On the Extension of the Code GAM for Parallel Computing*

Felice Iavernaro and Francesca Mazzia

Dipartimento di Matematica, Università di Bari.
Via Orabona, 4. I-70125 Bari (Italy)
`mazzia@dm.uniba.it`

Abstract. The code GAM numerically solves initial value ordinary differential equations by means of a family of variable-step variable-order block Boundary Value Methods. Here we consider the possibility of performing the code on parallel machines. Some numerical tests and comparisons are presented.

Key words. linear multistep formulas, Runge-Kutta methods, stiff initial value problems
AMS(MOS) subject classifications. 65L05, 65L20

1 Introduction

The code GAM [8] implements the Generalized Adams Methods (GAMs) of orders 3,5,7 and 9 [3, 6, 7] to solve initial value problems of the form

$$\begin{cases} \mathbf{y}'(t) &= \mathbf{f}(t,\mathbf{y}), \qquad \mathbf{f} : [a, a+T] \times H \to H, \\ \mathbf{y}(a) &= \mathbf{y}_0, \end{cases} \tag{1}$$

where $H = \mathbb{R}^r$. Its effectiveness has been established on the basis of its good behaviour on a wide variety of test problems as compared to the performance of some well-known codes such as RADAU5 and MBDFDAE (see [7]). At the moment there exist two different versions of the code, both conceived to run on sequential machines. In this context our interest is in deriving some techniques that may allow an efficient implementation of the code on a parallel computer. It is our experience that most of the time (about 85 %) spent to advance the solution is devoted to solving the nonlinear systems underlying the integration procedure: the whole performance of the code will substantially benefit from an efficient parallelization of this part and our efforts will be concentrated to face this problem. In particular the result presented here will concern one of the above mentioned versions of the code as specified in the next section. The organization of the paper is as follows. In the next section the GAMs are briefly introduced

* Work supported by MURST (40% project).

together with an outline on how these implicit formulae have been solved, with a closer inspection of those steps that will be subject to parallelization, as described in section 3. Here a possible approach is reported and analyzed for achieving parallelization at different levels: this matter forms the background for a future gradual development of the parallel code. In section 4 we present some numerical results related to the very first experiments concerning the parallel version of GAM. In the sequel, when needed, an $N_1 \times N_2$ matrix T, will be viewed as a linear operator $H^{N_2} \longrightarrow H^{N_1}$, that is for example, given a block vector $Y \in H^{N_2}$, AY will stand for $(A \otimes I_r)Y \in H^{N_1}$ with I_r the r-dimensional identity matrix.

2 Inside the Code GAM

Hereafter we give a short account of those sections of the code that, via a suitable modification, will be performed in parallel by a number of processes. Details about definitions, properties, implementation techniques and the overall functionality of the code may be found in [3, 6, 7]. During the integration process the continuous problem (1) is solved over adjacent time intervals

$$W_s = [t^{(s-1)}, t^{(s)}], \qquad s = 1, \ldots, M,\ t^{(0)} = a,\ t^{(M)} = a + T.$$

In each W_s the solution of (1), say $\hat{\mathbf{y}}(t)$, is approximated by a vector $\widetilde{Y}^{(s)} = [\mathbf{y}_0^{(s)}, \ldots, \mathbf{y}_{N_s}^{(s)}]^T \in H^{N_s+1}$, that is $\mathbf{y}_i^{(s)} \simeq \hat{\mathbf{y}}(t_i^{(s)})$, where $\{t_i^{(s)}\}_{i=0}^{N_s}$ is a uniform mesh over W_s such that $t_0^{(s)} = t^{(s-1)}$ and $t_{N_s}^{(s)} = t^{(s)}$; the positive number $h_s = t_i^{(s)} - t_{i-1}^{(s)}$ is the stepsize of integration. It must be observed that since $W_{s-1} \cap W_s = \{t^{(s-1)}\}$, we also have $\mathbf{y}_0^{(s)} = \mathbf{y}_{N_{s-1}}^{(s-1)}$ that is the first component of the block vector $\widetilde{Y}^{(s)}$ should not be treated as an unknown because it takes information from the preceding step. The other N_s components of $\widetilde{Y}^{(s)}$ are the solution of the following algebraic system of dimension N_s:

$$\widetilde{A}_s \widetilde{Y}^{(s)} - h_s \widetilde{B}_s \widetilde{F}(\widetilde{Y}^{(s)}) = \mathbf{0}, \qquad s = 1, \ldots, M, \tag{2}$$

where $\widetilde{A}_s = \{\alpha_{ij}\}$ and $\widetilde{B}_s = \{\beta_{ij}\}$, $i = 1, \ldots, N_s$, $j = 0, \ldots, N_s$, are $(N_s+1) \times N_s$ real matrices and $\widetilde{F}(\widetilde{Y}^{(s)}) = [\mathbf{f}(t_0^{(s)}, \mathbf{y}_0^{(s)}), \ldots, \mathbf{f}(t_{N_s}^{(s)}, \mathbf{y}_{N_s}^{(s)})]^T$. Formula (2) may be viewed as a set of linear combinations of $\mathbf{y}_i^{(s)}$ and $\mathbf{f}(t_i^{(s)}, \mathbf{y}_i^{(s)})$; it defines a block-GAM of odd order p and dimension N_s if the following conditions are fulfilled (to simplify the notation we omit in the sequel the superscripts (s)):

(i) each component of (2) assumes the form

$$\mathbf{y}_i - \mathbf{y}_{i-1} = h \sum_{j=-k_1^{(i)}}^{k_2^{(i)}} \beta_{ij} \mathbf{f}_{i+j}, \qquad i = 1, \ldots, N_s, \tag{3}$$

with $k_1^{(i)}$ and $k_2^{(i)}$ nonnegative integers such that $k_1^{(i)} + k_2^{(i)} = p - 1$ and

$$k_1^{(i)} = \begin{cases} i & \text{for } i = 1, \dots, (p-3)/2, \\ (p-1)/2 & \text{for } i = (p-1)/2, \dots, N - (p-1)/2 \ , \\ i - N + p - 1 & \text{for } i = N - (p-3)/2, \dots, N; \end{cases}$$

(ii) assuming that $\mathbf{y}_0 = \hat{\mathbf{y}}(t_0)$, then for each $i = 1, \dots, N$, $\mathbf{y}_i = \hat{\mathbf{y}}(t_i) + O(h^{p+1})$ that is the coefficients β_{ij} are (uniquely) determined in order to provide a $(p+1) - st$ order approximation to the true solution at each time t_i.

From the $(p-1)$-step linear multistep formulae (3) it is deduced that $\widetilde{A}$ is bidiagonal and Toeplitz with $\alpha_{ii} = 1$ and $\alpha_{i+1,i} = -1$ as diagonal and lower diagonal entries. We remark that the dimensions N_s of each system (2) could be in principle arbitrarily large so as to cover (under the same h_s) wider or smaller intervals W_s. All the same, inside the code, once the order has been selected, the dimension of the corresponding formula remains fixed and is equal to 4, 6, 8, 10 for the orders 3, 5, 7 and 9 respectively. These dimensions allow the estimation of the error and the order changing routines to operate (for an explicit list of the coefficients β_{ij} see [6]).

Performing the partitions

$$\begin{aligned} &\widetilde{Y} = [\mathbf{y}_0, Y^T]^T, &\qquad& \widetilde{F}(\widetilde{Y}) = [\mathbf{f}(t_0, \mathbf{y}_0), F(Y)^T]^T, \\ &\widetilde{A} = [\mathbf{a}_0, A], && \widetilde{B} = [\mathbf{b}_0, B], \end{aligned}$$

with $Y = [\mathbf{y}_1, \dots, \mathbf{y}_N]^T$, $F(Y) = [\mathbf{f}(t_1, \mathbf{y}_1), \dots, \mathbf{f}(t_N, \mathbf{y}_N)]^T$ and $\mathbf{a}_0$, $\mathbf{b}_0$ the first column of $\widetilde{A}$ and $\widetilde{B}$ respectively, we can move in (2) all the known terms to the right hand side, thus obtaining

$$AY - hBF(Y) = \mathbf{b}, \tag{4}$$

with $\mathbf{b} = -\mathbf{a}_0\mathbf{y}_0 + h\mathbf{b}_0\mathbf{f}(t_0, \mathbf{y}_0)$ ($\mathbf{a}_0$ and $\mathbf{b}_0$ are used as linear operators $H \to H^N$). We now consider one of the two approaches adopted to solve equation (4).

