k+1}]$
   end

6. set $u_n^{k+1} = u_n^k$
   set $\Theta = 0$
   for $i = n+1, \cdots, N$, (S)
   compute $\Theta = max(\Theta, \tau_i^{k+1})$
   if $\Theta < \epsilon$ **then** set $p = i$
   compute $u_i^{k+1} = v_i^{k+1} + \Delta_i^{k+1}(u_{i-1}^{k+1} - u_{i-1}^k)$
   end
   if $p = N$ **then**
   STOP

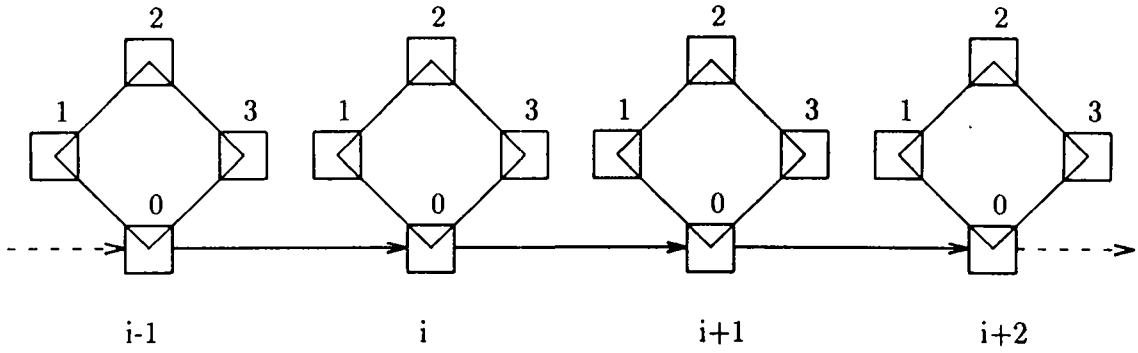

Figure 2: Model of architecture where $m = 3$

else
set $n = p$
endif

7. set $k = k + 1$
goto 2

Remark 3 *We decided to merge the two recurrences which occur naturally in the expression of Algorithm 2 in order to reduce the communication cost. However, we iterate on several values that are already accurate.*

Remark 4 *In the real implementation of the algorithm, $\varphi_i(u)$ will have to be computed by a numerical integrator with an error that should not exceed the precision asked for the whole process (i.e. ϵ).*

In order to explain the algorithm more clearly, we now present it for a specific architecture. In this model, processors are organized as a ring of clusters of $(m + 1)$ processors (see Fig. 2), where the $(m + 1)$ processors of a cluster make up a small hypercube. This architectural model is relevant since it can be easily mapped onto a grid.

Furthermore, in order to estimate the computation and communication costs, and to evaluate the speed-up factor, we introduce the following parameters:

- C: the number of floating point operations necessary to approximate the solution of (1) on $[x_0, X]$ with a given numerical solver.
- k^* : the number of iterations necessary to get an accurate approximation.
- τ_c : the average time necessary to transfer a word from one processor to a neighbour.

Step	P_i^0 $P_i^1 \quad P_i^2 \quad \cdots \quad P_i^{m-1} \quad P_i^m$
1	send u_{i-1}^k from P_i^0 $\swarrow \quad \searrow$ $P_i^1 \cdots\cdots P_i^m$
2	compute $v_i^{k+1} = \varphi_i(u_{i-1}^k)$ and $w_{i,1}^{k+1} = \varphi_i(u_{i-1}^k + \eta e_1) \cdots\cdots w_{i,m}^{k+1} = \varphi_i(u_{i-1}^k + \eta e_m)$
3	compute $\tau_i^{k+1} = v_i^{k+1} - u_i^k$ and $\|\tau_i^{k+1}\|$
4	send v_i^{k+1} from P_i^0 $\swarrow \quad \searrow$ $P_i^1 \cdots\cdots P_i^m$
5	compute $C_{i,1}^{k+1} \cdots\cdots C_{i,m}^{k+1}$ (columns of Δ_i^{k+1})
6	send $C_{i,1}^{k+1} \cdots\cdots C_{i,m}^{k+1}$ from $P_i^1 \cdots\cdots P_i^m$ $\searrow \quad \swarrow$ P_i^0
7	compute $\Delta_i^{k+1} = [C_{i,1}^{k+1}, \cdots, C_{i,m}^{k+1}]$
8	$\xrightarrow{u_{i-1}^{k+1}} \xrightarrow{\Theta}$ compute $u_i^{k+1} = v_i^{k+1} + \Delta_i(u_{i-1}^{k+1} - u_{i-1}^k)$ and compute $\Theta \xrightarrow{u_i^{k+1}} \xrightarrow{\Theta}$

