ELSEVIER

Ecological Modelling 80 (1995) 69–85

ECOLOGICAL MODELLING

Using parallel computers in environmental modelling: a working example

R. Pastres *, Davide Franco, G. Pecenik, C. Solidoro, C. Dejak

University of Venice, Dept. of Physical Chemistry, Sect. of Environmental Physical Chemistry, Dorsoduro 2137, 30123 Venice, Italy

Received 15 April 1993; accepted 22 February 1994

Abstract

An application of the utilization of parallel supercomputers for a 3D eutrophication–diffusion macromodel of the Venice lagoon is presented.

Problems encountered in program restructuration, in the choice and in the introduction of parallel algorithms for solving the diffusion equation are discussed, together with the approach used to exploit multitasking performances.

Results obtained show that, through appropriate coding, execution times for a full year simulation of the model, involving the diffusion and the trophic interactions of eight state variables, with a time step of one hour, have been decreased by about an order of magnitude.

Keywords: Diffusion; Eutrophication; Lagoon ecosystems

1. Introduction

Ecological models represent a valuable tool for understanding, within a holistic synthesis, the behaviour of the multiplicity of biotic and abiotic, conservative and reacting components, and the linear and nonlinear interrelationships among bio-physico-chemical processes and advective–diffusive transport phenomena. To tackle such complexity and to comply with real world phenomena, ecological models must necessarily include biochemical, population dynamic, hydrodynamic and meteoclimatic submodels and be capable of reproducing the behaviour of relevant environmental variables at regime state conditions in a three-dimensional system.

Until recently, a bottleneck in designing models of such complexity was represented by the limited computational potentialities and storage capacity of available machines. The computational requirements of a model can be roughly estimated by multiplying the number of data handled at each time step of the main temporal loop by the number of time steps. For example, for a finite-difference model, the product can be computed by multiplying the number of grid points by the number of the state variables to update at each time step and by the number of time steps

* Corresponding author.

SSDI 0304-3800(94)00049-N

that are required for a meaningful simulation. Accordingly, a model may be defined a mega-model if this product is of the order of 10^6, giga-model if it is of the order of 10^9, and so on.

The computer generation of the seventies could easily deal with megamodels, but difficulties were met in coping with gigamodels, or bigger ones. The current trend in mathematical modelling is represented by the elaboration of tera-models, and a new generation of mainframes is on the way to satisfy the growing computational demand. In the last decade, computer technology has undergone an overwhelming upgrading, particularly with the arrival of the first vector and later, of parallel supercomputers. The possibility of exploiting these new machines in the field of environmental modelling was early recognized by several authors (Karplus 1978; Ginsberg, 1983; Benyon, 1985; Duff, 1985), who also pointed out the necessity of designing new algorithms for taking full advantage of their potentialities (Jordan, 1982).

2. Objectives of the work

To assess the impact of industrial and municipal pollution, as well as of thermal discharges in the Venice lagoon ecosystem, following a long-term interdisciplinary research, a combined, finite-difference, mathematical 3-D eutrophication–diffusion model was developed, comprising the area most suffering from pollution effects. The satisfactory results obtained so far confirmed the prospects of extending and improving the model descriptive potentialities by the introduction of additional state variables, as well as of more detailed formulations of physico-chemical and biological processes.

Such an aim may be achieved only by containing within acceptable limits the growing expenses deriving from the greater demand of computer memory and increase of computation time. Thus, the recoding of the program has been undertaken, directed at optimizing the performances on the supercomputers Cray of the series X-MP/12 and /48 and YMP, which have been installed at the Interuniversity Computational Center (CINECA, Casalecchio di Reno, Bologna) starting from 1986.

3. Main features of the model

The model covers an area of about 180 km^2, whose discretization, with a $\Delta x = 100$ m, $\Delta y = 100$ m and a vertical step $\Delta z = 1$ m, leads to a computational domain of $140 \times 128 \times 20 = 358\,400$ grid points, whereas the time step is $\Delta t = 1$ h. According to the terminology previously introduced, it can be classified between a gigamodel and a teramodel, since the $358\,400 \times 8 = 2\,867\,200$ data (grid points $\times$ number of state variables) must be followed for at least one year, i.e. 8760 time steps, yielding about $25.1 \cdot 10^9$ processed data.

On account of the fact that the hydrodynamic regime of the Venice lagoon is predominantly governed by tidal movement, the transport process of soluble and/or suspended material is described through an eddy-diffusion mechanism. This includes space- and time-constant eddy diffusivities, which were estimated and calibrated with the aid of a specifically elaborated 2D advection diffusion model (Dejak et al., 1987a), in order to reproduce appropriately tidal agitation.

Boundary conditions were derived through a second-order finite difference analysis of the logarithmic concentrations, i.e., by means of a delocalized Gauss function, which is the analytical solution of the diffusion equation under simple boundary and initial conditions. Wherever possible, the extrapolation of boundary values is performed using the values at three internal contiguous water cells adjacent to the boundary, but alternative formulations are required to deal with different bathymetric situations (Dejak et al., 1987b). System three dimensionality, together with the adopted non-linear open boundaries conditions enable the achievement of steady states, in presence of continuous sources of pollutants, making it possible to follow their succession in time up to attaining a regime state. Model seasonalization is effected by specifically elaborated submodels which, based on the quantification of energy balance at air–water interfaces,

perform the computation of the photoperiod throughout the seasons, while permitting the assignment of water temperature values at all grid points at each time step and the description of the evolution of the thermal stratification in the deepest channels (Dejak et al., 1992). The biolog-

Table 1
a. System of differential equations which describes the evolution of the state variables within each cell of the grid. Fluxes between cells are accounted for by the diffusion process

State variables:

$[F]$ = phytoplankton density $[\text{mg C l}^{-1}]$
$[Z]$ = zooplankton density $[\text{mg C l}^{-1}]$
$[NO_x^-]$ = concentration of nitrogen in oxidized form $[\text{mg N l}^{-1}]$
$[NH_4^+]$ = concentration of nitrogen in reduced form $[\text{mg N l}^{-1}]$
$[PO_4^{3-}]$ = concentration of reactive phosphorus $[\text{mg P l}^{-1}]$
[BOD] = concentration of organic matter which undergoes biochemical oxidation $[\text{mg O l}^{-1}]$
[DO] = concentration of dissolved oxygen $[\text{mg O l}^{-1}]$

