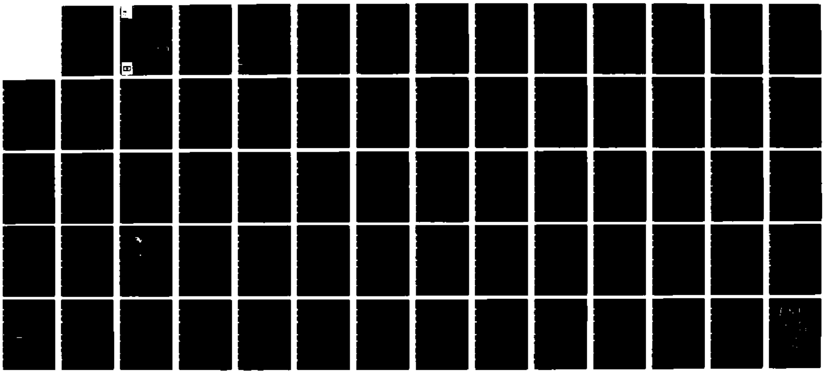
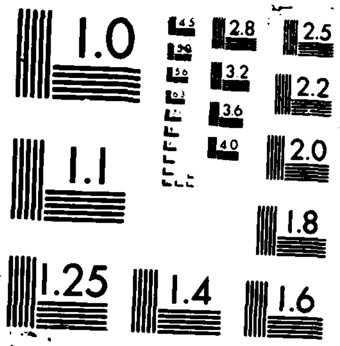
AD-A185 499 METHODS FOR REDUCING COMPUTATIONAL COSTS OF TYPICAL FINITE ELEMENT UNSTEA (U) ARMY ENGINEER WATERWAYS EXPERIMENT STATION VICKSBURG MS HYDRA.. 1/1

UNCLASSIFIED A J BAKER ET AL. SEP 87 WES/MP/HL-87-5 F/G 12/1 NL





ROCOPY RESOLUTION TEST CHART

DTIC FILE COPY

(2)

US Army Corps of Engineers

AD-A185 499

MISCELLANEOUS PAPER HL-87-5

METHODS FOR REDUCING COMPUTATIONAL COSTS OF TYPICAL FINITE ELEMENT UNSTEADY HYDRODYNAMIC MODELS

by

A. J. Baker, P. D. Manhardt

Computational Mechanics Corporation
3601 A Chapman Highway
Knoxville, Tennessee 37920

and

B. H. Johnson

Hydraulics Laboratory

DEPARTMENT OF THE ARMY
Waterways Experiment Station, Corps of Engineers
PO Box 631, Vicksburg, Mississippi 39180-0631

September 1987

Final Report

HYDRAULICS LABORATORY

Prepared for DEPARTMENT OF THE ARMY
Assistant Secretary of the Army (R&D)
Washington, DC 20310
Under ILIR Project No. 4A061101A91D

87 10 6 045

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) Miscellaneous Paper HL-87-5	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION See Reverse; 6b. OFFICE SYMBOL (If applicable) WESHR-M	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) See Reverse	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Assistant Secretary of the Army (R&D); 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Washington, DC 20310	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. See Reverse / TASK NO. / WORK UNIT ACCESSION NO.

11 TITLE (Include Security Classification)
Methods for Reducing Computational Costs of Typical Finite Element Unsteady Hydrodynamic Models

12. PERSONAL AUTHOR(S)
Baker, A. J.; Johnson, B. H.; Manhardt, P. D.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final Report	FROM ________ TO ______	September 1987	67

16. SUPPLEMENTARY NOTATION
Available from National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Computational; Hydrodynamics; Finite element; Numerical; Fluid dynamics

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Most three-dimensional finite element unsteady hydrodynamic models require excessive computer times even for short-term simulations on relatively sparse grids. The primary reason lies in the manner by which the resulting system of algebraic equations is solved. Historically, finite element modelers have developed codes for use on completely unstructured grids and have employed an implicit time integration scheme. Direct solution solvers are normally then employed to obtain a solution of the resulting linear algebra problem. Such direct solution solvers are virtually never employed by finite difference modelers. Instead, much more efficient techniques such as iterative solvers and factorization algorithms are normally employed. Of course, finite difference models require some degree of regularity in the grids upon which they are employed.

(Continued)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c OFFICE SYMBOL

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

6a. NAME OF PERFORMING ORGANIZATION (Continued).

USAEWES
Hydraulics Laboratory

and

Computational Mechanics Corporation

6c. ADDRESS (Continued).

PO Box 631
Vicksburg, MS 39180-0631

and

3601 A Chapman Highway
Knoxville, TN 37920

10. PROJECT NO. (Continued).

Funding provided by In-House Laboratory Independent Research Program Project No. 4A061101A91D, sponsored by the Assistant Secretary of the Army (R&D).

19. ABSTRACT (Continued).

Ideas suggested to reduce computational costs of typical finite element hydrodynamic codes are presented for implementation in both a short- and long-term time frame. Short-term suggestions include employing iterative solution solvers with perhaps a multigrid algorithm, uncoupling the governing equations and using linear interpolation for velocity to reduce the "size" of the coefficient matrix, and using explicit time integration with a condensed coefficient matrix. A top-down algorithm/code design from theory through to parallel processing is presented as a long-term solution. A tensor-product solution on at least a semi-regular block structured grid is proposed. With the finite element algorithm derived from the Taylor weak statement, stabilizing mechanisms are embedded such that selective dampening of the spurious short wavelength error modes is realized without a degradation of solution accuracy. Three-dimensional finite element hydrodynamic models based upon these concepts will result in cost-effective solutions that possess good stability properties yet yield accurate solutions in high gradient regions.

PREFACE

The study reported herein was conducted during the period February 1986 to September 1987 by the Hydraulics Laboratory (HL) of the US Army Engineer Waterways Experiment Station (WES) under the general supervision of Messrs. F. A. Herrmann, Jr., Chief, HL, and M. B. Boyd, Chief, Hydraulic Analysis Division (HAD). The study was funded by Department of the Army Project 4A061101A91D, In-House Laboratory Independent Research, sponsored by the Assistant Secretary of the Army (R&D).

Dr. B. H. Johnson, HAD, directed the study and prepared the report with coauthors Dr. A. J. Baker and Mr. P. D. Manhardt of Computational Mechanics Corporation, Knoxville, Tennessee.

COL Dwayne G. Lee, CE, is the Commander and Director of WES. Dr. Robert W. Whalin is the Technical Director.

This report should be cited as:

Baker, A. J., Johnson, B. H., and Manhardt, P. D. 1987 (Sep). "Methods for Reducing Computational Costs of Typical Finite Element Unsteady Hydrodynamic Models," Miscellaneous Paper HL-87-5, prepared by Computational Mechanics Corporation, Knoxville, Tenn., and US Army Engineer Waterways Experiment Station, Vicksburg, Miss.

1

CONTENTS

METHODS FOR REDUCING COMPUTATIONAL COSTS OF TYPICAL FINITE ELEMENT UNSTEADY HYDRODYNAMIC MODELS

PART I: INTRODUCTION

1. In general, either the finite difference method or the finite element method are employed to obtain numerical solutions of the partial differential equations governing the motion of bodies of water. Historically, the majority of numerical hydrodynamic models have employed the finite difference method. The major reason for this is probably because the replacement of derivatives by divided differences is much easier for the typical engineer to understand than basic ideas in variational calculus, upon which the finite element method is based. However, as will be demonstrated, the two methods share much common ground. In fact, most computational methods can be derived from the method of weighted residuals, with the major difference in the methods determined by the selection of the weighting function to which the solution error is made orthogonal. Undoubtedly, the major reason the finite element method came to be applied in numerical hydrodynamic modeling is because of its ability to resolve complex geometry in the computational domain. In particular, the ability to resolve winding navigation channels in estuarine sedimentation studies is virtually a necessity. However, despite its ability to resolve complex geometry in the physical domain, the finite element method suffers from the fact that it is relatively more complicated and expensive to program. In addition, most existing hydrodynamic finite element models appear to require significantly more computational effort per time-step. However, this may be attributed to the techniques commonly employed by finite element modelers in solving the resulting matrix equation system rather than the method itself.

2. A good example of typical finite element hydrodynamic models is the set of models developed by Resource Management Associates (RMA) for the U.S. Army Corps of Engineers Waterways Experiment Station's Hydraulics Laboratory. Codes for laterally averaged water bodies (RMA-7), vertically averaged estuaries and bays (RMA-2) and completely three-dimensional (RMA-10) have been developed. These codes are rather general in that non-structured grids that can be composed of combinations of element types, e.g. quadrilaterals,

triangles, etc. are employed. Since the governing equations are not linearized, Newton iteration is required to solve the resulting set of nonlinear algebraic equations. At each iteration of the Newton scheme a set of linear equations is solved using a form of Gaussian elimination called a frontal solver.

3. Such models can be efficiently applied in studies where steady state solutions are desired. However, even two-dimensional computations can become costly for relatively long term (many tidal cycles) simulations on relatively modest grids. Long term three dimensional simulations are virtually impossible from an economic standpoint with such solution schemes.

4. The major purpose of this study was to suggest ideas for reducing the computational time of typical finite element hydrodynamic models. The approach taken has been to consider "quick fix" ideas, e.g. implementation of explicit time integration with a diagonalized mass matrix, as well as, a more "permanent fix" involving a tensor product implicit solution programmed for parallel processing. The comprehensive permanent solution suggested has benefitted from the substantial progress made in aerodynamics computational fluid dynamics research over the past decade. The algorithm construction presented represents a step forward in top-down CFD algorithm/code design from theory through to parallel processing organization.

5. Before presenting ideas for increasing the computational efficiency of typical finite element hydrodynamic models, features of both the finite difference and finite element methods are discussed. This discussion focuses on the common basis of the two computational methods as well as reasons why the finite element method can be so costly.

PART II: SOLUTION METHODS

Types of Numerical/Free-Surface Hydrodynamic Models

6. Numerical hydrodynamic models can differ widely, depending upon such things as the solution technique applied to the governing differential equations representing the physical processes, the assumptions made in the derivation of the governing equations, whether the phenomena are steady or time-varying, and the spatial dimensionality considered.

7. If the complete three-dimensional (3-D) equations of motion are integrated over a cross section, one-dimensional (1-D) models result. Such models are commonly applied in computing river hydrodynamics, e.g., the computation of floods. Averaging over either the depth or the width results in two-dimensional (2-D) models. Vertically averaged models are applicable for the computation of nearly horizontal flow in relatively shallow and well-mixed bodies of water, whereas laterally averaged models are appropriate when dealing with relatively narrow and deep bodies of water experiencing vertical stratification of the water density.

8. Even though a free surface exists on open bodies of water, some modelers have treated the surface as a rigid lid when very little motion of the free surface occurs. The surface then becomes in essence a solid boundary and the normal component of the velocity must be zero. In addition, the pressure can no longer be prescribed at the surface but rather must be computed. The pressure boundary condition then takes the form of a derivative boundary condition, i.e., a Neumman condition as opposed to a Dirichlet condition in the true free-surface case. All hydrodynamic modeling discussed in this report treats the surface as being free to move so that free-surface waves, e.g., tidal waves in estuaries, are free to be computed.

9. The governing hydrodynamic equations are nonlinear partial differential equations, which in a strict mathematical sense are classified as being of the parabolic type. However, outside the boundary layer the equations exhibit a strong hyperbolic or wave character due to the dominance of the convective terms and thus are often considered as being of the hyperbolic type. In any case, because of the nonlinearity, analytical solutions do not generally exist and one must resort to numerical methods to obtain an approximation of the continuous solution of the differential equations. Such methods

consist primarily of the use of either finite differences or finite elements.

Finite Difference Method

10. In the finite difference method, the domain of the independent variables is replaced by a finite set of points, referred to as net or mesh points, which are structured, i.e., there are ordered directions in the mesh. One then seeks to determine approximate values for the desired solutions at these points. The values at the mesh points are required to satisfy difference equations that can be derived in several ways, although they are usually obtained by replacing partial derivatives by difference quotients. Three approaches are presented below.

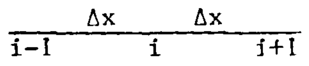
Polynomial fitting

11. With the polynomial approach the dependent variables are represented by a polynomial function with the coefficients determined by the method of collocation, i.e., evaluation of the polynomial at the net points. The polynomial is then analytically differentiated to yield difference expressions for various order derivatives.

12. Consider the representation in one dimension of a dependent variable as a quadratic, i.e.

$$f(x) = a + bx + cx^2 \qquad (1)$$

Evaluating f(x) at the mesh points shown below



yields

$$f_{i-1} = a - b\Delta x + c\Delta x^2 \qquad (2)$$

$$f_i = a \qquad (3)$$

$$f_{i+1} = a - b\Delta x + c\Delta x^2 \qquad (4)$$

from which

$$a = f_i \tag{5}$$

$$b = \frac{f_{i+1} - f_{i-1}}{2\Delta x} \tag{6}$$

$$c = \frac{f_{i+1} + f_{i-1} - 2f_i}{2\Delta x^2} \tag{7}$$

Now, differentiating Equation (1) and substituting the expressions from Equations (5) - (7) yields the centered difference expressions below for first and second order derivatives.

$$\left(\frac{\partial f}{\partial x}\right)_i = \frac{f_{i+1} - f_{i-1}}{2\Delta x} \tag{8}$$

$$\left(\frac{\partial^2 f}{\partial x^2}\right)_i = \frac{f_{i+1} + f_{i-1} - 2f_i}{\Delta x^2} \tag{9}$$

13. As will be seen later, the idea of representing the dependent variables by piecewise continuous polynomials is basic to the finite element method. However, unlike the approach taken above, the method of collocation is not employed.

Integral method

14. In the integral method, the governing equation is approximately satisfied in an integral rather than a differential sense. Consider the 1D equation

$$\frac{\partial \phi}{\partial t} = - \frac{\partial (\phi u)}{\partial x} , \tag{10}$$

with an integration over the spatial domain from $(x - \Delta x/2)$ to $(x + \Delta x/2)$

$$\therefore \quad \int_{x-\Delta x/2}^{x+\Delta x/2} \dot{\phi}\, dx = - \int_{x-\Delta x/2}^{x+\Delta x/2} \frac{\partial(\phi u)}{\partial x}\, dx$$

$$= - \left((\phi u)_{x+\Delta x/2} - (\phi u)_{x-\Delta x/2} \right) \qquad (11)$$

By the mean-value theorem of calculus

$$\int_{x-\Delta x/2}^{x+\Delta x/2} \dot{\phi}\, dx \cong \dot{\phi}(\bar{x})\, \Delta x \qquad (12)$$

where $\bar{x}$ lies between $(x-\Delta x/2)$ and $(x+\Delta x/2)$. Using a mid point evaluation yields

$$\dot{\phi}_i \cong - \frac{(\dot{\phi} u)_{i+1/2} - (\dot{\phi} u)_{i-1/2}}{\Delta x} , \qquad (13)$$

which can be solved for ϕ using some time integration scheme. Note that once again the replacement of the spatial derivative is the equivalent of a centered difference expression.

15. The idea of satisfying the differential equation in an integral sense rather than a differential sense is also basic to the finite element method. However, in the finite element method, rather than setting the integral of the equation over the spatial domain equal to zero, the integral of the solution error multiplied by a non constant weighting function is set to zero.

<u>Taylor series expansion</u>

16. The most common approach taken in the finite difference method is to use Taylor series expansions to derive difference quotients that replace the partial derivatives in the governing equations. For example, $f(x)$ evaluated at $x = i+1$ would be written

$$f_{i+1} = f_i + \left.\frac{\partial f}{\partial x}\right|_i \Delta x + \left.\frac{\partial^2 f}{\partial x^2}\right|_i \frac{\Delta x^2}{2} + \ldots \qquad (14)$$

and thus

$$\frac{\partial f}{\partial x} = \frac{f_{i+1} - f_i}{\Delta x} + O(\Delta x) \qquad (15)$$

where $O(\Delta x)$ refers to additional terms with factors of size Δx . Equation 15 is referred to as a forward difference approximation to the first derivative of $f(x)$. Expanding a Taylor series backwards to $x=i-1$ yields a backwards difference expression. A subtraction of the forward and backward expression yields the centered difference expression presented in Equation 8. Expressions for higher order derivatives of various degrees of accuracy can be derived in a similar fashion. The finite difference approximation of the differential equation is then obtained by replacing the partial derivatives by the difference quotients. Roache (1972) presents an excellent discussion of the use of the finite difference method in solving partial differential equations.

17. All of these approaches to deriving a finite difference recursive algorithm can lead to the same expressions under certain conditions. However, this is not true in general. For example, beyond the second order polynomial the expressions obtained are not identical to those from higher order Taylor series expansions. Similar results will be demonstrated for the finite element method. For example, the finite element method always yields the appropriate order-accurate finite difference recursion relation on a uniform discretization in one dimension if the governing differential equation is linear and linear polynomials are employed. This is not normally true for nonlinear equations and/or nonuniform discretizations and/or multidimensional problems.