Table 1: Algorithm 2' mapped onto our architecture

- β the start-up time of a communication.
- τ_{op} : the time necessary to execute one floating point operation.

Using these parameters and making the assumption that the complexity on every interval $[x_{i-1}, x_i], i = 1, \cdots, N$ is constant, we can compute the times spent for each step:

1. $\Delta t = \log_2(m+1)(\beta + m\tau_c)$
2. $\Delta t = (C/N)\tau_{op}$
3. $\Delta t = (2m-1)\tau_{op}$
4. $\Delta t = \log_2(m+1)(\beta + m\tau_c)$
5. $\Delta t = 2m\tau_{op}$
6. $\Delta t = \beta \log_2(m+1) + m(m+1)\tau_c$
7. $\Delta t = 0\tau_{op}$
8. $\Delta t = p(k)[m(2m+1)\tau_{op}] + (p(k)-1)[\beta + (m+1)\tau_c]$ where $p(k)$ represents the length of the recurrence involved in the k^{th} iteration.

If we neglect the initialization (step 1 of Algorithm 2'), then we get the "parallel time":

$$T_p = k^*[(C/N)\tau_{op} + (4m-1)\tau_{op} + m(m+1)\tau_c + 2\log_2(m+1)\tau_c + 3\log_2(m+1)\beta] + (\sum_{k=1}^{k^*} p(k))[m(2m+1)\tau_{op}] + (\sum_{k=1}^{k^*} p(k) - k^*)[\beta + (m+1)\tau_c]$$

The sequential time is,

$$T_s = C\tau_{op}$$

by definition of C.

Now we can estimate $p(k)$ in three different ways:

- (P) A very pessimistic approach is to consider that the Algorithm reaches the desired accuracy at the last iteration on all intervals $[x_{i-1}, x_i], i = 1, \cdots, N$. This hypothesis obviously leads to an over-estimation of T_p, since it is known that at least one new exact value is obtained at each iteration. Nevertheless, it shall give us an lower bound of the speed-up factor. We have, in this case,

$$S_P = \sum_{k=1}^{k^*} p(k) = k^* N.$$

Hypothesis	C_0	C_1	C_2
P	$k^*C\tau_{op}$	$k^*(4m-1)\tau_{op}$ $+k^*[2m\ln_2(m+1)+(m+1)(m-1)]\tau_c$ $+k^*[-1+3\ln_2(m+1)]\beta$	$k^*m(2m+1)\tau_{op}$ $+k^*(m+1)\tau_c$ $+k^*\beta$
O	$k^*C\tau_{op}$	$k^*(7/2m-m^2+1/2mk^*-1+m^2k^*)\tau_{op}$ $+k^*[2m\ln_2(m+1)+m^2+(k-1)(m+1)/2-1]\tau_c$ $+k^*[3\ln_2(m+1)+1/2(k^*-3)]\beta$	$m(2m+1)\tau_{op}$ $+(m+1)\tau_c$ $+\beta$
M	$k^*C\tau_{op}$	$k^*(4m-1)\tau_{op}$ $+k^*[2m\ln_2(m+1)+(m+1)(m-1)]\tau_c$ $+k^*[3\ln_2(m+1)-1]\beta$	$m(m+1/2)(k^*+1)\tau_{op}$ $+[1/2(k^*+1)(m+1)]\tau_c$ $+1/2(k^*+1)\beta$

