AIAA 81-1013R

Low-Cost Three-Dimensional Flow Computations Using a Minisystem

M. Enselme,* J. Brochet,† and J. P. Boisseau‡
Office National d'Etudes et de Recherches Aérospatiales, Châtillon, France

A three-dimensional Euler code for the computation of transonic steady flows using an explicit pseudo-unsteady method has been vectorized on an array processor (AP 120B) linked to a host computer (SEL 32/77). Computation speed comparable to that of a CDC 7600 computer is achieved by executing strings of vectorized operations in the array processor simultaneously with calculations at boundary points in the SEL computer and with data transfers between the mass memory and the host computer. A numerical application is presented for transonic flow in a fan rotor.

I. Introduction

As a result of the rapid progress in numerical methods and in computer hardware, the importance of computational fluid dynamics in the field of aircraft design has been steadily increasing. The computer is becoming a complementary tool to the wind tunnel, which makes it possible to shorten design times and to obtain better solutions. Nevertheless, computers which are able to compute the flowfield around a complete aircraft configuration in a short time and which take into account all the essential physical features of the flow are not yet available. Such computers could become available with the NAS project.[1] Meanwhile, many needs can be met by using small fast units able to run long computations at low cost, as proposed by Taylor and Widhopf.[2] This possibility is being studied at ONERA using a system made up of an array processor AP 120B linked to a host computer SEL 32/77. The numerical codes selected for evaluating the performance of this miniprocessing system are based on explicit pseudo-undsteady finite-difference methods developed at ONERA[3,4] for the calculation of steady two- or three-dimensional flows. These methods are well suited for this evaluation since the numerical codes can be easily vectorized and a continuous data flow between the memories and the processing unit can be assured. A first evaluation of the performance of this system has been reported in Ref. 5. Recent results obtained for three-dimensional flow computations are presented here. After a review of the numerical method and a presentation of an application to a fan rotor (Sec. II), the structure and the operating rules of the system are presented in Sec. III. The performances achieved are discussed in Sec. IV.

II. Numerical Method

Pseudo-unsteady System

The flows studied in this paper are inviscid and steady in a rotating reference frame with a constant angular velocity Ω about the axis $(0,\vec{e}_z)$. Such flows are governed by the Euler equations, which are steady when they are written in the rotating frame. For transonic flows, these equations are solved by means of a pseudo-unsteady (PU) method. The steady-state solution is thus obtained by solving a so-called PU system step by step with respect to time. A general discussion of the PU approach has been presented by Veuillot and Viviand[3,4] and Viviand.[6] The PU system used here is an extension to steady rotating flows of the conventional PU system, which is based on replacing the unsteady energy equation by the steady Bernoulli equation. It is made up of two unstedy equations for density and momentum, namely:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla(\rho \bar{W}) = 0 \\ \dfrac{\partial \rho \bar{W}}{\partial t} + \nabla(\rho \bar{W} \times \bar{W} + p\bar{I}) + 2\Omega \vec{e}_z \wedge \rho \bar{W} - \rho \Omega^2 \bar{r} = \bar{0} \end{cases} \tag{1}$$

In these equations, ρ, $\bar{W}$, p, $\bar{r}$ designate the density, the relative velocity, the static pressure, and the radius vector, respectively. For an ideal gas with constant specific heats of ratio γ, the PU system (1) is completed by the relation

$$\frac{\gamma}{\gamma - 1}\frac{p}{\rho} + \frac{\bar{W}^2 - \Omega^2 \bar{r}^2}{2} = \text{const} \tag{2}$$

This relation signifies that the rothalpy is constant; it is an exact first integral of the Euler equations for steady rotating flows with uniform rothalpy far upstream. The weak steady solutions of the PU system (1) verify the Euler equations including the exact jump relations across discontinuities (shock waves and slip surfaces). The computational variables are the density and the three Cartesian coordinates of the relative momentum. The static pressure is a known function of these variables given by relation (2).

Boundary conditions and initial conditions complete the PU system (1). In order for this initial-value problem to be well posed, system (1) must be hyperbolic with respect to time. This condition, which is known to be satisfied when $\Omega = 0$ (see, e.g., Ref. 7), remains satisfied even if $\Omega \neq 0$. Indeed, the rotation of the frame introduces only underived terms which play no role in the mathematical nature of the PU system. Therefore the mathematical properties of system (1) (which later will be used in treating the boundary conditions) will only be briefly recalled.

