Parallel Methods for Integrating Ordinary Differential Equations

J. Nievergelt
University of Illinois, Urbana, Illinois

This paper is dedicated to the proposition that, in order to take full advantage for real-time computations of highly parallel computers as can be expected to be available in the near future, much of numerical analysis will have to be recast in a more "parallel" form. By this is meant that serial algorithms ought to be replaced by algorithms which consist of several subtasks which can be computed without knowledge of the results of the other subtasks. As an example, a method is proposed for "parallelizing" the numerical integration of an ordinary differential equation, which process, by all standard methods, is entirely serial.

Introduction

For the last 20 years, one has tried to speed up numerical computation mainly by providing ever faster computers. First, this was achieved by improvements in hardware. Today, as it appears that one is getting closer to the maximal speed of electronic components, the emphasis is put on improving the logical organization of computers, allowing certain operations to be performed in parallel or at least in overlapping time intervals. In order to achieve this, designers are quite willing to introduce considerable redundancy in hardware and to use methods (for instance for performing arithmetic operations) which are inefficient as far as the ratio (time)/(cost) is concerned, as long as the total time is minimized.

The problem of speeding up computation has also been attacked at the mathematical level, long before the advent of computers, but there it has taken an entirely different direction. In numerical analysis, one has always tried to speed up computation by reducing the amount of work to be done, not by performing redundant computations. Typical problems studied in numerical analysis are, e.g., to find algorithms which produce some result in the least number of operations, to improve the rate of convergence of a given algorithm (so that fewer steps need be performed to achieve a given accuracy), and so forth.

In this paper we propose a different approach to the problem of speeding up computation at the mathematical level, an approach similar to the one which has been in use for some time in computer design: by introducing redundancy in computation, we try to recast an essentially serial algorithm into a form in which it consists of several subtasks which can be performed in parallel, i.e., can be computed without knowledge of the results of the other subtasks.

Since there seems to be no general procedure for "parallelizing" serial algorithms, we have chosen one problem as an example, namely, the integration of $y' = f(x, y)$, $y(a) = c \quad (a \leqq x \leqq b)$.

Compared to any standard (serial) integration method, our methods are inefficient in the sense that, for equal accuracy, they involve more work. However, if we have at our disposal a computer capable of performing the parallel subtasks simultaneously, we will compute the result faster.

The utility of the suggested methods therefore is based on the following two assumptions:

1. There are computations in which ultimate speed is essential (e.g., real time processes like orbit calculations and weather prediction).
2. A computer capable of executing many computations simultaneously is available.

Assumption 1 has been a reality for quite some time now. Concerning 2, it seems certain that within the near future the current work on multiprocessors and multicomputer systems will result in a machine which meets those specifications (see e.g., [1]).

An important restriction we impose on a parallel method is that the computation time of each subtask is long compared to the average operation time of the computer, so that the time of providing every computing unit with initial data, collecting and transferring results, and bookkeeping time in general is negligible compared to computation time proper. This last requirement rules out what might be called *microparallelization*, namely, the attempt to exploit that amount of parallelism which is inherent in almost every mathematical expression (as, e.g., in $e = (a + b)(c + d)$, where the two additions could be performed simultaneously).

Outline of the Method

The idea is to divide the integration interval [a, b] into N equal subintervals $[x_{i-1}, x_i]$, $x_0 = a$, $x_N = b$, $i = 1, 2, \cdots, N$, to make a rough prediction y_i^0 of the solution $y(x_i)$, to select a certain number M_i of values y_{ij}, $j = 1, 2, \cdots, M_i$ in the vicinity of y_i^0, $i = 1, 2, \cdots, N$, and then to integrate *simultaneously* with an accurate integration method $\mathfrak{M}$ all the initial value problems (see Figure 1):

$$y' = f(x, y), \qquad y(a) = c, \qquad a \leqq x \leqq x_1$$

$$y' = f(x, y), \qquad y(x_i) = y_{ij}, \qquad \begin{cases} x_i \leqq x \leqq x_{i+1}, \\ j = 1, \cdots, M_i, \\ i = 1, \cdots, N - 1 \end{cases}$$