Table 2: Values of the constants for the different hypothesis

- (O) On the contrary, we may assume that the Algorithm converges at the first iteration on all intervals except the last (k^*-1). This leads us to the following estimations of $p(k)$: $p(1)=N$ and $\forall k \in [2,k^*], p(k)=k^*+1-k$. Thus, in that case we have:

$$S_O = \sum_{k=1}^{k^*} p(k) = N + \frac{k^*(k^*-1)}{2}.$$

- (M) Finally, a perhaps more realistic assumption is that the algorithm converges regularly, that is to say that the number of intervals $[x_{i-1},x_i]$ for which u_i^k is sufficiently accurate increases as a monotone function of k. Thus we have $p(k)=N-(k-1)\frac{N}{k^*}$, so that:

$$S_M = \sum_{k=1}^{k^*} p(k) = \frac{(k^*+1)}{2}N$$

Finally, we get the speed-up factor:

$$s = \frac{T_s}{T_p} = \frac{CN}{C_0+C_1N+C_2N^2},$$

where the constants C_0, C_1 and C_2 are given in table 2. We, of course, have:

$$s_P \leq s_M \leq s_O \leq \frac{N}{k^*}$$

Remark 5 *The hypothesis (P) allows us to derive a lower bound of the optimal speed-up with respect to N that is proportionnal to the number of iterations k^*, as well as an estimation of the optimal number of intervals $N_{opt}=\sqrt{C_0/C_2}$.*

Problem	$\epsilon = 10^{-6}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-10}$
1	2.05E05	3.37E05	5.32E05
2	7.98E05	1.12E06	1.51E06
3	5.77E06	8.90E06	1.27E07

Table 3: Values of C for different tolerances

5 Numerical experiments

We performed a simulation of Algorithm 2', in order to get the value of k^* for different problems and different values of N, and then to give the corresponding estimations of the speed-up factor , as well as a lower-bound of the optimal speed-up. The code used to compute the solution of (1) on each interval $[x_{i-1}, x_i], i = 1, \cdots, N$ is the extrapolation code "Odex" from [1]. Incidentally, this code was also used to determine the value of C for each problem (see table 3).

Problem 1: $[x_0, X] = [0, 100]$

$$y'(x) = cos(y)sin(y) - 2y + e^{-x/100}sin(x^2) + ln(1+x)cos(x) \quad y(0) = 1$$

Problem 2: $[x_0, X] = [0, 100]$

$$\begin{cases} y_1'(x) = 1 + \frac{ln(1+y_2^2)}{1+y_1} + xcos^2(x) & y_1(0) = 10 \\ y_2'(x) = -2y_2\frac{ln(1+y_1)}{1+y_2^2} - 2y_2 + sin(2y_2) + cos(5x) & y_2(0) = 20 \end{cases}$$

Problem 3 : $[x_0, X] = [0, 1000]$

$$\begin{cases} y_1'(x) = -y_2 - 0.3y_1^3 + cos(3x) & y_1(0) = 1 \\ y_2'(x) = y_1 + y_3 + x^{1/5} & y_2(0) = 0 \\ y_3'(x) = -y_2 - y_4 + sin(x)\frac{ln(1+x)}{1+x^2} & y_3(0) = 0 \\ y_4'(x) = -0.01y_4^3 + y_3 - 2y_4 + 2cos(y_4) & y_4(0) = 0 \end{cases}$$

All problems (Pb1, Pb2 and Pb3) are obviously dissipative with respect to the euclidean norm so that it is relevant to apply algorithm 2'. In table 4, 5 and 6, we give the results for different values of the tolerance ϵ and for different values of N (we have taken $\beta/\tau_{op} = 69.1$ and $\tau_c/\tau op = 0.3$, which are the values observed on an IPSC/2). As no information on complexity of the numerical integration with respect to x is available, we have considered regular subdivisions and have consequently assumed that the computing work was regularly distributed on the whole interval $[x_0, X]$.

