UNIVERSITE DE MONTREAL

Solution of the 3-D Euler Equations using a Multigrid Method

par

Stéphane Major

DEPARTEMENT DE GENIE MECANIQUE

ECOLE POLYTECHNIQUE

RAPPORT DE PROJET PRESENTE EN VUE DE L'OBTENTION

DU GRADE DE MAITRE EN INGENIERIE (M. Ing.)

GENIE MECANIQUE

Mai 1992

UNIVERSITE DE MONTREAL

ECOLE POLYTECHNIQUE DE MONTREAL

Ce rapport de projet intitulé:

Solution of the 3-D Euler Equations Using a Multigrid Method

présenté par : Stéphane Major

en vue de l'obtention du grade de : Maître en Ingénierie

a été dûment accepté par le jury constitué de :

Prof. Barun C. Basu, directeur de recherche

Prof. Ion Paraschivoiu

SOMMAIRE

La résolution des équations d'Euler est une façon efficace et économique d'obtenir des approximations par ordinateur (CFD) d'écoulements aérodynamiques complexes. Ces équations d'Euler sont intéressantes car elles sont obtenues directement des équations les plus précises que l'on connaisse: les équations de Navier-Stokes. En fait, pour obtenir les équations d'Euler, il suffit de négliger les termes visqueux des équations Navier-Stokes. En conséquence les équations d'Euler permettent de résoudre un écoulement compressible, non-visqueux, instationnaire et rotationel.

En trois dimensions, la résolution de ces équations requière un très grand nombre d'opérations mathématiques et un espace mémoire très vaste, car le programme doit résoudre simultanément cinq équations aux dérivées partielles. Il est donc de rigueur d'introduire, dans la formulation numérique et informatique, des accélérateurs de convergence. Ces accélérateurs ont pour seul but d'obtenir des résultats satisfaisants avec le plus petit nombre d'itérations possible, minimisant ainsi les coûts. Pour cela on utilise de la viscosité artificielle qui stabilise le schéma numérique, un intégrateur Runge-Kutta dans le temps et on veut introduire, ce qui est le but de ce projet, un modèle multigrid.

Le schéma multigrid est une technique qui a été développée au début des années 80 et dont il y a presqu'autant de variations qu'il y a de programmes qui résolvent des écoulements numériquement. Ce projet aborde un schéma très simple qui performe correctement en 2-D. Lors de ce projet le modèle 2-D est modifié pour le rendre compatible avec un programme 3-D qui est fourni . Ce schéma multigrid est sélectionné d'abord et avant tout pour sa simplicité et car il a été élaboré par le concepteur du programme Euler.

Le schéma 3-D est réalisé en s'assurant qu'il respecte la conservation des paramètres sur un maillage 3-D et qu'il soit conforme aux principes établis en 2-D. Les résultats qui sont obtenus en 3-D sont moins intéressants que ceux en 2-D, bien qu'une amélioration du taux de convergence d'environ 10% soit notée après 400 itérations. Il semble que l'effet du schéma multigrid prenne de l'ampleur au fur et à mesure des itérations. En 2-D, on pouvait se permettre de faire de 1000 à 2000 itérations sans que le coût soit prohibitif, et dans ces conditions le multigrid permettait de doubler la précision du calcul avec 50% moins d'itérations. Le problème en 3-D est que le coût par itération est beaucoup plus élevé, même pour un maillage qui n'est pas très précis (48x16x16 cellules). On parle ici d'environ 10$ CPU par itérations, donc 2000 itérations sont suffisantes pour vider un compte de 20000$. En 3-D il est donc recommandé de faire un effort supplémentaire pour introduire un schéma multigrid plus efficace, au prix d'une complexité accrue.

ABSTRACT

Euler solvers are very efficient CFD tools for the prediction of complex aerodynamic flows. The Euler equations provide precise predictions since they are directly obtained from the Navier-Stokes equations (the only simplification is the elimination of viscous terms). Consequently an Euler solver produces a solution for an unsteady, compressible, invicid and rotational flow.

A 3-D computation of the Euler equations requires large memory capacities and involves a great number of mathematical operations. These requirements are originating from the fact that Euler equations are a set of five partial derivative equations that are to be solved simultaneously. In order to be able to use efficiently this type of solver, the computer program must include convergence accelerators that reduce the time needed to obtain reasonable results. The basic solver is a finite volume scheme, stabilized using artificial viscosity. A Runge-Kutta time stepping scheme is already in place to speed up the integration in time. The purpose of this project is to implement a multigrid scheme accelerating convergence in the spatial domain.

Multigrid is a technique, developed in the early 1980's, for which a quantity of variants exist. For this project a very simple method was studied and applied to the original solver that was provided. This simplified scheme is an extension of a 2-D approach developed by the author (S. Saha). According to some tests discussed in the author's thesis (1), his 2-D multigrid scheme improves satisfactorily 2-D Euler solver's convergence rate.

The 3-D extension of the multigrid scheme is realized while keeping in mind the principles established in 2-D and making sure that parameters are conserved on the 3-D grid. Results obtained in 3-D are not as interesting as those discussed in 2-D. Still a 10% increase in the convergence rate is observed after 400 iterations.

It seems that the improvement of the convergence is more important as the number of iteration increases. The effective difference between 2-D and 3-D resides in the cost of memory space and computer time needed to solve for a given number of iterations. In 2-D, it is possible to perform from 1000 to 2000 iterations without spending prohibitive amount of money (CPU money = CPU time). In 3-D, for a medium quality grid having 48x16x16 cells the cost of a single iteration is around 10$ CPU. This means that 2000 iterations involve an expense of 20000$ CPU (and a lot of real time), which is not very reasonable in Ecole Polytechnique's context. It is therefore recommended, for 3-D problems, to implement another, more efficient, multigrid scheme (and more complex) which will allow a larger reduction of the residual error in a limited number of iterations.

RESUME FRANCAIS

R.1 INTRODUCTION

Depuis les années 1970 le domaine de l'aérodynamique a été révolutionné par l'apparition de divers modèles numériques qui permettent de solutionner des écoulements complexes. Le CFD (Computational Fluid Dynamics) est alors né. Les lois de la mécanique des fluides fournissent un modèle mathématique qui caractérise les écoulements aérodynamiques. Il s'agit des équations de Navier-Stokes. A l'heure actuelle il est assez difficile de résoudre les équations de Navier-Stokes pour des cas complexes, car la modélisation de la turbulence représente toujours une difficulté.

Le modèle qui offre le plus de précision et qui est bien maîtrisé à l'heure actuelle est le modèle des équations d'Euler. Les équations d'Euler ne sont ni plus ni moins que les équation Navier-Stokes avec les termes visqueux en moins. Il y a bien sûr plusieurs autres modèles numériques qui sont construits à partir d'équations de plus en plus simplifiées. Il est à noté qu'un modèle d'Euler est assez complexe, il requière la résolution simultanée de cinq équations aux dérivées partielles non-linéaires et couplées.

Le projet discuté dans ce rapport consiste à introduire un schéma multigrid dans un programme de résolution des équations d'Euler (code Euler). Un schéma multigrid c'est un accélérateur de convergence. Ce type schéma est utile car il permet en général d'augmenter le taux de convergence d'un modèle donné sans affecter la qualité des résultats. Le taux de convergence tel qu'utilisé dans ce rapport est lié au nombre d'itérations nécessaires pour atteindre un niveau de précision donné. La précision quant à elle est mesurée à l'aide de la valeur de l'erreur moyenne et maximum de la quantité de mouvement dans la direction principale de l'écoulement.

R.2 RESOLUTION NUMERIQUE DES EQUATIONS D'EULER

Les équations d'Euler peuvent être exprimées sous une forme conservative, qui se prête particulièrement bien à une résolution numérique. En fait le sytème se ramène alors à une seule équation vectorielle telle que montré aux équations 1.1 à 1.3. Pour le genre de problèmes qui nous occupent il est aussi possible de supposer que l'enthalpie totale dans l'écoulement est constante. Il est alors possible de supprimer l'équation de l'énergie et ainsi économiser 20% de temps de calcul.