Ordinary differential equations:

$$\frac{d[F]}{dt} = \{\beta - (K_{mf}\vartheta(T) + K_{sedf} + K_{rf}\vartheta(T))\}[F] - K_{grz}\vartheta(T)f([F])[Z] \quad (1)$$

$$\frac{d[Z]}{dt} = \{K_{grz}\vartheta(T)f([F])E_{ff} - K_{mz}\vartheta(T) - K_{escz}\vartheta(T)\}[Z] \quad (2)$$

$$\frac{d[NO_x^-]}{dt} = K_{nit}\vartheta(T)[NH_4^+] - R_{NC}\left(\frac{[NO_x^-]}{[N_{tot}]}\right)\beta[F] \quad (3)$$

$$\frac{d[NH_4^+]}{dt} = R_{NC}\{K_{rf}\vartheta(T)[F] + K_{escz}\vartheta(T)[Z]\} + K_{rsn} - K_{nit}\vartheta(T)[NH_4^+] - R_{NC}\left(\frac{[NH_4^+]}{[N_{tot}]}\right)\beta[F] \quad (4)$$

$$\frac{d[PO_4^{3-}]}{dt} = R_{PC}\{K_{rf}\vartheta(T)[F] + K_{escz}\vartheta(T)[Z]\} + K_{rsp} - R_{PC}\beta[F] \quad (5)$$

$$\frac{d[BOD]}{dt} = R_{OC}(K_{mf}\vartheta(T) + K_{sedf})[F] + R_{OC}(K_{mz}\vartheta(T)[Z]) + K_{dec}\vartheta(T)[BOD] \quad (6)$$

$$\frac{d[DO]}{dt} = K_{rear}\Delta_{DO} + R_{OC}(\beta - K_{rf}\vartheta(T))[F] - K_{dec}\vartheta(T)[BOD] - R_{ON}(K_{nit}\vartheta(T)[NH_4^+]) - K_{osed} \quad (7)$$

Functional relationship:

$\beta(T,N,P,I) = \mu(T)\, f([N_{tot}])\, f([P])\, f(I)$
where:
$\mu(T) = \mu_o[(T_a - T)/(T_a - T_o)]^{b(T_a - T_o)} \exp[b(T - T_o)]$ = maximum phytoplankton gross growth rate at temperature T;
$f(N_{tot}) = [N_{tot}]/(K_n + [N_{tot}])$ = limitation of phytoplankton growth due to nitrogen;
$f(P) = [P]/(K_p + [P])$ = limitation of phytoplankton growth due to phosphorus;
$f(I) = e/a_1\{\exp[-a_2\exp(-a_1)] - \exp(-a_2)\}$ = limitation of phytoplankton growth due to incident light intensity (I)
where:
$a_1 = K_h z$;
$a_2 = I/I_o$; z = depth;
$f(F) = [F]/(K_f + [F])$
$\vartheta(T) = 1.07^{(T-20)}$
$\Delta_{DO} = [DO_{sat}] - [DO]$ = Oxigen deficit
where:
$[DO_{sat}] = 14.6244 - 0.367134T + 0.0044972T^2 - 0.0966S + 0.00005TS + 0.0002739S^2$

Table 1 (continued)
b. Functional relationship between forcing functions (T,I), and kinetic parameters and list of parameter values which appear in Table 1a

Parameters:
$\mu_o = 0.12$ [h^{-1}] maximum specific growth rate of phytoplankton at optimal temperature T_o
$K_{nit} = 0.0023$ [h^{-1}] nitrification rate
$K_{dec} = 0.0048$ [h^{-1}] BOD decay rate
$K_n = 0.05$ [mg N l^{-1}] nitrogen half-saturation constant
$K_p = 0.01$ [mg P l^{-1}] phosphorus half-saturation constant
$K_{mf} = 0.005$ [h^{-1}] phytoplankton mortality rate
$K_{rf} = 0.004$ [h^{-1}] phytoplankton respiration rate
$K_{mz} = 0.005$ [h^{-1}] zooplankton mortality rate
$K_{escz} = 0.002$ [h^{-1}] zooplankton escretion rate
$K_{grz} = 0.05$ [h^{-1}] grazing zooplankton rate
$K_{fz} = 1$ [mg C l^{-1}] grazing half-saturation constant
$K_{sedf} = 0.004$ [h^{-1}] phytoplankton sedimentation rate
$K_{rsn} = 0.0007$ [mg N l^{-1} h^{-1}] rate of nitrogen release from the sediment
$K_{rsn} = 0.0001$ [mg N l^{-1} h^{-1}] rate of phosphorus release from the sediment
$K_{osed} = 0.001$ [mg DO l^{-1} h^{-1}] rate of sediment uptaken of oxygen
$K_{rear} = 0.045$ [h^{-1}] reareation constant
$E_{ff} = 0.7$ efficiency in biomass conversion from phytoplankton to zooplankton
$I_o = 50.000$ [lux] optimal light intensity for photosynthesis
$K_h = 0.4$ [m^{-1}] light shading coefficient
$T_o = 31$ [°C] optimal temperature for phytoplankton growth
$T_a = 35$ [°C] temperature at which phytoplankton growth stops
$b = 0.114$ [$°C^{-1}$] empirical coefficient
S = salinity of the lagoon (30‰ average value)

ical compartment includes seven chemical and biological state variables, considered as the most representative of the eutrophication process. Interrelationships among state variables were formulated after extensive in situ research (Bertonati et al., 1987) and are reported in Table 1a. The model involves for each state variable, the solution of the following second order partial differential equation:

$$\frac{\partial c_j}{\partial t} = \frac{\partial \Phi_{jx}}{\partial x} + \frac{\partial \Phi_{jy}}{\partial y} + \frac{\partial \Phi_{jz}}{\partial z} + \Lambda_j + \sum_k S_{kj} \qquad (1)$$

where c_j represents the concentration of the j_{th} variable, $\Phi_{jx} = \boldsymbol{K}\Gamma_j$ is the flux vector along the x direction, Γ_j being the spatial gradient and $\boldsymbol{K}$ a diffusivity tensor, Λ_j a set of differential equations interconnecting the state variables, and S_{kj}, the sinks and/or sources either instantaneous or continuous. If the diffusivities are supposed to be uncorrelated and constant in respect of time and space, the diffusivity tensor becomes diagonal:

$$\begin{pmatrix} \Phi_x \\ \Phi_y \\ \Phi_z \end{pmatrix} = \begin{pmatrix} K_{xx} & 0 & 0 \\ 0 & K_{yy} & 0 \\ 0 & 0 & K_{zz} \end{pmatrix} \begin{pmatrix} \Gamma_x \\ \Gamma_y \\ \Gamma_z \end{pmatrix} \qquad (2)$$

and Eq. 1 takes the more familiar form:

$$\frac{\partial c_j}{\partial t} = K_{xx}\frac{\partial^2 c_j}{\partial x^2} + K_{yy}\frac{\partial^2 c_j}{\partial y^2} + K_{zz}\frac{\partial^2 c_j}{\partial z^2} + \Lambda_j + \sum_k S_{kj}. \qquad (3)$$

4. Structure of the program

The program is structured according to the flow-chart of Fig. 1a, and is described in the caption. The numerical integration of the diffu-

sion terms of Eq. 3 is performed by applying the fractional step method, usually employed to deal with the diffusion along the three space coordinates separately: instead of solving numerically the global three-dimensional equation, the three contributions to the flux are computed subdividing the time step into three substeps and solving a one-dimensional diffusion equation at a time. The contribution to the change of concentration due to biochemical reactions inside each cell is then added.

