Computing 51, 209–236 (1993)

Computing

A Parallel Shooting Technique for Solving Dissipative ODE's

P. Chartier*, St Quentin Yvelines, and **B. Philippe**, Rennes

Received November 16, 1992; revised July 30, 1993

Abstract — Zusammenfassung

A Parallel Shooting Technique for Solving Dissipative ODE's. 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.

AMS Subject Classification: 65L05, 65W05, 65Q05

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

Eine parallele "shooting" Technik zur Lösung dissipativer gewöhnlicher Differentialgleichungen. In diesem Artikel studieren wir verschiedene Versionen einer Klasse paralleler Algorithmen, die ursprünglich von A. Bellen und M. Zennaro für Differenzengleichungen konzipiert und von ihnen "across the steps" Methode genannt worden ist. Die Autoren verfolgten den Zweck, Anfangswertprobleme bei gewöhnlichen Differentialgleichungen anhand eines massiv parallelen Rechner zu lösen. Wir behandeln die Anwendung auf dissipative Systeme und erreichen eine effiziente Lösung dieser Probleme. Dies wird in einigen Simulationen illustriert.

Part 1: Theoretical Analysis

1. Introduction

Parallel algorithms for solving initial value problems (IVP's) for differential equations have received only marginal attention in the literature compared to the enormous work devoted to parallel algorithms for linear algebra. It is indeed generally admitted that the integration of a system of ordinary differential equations in a step-by-step process is inherently sequential. However, a few solutions to circumvent this barrier have been proposed. A rather obvious way to parallelize is to distribute the components of the right-hand-side of the system amongst the available processors. This technique, called "across the problem" by Gear (see [7]) is rather effective for large systems. Parallelism "across the method" exploits the

* Supported in part by the ONERA and by the DRET under grant n° 89.34.401.00.470.75.01

parallelism available within the method itself. This idea has led to various methods such as block methods (see [3, 5, 6, 14, 15]) or "parallel iterated Runge-Kutta methods" (see [16]). However, the accelerations obtained are generally kept below ten.

Still another approach, that we will focus on here, has been considered by A. Bellen and M. Zennaro. In [2], 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 [1] 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 (see [17]).

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 particular, this assumption helps considerably to improve load-balancing.

In [2], the authors study the fixed-point problem arisen from the application of a step-by-step method to a system of ODE's. In Section 2, we adopt a similar approach. However, our formulation is based on the exact solution of the differential system with no reference to any numerical step-by-step method. In the sequel, this new formulation is used to determine what kind of IVP's should be considered.

A very simple algorithm aimed at solving the fixed-point problem is introduced in Section 3, and its convergence is studied in terms of the logarithmic norm of the right-hand-side. This algorithm already emphasizes the necessity of dealing only with a restricted class of problems.

Newton's method is then considered in Section 4. We first make a non-restrictive assumption on the IVP to derive the usual "local quadratic convergence" of Newton's method for our problem. The main result of the paper, establishing sufficient conditions for *global convergence*, is then presented. The class of dissipative problems is shown to be particularly appropriate.

Section 5 presents numerical experiments aimed at verifying the theoretical *convergence* results for the proposed algorithm.

In Section 6 of Part 2, we will study the discrete-time version of the algorithm. Especially the influence of the perturbations arisen from the introduction of approximations will be analysed.

Finally, Sections 7 and 8 of Part 2 will be devoted to the implementation on a hypercube as well as the evaluation of the performance on that architecture. Simulations will be presented that prove 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 the initial-value problem for the m-dimensional system of ODE's

$$y'(x) = f(x, y(x)), \tag{1}$$

$$y(x_0) = y_0, \tag{2}$$

on the interval of $\mathbb{R}$, $[x_0, X]$. We make the usual assumption that f is continuous and satisfies a Lipschitz condition on the region $[x_0, X] \times \mathbb{R}^m$. We also need the stronger assumption that f is continuously differentiable on the same region. Now, let $x_0, x_1, \ldots, x_N = X$ 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 $i = 1, \ldots, N$, let $\varphi_i(u)$ represent the value of $y(x_i, x_{i-1}, u)$. φ_i is the map:

$$\begin{aligned} &\varphi_i\colon \mathbb{R}^m \to \mathbb{R}^m \\ &\qquad u \mapsto \varphi_i(u) = y(x_i, x_{i-1}, u) \end{aligned} \tag{3}$$

Remark 1. *The functions φ_i's are well-defined owing to the existence and uniqueness of the solution of* (1) *on any sub-interval of $[x_0, X]$ and for any initial condition.*

Remark 2. *If* (1) *is autonomous and the grid is regular, then all the φ_i's are equal.*

Definition 2. *Φ is defined to be:*

$$\begin{aligned} &\Phi\colon \mathbb{R}^{m\times(N+1)} \to \mathbb{R}^{m\times(N+1)} \\ &U = (u_0^T, u_1^T, \ldots, u_N^T)^T \mapsto \Phi(U) = (y_0^T, \varphi_1(u_0)^T, \ldots, \varphi_N(u_{N-1})^T)^T \end{aligned} \tag{4}$$

Let us illustrate the definitions on a very simple example:

Example 1. *For the Prothero-Robinson problem (see* [13]*):*

$$\begin{cases} y'(x) = -A(y(x) - \Psi(x)) + \Psi'(x) \\ y(x_0) = \Psi(x_0), x \in [x_0, X] \end{cases} \tag{5}$$

where A is a symmetric positive definite $m \times m$ real matrix, we have:

$$\varphi_i(u) = \Psi(x_i) + e^{-A(x_i - x_{i-1})}(u - \Psi(x_{i-1})) \tag{6}$$

Hence the following bound holds:

$$\|\varphi_i(u) - \Psi(x_i)\| \leq \rho \|u - \Psi(x_{i-1})\| \tag{7}$$

where $\rho = \|e^{-A(x_i - x_{i-1})}\| < 1$ for a suitable choice of the norm $\|\cdot\|$. This property motivates the algorithms considered in the sequel.

The above definition of Φ shows that it is possible to formulate the problem (1, 2) as a fixed-point problem. As a matter of fact, finding the exact solution of (1, 2) is equivalent to finding a fixed-point of Φ. Existence and uniqueness of such a point U^* follows from the fact that $\Phi^{N+1}(U) = U^*$ for any U, where $\Phi^{N+1}(U)$ denotes the $(N+1)^{th}$ application of Φ. Hence an iterative method will be suitable for obtaining the fixed point of Φ.

3. Solving the Fixed-Point Problem

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

Algorithm 1.

- $U^0 = (y_0^T, y_0^T, \ldots, y_0^T)^T$
- **repeat** *[*** compute* $(U^{k+1} = \Phi(U^k))$ ****]*
  $u_0^{k+1} = y_0$
  for $i = 1 \ldots N$,
  $u_i^{k+1} = \varphi(u_{i-1}^k)$
  end
  until $\|U^{k+1} - U^k\| \leq \varepsilon$

where ε is a parameter that should be defined by the user. U^k will converge to the exact solution of (1, 2) within N iterations. However, in order to achieve any speed-up in solving (1, 2), it is necessary to reach a global convergence in many fewer than N iterations.

For convenience, we recall two classical results of the theory of ordinary differential equations and the definition of the logarithmic norm (see [8]). In the sequel, $\|\cdot\|$ will denote both a vector norm on $\mathbb{R}^m$ and the associated matrix norm.

Theorem 1. *Suppose that v is an approximate solution of the system of differential equations* (1, 2), *where f is L-Lipschitz, satisfying:*

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

where ρ is the initial error and ε the defect of the approximate solution v. Then, for $x \geq x_0$ we have the error estimate:

$$\|y(x) - v(x)\| \leq \rho e^{L(x-x_0)} + \frac{\varepsilon}{L}(e^{L(x-x_0)} - 1) \tag{8}$$

Definition 3. *Let $Q = (q_{i,j})_{1 \leq i,j \leq n}$ be a real $n \times n$ matrix and let $\|\cdot\|$ be a norm on $\mathbb{R}^{n \times n}$ associated with a vector norm. We call:*

$$\mu(Q) = \lim_{h \to 0} \frac{\|I + hQ\| - 1}{h} \tag{9}$$

the logarithmic norm of Q.