A l'intérieur du programme, les équations sont résolues sous une forme intégrale, eq. 1.4 et 1.5 qui est équivalente à la forme différentielle discutée ci-dessus. La méthode numérique de solution de cette forme intégrale est appelée méthode des volumes finis. Ce type de modèle est intimement lié à son inventeur le Dr. Jameson (2,3), expliquant pourquoi plusieurs personnes parlent de modèles de type Jameson.

Partant du fait que l'ensemble du domaine autour de l'aile est discrétisé par un ensemble fini et structuré de petits volumes. Le principe de base est alors que la solution pour l'ensemble du domaine est obtenu par la résolution successive de la forme intégrale sur chacun des petits volumes lors de chacune des itérations.

Pour pouvoir résoudre numériquement la forme intégrale sur chacun des volumes il faut remplacer cette dernière par une équation discrète. La forme discrete utilisée ici est dite centrée car une seule valeur moyenne au centre de la cellule est évaluée. L'équation 1.6 est la forme discrete qui est solutionnée dans le programme. Dans cette équation, Ω_{IJK} représente le volume d'une cellule de coordonnée i,j,k, Q_{IJK} est la moyenne volumétrique des paramètres au centre de la cellule et $\delta\ [\ H(Q)\cdot S\ ]_{IJK}$ est la somme des flux à travers les surfaces de la cellule. Pour résoudre l'équation 1.6 il faut connaître $H(Q)$ sur les surfaces de la cellule, ce qui est fait en prenant la moyenne des Q entre les cellules voisines. Cette approche par volumes finis a comme principal avantage qu'elle est valide sur n'importe quel système de coordonnées curvilignes.

Le schéma volumes finis tel que présenté ci-dessus est de nature dispersive, ce qui signifie qu'il peut présenter certaines instabilités et causer des oscillations à proximité des ondes de choc. Pour résoudre ce problème, des termes supplémentaires, de nature dissipative, sont introduits pour stabilisé le schéma. Ces termes sont connus sous le nom de viscosité artificielle. La façon la plus efficace d'insérer ces termes dans la formulation est celle qui a été définit par Jameson (2,4), qui consiste à utiliser une combinaison de termes du deuxième ordre et du quatrième ordre. Les termes du deuxième ordre adoucissent la solution là où des discontinuités apparaissent. Les termes du quatrième ordre servent à limiter l'effet des termes du deuxième ordre là où la solution se comporte bien à priori. Les expressions mathématiques de ces différents termes sont données aux équations 1.7 à 1.12 et l'équation 1.13 exprime la nouvelle forme discrète où la viscosité artificielle a été ajoutée.

Les équations 1.14 à 1.20 illustrent comment l'intégration dans le temps est effectuée grâce à un procédé Runge-Kutta en trois étapes. Le résultat de ce type d'intégration numérique est précis à ordre deux. Le traitement des conditions frontières quant à lui est des plus classiques. Bien qu'une imposition correcte des conditions frontières est essentielle pour obtenir des résultats valables aucune élaboration approfondie n'est faite ici. Comme remarque finale, il est à noté que le modèle tel que décrit dans cette section est suffisant pour résoudre un cas réel, mais son taux de convergence n'est pas très bon, d'où la nécessité d'introduire un nouvel accélérateur de convergence.

R.3 SCHEMA MULTIGRID

Le principe de base d'un schéma multigrid est de prendre avantage du fait que sur un maillage grossier l'intégration dans le temps progresse plus rapidement que sur un maillage fin, en plus de faire intervenir moins d'inconnues. Le problème avec un maillage grossier est que l'erreur de troncature est très élevée, et il est certain que la précision n'est pas très bonne. Le schéma multigrid fait donc intervenir un cycle de solution qui passe par un maillage fin, puis un maillage plus grossier et ainsi de suite. Lorsque la solution est obtenu sur le maillage le plus grossier, les paramètres sont alors interpolés vers le maillage le plus fin, puis la boucle recommence.

L'utilisation d'une telle procédure permet d'augmenter grandement le taux de convergence car elle facilite la propagation rapide des corrections du maillage fin à tout le domaine tout en gardant les erreurs de troncature basses. Les maillages grossiers nécessaires pour cette procédure sont générés ici en supprimant un point sur deux dans chacune des directions relatives, tel que montré à la figure 2.1. Notant h la longueur caractéristique d'une maille fine les mailles grossières seront notées par 2h, 4h, etc. Le schéma multigrid, tel que décrit ci-dessus, est utilisé dans pratiquement tous les programmes de Jameson (2,3).

Implanter ce type de schéma s'avère une tâche assez complexe, il a donc été décidé d'étudier un schéma simplifié qui est plus à la mesure d'un projet de six crédits. La méthode simplifié qu'a utilisé Saha (1) en 2-D a donc été généralisée pour un programme en 3-D. Le schéma simplifié calcule les corrections sur le maillage fin et par la suite propage ces corrections dans le domaine grâce à des maillages grossiers sans toutefois faire de nouveaux calculs sur ces maillages. Ceci fait en sorte qu'on ne peut bénéficier de plus grands pas dans le temps.

Les corrections sont propagées par un processus d'accumulation sur des mailles de plus en plus grosses (jusqu'au maillage le plus gros), suivit par une interpolation successive des valeurs vers le maillage fin. La valeur de la correction sur le maillage fin est alors modifiée par une moyenne pondérée de l'ancienne valeur et de la valeur interpolée. Cette moyenne pondérée est présentée à l'équation 2.1. Dans cette équation les P sont les opérateurs spatiaux aux différents niveaux, I est un interpolateur et w est le facteur de pondération. Les opérateurs sur les mailles grossières sont obtenus par des moyennes volumétriques tel que montré aux équations 2.2 et 2.3.

Le qualité du processus d'interpolation peut influencer grandement l'efficacité du schéma multigrid. En effet, si l'interpolation n'est pas conservative, le schéma multigrid va introduire des erreurs et pourrait donc avoir l'effet inverse de celui escompté. Pour choisir un modèle d'interpolation une série de tests a été effectuée. Plusieurs approches, décrites à la section 2.3.1, ont été élaborées et testées sur un cas simple en 2-D. Les résultats de ces tests sont présentés à l'annexe A. La méthode choisie consiste à reporter la valeur d'une grosse maille sur le noeud du maillage fin qui correspond à son centre (noeuds pairs du maillage fin). Les autres noeuds sont ensuite définis par la moyenne de leurs voisins. La valeur pour chacune des cellules fines est obtenue par la moyenne des huit noeuds qui la composent. Ensuite une moyenne est faite à toutes les cellules entre la nouvelle valeur et la valeur de l'opérateur sur la cellule grossière qui englobait la cellule fine.

R.4 PROGRAMMATION DU SCHEMA MULTIGRID

Avant d'introduire de nouvelles sous-routines dans le programme de départ il est important de bien en comprendre la structure et de le préparer, si nécessaire, à recevoir du nouveau code. Le programme est structuré en trois sections, soit la préparation des conditions initiales, le calcul et la sortie des résultats.

Pour préparer les conditions initiales du calcul le programme doit d'abord lire les fichiers d'entrée, incluant le maillage. Ensuite il faut calculer la courbure des surfaces de cellules qui forment la surface de l'aile et les volumes des cellules. Il faut aussi évaluer les normales de toutes les surfaces des cellules, et finalement les conditions à l'infini sont imposées comme conditions initiales dans toutes les cellules du maillage. La séquence d'exécution de ces opérations et la description des fichiers d'entrée sont fournies dans la section 3.1.1 du rapport. La section calcul se résume à une grosse boucle DO qui se répète aussi longtemps que le maximum d'itérations n'est pas atteint. Dans chacune des passes de la boucle les opérateurs spatiaux sont évalués trois fois comme le requière l'intégration Runge-Kutta dans le temps. La progression des calculs est décrite dans la section 3.1.2 du rapport. La sortie des résultats est de la loin la partie la plus simple du programme et ne mérite pas qu'on s'y attarde. Les fichiers de sortie sont décrits dans la section 3.1.3 du rapport.

De petites modifications ont été effectuées avant d'introduire les nouvelles sous-routines du schéma multigrid. D'abord les suffixes des fichiers sont devenus MULTI au lieu de EULER. Certains commentaires ont été ajoutés à travers le programme et certaines des opérations qui se trouvaient dans le programme principal ont été transférées dans des sous-routines indépendantes.