According to the definition of the hyperbolicity with respect to time, the characteristic matrix associated with any space direction $\vec{\xi}$ has four real eigenvalues (as many as the equations of the PU system). These eigenvalues, denoted λ_-, λ_+, and

Presented as Paper 81-1013 at the AIAA Fifth Computational Fluid Dynamics Conference, Palo Alto, Calif., June 22-23, 1981; submitted June 29, 1981; revision received Jan. 4, 1982.

*Assistant Head, Informatics Department.
†Research Engineer, Aerodynamics Division.
‡Research Engineer, Informatics Department.

λ_0 (λ_0 is double), have the following expressions:

$$\lambda_\pm = C_\pm + W_\xi/\gamma$$
$$\lambda_0 = W_\xi \tag{3}$$

with

$$C_\pm = \frac{\gamma-1}{2\gamma} W_\xi \pm \sqrt{\frac{a^2}{\gamma} + \left(\frac{\gamma-1}{2\gamma} W_\xi\right)^2}$$

$$W_\xi = \bar{W}\bar{\xi}/|\bar{\xi}|$$

The characteristic velocities C_+ and C_- play the same role as the algebraic speeds of sound $+a$ and $-a$ in the original unsteady Euler equations. Four independent left-eigenvectors are associated with these eigenvalues. It is convenient to represent the four components of each eigenvector by a set (α, $\bar{\beta}$), where α is a scalar and where $\bar{\beta}$ is identified with a space vector. The corresponding compatibility relations can thus be written in this simple intrinsic form:

$$\alpha\left\{\frac{\partial\rho}{\partial t} + \nabla(\rho\bar{W})\right\} + \bar{\beta}\left\{\frac{\partial\rho\bar{W}}{\partial t} + \nabla(\rho\bar{W}\times\bar{W} + p\bar{\bar{I}})\right.$$
$$\left. + 2\Omega\bar{e}_z \wedge \rho\bar{W} - \rho\Omega^2\bar{r}\right\} = 0 \tag{4}$$

The method used to calculate these coefficients and their values when $\Omega = 0$ is given in Ref. 7.

Boundary Conditions

The treatment of the boundary conditions is based on two factors: first, the use of the particular compatibility relations associated with the unit outward vector $\bar{\xi}$ normal to the boundary surface; second, the interpretation of these four compatibility relations as "transport equations" along directions of the plane $(\bar{\xi},t)$ with a slope equal to $(1/\lambda)$, λ being one of the four eigenvalues previously defined. According to this interpretation, the relations corresponding to negative values of λ transport information coming from outside the computational domain. Therefore they cannot be used and they are replaced by boundary conditions the nature of which depends on the problem treated. The variables on a boundary surface are thus determined from a system Σ consisting of the m compatibility relations associated with the m positive or zero eigenvalues $\lambda(\bar{\xi})$ and of $(4-m)$ boundary conditions. The value of the integer m depends only on the normal relative Mach number W_ξ/a.

Numerical Scheme

The PU system (1) is integrated step by step with respect to time by means of an explicit, predictor-corrector, finite-difference scheme. The discretization of the equations is performed directly in the physical space using an arbitrary curvilinear mesh. The expression of this numerical scheme, which is derived from the unsplit MacCormack scheme, is given in Ref. 7. We are interested in explicit schemes because they are very simple to program on a multiprocessing system.

Discretization of the Boundary Conditions

The treatment of the boundary conditions is performed in two steps in the same manner as the one proposed by Veuillot and Viviand.[3,4] In the first step, the compatibility relations, Eq. (4), used at the boundary points are discretized by means of the same numerical scheme as the one applied at the inner points. The discretized compatibility relations have a very simple form when they are expressed in terms of the values of the variables given by the numerical scheme at the boundary points. At the time level $(n+1)$, they are simplified to linear

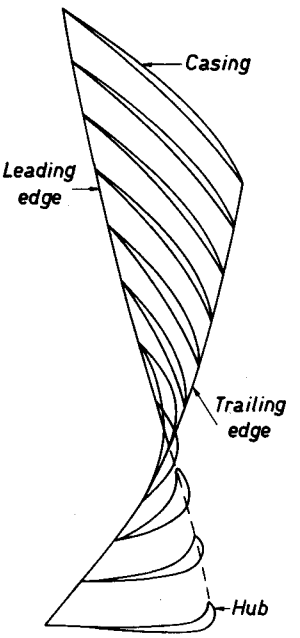

Fig. 1 Perspective view of one blade of a fan rotor.