The logarithmic norm can be estimated as follows:

- For the Euclidean norm

$$\mu(Q) = \max\left\{\lambda/\det\left(\frac{1}{2}(Q + Q^T) - \lambda I\right) = 0\right\}. \tag{10}$$

- For the ∞-norm

$$\mu(Q) = \max_k\left(q_{kk} + \sum_{i \neq k} |q_{ki}|\right). \tag{11}$$

- For the 1-norm

$$\mu(Q) = \max_i\left(q_{ii} + \sum_{k \neq i} |q_{ki}|\right). \tag{12}$$

Example 2. *Consider the two-body system in two dimensions*

$$\begin{cases} y_1'(x) = y_2(x) \\ y_2'(x) = -\dfrac{\mu y_1(x)}{(y_1(x)^2 + y_3(x)^2)^{3/2}} \\ y_3'(x) = y_4(x) \\ y_4'(x) = -\dfrac{\mu y_3(x)}{(y_1(x)^2 + y_3(x)^2)^{3/2}}. \end{cases} \tag{13}$$

Computation of the Jacobian matrix leads to:

$$\frac{1}{2}\left(\frac{\partial f}{\partial y} + \frac{\partial f^T}{\partial y}\right)$$

$$= \begin{bmatrix} 0 & 1/2 + \dfrac{\mu(2y_1^2 - y_3^2)}{2(y_1^2+y_3^2)^{5/2}} & 0 & \dfrac{3\mu y_1 y_3}{2(y_1^2+y_3^2)^{5/2}} \\ 1/2 + \dfrac{\mu(2y_1^2 - y_3^2)}{2(y_1^2+y_3^2)^{5/2}} & 0 & \dfrac{3\mu y_1 y_3}{2(y_1^2+y_3^2)^{5/2}} & 0 \\ 0 & \dfrac{3\mu y_1 y_3}{2(y_1^2+y_3^2)^{5/2}} & 0 & 1/2 - \dfrac{\mu(y_1^2 - 2y_3^2)}{2(y_1^2+y_3^2)^{5/2}} \\ \dfrac{3\mu y_1 y_3}{2(y_1^2+y_3^2)^{5/2}} & 0 & 1/2 - \dfrac{\mu(y_1^2 - 2y_3^2)}{2(y_1^2+y_3^2)^{5/2}} & 0 \end{bmatrix} \tag{14}$$

whose eigenvalues are ($r = \sqrt{y_1^2 + y_3^2}$):

$$\lambda = \pm\left(\frac{1}{2} + \frac{\mu}{r^3}\right), \tag{15}$$

$$\lambda = \pm\left(\frac{1}{2} - \frac{\mu}{2r^3}\right), \tag{16}$$

thus the logarithmic norm of $\dfrac{\partial f}{\partial y}$ *for the Euclidean norm is simply:*

$$\mu\left(\frac{\partial f}{\partial y}\right) = +\frac{1}{2} + \frac{\mu}{r^3}. \tag{17}$$

Remark 3. *For Example* 1, *we have straightforwardly,*

$$\mu\left(\frac{\partial f}{\partial y}\right) = -\xi \tag{18}$$

where ξ *is the smallest eigenvalue of* A.

Theorem 2. *Let us assume that there exist two real functions* l *and* δ, *and a positive number* ρ *such that:*

$$\forall x \in [x_0, X], \qquad \forall \eta \in [y(x), v(x)], \qquad \mu\left(\frac{\partial f}{\partial y}(x, \eta)\right) \leq l(x), \tag{19}$$

$$\forall \in [x_0, X], \qquad \|v'(x) - f(x, v(x))\| \leq \delta(x), \tag{20}$$

$$\|v(x_0) - y(x_0)\| \leq \rho. \tag{21}$$

As in Theorem 1, ρ *is called the initial error and* δ *the defect of the approximate solution* v. *Then for* $x \geq x_0$ *we have:*

$$\|y(x) - v(x)\| \leq e^{L(x)}\left(\rho + \int_{x_0}^{x} e^{-L(s)}\delta(s)\,ds\right) \tag{22}$$

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* $l \in L^1([x_0, X])$ *(i.e. summable over* $[x_0, X]$*) such that:*

$$\forall x \in [x_0, X], \qquad \forall y \in \mathbb{R}^m, \qquad \mu\left(\frac{\partial f}{\partial y}(x, y)\right) \leq l(x). \tag{23}$$

Let $q_i \geq e^{\int_{x_{i-1}}^{x_i} l(x)\,dx}$, $i = 1, \ldots, N$, $\lambda \in]0, 1[$ *and let* D *denote the block-diagonal matrix:*

$$D = \operatorname{diag}(d_0 1_m, \ldots, d_N 1_m) \tag{24}$$

where $d_0 = 1$ *and* $d_i = \dfrac{\lambda d_{i-1}}{q_i}$, $i = 1, \ldots, N$. *Then for any vector* $U = (u_0^T, \ldots, u_N^T)^T \in \mathbb{R}^{m\times(N+1)}$ *we define the weighted norm* $\|\cdot\|_D$ *to be one of the following:*

$$\|U\|_D = \|D \cdot U\|_1 = \sum_{i=0}^{N} d_i \|u_i\|_1 \tag{25}$$

$$\|U\|_D = \|D \cdot U\|_2 = \sqrt{\sum_{i=0}^{N} d_i^2 \|u_i\|_2^2} \tag{26}$$

$$\|U\|_D = \|D \cdot U\|_\infty = \max_{0 \le i \le N} d_i \|u_i\|_\infty \tag{27}$$

depending on the norm defining μ.

To prove convergence results, we will need the following lemma:

Lemma 1. *Let us assume that there exists* $l \in L^1([x_0, X])$ *such that* (23) *is satisfied. Then for any chosen norm in* $\mathbb{R}^m$, *we have*

$$\forall i \in [1, N], \qquad \forall (u, v) \in \mathbb{R}^m \times \mathbb{R}^m, \qquad \|\varphi_i(u) - \varphi_i(v)\| \le q_i \|u - v\| \tag{28}$$

Proof: From Theorem 2, we have:

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

which is exactly the desired result. □

Theorem 3. *Let us assume that there exists* $l \in L^1([x_0, X])$ *such that* (23) *is satisfied. Then* Φ *is a contraction map with respect to the weighted norm.*

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

$$\|\Phi(U) - \Phi(V)\|_D = \max_{i=1,\dots,N} d_i \|\varphi_i(u_{i-1}) - \varphi_i(v_{i-1})\|_\infty \tag{29}$$

$$\le \max_{i=1,\dots,N} d_i q_i \|u_{i-1} - v_{i-1}\|_\infty \tag{30}$$

$$= \lambda \max_{i=0,\dots,N-1} d_i \|u_i - v_i\|_\infty$$

$$\le \lambda \|U - V\|_D$$

(30) is induced by (29) as a simple application of the previous lemma. □

This theorem establishes in a general context the contraction property of Φ already observed for the Prothero-Robinson problem (see Example 1).

Definition 5. *An initial value problem* (1, 2) *is said to be dissipative iff*

$$\forall x \in [x_0, X], \qquad \forall y \in \mathbb{R}^m, \qquad \mu\left(\frac{\partial f}{\partial y}(x, y)\right) \le l(x) < 0. \tag{31}$$

For dissipative problems, we now have the following more convenient result:

Corollary 1. *Let us assume that condition* (31) *is satisfied. Then the conclusion of Theorem* 3 *remains true for* $D = I$, *i.e. with the norms* $\|U\| = \|U\|_1$, $\|U\| = \|U\|_2$ *and* $\|U\| = \|U\|_\infty$.

Proof: The proof is just as in Theorem 3, except that we now must set:

$$\lambda = q = \max_{i=1,\dots,N} e^{\int_{x_{i-1}}^{x_i} l(x)\,ds} < 1 \tag{32}$$

and set all the q_i's equal to q. □

4. Convergence Results for Newton's Algorithm

The above theorem shows that global convergence can be achieved using a non-uniform norm. It guarantees that Algorithm 1 is robust, but does not ensure computational efficiency. Considering the convergence in the weighted norm $\|\cdot\|_D$ is only relevant if D is close to the identity (i.e. $\|\cdot\|_D$ close to a uniform norm). Hence, the rate of convergence is determined by the highest q_i of Definition 4. 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 $\mathbb{R}^{m\times(N+1)}$ and Φ' is a $(N+1)\times(N+1)$ block bidiagonal matrix with block size m of the form*

$$\forall U \in \mathbb{R}^{m\times(N+1)}, \qquad \Phi'(U) = \begin{bmatrix} 0 & & & \\ \varphi_1'(u_0) & 0 & & \\ 0 & \varphi_2'(u_1) & & \\ \vdots & \ddots & \ddots & \\ 0 & \dots & \varphi_N'(u_{N-1}) & 0 \end{bmatrix} \tag{33}$$

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, [8] p. 97). Hence, by definition of φ_i, it is easily seen that $\varphi_i'(u) = \dfrac{\partial y(x_i, x_{i-1}, u)}{\partial u}$. □

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

Algorithm 2.
- $U^0 = (y_0^T, y_0^T, \dots, y_0^T)^T$
- **repeat** *[*** solve* $U^{k+1} = U^k - (I - \Phi'(U^k))^{-1}(U^k - \Phi(U^k))$ ****]*
  $u_0^{k+1} = y_0$
  for $i = 1 \dots N$,
  $u_i^{k+1} = \varphi_i(u_{i-1}^k) + \varphi_i'(u_{i-1}^k)(u_{i-1}^{k+1} - u_{i-1}^k)$
  end
  until $\|U^{k+1} - U^k\| \le \varepsilon$

According to the theory of Newton's method, we have the convergence result given below in Theorem 5, whose proof can be found in [12]. The following lemma ensures that its hypotheses are satisfied:

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

$$\forall U \in \mathbb{R}^{m\times(N+1)}, \qquad \|\Phi'(U) - \Phi'(U^*)\| \le \alpha \|U - U^*\| \tag{34}$$

where U^* denotes the fixed-point of Φ.