After a time which equals the time used for integrating from x_0 to x_1 with the accurate method $\mathfrak{M}$, we will have covered the whole interval $[a, b]$ with lines of length $(b - a)/N$ which are solutions of $y' = f(x, y)$ but, of course, do not join together at their ends. Let us call these lines *solution branches* or just *branches*. The connection between the branches is now brought about by interpolating the end value of the unique branch in $[x_0, x_1]$ with the M_1 branches in $[x_1, x_2]$, then interpolating the end value of x_2 with the set of M_2 branches in $[x_2, x_3]$, etc., until we reach $x_N = b$.

If only the values at $x_1, \cdots, x_N$ are wanted, we can

now stop; otherwise, if the solution is also needed at intermediate points (used by the method $\mathfrak{M}$), we still have to do a number of interpolations. These, however, can all be done simultaneously (at least in principle).

The result will have been computed in one Nth the time used for (serial) integration from a to b using $\mathfrak{M}$, plus the time for the initial predictions y_i^0, for interpolation, and for some additional bookkeeping due to the parallelism of the method. Under the usual assumption that the really time-consuming part is the evaluation of $f(x, y)$, these contributions to the total time of computation become negligible, so that we end up with a saving in time by roughly a factor $1/N$.

In order to compare this method with serial integration from a to b using $\mathfrak{M}$, one has to study the error introduced by interpolating at $x_1, x_2, \cdots, x_N$. This interpolation error I depends on the equation, not on the method, and thus one is led to studying classes of equations for which this error can be bounded or estimated.

Linear Equations

For first-order linear differential equations

$$y' = u(x)y + v(x) \tag{1}$$

the problem is very simple and neat, due to the fact that interpolation does not introduce any error.

THEOREM. *For a linear equation* (1) *the accuracy of the parallel method described above is the same as if the method* $\mathfrak{M}$ *had been used over the whole interval* $[a, b]$.

PROOF. Let $z(t; x, y)$ denote the exact solution of the initial value problem $z' = f(t, z)$, $z(x) = y$.

The general solution of (1) has the form $y(x) = g(x) + kh(x)$ with fixed functions g and h and an arbitrary constant k. With this notation we have $z(t; x, y) = g(t) + kh(t)$ where $k = (y - g(x))/h(x)$.

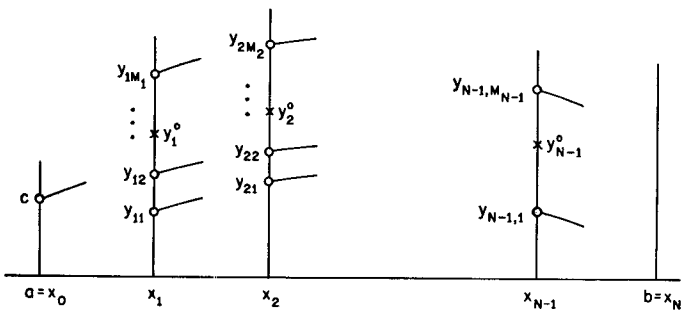

FIG. 1

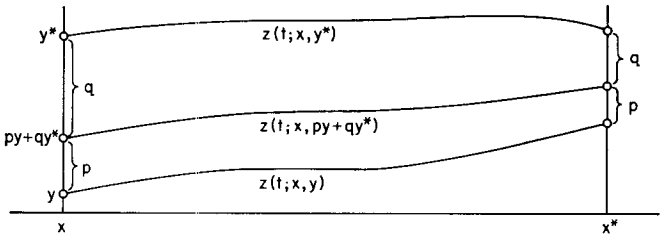

FIG. 2

Now consider linear inter polation between two branches $z(t; x, y)$ and $z(t; x, y^*)$. See Figure 2. For any p, q such that $p + q = 1$,

$$\begin{aligned} z(x^*; x, py + qy^*) &= g(x^*) + \frac{py + qy^* - g(x)}{h(x)} h(x^*) \\ &= p\left[g(x^*) + \frac{y - g(x)}{h(x)} h(x^*)\right] \\ &\quad + q\left[g(x^*) + \frac{y^* - g(x)}{h(x)} h(x^*)\right] \\ &= pz(x^*; x, y) + qz(x^*; x, y^*). \end{aligned}$$