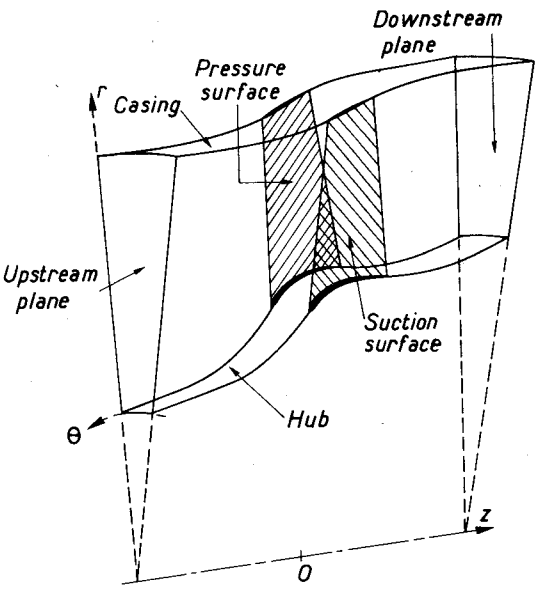

Fig. 2 Computational domain for turbomachine blade row.

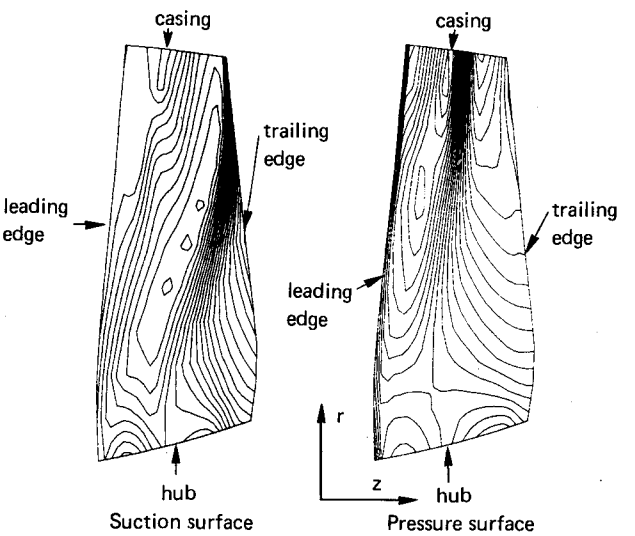

Fig. 3 Isobaric lines on a blade ($\Delta p/p_0 = 0.05$).

equations of the type (see Ref. 7):

$$\alpha^n(\rho^{n+1}-\rho_s^{n+1})+\tilde{\beta}^n(\rho\tilde{W}^{n+1}-\rho\tilde{W}_s^{n+1})=0 \qquad (5)$$

In this expression the subscript s denotes the values given by the numerical scheme and the superscripts n and $n+1$ indicate at which time level the values are evaluated. In the second step, the discretized system Σ, which is made up of m equations like the above Eq. (5) and of $(4-m)$ boundary conditions, is solved. In all cases, this solution is very simple. The main advantage of this two-step technique is that treating the boundary conditions amounts to simple modifications of the values given by the numerical scheme at the boundary points. This makes it very easy to program the treatment of the boundary conditions. This two-step technique can be used with all explicit numerical schemes.

