ELSEVIER

Applied Numerical Mathematics 31 (1999) 271–293

APPLIED NUMERICAL MATHEMATICS

www.elsevier.nl/locate/apnum

An unstructured grid-based, parallel free surface solver

Rainald Löhner [a,*], Chi Yang [a], Eugenio Oñate [b], Sergio Idelssohn [b]

[a] *GMU/CSI, George Mason University, Fairfax, VA 22030-4444, USA*
[b] *CIMNE, Universidad Politécnica de Catalunya, Barcelona, Spain*

Abstract

An unstructured grid-based, parallel-free surface solver is presented. The overall scheme combines a finite-element, equal-order, projection-type 3-D incompressible flow solver with a finite element, 2-D advection equation solver for the free surface equation. For steady-state applications, the mesh is not moved every timestep, in order to reduce the cost of geometry recalculations and surface repositioning. A number of modifications required for efficient processing on shared-memory, cache-based parallel machines are discussed, and timings are shown that indicate scalability to a modest number of processors. The results show good quantitative comparison with experiments and the results of other techniques. The present combination of unstructured grids (enhanced geometrical flexibility) and good parallel performance (rapid turnaround) should make the present approach attractive to hydrodynamic design simulations.

Keywords: Finite elements; Unstructured grids; Free surface; Parallel computing

1. Introduction

The prediction of the Kelvin wave pattern and wave resistance of ships has challenged mathematicians and hydrodynamicists for over a century. Only in recent years has the rapid development of computer hardware and software enabled large scale computer simulation of steady ship waves. However, the wave resistance, particularly for unsteady ship and/or wave motion, is not predicted with sufficient accuracy and efficiency to substitute model experiments.

The Boundary Element Method forms the basis for the majority of computational algorithms for the prediction of the ideal wave pattern of ships advancing with constant forward speed. These numerical schemes may be classified in two categories, based on the choice of elementary singularity.

The first class of schemes uses the Kelvin wave source as the elementary singularity. The major advantages of such a scheme are the elimination of the integration over the free surface from the resulting boundary integral equation and the automatic satisfaction of the radiation condition. However,

* Corresponding author. E-mail: lohner@rossini.gmu.edu.

0168-9274/99/$20.00
PII: S0168-9274(99)00002-1

this scheme can not be extended to include nonlinear wave effects. The theoretical background of this method was reviewed by Wehausen [45], while computational aspects can be found in the literature and in a series of Wave Resistance Workshops [2,33]. Noblesse et al. [32] recently developed a new theoretical formulation, called Fourier–Kochin theory, which offers an alternative way of solving steady wave problems.

The second class of schemes uses the Rankine source as the elementary singularity. This scheme was first presented by Dawson [8]. The Dawson method has been applied widely as a practical method for predicting wave resistance, and many improvements have been made to account for the nonlinear effects. Among them a successful example is the Rankine Panel Method [3,16,17,20,30,31,36–39,41, 46]. Considerable effort has been devoted to increasing efficiency and accuracy, resulting in the so-called patch method, desingularized method, RAPID method and so on. From a ship design point of view, the Rankine Panel Method still has an unacceptable sensitivity to numerical parameters such as the domain size and paneling.

In recent years, the advent of advanced numerical schemes for the Euler and Navier–Stokes equations has enabled a more realistic prediction of wave resistance. The most accurate of these schemes have used a 3-D, i.e., volumetric incompressible flow solver coupled with a free surface equation. The velocities obtained at the free surface from the 3-D incompressible flow solver are given to the free surface solver to update the free surface height. This new height changes the (prescribed) pressure at the free surface for the 3-D incompressible flow solver, closing the loop. The free surface height also serves as the basis for the mesh motion. There exist two main types of incompressible flow solvers.

The first class is based on projection schemes [1,5,13,18,19,21,26,27,29,35,42]. A velocity field is predicted in a first step. The conservation of mass is enforced in a second step by solving a Poisson equation, which results in a new pressure. Finally, the velocity field is updated with this new pressure.

The second class is based on artificial compressibility schemes [4,7,10–12,14,15,28,34,40]. The infinite speed of sound of the incompressible medium is reduced to a finite number by adding a time derivative to the divergence equation. This enables the effective use of all the techniques developed for compressible flow simulation, e.g., limitors, upwind differencing, residual smoothing, multigrid acceleration, etc. At steady state, the time derivative in the divergence equation vanishes, yielding the proper incompressible solution.

Both families of solvers have been used successfully for free surface prediction. We prefer the first one, because the pressure is specified over a considerable portion of the domain (free surface, entry plane, bottom), resulting in very fast convergence for the Poisson solver.

An unstructured grid is used in the present finite element method to enhance geometry flexibility and to speed up the initial modeling time. An automatic unstructured grid generator based on the advancing front method is used to generate triangular surface grids and tetrahedral volume grids. In addition, the unstructured grid generator is linked to the flow solver, such that an automatic remeshing can easily be performed to simulate fully nonlinear waves.

The present paper is organized as follows: Section 2 summarizes the equations used to describe the flow and free surface; Section 3 describes the numerical methods used to solve these equations; Section 4 and 5 are devoted to mesh update and parallelization; some examples are shown in Section 6; finally, some conclusions are drawn and an outlook for further research is given in Section 7.

2. Equations solved and boundary conditions

The equations solved are the incompressible Euler equations, given, in non-dimensional form, by the conservation of mass and momentum:

$$\nabla \cdot \boldsymbol{v} = 0, \tag{1}$$

$$\boldsymbol{v}_{,t} + \boldsymbol{v} \cdot \nabla \boldsymbol{v} + \nabla \Psi = 0, \tag{2}$$

where $\boldsymbol{v} = (u, v, w)$ denotes the velocity vector and Ψ the pressure plus hydrostatic pressure:

$$\Psi = p + \frac{z}{\mathrm{Fr}^2}, \quad \mathrm{Fr} = \frac{|\boldsymbol{v}_\infty|}{\sqrt{g \cdot L}}, \tag{3}$$

and the coordinate orientation shown in Fig. 1 is employed. A particle on the free surface must remain so for all times, which implies that the free surface elevation β obeys the advection equation

$$\beta_{,t} + u\beta_{,x} + v\beta_{,y} = w. \tag{4}$$

The boundary conditions are as follows:

(a) *Inflow plane*: At the inflow plane, the velocity, pressure and free surface height are prescribed:

$$\boldsymbol{v} = (1, 0, 0), \qquad \Psi = 0, \qquad \beta = 0. \tag{5}$$

(b) *Exit plane*: At the exit plane, none of the quantities are prescribed. The natural Neumann conditions for the pressure and extrapolation boundary conditions for the velocities and free surface height are imposed automatically by the numerical scheme used.

(c) *Free surface*: At the free surface, the pressure p is prescribed to be $p = 0$, implying that Ψ is given by

$$\Psi = \beta \mathrm{Fr}^{-2}. \tag{6}$$

The velocity is allowed to float freely, allowing leakage of fluid through the free surface.

(d) *Bottom*: At the bottom, one may either impose the boundary conditions for a:
- *Wall*: vanishing normal velocity, Neumann conditions for the pressure, or
- *Infinite Depth*: prescribed pressure, no boundary conditions for the velocities.

(e) *Ship hull*: On the ship hull, the normal velocity must vanish, i.e.,

$$\boldsymbol{v} \cdot \boldsymbol{n} = 0, \tag{7}$$

where $\boldsymbol{n}$ is the normal to the hull.

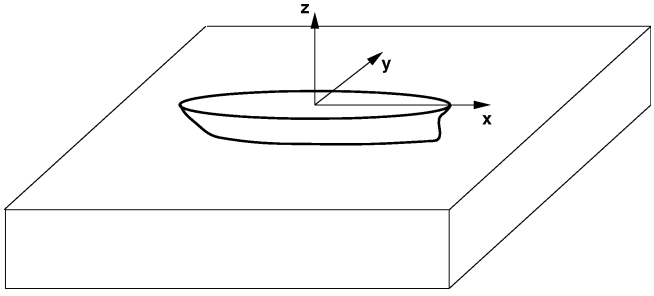

Fig. 1. Reference frame and ship position.

(f) *Side walls*: On the side walls of the computational domain, we impose the same conditions as for the hull, i.e., vanishing normal velocity.

3. Numerical implementation

For the solution of the 3-D incompressible flow equations, a pressure projection scheme is used. In this way, the pressures are integrated implicitly, correctly reflecting the infinite propagation speed of sound. A complete timestep consists of three parts:

(a) *Advective prediction*: $\boldsymbol{v}^n \to \boldsymbol{v}^*$

$$\frac{\boldsymbol{v}^* - \boldsymbol{v}^n}{\Delta t} + \boldsymbol{v}^n \cdot \nabla \boldsymbol{v}^n + \nabla \Psi^n = 0; \tag{8}$$

(b) *Pressure correction*: $\Psi^n \to \Psi^{n+1}$

$$\nabla \cdot \boldsymbol{v}^{n+1} = 0, \tag{9a}$$

$$\frac{\boldsymbol{v}^{n+1} - \boldsymbol{v}^*}{\Delta t} + \nabla\left(\Psi^{n+1} - \Psi^n\right) = 0, \tag{9b}$$

which results in

$$\nabla^2\left(\Psi^{n+1} - \Psi^n\right) = \frac{\nabla \cdot \boldsymbol{v}^*}{\Delta t}, \tag{10}$$

(c) *Velocity correction*: $\boldsymbol{v}^* \to \boldsymbol{v}^{n+1}$

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^* - \Delta t \nabla\left(\Psi^{n+1} - \Psi^n\right). \tag{11}$$

Observe that at steady state, the residuals of the pressure correction vanish, implying that the result is neither dependent on the projection scheme itself nor the timestep Δt.