Problem N	10	20	40	60	80	100	150	200
$s_P(Pb1)$	3.21	5.80	6.32	6.53	6.06	5.46	3.32	2.62
$s_M(Pb1)$	3.25	6.05	7.31	8.25	8.17	7.69	5.14	4.17
$s_O(Pb1)$	3.27	6.28	8.49	10.93	12.21	12.66	10.89	9.88
$s_P(Pb2)$	1.97	3.83	6.85	8.74	9.65	8.24	7.61	6.60
$s_M(Pb2)$	1.98	3.89	7.25	9.77	11.42	10.39	10.66	9.87
$s_O(Pb2)$	1.99	3.94	7.63	10.95	13.79	13.75	17.32	18.96
$s_P(Pb3)$	2.49	6.61	9.70	14.03	17.82	21.02	21.08	22.88
$s_M(Pb3)$	2.50	6.62	9.80	14.36	18.56	22.32	23.87	27.53
$s_O(Pb3)$	2.50	6.65	9.93	14.77	19.49	24.05	28.02	35.65

Table 4: Speed-up factors for $\epsilon = 10^{-6}$

Problem N	10	20	40	60	80	100	150	200
$s_P(Pb1)$	2.44	6.11	7.39	8.38	8.34	6.29	5.10	4.14
$s_M(Pb1)$	2.46	6.28	8.17	10.01	10.64	8.64	7.61	6.43
$s_O(Pb1)$	2.47	6.43	9.03	12.23	14.41	13.37	14.51	14.01
$s_P(Pb2)$	1.98	3.23	5.95	7.90	9.08	9.64	9.51	7.35
$s_M(Pb2)$	1.99	3.27	6.22	8.63	10.43	11.65	12.76	10.74
$s_O(Pb2)$	1.99	3.30	6.46	9.42	12.10	14.48	19.00	19.31
$s_P(Pb3)$	3.33	4.97	9.80	14.35	18.53	17.81	23.55	26.93
$s_M(Pb3)$	3.33	4.98	9.87	14.58	19.04	18.61	25.72	30.92
$s_O(Pb3)$	3.33	4.99	9.95	14.85	19.67	19.58	28.69	37.07

Table 5: Speed-up factors for $\epsilon = 10^{-8}$

Problem N	10	20	40	60	80	100	150	200
$s_P(Pb1)$	2.46	4.73	8.17	8.00	8.49	8.40	7.33	6.16
$s_M(Pb1)$	2.48	4.82	8.76	9.21	10.42	10.91	10.47	9.29
$s_O(Pb1)$	2.48	4.90	9.36	10.69	13.25	15.22	17.90	18.39
$s_P(Pb2)$	0.99	2.17	5.25	6.27	5.40	5.90	9.71	7.96
$s_M(Pb2)$	1.00	2.19	5.43	6.74	6.09	7.00	12.63	11.30
$s_O(Pb2)$	1.00	2.20	5.59	7.20	6.86	8.38	17.71	18.90
$s_P(Pb3)$	2.50	4.98	7.89	11.63	15.16	18.41	25.17	24.88
$s_M(Pb3)$	2.50	4.99	7.93	11.77	15.47	19.00	26.87	27.77
$s_O(Pb3)$	2.50	4.99	7.98	11.93	15.84	19.70	29.07	31.82