Finite Element Method

18. In the finite element approach, the field is divided into smaller regions of convenient shapes, such as triangles or quadrilaterals, and the solution is approximated on each element by interpolation from nodal values on the element. Using a variational principle for Sturm-Liouville type equations, or a weighted-residual method for general equations, the partial differential equations are then transformed into finite element equations

governing each isolated element. These local equations are then collected to form a global system of ordinary differential (in time), or algebraic, equations including a proper accounting of boundary conditions. The nodal values of the dependent variables are then determined from solution of this matrix equation system. Baker (1983) introduces the method in the following manner

> "the finite-element algorithm is perhaps most easily interpreted as an approximate transformation of a partial differential equation (system) into a larger system of lower order differential equations. The computational mesh is formed by the union (summation) of nonoverlapping subdomains, called finite elements, and the mesh can be "arbitrarily" nonregular. Approximation polynomials (of degree k) are prescribed on a local basis within each finite element, to represent all dependent variables and parameters, such as viscosity, Reynolds stress, and thermal conductivity. These approximation basis function sets are "cardinal," i.e., they reduce to zeroes and ones at predetermined locations, within or on the boundary of each computational subdomain called nodes. The boundary of the solution domain need not coincide with surfaces of a global coordinate system. Nevertheless, the "natural" elliptic boundary condition specification, relating the dependent variable and its normal derivative, is routinely implemented along any (all) disconnected portions of the solution domain boundary. The ultimate algebraic equation solution matrix structure produced by the finite element algorithm is sparse and banded, with band width being a function of problem dimension, the degree of interpolation, and the discretization node numbering. The resultant system is directly solvable, or an iterative procedure may be developed using tensor matrix products or other formulations. For parent equation systems exhibiting initial value character, the algorithm yields a system of ordinary differential equations that can be integrated using any explicit or implicit procedure."

Variational approach

19. In contrast to the calculus of a function, the finite element method is based upon the calculus of variations. In general, if we wish to find functions $\phi_i(x,y)$ that are differentiable on (x,y) and satisfy fixed

constraints on the boundary of the domain that minimize some integral functional

$$I = \int_D f\left(x,y,\phi_i, \frac{\partial\phi_i}{\partial x}, \frac{\partial\phi_i}{\partial y}\right) dx\ dy \tag{16}$$

the calculus of variations determines that $\phi_i(x,y)$ are the solutions of the Euler-Lagrange equations

$$\nabla \cdot \left[\frac{\partial f}{\partial(\nabla\phi_i)}\right] - \frac{\partial f}{\partial\phi_i} - 0 \tag{17}$$

Thus, if the problem is to determine solutions of linear partial differential equations similar to equation 17 one can instead extremize the integral statement given by equation 16 once the functional has been determined by inspection. Identification of the functional

$$f\left(x,y,\phi_i, \partial\phi_i/\partial x, \partial\phi_i/\partial y\right) \tag{18}$$

is, of course, a crucial step. Construction of the finite element solution is based upon an approximate evaluation of the integral, followed by construction of the extremum.

20. In the evaluation of the integral, a functional form must be assumed for ϕ_i which contains expansion coefficients. From a review of properties of fundamental solutions of Sturm-Liouville type equations such as Equation 17, Baker (1983) discusses the desirable properties of orthogonality and completeness and their relationship to the selection of the functional form for ϕ_i. The basic definition of these two properties are given below. Two functions are orthogonal on the interval $x_1 \leq x \leq x_2$ if

$$\int_{x_1}^{x_2} W(x)\ U_n(x)\ U_m(x) = A_n\ \delta_{nm}\ , \tag{19}$$

where δ_{nm} is the Kronecker delta. The finite element procedure leads to the establishment of locally orthogonal approximation functions, i.e. Equation 19 is satisfied with the exception that the integral fails to vanish on an individual finite element. The orthogonality property is important since it results in sparse coefficient matrices in the resulting linear algebra problem.

21. A function set $U_n(x)$ is defined as complete on $x_1 \leq x \leq x_2$ if for every $\delta > 0$ there exists a number $N > 0$ such that the distance between an arbitrary square integrable function $f(x)$ and a series expansion in $U_n(x)$ can be made arbitrarily small, i.e.

$$\int_{x_1}^{x_2} \left[f(x) - \sum_{n=0}^{N} C_n U_n(x) \right]^2 dx < \delta \tag{20}$$

With this property an increase in accuracy for the solution $f(x)$ as N increases is guaranteed.

22. The use of local polynomials as the approximating functions provides some degree of orthogonality coupled with the assurance of reducible error as either the grid is refined or as the degree of the polynomial is increased, i.e., as a more complete basis is utilized. Other functions could be used, e.g., completely orthogonal functions such as Bessel functions or perhaps a polynomial spanning the entire spatial domain rather than an individual element. However, as Baker (1983) notes, the use of completely orthogonal functions severely constrains boundary condition flexibility, although it does admit improved solution accuracy with the use of a more complete basis. The use of polynomials spanning the complete domain results in no degree of orthogonality and thus the coefficient matrix that is formed in the linear algebra problem becomes extremely dense and numerical error eventually swamps the solution as higher degree polynomials are employed. The finite element method is synonymous with the use of piecewise local interpolation polynomials for approximating functions.

Method of weighted residuals

23. Since the equations of fluid mechanics are written in an Eulerian frame of reference they are nonlinear. Thus the variational approach to the finite element method cannot be used since the functional given by Equation 18 cannot be found. Therefore, for nonlinear equations the finite element method

is based upon the method of weighted residuals. As a matter of fact, virtually all computational methods can be derived in this manner.

24. As discussed by Baker (1983), the fundamental concept is the constraint of the solution error. One deals directly with the differential equation $L(q)$ and the boundary condition $\ell(q)$, where $L(\cdot)$ and $\ell(\cdot)$ are differential operators. Assuming that q^h is the approximate solution, then $L(q^h)$ and $\ell(q^h)$ are statements of the error in the solution approximation. This error is then required to be orthogonal to some set of weighting functions $W(x)$ over the domain and its boundary i.e.,

$$\int_{\Omega} \left\{ W(\bar{x}) \right\} L(q^h)\, d\bar{x} - \lambda \int_{\partial\Omega} W(\bar{x})\, \ell(q^h)\, d\sigma \equiv \{0\} \qquad (21)$$

where

λ = Arbitrary multiplier

x = x, y, z

Ω = Computational domain

$\partial\Omega$ = Boundary of domain

$d\sigma$ = Boundary area differential

dx = Domain volume differential

There are as many scalar equations as members of the weight set $\{W(x)\}$. The next step is to write Equation 21 as the global assembly of the equation applied over each element, i.e.

$$\int_{e} \left[\int_{\Omega_e} \{W(\bar{x})\} L(q_e)\, \bar{d}x - \lambda \int_{\partial\Omega_e \cap \partial\Omega} \{W(\bar{x})\} \ell(q_e)\, d\sigma \right] \equiv \{0\} \qquad (22)$$

where

$$q^h(\bar{x}) = \bigcup_{e=1}^{M} q_e(\bar{x}) \qquad (23)$$

and M is the total number of elements. The basic distinction between the finite element method and other computational methods, e.g. finite volumes, is the selection of the weight function basis $\{W(x)\}$.

25. Baker (1983) demonstrates that in order to reproduce the energy functional extremization for the linear, steady state heat conduction problem

the weight function basis must employ the same basis as used in the approxima-tion of the solution, i.e.

$$q_e(\bar{x}) = \{N_k(\bar{x})\}^T \{Q\}_e \qquad (24)$$

and thus

$$\{W(\bar{x})\} = \{N_k(\bar{x})\} \qquad (25)$$

The comparison is exact except for a closed surface integral over the boundary of each element. For the analogy to be exact this integral must be zero. This is not the case for an individual element but the assembly of the surface integral over the complete domain does yield a zero value. However, this implies that the derivative of a dependent variable may be discontinuous at element intersections.

26. A finite difference algorithm can be derived by setting the weight function basis $\{W(x)\}$ to be a set of constants. The surface integral noted above then becomes zero over each element since a derivative of $\{W(x)\}$ appears in the integral and the derivative of a constant is zero. Therefore, derivatives of the dependent variables are continuous at finite difference cell interfaces. This is a fundamental difference between the two methods as derived by the method of weighted residuals. The reason for defining the finite element method by forcing the method of weighted residuals to reproduce the variational problem for a linear problem is because the theoretical structure for such linear problems guarantees an optimally accurate solution.

The Linear Algebra Problem

27. Consider, for example, the depth-averaged 2-D equations for free-surface hydrodynamic flows. A finite element (FE) algorithm for these equa-tions formally states the requirement to establish an approximation $q^h(\cdot)$ to the solution set $q(x,t) \equiv \{\phi(\cdot), u_i(\cdot)\}$ where ϕ is the water surface and u_i is the velocity, to be constructed on a discretization Ω^h of the solu-tion domain $\Omega \subset R^2 \times t \in \{x_i: 1 \leq i \leq 2, t \geq t_o\}$. This approximation is

constructed as the union of elemental contributions $q_e(x_i,t)$, which in turn are defined as expansions in the k^{th} degree polynomial cardinal basis set $\{N_k(\cdot)\}$ and time-dependent nodal expansion coefficients $\{Q(t)\}_e$. Mathematically, this statement of approximation is

$$q^h \equiv \underset{e}{\cup}\, q_e(x_i,t) \tag{26}$$

where

$$q_e \equiv \{N_k(x_i)\}^T\{Q(t)\}_e \tag{27}$$

28. Since Equations 26-27 define the approximation, substitution into the flow equations defines the associated error. The second step of a FE algorithm is therefore a formal statement of constraint on this error. The form of this constraint which, as previously noted, enjoys an optimal error estimate for a linear elliptic problem statement, is the so-called Galerkin statement, which requires the distribution of the error to be orthogonal to the function space $\{N_k(\cdot)\}$ used to construct q^h . For the more general problem statement wherein nonsmooth solutions to hyperbolic equations are sought, the Galerkin statement is typically augmented with an additional dissipative constraint, Baker (1983). The mathematical statement of error constraint is thus of the form

$$\int_{R^2}\{N_k(\cdot)\}\, L(q^h) + \vec{\beta}\cdot \int_{R^2}\nabla\{N_k(\cdot)\}L^c\,(q^h) \equiv \{0\} \tag{28}$$

where $\vec{\beta}$ is a parameter set that can be optimized and $L^c(\cdot)$ is the substantial derivative operator.

29. Upon evaluation of the integrals in Equation 28, there results an ordinary differential equation system written on the time evolution of the expansion coefficient set {Q(t)} in the form

$$[M]\,\frac{d}{dt}\,\{Q\} + \left(\{U\}^T\,[C] + \{G\}\right) \equiv \{0\} \tag{29}$$

where

$[M]$ = mass matrix

$\{U\}$ = convection velocity field

$[C]$ = associated (convective derivative) influence on the dependent variable set $\{Q\}$

$\{G\}$ = remaining source terms

For the time-accurate solution, Equation 29 is employed to evaluate the Taylor series

$$\{F\} \equiv \{Q\}_{j+1} - \{Q\}_j - \Delta t \frac{d}{dt} \{Q\}_{j+\theta} - \dots \equiv \{0\} \tag{30}$$

Equation 30 is a nonlinear algebraic equation system for $\theta > 0$, the solution statement for which is cast using a Newton iteration algorithm in the form

$$[J] \{\delta Q\}_{j+1}^{p+1} \equiv - \{F\}_{j+1}^{p} \tag{31}$$

where p is the iteration index. The solution field is defined as

$$\{Q\}_{j+1}^{p+1} \equiv \{Q\}_{j+1}^{p} + \{\delta Q\}_{j+1}^{p+1} \tag{32}$$

and the Newton Jacobian is constructed as

$$[J] \quad \frac{\partial\{F\}}{\partial\{Q\}} \tag{33}$$

All computational models of unsteady physical conservation laws eventually produce the algebraic equation system (Equation 30) the numerical solution of which (Equations 31-33) constitutes the heart of a code.

30. In the finite element method the Jacobian or coefficient matrix is a relatively sparse matrix whose bandwidth crucially impacts upon the computational costs of solving equation 31. The bandwidth is dependent upon the dimensionality of the problem, the number of coupled variables being computed, the number of elements and the degree of the approximating polynomials. Since

the Jacobian is independent of x, any direct solution procedure is usable. Most existing finite element hydrodynamic models, e.g. RMA models, employ a solution scheme which reduces the bandwidth before employing Gaussian elimination. However, the computational costs for multidimensional (especially 3D) models can still become astronomical for time dependent problems.

Summary

31. The two major computational methods, i.e. finite difference and finite element, have been discussed. Common features as well as major differences have been noted. The finite element method for general equations must be derived using the method of weighted residuals, with the weighting function being the same as that used to approximate the dependent variable. It can be shown for a linear equation that this results in the lowest possible solution error. However, this can not be shown for nonlinear equations and, in addition, as a result of a nonzero surface integral on individual elements, derivatives of the dependent variables are not generally continuous. Therefore, mass continuity may not be satisfied locally in hydrodynamic models, although it will be satisfied globally, i.e. over the complete domain.

32. Finite difference algorithms are normally derived by replacing the derivatives by difference quotients derived from Taylor series expansions. The resulting difference equations are then evaluated at each net point. As demonstrated, however, there are other approaches, e.g. the use of polynomials and the integral method. Each possesses features of the finite element method. In fact, if the weighting functions are taken as constants, finite difference type algorithms can be developed from the method of weighted residuals. In particular, setting the constants equal to one results in what is referred to as a finite volume algorithm. On a structured grid with uniform grid spacing, a finite difference algorithm that would result from application of the integral method results. With constants used for the weighting functions, the continuity of derivatives is assured and thus mass conservation is satisfied even on the element level. However, the global solution error may be larger than when the weighting function is taken as the basis function.

33. Finally, it has been demonstrated that regardless of the computational theory, ultimately a linear algebra problem must be solved. The manner

in which this problem is handled is the major factor that determines the computational cost in three dimensional time varying hydrodynamic models.

PART III: SUGGESTION FOR IMPLEMENTATION IN THE SHORT-TERM

34. The major computational costs involved in multidimensional numerical hydrodynamic models occur in solving the linear algebra problem given by Equation 31. Note that even if the original system of equations are nonlinear, a linear system of the form

$$[A]\{x\} = \{b\} \tag{34}$$

must be solved for each Newton iteration. Therefore, the major consideration in attempting to reduce the computational costs involves the manner in which the linear algebra problem is solved.

35. As previously noted, the size or bandwidth of the coefficient matrix A is dependent upon the problem dimensionality, the number of coupled dependent variables, the number of finite elements, and the degree of the interpolating polynomials. Anything that can be done to reduce the size of A should reduce computational costs, regardless of the solution procedure employed for the linear algebra problem.

36. Depending upon the type of element shapes employed, the complexity of various integrals formed in the finite element method varies. If numerical quadrature is required for the evaluation of these integrals over each element, a substantial portion of the computational costs for one and two dimensional hydrodynamic models can be related to construction of the coefficient matrix. Therefore, one area to consider for possible reduction of computational costs is in the construction of the coefficient matrix.

37. Numerical hydrodynamic models are typically time dependent models for computing flows over time periods ranging from a few tidal cycles to perhaps several months. In recent years most models, both finite difference and finite element, have employed implicit time integration schemes. However, depending upon the application, explicit schemes may be more economical and should be considered as a solution that could be implemented in a short-term time frame.

Linear Algebra Solvers

38. Any numerical linear algebra procedure is a candidate for solving

Equation 34. A key implementation aspect relates to code efficiency. Finite element fluid mechanics codes have gained a reputation for being less efficient than their finite difference counterparts. As noted earlier, this is not a consequence of the finite element theory itself, but instead reflects directly on implementation choices for the linear algebra statement. Based on the finite element structural mechanics historical development of direct, out-of-core solvers, many (most) finite element code implementations have relied on direct extension of these solver software packages to the fluid mechanics problem classes. In distinction, a direct solution (of even a linear Poisson equation) is never attempted in a FD code, but instead a matrix iteration procedure is defined that uses easily formed (usually block-tridiagonal) approximations to the Newton algorithm Jacobian (Equation 33). The computer storage requirement for the approximate Jacobian is negligible in comparison, as is the CPU needed to execute an LU decomposition and back substitution. The computational penalty is a much reduced convergence rate in comparison to a Newton iteration, but each iteration proceeds so rapidly that the many grid sweeps required for the FD code are typically completed with minimal expenditure of computer resources.

39. Candidate methods employed in finite difference models include point iterative methods (Jacobi, Gauss-Seidel, successive over-relaxation), line iterative methods (vertical/horizontal over-relaxation, alternating direction implicit, approximate factorization), conjugate gradient methods and multi-grid techniques.