Proof: The proof is standard and can be found in [4]. □

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

Theorem 5. *Let us assume that f is continuously differentiable on $\mathcal{R} = [x_0, X] \times \mathbb{R}^m$ 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 hypotheses of the "Newton Attraction Theorem" (see [12] p. 312): F is continuously differentiable on $\mathbb{R}^{m\times(N+1)}$ and $F'(U^*)$ nonsingular, $F(U^*) = 0$, and for all $U \in \mathbb{R}^{m\times(N+1)}$, $\|(I - \Phi')(U) - (I - \Phi')(U^*)\| \leq (\alpha + 1)\|U - U^*\|$. □

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 (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 estimate 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 [2]). Nevertheless this technique was shown to be inefficient when implemented on a hypercube (see [17]) 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 (Oxy-plane) we present the first two iterations of Algorithm 2 when applied to the following problem:

$$\begin{cases} y' = \cos(x)\sin(y^2), \\ y(x_0) = 1, x \in [0, 30] \end{cases} \tag{35}$$

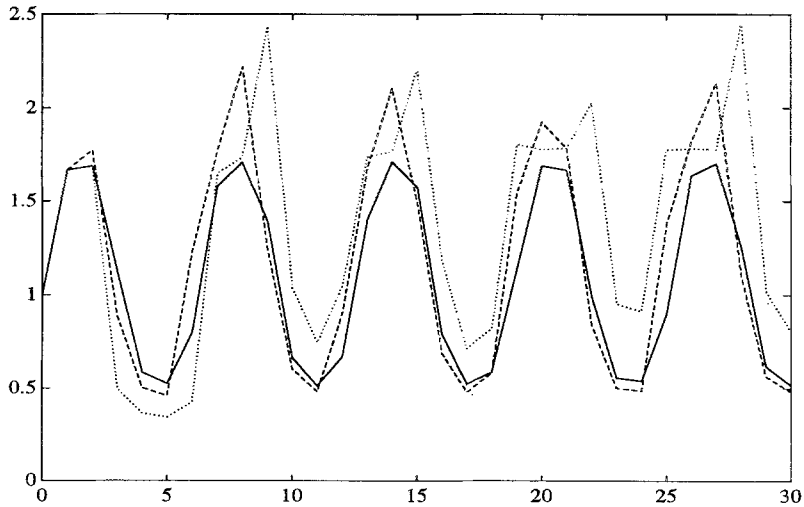

Figure 1. Algorithm 2 applied to the non-dissipative problem (35)

with a coarse grid. The exact solution is plotted 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 suitable neighbourhood of the solution. The same phenomenon may be observed on Example 2: whereas Algorithm 2 is robust, the norm $\|\cdot\|_D$ in which convergence results are derived is dramatically far from uniform. Let us suppose for example that the solution of (13) is a circulant movement of speed ω. We have in that case $\mu/r^3 = \omega^2$, so that

$$\mu\left(\frac{\partial f}{\partial y}\right) = 1/2 + \omega^2.$$

Hence, the q_i's of Definition 4 are very large as soon as ω is large and the decrease of $\|U^k\|$ is not of practical interest. This example emphasizes the crucial importance of $\|\cdot\|_D$, which is directly connected to the function $l(x)$ involved in (23). Large positive functions $l(x)$ prevent Algorithm 2 to be efficient. Therefore, it does not seem possible to handle all problems with this algorithm, whereas it seems natural to restrict ourselves to ODE problems which have a dissipative function f. We have indeed the following global convergence result, and its corollary for dissipative right-hand side which gives the main result of the paper:

Theorem 6. *Suppose that* (23) *is satisfied. If we have either* $\lambda \leq 1/3$ *or* $(N+1) < \dfrac{\ln(3\lambda - 1) - \ln(1+\lambda)}{\ln(\lambda)}$ *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 matrix-norm on* $\mathbb{R}^{m\times m}$, *we have*

$$\forall i \in [1, N], \qquad \forall u \in \mathbb{R}^m, \qquad \|\varphi_i'(u)\| \leq q_i. \tag{36}$$

Proof: By definition of φ_i, we have:

$$\varphi_i'(u) = \frac{\partial y(x_i, x_{i-1}, u)}{\partial u} = R(x_i, x_{i-1}, u) \tag{37}$$

where R is the solution of the differential system:

$$\begin{cases} \dfrac{\partial R}{\partial x}(x, x_{i-1}, u) = \dfrac{\partial f}{\partial y}(x, y(x, x_{i-1}, u))R(x, x_{i-1}, u) \\ R(x_{i-1}, x_{i-1}, u) = I \end{cases} \tag{38}$$

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

$$\|R(x, x_{i-1}, u)\| \leq e^{\int_{x_{i-1}}^{x} l(s)\,ds}\|I\| \tag{39}$$

i.e. $\|\varphi_i'(u)\| \leq q_i$, since $\|I\| = 1$. □

Proof of Theorem 6: For $k \in \mathbb{N}$ 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

$$U^{k+1} - U^* = U^k - U^* - L_k(U^k - U^*) + L_k(\Phi(U^k) - \Phi(U^*)). \tag{40}$$

Multiplying by D and taking the norm, we get:

$$\begin{aligned}\|D(U^{k+1} - U^*)\| \leq \|D(I - L_k)D^{-1}\| \cdot \|D(U^k - U^*)\| \\ + \|DL_kD^{-1}\| \cdot \|D(\Phi(U^k) - \Phi(U^*))\|.\end{aligned} \tag{41}$$

We have

$$DL_kD^{-1} = D(I - \Phi')^{-1}D^{-1} = [I - D\Phi'D^{-1}]^{-1}. \tag{42}$$

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

$$[I - T]^{-1} = I + \sum_{i=1}^{N} T^i \tag{43}$$

so that:

$$\|DL_kD^{-1}\| \leq 1 + \sum_{i=1}^{N} \|T\|^i. \tag{44}$$

We can compute T explicitly and obtain

$$T = \begin{pmatrix} 0_m & \cdots & \cdots & \cdots & 0_m \\ \frac{d_1}{d_0}\varphi_1' & 0_m & & & \vdots \\ 0_m & \frac{d_2}{d_1}\varphi_2' & 0_m & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0_m & \cdots & \cdots & \frac{d_N}{d_{N-1}}\varphi_N' & 0_m \end{pmatrix}. \tag{45}$$

According to the previous lemma, $\|\varphi_i'\| \leq q_i$, 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:

$$TT^* = \begin{pmatrix} \left(\frac{d_1}{d_0}\right)^2 \varphi_1'\varphi_1'^* & 0_m & \cdots & \cdots & 0_m \\ 0_m & \left(\frac{d_2}{d_1}\right)^2 \varphi_2'\varphi_2'^* & & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \left(\frac{d_N}{d_{N-1}}\right)^2 \varphi_N'\varphi_N'^* & 0_m \\ 0_m & \cdots & \cdots & 0_m & 0_m \end{pmatrix}. \tag{46}$$

Hence, it follows that:

$$\rho(TT^*) = \max_{i=1,\ldots,N} \rho\left[\left(\frac{\lambda}{q_i}\right)^2 \varphi_i' \varphi_i'^*\right] \leq \lambda^2 \tag{47}$$

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

$$\|DL_k D^{-1}\| \leq \frac{1-\lambda^{N+1}}{1-\lambda}. \tag{48}$$

We obtain similarly:

$$\|I - DL_k D^{-1}\| \leq \lambda \frac{1-\lambda^N}{1-\lambda}. \tag{49}$$

Furthermore, according to Theorem 3 we can write:

$$\|D(\Phi(U^k) - \Phi(U^*))\| \leq \lambda \cdot \|D(U^k - U^*)\|. \tag{50}$$

This finally gives us the estimate:

$$\|D(U^{k+1} - U^*)\| \leq \beta \cdot \|D(U^k - U^*)\| \tag{51}$$

where:

$$\beta = \lambda \frac{1-\lambda^N}{1-\lambda} + \lambda \frac{1-\lambda^{N+1}}{1-\lambda}. \tag{52}$$

Hence the map defined by the iteration is a contraction if $\beta < 1$, which leads to the result. □

Corollary 2. *Let us assume that condition* (31) *is satisfied. Then the conclusion of Theorem* 6 *remains true for* $D = I$, *with the norms* $\|U\| = \|U\|_1$, $\|U\| = \|U\|_2$ *and* $\|U\| = \|U\|_\infty$.