3.1. Spatial discretization

The spatial discretization via the Galerkin weighted residual method using linear elements results in an edge-based loop for the right-hand sides of the form [26]:

$$\boldsymbol{r}^i = d_k^{ij}\left(\boldsymbol{F}_j^k + \boldsymbol{F}_i^k\right), \tag{12}$$

where d_k^{ij} contains all the geometric parameters associated with the elements surrounding the edge i, j and the dimension k. The inner product over the dimensions k may be written in compact form as

$$r^i = D^{ij}\mathcal{F}_{ij} = D^{ij}(\boldsymbol{f}_i + \boldsymbol{f}_j), \tag{13}$$

where the $\boldsymbol{f}_i$ are the 'fluxes along edges', obtained from the scalar product

$$\boldsymbol{f}_i = S_k^{ij}\boldsymbol{F}_i^k, \qquad S_k^{ij} = \frac{d_k^{ij}}{D^{ij}}, \quad D^{ij} = \sqrt{d_k^{ij} d_k^{ij}}. \tag{14}$$

For the *advective terms* we have:

$$\mathcal{F}_{ij} = \boldsymbol{f}_i + \boldsymbol{f}_j, \tag{15}$$

where

$$\boldsymbol{f}_i = (S_k^{ij} v_i^k)\boldsymbol{v}_i, \qquad \boldsymbol{f}_j = (S_k^{ij} v_j^k)\boldsymbol{v}_j. \tag{16}$$

$\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ denote velocity vectors at node i and node j, respectively, and v_i^k and v_j^k denote the velocity components in dimension k at node i and node j, respectively. This form of the fluxes is of central difference character, and must be replaced by a consistent or stabilized numerical flux. For the advection system given by Eq. (4), this results in

$$\mathcal{F}_{ij} = \boldsymbol{f}_i + \boldsymbol{f}_j - |v^{ij}|(\boldsymbol{v}_i - \boldsymbol{v}_j), \tag{17}$$

where

$$v^{ij} = \tfrac{1}{2} S_k^{ij} (v_i^k + v_j^k). \tag{18}$$

This first-order scheme is improved to second order by replacing the values of $\boldsymbol{v}_i, \boldsymbol{v}_j$ by improved or reconstructed values $\boldsymbol{v}_{i'}, \boldsymbol{v}_{j'}$. Standard MUSCL limiting procedures [44] are used to obtain monotonicity preserving solutions even in the pure advection (i.e., Euler) case.

For the *continuity equation*, the edge-based expression used is

$$\mathcal{F}_{ij} = \boldsymbol{f}_i + \boldsymbol{f}_j, \tag{19}$$

where

$$\boldsymbol{f}_i = S_k^{ij} v_i^k, \qquad \boldsymbol{f}_j = S_k^{ij} v_j^k. \tag{20}$$

A consistent or stabilized numerical flux is given by

$$\mathcal{F}_{ij} = \boldsymbol{f}_i + \boldsymbol{f}_j - |\lambda^{ij}|(p_i - p_j), \tag{21}$$

with

$$\lambda^{ij} = \frac{\Delta t^{ij}}{l^{ij}}. \tag{22}$$

Here Δt and l are a characteristic advective timestep and length of the edge. This first-order scheme is replaced by a higher-order scheme of the form

$$\mathcal{F}_{ij} = \boldsymbol{f}_i + \boldsymbol{f}_j - |\lambda^{ij}|\left(p_i - p_j + \frac{l^{ij}}{2}(\nabla p_i + \nabla p_j)\right), \tag{23}$$

which is reminiscent of a fourth-order damping term for the divergence equation [18,34].

3.2. Free surface discretization

The free surface equation (4) is treated as a standard scalar advection equation with source terms for the x, y plane. The faces on the free surface are extracted from the 3-D volume grid. Points and elements are renumbered locally to obtain a 2-D triangular finite element mesh in x, y. As before, the spatial discretization of the advective fluxes results in

$$\mathcal{F}_{ij} = \boldsymbol{f}_i + \boldsymbol{f}_j, \quad \boldsymbol{f}_i = (S_x^{ij} u_i + S_y^{ij} v_i) \cdot \beta_i. \tag{24}$$

Fourth-order damping is added to stabilize these central difference terms, resulting in

$$\mathcal{F}_{ij} = \boldsymbol{f}_i + \boldsymbol{f}_j - |\lambda^{ij}|\left(\beta_i - \beta_j + \frac{l^{ij}}{2}(\nabla\beta_i + \nabla\beta_j)\right). \tag{25}$$

Following Hino [18], an additional damping term is added to the free surface equation near inflow and outflow boundaries, resulting in

$$\beta_{,t} + u\beta_{,x} + v\beta_{,y} = w - d_h(\boldsymbol{x})\beta, \tag{26}$$

where $d_h(\boldsymbol{x})$, for the outflow boundary, is given by

$$d_h = c_1\xi^2, \qquad \xi = \max\left(0, \frac{x - x_{\mathrm{d}\max}}{x_{\max} - x_{\mathrm{d}\max}}\right), \quad x_{\mathrm{d}\max} = x_{\max} - 2\pi\,\mathrm{Fr}^2, \tag{27}$$

and c_1 is a parameter of O(1). A similar expression is applied at the inflow boundary for steady-state problems. The vertical velocity w, as well as the additional damping term, are evaluated by simply using the lumped mass matrix. In order to damp out the wave height β completely at the downstream boundary, the w-velocity obtained from the 3-D incompressible flow solver is modified, yielding

$$\beta_{,t} + u\beta_{,x} + v\beta_{,y} = d_w(\boldsymbol{x})w - d_h(\boldsymbol{x})\beta, \tag{28}$$

where $d_w(\boldsymbol{x})$ is given by the Hermitian polynomial

$$d_w = 1 - 3\xi^2 + 2\xi^3, \tag{29}$$

and ξ is defined in Eq. (27). The final semi-discrete scheme takes the form

$$\boldsymbol{M}_l\beta_{,t} = \boldsymbol{r} = \boldsymbol{r}_{\mathrm{a}}(u, v, \beta) + \boldsymbol{r}_{\mathrm{s}}(d_w, w) + \boldsymbol{r}_{\mathrm{d}}(d_h, \beta), \tag{30}$$

where the subscripts a, s, d stand for advection, source and damping. This system of ODEs is integrated in time using a standard five stage Runge–Kutta scheme.

3.3. Overall scheme

One complete timestep consists of the following steps:
- Given the boundary conditions for the pressure Ψ, update the solution in the 3-D fluid mesh;
- Extract the velocity vector $\boldsymbol{v} = (u, v, w)$ at the free surface and transfer it to the 2-D free surface module;
- Given the velocity field, update the free surface β;
- Transfer back the new free surface β to the 3-D fluid mesh, and impose new boundary conditions for the pressure Ψ.

For steady-state applications, the fluid and free surface domains are updated using local timesteps. This allows some room for variants that may converge faster to the final solution, e.g., n steps of the fluid followed by m steps of the free surface, complete convergence of the free surface between fluid updates, etc. We have experimented with some of these. The results show that most of these variants prove unstable, or do not accelerate convergence measurably. Our current preference for steady-state applications is to use an equivalent 'time-interval' ratio between fluid and free surface of 1:8, e.g., a Courant-nr. of $C_f = 0.25$ for the fluid and $C_s = 2.0$ for the free surface.

4. Mesh update strategy

Previous work by Hino [14,15] and Farmer [10,11] marched the solution in time until a steady-state was reached. At each timestep, a volume update was followed by a free surface update. The

repositioning of points at each timestep implies a complete recalculation of geometrical parameters, as well as interrogation of the CAD information defining the surface. In our case, this would double CPU requirements. For this reason, when solving steady-state problems, we do not move the grid at each timestep, but only change the pressure boundary condition after each update of the free surface β. The mesh is updated every 100–250 timesteps, thereby minimizing the costs associated with geometry recalculations and grid repositioning along surfaces. We have observed that this strategy has the advantage of not moving the mesh unduly at the beginning of a run, where large wave amplitudes may be present.

One mesh update consists of the following steps:

- Obtain the new elevation for the points on the free surface from β. This only results in a vertical (z-direction) displacement field $\boldsymbol{d}_\Gamma$ for the boundary points.
- Apply the proper boundary conditions for the points on the waterline. This results in an additional horizontal (x, y-direction) displacement field for the points on the water line.
- Smooth the displacement field in order to avoid mesh distortion. The smoother used is of the form [24]:

$$\nabla \cdot k \nabla \boldsymbol{d} = 0, \tag{31}$$

  where k is a nonlinear stiffness coefficient that depends on the distance from the hull.
- Interrogate the CAD data to reposition the points on the hull.

Denoting by $\boldsymbol{d}_*, \boldsymbol{n}, \boldsymbol{t}$ the predicted displacement of each point, surface normals and tangential directions, the boundary conditions for the mesh movement are as follows (see Fig. 2):

(a) *Hull, on surface patch*: The movement of these points has to be along the surface, i.e., the normal component of $\boldsymbol{d}_*$ is removed:

$$\boldsymbol{d} = \boldsymbol{d}_* - (\boldsymbol{d}_* \cdot \boldsymbol{n})\boldsymbol{n}. \tag{32}$$

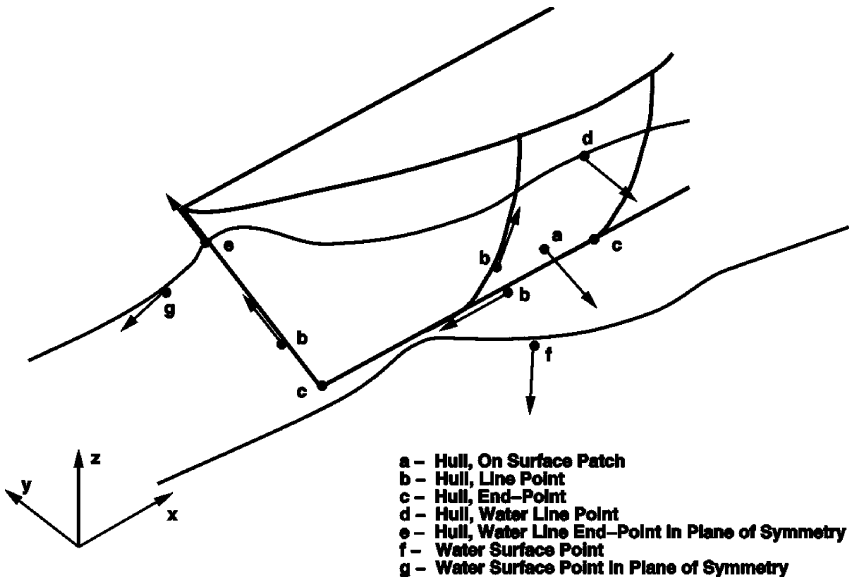

Fig. 2. Boundary conditions for mesh movement.