3 Simplified Newton Iteration

The system (4) may be recast as $Y - hCF(Y) = \boldsymbol{\delta}$, with $C = A^{-1}B$ and $\boldsymbol{\delta} = A^{-1}\mathbf{b}$, thus obtaining an expression analogous to that used to derive the internal stages of a Runge-Kutta formula. It follows that methods for handling the nonlinear systems arising from the application of a R-K method may be as well applied in this context. Indeed we followed the approach used in RADAU5 (see [4] pages 118-122), although we preferred to maintain the expression (4) because it allows an easy estimation of the error. The modified Newton method is used to linearize (4) whose solution is consequently obtained as the limit of the sequence $\{Y^k\}$ defined as

$$\begin{cases} Y^0 \quad \text{given}, \\ (A \otimes I_r - h(B \otimes I_r)D_J)\, \varDelta Y^k = \mathbf{b} - AY^k + hBF(Y^k), \\ Y^{k+1} = Y^k + \varDelta Y^k, \end{cases} \tag{5}$$

where $D_J = diag[\frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_1, \mathbf{y}_1^0), \dots, \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_N, \mathbf{y}_N^0)]$. The starting value Y^0 is obtained by extrapolation considering the solution computed at the previous step (see [7]). To avoid more than one Jacobian evaluation per step we make the approximation

$$\frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_i, \mathbf{y}_i^0) \simeq J \equiv \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_0, \mathbf{y}_0);$$

consequently the linear systems to be solved become

$$(A \otimes I_r - hB \otimes J)\, \Delta Y^k = G(Y^k), \tag{6}$$

with $G(Y^k) = \mathbf{b} - AY^k + hBF(Y^k)$. It is possible to further reduce the computational cost recasting (6) into block-diagonal form. This is done by first considering the block diagonal form of $A^{-1}B$ (its existence and well conditioning has been verified for all the considered GAMs)

$$T^{-1}A^{-1}BT = \Lambda, \qquad \Lambda = \begin{pmatrix} \alpha_1 & -\beta_1 & & & & \\ \beta_1 & \alpha_1 & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & \alpha_{N/2} & -\beta_{N/2} \\ & & & & \beta_{N/2} & \alpha_{N/2} \end{pmatrix}$$

and then performing the transformations of variables $Z^k = T^{-1}Y^k$.
Setting $S = AT$, multiplying both sides of (6) by S^{-1} and exploiting the relation $(\Lambda T^{-1}) \otimes J = (\Lambda \otimes J)(T^{-1} \otimes I_r)$, the iteration scheme becomes:

$$(I_N \otimes I_r - h\Lambda \otimes J)\, \Delta Z^k = S^{-1}G(TZ^k), \tag{7}$$

which consists of $N/2$ decoupled systems having the form

$$\begin{pmatrix} I_r - h\alpha_n J & -\beta_n J \\ \beta_n J & I_r - h\alpha_n J \end{pmatrix} \begin{pmatrix} \mathbf{u}_n \\ \mathbf{v}_n \end{pmatrix} = \begin{pmatrix} \mathbf{c}_n \\ \mathbf{d}_n \end{pmatrix}, \qquad n = 1, \dots, N/2, \tag{8}$$

where $\mathbf{u}_n = \mathbf{z}_{2n-1}^{k+1} - \mathbf{z}_{2n-1}^k$, $\mathbf{v}_n = \mathbf{z}_{2n}^{k+1} - \mathbf{z}_{2n}^k$ and analogously $\mathbf{c}_n$, $\mathbf{d}_n$ are the $(2n-1)$-st and $2n$-th block components of the vector $S^{-1}G(Y^k)$. Further amount of computation is gained transforming these $2r$-dimensional real subsystems into the r-dimensional complex systems

$$((I_r - h\alpha_n J) + i\beta_n J)\,(\mathbf{u}_n + i\mathbf{v}_n) = \mathbf{c}_n + i\mathbf{d}_n, \qquad n = 1, \dots, N/2. \tag{9}$$

The total work to obtain the solution of (4) is then shared as follows: one Jacobian evaluation; $N/2$ complex LU factorizations; for each iteration (9) the computation of $Y^k = TZ^k$, $F(Y^k)$, $S^{-1}G(Y^k)$ and the solutions of N triangular complex systems of dimension r.