40. The RMA models employ a direct solver called a frontal solution procedure. In the frontal method the assembly and elimination of equations is interweaved, based on the fact that an equation can be eliminated if all the elements contributing to the equations are assembled. The front is defined by all the active coefficients that are not yet eliminated. Thus, the equations are not assembled and eliminated sequentially. New equations take up the positions of equations which have ceased to be active and the front matrix contains only a small number of zero coefficients. When employed in a finite element context, the efficiency of a frontal scheme is dependent upon the ordering of the elements, with the ordering of the nodes being immaterial since the equations are assembled on an element-by-element basis. A detailed discussion of the frontal approach is given by Irons and Ahmad (1980). Sloan

and Randolph (1983) show that an upper bound on the number of operations is given by

$$T = \frac{N}{2} (W^2 + W - 2) \tag{35}$$

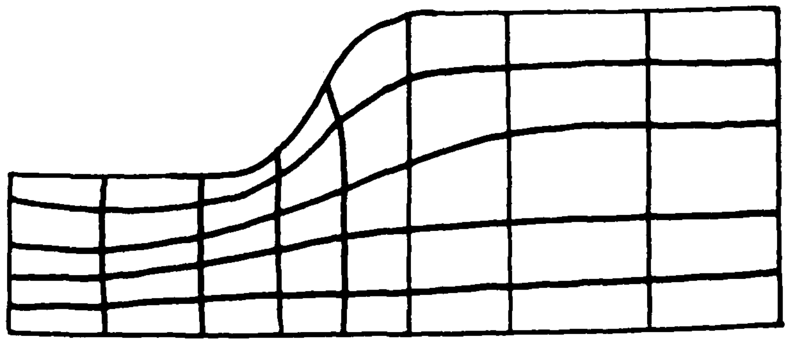
where W is the maximum front width and the rank of the coefficient matrix is N. By comparison, Gaussian elimination operating directly on [A] involves a total number of operations proportional to N^3.

Structured vs Unstructured Grids

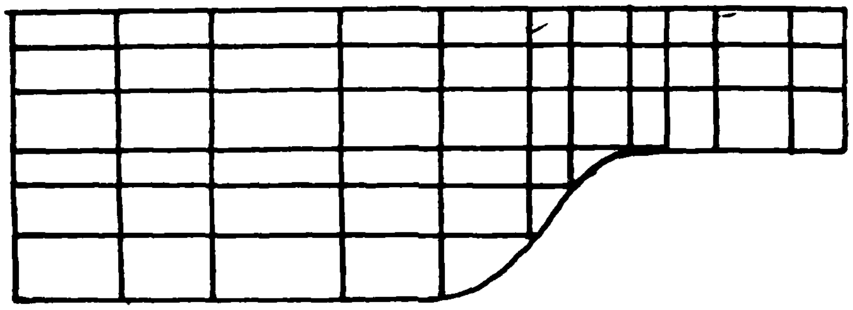
41. The solution cost can be directly related to the type of grid employed. A mesh or grid is said to be structured or regular if the geometric domain is divided into q rows with p elements per row. With regular grids, the unknowns are ordered such that the nonzero entries of the coefficient matrix lie in a predictable pattern. A completely unstructured grid implies that the non-zero entries are scattered rather haphazardly throughout the matrix. Between these two types of grids are semi regular grids. With semi-regular grids the domain is divided into rows and columns but there may be a variable number of elements per row. Figure 1 illustrates the three types of grids.

42. Usually, the more flexible and general a model is the greater are its solution costs. Therefore, allowing for completely unstructured grids in the RMA codes significantly increases the matrix solution cost. The factors which most affect matrix solution costs are total arithmetic operations required, storage requirements and overhead due to data transmission and to logical operations associated with the solution method. The relative importance is dependent upon the computer being used and the size of the problem being solved.

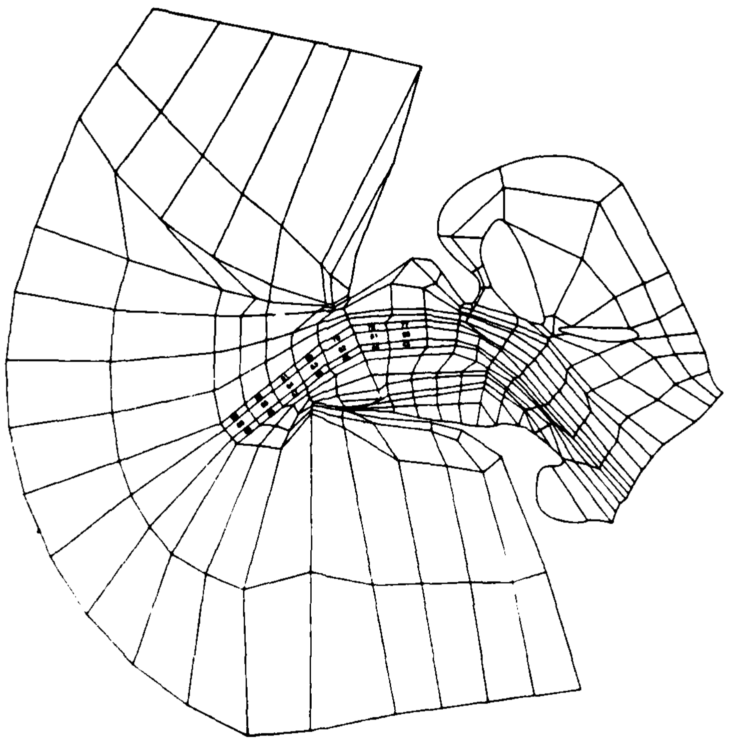
43. Many efficient direct methods exist, e.g. the frontal solution scheme employed in the RMA codes; however, depending upon the problem an iterative method can be more efficient. Among the factors to be considered are computer storage requirements and the number of arithmetic operations required. Hageman and Young (1981) state that for many problems there is a cross-over point in the number of unknowns, above which a good iterative



a. Regular grid



b. Semi-regular grid



c. Unstructured grid

Figure 1. Types of grids

method becomes more cost effective than a good direct method. In particular, they indicate that for unstructured grids the block CCSI (cyclic Chebyshev semi-iterative) scheme, used with an algorithm to minimize bandwidth, is worthy of consideration. A recent paper by Mavriplis and Jameson (1987) demonstrates that the efficiency of a multigrid solution on unstructured triangular meshes employing Jacobi iteration is competitive with available structured mesh Euler solvers. Therefore, one measure worth considering for implementation in codes such as the RMA codes is an iterative solution scheme. A more permanent solution to the reduction of linear algebra costs is to reduce the generality of the model to allow at most only semistructured grids such that a tensor product factorization of the coefficient matrix is possible. Through the use of grids composed of blocks, significant generality is retained. With such an approach the finite element method becomes a viable solution method for computing long term hydrodynamics. This approach is discussed in PART IV.

Reduction of the Size of the Coefficient Matrix

44. The size, i.e., bandwidth of the coefficient matrix in the linear algebra problem is dependent upon the number of coupled partial differential equations to be solved, the dimension of the problem, the number and ordering of the finite elements and the degree of the basis function, i.e., the approximating polynomial. Thus, in order to reduce the size of the matrix one or more of the above factors must be addressed. Since the problem dimension and the number of finite elements employed are problem dependent these are not considered. However, the coupling of equations and the completeness of the basis function are certainly areas in which modifications should be considered.

Reduction of matrix rank

45. Typical finite element hydrodynamic codes, e.g. the RMA models, are completely implicit in that all variables are coupled. For example, in the 3D RMA-10 model the coupled variables become the water surface, three velocity components and salinity. One approach taken in numerical hydrodynamic models that employ the finite difference method is to compute the water surface implicitly through the derivation of a frictionally damped wave equation and to then use the implicitly computed water surface in explicit computations for

the velocity components, which are then employed in computations for the salinity, i.e., the equations are uncoupled. With this approach, restrictions on the allowable time step will occur; however, the stability criteria are primarily related to the speed of a water particle rather than the speed of a free surface wave, which can often be quite restrictive.

46. With this approach employed in RMA-10, five matrix equations would need to be solved but each matrix would only be 1/5 the size of the matrix arising from the coupled case. Since the arithmetic operations required for solution are proportional to the rank of the matrix times the square of the bandwidth, obviously the solution of five small problems will be much less expensive than that of the large problem. This should be true even though the computational time step allowed will be smaller than that for the completely implicit case.

<u>Bandwidth reduction</u>

47. The second way in which the size of the coefficient matrix can be reduced is related to the degree of the approximating polynomial. The greater the degree of the polynomial the denser the coefficient matrix becomes since the higher degree elements connect nodes over a larger stencil. Thus, the bankwidth is significantly larger. Baker (1983, Ch. 3) compares the computational requirements for a potential flow solution using linear and quadratic approximating polynomials. A discretization containing 624 nodes was solved in both cases. The quadratic solution required approximately twice the computer storage and 40 times the CPU. The two solutions were comparable in accuracy.

48. In the RMA codes, the water surface is approximated with a linear basis function; whereas, the velocity components are approximated by a quadratic. For a non-free surface incompressible fluid, the velocity is required to be approximated by a polynomial of one degree greater than the pressure. However, free surface hydrodynamic computations are analogous to compressible flow computations in aerodynamics. For example, hydraulic jumps develop when the water velocity become supercritical while shock waves develop in air when the Mach number exceeds one. Finite element researchers in the aerodynamics field routinely employ the same degree basis function for the velocity and pressure. Therefore, one possible means for reducing the computational costs of codes such as the RMA finite element models would be to employ a linear basis function for the velocity components also. Naturally this will reduce

the theoretical accuracy of the computations for a fixed mesh size. However, as illustrated by the potential flow solution example above, this may be a preferred alternative. One point to consider is that rather large eddy diffusion coefficients must often be employed in models, such as the RMA codes, that employ the standard Galerkin finite element method in order to dampen dispersion error waves that are created. The use of eddy coefficients multiplied by second derivatives, however, does not provide selective damping. Therefore, the flow solution of interest is also damped with the net effect being that a lower order approximation with built in numerical diffusion might as well have been used. A discussion of finite element solutions based on the Taylor weak statement, which provides selective numerical damping (dampens primarily the short wave lengths) is presented in PART IV.

Static condensation

49. One additional method for reducing the size of the coefficient matrix is referred to as static condensation. This technique can only be applied to linear equations using higher degree basis functions. The procedure involves reducing out the interior nodes of a higher degree element from the problem. The effect is an overall reduction in the rank of the coefficient matrix and thus substantial potential savings in computer costs. Of course, due to the nonlinearity of the hydrodynamic equations static condensation can not be applied unless the equations are linearized. However, this is often done in finite difference models where the nonlinear convective terms, e.g. $u \frac{\partial u}{\partial x}$, are handled by taking the u at the previous time step or perhaps, as in the uncoupled frictionally damped wave equation approach, the complete term is lagged at the previous time step. This effectively linearizes the equations at a time step and thus static condensation could be employed if higher degree basis functions are assumed.

50. One additional comment concerning the lagging of the nonlinear terms at the previous time step should be made. With such an approach, the matrix equation is solved only once per time step as opposed to the iterative Newton Raphson scheme employed in models such as the RMA codes to solve systems of nonlinear algebraic equations. The RMA codes commonly require 2-3 iterations per time step. Lagging the nonlinear terms will immediately reduce the computational cost by a significant factor, even if nothing else is done. Lagging the nonlinear terms should not significantly influence the solutions

generated by codes such as the RMA models since they are normally applied for the computation of gradually varying phenomena, e.g. tidal circulation in estuaries. This would not be the case if the models were intended to compute rapidly varying surges or perhaps hydraulic jumps, i.e., highly nonlinear phenomena.

Reduction of Costs in Problem Setup

51. In the finite element method, integrals involving the basis functions and derivatives of the functions arise. If the finite elements are restricted to be triangles in two dimensions (N = 2) and tetrahedrons in three dimensions (N = 3), all integrals can be analytically evaluated through the use of the expression below

$$\int_{\Omega_e c R^N} \delta_1^{N_1} \delta_2^{N_2} \delta_3^{N_3} \delta_4^{N_4} \, d\bar{x} = D_e^N \frac{N_1! N_2! N_3! N_4!}{\left(N_1+N_2+N_3+N_4\right)!} \tag{36}$$

for linear as well as quadratic basis functions. However, if quadrilaterals and/or quadrahedrons are employed the resulting integrals can not in general be evaluated analytically. Therefore, the integrals must be evaluated numerically with Gaussian quadrature commonly employed. This of course increases the computational costs involved in forming the matrix equation, with these costs being directly related to the order of the quadrature employed. Since the quadrature error is $O(h)^{2k}$, where k is the order of the quadrature formula and h is a measure of the mesh, using $k=2$ should be quite sufficient for models employing quadratic basis functions. Reducing the order of the numerical quadrature in finite element models to be no greater than the order of the basis function employed will reduce problem setup costs.

Implicit vs Explicit Time Integration

52. The finite element algorithm applied to time dependent partial differential equations yields a large order system of ordinary differential equations. In other words, the matrix equation involves unknowns that are time derivatives of the nodal values of the dependent variables. Any integration

formula can be applied to transform the system of ordinary differential equations to an algebraic system.

53. The general form of the system of ordinary differential equations becomes

$$[C]\,\frac{d\{Q(t)\}}{dt} + [D]\,\{Q(t)\} = \{B\} \tag{37}$$

Employing a one step time integration algorithm yields

$$\Big([C] + \Delta t\theta\,[D]\Big)\,\{Q\}^{n+1} + \Big((\Delta t(1-\theta)[D] - [C]\Big)\,\{Q\}^{n} = \Delta t\,\{B\} \tag{38}$$

where θ controls the implicitness, e.g., $\theta=1$ yields a completely implicit scheme while $\theta=0$ results in an explicit time integration scheme, i.e., the forward Euler integration scheme. An interesting observation is that even if $\theta=0$, i.e., the time integration is explicit, a system of algebraic equations remain to be solved in an implicit fashion since the matrix [C] is not a diagonal matrix. This, of course, is unlike the finite difference method which yields (by delinition) a diagonal recursive relationship for the unknowns when an explicit time integration scheme is employed.

54. Based upon the the fact that [C] is not diagonal, it would appear foolish to employ an explicit time integration scheme with the finite element method since essentially the same linear algebra problem remains, but with the computational time step restricted by stability criteria. One solution is to diagonalize [C]. As discussed by Baker (1983, Ch. 4), there are two common approaches taken. One method is row wise summation, i.e.,

$$C_{ii} = \sum_{i=1}^{n} C_{ij} \; ; \tag{39}$$

whereas, the second is normalization of the diagonal elements of [C] with the off diagonal terms then set to zero. The non-diagonalized matrix is referred to as a consistent matrix, whereas, the diagonalized form is referred to as a condensed matrix.

55. Baker (1983, Ch. 4) presents solutions for the one dimensional diffusion equation (without convection) for using both the consistent and

condensed forms on the same grid. The grid used was rather coarse, and dispersion error "wiggles" developed in the consistent solution. However, the wiggles disappeared in the condensed solution, confirming that the condensing operation yielded an artificial smoothing of the solution. It might be noted that the maximum allowable time step for explicit integration of the condensed form was about twice that of the consistent form.

56. To further illustrate that finite difference recursion relations can be derived from the weighted residual method used for generating the finite element method, Baker (1983, Ch. 4) shows that for a convection-diffusion problem in one dimension, with a linear basis function, the finite element method, employing the trapezoidal rule time integration algorithm, yields what is referred to as the Chapeau finite difference algorithm on a uniform discretization. If the trapezoidal rule integration is employed, but the coefficient matrix multiplying the vector containing the unknown derivatives of the nodal unknowns is diagonalized, the resultant recursion relation becomes identical to the Crank-Nicolson finite difference scheme.

57. It appears that if the hydrodynamics of a water body are to be computed on an unstructured grid, an explicit time integration scheme with a condensed matrix should be considered for implementation. Even though the time step will be restricted, the linear algebra problem becomes a trivial operation and the computational costs may well be significantly less. This will especially be true if the water surface is computed implicitly with all other variables computed explicitly using the condensed form, since the most severe stability criterion, i.e., the free surface wave speed, will be removed from the stability criteria.

PART IV: LONG TERM (PERMANENT) SOLUTION APPROACH

58. The discussion up to this point has focused on ideas for reducing computational costs that can be implemented in a relatively short period of time. A longer term research effort following the approach outlined below will yield not only a computationally efficient finite element model but one with excellent stability properties. Much of the suggested approach is a result of a study of the fairly typical finite element hydrodynamic code RMA-10 which was developed by King (1982).