Avec l'implantation du schéma multigrid sont venues s'ajouter de nouvelles variables dans le programme ainsi que deux nouvelles informations lues dans le fichier de données (INPUT MULTI). Ces nouvelles données sont MGL, le nombre total de maillages qui seront impliqués dans le multigrid et OMEG, le facteur de pondération utilisé dans la correction de l'opérateur spatial.

Enfin, deux nouvelles sous-routines sont ajoutées. La première est COARSE, qui prépare les maillages grossiers qui seront utilisés dans le schéma. Au cours de cette sous-routine il faut définir les niveaux de maillage, identifier les noeuds des nouvelles cellules, calculer les nouveaux volumes et les storer. Le listing complet de cette sous-routine est fournis à l'annexe B. La seconde sous-routine est MULTI, qui effectue les différentes étapes du schéma multigrid simplifié telles que décrites plus tôt. Le listing complet de cette sous-routine est fournis à l'annexe C. Un guide d'utilisation du nouveau programme est inclu à l'annexe D.

R.5 RESULTATS ET DISCUSSION

Le résultat le plus important à vérifier pour évaluer la performance du schéma multigrid est de toute évidence la courbe de l'évolution de l'erreur avec le nombre d'itération. Il est aussi important de s'assurer que le schéma n'induit pas de perte de précision dans la représentation des distributions de pression sur l'aile. Le cas standard de calcul qui a été étudié est celui de l'aile Onera M6 à un angle d'attaque 3.06 degrés et un nombre de Mach de 0.84. Ce choix est fait car c'est un cas bien documenté et que des données expérimentales sont disponibles. Il est bon de rappeler que les calculs sont effectués sur un maillage type OH avec 48x16x16 cellules, le nombre de Courant est 1.8 et les constantes de viscosité artificielle sont c2 = 2.0 et c4 = 1.0. Les courbes de convergences sont données aux figures 4.1 et 4.2, alors que les courbes de C_p sont sur les figures 4.3 à 4.6.

Le taux de convergence obtenu avec le multigrid montre une légère amélioration par rapport au calcul non accéléré, de l'ordre d'environ 10%. Ceci est visible surtout sur la courbe de l'erreur moyenne. Il est à noter que dans ce cas comme dans d'autres qui ont été étudiés, l'écart entre les deux courbes de convergence semble être plus visible au fur et à mesure que les calculs progressent. Il est bien évident que la méthode simplifiée de multigrid est beaucoup moins efficace que les schémas présentés dans la littérature pour lesquels les courbes de convergence ressemblent à la figure 2.3.

Les distributions de pression qui sont présentées pour quatre sections de l'aile démontrent hors de tout doute que le multigrid ne diminue pas la qualité des prédictions. En fait il semble même qu'à certain moment la prédiction avec le multigrid soit meilleure que celle obtenue avec le programme original, sourtout autour du pic de pression.

R.6 CONCLUSION

Malgré son succès apparant en 2-D, le schéma multigrid simplifié en 3-D ne permet pas une amélioration suffisante du taux de convergence pour justifier les calculs supplémentaires qui sont effectués à chaque itération. Peut être pourrait-on gagner quelque peut si un très grand nombre d'itérations étaient effectuées, mais une fois de plus ceci n'est pas réaliste dans le contexte de l'Ecole Polytechnique car le coût de chacune des itérations en 3-D est d'environ 10$ CPU.

La conclusion de ce projet est donc qu'il faut introduire un schéma complet, bien que beaucoup plus complexe, pour augmenter significativement le taux de convergence et ainsi favoriser l'utilisation fréquente d'un code Euler 3-D. Il serait peut-être intéressant de faire l'étude de certains autres accélérateurs de convergence tel que le "implicit smoothing" par exemple. L'accès à une station graphique munie d'un programme de visualisation en 3-D n'était pas possible lors de la réalisation de ce projet. Ceci a eu pour effet de ralentir considérablement la poursuite des travaux. Il serait donc important que ce type d'outil de travail soit à l'avenir disponible pour des travaux semblables à ce projet.

ACKNOWLEDGEMENT

The author would like to thank everybody who supported and helped him through the project and the writing of the report.

Special thanks are addressed to Professor Basu and Professor Paraschivoiu who accepted to supervise and judge this work.

The author also wishes to thank the Natural Science and Engineering Research Council of Canada for its financial support during his Master program.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDIX

INTRODUCTION

In the past twenty years, the development of accurate and effective numerical prediction methods revolutionized the aerodynamic field. CFD (Computational Fluid Dynamics) methods permits now to have cost effective evaluation of flow parameters, even for transonic cases where the governing equations are non-linear. Cost effectiveness of CFD methods is related to the fact that a lot of very expensive wind tunnel tests can now be avoided and replaced by numerical predictions. The CFD methods also allow the aerodynamic engineer to investigate more complex flow problems, even some problems that would almost be impossible or prohibitively expensive to test in wind tunnels.

There is a wide variety of flow solvers circulating in industries and universities at the time being. They do not all provide the same degree of reliability and precision. Fluid dynamics theory prescribes that the most accurate solution for aerodynamic flows should be obtained from the resolution of Navier-Stokes equations. The resolution of Navier-Stokes equations requires extensive computational capacities, and even nowadays most Navier-Stokes solvers are solving simple 2-D problems. Since it is not very practical to solve these equations, it is essential to look for simplifications.

Neglecting the viscous terms of the Navier-Stokes equations permits to rewrite them in a new form known as the Euler equations. An Euler flow solver is still quite complex, because it requires the simultaneous solution of five coupled non-linear equations. But the degree of sophistication of computer technology today allow the use of such solvers for almost any kind of problems (even complex 3-D problems). These flow solvers are, for the moment, the best solvers widely used by the aeronautical industry. The subject of the current project is related to this type of solver.

The project presented in this report deals with the introduction of a multigrid scheme in 3-D isolated wing Euler solver. Multigrid schemes are convergence accelerators. It will be discussed in details how such schemes can be applied to the resolution of Euler equations. For this project various multigrid schemes are examined and a selection is made based on simplicity. It is expected that the convergence accelerator will reduce the number of iterations needed to reach a certain level of precision, and even allow the solver to reach more precise solutions. The degree of precision is measured here by the order of magnitude of the maximum and average variation of the momentum in the x direction.

Further simplifications of the equations are not relevant to the execution of this project, therefore, simpler CFD models will not be discussed in this report. The details about the generation of a grid around a wing, will also not be presented in this report, because it is assumed that an algebraic grid generation program is provided and working perfectly. There will be a discussion on the interaction between the grids and the quality of the results that can be expected, and also the possible links between the grids and the multigrid schemes. All calculations are made on an O-H grid topology modelling a full 3-D Onera M6 wing.

The different calculations, necessary for the realization of this project, are made on the Ecole Polytechnique main frame via the CMS operating system. The capacities (memory and available CPU time) of this system are sufficient to solve for relatively precise grids (48x16x16 cells).

The report contains four chapters. The first chapter presents the Euler equations, and explain how they are solved for steady flow by integrating the unsteady equations toward steady state (Jameson's finite volume multi stage integration). The second chapter elaborates on the theory behind multigrid schemes, describes possible formulations suitable for this project and gives all details related to the scheme finally selected. The third chapter gives some insights on the original Euler solver, explains all avenues that were studied and programmed, as well as the structure of the final program. The fourth and final chapter presents the most interesting results and provides a discussion on the quality of the results.

CHAPTER I

3-D EULER SOLVER

1.1 INTRODUCTION

The Euler equations are extremely useful because they can predict very accurately shock location on a wing, they can determine rotational flow field behind a strong shock, they admit distributed vorticities in the flow field and they satisfy correctly the Rankine-Hugoniot jump conditions across a shock. The Euler equations are capable of providing realistic solutions to very complex problems such as : unique shock solution in converging-diverging passages formed by turbomachinery blades, separated vortex flows around delta wings (small aspect ratio) with sharp leading edges, and flows where shock induced separation occurs.