Numerical Application

Computations of various internal transonic flows have been carried out by means of the above-described method. An example is presented hereafter for the rotating row of a fan. The rotating row has been chosen, in particular, on account of the high degree of twisting of its blades (about 70 deg). The twisting can be seen on the perspective view of one blade (Fig. 1). The computational domain is limited to a blade-to-blade channel (Fig. 2). This channel is extended upstream and downstream up to two planes normal to the z axis of the turbomachine and far away from the blades. This computational domain is discretized with $65\times11\times11=7865$ grid points. There are 11 axisymmetrical surfaces of index k regularly spaced between the hub ($k=1$) and the casing ($k=11$). There are also 11 surfaces of index j regularly spaced between the suction surface ($j=1$) and the pressure surface ($j=11$) of two consecutive blades. Finally, 65 axisymmetrical surfaces of index i are spaced between the upstream plane ($i=1$) and the downstream plane ($i=65$). Of these, 20 are located upstream of the blades, 25 in the blade-to-blade channel, and 20 downstream of the blades. The number of blades is 25. The rotation speed is 1138 rad/s at an absolute stagnation temperature of 288 K. A uniform distribution of entropy is imposed on the upstream plane and the absolute velocity is assumed to be axial. A nonuniform distribution of static pressure is applied on the downstream plane. This distribution is obtained by means of an axisymmetrical mean flow computation which was carried out earlier. The slip condition is imposed on the hub, the casing, the pressure surface, and the suction surface. Finally, the flow is assumed to be periodic upstream and downstream of the blades. The periodicity condition is treated by means of compatibility relations in the same manner as the one proposed in Ref. 7. For the chosen back-pressure value, the upstream absolute Mach number is nearly constant on the upstream plane and is equal to 0.62. This corresponds to an upstream relative Mach number of 1.52 on the casing and 0.75 on the hub. Figure 3 shows isobaric lines on the surface of a blade. These lines are projected onto a meridian plane (z,r). Figure 4 shows isobaric lines on three grid surfaces of index k. These lines are projected onto the plane $(z, \tilde{r}\theta)$, $\tilde{r}$ being the mean radius of the grid surface of index k. The increment Δp is equal to $0.05p_0$, where p_0 is the upstream absolute stagnation pressure. The relative flow on the casing is of the supersonic-subsonic type with a strong shock near the trailing edge of the suction surface. This shock closes off the blade-to-blade channel entirely. As we go nearer to the hub, the shock strength decreases as does the upstream relative Mach number. Near the hub, the shock vanishes. Convergence to the steady state is rather slow. The root-mean-squared values of the time derivatives of the unknowns are approximately 2×10^{-3} after 3000 time steps (referred to as the absolute sonic values).

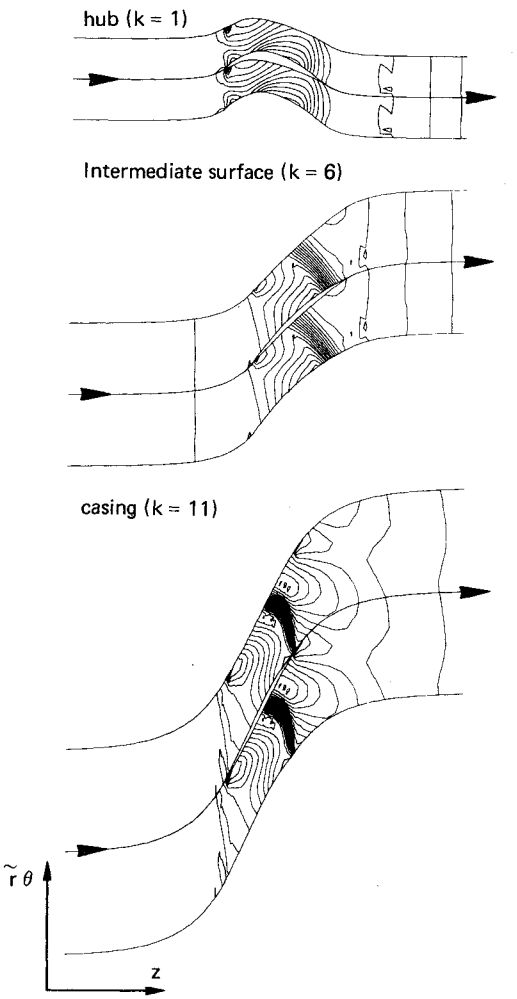

Fig. 4 Isobaric lines on three grid surfaces of index $k(\Delta p/p_0=0.05)$.

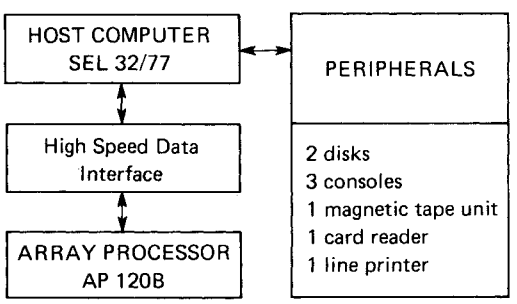

Fig. 5 Configuration of the processing system.