Problem Statement

59. The Reynolds-averaged Navier-Stokes equations, Cebeci and Smith (1974), are considered appropriate for hydrodynamic three-dimensional flows. The form usually considered as baseline for hydrodynamics analyses is given by King (1982) as,

$$\rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \rho v\frac{\partial u}{\partial y} + \rho w\frac{\partial u}{\partial z} - \frac{\partial}{\partial x}(\varepsilon_{xx}\frac{\partial u}{\partial x}) - \frac{\partial}{\partial y}(\varepsilon_{xy}\frac{\partial u}{\partial y}) \tag{40}$$

$$- \frac{\partial}{\partial z}(\varepsilon_{xz}\frac{\partial u}{\partial z}) - \frac{\partial p}{\partial x} - \tau_x^* = 0$$

$$\rho\frac{\partial v}{\partial t} + \rho u\frac{\partial v}{\partial x} + \rho v\frac{\partial v}{\partial y} + \rho w\frac{\partial v}{\partial z} - \frac{\partial}{\partial x}(\varepsilon_{yx}\frac{\partial v}{\partial x}) - \frac{\partial}{\partial y}(\varepsilon_{yy}\frac{\partial v}{\partial y}) \tag{41}$$

$$- \frac{\partial}{\partial z}(\varepsilon_{yz}\frac{\partial v}{\partial z}) - \frac{\partial p}{\partial y} - \tau_y^* = 0$$

$$\rho\frac{\partial w}{\partial t} + \rho u\frac{\partial w}{\partial x} + \rho v\frac{\partial w}{\partial y} + \rho w\frac{\partial w}{\partial z} - \frac{\partial}{\partial x}(\varepsilon_{zx}\frac{\partial w}{\partial x}) - \frac{\partial}{\partial y}(\varepsilon_{zy}\frac{\partial w}{\partial y}) \tag{42}$$

$$- \frac{\partial}{\partial z}(\varepsilon_{zz}\frac{\partial w}{\partial z}) - \frac{\partial p}{\partial z} - \rho g - \tau_z^* = 0$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{43}$$

where $x_i = \{x,y,z\}$ is the cartesian coordinate system, $u_i = \{u,v,w\}$ is the corresponding velocity field expressed in cartesian components, g is gravity, p is pressure, and ρ is the water density. Further, ϵ_{xx}, ϵ_{xy}, etc. are the turbulent eddy viscosity coefficients, and τ^*_x, τ^*_y, τ^*_z are the surface tractions operating on the boundaries only. Figure 2 schematically represents the water body and conventional directions for the axes and velocities. For the RMA-10 analysis, it is assumed that the vertical momentum equation (42) may be reduced to the form,

$$\rho g + \frac{\partial p}{\partial z} = 0 \tag{44}$$

i.e., the pressure in the vertical (z) direction is hydrostatic.

60. RMA-10 handles both homogeneous and stratified flow problem descriptions. Taking the general case, the assumption of Equation 44 yields the pressure gradient expressions

$$\frac{\partial p}{\partial x} = \frac{\partial}{\partial x}\int_z^{a+h} \rho g \; dz \tag{45}$$

$$\frac{\partial p}{\partial y} = \frac{\partial}{\partial y}\int_z^{a+h} \rho g \; dz \tag{46}$$

For the homogeneous flow assumption, Equations 45-46 then simplify to,

$$\frac{\partial p}{\partial x} = \rho g\left(\frac{\partial h}{\partial x} + \frac{\partial a}{\partial x}\right) \tag{47}$$

$$\frac{\partial p}{\partial y} = \rho g\left(\frac{\partial h}{\partial y} + \frac{\partial a}{\partial y}\right) \tag{48}$$

where the definitions h(x,y,t) and a(x,y) are noted in Figure 2. The RMA-10 equation set is completed with the inclusion of an advection-diffusion equation for transport of scalar fields, eg., temperature, salinity, sediment, in the form (with temperature as the example),

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} + w\frac{\partial T}{\partial z} - \frac{\partial}{\partial x}(D_x\frac{\partial T}{\partial x}) - \frac{\partial}{\partial y}(D_y\frac{\partial T}{\partial y}) - \frac{\partial}{\partial z}(D_z\frac{\partial T}{\partial z}) - \theta_s = 0 \quad (49)$$

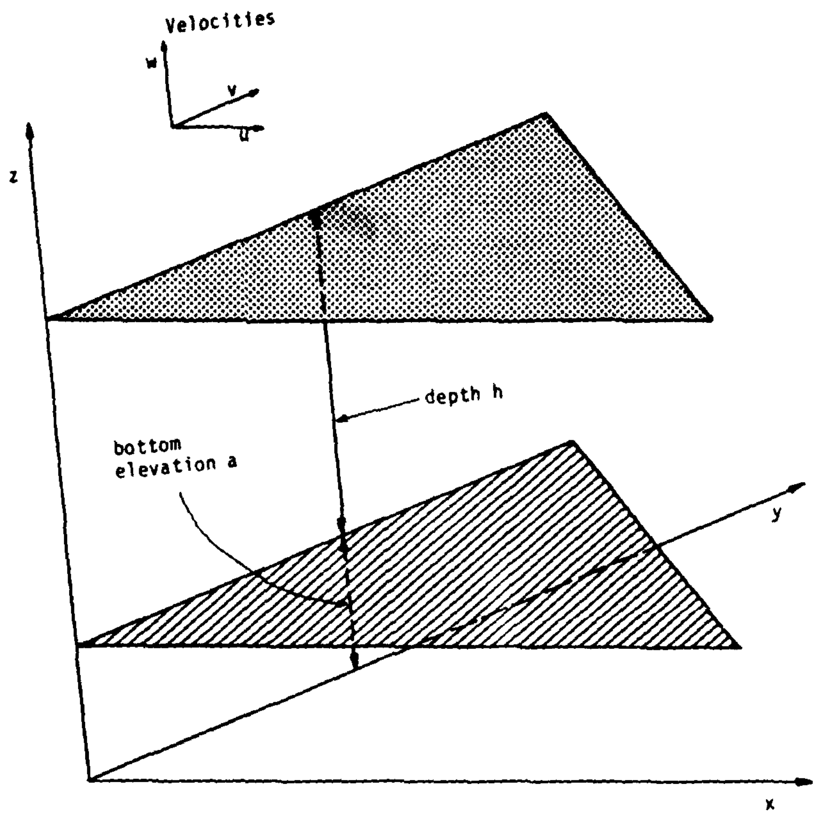


Figure 2. Coordinate System and Geometry Definitions For 3D Hydromechanical Flow Analysis, From King (1982, Figure 2.1).

where θ_s is a source-sink term and a state equation of the form $\rho=\rho(S,T)$ is added. For completeness, stresses acting on the water surface area are assumed of the form,

$$\tau_x = \psi \cos \theta \, w^2 \quad \text{and} \quad \tau_y = \psi \cos \theta \, w^2 \quad (50)$$

where ψ = a constant, θ = wind direction, and w = wind speed. While RMA-10 assumes these represent fixed tractions, hence are treated as constants, the general case permits their time dependence. Stresses acting on the bottom are of the form,

$$\tau_x = -\frac{\rho g\, u\, V}{C^2} \qquad \tau_y = -\frac{\rho g\, v\, V}{C^2} \tag{51}$$

where $V = (u^2 + v^2)^{1/2}$, the surface velocity magnitude and C is the Chezy coefficient. These stresses are not constant and thus are incorporated into the nonlinear solution dependent upon the local velocities (only over surfaces).

61. As a final formulational step, the hydrostatic pressure assumption deletes the remaining terms in Equation 42 as higher order. Thus, the continuity equation 43 becomes that governing the vertical velocity component w. Integrating over z yields

$$\int_a^{a+h} \left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right] dz = -\int_a^{a+h} \frac{\partial w}{\partial z}\, dz = -w(x,y,a+h) + w(x,y,a) \tag{52}$$

At the top surface,

$$w(x,y,a+h) = u(x,y,a+h)\,\frac{\partial(a+h)}{\partial x} + v(x,y,a+h)\,\frac{\partial(a+h)}{\partial y} + \frac{\partial h}{\partial t} \tag{53}$$

While at the bottom,

$$w(x,y,a) = u(x,y,a)\,\frac{\partial a}{\partial x} + v(x,y,a)\,\frac{\partial a}{\partial y} \tag{54}$$

Combining Equation 52-54 yields,

$$\frac{\partial}{\partial x}\int_a^{a+h} u\, dz + \frac{\partial}{\partial y}\int_a^{a+h} v\, dz + \frac{\partial h}{\partial t} = 0 \tag{55}$$

Thus, with Equation 55 the vertical velocity appears only in the lateral (x,y) momentum equations. RMA-10 treates only a minor coupling, computing w separately from the basic continuity equation 43. The pressure p can be eliminated via Equations 47-48 and replaced by depth h which is a function only of x , y and t , as governed by Equation 55. Finally, Equation 49 completes the set for scalar field convection-diffusion along with $\rho=\rho(S,T)$.

62. RMA-10 code implementation of a finite element algorithm for solution of Equations 40, 41, 43, 49 and 55 utilizes a coordinate transformation to a "regularized" solution domain, which in itself creates potential numerical instabilities. Contemplating the need to alter this, the first step to long term generalization would be to re-express the governing hydrodynamic equation system in the preferred form of a (hyperbolic) conservation law statement. The computational aerodynamics literature has become filled with the observation that this form is preferred for a variety of reasons including truncation error control, stability and exact satisfaction of sharp solution gradient (approximations). Thus, without loss of any generality, the first step would be to complete the details of re-expression of Equations 40-43 and 49 in the form,

$$L(\rho) = \frac{\partial}{\partial x_i}(\rho u_i) = \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} + \frac{\partial \rho w}{\partial z} = 0 \tag{56}$$

$$L(\rho u_i) = \frac{\partial(\rho u_i)}{\partial t} + \frac{\partial}{\partial x_j}\left(u_j \rho u_i + p\delta_{ij} - \sigma_{ij}\right) = 0 \tag{57}$$

$$L(\rho\phi) = \frac{\partial(\rho\phi)}{\partial t} + \frac{\partial}{\partial x_j}\left(u_i \rho\phi - d_j\right) + S_\phi = 0 \tag{58}$$

$$L(h) = \frac{\partial h}{\partial t} + \frac{\partial}{\partial x_j} \quad u_j h + f(a,h) \quad + S_h = 0 \tag{59}$$

Equation 56 expands the summation index convention, the variables have the same interpretation, and ϕ represents any scalar field.

63. The generalized stress tensor σ_{ij} is constituted of its laminar and Reynolds averaged contributions,

$$\sigma_{ij} = \mu E_{ij} - \overline{\rho u_i' u_j'} \tag{60}$$

and the integral of $E_{ij} = \partial u_i/\partial x_j + \partial u_j/\partial x_i$ yields the surface stresses explicitly expressed in Equations 40-42. The Reynolds stress $\overline{u_i' u_j'}$, as well as the diffusion flux d_j in Equation 58, would be expressed in a gradient-diffusion law statement, i.e.,

$$\overline{-u_i'u_j'} = \nu_{jk}^t\left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i}\right), \; d_j = - k_{ij}\,\frac{\partial \phi}{\partial x_i} \tag{61}$$

where ν_{jk}^t is a turbulent eddy viscosity tensor and k_{ij} is the corresponding diffusion tensor for ϕ. The terms s_ϕ and s_h are the corresponding source terms, and f(a,h) is a functional form to be established that contains the differentiability required in conversion of Equation 55 to 59. Once this is completed, the 3D hydrostatic form of the governing equation system can be compactly stated as,

$$L(q) = \frac{\partial q}{\partial t} + \frac{\partial f_j}{\partial x_j} + s = 0 \tag{62}$$

where the dependent variable set is $q = (\rho w, \rho u, \rho v, \rho\phi, h)$. The exact forms for the corresponding flux vectors f_j, and source term s would be derived. Equation 62 expresses the governing equation system in the preferred conservation law statement form.

Transformation for Variable h

64. The RMA-10 computer program employs a coordinate transformation to regularize the solution domain and eliminate the explicit appearance of a time-varying upper surface (h) location. Figure 3 illustrates the transformation, which is defined by King (1982) as

$$h(x,y,t)z' = z - a(x,y) \;,\; x' = x \;,\; y' = y \tag{63}$$

Thus,

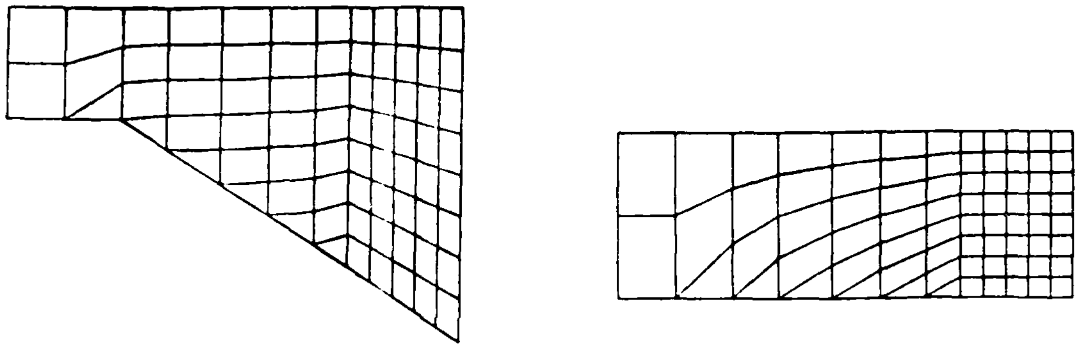
$$\frac{\partial u}{\partial x}(x,y,z(z')) = \frac{\partial u'}{\partial x'}(x',y',z') - \frac{1}{h}\left(\frac{\partial h}{\partial x}z' + \frac{\partial a}{\partial x}\right)\frac{\partial u'}{\partial z'} \tag{64}$$

$$\frac{\partial u}{\partial y}(x,y,z(z')) = \frac{\partial u'}{\partial y'}(x',y',z') - \frac{1}{h}\left(\frac{\partial h}{\partial x}z' + \frac{\partial a}{\partial x}\right)\frac{\partial u'}{\partial z'} \qquad (65)$$

$$\frac{\partial u}{\partial z}(x,y,z(z')) = \frac{1}{h}\;\frac{\partial u'}{\partial x'}(x',y',z') \qquad (66)$$

$$\frac{\partial u}{\partial t}(x,y,z(z')) = \frac{\partial u'}{\partial t}(x',y',z') - \frac{1}{h}\left(\frac{\partial h}{\partial t}z'\right)\frac{\partial u'}{\partial z'} \qquad (67)$$

where $u' = u'(x',y',z'h+a)$. Similar expressions are generated for v' and w', which when substituted into the vertically integrated continuity equation 55 yields,



a) Domain Grid In Physical Space

b) Transformed Solution Domain

Figure 3. TABS-3 Solution Domain Transformation, From King (1982, Figure 2.2)

$$\int_0^1 \left\{\frac{\partial u'}{\partial x'} - \frac{1}{h}\left(\frac{\partial h}{\partial x}z' + \frac{\partial a}{\partial x}\right)\frac{\partial w'}{\partial z} + \frac{\partial v'}{\partial y'} - \frac{1}{h}\left(\frac{\partial h}{\partial y}z' + \frac{\partial a}{\partial y}\right)\frac{\partial v'}{\partial z'}\right\} h\,dz' \qquad (68)$$

$$+ u'(x',y',1)\frac{\partial(a+h)}{\partial x} - u'(x',y',0)\frac{\partial a}{\partial x} + v'(x',y',1)\frac{\partial(a+h)}{\partial y}$$

$$- v'(x',y',0)\frac{\partial a}{\partial y} + \frac{\partial h}{\partial t} = 0$$

65. After integrating and proceeding through the remaining equations, expressing all variables in the primal system then yields the final 3D governing equation system in RMA-10 as

$$\int_0^1 \left(\frac{\partial}{\partial x}(hu') + \frac{\partial}{\partial y}(hv')\right)dz' + \frac{\partial h}{\partial t} = 0 \tag{69}$$

$$\rho h\frac{\partial u'}{\partial t} + \rho hu'\frac{\partial u'}{\partial x'} + \rho hv'\frac{\partial u'}{\partial y} + \rho w'\frac{\partial u'}{\partial z'} - \rho(u's_{x'} + v's_{y'} + z'\frac{\partial h}{\partial t})\frac{\partial u'}{\partial z'} \tag{70}$$

$$- h\frac{\partial}{\partial x'}[\varepsilon_{x'x'}\frac{\partial u'}{\partial x'}] - h\frac{\partial}{\partial y'}[\varepsilon_{x'y'}\frac{\partial u'}{\partial y'}] - \frac{\partial}{\partial z'}(\frac{\varepsilon_{x'z'}}{h}\frac{\partial u'}{\partial z'}) - \frac{\partial p}{\partial x} - \tau_{x'} = 0$$

$$\rho h\frac{\partial v'}{\partial t} + \rho hu'\frac{\partial v'}{\partial x'} + \rho hv'\frac{\partial v'}{\partial y'} + \rho w'\frac{\partial v'}{\partial z'} - \rho(u's_{x'} + v's_{y'} + z'\frac{\partial h}{\partial t})\frac{\partial v'}{\partial z'} \tag{71}$$

$$- h\frac{\partial}{\partial x}[\varepsilon_{y'x'}\frac{\partial v'}{\partial x'}] - h\frac{\partial}{\partial y'}[\varepsilon_{y'y'}\frac{\partial v'}{\partial y'}] - \frac{\partial}{\partial z'}(\frac{\varepsilon_{y'z'}}{y}\frac{\partial v'}{\partial z'}) - \frac{\partial p}{\partial y} - \tau_{y'} = 0$$

$$\frac{\partial h}{\partial t} + \int_0^1 \{\frac{\partial}{\partial x'}(hu') + \frac{\partial}{\partial y'}(hv')\}\,dz' = 0 \tag{72}$$

$$h\frac{\partial T}{\partial t} + hu'\frac{\partial T}{\partial x'} + hv'\frac{\partial T}{\partial y'} + w'\frac{\partial T}{\partial z'} - (u's_{x'} + v's_{y'} + z'\frac{\partial h}{\partial t})\frac{\partial T}{\partial z'} \tag{73}$$

$$- h\frac{\partial}{\partial x'}(D_{x'}\frac{\partial T}{\partial x'}) - h\frac{\partial}{\partial y'}(D_{y'}\frac{\partial T}{\partial y'}) - \frac{\partial}{\partial z'}(D_{z'}\frac{\partial T}{\partial z'}) - h\theta_s = 0$$

Equations 70-73 are completed with the state equation $\rho=\rho(S,T)$, and after transformation,

$$p = \int_{z'}^1 \rho g h \, dz' \tag{74}$$

The prime notation on the eddy diffusion coefficients indicates their approximate similarity to the original coefficients.