(b) *Hull, line point*: The movement of these points has to be along the lines, resulting in a tangential boundary displacement of the form

$$\boldsymbol{d} = (\boldsymbol{d}_* \cdot \boldsymbol{t})\boldsymbol{t}. \tag{33}$$

(c) *Hull, end-point*: No displacement is allowed for these points, i.e., $\boldsymbol{d} = 0$.

(d) *Hull, water line point or water line, end-point*: The displacement of these points is fixed, given by the change in elevation Δz and the surface normal of the hull. Defining $\boldsymbol{d}_0 = (0, 0, \Delta z)$, we have

$$\boldsymbol{d} = \frac{\boldsymbol{d}_0 - (\boldsymbol{d}_0 \cdot \boldsymbol{n})\boldsymbol{n}}{1 - n_z^2}. \tag{34}$$

(e) *Hull, water line end-point in plane of symmetry*: The displacement of these points is fixed, and dictated by the tangential vector to the hull-line in the symmetry plane:

$$\boldsymbol{d} = \frac{(\boldsymbol{d}_0 \cdot \boldsymbol{t})\boldsymbol{t}}{1 - t_z^2}. \tag{35}$$

(f) *Water surface points*: These points start with an initial displacement $\boldsymbol{d}_0$, but may glide along the water surface, allowing the mesh to accommodate the displacements in the x, y-directions due to points on the hull. The normal to the water line is taken, and Eq. (31) is used to correct any further displacements.

(g) *Water surface points in plane of symmetry*: As before, these points start with an initial displacement $\boldsymbol{d}_0$, but may glide along the water surface, remaining in the plane of symmetry, thus allowing the mesh to accommodate the displacements in the x-direction due to points on the hull. The tangential direction is obtained from the sides lying on the water surface in the plane of symmetry, and Eq. (31) is used to correct any further displacements.

An option to restrict the movement of points completely in certain regions of the mesh has been found useful. Regions where such an option is required are transom sterns, as well as the points lying in the half-plane given by the minimum z-value of the hull. Should negative elements arise due to surface point repositioning, these are removed and a local remeshing takes place. Naturally, we try to avoid these situations as much as possible.

5. Parallelization

The two main CPU-intensive parts of the incompressible flow solver are the Poisson solver and the velocity update. Both involve loops over edges. For the Poisson solver, these are of the form:

```
      do 1600 iedge=1,nedge
      ipoi1=lnoed(1,iedge)
      ipoi2=lnoed(2,iedge)
      redge=geoed(  iedge)*(unkno(ipoi2)-unkno(ipoi1))
      rhspo(ipoi1)=rhspo(ipoi1)+redge
      rhspo(ipoi2)=rhspo(ipoi2)-redge
1600 continue
```

For MIMD machines, parallelization is accomplished via domain decomposition with message passing at interface boundaries [24,34]. Recently, a number of shared-memory, cache-based machines with multiple

CPU's have appeared. A typical example of this class of machine is the SGI Power Challenge, which presently allows up to 18 processors. When modifying solvers to run efficiently on this type of machine, techniques must be implemented that avoid:

- cache-misses (in order to perform well on each processor);
- memory contention (in order to allow pipelining); and
- cache overwrite (in order to perform well in parallel).

A low number of cache-misses is achieved by renumbering the points, so that any required point information for edges is as close as possible in memory when required by an edge. At the same time, as the loop progresses through the edges, the point information should be accessed as uniformly as possible. This may be achieved by first renumbering the points using a bandwidth-minimization technique and subsequently renumbering the edges according to the minimum point number on each edge [28]. All of these algorithms are of complexity $O(N)$ or at most $O(N \log N)$, and are well worth the effort.

In order to achieve pipelining or vectorization, memory contention issues must be avoided. Groups of edges are built such that none of the points is accessed by the edges in each group more than once. Given that in order to achieve good pipelining performance on current RISC-chips a relatively short vector length of 16 is sufficient, one can simply start from the edge-renumbering obtained in order to minimize cache-misses, and renumber it further into groups of edges that are 16 long and avoid memory contention [28]. As before, this renumbering is of complexity $O(N)$. The resulting loop, shown schematically in Fig. 3, translates, for the Laplacian operator, into

```
       do 1400 ipass=1,npass
       nedg0=edpas(ipass)+1
       nedg1=edpas(ipass+1)
c$dir ivdep                                          ! Pipelining directive
        do 1600 iedge=nedg0,nedg1
        ipoi1=lnoed(1,iedge)
        ipoi2=lnoed(2,iedge)
        redge=geoed(  iedge)*(unkno(ipoi2)-unkno(ipoi1))
        rhspo(ipoi1)=rhspo(ipoi1)+redge
        rhspo(ipoi2)=rhspo(ipoi2)-redge
 1600  continue
 1400 continue
```

The next stage is to port such a loop to a parallel, shared memory machine. If the loop is left untouched, the auto-parallelizing compiler will simply split the inner loop across processors. It would then appear that increasing the vector-length to a sufficiently large value would offer a satisfactory solution. However, this is not advisable, as it leads to either high start-up penalties (for short vector lengths) or a high rate of cache-misses (for long vector lengths). Moreover, this type of grouping does not take into consideration dirty cache-lines. As the points in a split group access a large portion of the edge-array, different processors may be accessing the same cache-line. When a 'dirty cache-line' overwrite occurs, all processors must update this line, leading to a large increase of interprocessor communication, severe performance degradation and non-scalability. Experiments on an 8-processor SGI Power Challenge showed a maximum speed-up of only 1:2.5 when using this option. This limited speed-up was attributed, to a large extent, to cache-line overwrites. In view of these consequences, additional renumbering strategies have to be implemented. The central idea is to achieve pipelining and parallelization by

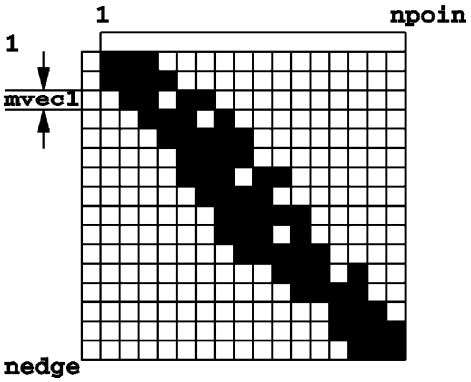

Fig. 3. Near-optimal renumbering of edges and points for 1-processor machine.

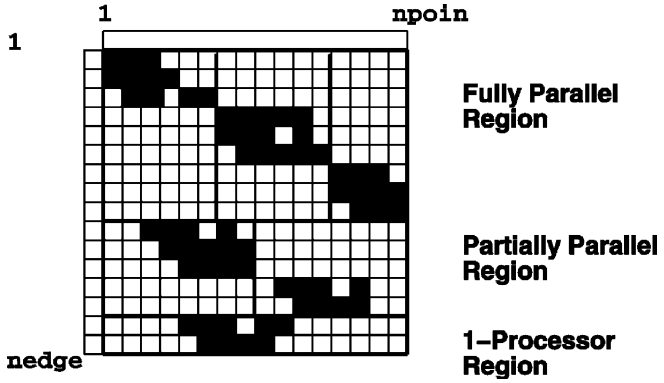

Fig. 4. Renumbering of edge-groups for shared-memory, cache-based parallel machine.

processing all the individual vector-groups in parallel at a higher level (see Fig. 4), in such a way that the point-range between macro-groups does not overlap [7]. This renumbering of edges is similar to domain-splitting, but does not require an explicit message passing or actual identification of domains. Rather, all the operations are kept at the (algebraic) array level. The number of sub-groups, as well as the total number of edges to be processed in each macro-group, is not the same. However, the imbalance is small, and does not affect performance significantly. The actual loop is given by

```
       do 1000 imacg=1,npasg,nproc
       imac0=          imacg
       imac1=min(npasg,imac0+nproc-1)
c                                              !  Parallelization directive
c$doacross local(ipasg,ipass,npas0,
c$&             npas1,iedge,nedg0,nedg1,
c$&             ipoi1,ipoi2,redge)
        do 1200 ipasg=imac0,imac1
        npas0=edpag(ipasg)+1
        npas1=edpag(ipasg+1)
         do 1400 ipass=npas0,npas1
         nedg0=edpas(ipass)+1
         nedg1=edpas(ipass+1)
c$dir ivdep                                       !  Pipelining directive
          do 1600 iedge=nedg0,nedg1
          ipoi1=lnoed(1,iedge)
          ipoi2=lnoed(2,iedge)
          redge=geoed(  iedge)*(unkno(ipoi2)-unkno(ipoi1))
          rhspo(ipoi1)=rhspo(ipoi1)+redge
          rhspo(ipoi2)=rhspo(ipoi2)-redge
 1600    continue
 1400   continue
 1200  continue
 1000 continue
```

As one can see, this type of renumbering entails two outer loops, implying that a certain amount of code rewrite is required. On the other hand, the original code can easily by retrieved by setting `edpag(1)=0, edpag(2)=npass, npasg=1` for conventional uni-processor machines. In all the examples shown below we have used a simple greedy-type algorithm [34] to group and renumber the edges.

6. Example cases

A number of example cases were run with the proposed methodology. For all of these cases, local timestepping was employed for the 3-D incompressible flow solvers as well as the free surface solver. At the start of a run, the 3-D flowfield was updated for 10 timesteps without any free surface update. Thereafter, the free surface was updated after every 3-D flowfield timestep. The mesh was moved every 100–250 timesteps. The wave drag was computed by integrating the pressure over the wetted surface.

6.1. Submerged NACA0012

The first case considered is a submerged NACA0012 with the submergence $s = 1.034$ and the angle of attack $\alpha = 5^\circ$. This same configuration was tested experimentally by Duncan [9] and modeled numerically by Hino [14,15]. Although the case is 2-D, it was modeled as 3-D, with two parallel walls in the y-direction. The mesh consisted of 1,475,036 tetrahedral elements, 299,771 points and 96,436 boundary points. The free surface had 12,196 triangular elements and 6,929 points. Figs. 5(a)–(d) show, respectively, the corresponding 2-D mesh and pressure contour at steady state, convergence history for the 3-D flowfield, and convergence history for the inviscid drag force. The same mesh was splited into four subdomains. Figs. 5(e)–(f) show the same result using a distributed memory, MIMD approach with four subdomains.