III. The Processing System and its Programming

The processing system is given in Fig. 5. The array processor AP 120B is linked to a host minicomputer, the SEL 32/77. This host computer has been chosen, in particular, because of its high speed data interface and its internal bus. The former allows a 3.2-Mbytes/s rate and the latter can handle a data flow up to 26.7 Mbytes/s. These values are compatible with a multiprocessing system mode using several array processors.

From the user's viewpoint, the array processor is constructed around a floating adder and a floating multiplier. These are pipelined and at best can provide two results at each cycle (166 ns) so that the maximum rate which can be reached is 12×10^6 floating operations per second (12 MFLOPS). This

optimum can be attained owing to two factors: the multiple 38-bit data paths which connect the input operators to different memories and output operators; and the instruction defined in a 64-bit word, which consists of as many as eight subinstructions executed simultaneously at each cycle. As the memory is insufficient to handle all the data needed in a three-dimensional application, it has been necessary to attach two disks, each allowing a continuous data transfer rate of 300K words/s. The data transfers between the disks and the AP main data memory are handled by the host computer and require two instructions: the first allowing the transfer between the disks and the SEL memory; the second allowing the transfer between the SEL memory and the AP main data memory. These two transfers are executed without stopping the program in the array processor. Nevertheless, this constitutes a bottle-neck during processing, especially as the average transfer rate during execution is 100K words per disk. This is mainly due to the access time needed at each data block transfer.

In order to assure a parallelism between data transfer and AP processing, it is necessary to characterize the architecture and define a suitable procedure. The first well-known parameter is the FLOPS number which characterizes the processing rate of the system. In fact, the architectural possibilities allow a theoretical FLOPS number which is not obtainable on account of the impossibility of correctly feeding the arithmetic operators: either the process does not need the two AP operators at the same time, or there is no free path to transfer data to the operator input.

For a task which is a part of a programmed algorithm, it is easy to define an average number of floating operations per variable accessed from the memory: $(\text{FLOPVAM})_{\text{algo}}$. We propose introducing a similar number, denoted $(\text{FLOPVAM})_{\text{archi}}$, making it possible to characterize the architectural possibilities of the system (SEL + AP). Figure 6 illustrates the relation between data processing and data transfer with the theoretical FLOPS number and the data transfer rate (DTR) in words/s. If the $(\text{FLOPVAM})_{\text{algo}}$ number is greater than the ratio FLOPS number/DTR, the data transfer time is lower than the data processing time and the processing rate is roughly equal to the FLOPS number of the unit. Otherwise, the execution speed of a program is limited by the data transfer rate and the processing rate is equal to the product $\text{DTR} \times (\text{FLOPVAM})_{\text{algo}}$ (Fig. 7). The parameter $(\text{FLOPVAM})_{\text{archi}}$ is the ratio FLOPS number/DTR. The processing rate is a function of the parameter $(\text{FLOPVAM})_{\text{algo}}$, and the best processing rate is achieved when the parameter $(\text{FLOPVAM})_{\text{algo}}$ is greater than the parameter $(\text{FLOPVAM})_{\text{archi}}$. So, a processing method which is well suited to the

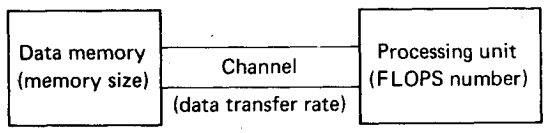

Fig. 6 Diagram of a processing system with its main characteristic parameters.

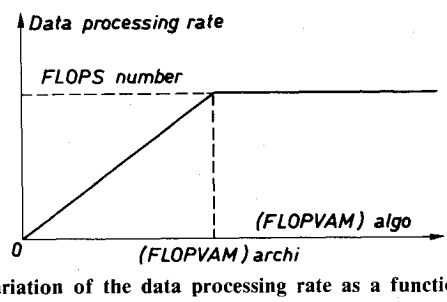

Fig. 7 Variation of the data processing rate as a function of the $(\text{FLOPVAM})_{\text{algo}}$ number.

Table 1 Values of the $(\text{FLOPVAM})_{\text{archi}}$ number

Memory type	Memory size (in kwords)	$(\text{FLOPVAM})_{\text{archi}}$
AP data memory	64	[0.5, 2]
SEL memory	2×128	[15, 20]
Disks	80,000	[40, 80]

architectural possibilities of a processing system would satisfy this inequality. The parameter $(\text{FLOPVAM})_{\text{archi}}$ varies according to the memory type from which data are fetched. The different values of this parameter in the system (SEL + AP) and the corresponding memory sizes are given in Table 1.