66. King (1982) notes that a price is paid in terms of mathematical consistency for the domain coordinate transformation. Specifically, parallel flow at the top or bottom of the new system transforms to flow parallel to the surface in the original system. Thus, at slope changes in the bottom profile there is a discontinuity in the flow trajectory and in the magnitude of the

vertical component. For example, (see Figure 3a), the vertical velocity is zero in the left-hand element and finite in the right-hand element. Thus, to record velocities at nodes for output requires averages be taken and slight inconsistencies are generated. Vertical steps (or sharp changes) in the bottom profile can only be treated by approximate boundary condition specifications, and in general the approach is unable to accurately describe such a system. Flow separation is not restricted, however, and the coordinate transformation does ensure global satisfaction of continuity.

67. Definition and use of this coordinate transformation requires a close examination, to be conducted in this second step analysis, since realistic bed profiles containing fairly sharp changes in a(x,y) will destabilize a RMA-10 computation. (Certainly to proceed to a vectorization with this intrinsic limitation does not make sense in the longer term.) The approach would be to examine the generalized (body-fitted) coordinate transformation concepts employed throughout aerodynamics (see Thompson, et al, 1985) to the developed conservation law statement form ie., Equation 62. The functional form is essentially identical to Equation 63, ie., $x'_i = x'_i\ (x_j, t)$, where the time-dependence is contained in h=h(x,y,t). The basic step is to re-express the derivative operators in the conservation law statement into the primed coordinate system. Quite simply, this is no more than

$$\left(\frac{\partial q}{\partial t}\right)_{x'_i} = \left(\frac{\partial q}{\partial t}\right)_{x_j} + \frac{\partial q}{\partial x_j}\left(\frac{\partial x_j}{\partial t}\right)_{x'_i} \tag{75}$$

where the bracket with subscript notation implies derivatives with the corresponding variable held fixed. That is, the left side of Equation 75 is the time derivative in transformed space where the surface $h(x_i)$ appears fixed (in time). The time derivative on the right is as seen in physical space, and in the third term $(\partial x_j/\partial t)'_{xi} = \dot{x}_j$ is the (grid) speed of the physical coordinate system (nodes) as seen in the fixed (x'_i) system. Hence, for Equation 62

$$\frac{\partial q}{\partial t} \rightarrow \left(\frac{\partial q}{\partial t}\right)_{x_j} = \left(\frac{\partial q}{\partial t}\right)_{x_i} - \dot{x}_j \left(\frac{\partial q}{\partial x_j}\right) \tag{76}$$

$$\frac{\partial f}{\partial x_j} \rightarrow \left(\frac{\partial f}{\partial x_j}\right)_{x'_i} = \left(\frac{\partial f}{\partial x'_i}\right)_{x'_i} \frac{\partial x'_i}{\partial x_j}$$

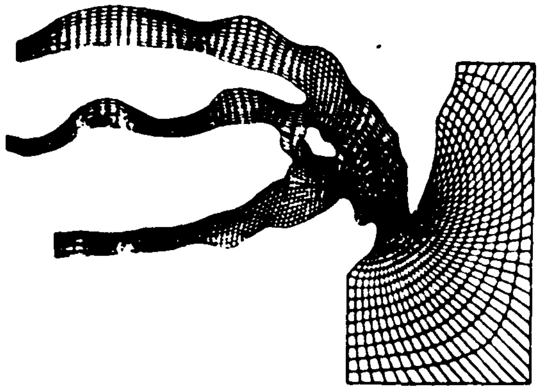
Re-expressing the second derivative in the first line of Equation 76 in terms of x'_i , the accounting for the variable lid $h=h(x,y,t)$ is then intrinsically embedded in the conservation law statement when rewritten as,

$$L'(q) = \frac{\partial q}{\partial t} + \left(\frac{\partial f}{\partial x'} - \dot{x}\,\frac{\partial q}{\partial x'}\right)\left(\frac{\partial x'}{\partial x}\right) + s = 0 \qquad (77)$$

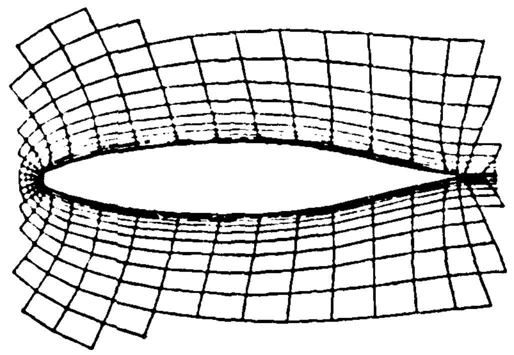
68. The specific form for Equation 77 must be derived for arbitrary (within the validity of the hydrostatic assumption) surface forms $h=h(x, t)$. The net output will be the specific form for the grid velocity x_i ; whereupon Equations 77 and 70-73 can be exactly compared term by term. Of importance, no a priori constraint is implied on variation of bed profile $a(x,y)$. Once established, the form of Equation 77 is ideally suited to the next two steps.

Body Fitted Coordinate Transformations

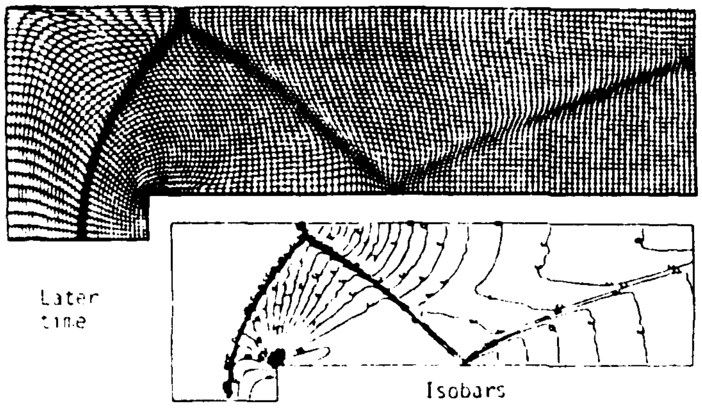
69. This item is the significant third step upon completion of step 2. Finite elements have always been viewed for their geometric versatility, ie., one can readily triangularize (or discretize into unions of tetrahedra, pentahedra and hexahedra in 3D) any region with geometrically-complicated boundary. Figure 3 gives an elementary example. However, especially for a moving boundary, and in general, highly structured coding procedures are one natural consequence of embedding a generalized coordinate transformation to the conservation law statement before writing the finite element algorithm statement. Equation 77, as the successor to eqn 62, is already written in the desired form which is therefore immediately applicable for use with any (body-fitted) coordinate transformation. For example, the finite element algorithm statement for Equation 77 would be immediately applicable to analysis of Charleston Harbor, Figure 4a, or to flow past an airfoil, Figure 4b. Conversely, it would also be immediately extensible to use with adaptive mesh procedures to place finer griddings in regions of sharp solution gradients, Figure 4c.



a) Charleston Harbor

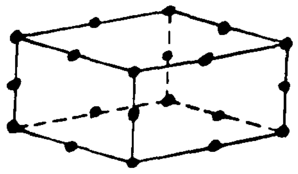


b) Aerodynamic Profile

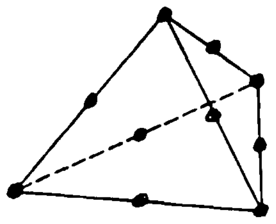


c) Solution - Adaptive Gridding

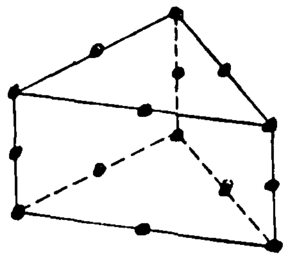
Figure 4. Examples of Body-Fitted Coordinate Transformations for which Eqn. 77 is Applicable, from Thompson, et. al. (1985).



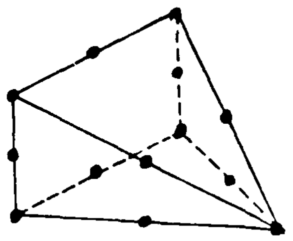
a) Brick



b) Tetrahedron



c) Prism



d) Pyramid

Figure 5. Finite Elements Used in the TABS-3 Computer Program, ( ) Denotes Node Locations, from King (1982).

70. The formulational details of the body-fitted coordinate transformation implementation must be completed. The Galerkin weak-statement finite element algorithm (the theoretical basis for RMA-10) form of Equation 77 is well known. Specifically, q is approximated by a projection onto functions generated from a suitable space of trial functions $S_h cH_1$ yielding,

$$q = q^h(x_i,t) = \sum_j \psi_j(x_i)\ Q_j(t) \tag{78}$$

or conversely, expressed in terms of the cardinal basis $\{N_k(\eta_j)\}$ over the union of elements of the discretization,

$$q^h = \underset{e}{U}\ \{N_k(\eta_j)\}^T\ \{Q(t)\}_e \tag{79}$$

71. Figure 5 shows the elements available in the RMA-10 program, for which the cardinal bases $\{N_k(\eta_j)\}$ are well known (and in which η_j is a local coordinate system). The Galerkin weak statement for Equation 77 is then,

$$S_e \int_{\Omega_e cR^n} \{N_k\}\ L(q^h)dx = \{0\} \tag{80}$$

where Ω_e is the generic finite element domain, recall Figure 5. For k=2 , the quadratic basis is defined, as is utilized in RMA-10 for the two transverse momentum equation, while k=1 defines the linear basis as used by RMA-10 for the integral continuity equation. Neglecting for the moment the grid velocity term, and using for example the scalar convection-diffusion equation (58), and anticipating the embedding of h, the expanded form of Equation 80 is,

$$S_e \int_{\Omega_e} \{N_k\} \left[\frac{\partial(\rho h\phi)}{\partial t} + \frac{\partial}{\partial x'_j}\left(u_i \rho h\phi - k_j\ \frac{\partial\phi}{\partial x_i}\right)\left(\frac{\partial x'_j}{\partial x_i}\right)\right]^h dx \tag{81}$$

72. In Equation 81, the superscript "h" on [ ] denotes use of Equation 79, and S_e is the familiar finite element assembly operator. Using the Green-Gauss form of the divergence theorm in Equation 81 to transport the functional support for $\partial/\partial x'_j$ yields the flux vector term as,

$$- S_e \int_{\Omega_e} \frac{\partial}{\partial x'_j} \{N_k\} \left(u_i \rho h \phi - k_{\underline{j}} \frac{\partial \phi}{\partial x'_k} \frac{\partial x'_k}{\partial x_i} \right)^h \left(\frac{\partial x'_j}{\partial x_i} \right) dx \tag{82}$$

73. The basic form of the diffusion term is immediately recognized, where k_j is the directional coefficient. One can recognize that $\partial\{N_k\}/\partial x'_j$ is a direct operation, and the key step is to use $\{N_k\}$ to interpolate the coordinate transformation on the (each) element domain Ω_e . Denoting the (global) node coordinates as the arrays $\{X\}_e$, $\{Y\}_e$ and $\{Z\}_e$, ie., $\{XI\}_e$, and using Equation 79, yields the local coordinate transformation

$$x_i = \{N_k(\eta_j)\}^T \{XI\}_e \tag{83}$$

As detailed in Baker (1983, Ch. 5), the Jacobian of the forward transformation on Ω_e is readily evaluated as a rank n square matrix. The Jacobian of the inverse transformation can be formed analytically, thus yielding the form,

$$\left(\frac{\partial x'_j}{\partial x_i} \right)_e \rightarrow \frac{1}{\det J'} \Big(\text{ETAJI} \Big)_e \tag{84}$$

where the $1 \leq (I,J) \leq n$ entries in the array $(\text{ETAJI})_e$ involve distributed differences in the entries in $\{XI\}_e$.

74. Noting that $dx = \det J'\, dx'$, and denoting the (contravariant) components of the velocity field u_i^h , that are parallel to the coordinate curves of the x'_j coordinate system, as $\bar{u}_j^h$, the first term in Equation 82 for a typical element becomes

$$- \int_{\Omega_e} \frac{\partial\{N\}}{\partial x'_j} (\rho h \phi)^h\, u_i^h \left(\frac{\partial x'_j}{\partial x_i} \right)_e \det J' dx' = - \int_{\Omega_e} \frac{\partial\{N\}}{\partial x'_j} \Big(\{\text{UBARJ}\}^T \{N\} \Big) \{N\}^T \{\text{RHPHI}\}_e dx' \tag{85}$$

The integrals defined in Equations 85 are evaluable using standard techniques. Det J' has cancelled out. $[\text{UBARJ}]_e$ contains the nodal values of velocities parallel to the x'_j coordinate system (hence, parallel to the free surface and $\{\text{RHPHI}\}_e$ contains the nodal values of $(\rho h\phi)_e$. The diffusion term involves (outer) products in $(\text{ETAJI})_e$ $(\text{ETALI})_e$/det J', and its evaluation is straightforward. This outlines the generalized coordinates formulation for Equation 77.

Enhanced FE Algorithm Stability

75. The finite element algorithm upon which RMA-10 is constructed is the Galerkin weak statement, ie., the Galerkin criteria for the weight functions in a weighted residual statement, as expressed in Equation 80. It has become well verified in the last few years that the Galerkin formulation is very limited in terms of stability for problem definitions with significant fluid convection, ie., large Reynolds-Peclet number fluid-thermal flows. Experience in using the RMA codes at WES for certain problem simulations, especially those with significant bed profiles, confirms that destabilizing mechanisms can become evident and a problem. One "numerical cure" for this is to increase the "eddy viscosity" coefficient, the essence of which is to reduce the computational Reynolds number to that required for Galerkin stability. Unfortunately, this severely compromises the ability to simulate flows with realistic Reynolds numbers. One goal in the reformulation would be to moderate the stability perturbations induced by geometrically-significant bed profiles as well as the dynamic free surface. The next step of this reformulation analysis is to thus enhance algorithm stability performance in a robust and mathematically consistent way.

76. The numerous artificial diffusion (viscosity) methods devised for finite difference CFD methods, as well as essentially all of the comparison finite element constructs, eg., Petrov-Galerkin, Taylor-Galerkin, characteristic-Galerkin, penalty-Galerkin,.., have been firmly established as belonging to the family of Taylor weak statements (TWS) by Baker and Kim (1987). Rather than arbitrarily adding diffusion-type terms to the algorithm statement, Equation 80, the TWS procedure generates appropriate functional expressions for any specific conservation law statement, e.g. Equation 77. Realizing that the coordinate transformation is a formulational detail, neglecting the associated

notational complication, expand the parent expression (Equation 62) in a Taylor series,

$$q^{n+1} = q^n + \Delta t\, q_t^n + 1/2\Delta t^2 q_{tt}^n + \frac{1}{6}\Delta t^3 q_{ttt}^n + \dots \tag{86}$$

where subscript "t" denotes order of temporal derivative at t_n and $t_{n+1} = t_n + \Delta t$. Equation 62 allows restatement of the time derivatives in Equation 86, neglecting the source term s, as

$$q_t = -f_x = -f_q q_x = -Aq_x \tag{87}$$

$$q_{tt} = -f_{xt} = -f_{tx} = -(f_q q_t)_x = (f_q f_x)_x = \left(\bar{a}Aq_t + \bar{\beta}\, Af_x\right)_x \tag{88}$$

$$q_{ttt} = -f_{xtt} = (-Aq_x)_{tt} = (AAq_x)_{xt} = (AAq_x)_{tx} = \left[\bar{\gamma}\left(A^2 \frac{\partial}{\partial x} + (A^2)_x\right)q_t + \bar{\mu}\left(A^2 \frac{\partial}{\partial x} + (A^2)_x\right)f_x\right]_x \tag{89}$$

The matrix A is the Jacobian of the flux vector f , the coefficients $\bar{\alpha}$, $\bar{\beta}$, $\bar{\gamma}$, and $\bar{\mu}$ are arbitrary, to within a convex sum constraint, and their specification ultimately reflects choices to be made for the final functional form. Combining Equations 86-89 and collecting terms yields,

$$\frac{q^{n+1} - q^n}{\Delta t} = -f_x^n + \frac{\Delta t}{2}\left(\bar{\alpha}\, Aq_t + \bar{\beta}\, Af_x\right)_x^n + \frac{\Delta t^2}{6}\left[\bar{\gamma}\left((A^2)_x + A^2 \frac{\partial}{\partial x}\right)q_t \; \bar{\mu}\left((A^2)_x + A^2 \frac{\partial}{\partial x}\right)f_x\right]_x^n + \dots \tag{90}$$

The left side of Equation 90 expresses the discrete approximation to q_t . Assuming the limit is taken, Equation 90 can then be re-expressed as $\Delta t \rightarrow 0$ without temporal approximation (but retaining the higher order terms) as

$$L_w(q) = q_t - \frac{\Delta t}{2}\left(\bar{\alpha}\, A\, q_t\right)_x - \frac{\Delta t^2}{6}\left[\bar{\gamma}\left((A^2)_x + A^2 \frac{\partial}{\partial x}\right)q_t\right]_x$$
$$+ f_x - \frac{\Delta t}{2}\left(\bar{\beta}\, A\, f_x\right)_x - \frac{\Delta t^2}{6}\left[\bar{\mu}\left((A^2)_x + A^2 \frac{\partial}{\partial x}\right)f_x\right]_x + \ldots \tag{91}$$

77. Equation 91 is the conservation law form desired for constructing the finite element Taylor weak statement algorithm wherein Δt will not vanish. Importantly, Equation 91 is identically satisfied by the exact solution to Equation 62 for $\bar{\alpha} = \bar{\beta}$ and $\bar{\gamma} = \bar{\mu}$ upon reintroduction of the source term.