6.2. Wigley hull

The second case considered is the well known Wigley hull, given by the analytical formula

$$y = 0.5 \cdot B \cdot \left[1 - 4x^2\right] \cdot \left[1 - \left(\frac{z}{D}\right)^2\right], \tag{36}$$

where B and D are the beam and the draft of the ship at still water. For the case considered here, we had: $-0.5 < x < 0.5$, $D = 0.0625$, $B = 0.1$. This same configuration was tested experimentally at the University of Tokyo [6] and modeled numerically by Farmer [10], Raven [38], and others. We first generated a fine triangulation for the surface given by Eq. (36). This triangulation was subsequently used to define, in a discrete manner, the hull. The surface definition of the complete computational domain consisted of discrete (hull) and analytical surface patches. The mesh consisted of 1,003,554 tetrahedral elements, 184,619 points and 28,140 boundary points. The free surface had 28,383 triangular elements and 14,495 points. The Froude-number was set to $\mathrm{Fr} = 0.25$. Figs. 6(a), (b) show, respectively, the surface mesh and wave elevation contour at steady state. In order to address the grid convergence, two more computational grids were considered for the given Froude number. The medium mesh consisted of 697,041 tetrahedral elements, 129,464 points and 21,927 boundary points, and the coarse mesh consisted of 433,426 tetrahedral elements, 81,626 points and 15,920 boundary points. Figs. 6(c)–(e)

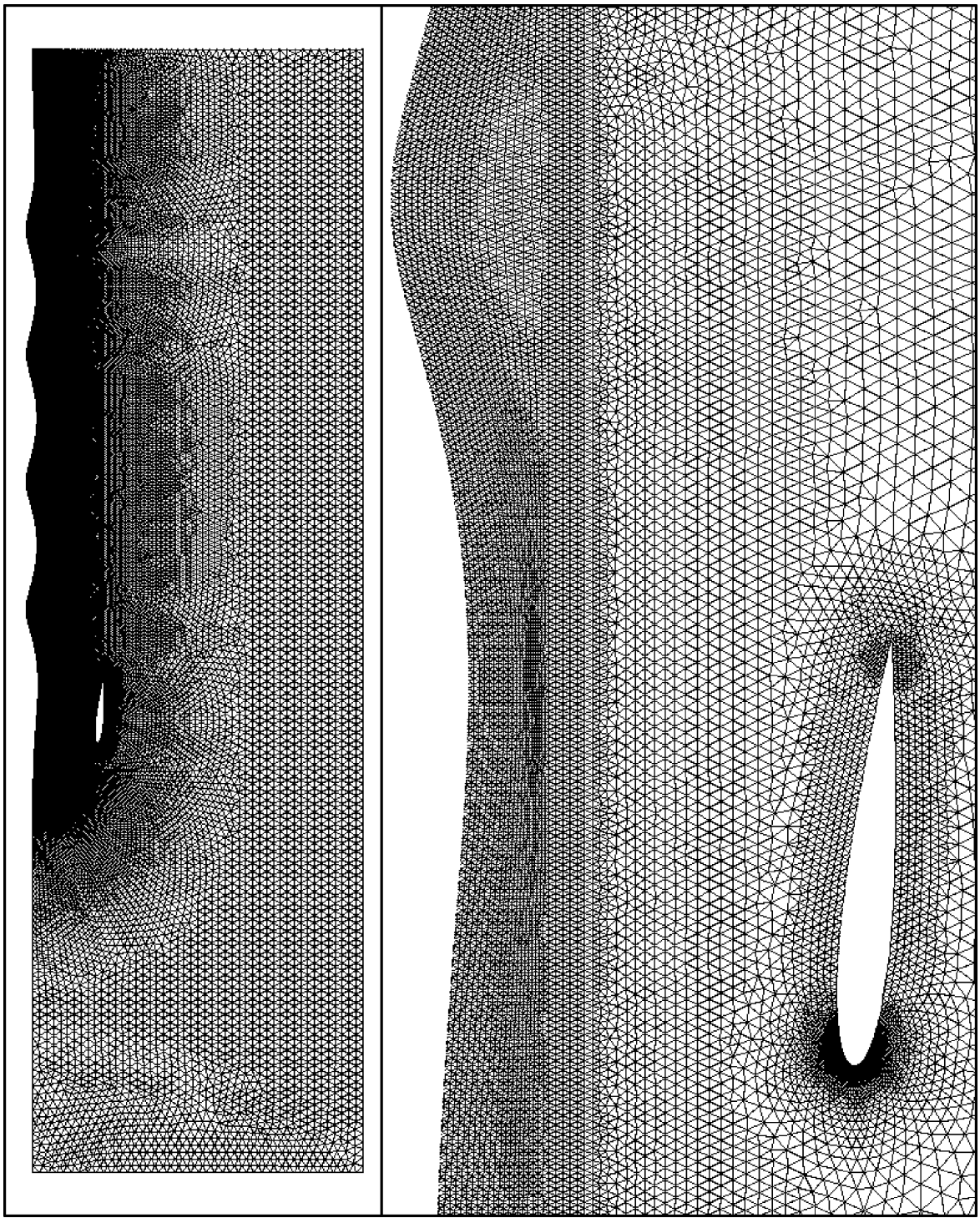

Fig. 5a. 2-D mesh.

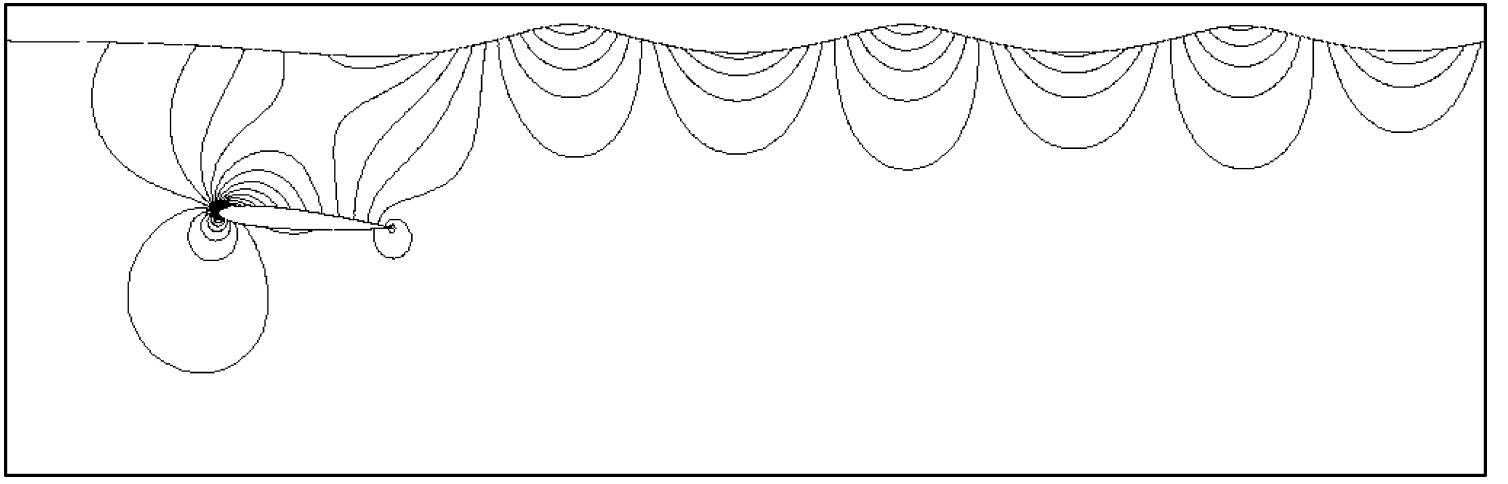

Fig. 5b. 2-D pressure contour.

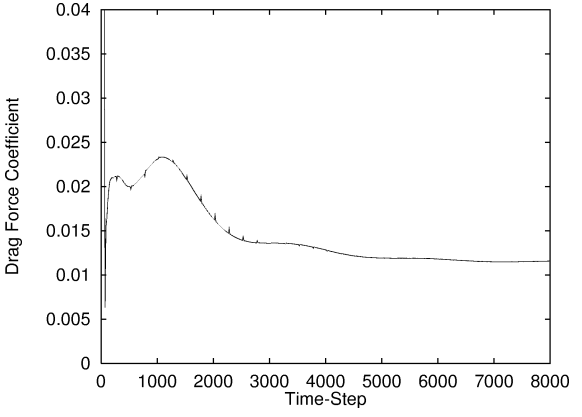

Fig. 5c. Inviscid drag convergence history.

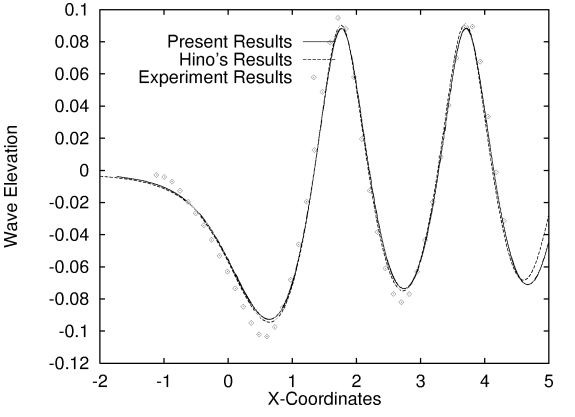

Fig. 5d. Comparison of wave profiles.