Now we have to answer some questions: how can such a system be used and what about the programming? The program is coded in the host computer with host FORTRAN instructions and some "calls" to AP Math Library routines or user's subroutines. It runs in the host computer which "calls" the array processor to execute some processing tasks on data which reside in the AP memories.

Software packages are supplied with the array processor to help the user in running and writing programs. An AP executive allows the user to communicate with the array processor via FORTRAN, interprets the user "calls," and finally directs the array processor to execute a specified task. The AP Math Library, which is an important part of the system, covers a wide range of processing needs. The library consists of approximately 300 routines and is, above all, well suited to the signal processing. However, in the case of the above-described numerical method, it has been necessary to write some specific subroutines. These subroutines, like the library routines, are coded using the AP assembly language. This makes it possible to employ all the possibilities of the array processor. This also improves the processing speed and minimizes the AP memory requirements. Let us consider the following example:

$$W(i) = a_1(i)\{V_1(i) - V_0(i)\} + a_2(i)\{V_2(i) - V_0(i)\}$$

with $1 \le i \le N$. Let us suppose that all the data concerned are stored in the AP memory. Two methods are compared: first, using library routines, and second, using hand-coded routines. In the first case, the routine is made with three "call" instructions which successively compute the three vectors D_1, D_2, and W in the following way:

$$D_1(i) = V_1(i) - V_0(i)$$

$$D_2(i) = V_2(i) - V_0(i)$$

$$W(i) = a_1(i)D_1(i) + a_2(i)D_2(i)$$

$(3+3+5)N = 11N$ cycles are necessary for fetching and storing data in the main memory. Therefore, to execute only $4N$ operations, the maximum speed is only 2 MFLOPS, and it is necessary to store the $2N$ intermediate values $D_1(i)$ and $D_2(i)$. In a hand-coded subroutine, it is possible to constitute $W(i)$ step by step without having to store intermediate vectors in the main memory but by storing only components in the AP register blocks. Access to these registers and to the main memory can be obtained at the same time. In this way, it is only necessary to use $6N$ cycles to fetch the five variables and store the result in the main memory. The run is almost twice as fast, and $2N$ memories are not used.

From this comparison a programming method is deduced according to the efficiency expected by the user with respect to speed, array size, and also the possibility of library routines. In the case of the above-described numerical method, these possibilities are very limited. In order to obtain a fast run, it was necessary to develop some hand-coded routines. However, in other cases, such as a pseudo-spectral method

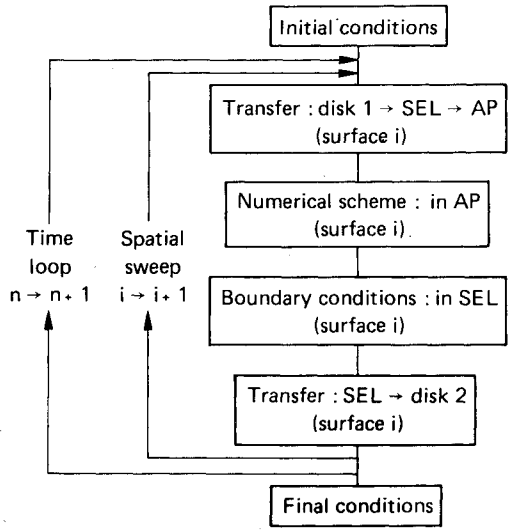

Fig. 8 Algorithm of the three-dimensional Euler code.