It should be emphasized that some mechanical systems whose energy is scattering are dissipative (see Example 3 below), in addition to others which are not (see Example 2). However, a large class of problems for which condition (31) is satisfied originates from the discretization of diffusion phenomena.

Remark 4. *It should be noted that in the context of the above corollary, lengthening the intervals* $[x_{i-1}, x_i]$, $i = 1, \ldots, N$ *favours both the convergence and the computations/communications ratio. This is the reason why we consider dissipative functions.*

Example 3. *Let us consider a mass* (m) *suspended to a spring* (k) *and hanging in a viscous liquid (leading to a force proportional* (h) *to the speed of the mass). Its movement is governed by the equation:*

$$m \cdot \frac{d^2X}{dt^2} + h \cdot \left|\frac{dX}{dt}\right| \cdot \frac{dX}{dt} + k \cdot X = 0$$

which can be decomposed into a first order system:

$$\begin{cases} X_1' = X_2 \\ X_2' = -\dfrac{h}{m} \cdot X_2 \cdot |X_2| - \dfrac{k}{m} \cdot X_1 . \end{cases}$$

Let $\Lambda = \mathrm{diag}(k, m)$. *The system is dissipative with respect to the norm* $\|V\|^2 = V^T \Lambda V$.

Example 4. *"The Brusselator", which modelizes a multi-molecular chemical reaction, is described in* [11]. *It leads to the system*

$$\begin{cases} \dfrac{\partial u}{\partial x} = A + u^2 v - (B+1)u + \alpha \dfrac{\partial u^2}{\partial w^2} \\ \dfrac{\partial v}{\partial x} = Bu - u^2 v + \alpha \dfrac{\partial v^2}{\partial w^2} \end{cases}$$

with $w \in [0,1]$, $A = 1$, $B = 3$, *and boundary conditions:*

$$u(0,x) = u(1,x) = 1, \qquad v(0,t) = v(1,t) = 3,$$
$$u(w,0) = 1 + \sin(2\pi w), \qquad v(w,0) = 3.$$

By replacing the second spatial derivatives by finite differences on a grid of points we get a dissipative system provided α *is sufficiently large (see next section).*

5. Numerical Results

We now demonstrate the convergence of Algorithm 2 on the following examples. The functions φ_i are not known exactly (neither are their derivative), so that they have to be approximated by using a standard ODE-solver (and their derivative by finite differences). We postpone the analysis of the discrete-time version of Algorithm 2 to Part 2.[1] The aim of this Section is to get first numerical results confirming the relevance of the presented theoretical results. In order to measure the speed of convergence, we have plotted the maximum relative error over all points of the subdivision with respect to the iteration number $\left(\max_{1 \le i \le N} \dfrac{\|y(x_i) - u_i^k\|}{\|y(x_i)\|} \right)$ and for several local tolerances (TOL) given as input to the ODE-solver.[2]

Example 5.

$$y'(x) = \cos(y)\sin(y) - 2y + e^{-x/100}\sin(5x) + \ln(1+x)\cos(x) \qquad y(0) = 1 \quad (53)$$

on $[x_0, X] = [0, 100]$. *The solution is drawn on Fig. 2 for* $x \in [0, 20]$. Condition (31) is satisfied with $\mu\left(\dfrac{\partial f}{\partial y}\right) \le -1$ so that the convergence is very fast (see Fig. 3).

---

[1] For the moment, we admit that the behaviour of Algorithm 2 is well approximated as far as we use sufficiently stringent tolerances.

[2] Any code can be used here, since we are only interested by the convergence of Algorithm 2.

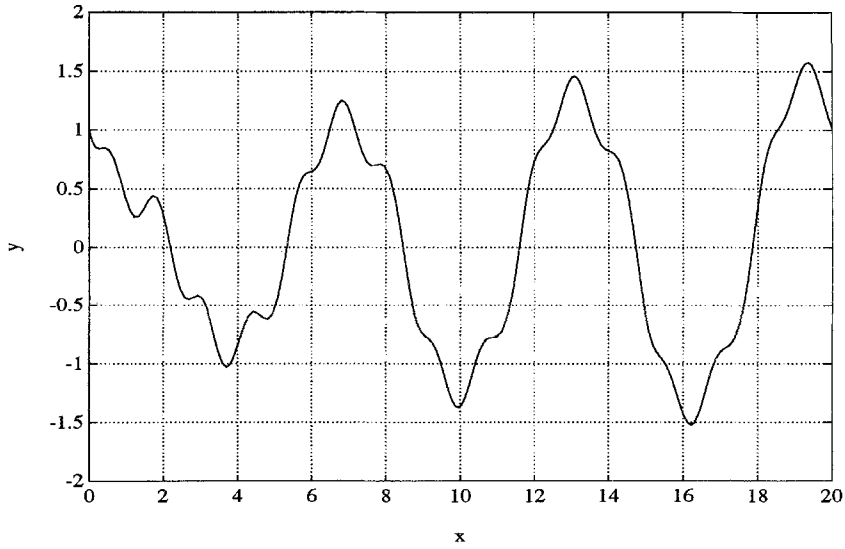

Figure 2. Solution of Example 5

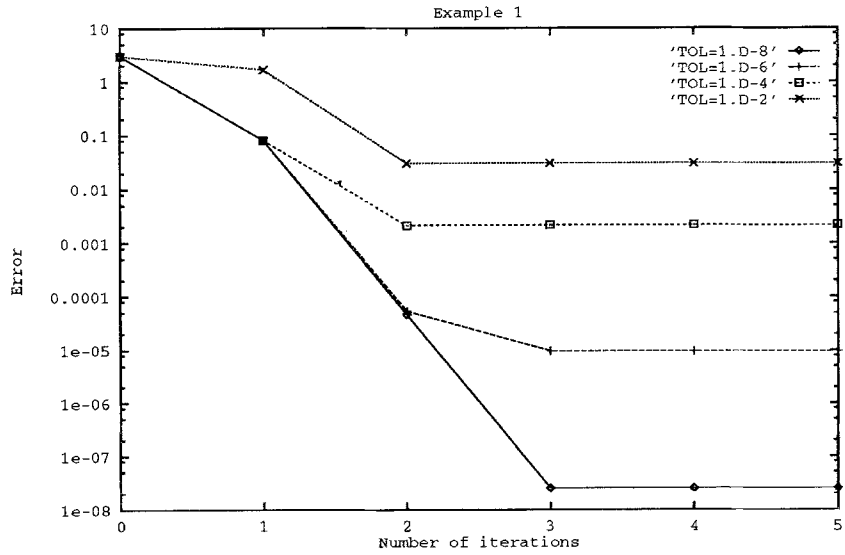

Figure 3. Convergence of Algorithm 2 for Example 5 (64 segments)

Example 6.

$$\begin{cases} y_1'(x) = -y_2 - 0.3y_1^3 + \cos(3x) & y_1(0) = 0 \\ y_2'(x) = y_1 + y_3 + x^{1/5} & y_2(0) = 1 \\ y_3'(x) = -y_2 - 0.01y_3 + \sin(x)\dfrac{\ln(1+x)}{1+x^2} & y_3(0) = 2 \end{cases} \tag{54}$$

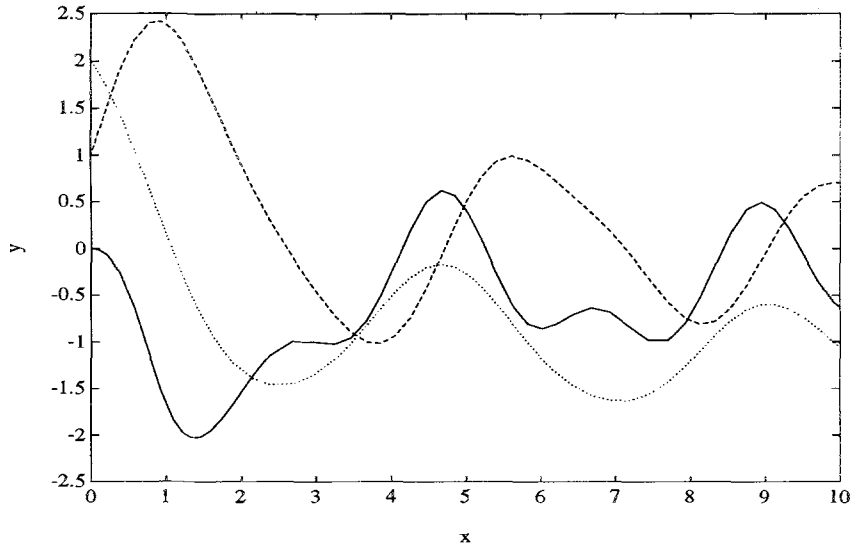

Figure 4. Components of the solution of Example 6

on $[x_0, X] = [0, 100]$. *The components of the solution are drawn on Fig. 4.* In this case, the matrix $1/2\left(\frac{\partial f}{\partial y} + \frac{\partial f^T}{\partial y}\right)$ has three different eigenvalues: $-0.9y_1^2$, 0 and -0.01. Though condition (31) is not "strictly" satisfied, convergence is not affected (see Fig. 5).