The three one-dimensional diffusion equations are solved by adopting the implicit Laasonen (Laasonen, 1949) scheme, which proved to be the most suitable in dealing with continuous sources of pollutant, as it leads to the best agreement between the numerical solution and the convolution of the analytical one, when the latter can be found (Dejak et al., 1987a). As the same procedure is repeated for each horizontal or vertical layer, in the following discussion, attention will be focused on the integration of a single layer, which can be represented by a 2D matrix, and we will generally refer to a line or a column of the matrix as an integration segment. The discretization of the diffusion equation, in the simplest case of costant diffusivities and of close boundaries, leads to the following tridiagonal system

$$\begin{aligned}
(1+D)c_1^{t+\Delta t} - Dc_2^{t+\Delta t} &= c_1^t \\
-Dc_1^{t+\Delta t} + (1+2D)c_2^{t+\Delta t} - Dc_3^{t+\Delta t} &= c_2^t \\
\cdots \quad \cdots & \\
-Dc_{i-1}^{t+\Delta t} + (1+2D)c_i^{t+\Delta t} - Dc_{i+1}^{t+\Delta t} &= c_i^t \\
\cdots \quad \cdots & \\
-Dc_{n-1}^{t+\Delta t} + (1+D)c_n^{t+\Delta t} &= c_n^t
\end{aligned} \tag{4}$$

where D is the dimensionless diffusion number,

$$D = K_{xx}\Delta t/\Delta x^2 = K_{yy}\Delta t/\Delta y^2.$$

The system was solved using the recursive Thomas algorithm, considered as the most suitable for the scalar computers. Defining α_i, β_i, γ_i, respectively as the coefficients of the unknowns c_{i-1}, c_i and c_{i+1}, the method requires the following computational procedure:

forward elimination:

$$f_1 = \frac{\gamma_1}{\beta_1} \qquad f_1 = -\frac{D}{1+D} \tag{5a}$$

$$f_i = \frac{\gamma_i}{(\beta_i - \alpha_i f_{i-1})} \qquad f_i = -\frac{D}{(1+2D+Df_{i-1})} \tag{5b}$$

$$f_n = \frac{1}{(\beta_n - \alpha_n f_{n-1})} \qquad f_n = \frac{1}{(1+D+Df_{n-1})} \tag{5c}$$

$$g_1 = \frac{c_1^t f_1}{\gamma_1} \qquad g_1 = -\frac{c_1^t f_1}{D} \tag{5d}$$

$$g_i = \frac{(c_i^t - \alpha_i g_{i-1})f_i}{\gamma_i} \qquad g_i = -\frac{(c_i^t + Dg_{i-1})f_i}{D} \tag{5e}$$

$$g_n = (c_n^t - \alpha_n g_{n-1})f_n \qquad g_n = (c_n^t + Dg_{n-1})f_n \tag{5f}$$

and back substitution:

$$c_n^{t+\Delta t} = g_n \tag{6a}$$

$$c_i^{t+\Delta t} = g_i - f_i c_{i+1}^{t+\Delta t} \tag{6b}$$

$$c_1^{t+\Delta t} = g_1 - f_2 c_2^{t+\Delta t} \tag{6c}$$

Since the vector $\boldsymbol{f}$, with constant diffusion numbers, depends only on the values of the coefficients of the unknowns, it can be evaluated at start up time, while only the vector $\boldsymbol{g}$ and the solution vector $\boldsymbol{c}$ need to be computed at each time step, respectively requiring $3n$ and $2n$ operations. Along both horizontal directions, each integration segment is made up of a variable number of land points m, which interrupt the transport process. Nevertheless, it is still possible to utilize the same algorithm, but the choice of the appropriate β_1 and β_n for each segment depends upon the local configuration of the bathymetry and this leads to the computation of varying f_is.

In the scalar version of the program a complex procedure was implemented, effecting the checking of the bathymetry at each time step and at each grid point within each integration segment.

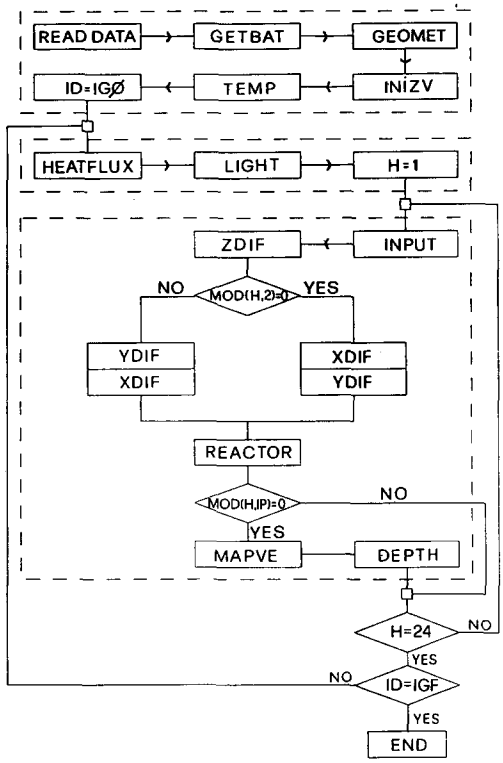

Fig. 1. Flow-chart of the seasonalized program in its present configuration. Subroutines GETBAT, GEOMET, INIZV, and TEMP are called before the main loop in time starts, for reading the bathymethry and performing the calculation of auxiliary vectors used during the integration of the diffusion equation. Subroutines HEATFLUX and LIGHT, called once a day, bring up to date the water temperature and calculate energy input for phytoplankton photosynthetic activity. The main body of the program is constituted by the three subroutines XDIF, YDIF, ZDIF, which perform the numerical integration of the diffusion equation, and REACTOR, which changes the concentration of state variables within each cell, according to the equation of Table 1. Subroutines MAPVE and DEPTH, give a first idea of the concentration pattern, printing out data regarding a particular layer or vertical section.

First, the occurrence of water or land was verified and then different values for f_i were chosen, according to the position occupied within a set of continuous water cells, or to the fact that the ith grid point was the first or the last water point. If m interruptions were found, the integration was performed by solving in succession $m + 1$ tridiagonal systems. Memory saving was achieved in this way, since the different values for f elements were rewritten over the same area, but this method led to implement highly nested conditional blocks, which made the computation rather cumbersome.

5. Optimization for Cray X-MP/12

The program, originally conceived to run on one of the fastest scalar computers available in the seventies (CDC 7600), was first optimized for the Cray X-MP/12, mostly to solve problems deriving from the computational domain, which was too large compared with the RAM memory resources. In fact, 2867000 memory locations were needed if the concentration values of the 8 state variables were associated with all the mesh points, regardless of their belonging to the waterbody. The lagoon being a shallow water basin, only about 30000 cells physically pertain to the modelled system, and constitute the real computational domain, giving an effective storage demand of 240000 locations. The concentration values were therefore associated with the elements of a vector, regardless of their physical location, and a $140 \times 128 \times 20$ matrix was constructed, which establishes a correspondence between the elements of the vector and their physical position inside the grid (see Fig. 2). This procedure, the so called indirect addressing, though permitting the maintenance of the memory request below the availability of RAM memory, affected the performance, as the gather-scatter operations, needed to read the state variable vectors, had to be carried out by an internal software routine.