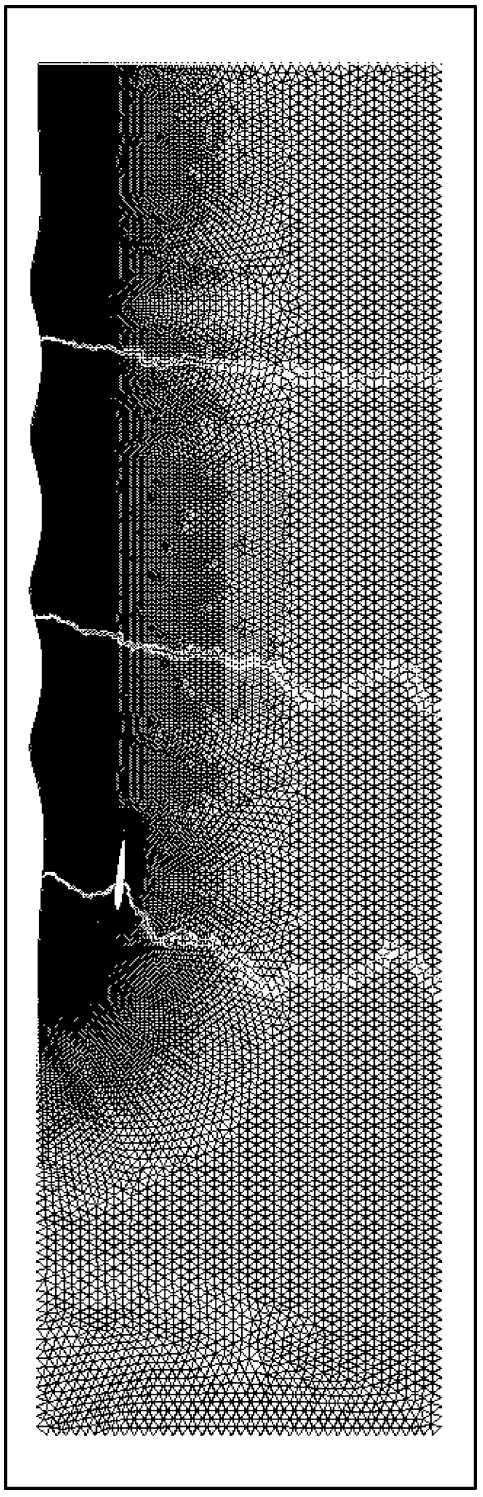

Fig. 5e. 2-D mesh with 4 subdomains.

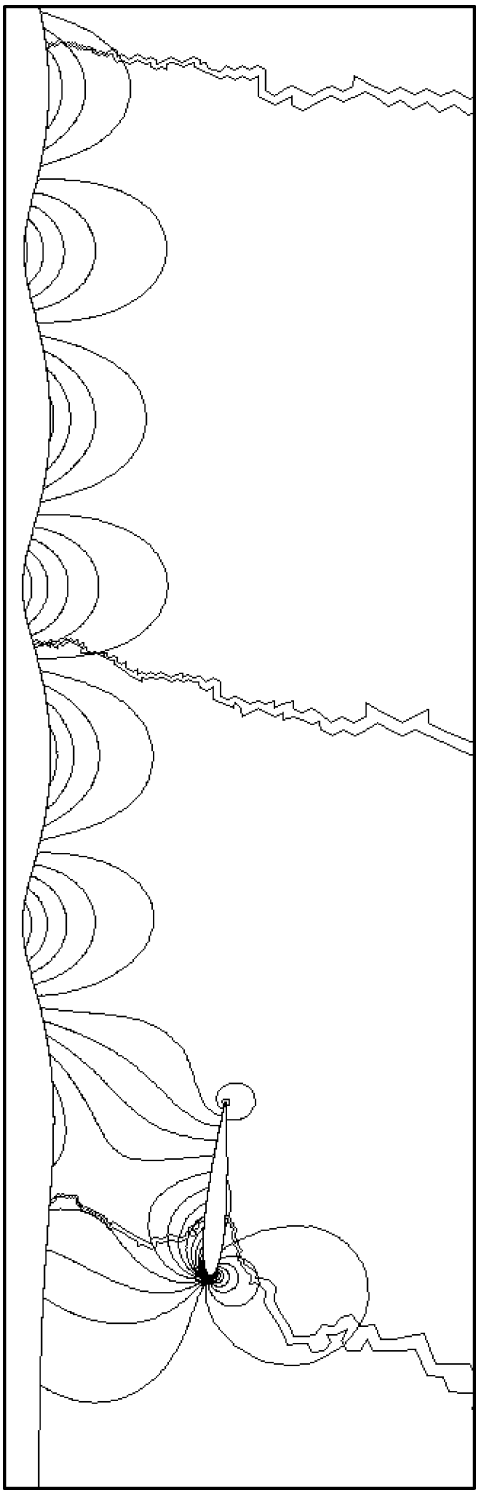

Fig. 5f. 2-D pressure contour with 4 subdomains.

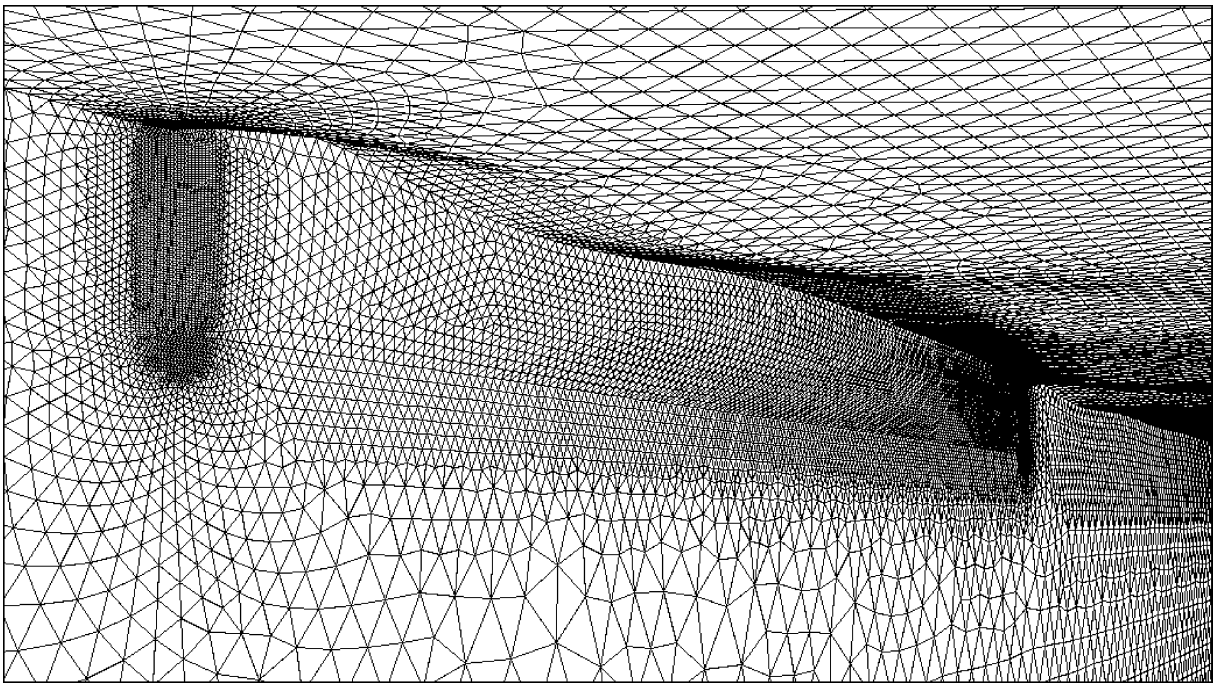

Fig. 6a. Surface mesh (Wigley hull, Fr = 0.25).

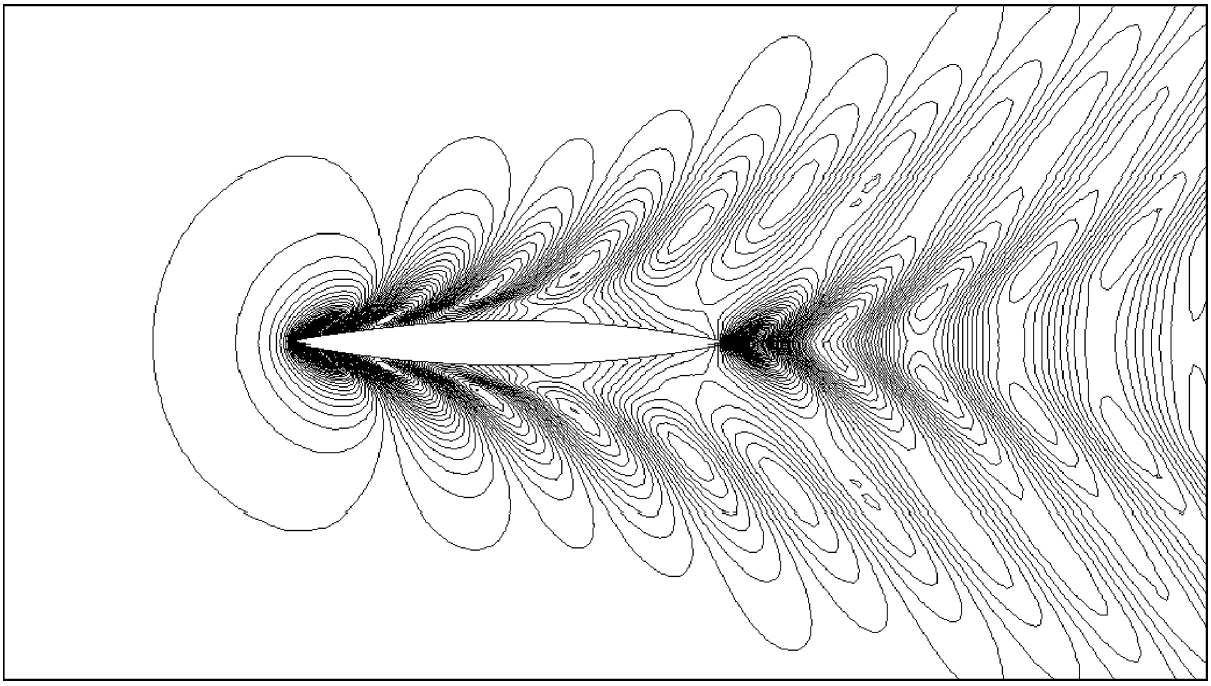

Fig. 6b. Wave elevation contour (Wigley hull, Fr = 0.25).