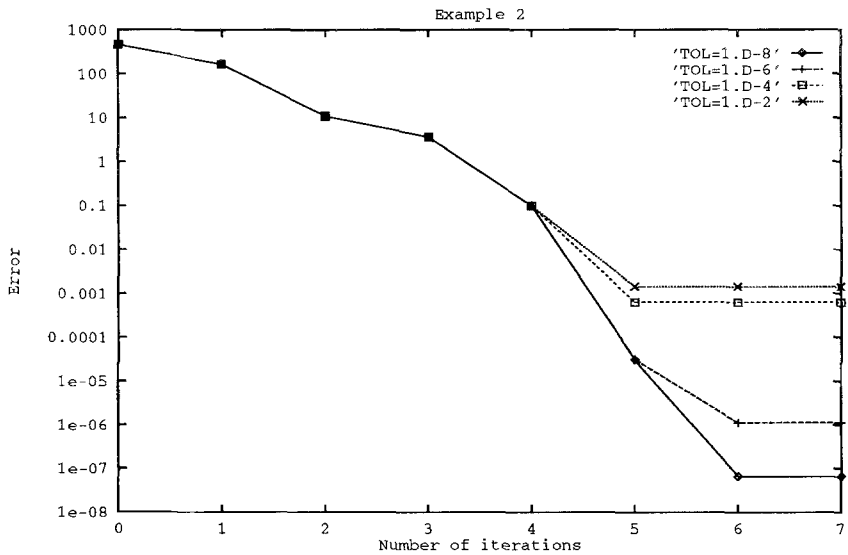

Figure 5. Convergence of Algorithm 2 for Example 6 (32 segments)

Example 7. *We consider once more the Brusselator (see Example 4 of Section 4) in one spatial variable w with* $0 \leq w \leq 1$, $A = 1$, $B = 3$, $\alpha = 1/40$. *The second spatial derivatives* $\frac{\partial^2 u}{\partial w^2}$ *and* $\frac{\partial^2 v}{\partial w^2}$ *are discretized by finite differences on a grid of M points*

$$\forall i \in [1, M], \qquad w_i = \frac{i}{M+1}. \tag{55}$$

We denote $\Delta w = \frac{1}{M+1}$ *and finally obtain the ODE system:*

$$\forall i \in [1, M], \quad \begin{cases} u_i' = 1 + u_i^2 v_i + \dfrac{\alpha}{(\Delta w)^2}(u_{i-1} - 2u_i + u_{i+1}) \\ v_i' = 3u_i - u_i^2 v_i + \dfrac{\alpha}{(\Delta w)^2}(v_{i-1} - 2v_i + v_{i+1}) \end{cases} \tag{56}$$

with

$$u_0(x) = u_{M+1}(x) = 1 \tag{57}$$

$$v_0(x) = v_{M+1}(x) = 3 \tag{58}$$

and the initial conditions

$$\forall i \in [1, M], \quad \begin{cases} u_i(0) = 1 + \sin(2\pi w_i) \\ v_i(0) = 3. \end{cases} \tag{59}$$

The solution is drawn for $M = 40$ *on Fig. 6.* Some easy computations leads to the following expression for $Q = \frac{1}{2}\left(\frac{\partial f}{\partial y} + \frac{\partial f^T}{\partial y}\right)$:

$$Q = \begin{pmatrix} \operatorname{diag}(2u_i v_i - 4) & \frac{1}{2}(\operatorname{diag}(u_i^2) + \operatorname{diag}(3 - 2u_i v_i)) \\ \frac{1}{2}(\operatorname{diag}(u_i^2) + \operatorname{diag}(3 - 2u_i v_i)) & \operatorname{diag}(-u_i^2) \end{pmatrix} + \frac{\alpha}{(\Delta w)^2}\begin{pmatrix} K & 0 \\ 0 & K \end{pmatrix} \tag{60}$$

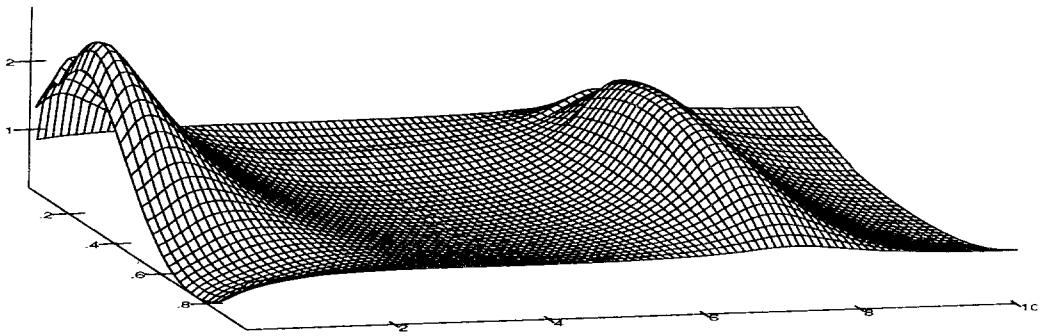

Figure 6. Component $u(w, x)$ of the Brusselator with $M = 40$

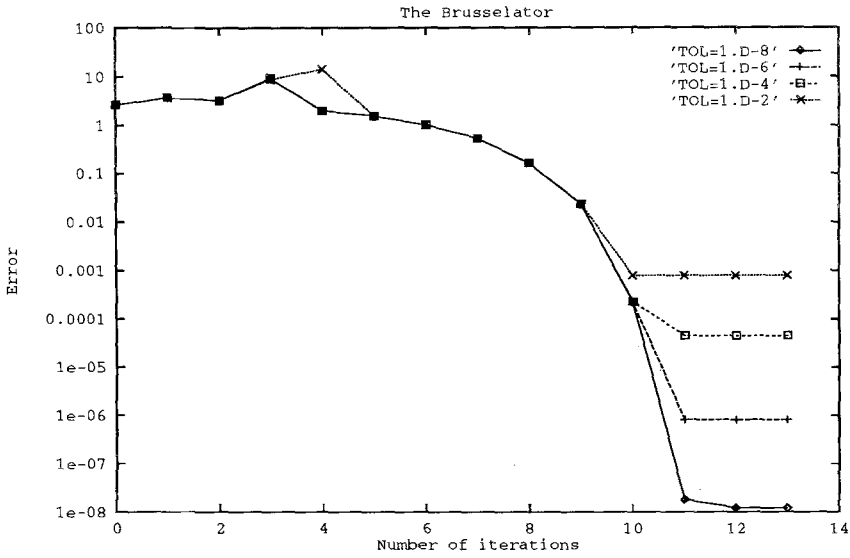

Figure 7. Convergence of Algorithm 2 for the Brusselator with $M = 10$ (32 segments)

where K is the usual matrix

$$\begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \tag{61}$$

whose eigenvalues are known to be

$$\lambda_k = -4\left(\sin\frac{\pi k}{2M+2}\right)^2, \qquad k = 1 \ldots M. \tag{62}$$

Since both matrices of (60) are symmetric, the first matrix can be considered as a small perturbation, provided α is sufficiently large. With this restriction, condition (31) is satisfied.

Part 2: Numerical Implementation

6. Convergence of the Actual Newton's Algorithm

In a real implementation of Algorithm 2, one has to approximate the function Φ and its first derivative by some way or another. We consequently modify Algorithm 2 by replacing φ_i by an approximation $\tilde{\varphi}_i$ computed by an ODE-solver and by replacing φ_i' by a standard finite differences approximation $\tilde{\varphi}_i'$, i.e.

$$\tilde{\varphi}_i'(u) = \left[\frac{1}{\eta}(\tilde{\varphi}_i(u + \eta e_1) - \tilde{\varphi}_i(u)), \ldots, \frac{1}{\eta}(\tilde{\varphi}_i(u + \eta e_m) - \tilde{\varphi}_i(u))\right] \tag{63}$$

where η is a small parameter and $(e_1, \ldots, e_m)$ the canonical basis of $\mathbb{R}^m$. However, the theory developed in Sections 3 and 4 is no longer applicable and a study of the convergence of the resulting algorithm, that we will call Algorithm 2', has to be developed. For this analysis, we will restrict ourselves to the case of dissipative systems, which have been previously shown to be particularly appropriate. Now, if $y(x)$ denotes the exact solution of (1, 2) over $[x_0, X]$, there exists a sufficiently large $\delta > 0$ such that the compact set $K_\delta = \{(x, y) \in [x_0, X] \times \mathbb{R}^m, \|y - y(x)\| \leq \delta\}$ contains the "initial guess" (that can be assumed for example to be the constant solution over $[x_0, X]$). Due to the dissipativity of the problem, $(x_i, \varphi_i(u))$ will lie in K_δ provided (x_{i-1}, u) lies in K_δ. If a p^{th}-order numerical ODE-method is used to approximate the function φ_i, and if h_i is the maximum stepsize considered over $[x_{i-1}, x_i]$, then the following estimate

$$\|\tilde{\varphi}_i(u) - \varphi_i(u)\| \leq C(x_i - x_{i-1})h_i^p \tag{64}$$

holds for all u such that (x_{i-1}, u) belongs to K_δ and for all sufficiently small h_i. C depends a-priori on the p^{th}-order derivatives of the function f, if f is assumed to be smooth enough, but can be bounded for example on $K_{2\delta}$. Now, if h_i is small enough, the numerical solution remains in $K_{2\delta}$ and (64) holds uniformly for all (x_{i-1}, u) in K_δ and all $h_i \leq H$, where H is independent of i. Using this estimate, we can now give:

Lemma 4. *Let us assume that* (1, 2) *satisfies* (31) *and let* $q = \max_{i=1,\ldots,N} \int_{x_{i-1}}^{x_i} l(x)\,dx$ *and* $\Delta x = \max_{i=1,\ldots,N}(x_i - x_{i-1})$. *Suppose in addition that the function* φ_i *are approximated by using a* p^{th}*-order numerical ODE-method. If* $U = (u_0^T, u_1^T, \ldots, u_N^T)^T \in \mathbb{R}^{m\times(N+1)}$ *is such that for all* $i = 0, \ldots, N$, (x_i, u_i) *belongs to* K_δ, *then we have;*

$$\|\tilde{\Phi}(U) - \Phi(U^*)\| \leq C\Delta x h^p + q\|U - U^*\|, \tag{65}$$

where $\tilde{\Phi}$ *is defined by the* $\tilde{\varphi}_i$*'s and where* $h = \max_{i=1,\ldots,N} h_i$.

Lemma 5. *For some* $i \in \{1, \ldots, N\}$, *let* $u \in \mathbb{R}^m$ *be such that* (x_{i-1}, u) *belongs to the interior* $\mathring{K}_\delta$ *of* K_δ. *In addition to previous hypotheses, we assume that the derivatives of the* φ_i*'s are approximated by using formula* (63) *and that* $h_i^p = \mathcal{O}(\eta^2)$. *Then there exists* η_0 *such that*

$$\|\tilde{\varphi}_i'(u) - \varphi_i'(u)\| = \mathcal{O}(\eta) \tag{66}$$

for $\eta \leq \eta_0$.

Proof: Let $(\varphi_i')_{.,j}$ be the j^{th} column-vector of the Jacobian matrix φ_i' and let $(\tilde{\varphi}_i')_{.,j}$ be the j^{th} column-vector of matrix $\tilde{\varphi}_i'$. By considering the 1-norm, it comes:

$$\|(\tilde{\varphi}_i'(u))_{.,j} - (\varphi_i'(u))_{.,j}\|_1 = \left\|\frac{1}{\eta}(\tilde{\varphi}_i(u + \eta e_j) - \tilde{\varphi}_i(u)) - (\varphi_i'(u))_{.,j}\right\|_1 \tag{67}$$

$$\leq \frac{2C\Delta x h_i^p}{\eta} + \left\|\frac{1}{\eta}(\varphi_i(u + \eta e_j) - \varphi_i(u)) - (\varphi_i'(u))_{.,j}\right\|_1. \tag{68}$$

Now, if f is smooth enough, the functions φ_i are smooth also, and we can bound the second term of (68) uniformly on K_δ and for η sufficiently small by a constant

times η. From $h_i^p = \mathcal{O}(\eta^2)$, we consequently get:

$$\|(\tilde{\varphi}_i'(u))_{.,j} - (\varphi_i'(u))_{.,j}\|_1 = \mathcal{O}(\eta) \tag{69}$$

and the result follows in an obvious way. □

Remark 5. *It is worth emphasizing the necessity of the condition $h_i^p = \mathcal{O}(\eta^2)$ for the finite differences approximation* (63) *to be accurate.*

Theorem 7. *Suppose that* $(1,2)$ *satisfies* (31) *with either* $q < 1/3$ *or* $(N+1) < \dfrac{\ln(3q-1) - \ln(1+q)}{\ln(q)}$ *and that the functions φ_i are approximated by using a p^{th}-order ODE-solver and their first derivative φ_i' by using formula* (63). *If the maximum stepsize h used by the ODE-solver is such that $h^p = \mathcal{O}(\eta^2)$, then for all sufficiently small values of η, there exist two constants $0 < \tilde{\beta} < 1$ and $\Gamma > 0$ such that*

$$\forall k \geq 1, \qquad \|U^k - U^*\| \leq \tilde{\beta}^k \|U^0 - U^*\| + \Gamma h^p \tag{70}$$

Proof: From Lemmas 3 and 5, we can assert the existence of two positive constants η_0 and M, such that:

$$\begin{gathered} \forall i \in \{1,\ldots,N\}, \qquad \forall \eta \leq \eta_0, \qquad \forall u \in \mathbb{R}^m, \\ ((x_{i-1},u) \in \mathring{K}_\delta) \Rightarrow (\|\tilde{\varphi}_i'\| \leq q + M\eta). \end{gathered} \tag{71}$$

We now proceed as in the proof of Theorem 6:

$$\forall k \in \mathbb{N}, \qquad U^{k+1} - U^* = (I - \tilde{L}_k)(U^k - U^*) + \tilde{L}_k(\tilde{\Phi}(U^k) - \Phi(U^*)) \tag{72}$$

where $\tilde{L}_k = (I - \tilde{\Phi}'(U^k))^{-1}$ and with obvious notations for $\tilde{\Phi}'(U^k)$. Using the nilpotency of $\tilde{\Phi}'(U^k)$, we get the estimates:

$$\|\tilde{L}_k\| \leq \frac{1-\tilde{\lambda}^{N+1}}{1-\tilde{\lambda}}, \qquad \|I - \tilde{L}_k\| < \tilde{\lambda}\frac{1-\tilde{\lambda}^N}{1-\tilde{\lambda}}, \tag{73}$$

where $\tilde{\lambda} = q + M\eta < 1$ for all sufficiently small η. Taking into account the estimate (65) of Lemma 4 leads to

$$\|U^{k+1} - U^*\| \leq \tilde{\beta}\|U^k - U^*\| + \tilde{\gamma} C \Delta x h^p, \tag{74}$$

where $\tilde{\beta} = q\dfrac{1-\tilde{\lambda}^{N+1}}{1-\tilde{\lambda}} + \tilde{\lambda}\dfrac{1-\tilde{\lambda}^N}{1-\tilde{\lambda}}$ and $\tilde{\gamma} = \dfrac{1-\tilde{\lambda}^{N+1}}{1-\tilde{\lambda}}$. Since $\tilde{\beta}$ tends to β as η tends to zero, it follows from the hypotheses that $\tilde{\beta} < 1$ for all sufficiently small values of η. This implies in particular that the successive iterates U^k remain in K_δ provided $h \leq \left(\dfrac{\delta(1-\tilde{\beta})}{\tilde{\gamma} C \Delta x}\right)^{1/p}$. Finally, a straightforward recursion gives:

$$\forall k \geq 1, \qquad \|U^k - U^*\| \leq \tilde{\beta}^k \|U^0 - U^*\| + \sum_{i=0}^{k-1} \tilde{\beta}^i \tilde{\gamma} C \Delta x h^p \tag{75}$$

$$\leq \tilde{\beta}^k \|U^0 - U^*\| + \frac{\tilde{\gamma} C \Delta x h^p}{1-\tilde{\beta}} \tag{76}$$

□

Remark 6. *Theorem 7 does not ensure the convergence of Algorithm 2': it asserts the existence of a h-neighbourhood of the exact solution in which U^k will enter and remain. The convergence of Algorithm 2' is however ensured within N iterations owing to the additional accurate component of U automatically gained at each iteration.*

Remark 7. *Due to the use of finite differences approximations for Φ', the theoretical quadratic convergence of Algorithm 2 is lost. This was the main motivation in* [2] *for considering Steffensen's method. However, Figs. 3, 5 and 7 from Section 5 of Part 1 show that, from a practical point of view, the error decreases superlinearly as soon as the computed solution is sufficiently close to the exact solution.*

7. 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 accelerations 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 7 ensures reasonable behaviour of U given reasonable conditions on $l(x)$. For the accepted values, we adopt the same strategy as in [2] based on the following theorem:

Theorem 8. *Let us assume that* (1) *satisfies* (31). *If the iteration error is defined to be: $\tilde{\tau}^k = \tilde{\Phi}(U^k) - U^k$, then, we have the following bound on $E^k = U^* - U^k$:*

$$\|E^k\| \leq \frac{1}{1-q}(\|\tilde{\tau}^k\| + C\Delta x h^p) \tag{77}$$

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

Proof: The proof is obvious and therefore omitted. □

Hence, in order 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 $\max_{j \leq n} \|\tilde{\tau}_j\| \leq \varepsilon$, where ε is to be defined by the user and represents the tolerance on the iteration error. Finally, we sketch the algorithm below and denote **dopar** the parallel loops and **doseq** the sequential ones:

Algorithm 2".
*** *Initialization phase* ***
set n = 0, set k = 0
dopar $i = n + 1, \ldots, N$,
  set $u_i^0 = u_0^0$
end
*** *Integration phase: the $\tilde{\varphi}_i$'s are computed by an ODE-solver* ***

dopar $i = n + 1, \ldots, N$,
  compute $v_i^{k+1} = \tilde{\varphi}_i(u_{i-1}^k)$
  dopar $j = 1, \ldots, m$,
    *** e_j *is the* j^{th} *vector of the canonical basis of* $\mathbb{R}^{m+1}$, *and* $\eta = 10^{-7}$ ***
    compute $w_{i,j}^{k+1} = \tilde{\varphi}_i(u_{i-1}^k + \eta e_j)$
  end
end
*** *Compute the error of the iterative process* ***
dopar $i = n + 1, \ldots, N$,
  compute $\tilde{\tau}_i^{k+1} = v_i^{k+1} - u_i^k$
  compute $\|\tilde{\tau}_i^{k+1}\|$
end
*** *Assemble the Jacobian matrices (computed by finite differences) of the* $\tilde{\varphi}_i$*'s* ***
dopar $i = n + 1, \ldots, N$,
  dopar $j = 1, \ldots, m$,
    assemble $C_{i,j}^{k+1} = \dfrac{w_{i,j}^{k+1} - v_i^{k+1}}{\eta}$
  end
end
dopar $i = n + 1, \ldots, N$,
  compute $A_i^{k+1} = [C_{i,1}^{k+1}, \ldots, C_{i,m}^{k+1}]$
end
*** *Recurrence phase* ***
set $u_n^{k+1} = u_n^k$
set $\Theta = 0$
set $p = n$
doseq $i = n + 1, \ldots, N$,
  compute $\Theta = \max(\Theta, \tilde{\tau}_i^{k+1})$
  if $\Theta < \varepsilon$ **then** set $p = i$
  compute $u_i^{k+1} = v_i^{k+1} + A_i^{k+1}(u_{i-1}^{k+1} - u_{i-1}^k)$
end
if $p = N$ **then**
  STOP
else
  set $n = p$
endif

set $k = k + 1$
goto *Integration phase*

Remark 8. *In our experiments, we have used the code DOPRI8 of E. Hairer and al. (see* [8]*), which is based on the explicit* 8^{th}*-order Runge-Kutta method of Prince and Dormand. Since our test problems are only middly stiff and since our main concern here was the results obtained for stringent tolerances (they indeed require the largest amount of computations), an explicit code is still appropriate. In other situations, codes based on implicit methods (such as RADAU5 of E. Hairer and al. (see* [9]*) or LSODE of A. Hindmarsh (see* [10]*)) are strongly recommended.*

Table 1. One iteration of Algorithm 2″ on "hypercube number i"

Step	Computations/Communications
1	**Broadcast** u_{i-1}^k from $P_{i,0}$ to $P_{i,1}, \ldots, P_{i,m}$.
2	**Compute** v_i^{k+1} on $P_{i,0}$ and $w_{i,1}^{k+1}, \ldots, w_{i,m}^{k+1}$ on $P_{i,1}, \ldots, P_{i,m}$.
3	**Compute** $\tilde{\tau}_i^{k+1}$ and $\|\tilde{\tau}_i^{k+1}\|$ on $P_{i,0}$.
4	**Broadcast** v_i^{k+1} from $P_{i,0}$ to $P_{i,1}, \ldots, P_{i,m}$.
5	**Compute** $C_{i,1}^{k+1}, \ldots, C_{i,m}^{k+1}$ (columns of Δ_i^{k+1}) on $P_{i,1}, \ldots, P_{i,m}$.
6	**Global send** of $C_{i,1}^{k+1}, \ldots, C_{i,m}^{k+1}$ from $P_{i,1}, \ldots, P_{i,m}$ to $P_{i,0}$.
7	**Assemble** $\Delta_i^{k+1} = [C_{i,1}^{k+1}, \ldots, C_{i,m}^{k+1}]$ on $P_{i,0}$
8	**Receive** u_{i-1}^{k+1} and Θ from $P_{i-1,0}$. **Compute** u_i^{k+1} and Θ on $P_{i,0}$. **Send** u_i^{k+1} and Θ to $P_{i+1,0}$.

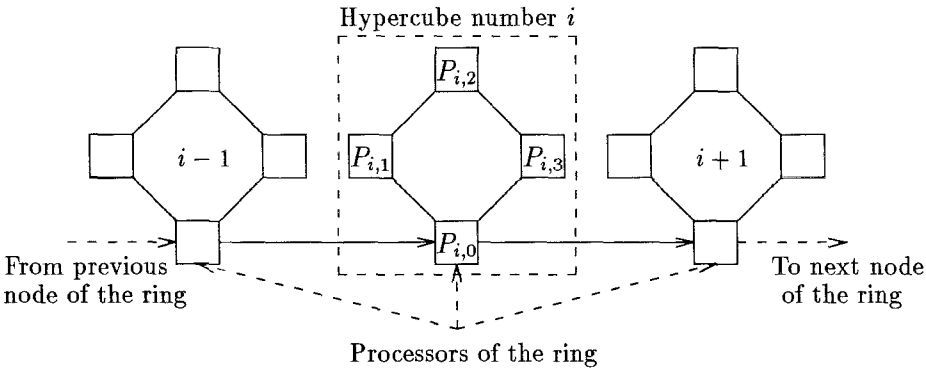

Figure 8. Model of architecture for $m = 3$

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 N clusters of $(m + 1)$ processors (see Fig. 8), where the $(m + 1)$ processors of a cluster make up a small hypercube. In fact, each hypercube has to be of dimension $\overline{\log_2(m+1)}$ where $\bar{x}$ denotes the ceiling function. This architectural model is relevant since it can be easily mapped onto a hypercube in a dynamical fashion: provided there are enough processors, the role of each node can be determined once N and m are known. We do not claim this model is optimal. However, it has the advantage to keep the communication cost at a reasonable level.

Now, 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.
- 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 $[x_{i-1}, x_i]$, $i = 1, \ldots, N$ is constant (this hypothesis will be discussed in Section 8.1), we can compute the times spent for each step:

1. $\Delta t = (\beta + m\tau_c)\log_2(m+1)$
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. $\Delta t = 0$
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:

$$T_p = k^*[(C/N)\tau_{op} + (4m-1)\tau_{op} + m(m+1)\tau_c + 2\log_2(m+1)\tau_c + 3\log_2(m+1)\beta] \tag{78}$$

$$+ \left(\sum_{k=1}^{k^*} p(k)\right)[m(2m+1)\tau_{op}] + \left(\sum_{k=1}^{k^*} p(k) - k^*\right)[\beta + (m+1)\tau_c] \tag{79}$$

The sequential time is,

$$T_s = C\tau_{op} \tag{80}$$

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 $[x_{i-1}, x_i]$, $i = 1, \ldots, N$. 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 a lower bound of the speed-up factor. We have, in this case,

$$\sum_{k=1}^{k^*} p(k) = k^*N. \tag{81}$$

- (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 estimates of $p(k)$: $p(1) = N$ and $\forall k \in [2, k^*]$, $p(k) = k^* + 1 - k$. Thus, in that case we have:

$$\sum_{k=1}^{k^*} p(k) = N + \frac{k^*(k^*-1)}{2}. \tag{82}$$

- (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 $p(k) = N - (k-1)\dfrac{N}{k^*}$, so that:

$$\sum_{k=1}^{k^*} p(k) = \frac{(k^*+1)}{2} N. \tag{83}$$

Finally, we get the speed-up factor:

$$s = \frac{T_s}{T_p} = \frac{CN}{C_0 + C_1 N + C_2 N^2}, \tag{84}$$

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

$$s_P \leq s_M \leq s_O \leq \frac{N}{k^*}. \tag{85}$$

Table 2. Values of the constants for the different hypotheses

Hypotheses	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	$k^*C\tau_{op}$	$k^*(4m-1)\tau_{op}$ $+k^*[2m\ln_2(m+1)+(m+1)(m-1)]\tau_c$ $+k^*[3\ln_2(m+1)-1]\beta$	$m(m+1/2)(k^*+1)\tau_{op}$ $+[1/2(k^*+1)(m+1)]\tau_c$ $+1/2(k^*+1)\beta$

Remark 9. *The hypothesis (P) allows us to derive a lower bound of the optimal speed-up with respect to N that is proportional to the number of iterations k^*, as well as an estimate of the optimal number of intervals $N_{opt} = \sqrt{C_0/C_2}$.*

8. Results

8.1. Adequacy of the Model

In order to evaluate the correctness of our performance model we performed a simulation of Algorithm 2′ on the Intel IPSC-860 hypercube of the ONERA with 128 processors. Based on Benchmarks on the IPSC-860, we took $\beta/\tau_{op} = 693$ and

Table 3. Values of C for different tolerances

Example	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-8}$
5	$2.3\ 10^5$	$3.5\ 10^5$	$6.8\ 10^5$
6	$4.9\ 10^5$	$4.9\ 10^5$	$6.4\ 10^5$

$\tau_c/\tau_{op} = 5.5$. The remaining parameters involved in the determination of the speed-up by formulas (82), (83) and (84) were determined as follows:

- the cost of integration (C) was computed via the output "NFCN" (number of right-hand side evaluations) of the ODE-solver DOPRI8 from [8].
- the number of iterations (k^*) was determined by simulations of Algorithm 2′ on a sequential machine (see Section 5).