Also, attempts were made to optimize the diffusion subroutines XDLAAV and YDLAAV, which take nearly 78% of the execution time, as shown in Table 3a, because of the non linearity of the open boundary conditions and of the abruptly changing lagoon bathymetry. The computation of the vertical diffusion is less cumbersome, since a simple reflective condition enables one to deal with the close boundaries. Earlier optimization efforts were mainly dedicated to removing all unvectorizable conditional blocks inside the main

loop by adequately coding the checking of the bathymetry and confining logical operations to the initialization phase. This aim was achieved by replacing each piece of code leading to alternative paths with a single general algebraic expression, containing several terms, which embodied all information associated with each grid point (i.e. water or land, segment lengths, distance of a water point from the nearest land point, closed or open boundaries). Switching on/off each term by means of multiplicative coefficients assuming the binary code 1 or 0, the correct values for each element of the auxiliary vector f were chosen. Through this substitution with more tractable algebraic operations, all nested conditional blocks could be eliminated, making it possible to fully vectorize both horizontal diffusion subroutines. After choosing the right values for the vectors f for a whole layer, the integration was carried out by means of vector operations, as the same step of the recursive algorithm can be computed simultaneously for all the systems belonging to the same layer. However, in spite of a fully achieved vectorization of the code, a consistent decrease of the execution time was yet to be obtained, as the overall performance was greatly affected by the continuous transfer of data to and from the main memory, a time-consuming operation on supercomputers (Dejak et al., 1988).

6. Optimization of the program for Cray X-MP/48

The availability, since 1988, of the more powerful Cray X-MP/48, 8 Mword shared memory and 4 CPUs, stimulated a general restructuration, which, again, started from the subroutines XDLAAV and YDLAAV. A different approach was undertaken, aimed at exploiting both the increased vector and parallel potentialities of the Cray X-MP 48 and its memory capacity. In fact, by recomputing values for α_i, β_i, γ_i, it becomes possible to associate an entire integration segment with only one system, having the maximum number $n_x = 128$ or $n_y = 140$ unknowns. The three coefficients uniquely define a value for f_i, so that, once they are assigned at each grid point and f_i is calculated, the simultaneous elimination of all logical blocks and of all shifting operations is achieved, greatly enhancing the program vectorizability.

The physical meaning of the procedure appears more clearly, for the general case of space-varying D_is (which had been already introduced along the vertical), by rewriting the ith equation in the modified form:

$$-D_i c_{i-1}^{t+\Delta t} + (1 + D_i + D_{i+1}) c_i^{t+\Delta t} - D_i c_{i+1}^{t+\Delta t} = c_i^t \tag{7a}$$

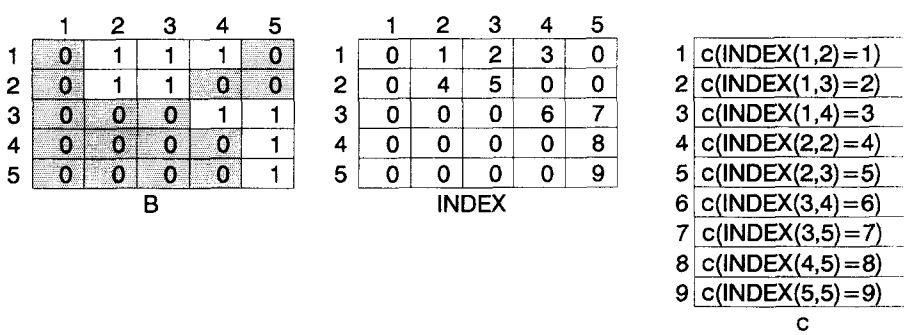

Fig. 2. A simple example of indirect addressing. A matrix $\boldsymbol{B}(i,j)$ is defined, whose elements $b_{i,j}$ can be either 0 or 1 if a land cell (shaded) or a water cell is found at the position i,j. A second matrix, $\boldsymbol{INDEX}(i,j)$, can be derived, whose elements are 0 if $b_{i,j}$ is 0, or a progressive number, increasing to unity any time that $b_{i,j}$ equals 1. These numbers will indicate the position of the cell i,j inside a vector $\boldsymbol{c}$, so that the matrix $\boldsymbol{INDEX}$ establishes a biunivocal correspondence between the coordinates i,j of any water cell and the elements of the vector: $b_{i,j} = 1 \leftrightarrow c(\boldsymbol{INDEX}_{i,j})$.

$$c_i^{t+\Delta t} + D_i\left(c_i^{t+\Delta t} - c_{i-1}^{t+\Delta t}\right) + D_{i+1}\left(c_i^{t+\Delta t} - c_{i+1}^{t+\Delta t}\right) = c_i^t \quad (7b)$$

Since the reflective condition is applied at physically closed boundaries, the two contributions to the flux along a given direction must vanish if a land cell is found at the ith position. Therefore, the diffusion number can be associated with the wall between two cells and set to 0 if either of them is a land cell, while it maintains its preassigned value for contiguous water cells. A matrix of appropriate diffusion numbers can be readily computed, checking the bathymetry at start up time for defining a matrix $\boldsymbol{B}(i,j,k)$, whose elements are equal to 1 if the cell belongs to the waterbody and equal to 0 otherwise. Following this procedure, the three coefficients for α_i, β_i, γ_i required for the ith equation along the x direction at the jth section of the kth layer, are calculated using the formulas:

$$\alpha_i = -D_{i-1}\left(B_{i-1,j,k}B_{i,j,k}\right) \quad (8a)$$

$$\beta_i = 1 + D_{i-1}\left(B_{i-1,j,k}B_{i,j,k}\right) + D_i\left(B_{i,j,k}B_{i+1,j,k}\right) \quad (8b)$$

$$\gamma_i = -D_i\left(B_{i,j,k}B_{i+1,j,k}\right) \quad (8c)$$

The coefficients so computed enable one to deal with all possible situations occurring within an integration segment as a consequence of the variation of the bathymetry. By assigning to $B_{n+1,j,k}$ or to $B_{0,j,k}$ the value 0, the case of closed boundary is included as well, being $\beta_1 = 1 + D_0(1 \cdot 0) + D_1(1 \cdot 1) = 1 + D_1$, and $\beta_n = 1 + D_{n-1}(1 \cdot 1) + D_n(1 \cdot 0) = 1 + D_n$. Furthermore, the procedure can easily be applied also to the case of space varied diffusion numbers, which may be estimated as a function of the different mean squared tidal velocities inside the water body (Dejak and Pecenik, 1991). After computing the coefficients for the whole grid, the vectors f are calculated for each segment along both directions and stored in two 3D matrices, which contain all the information sufficient to correctly deal with the bathymetry. In this way, with the exception of the reading of these matrices, no additional operations are needed during the main loop in time.

The solution of the set of m constant size tridiagonal systems, associated with the segments belonging to the same layer, is carried on by again applying the same recursive algorithm but in parallel, that is, computing the same step of the algorithm for the whole system by means of a single vector operation. The maintenance of this procedure is preferred, instead of solving sequentially the m systems using other proposed vectorial algorithms (Traub, 1973; Kershaw, 1982), because the ratio m/n is nearly 1 in the horizontal plane. In such instances, as demonstrated by Hockney and Jesshope (1988), it is more advantageous to keep adopting a recursive algorithm, which, besides guaranteeing a complete vectorization, presents the minimum number of operations and a lower memory demand for storing the auxiliary functions.

The solving of a set of tridiagonal systems with a constant number of unknowns, even when the segments comprise land cells, may, at first sight, appear a superfluous operation, and also, the new procedure seems to increase considerably the memory demand. As far as the first objection is concerned, the adoption of this method is justified by the fact that the minimization of both logical and input operations inside the main temporal loop allow one to fully exploit the vector potentialities of the machine, as will be shown in the result and discussion section. As regards the second point, a great saving is achieved by storing the auxiliary functions in three vectors having the same size of effective computational domain and using indirect addressing: in fact the auxiliary function value f_i is constantly 0 for any land cell. The extensive use of indirect addressing does not affect the performance on Cray X-MP48, as they are carried on by specifically designed hardware processors.