1.2 MATHEMATICAL FORMS FOR THE EULER EQUATIONS

The Euler equations are obtained, as mentioned previously, by taking the Navier-Stokes equations and neglecting all viscous terms.

1.2.1 Conservative form

The resulting five equations can be written in a conservative form that are presented on top of next page.

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}}{\partial x} + \frac{\partial \vec{F}}{\partial y} + \frac{\partial \vec{G}}{\partial z} = 0 \qquad (1.1)$$

where

$$\vec{Q} = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{Bmatrix} \quad \vec{E} = \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho u h_o \end{Bmatrix} \quad \vec{F} = \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ \rho v h_o \end{Bmatrix} \quad \vec{G} = \begin{Bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho w h_o \end{Bmatrix} \qquad (1.2)$$

and

$$h_o \equiv \frac{e + p}{\rho} = \frac{\gamma}{\gamma - 1} \frac{p}{\rho} + \frac{1}{2}\left(u^2 + v^2 + w^2\right) \qquad (1.3)$$

In the above equations, ρ is the density, u, v and w are the velocity components corresponding to the x, y and z cartesian coordinates, e refers to the total internal energy per unit of volume, p is the pressure, h_o is the total enthalpy, and γ is the specific heat ratio.

1.2.2 Homoenergetic hypothesis

In the situation of flow evaluation about a finite 3-D wing it is very useful and correct to assume that the flow is homoenergetic (h_o = constant). This hypothesis permits to eliminate the energy equation from the solution procedure and then economize on computing time. The pressure is then determined directly from equation (1.3). Computations made during this project are without solving the energy equation.

1.2.3 Integral form

For the type of Euler solver that is built in the original program (program provided at the beginning of the project) the integral form of the Euler equations (1.4) and (1.5) are discretized directly; this is called a finite volume approach. The finite domain integral system is :

$$\frac{\partial}{\partial t}\iiint_{\Omega} \vec{Q}\, dV + \iint_{\partial\Omega} \vec{H}\cdot\vec{n}\, dS = 0 \qquad (1.4)$$

where

$$\vec{Q} = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \end{Bmatrix} \qquad \vec{H} = \begin{Bmatrix} \rho u \\ \rho u^2+p \\ \rho uv \\ \rho uw \end{Bmatrix}\vec{i} + \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2+p \\ \rho vw \end{Bmatrix}\vec{j} + \begin{Bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2+p \end{Bmatrix}\vec{k} \qquad (1.5)$$

In this new form, where the homoenergetic hypothesis is applied, the vector $\vec{Q}$ represents the density and the rectangular components of momentum corresponding to the cartesian system x, y and z. The quantity $\vec{H}$ corresponds to the net flux of $\vec{Q}$ transported across a closed surface $\partial\Omega$ as well as the pressure acting on this surface that bounds a volume Ω with a unit normal vector $\vec{n}$.

1.3 CELL CENTRED DISCRETIZATION SCHEME

The scheme, that is now described in detail, is presented using the notation given in S. Saha's thesis (1). Originally the finite volume schemes for the resolution of the Euler equations were developed by Jameson et al. (2,3). It can be said that this type of resolution scheme is probably the most popular and most tested at the time being.

1.3.1 Finite volume discretization

Starting with the idea that the entire volume around the 3-D finite wing is discretized by a structured grid of finite cells, having relative coordinates i, j and k. The grid generation program builds the mesh according to the following conventions : i is going around the wing from and back to trailing edge; j originates from the symmetry plane of the wing and augments in the spanwise direction; k starts from the surface of the wing and extends up to the outer boundaries of the domain. On each side of a given cell, the unit normal vectors $\vec{n}_i$, $\vec{n}_j$ and $\vec{n}_k$ are defined perpendicular to the surfaces i, j and k equal constants, and are pointing in the positive direction of relative axis i, j and k (pointing from surface i toward surface i+1, etc.). Finally, for every cell, bounding surfaces can be represented by a family of coordinate surfaces : $\vec{S} = \{ \vec{S}_i, \vec{S}_j, \vec{S}_k \}$.

Because equations (1.4) and (1.5) are valid for any volume Ω, they can be applied on any single cell composing the domain. In order to numerically solve these equations it is necessary to reduce the integrals to a discrete approximation. The numerical scheme is said cell centred because only a single point evaluation per cell is considered, for the independent variables $\vec{Q}$ (single point = centre of the cell). Therefore the equation (1.4) becomes :

$$\Omega_{ijk} \frac{\partial \vec{Q}_{ijk}}{\partial t} + \bar{\delta} \left[\vec{H} \cdot \vec{S} \right]_{ijk} = 0 \qquad (1.6)$$

The value $\vec{Q}_{ijk}$ is interpreted as a volumetric average located at the centre of the cell and $\vec{\delta}$ [$\vec{H}(Q) \cdot \vec{S}$ $]_{IJK}$ is the summation of fluxes through the surfaces S. Application of this operator requires the evaluation of $\vec{H}(Q)$ on the cell surfaces. This value of $\vec{H}(Q)$ is obtained using the average value of $\vec{Q}$ between neighbouring cells.

The finite volume approach has the great advantage that no global coordinate transformation is necessary. This type of scheme can accommodate any type of curvilinear coordinate system. Once the cartesian coordinates of the eight vertices of the cell are given, all numerical terms used for the resolution of the discrete integral equation can be determined strictly by geometry principles.

1.3.2 Artificial viscosity

Stability analysis of the numerical scheme reveals the dispersive nature of the discrete equation. This type of discrete equation, if solved as described in the previous section, may be unstable and produce spurious jumps in the flow parameters near a shock. To ensure numerical stability of the scheme, a combination of second and fourth order "artificial viscosity" terms are introduced in the calculation. This approach is inspired by Jameson (2,4).

The second order dissipation term acts as artificial viscosity and the fourth order terms is used to limit the effect of the second order term where the solution is locally smooth by itself. The fourth order dissipation term makes sure that the final solution can converge to very small residuals. The dissipation operator is noted $D(\vec{Q})$ and is given as :

$$D(\vec{Q}) = D_x(\vec{Q}) + D_y(\vec{Q}) + D_z(\vec{Q}) \tag{1.7}$$

where (for example)

$$D_x(\vec{Q}) = d_{i+\frac{1}{2},j,k} - d_{i-\frac{1}{2},j,k} \tag{1.8}$$

and

$$d_{i+\frac{1}{2},j,k} = \frac{\Omega_{i+\frac{1}{2},j,k}}{\Delta t_{i+\frac{1}{2},j,k}} \left[\varepsilon^{(2)}_{i+\frac{1}{2},j,k} \left(\vec{Q}_{i+1,j,k} - \vec{Q}_{i,j,k} \right) - \varepsilon^{(4)}_{i+\frac{1}{2},j,k} \left(\vec{Q}_{i+2,j,k} - 3\vec{Q}_{i+1,j,k} + 3\vec{Q}_{ijk} - \vec{Q}_{i-1,j,k} \right) \right] \tag{1.9}$$

In equation (1.9), Δt is the time step which would bring the local Courant No. to 1 (averaged between neighbouring cells). The volume is also averaged between cells. The second order coefficient is proportional to local pressure gradients; it is expressed as :

$$\varepsilon^{(2)}_{i+\frac{1}{2},j,k} = \max\left(V_{i+1,j,k} , V_{ijk} \right) \tag{1.10}$$

where

$$V_{ijk} = k^{(2)} \frac{\left| P_{i+1,j,k} - 2P_{ijk} + P_{i-1,j,k} \right|}{\left| P_{i+1,j,k} \right| + 2\left| P_{ijk} \right| + \left| P_{i-1,j,k} \right|} \tag{1.11}$$

The fourth order coefficient is :

$$\varepsilon^{(4)}_{i+\frac{1}{2},j,k} = \max\left(0, k'\left(k^{(4)} - \varepsilon^{(2)}_{i+\frac{1}{2},j,k} \right) \right) \tag{1.12}$$

In equations (1.10) to (1.12) three constants are introduced, their role is to have a certain control on the intensity of the dissipation introduced in the scheme. In this project the value of $K^{(2)}$ is fixed to 0.5, $K^{(4)}$ is defined as the maximum value of $\varepsilon^{(2)}$ for a constant value of j and k (varying i) and K′ is 0.5. To conclude this section let's just note that equation (1.6) has now become :

$$\Omega_{ijk} \frac{\partial \vec{Q}_{ijk}}{\partial t} + \bar{\delta}\left[\vec{H}(Q)\cdot\vec{S}\right]_{ijk} = D(\vec{Q}) \qquad (1.13)$$

1.3.3 Runge-Kutta time-stepping

To have a time accurate solution of the Euler equations it would be necessary to solve for a constant time step over every cells of the mesh. This global time step should be equal to the smallest local time step over the entire grid. If only the steady state solution is of interest, it is much more efficient to use a local time step evaluated for each cell. The time accuracy is lost, but the rate of convergence is greatly increased. The local time step (Δt) is the minimum value, in a given cell (i,j,k), of the a Δt_{LOCAL} evaluated on each surface of the cell.