4 Parallel Simplified Newton Iteration

A basic level parallelization is easily achieved solving the $N/2$ decoupled systems (9) in parallel. The number of processors required for the generic step will therefore depend on the order of the GAM chosen to advance the solution: 2, 3, 4, 5 processors are needed for the orders 3, 5, 7 and 9 respectively. Obviously, if the changing order rule is allowed to select the method between the minimum and the maximum order, five processes must be initialized at the start up of the program but some of them will remain idle unless the order 9 formula is directly involved in the integration step. When evaluating the expected speed-up one should therefore get information about the global work carried out by each process. Suppose a problem has been solved and the orders 3, 5, 7, 9 formulae contributed to the solution by n_3, n_5, n_7 and n_9 steps respectively (the rejected steps must also be included). Then for that particular execution an estimation of the expected speed-up S_e is

$$S_e = \frac{2n_3 + 3n_5 + 4n_7 + 5n_9}{n_3 + n_5 + n_7 + n_9} \in [2, 5]. \tag{10}$$

More precisely formula (10) represents an upper bound of the theoretical speed-up since it does not take into account the sequential nature of some parts of the code such as the Jacobian evaluation, the stepsize selection and the computation of the initial guess of the Newton iteration.

We see that, besides being problem dependent, once a problem has been fixed, the value of S_e is also related to the input tolerances and other possible input values. However comparisons between the real speed-ups S_r and expected speed-ups S_e are still possible because the numbers n_i together with the execution times are available as output variables of the code. In detail the iteration scheme (7) executed in parallel proceeds as follows. Initially each of the $N/2$ processes involved at the current step performs the LU factorization of the coefficient matrix of the corresponding system (9). Later, the elements of the sequence $\{Z^k\}$ are generated until convergence is attained. Assume that the process n has evaluated its own piece $[\mathbf{z}^k_{2n-1}, \mathbf{z}^k_{2n}]^T$ of the solution Z^k; let us see how the construction of $[\mathbf{z}^{k+1}_{2n-1}, \mathbf{z}^{k+1}_{2n}]^T$ is carried out. For $n = 1, \dots, N/2$, we set

$$\begin{aligned}
Z^k_n &= [\mathbf{0}, \mathbf{z}^k_{2n-1}, \mathbf{z}^k_{2n}, \mathbf{0}]^T \in H^N, \\
Y^k_n &= [\mathbf{0}, \mathbf{y}^k_{2n-1}, \mathbf{y}^k_{2n}, \mathbf{0}]^T, F^k_n = [\mathbf{0}, \mathbf{f}(t_{2n-1}, \mathbf{y}^k_{2n-1}), \mathbf{f}(t_{2n}, \mathbf{y}^k_{2n}), \mathbf{0}]^T \in H^N, \\
\mathbf{b}_n &= [\mathbf{0}, b_{2n-1}, b_{2n}, \mathbf{0}]^T \in \mathbb{R}^N, \\
G^k_n &= \mathbf{b}_n - AY^k_n + hBF^k_n,
\end{aligned}$$

and observe that

$$G(Y^k) = \sum_{n=1}^{N/2} G^k_n. \tag{11}$$

The program now performs the following steps (points 1),3) and 5) are performed for $n = 1, \dots, N/2$):