6.1. Vertical structuration of the horizontal diffusion algorithm

The vertical diffusion subroutine ZDLAAV was restructured, following a procedure analogous to the one illustrated above. The absence of open boundaries only greatly facilitates the task: the tridiagonal systems, each simulating the diffu-

sion along a water column, were regrouped in vertical sections of the computational grid; for each section, the maximum depth was determined and only the concentration values of water columns deeper than 1 m were transferred to a 2D matrix and elaborated as discussed above.

The Venice lagoon is a shallow water basin, with only major navigation channel deeper than 4 m: below this depth, grid points associated with land are far more abundant than those associated with water and it frequently happens that no contiguous water cells are met within a given segment. Thus, the number of systems to be solved decreases with depth, as can be seen in Fig. 3, and an unnecessary waste of time would derive from a straightforward application of the above procedure. To avoid this, at each layer the integration segments containing at least two consecutive water elements are counted and their position inside the layer is memorized. By repeating the procedure for all layers and for both directions of integration, two 128×20 and 140×20 2D matrices are defined, respectively associated with the X and Y axes, containing numerical flags, 1 or 0, which verify whether integration is to be executed along a specified segment. These operations are again performed at start up time, outside the main time loop: inside it, at each time step, for each layer, only the concentration values located in the segments which have to be brought up to date are read from the main memory, temporarily stored in an auxiliary 2D matrix and elaborated. In this way, the integration is carried out, starting from the surface layer and proceeding to the bottom, by gradually reducing the dimension of the 2D matrix, as the depth increases.

6.2. Open boundaries

Program restructuration was also undertaken in respect of the purposely elaborated conditions on the "open boundaries". By adopting the terminology introduced by Song et al. (1986), the algorithm here adopted, although "independent", may be classified as "unhomogeneous" according to whether the integration segments present either closed or open ends. Three alternative formulations are adopted for extrapolating the boundary value c_0, based on a second and first difference analysis of the values c_1, c_2, c_3 of the state variable in three cells adjacent the bound-

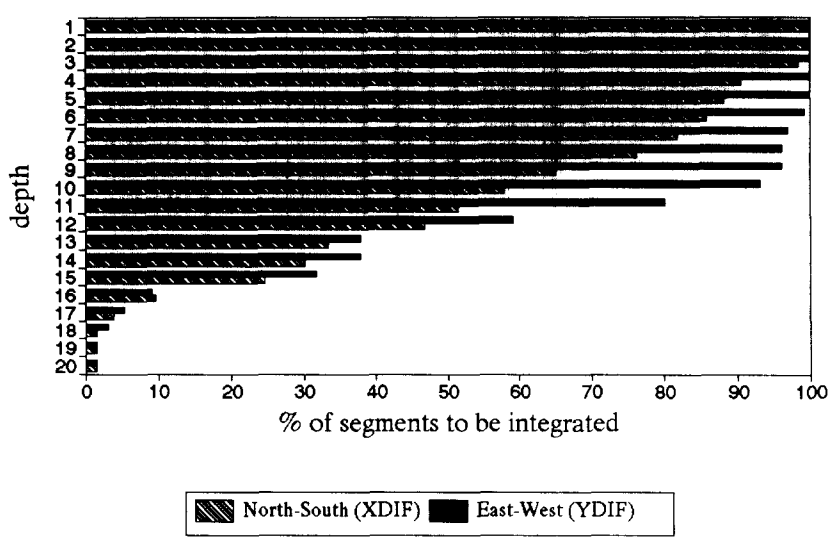

Fig. 3. Percentage of segments to be brought up-to-date, and, therefore, of the tridiagonal system to be solved at each time step along the two horizontal directions, as a function of depth. A marked decrease occurs, especially for the North–South direction (X axis), below the tenth layer, which suggested modification of the integration procedure, inserting in a 2-D matrix of variable size only those segments which are affected by the transport process.

ary. A Gaussian behaviour of the solution is assumed when three points belong to the computational domain (Eq. 9a), an exponential decrease if only two points are available (Eq. 9b), and, finally, a closed boundary (Eq. 9c) is simulated when any extrapolation is impossible.

$(\Delta_2 = 0)$

$$c_0 = c_3(c_1/c_2)^3$$

$$\ln c_0 - 3 \ln c_1 + 3 \ln c_2 - \ln c_3 = 0 \quad (9a)$$

$(\Delta_1 = 0)$

$$c_0 = c_1(c_1/c_2)$$

$$\ln c_0 - 2 \ln c_1 + \ln c_2 = 0 \quad (9b)$$

$$c_0 = c_1$$

$$\ln c_0 - \ln c_1 = 0 \quad (9c)$$

The extrapolated value cannot be directly introduced in the implicit integration scheme adopted, without refining it up to convergence, within a preassigned error. In fact, extrapolation is based upon the concentration distribution at the past time, t, and it becomes a first "guess" for either the first or the last unknown of the linear system (Eqs. 5). In order to avoid unrealistic outward fluxes, the extrapolation is repeated after solving the system and the new value c_o is compared with the old one: if the difference is greater than a preassigned error additional refining iterations are performed up to convergence, using the new c_o and c^t again. In the previous version of the program, the choice of the correct form of Eqs. 10 was made in the initialization phase, by storing in four matrices the logical variables which characterize the physical features of each boundary. Thus, the following general formulation has been introduced to replace Eqs. 9a, 9b, 9c:

$$c_0 = \left\{ \left[\min\left(\frac{c_1}{c_2}, \frac{c_2}{c_1} \right) \right]^k c_k \right\} h. \quad (10)$$

Because of the symmetry of the expression, it is sufficient to store the exponent of the ratio c_1/c_2, since it coincides with the second term of the product, so that $k = 3$ and $k = 1$ correspond to the presence respectively of three points and two points. The internal function $\min[(c_1/c_2), (c_2/c_1)]$ assures a smooth extrapolation whenever a random fluctuation, originated by fluxes along other directions, occurs in the concentration pattern, avoiding the introduction of further logical conditions. Values for k are computed for both ends of each segment along each direction and stored in four 2D matrices. The discrimination of the type of boundary is instead given by the value taken by the corresponding element of the matrix $\boldsymbol{B}$: $h = 0$ (land) is a closed boundary; $h = 1$ (water) is an open boundary. It must be remarked that the condition expressed by Eq. 9c is a reflective one: therefore, instead of introducing a logical condition, h can be set equal to 0 whenever the last point is isolated from the water body along the direction perpendicular to the boundary, including in the general formulation all possible cases, without introducing logical instructions explicitly.

Non linearity of boundary conditions, on the contrary, affects the computation time more heavily, since it demands inner loops at each time steps, whose number is not a priori predictable, because the convergence depend upon the time-varying distribution of concentrations within each segment. To reduce execution time due to this procedure, after solving the systems first, only those segments which require further refinement are replaced in the auxiliary 2D matrix used for handling the calculations for each layer. The number of systems to be solved gradually decreases, and the internal loop stops when all the extrapolated values are coherent with the new concentration patterns.