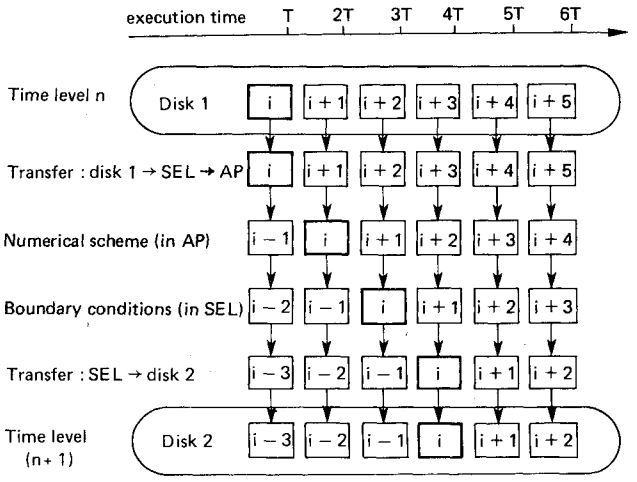

Fig. 9 Pipelined organization of the three-dimensional Euler code.

application, the library routines are well adapted. It should be added that the general layout of the complete system will have to be considered for choosing the type of programming. Indeed, if, for instance, the $(FLOPVAM)_{algo}$ number is too low, no improvement in performance is obtained by using hand-coded routines. Therefore it is not necessary to make such a programming effort in this case.

IV. Programming and Timing Performance

The sizes of the SEL main memory and of the AP data memory are insufficient to handle all the data required for using the above-described numerical method. In order to reduce the memory requirements of the three-dimensional Euler code, the computational domain is cut into a sequence of grid surfaces of index i. At each time level, the variables are successively computed on each of these surfaces. Only the data necessary for processing one of these surfaces (namely, the data relative to three consecutive surfaces) are brought into the central memory. The other data are stored on disk files. This conventional partitioning technique makes it possible to use a great number of grid points. The maximum number of grid points on each surface of index i is equal to 1000. The number of surfaces is limited only by the capacity of the disks. The treatment of each grid surface is made up of

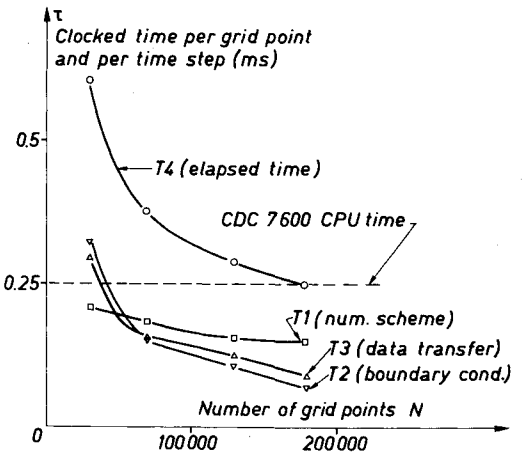

Fig. 10 Timing performance of the three-dimensional Euler code for a transonic flow computation in a nozzle.

four tasks, namely,

Task 1: transfer from the disk files to the AP data memory and to the SEL main memory of the data necessary for processing the surface of index i;

Task 2: application of the numerical scheme at all grid points on the surface of index i;

Task 3: correction of the values given by the numerical scheme at the boundary points on the surface of index i;

Task 4: writing on a disk file of the values which have just been computed.

The organization of the numerical code is represented in Fig. 8. Task 2 can be easily vectorized. The floating operations in this task are numerous, simple (most of them are additions or multiplications) and identical at all grid points. This task is executed in the array-processor by using hand-coded subroutines. On the contrary, the floating operations in task 3 are less numerous, complex (square roots, trigonometrical functions), and vary according to the nature of the boundary. This task is executed in the host computer by using FORTRAN subroutines which can be easily modified by the user.

The tasks relative to successive surfaces are chained together. The processing is organized as a "super-pipeline" where four stages can be defined in relation to the four above-mentioned tasks. The chaining of the tasks is explained in Fig. 9. The horizontal axis indicates the execution time. The cycle T is the execution time of the longest task. The squares represent the stages of the pipeline. In the middle of these squares is found the index of the surface treated. Data arriving at the first stage of the pipeline or moving within the pipeline are captured and held in the next stage at the end of each cycle T. This makes it possible to start the treatment of a new grid surface at each cycle, even though four cycles are required to complete the computations on a grid surface. When all stages of the pipeline are activated, four tasks on four consecutive sufaces are then in progress at the same time and a result emerges from the pipeline at the end of each cycle.

The first hand-coded program organized in this manner was developed to compute the flow in a nozzle. It will be considered a benchmark in the evaluation of AP processing possibilities.

In order to evaluate the processing performance and to detect any bottle-necks in the system, four different processing times have been measured according to the possibilities of the SEL executive system. They are the CPU time $T1$ in the array-processor; the CPU time $T2$ necessary for treating the boundary conditions in the host computer; the data transfer time $T3$ between the disks and the host memory (this time is measured with a clock system); and the elapsed time $T4$, which is the physical time necessary for executing the program.