Using $\beta = \gamma/(\gamma-1)$, the mathematical evaluation of Δt can be expressed as :

$$\Delta t \leqslant \min_{ijk}\left(\Delta t_{LOCAL}\right) \qquad (1.14)$$

where

$$\Delta t_{LOCAL} = \frac{\Omega_{ijk}}{\frac{1}{2\beta}\left(Q_I + Q_J + Q_K\right) + \left[\frac{1}{4\beta^2}\left(Q_I^2 + Q_J^2 + Q_K^2\right) + a^2\left(S_I^2 + S_J^2 + S_K^2\right)\right]^{\frac{1}{2}}} \tag{1.15}$$

and

$$Q_I = \left|\vec{V}\cdot\vec{S}_I\right| , \quad Q_J = \left|\vec{V}\cdot\vec{S}_J\right| , \quad Q_K = \left|\vec{V}\cdot\vec{S}_K\right| \tag{1.16}$$

$$\vec{V} = u\vec{i} + v\vec{j} + w\vec{k} \tag{1.17}$$

$$a^2 = \frac{1}{2\beta}\left(2h_o - u^2 - v^2 - w^2\right) \tag{1.18}$$

It is possible to improve even more the quality and the rate of convergence of the integration relative to time. This is done using a three-stage Runge-Kutta integration scheme for advancing the solution from time step n to n+1. Rewriting the complete spatial operator :

$$P(\vec{Q}) = \frac{1}{\Omega_{ijk}}\left\{\bar{\delta}\left[\vec{H}(\vec{Q})\cdot\vec{S}\right]_{ijk} - D(\vec{Q})\right\} \tag{1.19}$$

The intermediate terms of the integration are :

$$\begin{aligned} \bar{Q}^{(0)} &= \bar{Q}^{n} \\ \bar{Q}^{(1)} &= \bar{Q}^{(0)} - \Delta t \; P(\bar{Q}^{(0)}) \\ Q^{(2)} &= \bar{Q}^{(0)} - \Delta t/2 \; P(\bar{Q}^{(0)}) - \Delta t/2 \; P(\bar{Q}^{(1)}) \\ Q^{(3)} &= \bar{Q}^{(0)} - \Delta t/2 \; P(\bar{Q}^{(0)}) - \Delta t/2 \; P(\bar{Q}^{(2)}) = \bar{Q}^{n+1} \end{aligned} \tag{1.20}$$

In the present project, the three stages time integration scheme is used. This integration method is second order accurate in time. There are also other integration method that can be used, such as the five stages Runge-Kutta integration scheme that is widely used in Jameson type Euler solvers (2).

1.3.4 Boundary conditions

Only a brief presentation, with minimum details, on the implementation of the boundary conditions is given in this report. It does not mean that this topic is not important, since it is indisputable that an improper treatment of the boundary conditions can lead to instability or at least to erroneous results.

In all cases treated for this project, the flow at infinity is subsonic. According to characteristic theory for subsonic inflow and outflow through an outer boundary, three conditions may be specified for inflow and one for outflow. At the body surface the boundary conditions are applied by imposing the velocity tangent to the surface and by extrapolating the pressure from the cells next to the surface.

1.4 CONVERGENCE LIMITATIONS

The Euler equations permits very accurate predictions of flow behaviours, but to obtain this accuracy the equations need to be solved on a fine grid that will allow the capture of very small and local phenomena. The convergence rate for the Euler solver described in this chapter is of the order of the square of the mesh size. With the explicit time-stepping scheme the information is propagating only to the neighbouring cells at each step. These three conditions put together give a clear indication that convergence to steady state is going to be very slow. When it is considered that four equations are to be solved in every cell for each step, it is also obvious that the total computing time is going to be very large.

To overcome such limitations a multigrid scheme must be implemented in the program. This scheme will combine the convergence efficiency of a coarse mesh and the precision of a fine.

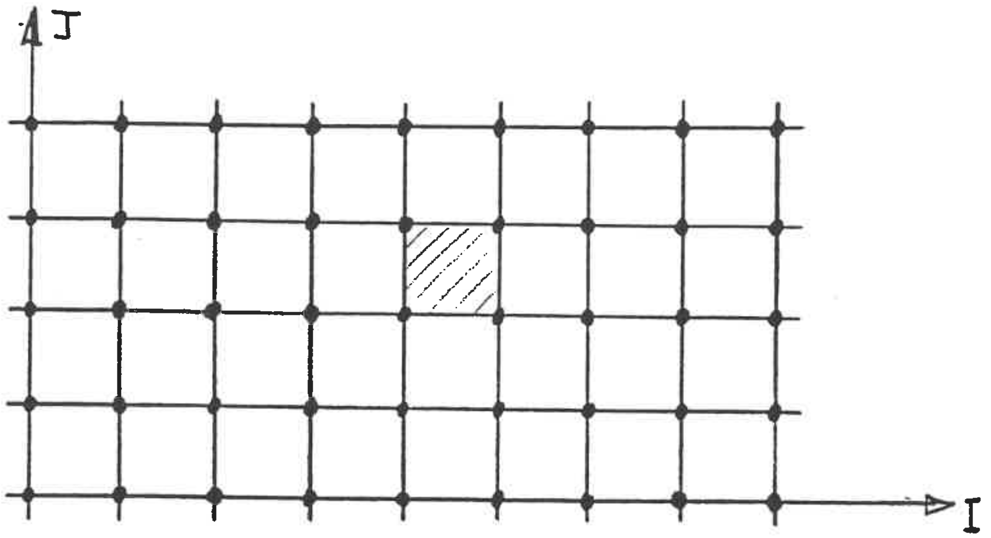
CHAPTER II

MULTIGRID SCHEMES

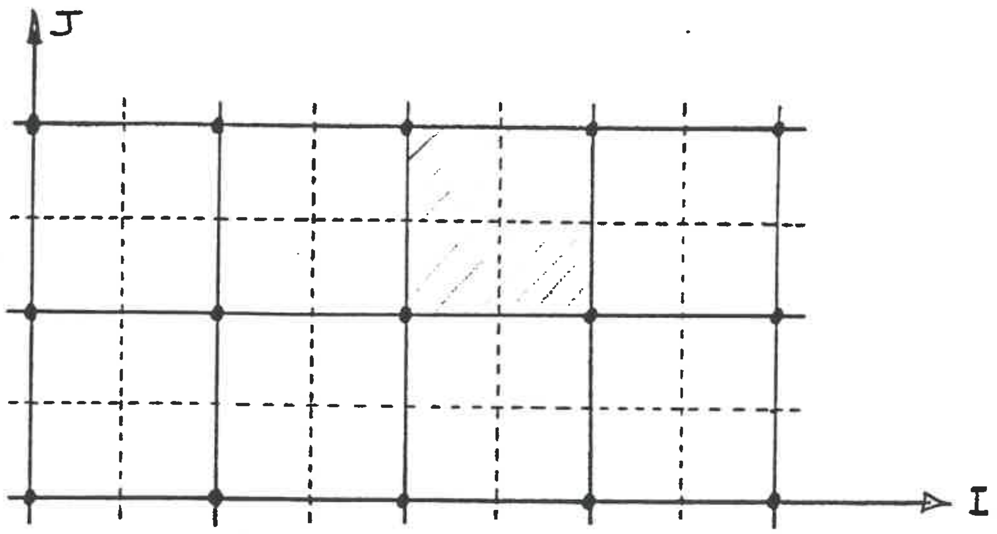
2.1 THEORETICAL JUSTIFICATION

To reduce computational time, one could intend to use a coarse mesh with which the number of unknowns is relatively small and the time step allowed (corresponding to Courant No.) is relatively large. As it was discussed at the end of chapter I, such practice may lead to inaccurate results due to large truncation errors. One way to overcome this problem is to use the "multiple grid" technique developed by Brandt (5), or the multigrid scheme, used by Jameson (2,3).