7. Parallellism on Cray systems: macro-tasking and micro-tasking

Besides being particularly suitable to be vectorized, the program presents in its main body a true parallel structure. Within each diffusion subroutine, the computation is carried on "independently" not only with regard to each integration segment, but also to each layer. These conditions greatly favour the exploitation of the four CPUs available on the CRAY X-MP/48. Such a possibility is referred to as "multitasking" and it

Table 2

a. Flowtrace of a test version of the original program (1 state variable, and no trophic interactions) obtained with Cray X-MP12, in dedicated mode for a 48-h simulation

Routine	Running time (s)	Percent of running time	Number of calls
MODELC	0.001	(0.00%)	1
GEOMET	1.697	(3.45%)	1
GETBAT	0.142	(0.29%)	1
INIZV	n.r.	(0.00%)	1
XDLAAV	29.429	(59.87%)	48
YDLAAV	12.294	(25.01%)	48
ZDLAAV	5.218	(10.62%)	48
MAPVE	0.275	(0.56%)	4
DEPTH	0.094	(0.19%)	4
Total	49.151	(100%)	156

b. Flowtrace of a test version of the original program (1 state variable, and no trophic interactions) obtained with Cray X-MP48, in dedicated mode for a 48-h simulation

Routine	Running time (s)	Percent of running time	Number of calls	Differences with Table 2a	
				(s)	(%)
MODELC	0.004	(0.00%)	1	+0.003	+300.0
GEOMET	1.685	(5.19%)	1	−0.012	−0.7
GETBAT	0.135	(0.42%)	1	−0.007	−4.9
INIZV	n.r.	(0.00%)	1	n.r	n.r.
XDLAAV	14.866	(45.82%)	48	−14.563	−49.5
YDLAAV	10.875	(33.52%)	48	−1.419	−11.5
ZDLAAV	4.493	(13.85%)	48	−0.725	−13.9
MAPVE	0.293	(0.90%)	4	+0.018	+6.1
DEPTH	0.092	(0.28%)	4	−0.002	−2.1
Total	32.444	(100%)	156	−16.707	−34.0

c. Flowtrace of a test version of the program (1 state variable, and no trophic interactions) after restructuring the subroutine which account for diffusion on the horizontal plane (XDIF, YDIF), obtained with Cray X-MP48, in dedicated mode for a 48-h simulation

Routine	Running time (s)	Percent of running time	Number of calls	Differences with Table 2b	
				(s)	(%)
MODELN	0.076	(0.57%)	1	+0.072	+1800
GEOMET	0.104	(0.78%)	1	−1.581	−93.8
GETBAT	0.135	(1.01%)	1	0.0	0.0
INIZV	0.530	(3.98%)	1	0.0	0.0
XDIF	4.887	(36.67%)	48	−9.979	−67.1
YDIF	2.794	(20.97%)	48	−8.081	−74.3
ZDLAAV	4.404	(33.05%)	48	−0.089	−2.0
MAPVE	0.303	(2.27%)	4	+0.010	−3.4
DEPTH	0.092	(0.69%)	4	0.0	0.0
Total	13.325	(100%)	156	−19.119	−58.9

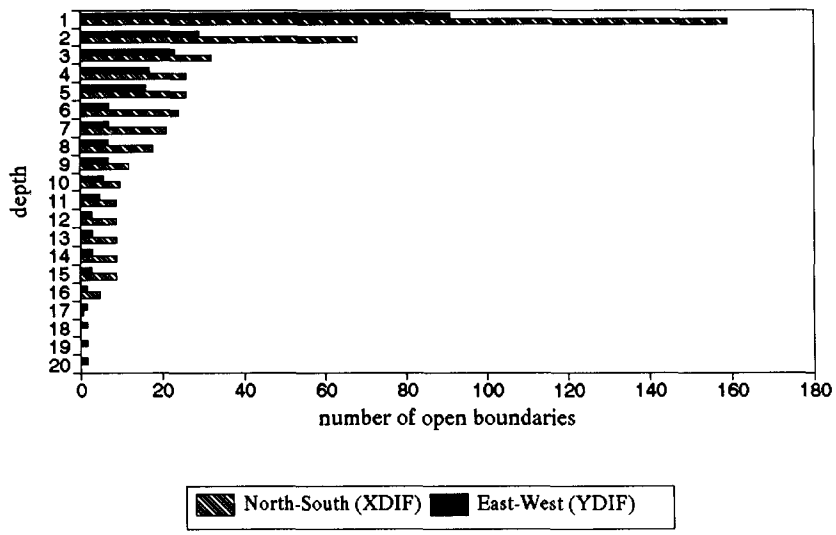

Fig. 4. Number of open boundaries along the two horizontal directions, as a function of depth. The great difference between the two directions explains the different results reported in Table 2a and 2b. Cray X-MP48 can perform gather–scatter operations, necessary to deal with the open boundaries, by means of hardware facilities, when, on Cray X-MP12, the same operations had to be carried out by internal subroutines. As a consequence of this hardware improvement, the execution time of XDLAAV is reduced by 50%, while the one of subroutine YDLAAV is reduced by only 12%.

may include macro- and microtasking according to whether independent tasks are carried out at a subroutine or at a DO loops level (CRAY, 1986). An example can be given: when employing macrotasking, the integration of the transport equation for different state variables could be concurrently carried on by the four CPUs, while the microtasking environment would allow one to

Table 3
a. Flowtrace of the original program, with eight state variables which undergo diffusion and biochemical processes, run on Cray X-MP48, in dedicated mode. Linear extrapolation from this 48 hours simulation, gives an expected running time of about 16h and 40 min for simulating one year

Routine	Running time (s)	Percent of running time	Number of calls
EUVELAV	0.058	(0.02%)	1
HEATFLUX	0.001	(0.00)	1
FOURIER	0.029	(0.01)	6
GEOMET	1.680	(0.51%)	1
GETBAT	0.140	(0.04%)	1
INIZV	n.r.	(0.00%)	1
LIGHT	n.r.	(0.00%)	2
MAPVE	1.669	(0.50%)	32
REACTOR	28.423	(8.59%)	48
TEMP	0.012	(0.01%)	1
TZDLAAV	0.009	(0.01%)	456
XDLAAV	123.401	(37.31%)	48
YDLAAV	134.599	(40.69%)	48
ZDLAAV	40.737	(12.32%)	48
Total	330.762	(100%)	695

b. Flowtrace of the program, with eight state variables, which undergo diffusion and biochemical processes, after restructurating the subroutines which account for the diffusion, run, for a 48-h simulation, on Cray X-MP48, in dedicated mode. Linear extrapolation from this 48-h simulation, yields 5 h and 30 min for simulating 1 year: running time was decreased by roughly two thirds

Routine	Running time (s)	Percent of running time	Number of calls	Differences with Table 3a	
				(s)	(%)
CRAY8	0.156	(0.14%)	1	0.0	0.0
HEATFLUX	0.001	(0.00)	1	0.0	0.0
FOURIER	0.337	(0.31)	6	+0.308	+1062.1
GEOMET	0.106	(0.10%)	1	−1.574	−93.7
GETBAT	0.135	(0.13%)	1	−0.005	−3.6
INIZV	0.552	(0.50%)	1	0.0	0.0
LIGHT	n.r.	(0.00%)	2	0.0	0.0
MAPVE	2.056	(1.88%)	32	+0.387	+23.2
REACTOR	28.532	(25.99%)	48	+0.109	+0.4
TEMP	0.014	(0.01%)	1	+0.002	+16.7
TZDLAAV	0.009	(0.01%)	456	0.0	0.0
XDIF	39.553	(36.03%)	48	−83.848	+67.9
YDIF	28.854	(26.28%)	48	−105.475	−78.6
ZDIF	9.510	(8.66%)	48	−31.227	−76.7
Total	109.775	(100%)	695	−220.987	−66.8

deal at the same time with the solution of the tridiagonal system pertaining to different layers, but concerning the same variable.