78. Constructing the finite element weak statement for Equation 91 follows the standard Galerkin finite element procedure. The approximation statements, Equation 78-79 are unchanged, and in Equation 80, $L(q_h)$ is replaced by $L_w(q_h)$. Letting $\{\cdot\}$ denote the global array of (Galerkin) weighting functions $\{N\}$, replacing f_x everywhere by Aq_x , and reintroducing the source term s , the TWS algorithm replaces Equation 80 with the expression,

$$\int_{\Omega\subset R^n} \{\Psi\}\left(q_t^h + A^h q_x^h + s^h\right)dx + \Delta t \int_{\Omega} \{\Psi\}_x \tilde{A}^h\left(\bar{\alpha} q_t^h + \bar{\beta} A^h q_x^h\right)dx$$
$$+ \Delta t^2 \int_{\Omega} \{\Psi\}_x (\tilde{A}^h)^2\left(\bar{\gamma} q_{tx}^h + \bar{\mu} A^h q_{xx}^h\right)dx = \{0\} \tag{92}$$

The first term in Equation 92 is the classic Galerkin statement, while over a dozen (dissipative) FD and PE algorithms are recognized (solely) by the designer's choice for the approximation (A^h) to the Jacobian and the coefficient set $\bar{\alpha}$, $\bar{\beta}$, $\bar{\gamma}$, $\bar{\mu}$ in the second term, cf., Baker and Kim (1987).

79. Figure 6 summarizes numerical results published by Baker and Kim (1987) for sample 1D hyperbolic problems of unsteady and steady-state solutions for linear and non-linear inviscid test problems. The implicit Galerkin algorithm sets the last two terms in Equation 92 to zero (as utilized for RMA-10), and propagates the linear unsteady sine wave with little diffusion but significant dispersion error (trailing wake), Figure 6a. The steady

linear wall-layer solution for Pe=180, Fig. 6b, confirms cascading of the dispersion error into the $2\Delta x$ oscillations intrinsic to Galerkin methods. In the non-linear stationary shock, Figure 6c, this basic instability cascades non-linearly to turn the solution to trash. In distinction, the Euler-Taylor-Galerkin (ETG) algorithm of Donea (1984), corresponding to $\bar{\alpha}=0=\bar{\mu}$ and $\bar{\beta}=1=\bar{\gamma}$ in Equation 92, does an excellent job of controlling dispersion error for the unsteady linear travelling wave, Figure 6d. Conversely, the penalty-Galerkin (pG) algorithm of Baker (1983), $\bar{\alpha}=1/2=\bar{\beta}$ and $\bar{\gamma}=0=\bar{\mu}$ in Equation 92 does an excellent job for the steady-state linear wall-layer and steady-state non-linear shock, Figure 6e-f.

80. This step will develop and derive the TWS finite element algorithm for the hydromechanics conservation law restatements as derived before. The goal is to establish a mathematically robust, highly phase-selective dissipative Galerkin algorithm and to thus identify needed modifications to the RMA-10 algorithm (successor) to permit consistent stabilization for simulations with large convection velocities and/or significant bed profiles.

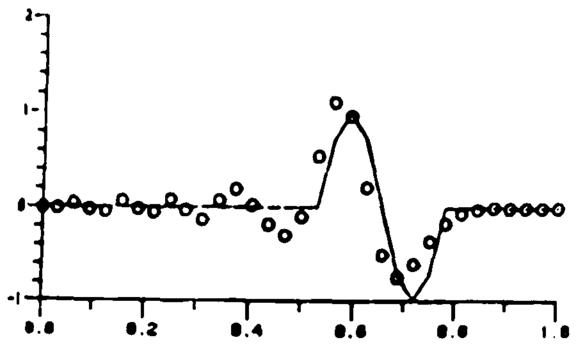
The Linear Algebra Statement for Supercomputing

81. Step 4 completes the detailed derivation of the TWS finite element algorithm form, Equation 92, for recasting, extension and/or total restatement of the RMA-10 algorithm. While this involves great detail, upon completion of the indicated integrals, Equation 92 ultimately becomes an ordinary differential equation written for the time evolution of the expansion coefficient set {Q}, Equation 79, of the form:

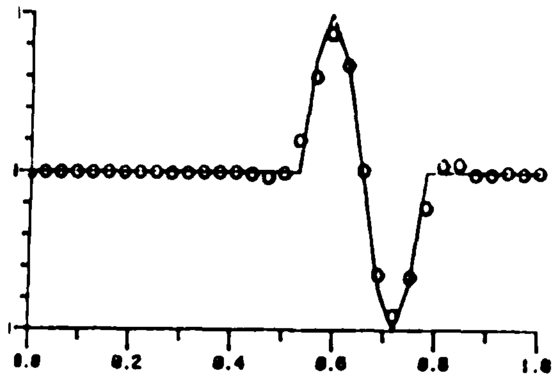
$$L(\{Q\}) = M\frac{d\{Q\}}{dt} + \{E^h - E^h_v\}_\xi + \{F^h - F^h_v\}_\eta + \{G^h - G^h_v\}_\zeta + S^h - \Delta t\Big(E^h(\alpha,\beta,\tilde{A})_{\xi\eta}$$

$$F^h(\alpha,\beta,A)_{\eta\eta} + \{G^n(\alpha,\beta,\tilde{A})\}_{\xi\eta}\Big) - \Delta t^2(\{E^h(\alpha,\beta,A^2)_{\xi\eta} + \ldots\})_\eta \quad (93)$$

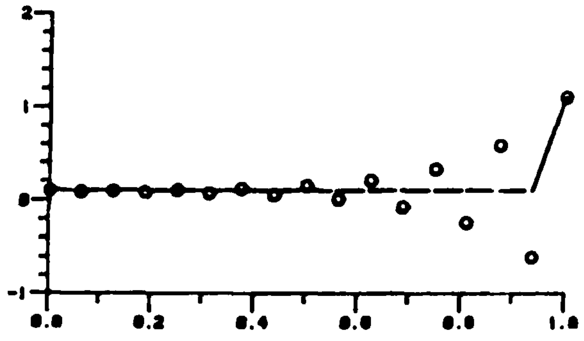
The first line of terms in Equation 93 constitutes the conventional Galerkin weak statement for the governing Equation 77, including the grid speed and source terms, and expressed in the (ξ,η,ζ) transformed coordinates. The second and third groups of terms constitute the Taylor weak statement



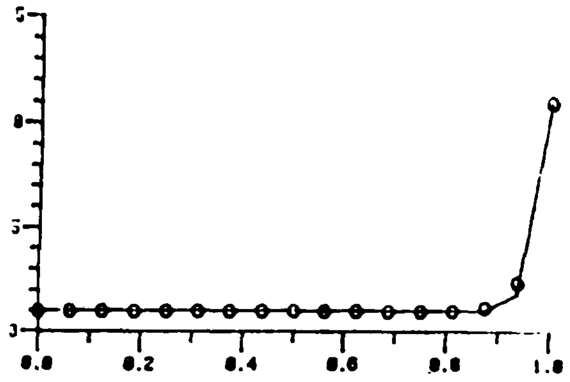
a) Galerkin, unsteady



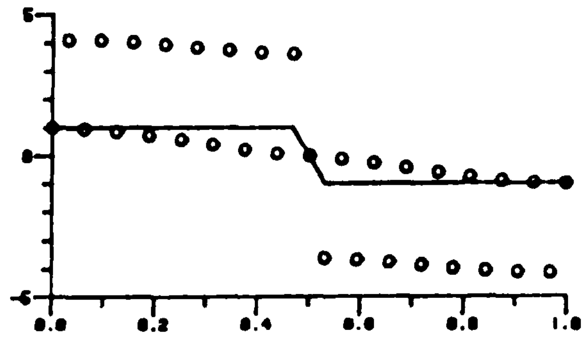
d) Taylor-Galerkin, unsteady



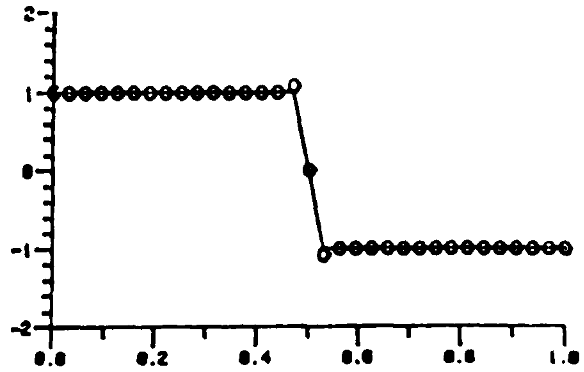
b) Galerkin, steady



e) Penalty-Galerkin, steady



c) Galerkin, steady non-linear



f) Penalty-Galerkin, steady non-linear

Figure 6. Comparison of Various Finite Element & Algorithm Solutions to Hyperbolic Test Problems, from Baker and Kim (1987)

additions for embedding stability mechanisms according to the definitions for $\bar{\alpha}$, $\bar{\beta}$, $\bar{\gamma}$, $\bar{\mu}$ and A .

82. The TWS finite element algorithm is ultimately reduced to a linear algebra statement for coding design for a supercomputer environment. For notational simplicity, functionally combining terms in Equation 93 yields the ODE matrix statement

$$M\,(\bar{\alpha},\bar{\gamma})\,\frac{d\{Q\}}{dt} + \{R\,(\{Q\},\,\bar{\beta},\,\bar{\mu},\,\{S\},\,A,\,\Delta t)\} = \{0\} \tag{94}$$

The specific construction for $M(\cdot)$ and $\{R(\cdot)\}$ depends explicitly on the choice of the trial space basis ($\{N_k\}$) and the TWS parameter choices. Once selected, Equation 94 permits evaluation of the time-derivative of the discrete Taylor series,

$$\{Q\}_{n+1} = \{Q\}_n + \Delta\,t\,\frac{d\{Q\}}{dt}\bigg|_{n+\theta} \tag{95}$$

where $1/2 \leq \theta \leq 1$ for the required implicit algorithm. Substituting Equation 94, arranging terms and writing the resultant in homogeneous form yields the final linear algebra statement.

$$\{F\} = \quad M\,\{\Delta Q\} + \Delta t\left(\theta\{R\}_{n+1} + (1-\theta)\,\{R\}_n\right) \tag{96}$$

where $\{\Delta Q\} = \{Q\}_{n+1} - \{Q\}_n$ and subscript (n+1, n) denotes evaluation at the corresponding time level. Linearizing around t_n and dividing through by Δt yields the Taylor weak statement computational linear algebra statement in delta form as,

$$\left[\frac{1}{\Delta t}\,M + \theta\,\frac{\partial\{R\}}{\partial\{Q\}}\right]\{\Delta Q\} = -\,\{R\}_n \tag{97}$$

83. The form of Equation 97 is ideally suited to rapid determination of the steady-state solution, as well as time-accurate predictions of the transient. The solution converges when $\{R(\cdot)\} \cong \{0\}$, hence depends only on the Taylor weak statement parameters $\bar{\beta}$, $\bar{\mu}$, and A . The rate of approach to

steady-state depends on $\bar{\alpha}$, $\bar{\beta}$, $\bar{\gamma}$, $\bar{\mu}$, Δt and θ, and in the limit $\Delta t \rightarrow \infty$ Equation 97 is a Newton algorithm. The choice of an implicit integration procedure is crucial to the algorithm, and a wide range of choices affecting stability and convergence rate exists within the Taylor weak statement.

84. Equation 97 ultimately resides in the (supercomputer) code, and the matrix iterative solution procedure to be selected is crucial to overall efficiency, hence utility of the Taylor weak statement finite element algorithm. The RMA-10 code now uses a decoupled full-dimensional approximation to the Jacobian [M/Δt + ...], in Equation 97, which is a candidate for replacement in a supercomputer implementation. The embedded coordinate transformation has conveniently established "generalized implicit lines (GILS)", cf., Lin (1985), upon which an efficient factorization of the Jacobian can be constructed. Therefore, the appropriate procedure appears to be a tensor matrix product approximation to the Jacobian, whereupon the left side of Equation 97 is replaced by a factored form

$$J \equiv \left[\frac{1}{\Delta t} M + \theta \frac{\partial\{R\}}{\partial\{Q\}} \right] \approx J_\xi \cdot J_\eta \cdot J_\zeta \tag{98}$$

The factors J_ξ and J_η, spanning the transverse (horizontal) plane will be classical constructions, i.e.,

$$J_\eta \equiv \left[M_\eta + \theta \frac{\partial\{R\}}{\partial\{Q\}} \eta \right] \qquad J_\xi \equiv \left[M_\xi + \theta \frac{\partial\{R\}}{\partial\{Q\}} \xi \right] \tag{99}$$

each of which is block (2k+1) diagonal, where k is the completeness degree of the finite element trial space basis $\{N_k\}$. The vertical direction factor J_ζ can be arranged to account for the continuity equation integration, but otherwise will be the classical formulation, i.e.,

$$J_\zeta \equiv \left[\frac{1}{\Delta t} M_\zeta + \theta \frac{\partial\{R\}_\zeta}{\partial\{Q\}_\zeta} \right] \tag{100}$$

85. This step completes the definitions of the Jacobian factorization, Equations 98-100, as a function of A and $\bar{\alpha}$, $\bar{\beta}$, $\bar{\gamma}$, $\bar{\mu}$ and θ of the TWS algorithm statement.

Design for the Supercomputer Environment

86. The first five steps have lead to this critical point, which anticipates that a major redesign of the RMA-10 linear algebra solution procedure is critical to an efficient supercomputer embodiment (code) of the algorithm. It is important to emphasize that this redesign focuses on the very internal workings only of the code, ie., the computationally dense DO loops over elements that form Equation 96 and manage the (proposed) solution steps, Equations 97-100. The entire "external" hierarchy regarding input/output, the element library, grid generation, graphics, etc., can be retained unchanged. It is well understood that these portions constitute the major fraction of any working code and that they remain essentially intact (on the host computer) while the intense number crunching operations are moved over to the supercomputer environment.

87. A second important point regarding step 5 is that the suggested use of tensor matrix factorization of the linear algebra Jacobian in no way limits the geometric generality of the finite element algorithm. Because of the hydrostatic assumption, leading to solution of the continuity equation in the z-direction as an initial-value problem, there is a corresponding organization to the 3D mesh, recall Figure 3. This is recognized in Equation 100, and there is no requirement that these columns contain the same number of nodes, as is the case with an approximately-factored finite difference algorithm. The concept of tensor matrix products and sweeping on (curvilinear) paths is directly applicable to the 3D mesh equivalent of that illustrated in Figure 3b, with the embedded tri-angular (ie., tetrahedron/pentahedron) elements providing required geometry transitions along the bed profile as presently done in RMA-10. It is also important to note that the "vertical" coordinate lines need not be exactly parallel and/or vertical, since the continuity equation algorithm redesign will become re-expressed in terms of the contravariant velocity components which are always parallel to (transformed) coordinate curves. The tensor transformation law is then used to provide the required physical velocity components.

88. Step 6 identifies the key matching facets of the RMA-10 algorithm restatement with the supercomputer vector/multi-processing/parallel processing environment. For this, construction of Equation 96 and 97 readily generalizes on the concept of a general implicit line.

The two basic loops for solution of Equations 96 and 97 therefore, are:

1. Over dependent variables on N^3 grid cells, form {F}={F(3D geom, Q, , , ...) and}
2. For an N^3 grid and all variables on three general planes, form $J_i = J_i$ (1D geom, U_i, ...)
3. Solve A, B, {Q}=f ($\bar{J}^1{}_i$, data).