Table 6: Speed-up factors for $\epsilon = 10^{-10}$

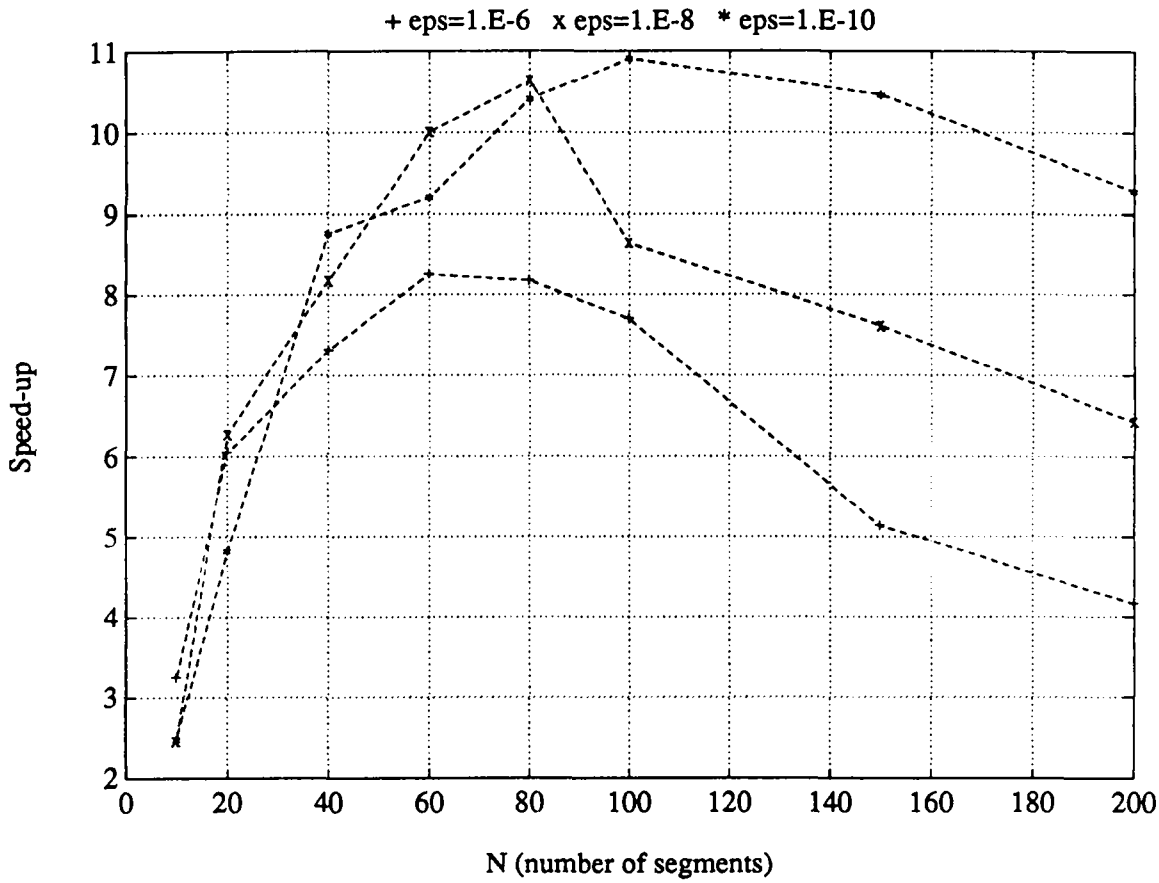

Figure 3: Performance of algorithm 2' for problem 1

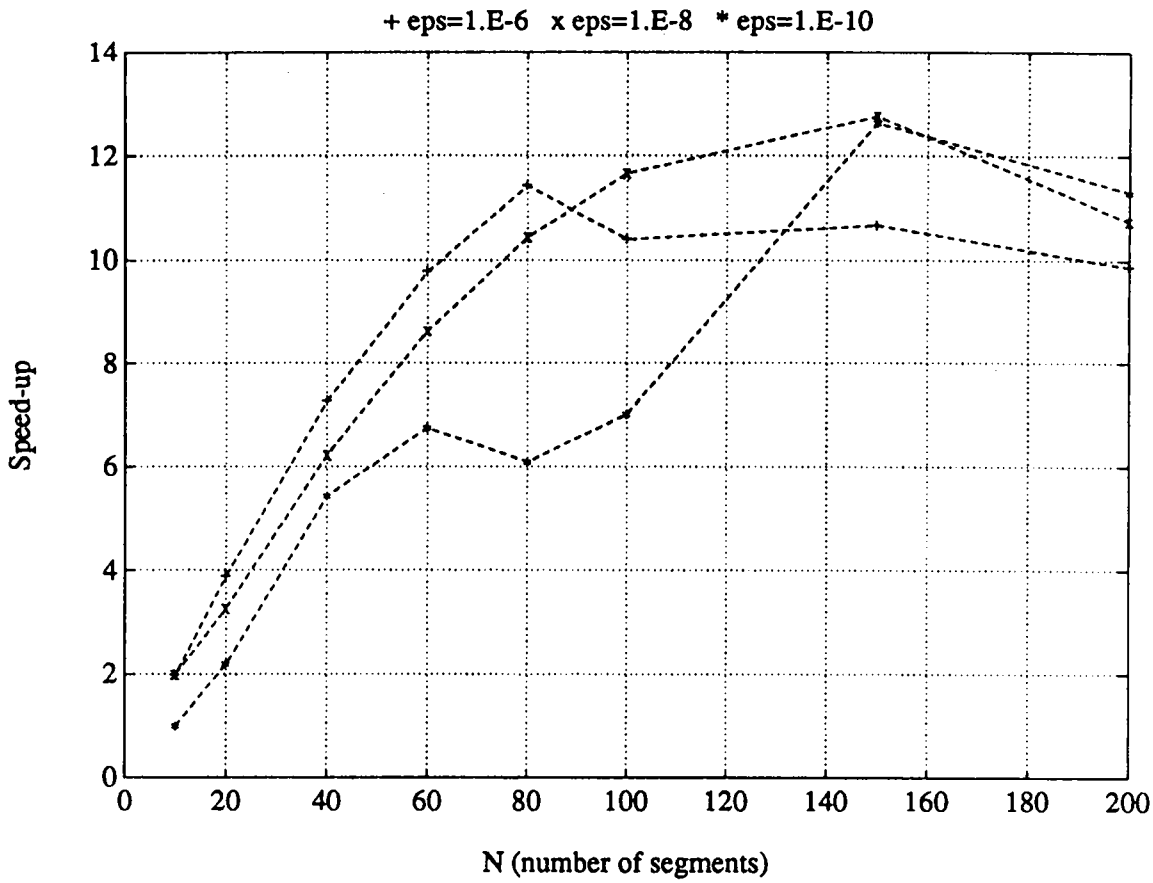

Figure 4: Performance of algorithm 2' for problem 2

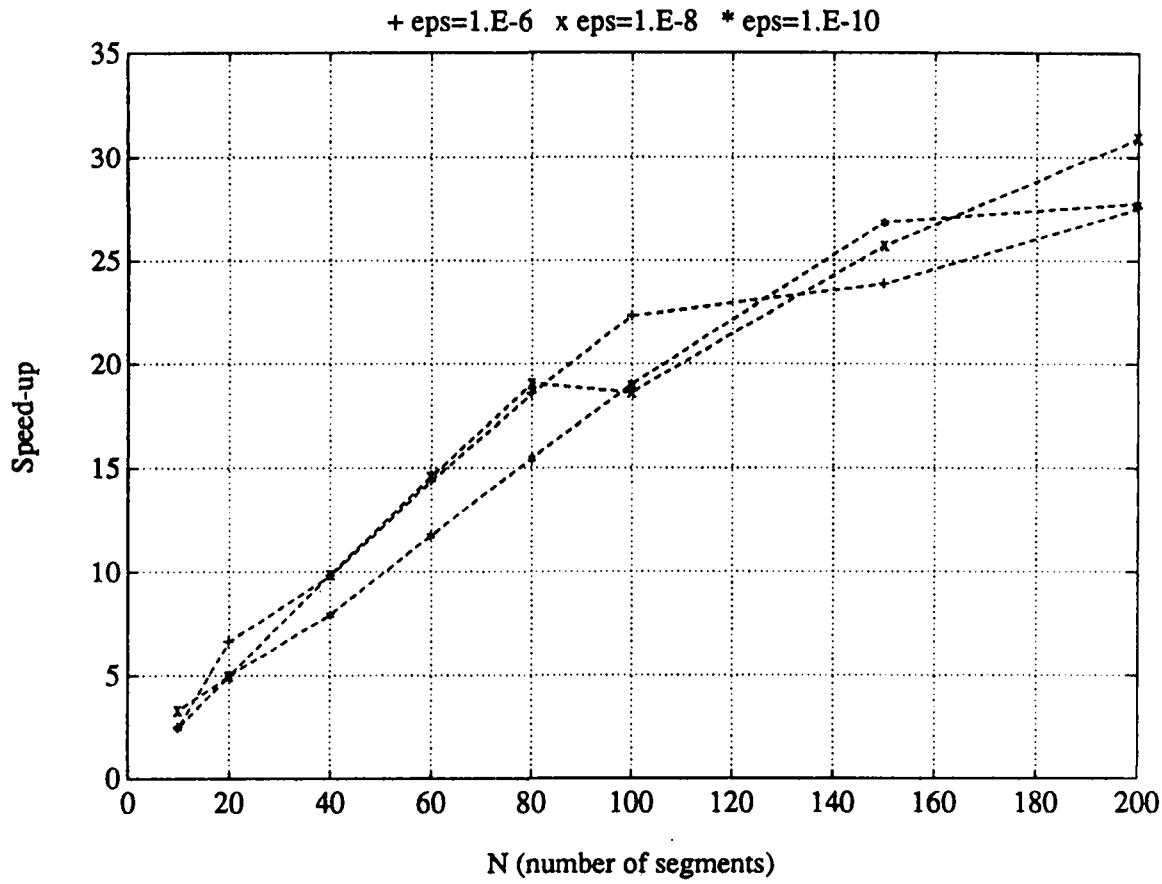

Figure 5: Performance of algorithm 2' for problem 3

6 Conclusion

A modified version of Bellen and Zennaro's algorithm for the integration of ordinary differential equation with dissipative functions is proposed. Global convergence is shown for dissipative problems and for reasonable conditions on the number of segments. This enables us to give up part of the error control process of the original algorithm and consequently to reduce the communication cost. Computer simulations have been carried out on an architectural model that can be easily mapped onto a grid. We proved that significant speed-up can be achieved with this model using an analysis that takes communication delays into account. The use of the algorithm is