1) the process n computes $W_n = TZ_n^k$;
2) each process takes part in the computation of $Y^k = \sum_{n=1}^{N/2} W_n$ and receives the corresponding block $[\mathbf{y}_{2n-1}^k, \mathbf{y}_{2n}^k]^T$ (and therefore Y_n^k);
3) the process n computes $[\mathbf{f}(t_{2n-1}, \mathbf{y}_{2n-1}^k), \mathbf{f}(t_{2n}, \mathbf{y}_{2n}^k)]^T$ and then $G_n(Y_n^k)$;
4) exploiting the relation (11), all processes contribute to the computation of $S^{-1}G(Y^k)$ and receive the corresponding blocks of the known term $[\mathbf{c}_n, \mathbf{d}_n]$ (see formula (9));
5) the process n can finally solve the system (9) and get the solution $[\mathbf{u}_n, \mathbf{v}_n]$ and hence $[\mathbf{z}_{2n-1}^{k+1}, \mathbf{z}_{2n}^{k+1}]^T$.

The implementation described above represents a parallelization across the method similar to that used for Runge-Kutta methods with associated matrix having real and distinct eigenvalues (see for example [2]). There are a number of starting points to take into consideration for subsequent developments of the parallel code. As an instance, to avoid that some processors remain idle during the integration, one may chose $N = 10$ as the dimension of all considered GAMs. As a generalization, more than five processors could be activated considering $N > 10$, even though, in this case, the dependence of the convergence properties of the simplified Newton iteration on the dimension N, should be carefully studied.

5 Numerical Tests

In this section we present some numerical results related to the parallelization of the code GAM as described in section 3. The experiments were performed on a Cray T3E machine with distributed memory, using at most five processors. In the following the parallel code GAM will be referred to as P-GAM or P-GAM(n), where n is the number of processors used. The communications are performed by the MPI routines [10]. For numerical comparisons we choose the codes RADAU in the version of April 1998 [5] and GAM in the version of September 1997 [8].

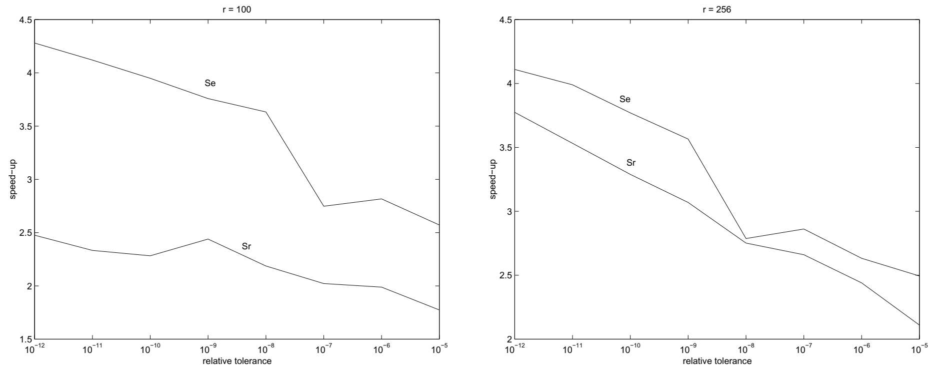

Fig. 1. SALESMAN problem

The first test problem is the travelling salesman problem described in [1]. We chose this problem in order to compare the expected speed-up (S_e) to the numerical one (S_r) by changing both the input parameters ($rtol = atol = h_0$) and the dimension r of the problem. In Figure 5 we show the results obtained for the dimensions $r = 100$ and $r = 256$ (the problem has a full Jacobian). The end is to quantify the dependence of the performance of the code on the communication times. We see that, how one should expect, overloading the processors, the obtained speed-ups are very close to the expected ones; this means that the communication times, which are $O(r)$, are almost negligible compared to the working times per step of each processor (which depends on $O(r^3)$). Better speed-ups for lower dimensions should be expected on shared memory parallel computers. We finally observe that the rise in speed-up (either S_e and S_r) when the accuracy of the solution is increased, is due to the involvement in the integration process of high order formulae which keep at work a greater number of processors.