In our particular case, microtasking clearly appears to be the preferable choice, because the program granularity is rather fine and the submitted tasks prove to be well balanced. Besides, the subroutines XDIF, YDIF and ZDIF were already organized in DO loops, by which the same calculations are repeated for each layer or vertical section, so that no change in the structure is required. Microtasking also permits a more im-

Table 4
Flowtrace of the new program, with eight state variables, run on Y-MP432, for a 48-h simulation, using one CPU. Linear extrapolation from these data, gives an expected CPU time of about 4 h and 2 min for a yearly simulation

Routine	Running time (s)	Percent of running time	Number of calls	Differences with Table 3b	
				(s)	(%)
CONTORNI	0.277	(0.33%)	48	0.0	0.0
COST	n.r.	(0.0%)	1	0.0	0.0
HEATFLUX	5.546	(6.49%)	26328	+5.545	n.r.
FOURIER	0.069	(0.08%)	6	−0.268	−79.6
GEOMET	0.085	(0.10%)	1	−0.021	−19.8
GETBAT	0.084	(0.10%)	1	−0.051	−37.8
INIZV	0.313	(0.37%)	1	−0.239	−43.3
LIGHT	0.043	(0.05%)	1097	0.0	0.0
MAPVE	1.036	(1.21%)	32	−1.020	−44.6
NEWMODEL	0.034	(0.04%)	1	0.0	0.0
REACTOR	33.516	(39.20%)	48	+4.984	+17.5
TEMP	0.042	(0.05%)	1	+0.028	+233.3
XDIF	23.486	(27.47%)	48	−16.067	−40.6
YDIF	16.497	(19.29%)	48	−12.357	−42.8
ZDIF	4.459	(5.22%)	48	−5.051	−53.1
Total	85.487	(100%)	277717	−24.288	−22.1

mediate and easier synchronization of the different jobs, and, consequently, a faster and less expensive execution. Also less difficulties are encountered by the programmer using this environment.

In 1990 the more powerful Y-MP/432 became available at the CINECA, but, because of the structural similarity with the Cray X-MP/48, further essential modifications were not required for exploiting the new machine at its best. The main difference is, in fact, a 20% shorter clock period, which leads to roughly the same decrease in terms of execution time.

Table 5
CPU time and elapsed time obtained using all four CPUs of Cray Y-MP432. Automatic parallelization does yet not provide a good performance when complex programs have to be dealt with: elapsed time has been decreased by 9% of that obtained using 1 CPU (see Table 4a). Instead, an appropriate use of microtasking facilities makes it possible to achieve a reduction of about 33%. Reported times refer to a 48-h simulation: linear extrapolation from these data yields an expected elapsed time of about 1 h and 45 min for a yearly simulation

	CPU Time (sec)	Elapsed time (sec)	P_s	F_p (%)
Automatic parallelization	310.363	78.233	1.093	11.3
Microtasking	96.613	28.667	2.982	88.5

8. Optimization results

8.1 Vectorization

Trial runs were performed, using an incomplete version of the model, momentarily neglecting subroutines implementing model seasonalization (subroutines FOURIER, HEATFLUX, LIGHT, TEMP) and all biochemical computation (subroutine REACTOR) and considering only the diffusion of one conservative variable. The new diffusion subroutines were named XDIF, YDIF and ZDIF. All runs include a 48-h test simulation, to guarantee homogeneity in interpreting the results. Times are always reported in seconds and a random fluctuation of 10% around their values may occur for different runs, depending on the working condition of the machine.

Comparison of Table 2a and 2b shows the advantages automatically provided by the more powerful hardware of the Cray X-MP/48: the overall reduction in running time is roughly 34%, due partly to the shorter clock-period and partly to the fact that gather/scatter operations are carried out much more quickly by the specifically designed hardware. The latter improvement explains the much greater decrease, nearly 50%, in running time for the subroutine XDLAAV, compared with an average decrease of about 12% for the subroutines YDLAAV and ZDLAAV: the greater number of open boundaries along the S–N direction, see Fig. 4, demands a greater amount of inner iterations, which involve gather/scatter operations.

The saving obtained by carefully restructuring the program clearly appears from the data of Table 2c: the time spent in the two restructured subprograms, XDIF and YDIF, drops from about 25.7 s to 7.7 s, which is a reduction of roughly two thirds.

Trial runs regarding the whole seasonalized program confirmed the results previously obtained, as the comparison between Table 3a and 3b shows. The restructured program runs about 3 times faster than the program optimized for the CRAY X-MP/12, the estimated running time for a yearly simulation being only about 5 h and 30 min. As has been mentioned, the Cray X-MP/48 was substituted with the new model Cray Y-MP/432. As expected, a reduction of the execution time of about 20% was achieved, as one can see comparing the data of Table 4, from which it appears that a full year simulation takes about 4 h 2 min, and Table 3b.

8.2 Parallelization

The results achieved with the optimization can be evaluated by computing the parallel speed-up (P_s), that is, the ratio between the elapsed time obtained using 1 CPU and the one activating all 4 CPUs, and adopting the empirical Amdhal's law (Eq. 12), (Hockney and Jesshope, 1988) for esti-

mating the fraction of the program actually executed in parallel by the four CPUs:

$$P_s = \left(F_s + \sum_{p=1,N} \frac{F_p}{m} + O_p \right)^{-1} \quad (11)$$

with: P_s = Parallel speed-up; F_s = Fraction of instructions executed sequentially; F_p = Fraction of instructions executed concurrently for each task p; m = number of processors (CPUs); N = number of tasks; O_p = overhead due to internal routines which control the parallel execution.

Disregarding the overhead, which nevertheless leads to an underestimation of the fraction of the code actually parallelized, and considering the whole program as a unique task, one gets:

$$P_s = \left(F_s + \frac{F_p}{m} \right)^{-1} \quad (12)$$

which can be solved explicitly for F_p, since $F_p + F_s = 1$.

$$F_p = \frac{m(P_s - 1)}{(m-1)P_s}. \quad (13)$$

Table 5 summarizes the above calculation and compares the performances obtained by applying the parallelization automatically provided by the compiler and the one acquired by self-managing microtasking routines. The compiler does not seem to cope with the complexity of the program, as the parallel speed up is 1.09, which means that only 11% of the code is executed simultaneously by the four CPUs. On the contrary, a speed up of 2.98 is obtained using microtasking routines, which indicates that at least 88% of the code has been efficiently parallelized.