Since we have not used the restriction $p \geqq 0$, $q \geqq 0$, this result holds for extrapolation as well as interpolation, and thus has the surprising consequence that, for linear equations, we need to compute only two branches in every interval $[x_i, x_{i+1}]$, with whatever initial values y_{i1}, y_{i2} we like, and by mere linear inter- or extrapolation between these two branches, construct a solution with the same accuracy as the integration over the whole interval $[a, b]$. Notice that, by roughly doubling the amount of work, we have reduced the time by a factor $1/N$ (where N, of course, cannot be chosen arbitrarily large because $(b - a)/N$ still has to be large compared to the step size h of $\mathfrak{M}$).

Nonlinear Equations

For nonlinear equations interpolation does introduce an error. This forces us to consider a number of problems which one can, at least theoretically, ignore in the case of linear equations, namely:

1. How to choose the predicted values y_i^0.
2. How to choose the multiplicity numbers M_i.
3. How to choose the distance between the various initial values $y_{i1}, y_{i2}, \cdots, y_{iM_i}$ (the "branch width").

The goal we want to attain is that, at every point x_i, $i = 1, 2, \cdots, N - 1$, the just constructed solution will end up between two outgoing branches. Provided the equation has no singularity in the region under consideration, this will guarantee that at the next point x_{i+1}, the continuation of the solution will still be between the two selected branches. The fact that there is no crossover of solution branches is all that one can assert in the general case. Therefore, a reasonable method will have to make a compromise between the following conflicting requirements:

1. A small multiplicity number M_i in order to reduce the amount of work.
2. A large total width between the two outermost branches, in order to guarantee that one can always interpolate, never extrapolate.
3. A small branch width so that the continuation of the solution can be done accurately.

Unless one makes some assumption about the family of solutions, the choice of multiplicity and branch width will be entirely arbitrary. The kind of assumption one

can make is that the (linear) interpolation error

$$I(x, x^*, y, y^*, p, q) = | z(x^*; x, py + qy^*) - pz(x^*; x, y) - qz(x^*; x, y^*) |$$

can be estimated or bounded.

Then a reasonable choice of multiplicity and branch width would be such that the interpolation error is of the same order of magnitude as the truncation error of the method $\mathfrak{M}$.

The following section will pursue these ideas somewhat further in the case of a specific method which is more amenable to treatment than the general case.

A Specific Method

Let us first define a class of differential equations which has the property that, for linear interpolation between any two of its solution branches over an interval of length equal to or less than l, the interpolation error is bounded by the right-hand expression below.

Definition: $y' = f(x, y)$ is in the class $C(l, k)$ in $[a, b]$ iff for all x, x^*, y, y^*, p, q such that $x, x^* \in [a, b]$, $| x - x^* | \leqq l$, $p \geqq 0$, $q \geqq 0$, $p + q = 1$, we have:

$$| z(x^*; x, py + qy^*) - pz(x^*; x, y) - qz(x^*; x, y^*) | \leqq k(x - x^*)^2$$

The parallel method to be described will use Euler's method both for computing the predicted values y_i^0 and as the "accurate" method $\mathfrak{M}$, in the first use with a step size $H = (b - a)/N$, in the second with a step size h which is a submultiple of H. For this simple and not very efficient method there exist convenient error bounds, and we will use the following two theorems from [2]:

THEOREM 1.3. *Let $f(x, y)$ satisfy the Lipschitz condition*:

$$| f(x, y) - f(x, y^*) | \leqq L | y - y^* |$$

for all $x \in [a, b]$. *Let the exact solution $y(x)$ of $y' = f(x, y)$, $y(a) = c$, be twice continuously differentiable in $[a, b]$, and let*

$$N(x) = \tfrac{1}{2} \max_{a \leqq t \leqq x} | y''(t) |.$$

Let $y_0 = c$, $y_{i+1} = y_i + hf(x_i, y_i)$, $x_{i+1} - x_i = h$. Then the error $| y_i - y(x_i) |$ of Euler's method satisfies

$$| y_i - y(x_i) | \leqq hN(x_i) \frac{\exp(L(x_i - a)) - 1}{L}.$$

THEOREM 1.4. *Under the same hypotheses on $y(x)$, $N(x)$ and $f(x, y)$, let y_i be any sequence of numbers satisfying*

$$y_0 = c, \quad y_{i+1} = y_i + hf(x_i, y_i) + \theta_i h^2 k$$

where $k \geqq 0$ is a constant and θ_i are numbers such that $| \theta_i | \leqq 1$. Then

$$| y_i - y(x_i) | \leqq h[N(x_i) + k] \frac{\exp(L(x_i - a)) - 1}{L}.$$

Our parallel method now is as follows:

1. Compute the predicted values

$$y_i^0 = c, \quad y_i^0 = y_{i-1}^0 + Hf(x_{i-1}, y_{i-1}^0), \quad i = 1, 2, \cdots, N - 1$$

2. Choose a constant multiplicity number $M_i = 2$.
3. Let the branch width be any number b_i such that

$$b_i \geqq H[N(x_i) + k] \frac{\exp(L(x_i - a)) - 1}{L}.$$

(Since the solution $y(x)$ is not yet known, $N(x)$ is also unknown. However it is often possible to give an estimate or bound for $y''(x)$, hence also for $N(x)$.)

4. Choose the initial values for the branches in $[x_i, x_{i+1}]$ to be $y_{i1} = y_i^0 - b_i/2$, $y_{i2} = y_i^0 + b_i/2$, $i = 1, 2, \cdots, N - 1$.
5. Solve the $2N - 1$ initial value problems

$$y' = f(x, y), \quad y(0) = c \qquad (a \leqq x \leqq x_1)$$

$$y' = f(x, y), \quad y(x_i) = y_{i1}, \quad \text{and } y(x_i) = y_{i2} \qquad \begin{cases} x_i \leqq x \leqq x_{i+1} \\ i = 1, 2, \cdots, N - 1 \end{cases}$$

using Euler's method with a step size $h \leqq l$ which is a submultiple of H.

6. At $x_1, x_2, \cdots, x_{N-1}$, interpolate linearly the previously found solution over the next interval to the right.

THEOREM. *For an equation of the class $C(l, k)$, the above method guarantees that continuation of the solution can always be done by interpolation, and the error of the method satisfies*

$$| y_i - y(x_i) | \leqq h[N(x_i) + k] \frac{\exp(L(x_i - a)) - 1}{L}$$

at every point x_i used by the Euler method with step size h.

PROOF. Straightforward consequence of Theorems 1.3 and 1.4.

Conclusion

The integration methods introduced in this paper are to be regarded as tentative examples of a much wider class of numerical procedures in which parallelism is introduced at the expense of redundancy of computation. As such, their merits lie not so much in their usefulness as numerical algorithms as in their potential as prototypes of better methods based on the same principle. It is believed that more general and improved versions of these methods will be of great importance when computers capable of executing many computations in parallel become available.

Acknowledgments. I would like to thank Professor C. W. Gear, Dr. G. Batten, and Messrs. C. Koffmann, J. Presti and F. Schaffer for the opportunity of discussing these ideas.

RECEIVED JUNE, 1964

REFERENCES

1. COMFORT, W. T. Highly Parallel Machines. In *Workshop on Computer Organization*, Barnum, Knapp (Eds.), Spartan Books, 1963.
2. HENRICI, P. *Discrete Variable Methods in Ordinary Differential Equations*. John Wiley, New York, 1962.