Using these techniques, the solution on the fine grid is obtained by cycling the numerical procedures between fine and coarse grid systems. The basic idea behind the multigrid technique is the use of coarser grids to propagate the fine grid corrections rapidly and correctly throughout the calculation domain. Doing so improves the convergence rate to steady state while maintaining low truncation errors. To implement such technique, coarser meshes have to be generated. In the present project, a coarse grid is constructed by removing every second line in the fine grid system as shown in fig. 2.1 (for a 2-D grid). Noting h, the characteristic length on a fine mesh, the subsequent coarser grids will be referred as grid 2h, grid 4h, etc.



(a) Fine mesh (grid h)



(b) Coarse mesh (grid 2h)

Figure 2.1 : Coarse grid construction in 2-D problems

2.2 COMPLETE MULTIGRID TECHNIQUE

The sequence of operations, that is described in this section, has been presented by Jameson (2,3). This procedure advances the solution on the coarse grids, while using the large time steps associated to these grids, then it interpolates the corrections back on the finest grid. To control the truncation errors, the parameter corrections evaluated on the finest grid are transferred on the coarse grids using a weighted average. With this method, the solution progress rapidly in time, and the information is propagated a much larger distance than the width of the fine grid cells.

2.2.1 Step by step procedure

- Normal update of the flow on the finest grid (grid h).

- Evaluation of the residuals (P(Q)) on the fine grid without updating the flow variables.

- Switch to medium grid level (grid 2h). Switching includes modification of the numerical calculation parameters, change of the current volume and surface arrays and evaluation of the new local time steps.

- Prior to solving on the grid 2h, the residuals from the grid h are accumulated and averaged on the 2h cells. The accumulated and averaged value for the residuals will be used as a reference when updating flow variables during coarser grid resolutions (reduction of truncation errors).

- The equations are now solved, using the standard time-stepping scheme, on the grid 2h. Every residuals evaluated on the grid 2h are corrected using the averaged values from grid h. Flow variables are updated.

- Final values of corrected residuals are kept for accumulation on a coarser mesh (grid 4h).

- Switch from grid 2h to grid 4h following the rules established for the h to 2h switch.

- Residuals from grid 2h are accumulated and averaged on the grid 4h.

- The equation are solved on the coarse mesh (coarsest mesh for the present demonstration) ensuing the principles stated for the resolution on the grid 2h.

- Switch from grid 4h to grid 2h.

- Interpolation of the 4h parameters on the 2h cells.

- Switch from grid 2h to grid h.

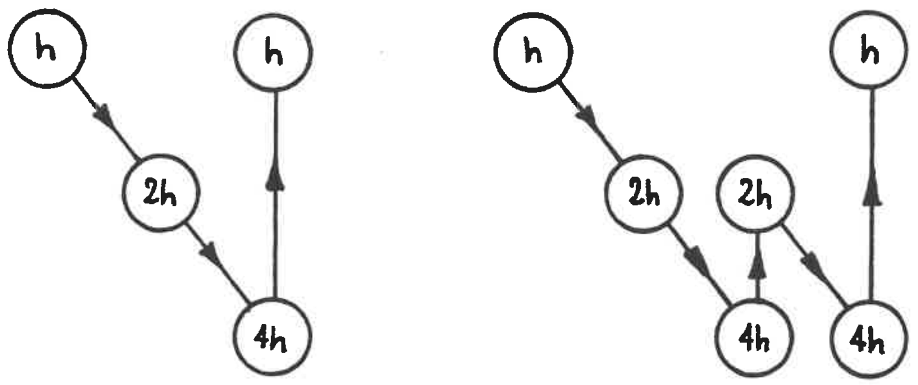
- Interpolation of the 2h parameters on the h cells.

- Normal update of the flow on the finest mesh.

- Etc.

2.2.2 Particularities

The procedure presented in section 2.2.1 is named V cycle, since the successive steps can be schematically represented in a V shape as shown on figure 2.2 (a). Another variant of multigrid procedure is the W cycle (fig. 2.2 (b)).



(a) V cycle (b) W cycle

Figure 2.2 Multigrid procedure variants

Typical procedures, presented in figure 2.2, are constructed for two coarse grid levels, but in fact the limit of coarse grid levels comes from the size of the finest mesh. It is not possible to have less than two cells in any direction on the coarsest mesh. Respecting this, if the finest mesh is very fine, there is no reason why four or five coarse grids could not be used. In general, the most efficient way to use multigrid schemes is to take advantage of the maximum number of coarse grid levels. From the step by step procedure presented in section 2.2.1, it may seem relatively easy to implement a multigrid scheme in an Euler solver. It is important to realize that most steps require a lot of computational operations, they also necessitate the creation and continual revision of numerous control variables. Considering the number of quite complex operations that have to be performed at each cycle, the benefit from the multigrid implementation must be important. In fact the gain is great; convergence toward a given residual level, may be reached five to ten times faster. Figure 2.3 illustrates the convergence history for a 2-D transonic flow in a channel, first solved on the fine grid only, and then solved using a multigrid scheme with three coarse grid levels.

2.3 SIMPLIFIED MULTIGRID APPROACH

To reduce the complexity of the multigrid procedure and still have some gain in the convergence rate, S. Saha (1) used a simpler approach than the one presented in section 2.2 for his 2-D Euler solver. In this project, to avoid unnecessary complications, Saha's simplified method is extended for 3-D, and implemented in the 3-D solver. This simpler procedure updates the flow variables using a combination of the actual fine grid correction and an averaged correction. The averaged correction is obtained by accumulating fine grid corrections

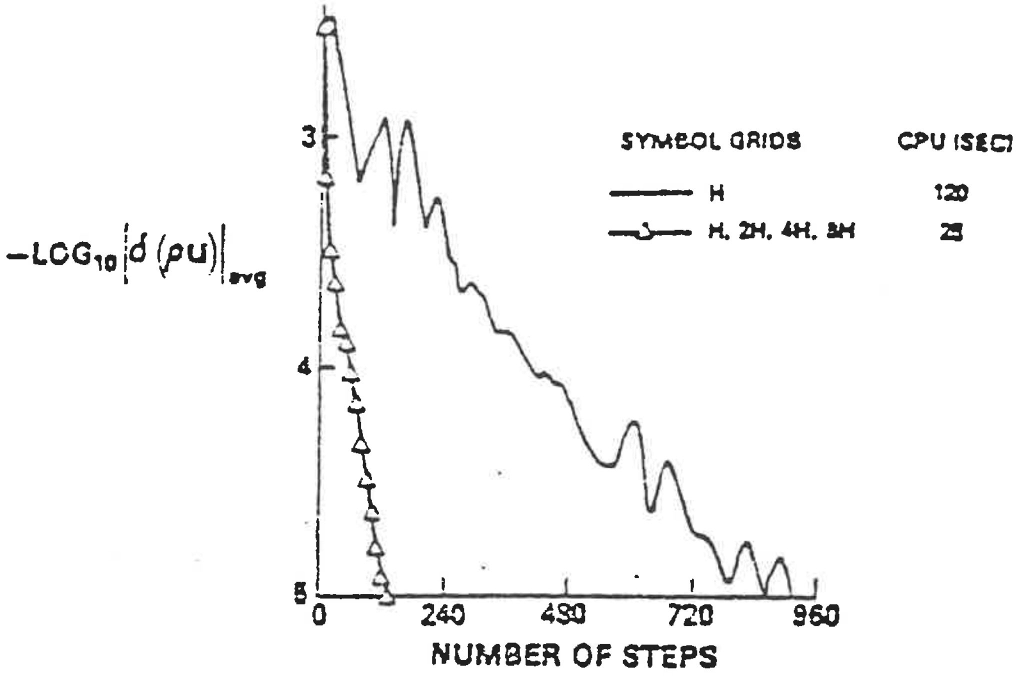


Figure 2.3 Convergence history with & without multigrid

on a coarser mesh, by operating a weighted averaging and by interpolating accumulated values back onto the fine grid. For this formulation, larger time steps cannot be used, but during every cycle the information is propagated much farther than the finest grid neighbouring cells. For 2-D solutions, this scheme augmented convergence rate by two, and also permitted to reduce final residuals (10^{-6} compared to 10^{-3}).