Table 6
The table summarizes the effects of both hardware and software improvement on the time request for a yearly simulation of the macromodel. In the upper part performances reffer to diffusive programs only, in the middle part, performances are reported of seasonalized eutrophic–diffusive programs and in the lower one the results achieved exploiting parallel facilities are shown. Following a row, one can see the effect of the hardware improvements, as the data are obtained runnig the same program on different machines. Benefits achieved through the optimization of the code are instead illustrated by the decrease of running time along the columns of the table

One state variable 1 CPU	CDC 7600	Δ	Cray X-MP/12	Δ	Cray X-MP/48
Scalar	60.9	(−92%)	4.7		
			(47%)		
Vectorized I version			2.5	(6%)	1.6
					(−56%)
Vectorized II version					0.7
Eight state variables					
Scalar	511.56	−92%	39.62	(−40.11%)	23.73
			(−30.7%)		(−29.2%)
Vectorized I version			27.46	(−38.8%)	16.8
					(−67.2%)
Vectorized II version					5.5
Eight state variables Dedicated mode	Cray X-MP/48 1 CPU	Δ	Cray X-MP/12 4 CPU	Δ	Cray Y-MP/432 4 CPU
CPU time	5.5		8.0		4.9
	(+3%)		(−72.5%)		(−74%)
Elapsed time	5.7	(−61%)	2.2 [a]	(−21%)	1.75
CPU time/4	1.375		2.0		1.225

[a] Time estimated from running a test program, in which the eight state variables were not interconnected by the routine REACTOR.

9. Conclusion and discussion

The increase in total CPU time observed when exploiting parallelism (first column in Table 5) in respect of the CPU time obtained using only one CPU, should not surprise, since a P_s equal to the number of CPUs, 4 in our case, is purely theoretical. In fact, some tasks are intrinsically non-parallelizable, for example output operations, and they are necessarily carried out by one CPU, while the remaining three are waiting. Also unavoidable is the so called overhead, that is, the amount of time used up by the internal routines which manage the parallel facilities. In view of that, a P_s of at least 88% must be considered as satisfactory.

The effects of both hardware and software improvement, over a decade, on the running time of the macromodel are summarized in Table 6, which reports the estimated time for a yearly simulation, in hours, on different machines and for different versions of the program. Data were obtained from test simulations of usually two days and then extrapolated for the year, after subtracting the time used up by the inizialization subroutines. In the upper part, performances are referred to *diffusive programs* only, which served as a test program for checking both the correctness of the output and the effectiveness of the restructuration. Times for the *seasonalized eutrophic–diffusive programs*, obtained with different machines but using always one CPU, are reported in the middle part, while the lower part compares the results obtained using *parallel facilities*. Following a row, one can see the effect of the hardware improvements, as the data are obtained running the same program on different machines. In fact, the introduction of the vector processors caused a decrease of running time of about an order of magnitude (92%), making it feasible to perform a yearly simulation of the complete program, as less than two days were required, while about 21 days were necessary on CDC 7600.

The necessity of careful programming for an optimal exploitation of vector and parallel machines, at least for complex programs, clearly appears if one follows the data along a *column*, referring to differently structured programs run on the same machine. For example, an overall reduction of 76.8% has been obtained through the two recodifications, allowing one to run the program on Cray X-MP/48 in 5.5 h instead of in approximately a day, with obvious benefits for the cost of simulations. Furthermore, as one can see from the lower part of the table, an optimization aimed at exploiting vector processors is usually helpful in locating the different tasks which can be carried out concurrently if a parallel machine is available. In fact, the final version of the program proved to be particularly suitable to be parallelized, leading to an estimated execution time of only 1.75 h for a yearly simulation. These drastic reductions not only present economic benefits, but, more important, they offer concrete possibilities of introducing additional state variables and a more detailed formulation of their interactions, so enhancing the descriptive capabilities of the model.

Acknowledgments

The authors wish to thank Mr. Sergio Manzi for the technical support given during the earliest elaborations, the CINECA (Casalecchio di Reno, Bologna) for the favourable conditions offered for the trial runs of the programs and Dr. G. Erbacci, for his helpful suggestions. They would also like to thank Mrs. H.M. Maguire, for carefully editing the manuscript. The work has been funded by the Progetto Finalizzato Laguna di Venezia of the National Research Council (CNR).

References

CRAY Research Inc., 1986. Multitasking User Guide. CRAY Computer Systems Technical Note. Pub. No. SN-0222, Mendota Heights, MN.

Benyon, P.R., 1985. Exploiting vector computer for simulation. Math. Comput. Simul., 27: 121–127.

Bertonati, M., Dejak, C., Mazzei, I. and Pecenik, G., 1987. Eutrophication model of the Venice lagoon: statistical treatment of "in situ" measurements of phytoplankton growth parameters. Ecol. Model., 37: 103–130.

Dejak, C. and Pecenik, G., 1991. A physico-chemical approach to modelling transport process: an application to perturbed waterbodies. In: C. Rossi and E. Tiezzi (Editors), Ecological Physical Chemistry, Proc. of International

Workshop 8–12 Nov. 1990, Siena, Italy. Elsevier, Amsterdam, pp. 410–418.

Dejak, C., Mazzei Lalatta, I., Molin, M. and Pecenik, G., 1987a. Tidal three-dimensional diffusion model of the lagoon of Venice and reliability conditions for its numerical integration. Ecol. Model., 37: 81–101.

Dejak, C., Mazzei Lalatta, I., Messina, E. and Pecenik, G., 1987b. A two-dimensional diffusion model of the Venice Lagoon and relative open boundaries conditions. Ecol. Modelling, 37: 21–45.

Dejak, C., Pecenik, G., Pastres, R. and Zane, E., 1988. Optimizing an eutrophication-diffusion 3-D macromodel for the CRAY X-MP/12. Environ. Software, 3(4): 56–64.

Dejak, C., Franco, D., Pastres, R. and Pecenik, G., 1992. Thermal exchanges at air-water interfacies and reproduction oftemperature vertical profiles in water columns. J. Mar. Syst., 3: 465–476.

Duff, I.S., 1985. The use of supercomputer in Europe. Comput. Phys. Commun., 37: 15–25.

Ginsberg, M., 1983. Some observations on supercomputer computational environments. In: M. Ruschitzka et al. (Editors), Parallel and Large-scale Computers: Performance, Architecture, Application. North Holland Publishing, Amsterdam, pp. 173–184.

Hockney, R.W. and Jesshope, C.R., 1988. Parallel Computers 2nd edition. Adam Hilger, Bristol, UK.

Jordan, T.L., 1982. A guide to parallel computation and some CRAY-1 experiences. In: G. Rodrigue (Editor), Parallel Computations. Academic Press, New York, pp. 1–50.

Karplus, W.J., 1978. The impact of new computer architectures on the simulation of environmental system. In: Vansteenkiste (Editor), Modelling Identification and Control in Environmental Systems. North-Holland Publishing, Amsterdam, pp. 1002–1009.

Kershaw, D., 1982. Solution of single tridiagonal systems and vectorization of the ICCG algorithm on the CRAY-1. In: G. Rodrigue (Editor), Parallel Computations. Academic Press, New York, pp. 85–99.

Laasonen, P., 1949. Über eine Methode zur Lösung der Warmeleitungsgleichung. Acta Math., 81: 309–314.

Song, J.L., Pielke, R.A. and Segal, M., 1986. Vectorizing a mesoscale meteorological model on the Cyber 205. Environ. Software, 1: 10–16.

Traub, J.F., 1973. Iterative solution of tridiagonal systems on parallel or vector computer. In: J.F. Traub (Editor), Complexity of Sequential and Parallel Algorithms. Academic Press, London, pp. 49–82.