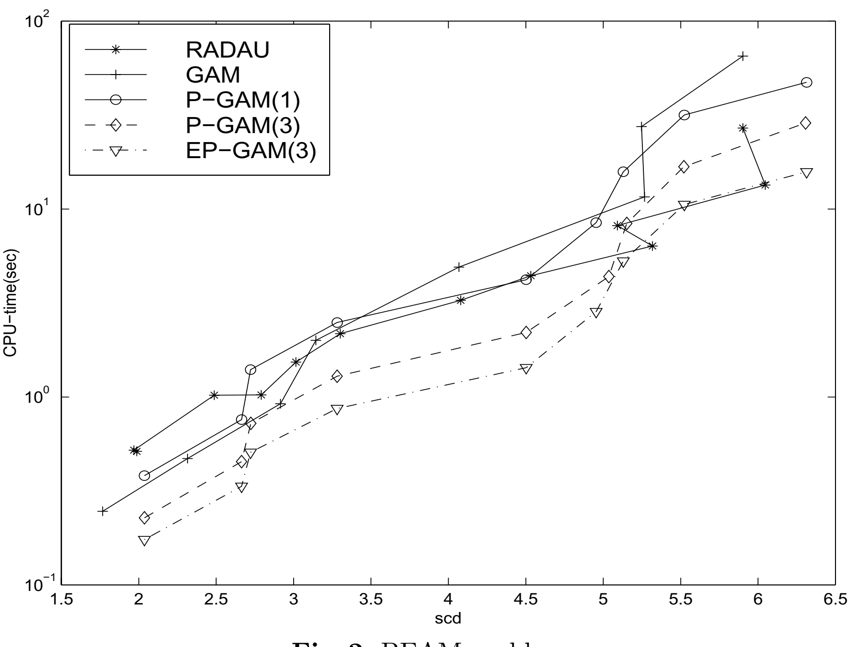

Fig. 2. BEAM problem

The second test is the BEAM problem described in [9], an ODE of dimension 80. In Figure 5 we report the work precision diagram, that is a range of input tolerances and a range of initial stepsizes were used to produce a plot of the resulting minimum number of significant digits in the components of the numerical solution at the endpoint (*scd*) against the *CPU* time in seconds needed for the run (observe that a logarithmic scale is used for the *y*-axis). The format of this diagram is as in [4]; naturally it strongly depends on the input tolerances and on the default parameters of each code; the variation of a parameter may considerably change the diagrams. For all problems we fixed $atol = rtol = h_0$ while the values of $rtol$ were chosen as follows: $10^{-(2+m)}$, $m = 0, \ldots, 10$, for RADAU and $10^{-(2+m)}$, $m = 0, \ldots, 8$ for GAM and P-GAM. A lower bound for the expected execution times is obtained dividing the CPU times of P-GAM(1) by S_e and is drawn as EP-GAM(3). The expected speed-up is for all tolerances less than 3 since the maximum order used is 5, the numerical speed-up is about 2.

References

[1] Bellen, A.: Pade test - A set of real-life test differential equations for parallel computing, Technical Report 103, Università di Trieste (1992)

[2] Bendtsen, C.: A parallel stiff ODE solver based on MIRKs, Adv. Comput. Math. **7** 1-2 (1997) 27-36

[3] Brugnano L., Trigiante D.: Solving Differential Problems by Multistep Initial and Boundary Value Methods, Gordon & Breach, Amsterdam, (1998)

[4] Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II. Stiff and Differential–Algebraic Equations, Springer Series in Computational Mathematics, **14**, Springer-Verlag, Berlin, (1996)

[5] Hairer, E., Wanner, G.: RADAU, April 1998. Available via WWW at URL ftp://ftp.unige.ch/pub/doc/math/stiff/radau.f

[6] Iavernaro, F., Mazzia, F.: Block-Boundary Value Methods for the solution of Ordinary Differential Equations, SIAM J. Sci. Comput. (to appear)

[7] Iavernaro, F., Mazzia, F.: Solving ordinary differential equations by Generalized Adams Methods: properties and implementation techniques , Appl. Num. Math. **28** 2-4 (1998) 107–126

[8] Iavernaro, F., Mazzia, F.: F GAM August 1997. Available via WWW at URL http://www.dm.uniba.it/~mazzia/ode/readme.html.

[9] Lioen, W. M., de Swart, J.J.B., van der Veen, W. A.: Test Set for IVP Solvers, CWI, Department of Mathematics, Amsterdam, Report NM-R9615, (1996)

[10] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard, (1995)