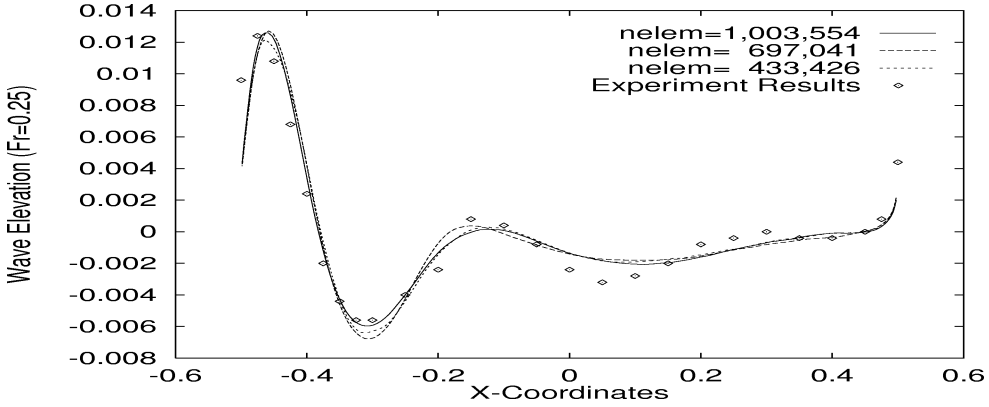

Fig. 6c. Comparison of wave profiles (Wigley hull, Fr = 0.25).

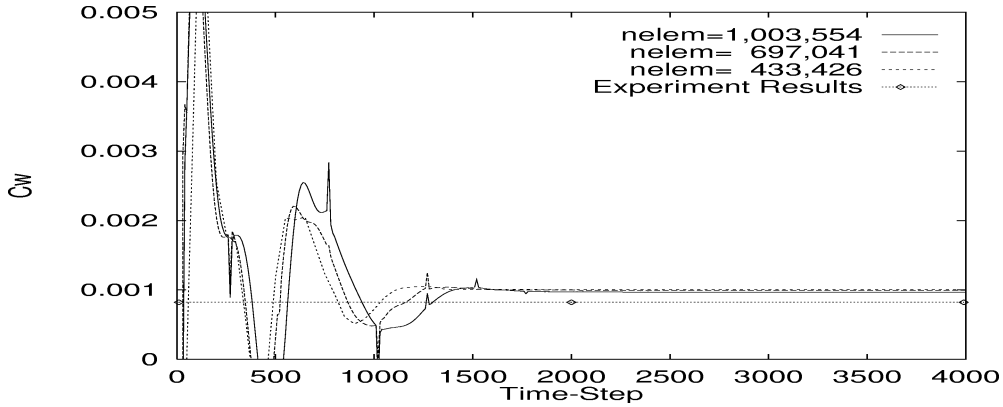

Fig. 6d. Comparison of wave drag coefficients (Wigley hull, Fr = 0.25).

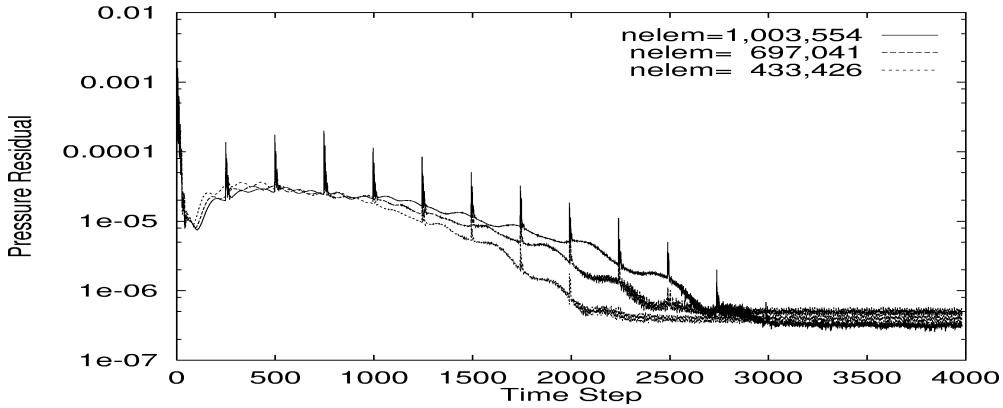

Fig. 6e. Comparison of pressure residuals (Wigley hull, Fr = 0.25).

Table 1
Timings for Wigley hull (R-10000)

nproc	time (sec)	CPU (sec)	Speedup
1	3536	3573	1.00
2	1740	3436	2.05
4	1136	4480	3.14
8	682	5363	5.23
12	494	5810	7.23

show, respectively, the comparison of the wave profile to experimental results, convergence history for the inviscid wave drag coefficient, convergence history for the 3-D flowfield. One can observe that both computed wave profile and wave drag coefficient show better agreement when the computational grid get finer. One can also observe that the residuals increase for a few timesteps after each mesh movement, but that the overall shape of the convergence curve remains unaffected. Table 1 shows the speed-up observed on a 64-processor R-10000 SGI Origin 2000 for 100 timesteps running in shared-memory mode. These timings include the CPU required for the flow solvers, i/o, movie dump-files, build-up of data structures, etc., and may therefore be considered realistic.

6.3. Series 60 model

The third case considered is the Series 60 model, a configuration that was tested experimentally by Toda [43] and modeled numerically by Farmer [11], Tahara [42], Raven [38], as well as others. We first generated a triangulation directly from the offset data provided by DTRC. This triangulation (10,000 triangles) was subsequently used to define, in a discrete manner, the hull. The surface definition of the complete computational domain consisted of discrete (hull) and analytical surface patches. The mesh consisted of approximately 502,492 tetrahedral elements, 102,782 points and 34,513 boundary points. The free surface had 30,150 triangular elements and 15,417 boundary points. The Froude-number was set to $\mathrm{Fr} = 0.32$. Figs. 7(a)–(d) show, respectively, the surface mesh, wave elevation contour at steady state, comparison of the wave profile to experimental results, convergence history for the inviscid wave drag coefficient, and convergence history for the 3-D flowfield. As one can see, the agreement with the experiment is very good along the most part of the hull. The only discrepancy noted is near the stern, which indicates that the strategy employed for the mesh movement in the cruiser stern region needs to be improved further.

6.4. Submerged DARPA submarine model

The fourth case considered is a submerged DARPA submarine model without propulsive effects. The mesh consisted of 319,931 tetrahedral elements, 64,706 points and 20,424 boundary points, and the free surface had 8,769 triangular elements and 4,495 points. The Froude-number was set to $\mathrm{Fr} = 0.25$. Figs. 8(a), (b) show the surface mesh and free surface at steady state. The solution converged in approximately 1,500 timesteps.

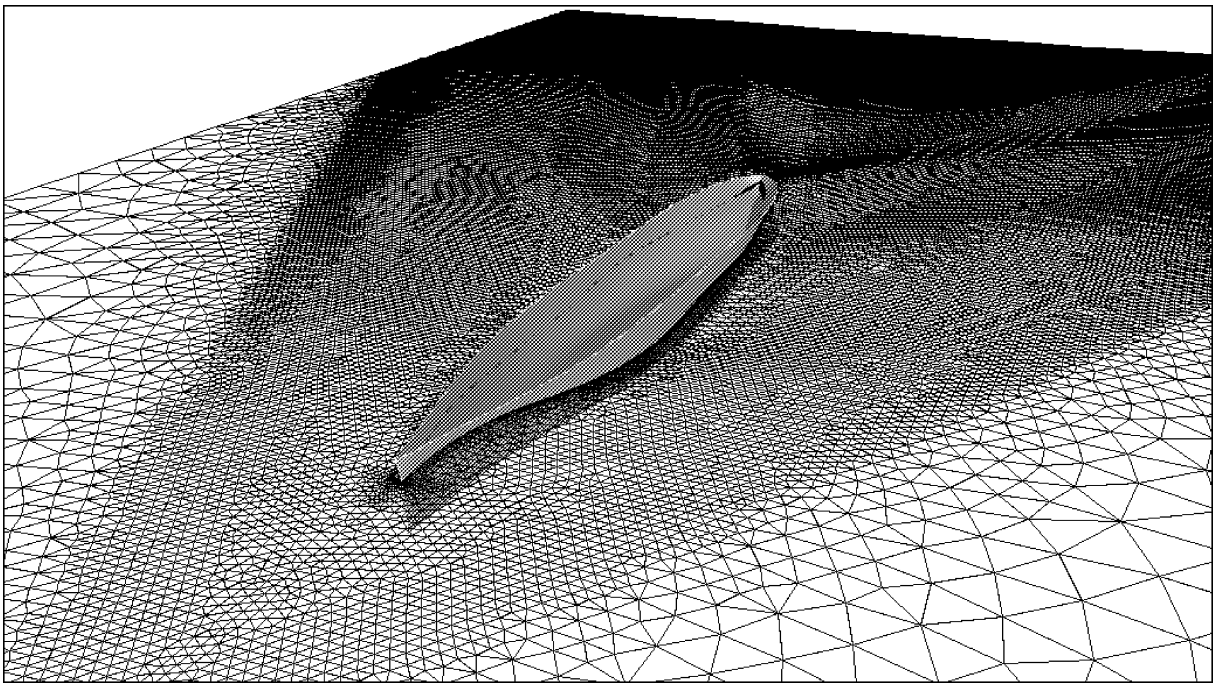

Fig. 7a. Surface mesh (Series 60 hull, Fr = 0.32).