obviously restricted. However, further investigations could reveal that the larger class of problems where the solution is bounded and for an appropriate choice of the length of segments has similar properties. Our next step is to examine real implementations so as to analyse the behaviour of the algorithm in cases where the complexity varies from one subdivision to another.

References

[1] E. Hairer, S.P. Norsett and G. Wanner *Solving Ordinary Differential Equations I. Nonstiff Problems* (Springer, Berlin, 1987)

[2] J.M. Ortega and W.C. Rheinbolt *Iterative Solution of Nonlinear Equations in Several Variables* (Academic Press, New-York, 1970)

[3] A. Bellen and M. Zennaro *Parallel Algorithms for Initial Value Problems for Difference and Differential Equations* in J. Comput. Appl. Math. 25 (1989), 341-350

[4] R. Vermiglio *Parallel Step Methods for Difference and Differential Equations: implementation on a hypercube* (Internal Report)

[5] A. Bellen, R. Vermiglio and M. Zennaro *Parallel ODE-solvers with step-size control* (Preprint)

LISTE DES DERNIERES PUBLICATIONS INTERNES IRISA

PI 587 ON FAILURE DETECTION AND IDENTIFICATION : AN OPTIMUM ROBUST MIN-MAX APPROACH
Elias WAHNON, Albert BENVENISTE
Mai 1991, 24 Pages.

PI 588 BOUNDED-MEMORY ALGORITHMS FOR VERIFICATION ON THE FLY
Claude JARD, Thierry JERON
Mai 1991, 14 pages.

PI 589 UNE APPROCHE MULTIECHELLE A L'ANALYSE D'IMAGES PAR CHAMPS MARKOVIENS
Patrick PEREZ, Fabrice HEITZ
Juin 1991, 32 pages.

PI 590 THE IDEMPOTENT SOLUTIONS OF THE SEMI-UNIFICATION PROBLEM
Pascal BRISSET, Olivier RIDOUX
Juin 1991, 16 pages.

PI 591 AVARE UN PROGRAMME DE CALCUL DES ASSOCIATIONS ENTRE VARIABLES RELATIONNELLES
Mohamed OUALI ALLAH
Juin 1991, 32 pages.

PI 592 SCHEDULING IN DISTRIBUTED SYSTEMS : SURVEY AND QUESTIONS
Yasmina BELHAMISSI, Maurice JEGADO
Juin 1991, 36 pages.

PI 593 APPLICATION OF BELLEN'S PARALLEL METHOD TO ODE's WITH DISSIPATIVE RIGHT-HAND SIDE
Philippe CHARTIER
Juin 1991, 24 pages.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique

ISSN 0249 - 6399