The idea behind the simple multigrid scheme is to correct the fine grid spatial operator $P^H(Q)$ using an averaged coarse grid operator $P^{2H}(Q)$. If it is supposed here that only one coarse grid is used (fine : grid h & coarse : grid 2h), the corrected value of $P^H(Q)$ can be expressed as :

$$P^H(\bar{Q}) = \omega P^H(\bar{Q}) + (1-\omega)\, I^H_{2H}\, P^{2H}(\bar{Q}) \qquad (2.1)$$

In this expression, w is an weighting factor. For 2-D calculations, the factor w was set to 0.8. Tests are done to verify that value 0.85 is more suited to 3-D cases. The term I^H_{2H} is an interpolation operator to transfer the coarse grid residual on the fine grid. In section 2.3.1, various interpolation procedures are discussed, and it is verified if each of these interpolation processes is conservative in respect to the flow parameters.

Coarse grid residual operators are constructed by accumulating the fluxes calculated on the fine grid cells composing a coarser grid cell and by averaging them proportionally to their volume relative to the volume of the coarse cell. The expression for $P^{2H}(Q)$ is then :

$$P^{2H}_{ijk} = \frac{[\Omega^H P^H]_{ijk} + [\Omega^H P^H]_{i+1,j,k} + [\Omega^H P^H]_{i,j+1,k} + \dots + [\Omega^H P^H]_{i+1,j+1,k+1}}{\Omega^{2H}_{ijk}} \qquad (2.2)$$

In the same manner it is possible to build even coarser grid residual operators as it is demonstrated here. For the operator $P^{4H}(Q)$ the expression is :

$$P^{4H}_{ijk} = \frac{[\Omega^{2H} P^{2H}]_{ijk} + [\Omega^{2H} P^{2H}]_{i+1,j,k} + [\Omega^{2H} P^{2H}]_{i,j+1,k} + \dots + [\Omega^{2H} P^{2H}]_{i+1,j+1,k+1}}{\Omega^{4H}_{ijk}} \qquad (2.3)$$

The number of coarse grid levels is again restricted by the nature of the finest grid. The multigrid algorithm is first building accumulation residual operators on successively coarser grid until it reaches the coarsest mesh. The second step is to interpolate on successively finer grids using the principle characterized in equation (2.1) toward the finest grid. This technique enhances stability in a way similar to implicit residual averaging schemes.

The outermost cells on the finest grid, where inflow and outflow boundary conditions are applied, do not participate in the flux calculations. Hence, the boundary conditions do not need any special treatment during the application of the multigrid scheme.

2.3.1 Conservativeness of interpolation

The reliability of the simplified multigrid approach depends greatly on the quality of the interpolation scheme used. To be confident about the choice of such an interpolation procedure, three different schemes were formulated and compared. The comparison was performed "manually" on a simple 2-D grid, with one multigrid level, to evaluate the conservativeness of the schemes for one iteration. Further testing was made through comparison of results for about one hundred iterations in the 3-D Euler program. During these latter tests some variants of the schemes were tested and a final choice was made, based on simplicity of the programing and quality of the results.

The first interpolation studied, referred as New1, starts by imposing the coarse grid operator value on the fine grid nodes corresponding to the centre of the coarse cells. These nodes refer to fine grid points having even values of relative coordinates : for example coarse cell (1,1,1) has its centre node located in the fine grid system at coordinate (2,2,2). The values on the other fine grid points are obtained by averaging the known neighbours successively in the three relative directions. As an example rho(5,2,2) = 0.5 x (rho(4,2,2) + rho(6,2,2)). Once parameters are evaluated at each fine grid points, the value in a fine cell is approximated as the average of values on its eight corners. The next step consists of a global "smoothing" done by averaging the values on each fine cells. A weighted blending of the values from the actual cell and all surrounding cells (28 cells in total) is made. The final step is the correction of the fine grid spatial operator using equation (2.1).

The second approach, New2, is very similar, except that coarse grid values are not imposed in the centre of the coarse cells, instead they are averaged on the corners of these coarse cells. This practically means that values are now determined on the nodes having odd relative coordinates in the fine mesh (ex node(3,7,5)). Similarly to what was performed in New1, parameters are evaluated for every fine grid points, averaged for every fine grid cells, averaged again through the global "smoothing" and finally the fine grid spatial operators are corrected.

A variant for these first two schemes will be to operate a mixing between the value on the fine cells (averaged from eight corners) and the value of coarse grid operator relative to this fine cell. A 50-50% mixing ratio is selected. This mixing is again followed by the smoothing and the correction of the spatial operator on the fine grid. This variant should render the schemes more conservative.

The final scheme studied, New3, is a direct 3-D extension of the scheme programmed in 2-D by S. Saha (1). For this approach the values on the fine cells are directly obtained from a combination of the coarse grid operator and the gradients of the coarse grid operators in the three relative directions. Details will be presented in the program description section, Chapter III. Once again, when values are approximated on the fine cells, a smoothing is performed, followed by the correction of the fine grid spatial operator.

Many figures are generated for the comparison of the three schemes and their variants. The most important ones are grouped in Appendix A. Figure A.1 shows the initial values of a parameter on every cells of a simple 2-D grid. Figure A.2 displays accumulated values of the parameters obtained using equation (2.2) on the cells of the coarse grid (grid 2h). Volume of each fine and coarse cells is indicated between parenthesis. Figures A.3 to A.7 give the resulting values of the parameter after one interpolation process. Table A.1 indicate how the parameter is "theoretically" conserved during the operations. To compare conservation, a summation over the entire grid is operated for the value of the parameter multiplied by the volume on each cell. The sum for each scheme is then compared with the sum obtained with the initial values.

The last set of figures try to illustrate how the schemes really compare to each other in the real program, and also how the 50-50 mixing and the value of the weighting factor w influence the convergence rate of the solution. Figures A.8 and A.9 show how the mixing influence the convergence for the scheme New1 and New2. Figures A.10 and A.11 illustrate the gain in convergence made by using w=0.85 instead of w=0.8 for 3-D cases. Figures A.12 and A.13 show the final comparison performed between the three schemes for w=0.85 and using 50-50 mixing for New1 and New2.

After a close examination of figures and table presented in Appendix A, it is decided that the interpolation scheme New1 with mixing is to be used in the program. Even though the 2-D theoretical study show that New2 with mixing is the best approach, it is quite obvious, from figure A.12 and A.13, that a difference in not really noticeable. Therefore New1 offers the advantage of being the simplest method to program and should minimize the additional computing time necessary to solve a 3-D flow using multigrid scheme.

CHAPTER III

MULTIGRID PROGRAMMING

3.1 ORIGINAL PROGRAM

In order to implement multigrid in a 3-D Euler solver, it is fundamental to understand the basic structure of the original program (E3OHS FORTRAN) that is provided at the beginning of the project. For the current project, the original program is provided by S. Saha (1), from his doctoral research work.

The program can be subdivided into three main portions, first, the settings, second, the time stepping (resolution) and third, the outputs. The purpose of this report is not to describe thoroughly the original program, but just give enough insights so that it is possible to evaluate what modifications should be performed to it, and where should the multigrid scheme be introduced. In the coming sections each portions of the program will be explained, and the general computation sequences will be presented (without mathematical details).

3.1.1 Settings

Before any of the calculations are done, the program must establish the conditions in which the solution is to be obtained. The primary informations (flow conditions) are read from the input file, and the geometry around which the flow parameters are to be solved is read from a grid file. The settings portion of the program is in charge of reading all this information and processing it. The most important result from this treatment is the preparation and storage of the initial solution.