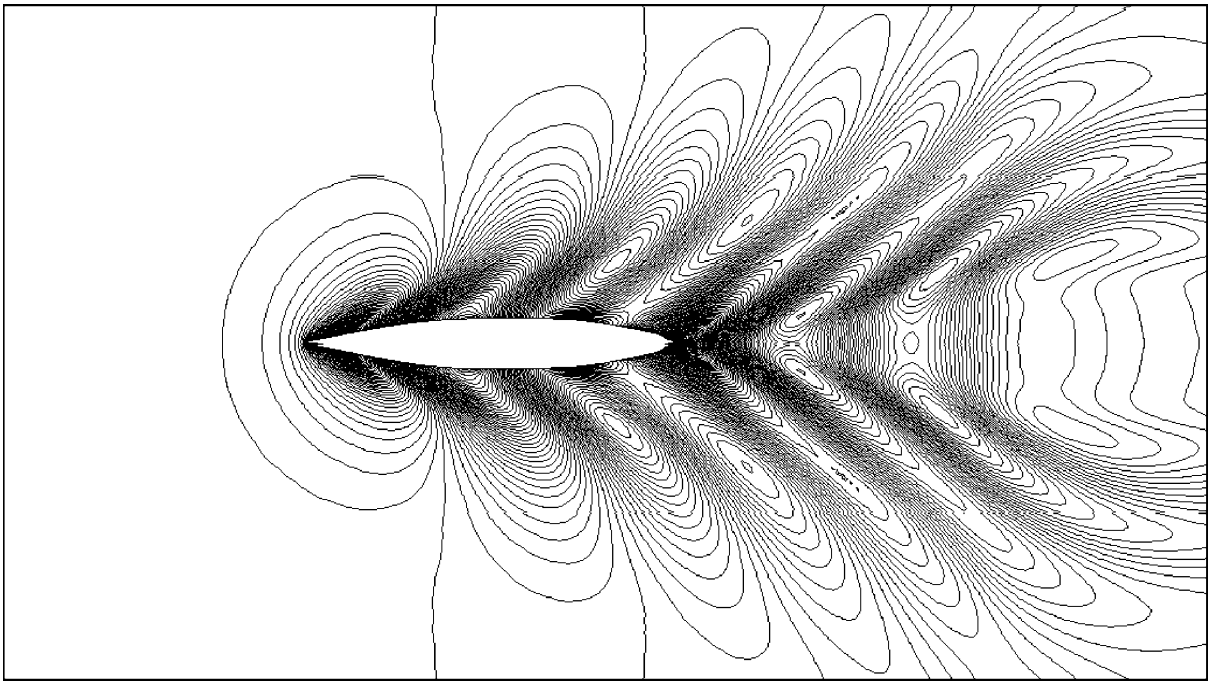

Fig. 7b. Wave elevation contour (Series 60 hull, Fr = 0.32).

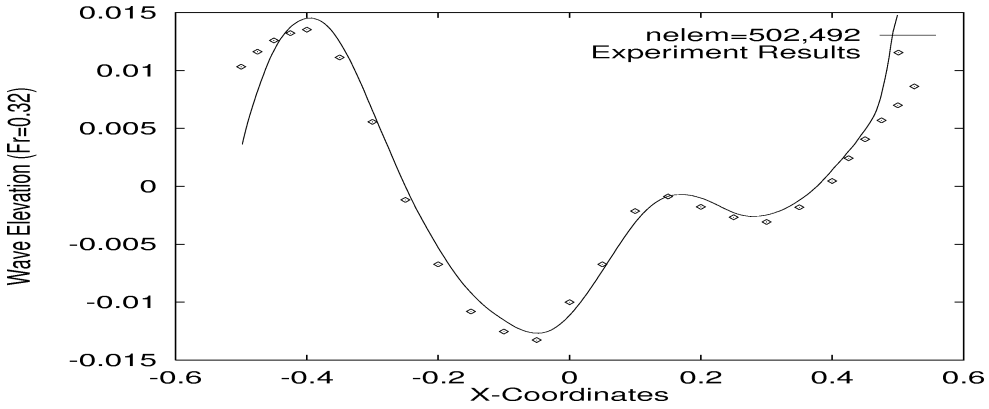

Fig. 7c. Comparison of wave profiles (Series 60 hull, Fr = 0.32).

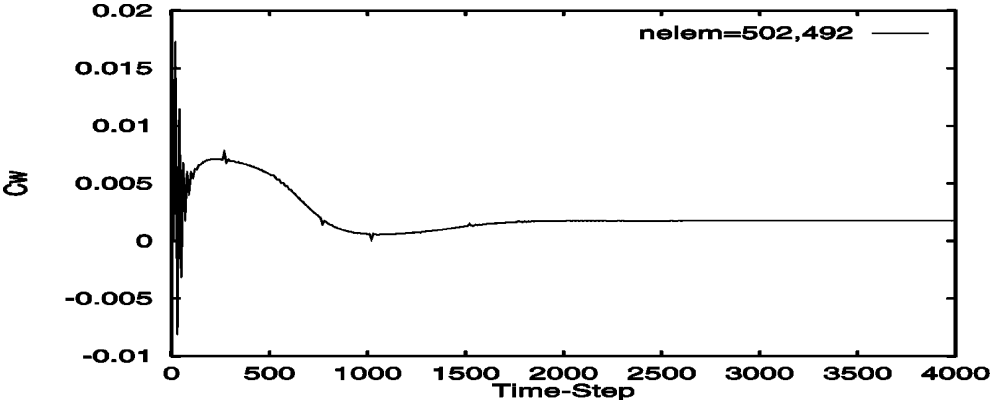

Fig. 7d. Comparison of wave drag coefficients (Series 60 hull, Fr = 0.32).

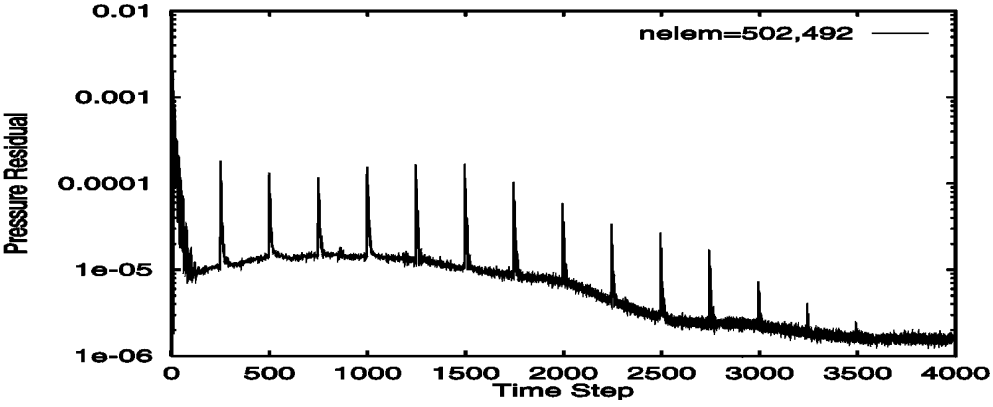

Fig. 7e. Comparison of pressure residuals (Series 60 hull, Fr = 0.32).

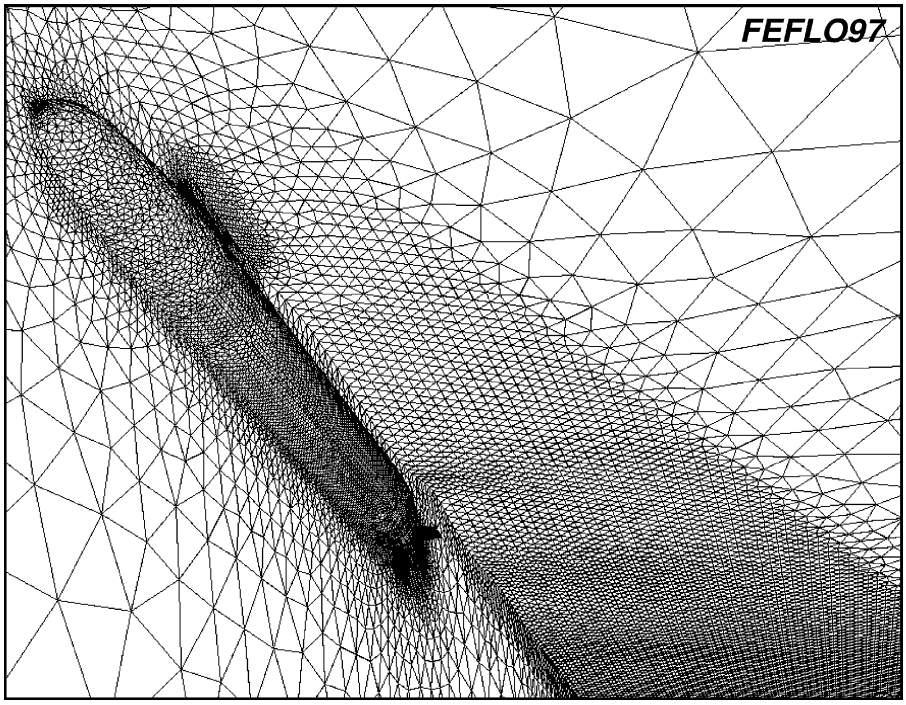

Fig. 8a. Submarine surface mesh.

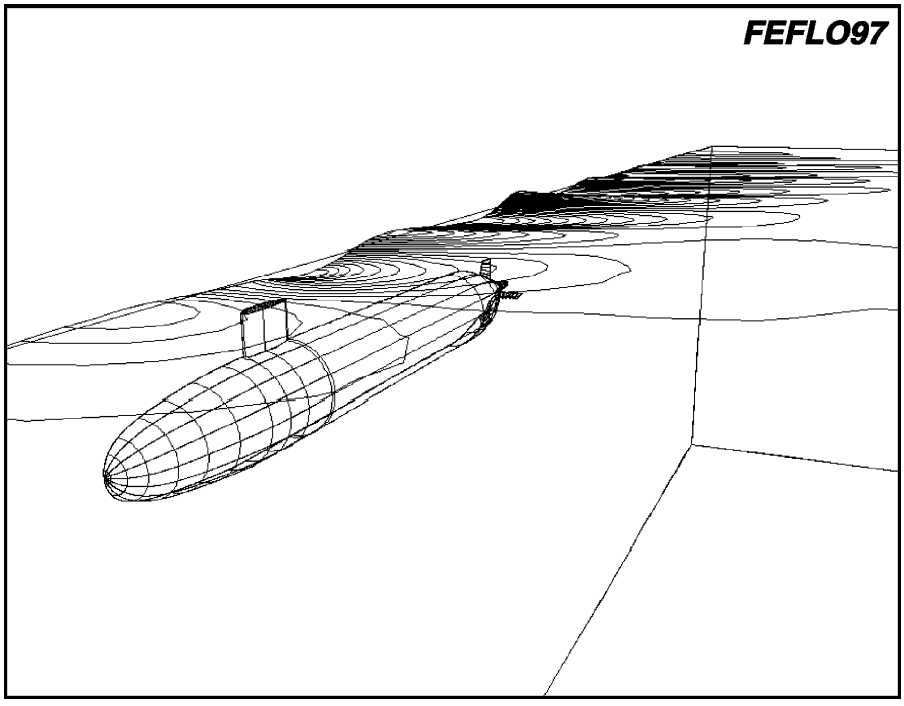

Fig. 8b. Submarine wave pattern.