Thus, the linear algebra solution sequence defined by Equation 98 effectively sub-divides an N^3 problems into $3N^2$ problems of N length without loss of accuracy or connectivity, making the algorithm construction highly suited for concurrent (MIMD) parallel processing implementation.

89. In addition to parallelism derived from the factoring of Equation 97, parallelism also occurs throughout the algorithm at various levels including:

1. Independence of {F} and J_i in the same time period, as illustrated in Figure 7.
2. Parallel formation of J_1 and J_2 on different planes, as illustrated in Figure 8.
3. Simultaneous formation of equation terms for the {F} and J_i arrays. The terms are position independent in each differential equation.
4. Parallel and systolic methods for solving the formed system of equations.

90. Figure 7 is a schematic diagram of logic and data flow for the defined general implicit line algorithm. The processing direction is generally clock-wise beginning at the upper left. The vertical divisions (sweep 1,2,3) indicate operations performed in each sweep direction. The rectangular boxes are the operations performed and the bracketed symbols represent intermediate data temporarily retained for use in the next set of operations. In each sweep direction a loop over a row of the discretized geometry builds an equation system which is solved using Gauss elimination. At the end of sweep direction 3, Equation 98 is complete and a time step is taken proceeding from two to one. Closure data is evaluated at the new time step and the cycle is re-initiated.

91. Upon closer examination of the serial processes in Figure 7, and with an understanding of pipelining, it becames obvious that any serial set of operations can be made parallel through the use of pipelining (also defined as systolic, Kung, 1980). The simple requirement is that the processors involved

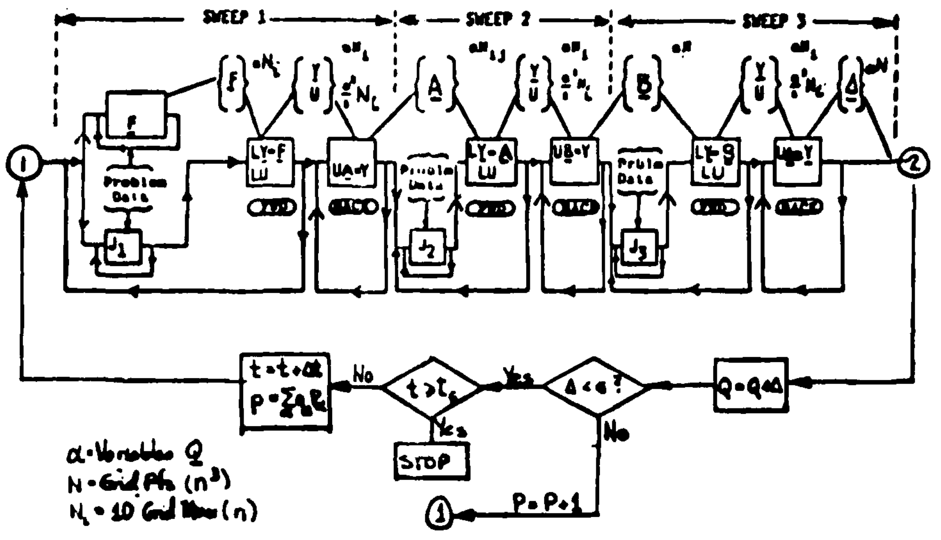


Figure 7. Algorithm Structure

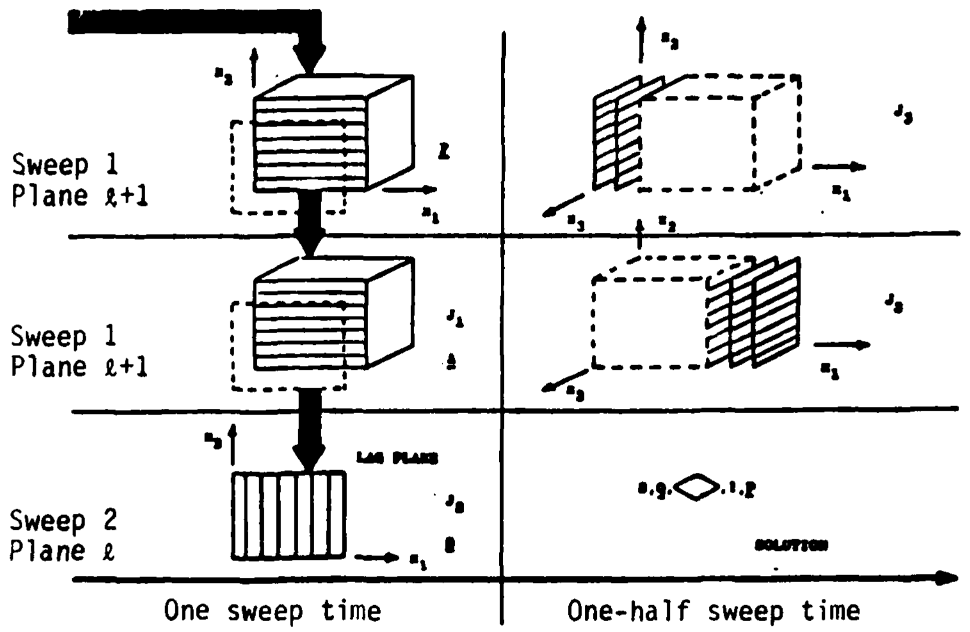


Figure 8. Planar Level Parallel Processing

are synchronized such that the output from pairs of processors forms the input to the next processor. The output can be bit, byte, word, or array sized, utilizing appropriate intermediate storage for proper synchronization. Hence, all processors are continuously busy working on the same job in different time slots of job completion.

92. Pipelining is illustrated in Figure 7. If $\{F\}$ and J_i are swept in planes normal to the x_3 direction, the completion of the first direction sweep in each plane produces the data $\{A\}$ required for the second direction sweep in that plane. Hence, the first and second direction sweeps, which are defined serially, can be made parallel through a processing time delay and use of an intermediate buffer array to hold the intermediate data.

93. Each differential equation term in $\{F\}$ and its derivative J consists of products and sums which are position independent. This means that the terms can be evaluated in parallel and finally combined for each gridpoint on a sweepline. Parallelism at this level has the capability to consume large quantities of problems data. The utility of this parallelism, therefore, is strongly dependent upon memory/processor speed ratio, data communication speed and the number of paths between memory and processors.

94. Finally, parallel processing of the equation solving process is a consideration. Due to its broad application arena, this area has received considerable attention by various researchers e.g., Ortega and Voight (1985). Also, as illustrated in Figure 9, the equation solving process represents only about 15% of the work required for a tensor product factored CFD solution. The formation of the coefficients and right side (F) account for 85%; therefore, this step concentrates on parallelism in the formation aspects of the $\{F\}$ and J_i components, and re-assembly of the data for an efficient, parallel, iterative solution.

Data organization

95. Local spatial coupling of data is an extremely important characteristic of numerical PDE algorithms. This coupling requirement complicates the separation of problem data for parallel processing. This suggests the formation of data in hierarchical packets with certain data repeated among the packets to meet the local coupling requirements. Packets at the top of the hierarchy would contain a few large groups of data with less repetition, while packets at the bottom of the hierarchy would be many and small and be more

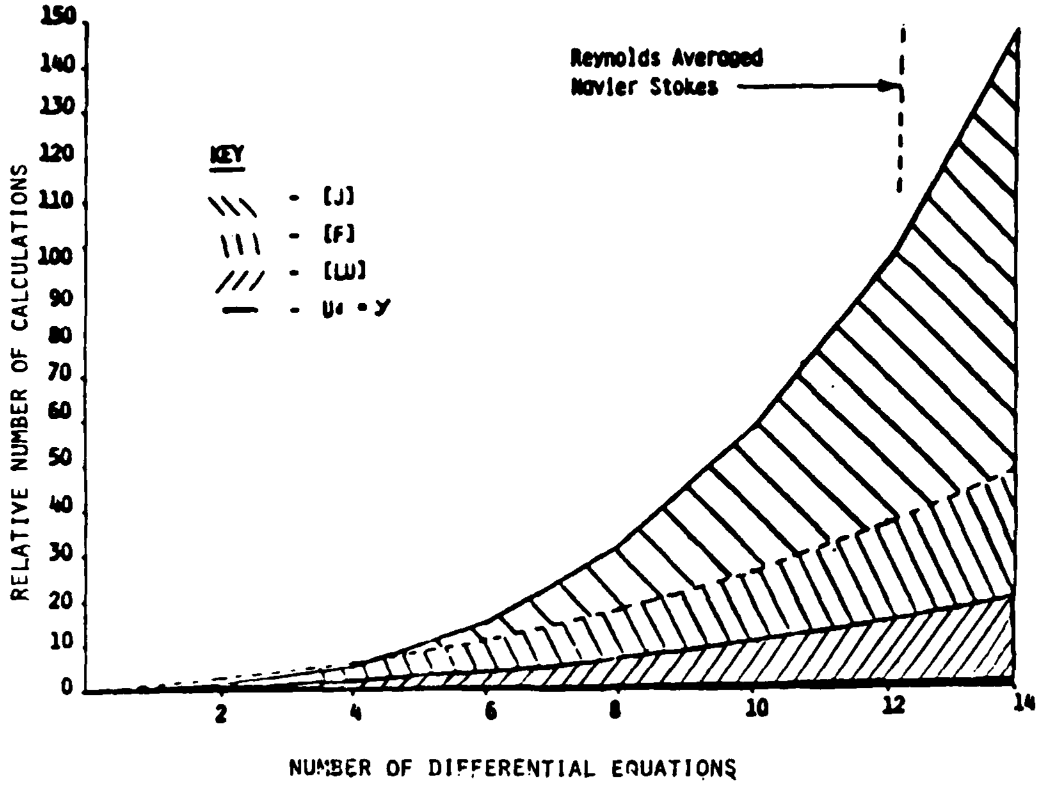
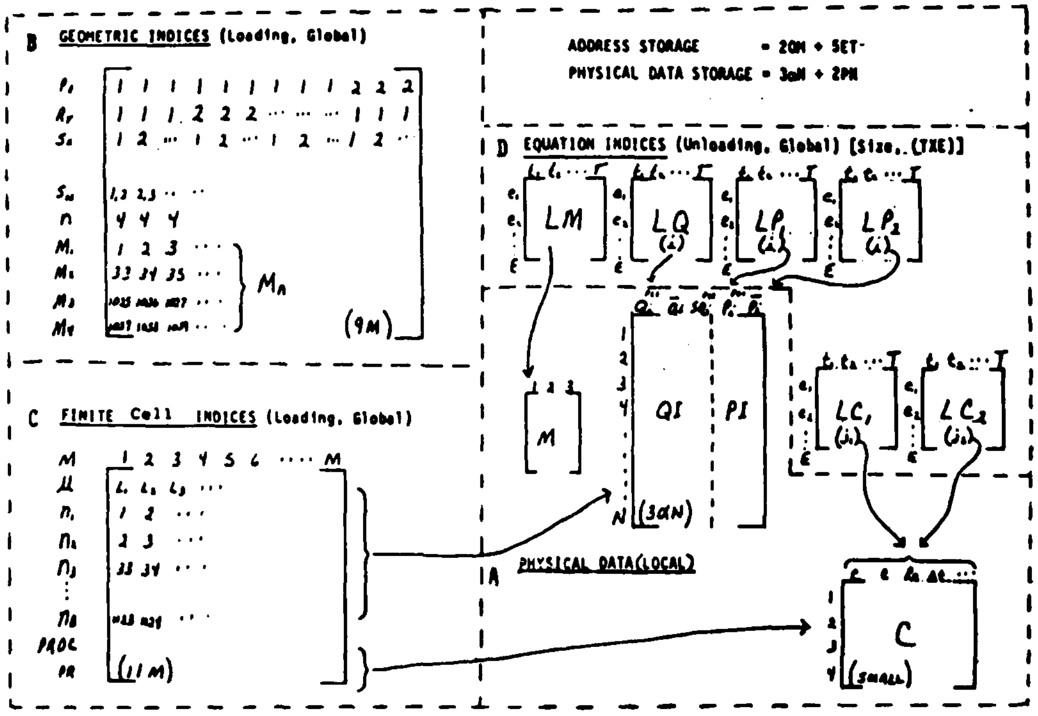


Figure 9. Computational Load Distribution



B GEOMETRIC INDICES (Loading, Global)
ADDRESS STORAGE = 20M + 5ET
PHYSICAL DATA STORAGE = 3αN + 2PN
D EQUATION INDICES (Unloading, Global) [Size, (TXE)]
LM
LQ
LP
LP₂
M
QI
PI
LC₁
LC₂
(9M)
Mₙ
C FINITE Cell INDICES (Loading, Global)
(11M)
(3αN)
A PHYSICAL DATA (LOCAL)
C
(small)

Figure 10. Data Structures (F Array)

repetitious. Hence, the data would be distributed among parallel machines for concurrent processing.

96. Full categorization of parallel and concurrent processing will require definition of the concurrency constraints and penalty imposed by data unavailability of each level of the hierarchy. Categorical subdivision of the data into specified problem data and partially processed data (intermediate data) provides some insight. The problem data is relatively stagnant, changing only occasionally toward the top of the hierarchy. Intermediate data, on the other hand, must be accumulated for portions of the algorithm at lower hierarchal levels. At each level, progressively smaller memories or registers (buffers) serve to hold data until it can be processed. Ideally, each buffer has a path to all processors requiring the memory data, and the data is "used up" requiring no further transfer.

97. Since data storage and communication are increasingly important factors as the number of processors increases, a system for decreasing data motion is beneficial for parallel processing. In a synchronous system, data must be available to the processors each cycle in order to maintain effective processor use. Data manipulation and overhead are the principal culprits in current supercomputer inability to maintain optimal speed. The need to continually rearrange data into long vectors for efficient pipeline operations wastes cycle time and adds a tremendous overhead burden to an algorithm.

98. One method for significantly reducing this problem for parallel processing is through use of indirect addressing pointers. The problem data is stored in a global set and pointers are generated for random data selection from the global set. Using this method, data manipulation is minimized with low overhead penalty and lower level packet data repetition can be minimized.

99. Figure 10 illustrates an indirect addressing scheme for the {F} array problem data which uncouples the global geometry and the physics. The problem description data is represented in the lower right of Figure 10(a) as two dimensional arrays. Four types of data are required for complete physical descriptions. These are: initial values of the dependent variables (Q), spatially distributed parameter coefficients (P), local field constants (C), and differential operator arrays (M). Each of the data categories are stored in two dimensional arrays for ROW/COLUMN addressing. The row pointers are derived from geometric indices (10b, 10c) which are a function of geometric

location (grid points). The column pointers (10d) select the differential equation variables.

Hardware variables and constraints

100. Computer hardware architectures for parallel processing are as diverse as algorithm variations in CFD. Each architectural design has its strong and weak points, each of which may be dependent upon the specific application involved. It appears, however, that there are certain design criteria which can be utilized to determine the ultimate effective speed of a machine.

101. As computational chip speeds increase, it is becoming increasingly apparent that the effectiveness of a parallel architecture is ultimately measured in the ability to communicate the data from memory to processor and between processors. In a parallel system, the limit of the number of processors effectively used is determined by the ability to get the data to the processor at every time cycle. The inability to do this wastes processor cycles and, therefore, compromises solution speed.

102. Assuming two inputs and one output are required for each processor operating from a single memory bank, for example, the memory access speed to keep six processors busy must be at least 18 times the speed of each processor. If the memory access speed is only 12 times the speed of each processor, for example, a bottleneck will result and processor cycles will be wasted waiting for the next data to arrive.

103. Some of these machine limitations, however, can be circumvented at the algorithm level. In the above example, the problem might be altered to allow for four of the processors to operate in parallel on data from memory and the other two processors to operate on data from the output of the four processors in pipeline fashion. This would reduce the memory access requirements from 18 to 10 accesses per processor cycle, thus keeping all processors continuously busy. The limiting factor for pipelines, however, is pipeline initiation/end cost which in this case is two process cycles.

104. By knowing the hardware memory and processor speeds, together with communication links, therefore, algorithm components may be made to keep the hardware busy. Since these relative numbers of hardware components are different for different machines, the resulting detailed algorithm structure will appear differently on each machine.

105. Each of these precepts was independently analyzed to construct the

COMSYS (TM) aerodynamics algorithm, resulting in a prototype vector/parallel processing structure. This was then coded (in Fortran) for testing and verification runs on a serial computer, ie., a Cyber-170/835. This machine then serves as the host for an echos simulation of the assembly language code, as well as the actual execution on a parallel processor which in practice to date has been a Control Data Cyberplus attached parallel processor.