The informations collected were then used to compute the speed-up in two different ways. On the one hand, we applied the formulas given above for the three different hypotheses ("P", "M" and "O"). On the other hand, we simply measured the execution time of Algorithm 2′ on $N \times (m + 1)$ processors and the execution time of "DOPRI8" on one processor.

Results are listed in Table 4. Let us first notice that estimates for Example 6 are the same for the first two tolerances. This is due to the unstable behaviour of the code DOPRI8. For low tolerances, the computational cost is indeed almost constant. Secondly, our hypotheses seem to lead to an over-estimation of the speed-up for Example 6. In fact, the difference between the lowest estimate and the observed speed-up decreases for small tolerances. This behaviour can be easily explained by the presence of an important transient phase in the solution of Example 6. This transient phase partly modifies the load-balancing so that the computational work is no longer strictly independent of the segment under consideration. For small tolerances however, this phenomenon vanishes. Now, as far as the transient phase is not too important (see Example 5), our estimates are in good agreement with the observed speed-up the our model is relevant. Those remarks show that Algorithm

Table 4. Speed-up's for Examples 5 and 6

Example	Speed-up	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-8}$
5	Real Speed-up	3.0	5.0	8.0
	Estimate P	2.3	3.4	6.0
	Estimate M	3.0	4.3	7.5
	Estimate O	4.2	6.0	9.9
6	Real Speed-up	1.4	1.6	1.9
	Estimate P	1.7	1.7	2.0
	Estimate M	2.3	2.3	2.6
	Estimate O	3.0	3.0	3.3

2′ should be applied to specific problems. Generally speaking, the problem to be solved should possess the following characteristics:

- dissipativity of the right-hand side (see Section 4)
- long interval of integration (see Remark 4)
- high computational cost per step (this is a general requirement for parallel methods).

8.2. *Attainable Speed-up*

When applied to a m-dimensional system with N segments, Algorithm 2′ (implemented according to our description) requires $N \times (m+1)$ processors. Since the speed-up is limited by N/k^*, the efficiency is severely bounded by $\dfrac{1}{k^* \times (m+1)}$. This bound emphasizes the redundancy of computations involved in Algorithm 2′, whose necessity comes from the sequential nature of the numerical integration process. However Algorithm 2′ can provide a large speed-up when a large number of processors is available. Extrapolation of the speed-up curves (see Fig. 9) by mean of formula (83) indeed shows that an acceleration of 15 is attainable for Example 5 with $TOL = 10^{-10}$. The same curve for Example 6 seems less convincing. Nevertheless, one should keep in mind the fundamental influence of the ratios β/τ_{op} and τ_c/τ_{op} on the efficiency of the algorithm. Much better results would have been obtained for instance by considering the values observed on the Intel IPSC-2 machine ($\beta/\tau_{op} = 69.3$ and $\tau_c/\tau_{op} = 0.3$): the 'M'-estimate for Example 6 with $TOL = 10^{-10}$ gives a speed-up of 11 for 400 processors. Finally, let us notice that whatever the values of these ratios are, a high computational cost per step or a long interval of integration

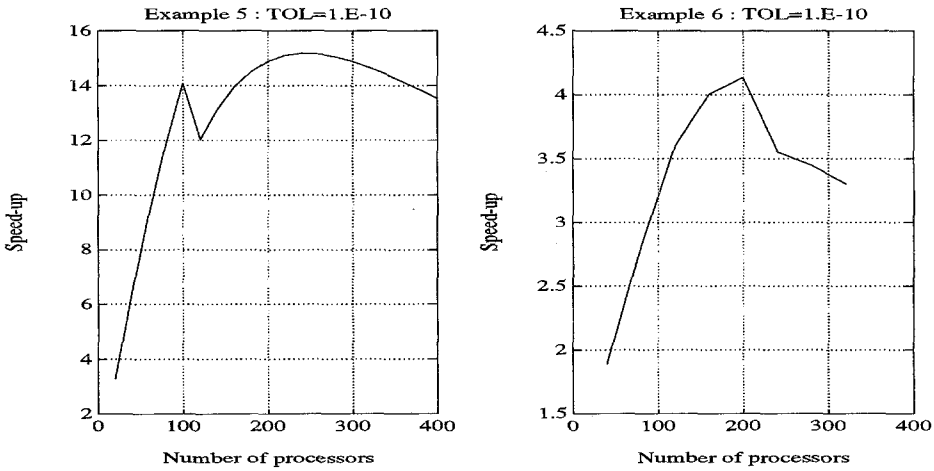

Figure 9. Speed-up estimates for Examples 5 (Number of processors = $2 \times N$) and 6 (Number of processors = $4 \times N$)

will lead to a good *computation cost/communication cost* ratio and consequently to a high acceleration on a massively parallel computer.

9. Conclusion

A parallel algorithm based on an idea of A. Bellen and M. Zennaro 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. In addition to this, real experiments were reported that confirm the interest of the method for specific problems.

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 problems so as to analyse the behaviour of the algorithm in cases where the complexity varies from one subdivision to another.

Acknowledgements

We would like to thank Professor M. Crouzeix for his advices. We are also grateful to P. Leca for allowing us access to the Intel IPSC-860 at ONERA.

References

[1] Bellen, A., Vermiglio, R., Zennaro, M.: Parallel ODE-solvers with step-size control. J. Comp. Appl. Math. *31*, 277–293 (1990).
[2] Bellen, A., Zennaro, M.: Parallel algorithms for initial value problems. J. Comp. Appl. Math. *25*, 341–350 (1989).
[3] Birta, L., Abou-Rabia, O.: Parallel block predictor-corrector methods for ODE's. IEEE Transactions on Computers *C-36*, 299–311 (1987).
[4] Chartier, P.: Application of Bellen's method to ODE's with dissipative right-hand side. Research Report 593, IRISA, Campus de Beaulieu, Rennes, France, 1991.
[5] Chartier, P.: L-stable parallel one-block methods for ordinary differential equations. SIAM J. Numer. Anal. (1993) (to appear).
[6] Franklin, M.: Parallel solution of ordinary differential equations. IEEE Transactions on Computers *C-27*, 413–420 (1978).
[7] Gear, C.: Parallel methods for ordinary differential equations. Research R-87-1369, University of Illinois, Urbana, IL, 1986.
[8] Hairer, E., Norsett, S., Wanner, G.: Solving ordinary differential equations I. Nonstiff problems, vol. 1. Berlin, Heidelberg: Springer 1987.
[9] Hairer, E., Wanner, G.: Solving ordinary differential equations II. Stiff and differential-algebraic problems, vol. 2. Berlin, Heidelberg, New York, Tokyo: Springer 1991.
[10] Hindmarsh, A.: LSODE and LSODI, two new initial value ordinary equation solvers. ACM/SIGNUM Newsletter *15*, 10–11 (1980).

[11] Lefever, R., Nicolis, G.: Chemical instabilities and sustained oscillations. J. Theor. Biol. *30* 267–284 (1971).
[12] Ortega, J., Rheinbolt, W.: Iterative solution of nonlinear equations in several variables. New York, San Francisco, London: Academic Press, 1970.
[13] Prothero, A., Robinson, A.: On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. Math. Comput. *28*, 145–162 (1974).
[14] Shampine, L., Watts, H.: A-stable implicit one-step methods. BIT *12*, 252–266 (1972).
[15] Sommeijer, B., Couzy, W., Houwen, P. van der: A-stable parallel block methods for ordinary and integro-differential equations. PhD thesis, Universiteit van Amsterdam, CWI, Amsterdam, 1992.
[16] Houwen, P. van der; Sommeijer, B.: Iterated Runge-Kutta methods on parallel computers. SIAM J. Sci. Statist. Comput. *12*, 1000–1028 (1991).
[17] Vermiglio, R.: Parallel step methods for difference and differential equations. Tech. Rep., C.N.R. Progetto Finaizzato "Sistemi Informatici e Calcolo Parallelo", 1989.

P. Chartier
SIMULOG
1 rue James Joule
F-78182 St Quentin Yvelines Cedex
France

B. Philippe
IRISA/INRIA
Campus de Beaulieu
F-35042 Rennes Cedex
France