The values per time step and per grid point of these four times are given in Fig. 10 for a transonic flow computation in a nozzle. These values show the advantage of placing the largest number of grid points in the computational array. The host overhead time and the disk access time are then minimized. The values of the three times $T1$, $T2$, and $T3$ are almost the same. This shows that a good balance exists between the three principal actions in the processing system: processing in the array processor, data transfer between the disks and the AP memory, and processing in the host computer. Except when the mesh is very coarse, the data transfer time is shorter than the CPU time in the array processor. According to Fig. 7, this confirms that the algorithm used is well suited to the architectural possibilities of the system (SEL + AP).

These processing times are compared with those obtained with the original scalar code on a CDC 7600 computer. When the number of grid points is sufficient, the CPU times $T1$ and $T2$ are shorter than the CPU time obtained on a CDC 7600 computer (Fig. 10). The elapsed time obtained on the system (SEL + AP) is roughly half that on a CDC 7600 computer.

In order to compute transonic flows in a rotating row of a fan, this code has been adapted while retaining the same above-defined organization (Fig. 9). The values of the processing times $T1$, $T2$, $T3$, and $T4$, for a grid point number of 16,500 are equal to 0.31, 0.24, 0.30, and 0.65 ms per grid point and per time step, respectively. These values are consistent with those represented in Fig. 10. As in the case of the nozzle, a better performance can be expected for higher grid point numbers.

V. Conclusion

In order to examine the possibilities of array processing in the field of aerodynamics, a system made up of a host computer SEL 32/77 and an array processor AP 120B has been set up. An explicit three-dimensional Euler code was selected and vectorized for evaluating the performance of this system. An execution speed very close to that of a CDC 7600 computer was obtained but with a lower cost. This confirms Taylor's ideas concerning the capacities of these computers.

Although the programming investment is greater than on a conventional computer, it can be made profitable by running the vectorized codes a large number of times. The excellent reliability of this system allows long runs, and thus opens up a new kind of processing: the dissemination of such systems to small groups of users having similar problems.

We would like to emphasize two areas of study which can be pursued with the help of such array processing systems. First, these systems, which are more flexible than conventional computers, could be used in experimental studies on parallel processing to test certain theories. For instance, the underlying data flow problem of multiprocessing could be studied with minimum hardware extensions, allowing one to detect the bottle-necks often not apparent in a theoretical project or a machine simulator. Second, such a system could be used to accustom the user to vectorization and parallelism, which will be the basis of computing in the near future. In this way, the algorithm's designer would learn to take into account the possibilities and the constraints of these new computers.

Acknowledgment

This work was performed with the financial support of DRET and STPA.

References

[1] Peterson, V. L., "Computational Aerodynamics and the Numerical Aerodynamics Simulation Facility," NASA CP 2032, Feb. 1978, pp. 5-30.

[2] Taylor, T. D. and Widhopf, G. F., "A Trend Toward Inexpensive Computer Simulations," *Astronautics & Aeronautics,* Vol. 17, April 1979, pp. 34-41.

[3] Viviand, H. and Veuillot, J. P., "Méthodes Pseudo-Instationnaires pour le Calcul d'Ecoulements Transsoniques," ONERA, Publication No. 1978-4, 1978; English translation ESA TT 561.

[4] Veuillot, J. P. and Viviand, H., "A Pseudo-Unsteady Method for the Computation of Transonic Potential Flows," AIAA Paper 78-1150, 1978.

[5] Enselme, M., Guiraud, D., and Boisseau, J. P., "Contribution à l'Evaluation de l'Intérêt de Calculateurs à Structure Parallèle pour la Résolution de Systèmes d'Equations aux Dérivées Partielles," ONERA TP 1980-160, Dec. 1980.

[6] Viviand, H., "Pseudo-Unsteady Methods for Transonic Flow Computations," *Lecture Notes in Physics,* Vol. 141, Springer-Verlag, Berlin, 1980, pp. 44-54.

[7] Brochet, J., "Calcul Numérique d'Ecoulements Internes Tridimensionnels Transsoniques," *Recherche Aérospatiale,* Vol. 5, Sept. 1980, pp. 301-315; English translation ESA TT 673.