7. Conclusions and outlook

An unstructured grid-based, parallel free surface solver has been developed. The overall scheme combines a finite-element, equal-order, projection-type 3-D incompressible flow solver with a finite element, 2-D advection equation solver for the free surface equation. For steady-state applications, the mesh is not moved every timestep in order to reduce the cost of geometry recalculations and surface repositioning. A number of modifications required for efficient processing on shared-memory, cache-based parallel machines were implemented. Scalability to a modest number of processors was demonstrated on both shared-memory, cache-based machines, as well as distributed memory parallel machines. The results obtained show good quantitative comparison with experiments and the results of other techniques. The combination of unstructured grids (enhanced geometrical flexibility) and good parallel performance (rapid turnaround) should make the present approach attractive to hydrodynamic design simulations.

The present capability is considered only a first step in a much more ambitious undertaking. Future work will center on:

- extension to transient problems,
- acceleration techniques for steady-state,
- extension to viscous problems,
- coupling to ship motion for seakeeping analysis,
- incorporation of further mesh movement options,
- development of a toolkit for free surface hydrodynamics.

Acknowledgements

A considerable part of this work was carried out while the first author was visiting the Centro Internacional de Métodos Numéricos en Ingeniería, Barcelona, Spain, in the summer of 1996. The support for this visit is gratefully acknowledged. This work was also partially funded by NRL LCP&FD. Dr. William Sandberg was the technical monitor.

The authors would also like to thank Dr. Francis Noblesse, DTRC, for providing some of the data sets and experimental results, as well as his interest and encouragement.

References

[1] B. Alessandrini, G. Delhommeau, A multigrid velocity-pressure-free surface elevation fully coupled solver for calculation of turbulent incompressible flow around a hull, in: Proc. of the 21st Symp. on Naval Hydrodynamics, Trondheim, Norway, 1996, pp. 40–55.

[2] K.J. Bai, J.H. McCarthy (Eds.), Proceedings of the Workshop on Ship Wave-Resistance Computations, Bethesda, MD, USA (1979).

[3] R.F. Beck, Y. Cao, T.-H. Lee, Fully nonlinear water wave computations using the desingularized method, in: Proc. of the 6th Int. Conf. on Num. Ship Hydrodynamics, Iowa City, IA, 1993.

[4] A.J. Chorin, A numerical solution for solving incompressible viscous flow problems, J. Comput. Phys. 2 (1967) 12–26.

[5] A.J. Chorin, Numerical solution of the Navier–Stokes equations, Math. Comp. 22 (1968) 745–762.

[6] Cooperative experiments on Wigley parabolic models in Japan, 17th ITTC Resistance Committee Report, 2nd ed., 1983.
[7] G. Cowles, L. Martinelli, Fully nonlinear hydrodynamic calculations for ship design on parallel computing platforms, in: Proc. of the 21st Symp. on Naval Hydrodynamics, Trondheim, Norway, 1996.
[8] C.W. Dawson, A practical computer method for solving ship-wave problems, in: Proc. 2nd Int. Conf. Num. Ship Hydrodynamics, USA, 1977.
[9] J.H. Duncan, The breaking and non-breaking wave resistance of a two-dimensional hydrofoil, J. Fluid Mech. 126 (1983) 507–516. Also: AIAA J. 32 (6) (1993) 1175–1182.
[10] J.R. Farmer, L. Martinelli, A. Jameson, A fast multigrid method for solving incompressible hydrodynamic problems with free surfaces, AIAA J. 32 (6) (1993).
[11] J.R. Farmer, L. Martinelli, A. Jameson, Multigrid solutions of the Euler and Navier–Stokes equations for a Series 60 Cb = 0.6 hull for Froude numbers 0.160, 0.220 and 0.316, in: Proc. of CFD Workshop, Tokyo, Japan, 1994.
[12] H.J. Haussling, R.W. Miller, Renolds-averaged Navier–Stokes computation of free-surface flow about a Series 60 hull, in: Proc. of CFD Workshop, Tokyo, Japan, 1994.
[13] T. Hino, Computation of free surface flow around an advancing ship by the Navier–Stokes equations, in: Proc. of the 5th Int. Conf. Num. Ship Hydrodynamics, Hiroshima, Japan, 1989, pp. 103–117.
[14] T. Hino, An unstructured grid method for incompressible viscous flows with a free surface, AIAA-97-0862, 1997.
[15] T. Hino, L. Martinelli, A. Jameson, A finite-volume method with unstructured grid for free surface flow, in: Proc. of the 6th Int. Conf. Num. Ship Hydrodynamics, Iowa City, IA, 1993, pp. 173–194.
[16] C. Janson, L. Larsson, A method for the optimization of ship hulls from a resistance point of view, in: Proc. of the 21st Symp. on Naval Hydrodynamics, Trondheim, Norway, 1996.
[17] G. Jenson, H. Soding, Ship wave resistance computation, Finite Approx. Fluid Mech. II 25 (1989).
[18] Y. Kallinderis, A. Chen, An incompressible 3-D Navier–Stokes method with adaptive hybrid grids, AIAA-96-0293, 1996.
[19] K.-J. Kang, M.-S. Shin, S.-H. Van, Numerical simulation of free surface flows around a Series 60 (Cb = 0.6) model ship, in: Proc. of CFD Workshop, Tokyo, Japan, 1994.
[20] Y.H. Kim, T. Lucas, Nonlinear ship waves, in: Proc. 18th Symp. on Naval Hydrodynamics, MI, 1990.
[21] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier–Stokes equations, J. Comput. Phys. 59 (1985) 308–323.
[22] R. Löhner, Some useful renumbering strategies for unstructured grids, Internat. J. Numer. Methods Engrg. 36 (1993) 3259–3270.
[23] R. Löhner, Renumbering strategies for unstructured-grid solvers operating on shared-memory, cache-based parallel machines, AIAA-97-2045-CP, 1997.
[24] R. Löhner, Chi Yang, Improved ALE mesh velocities for moving bodies, Comm. Numer. Methods Engrg. 12 (1996) 599–608.
[25] R. Löhner, R. Ramamurti, A load balancing algorithm for unstructured grids, Comput. Fluid Dyn. 5 (1995) 39–58.
[26] H. Luo, J.D. Baum, R. Löhner, A finite volume scheme for hydrodynamic free boundary problems on unstructured grids, AIAA-95-0668, 1995.
[27] D. Martin, R. Löhner, An implicit linelet-based solver for incompressible flows, AIAA-92-0668, 1992.
[28] L. Martinelli, J.R. Farmer, Sailing through the nineties: Computational fluid dynamics for ship performance analysis and design (Chapter 27), in: D.A. Caughey, M.M. Hafez (Eds.), Frontiers of Computational Fluid Dynamics, Wiley, New York, 1994.
[29] H. Miyata, Time-marching CFD simulation for moving boundary problems, in: Proc. of the 21st Symp. on Naval Hydrodynamics, Trondheim, Norway, 1996, pp. 1–21.
[30] D.E. Nakos, P.D. Sclavounos, On steady and unsteady ship wave patterns, J. Fluid Mech. 215 (1990) 256–288.

[31] D.E. Nakos, P.D. Sclavounos, Kelvin wakes and wave resistance of Cruiser and Transom–Stern ships, J. Ship Research 38 (1994).
[32] F. Noblesse, X.-B. Chen, C. Yang, Fourier–Kochin theory of free surface flows, in: Proceedings of the 21st Symposium on Naval Hydrodynamics, Trondheim, Norway, 1996.
[33] F. Noblesse, J.H. McCarthy (Eds.), Proceedings of the 2nd DTNSRDC Workshop on Ship Wave-Resistance Computations, Bethesda, MD, 1983.
[34] J. Peraire, K. Morgan, J. Peiro, The simulation of 3D incompressible flows using unstructured grids (Chapter 16), in: D.A. Caughey, M.M. Hafez (Eds.), Frontiers of Computational Fluid Dynamics, Wiley, New York, 1994.
[35] R. Ramamurti, R. Löhner, A parallel implicit incompressible flow solver using unstructured meshes, Comput. Fluids 5 (1996) 119–132.
[36] H.C. Raven, A practical nonlinear method for calculating ship wavemaking and wave resistance, in: Proc. of the 19th Symp. on Naval Hydrodynamics, Seoul, Korea, 1992.
[37] H.C. Raven, Nonlinear ship wave calculations using the RAPID method, in: Proc. of the 6th Int. Conf. on Num. Ship Hydrodynamics, Iowa City, IA, 1993.
[38] H.C. Raven, A solution method for the nonlinear ship wave resistance problem, Doctoral Thesis, Maritime Research Institute, Netherlands, 1996.
[39] A.M. Reed, J.G. Telste, C.A. Scragg, D. Liepmann, Analysis of Transom–Stern flows; Proc. 17th Symp. on Naval Hydrodynamics, The Hague, The Netherlands, 1990.
[40] A. Rizzi, L. Eriksson, Computation of inviscid incompressible flow with rotation, J. Fluid Mech. 153 (1985) 275–312.
[41] H. Soding, Advances in panel methods, in: Proc. of the 21st Symp. on Naval Hydrodynamics, Trondheim, Norway, 1996.
[42] Y. Tahara, F. Stern, A large-domain approach for calculating ship boundary layers and wakes for nonzero Froude number, in: Proc. of CFD Workshop, Tokyo, Japan, 1994.
[43] Y. Toda, F. Stern, J. Longo, Mean flow measurements in the boundary layer and wake and wave field of a Series 60 $c_b = .6$ ship model for Froude numbers .16 and .316, IIHR Report 352, Iowa Institute of Hydraulic Research, 1991.
[44] B. van Leer, Towards the ultimate conservative scheme. II. Monotonicity and conservation combined in a second order scheme, J. Comput. Phys. 14 (1974) 361–370.
[45] J.V. Wehausen, The wave resistance of ships, Adv. Appl. Mech. (1970).
[46] F. Xia, Numerical calculation of ship flows with special emphasis on the free surface potential flow, Doctoral Thesis, Chalmers University of Technology, Sweden, 1986.