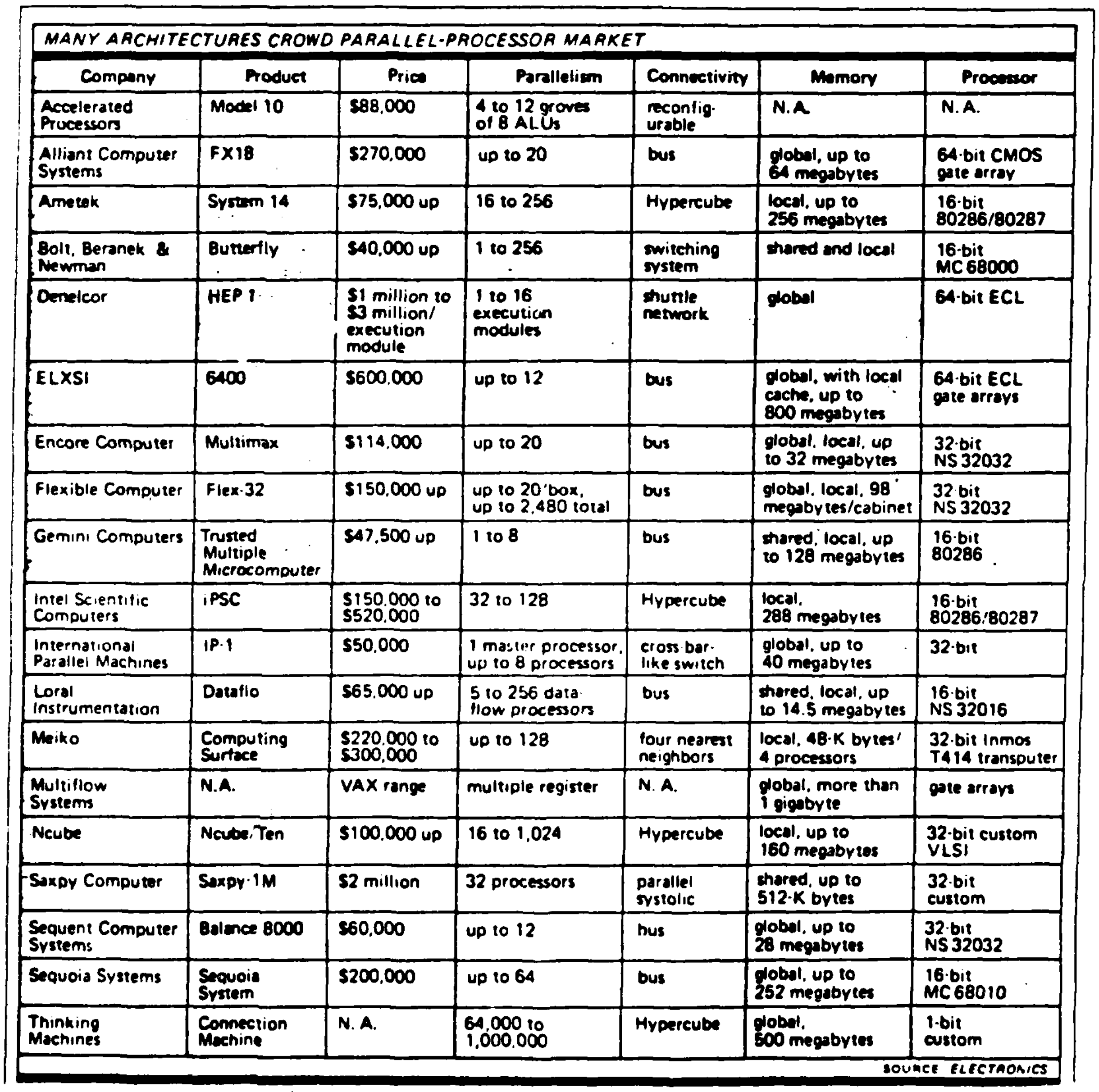
106. A (each) Cyberplus machine consists of a base (integer) processor containing 11 units, a floating point processor having 3 units, and a 512-K word high performance memory (HPM). All processor and memory interfaces operate simultaneously on a 20-ns clock, and communicate through a fully connected crossbar on every cycle. The HPM is used to maintain a plane of dense mesh raw data, together with intermediate data awaiting sequential use. The integer finite element mesh connection data string is a critically important speed feature for parallel processing that heretofore has saddled serial machine finite element code implementations with speeds a factor of 2-4 slower than finite difference codes. These data are transferred from the HPM to the base processor memories and indices pointing to problem data stored in the HPM are calculated. In the floating point processor, multiply and add units operate on pre-selected data to form the problem solution arrays.

107. Example executions of a model 3D problem discretization using bilinear hexahedron elements on meshes up to 10^6 nodes (ie., $10^2 \times 10^2 \times 10^2$) have a Cyberplus speedup factor of 70 over host speed. The Cyber 170-835 host speed is about one Mflop, yielding a predicted operating speed on Cyberplus of 62 Mflop. The base and floating point processors in Cyberplus are high performance (50 Mflop each), hence the present verification has achieved about 60% of the available speed.

108. A host of parallel architectures (Figure 11) have recently come into existence and are available for algorithm testing. Many of these use standard market chips such as the Motorola Series 6800. While software is limited for these machines, many have some Fortran capability under Unix. This utility, together with strong assembly language capability, provides the basis for coding and testing various aspects of algorithm parallelism as was done on the Cyberplus.

Example, FLEX/32 Architecture Variations

109. With the background established, algorithm/architecture variations for real machine configurations require testing. The enhanced FLEX/32



MANY ARCHITECTURES CROWD PARALLEL-PROCESSOR MARKET

Company	Product	Price	Parallelism	Connectivity	Memory	Processor
Accelerated Processors	Model 10	$88,000	4 to 12 groves of 8 ALUs	reconfig-urable	N.A.	N. A.
Alliant Computer Systems	FX18	$270,000	up to 20	bus	global, up to 64 megabytes	64-bit CMOS gate array
Ametek	System 14	$75,000 up	16 to 256	Hypercube	local, up to 256 megabytes	16-bit 80286/80287
Bolt, Beranek & Newman	Butterfly	$40,000 up	1 to 256	switching system	shared and local	16-bit MC 68000
Denelcor	HEP 1	$1 million to $3 million/ execution module	1 to 16 execution modules	shuttle network	global	64-bit ECL
ELXSI	6400	$600,000	up to 12	bus	global, with local cache, up to 800 megabytes	64-bit ECL gate arrays
Encore Computer	Multimax	$114,000	up to 20	bus	global, local, up to 32 megabytes	32-bit NS 32032
Flexible Computer	Flex-32	$150,000 up	up to 20/box, up to 2,480 total	bus	global, local, 98 megabytes/cabinet	32-bit NS 32032
Gemini Computers	Trusted Multiple Microcomputer	$47,500 up	1 to 8	bus	shared, local, up to 128 megabytes	16-bit 80286
Intel Scientific Computers	iPSC	$150,000 to $520,000	32 to 128	Hypercube	local, 288 megabytes	16-bit 80286/80287
International Parallel Machines	IP-1	$50,000	1 master processor, up to 8 processors	cross-bar-like switch	global, up to 40 megabytes	32-bit
Loral Instrumentation	Dataflo	$65,000 up	5 to 256 data-flow processors	bus	shared, local, up to 14.5 megabytes	16-bit NS 32016
Meiko	Computing Surface	$220,000 to $300,000	up to 128	four nearest neighbors	local, 48-K bytes/ 4 processors	32-bit Inmos T414 transputer
Multiflow Systems	N.A.	VAX range	multiple register	N. A.	global, more than 1 gigabyte	gate arrays
Ncube	Ncube/Ten	$100,000 up	16 to 1,024	Hypercube	local, up to 160 megabytes	32-bit custom VLSI
Saxpy Computer	Saxpy-1M	$2 million	32 processors	parallel systolic	shared, up to 512-K bytes	32-bit custom
Sequent Computer Systems	Balance 8000	$60,000	up to 12	bus	global, up to 28 megabytes	32-bit NS 32032
Sequoia Systems	Sequoia System	$200,000	up to 64	bus	global, up to 252 megabytes	16-bit MC 68010
Thinking Machines	Connection Machine	N. A.	64,000 to 1,000,000	Hypercube	global, 500 megabytes	1-bit custom

SOURCE ELECTRONICS

Figure 11. Parallel Architectures Sample

parallel processor provides an ideal setting to study these phenomena. Each cabinet can contain up to 20 processors and 30 high speed co-processors, large main memory of up to 18 megabytes and local memories for each processor of up to one megabyte each. The processors and memories can be interconnected variously to provide a system adaptable to the optimal configuration defined by the parallelism inherent in each algorithm component.

110. On the FLEX/32 system both parallel (MIMD) and pipelined (SYSTOLIC) algorithm configurations can be tested within the same framework. The comparative machine architectures are illustrated in Figure 12 where CCC and CAC represent common memory access and each C2C represents 68020-based chips in parallel with micro-code-capable computer with high speed floating point processor. In Figure 12a, the global memory across the top is conveyed to all local processor/memory systems (C2C) equally for parallel processing. In Figure 12b, the global memory feeds a single computer and results are cascaded to other computers, thus reducing common memory access speed requirements for the same number of processors. Since each of these configurations is anticipated as optimal for portions of a candidate PDE algorithm, the FLEX/32 architecture provides an ideal test bed.

111. Coding of the learned CFS/parallel concepts on a FLEX/32 provides the real machine environment experience which exposes the solution capability and limitations.

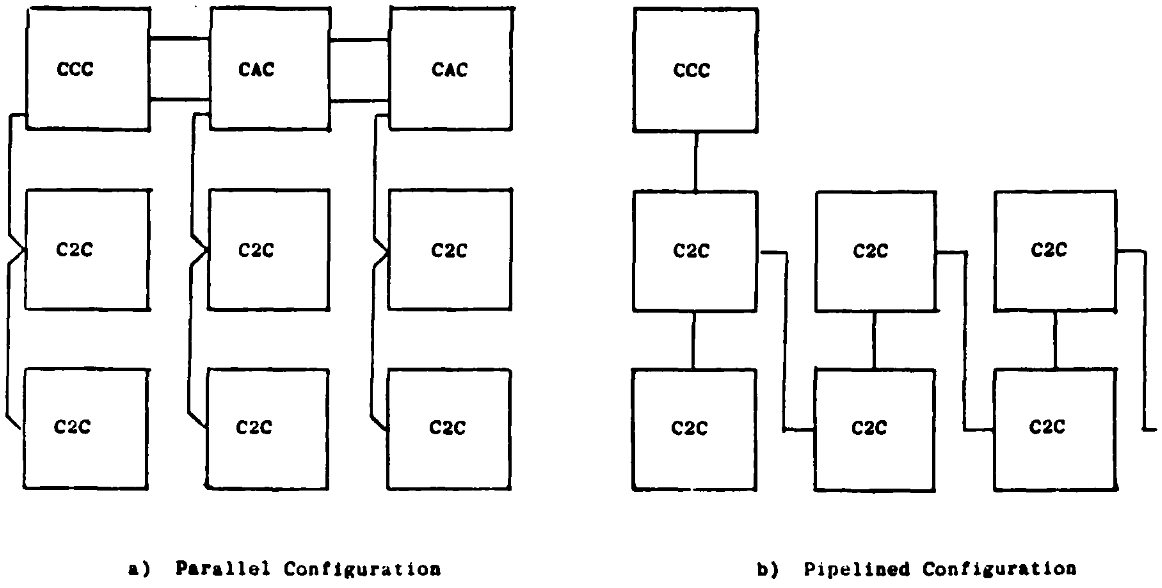


Figure 12. FLEX/32 Multicomputer Configurations

PART V: SUMMARY AND CONCLUSIONS

Summary

112. Despite its ability to resolve complex geometry in computational domains such as estuaries, rivers and coastal areas, the finite element method, as commonly applied on unstructured grids, suffers from the fact that it requires excessive computational effort per time-step. However, this can be attributed more to the techniques commonly employed by finite element modelers in solving the matrix equation system than the method itself.

113. This study was conducted with three objectives in mind. The first was to provide the average reader with an increased understanding of the two computational methods most often applied in computational fluid mechanics; namely, finite differences and finite elements. Hopefully the discussion provided has amply demonstrated that the two methods share common ground. As noted, all computational methods can be cast into the form of a weighted residual statement. Different methods result depending upon the selection of the weighting functions and/or stability constraints. The second objective of the study was to offer suggestions or ideas that might be implemented in a relatively short timeframe to reduce the computational costs of typical finite element hydrodynamic models. Ideas such as implementing an iterative solver for the linear algebra problem, employing an explicit time integration scheme with a condensed matrix, utilizing only triangular type elements to eliminate the computation time associated with numerical quadrature, and uncoupling the dependent variables should all be considered. The final objective was to present a series of steps to be accomplished over a longer time frame that will ultimately result in not only a computationally efficient model but also a robust one providing accurate solutions while possessing desirable stability characteristics.

114. The construction depends upon the use of a grid with some degree of structure, although, a completely structured grid is not required. With a semi-structured grid, the coefficient matrix; i.e., the Jacobian, associated with the linear algebra problem can be factored such that computational sweeps along curvilinear coordinate lines can be made. With a linear basis function, the linear algebra problem reduces to block-tridiagonal systems in each direction which can easily be solved. Of course, higher order basis functions will

result in pentadiagonal, etc., systems. Such an algorithm programmed for solution in a parallel processing environment will result in an extremely efficient computational model for computing free surface hydrodynamics over long periods of times. In addition, with the finite element algorithm based upon the Taylor weak statement, the numerical model will have stability mechanisms embedded in the algorithm that will provide for selective dampening, resulting in accurate solutions in high gradient regions.

Conclusions

115. The first conclusion from this effort is that completely implicit three dimensional finite element models that are programed for solution on completely unstructured grids cannot be considered for long term flow computations, e.g. weeks to months. In fact it is questionable if such models can even be considered for studies that only involve a few tidal cycles.

116. If unstructured grids are considered essential, probably the best short term solution is to employ a split mode approach. The external model would consist of implicitly solving a frictionally damped wave equation for the water surface elevations, with the vertically averaged velocities and salinity and/or temperature computed explicitly using the condensed matrix concept. The internal model, which computes the deviations from the vertically averaged components, would then also employ an explicit time integration scheme with condensed matrices.

117. Perhaps the simplest approach for implementation in the short term for an unstructured grid model to be applied over at most a few tidal cycles would be to leave the existing finite element algorithm intact but employ an explicit time integration scheme with a condensed matrix. Even though the computational time step will be limited by the speed of a free surface gravity wave, the linear algebra problem at each time step becomes a trivial operation. Therefore, the computational costs should be less than that required in the implicit model as long as the element size does not become so ridiculously small that extremely small time steps are required.

118. To achieve not only a computationally efficient but also a stable 3D time-accurate finite element hydrodynamic model, the tensor product algorithm developed from a Taylor weak statement should be implemented. Such factorizations require some degree of regularity for the grid; however,

through the use of block structured grids a significant degree of generality can still be achieved. In the long term, it is believed that such a model, programed for parallel processing, offers the best hope for 3D time-varying finite element hydrodynamic computations.

REFERENCES

Baker, A. J., and Kim, J. W., (1987), "A Taylor Weak Statement Algorithm for Hyperbolic Conservation Laws," Int. J. Num. Mtd. Fluids, V. 7, p. 489-520.

Baker A. J., and Manhardt, P. D., (1985), "3D CFD Algorithms for Parallel Processing," in M. Ferrate (ed), Proc. Control Data Executive Seminar, Minneapolis, p. 1-18.

Baker, A. J. (1983) Finite Element Computational Fluid Mechanics, Hemisphere/ Harper & Row, New York.

Beam, R. M., and Warming, R. F. (1978), An Implicit Factored Scheme for Compressible Navier Stokes Equations, AIAA J., V. 16, p. 393-402.

Cebeci, T., and Smith, AMO, (1974), Analysis of Turbulent Boundary Layers, Academic Press, New York.

Flynn, M. J. (1972), "Some Computer Organizations and Their Effectiveness," IEEE, Trans. Computer, Vol. C-21, No. 9.

Hageman, L. A. and Young D. M., (1981), Applied Iterative Methods, Academic Press, New York, NY.

Irons, B. M. and Ahmad S., (1980), Techniques of Finite Elements, Ellis Horwood, Chichester, UK.

King, I. P., (1982), A Finite Element Model for Three Dimensional Flow, Technical Report RMA 9150, Resource Management, Assoc., Lafayette, CA.

Kung, H. T., (1980), "The Structure of Parallel Algorithms," Advances in Computers, Vol. 19, Academic Press, NY.

Lin, A., (1985), "Towards Generalization and Optimization of Implicit Methods," Int. Journal Numerical Methods in Fluids, Vol. 5 p. 357.

Manhardt, P. D., Orzechowski, J. A., and Baker, A. J., (1985), "Solution of Non-linear PDE's on the Cyberplus Attached Super Computer," Technical Report COMCO 85-TR-11.1, COMCO, Inc., Knoxville, TN.

Mavriplis, D., and Jameson, A., (1987), "Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes," AIAA 25th Aerospace Sciences Meeting, Reno, Nevada.

Ortega, J. M., and Voight, R. G., (1985), "Solution of Partial Differential Equations on Vector and Parallel Computers," SIAM, NY.

Sloan, S. W. and Randolph, M. F., (1983) "Automatic Element Recording for Finite Element Analysis with Frontal Solution Schemes," Int. J. Num. Mtd. Engr., Vol 19, p 1153-1181.

Stone, H. S. (1980), "Introduction to Computer Architecture," Sec. Ed., Ch. 8.