The grid file is simply an organized sequence of spatial coordinates corresponding to every nodes of the 3-D mesh. On each line of the grid file the coordinates of a node, of relative position (i,j,k), are given using the three real numbers x(i,j,k), y(i,j,k), z(i,j,k).

The information that must be provided by the user to the program via the input file (INPUT EULER) are :

ALPHA	: the angle of attack (in degrees)
MACH	: the Mach number of the incoming flow
NI, NJ, NK	: number of cells in each relative direction
KTIP	: position of the tip of the wing (K direction)
C1	: Currant number (max value = 2.0)
C2, C4	: artificial dissipation constants
INFLOW	: type of inflow boundary conditions
J0, KHIGH	: constants for the external B.C.
NOTS	: number of time steps per cycle
NCYCLE	: number of cycles

The sequence, in which settings operations are made in the program, is the following (see next page) :

1- Reading of the grid file.
2- Computation of the curvature of all cells on the surface of the wing.
3- Reading of the input file.
4- Prescription of freestream quantities.
5- Evaluation of normals on all cells' surfaces.
6- Calculation of all cells' volumes.
7- Preparation of initial solution.

3.1.2 Time-stepping

This second portion of the program is the heart of the Euler solver in 3-D. Practically the time stepping is enclosed in a big DO loop, repeated until the maximum number of iterations is reached (counters : NOTS and NCYCLE). Inside every pass through the loop, the spatial operator is solved three times in order to accomplish the three steps Runge-Kutta integration described in Chapter I, equation (1.20). The progression of operations inside the loop is the following :

1- Copy of the parameters in a secondary variable ($Q^{(0)} = Q^n$).
2- Evaluation of local time step for each cell.
3- Evaluation of spatial operator.
4- First Integration step ($Q^{(1)}$).
5- Evaluation of spatial operator.
6- Second Integration step ($Q^{(2)}$).
7- Evaluation of spatial operator.
8- Third and final Integration step ($Q^{(3)} = Q^{n+1}$).
9- Convergence parameters evaluation.

3.1.3 Outputs

This portion is the simplest, since it is very straight forward. There is a total of four output files :

HISTORY EULER:	This first file is constructed during the computation. At the end of each time step convergence parameters are evaluated and written in this file. At each step maximum error, averaged error, C_L, C_D and (I,J,K) coord. of max. error cell are provided.
CPLOT EULER:	This file is written after all iterations are completed. It provides 2-D C_P distributions on sections of the wings corresponding to interfaces between cells.
OUTPUT EULER:	This file is written after all iterations are completed. It provides 2-D C_P distributions on sections of the wings corresponding to centres of the cells.
RESULT EULER:	This file is also written at the end of the time stepping process. In this file the values of all parameters for every cell are stored.

It is to be noted that the latter file was removed (commented out) from the output process, in many cases, because it uses an enormous quantity of memory and it necessitates the use of an elaborated visualization program to extract any valuable information.

3.2 NEW PROGRAM

3.2.1 Reorganization

Prior to writing any new line of code it is essential to operate some minor modifications to the original program. To help differentiate both program the name is changed to E3NEW FORTRAN and all input/output files now have the suffix MULTI instead of EULER. The modifications made, other than adding comments, come down to writing some operations, previously performed inside the main program, into individual subroutines. The purpose of such modifications is to enable the multigrid process to use some already written routines, such as volume calculations. It is also needed to add new variables and arrays to the prior commons. Finally two new variables are added to the input file :

MGL	: multigrid level (total number of grids)
OMEG	: weighting factor used for spatial operator correction

New subroutines must now be created to implement the multigrid procedure inside this program. A total of two new subroutines are generated. The first one is inserted at the end of the settings portion of the program and is needed to generate the coarse grids used during the multigrid process. The second is called after each spatial operator evaluation in the time stepping loop, and is executing the procedure described in section 2.3 of this report.

3.2.2 Subroutine COARSE

This new subroutine, for which the listing is provided in appendix B, is called just before the time stepping loop engages, if multigrid is to be used (MGL > 1). For every new coarse grid that has to be constructed and stored a typical cycle is executed inside a loop. The loop is starting at coarse grid level 1 (grid 2h) and stopping at coarse grid level MGL-1. Per example it means that if MGL = 3 (three grid levels), two coarse grid levels exist. The grid construction cycle is :

1- Set the coarse grid level.
2- Define coarse grid points by removing every second points from the finer grid and store them in secondary arrays.
3- Divide NI, NJ,and NK by two.
4- Evaluate the volume of the coarse cells and store values in a permanent array (in common).

Once all the new coarse grids volumes are determined, NI, NJ, and NK must be put back to their original values. Some programing details about manipulation of the arrays are not detailed here, but can be examined in the listing of the subroutine provided in appendix B.

3.2.3 Subroutine MULTI

This subroutine, for which listing is provided in appendix C, is called from inside the time stepping loop after spatial operator evaluation, if multigrid is to be used (MGL greater than 1). The entire multigrid procedure, described in section 2.3, is enclosed in this subroutine. A "V cycle" is implemented in this program, therefore the subroutine is separated in two parts. First part, residuals are accumulated on successive coarse grids (loop 1). Second part consists of repeatedly applying the interpolation model toward the finest grid (loop 2).

The steps performed inside each of the two loops are now presented, starting with loop 1 :

1- Update of coarse grid level counter.
2- Separate residual collections for grid 2h and for coarser grids (4h, 8h ...). (equations (2.2) & (2.3))
3- NI, NJ,and NK are divided by two.

When the coarsest grid level is attained (level MGL-1), the program switches to loop 2 :

1- Update of coarse grid counter (reversed progression).
2- Coarse grid operator is copied in the centre of the finer cells of the next finer mesh.
3- Parameters are averaged on other nodes of the finer mesh.
4- Operator on a finer cell is set equal to the average of its value on the eight corners of the cell.
5- 50-50 mixing with the coarse grid residual.
6- Global smoothing (averaging of interpolated residuals).
7- Correction on coarse grids (coarse grid level > 1).

8- ** The last pass in loop 2 (coarse grid level = 1) is completed by a correction on the finest grid (step 7 is bypassed).

To conclude this third chapter, it is to be noted that the implementation of the two new subroutines do not affect the way in which the program is to be run. A short user's guide is provided in appendix D for the correct procedure to run the program on the Ecole Polytechnique CMS system.

CHAPTER IV

RESULTS AND DISCUSSION

4.1 TYPICAL RESULTS

Now that implementation of the multigrid scheme is well understood, it is of a primary importance to measure its effect on the results of the 3-D Euler solver. The most important result is observed on a convergence graph. This type of graph shows the log of the average and/or maximum error, for a given case run with and without multigrid, in respect to the iteration number. The error, in the sense of this report, is the variation of the momentum in the x direction (principal direction of the flow) occurring during a single iteration. This is more or less an arbitrary choice based on current practice in scientific publications where convergence rates are discussed.

The other results presented, are 2-D C_p plots, taken at various wing stations. The "multigrid" C_p distributions are compared with original program solutions and with experimental results. Since C_p plots are secondary results, only one case will be presented here. This case is the standard Onera M6 wing at alpha = 3.06 deg. and Mach number 0.84. This case is very well documented, and the experimental results may be directly extracted from S. Saha's thesis (1). In fact, C_p curves are secondary results because they are only examined in order to be confident that the multigrid process do not reduce the quality of the solver predictions.

Before looking at the graphs, one must establish the conditions in which the tests were performed. Of coarse numerical test are involved here, then the conditions refer to the numerical parameters transmitted to the program via the input file and also refer to the actual size of the grid used during the computation. As for the grid, it is O-H type and its size is 49x17x17 points or 48x16x16 cells. As it was mentioned previously, the angle of attack is 3.06 deg. and the Mach number is 0.84. The Currant number and the two artificial viscosity constants are inspired from the values proposed by S. Saha (1). Their values are : Currant = 1.8, C2 = 2.0 and C4 = 1.0. A total of 400 iterations are performed and the error is reduced to about 2.5 orders of magnitude during the process. This was not the only case conducted, but the results obtained for these particular conditions are very representative of the overall numerical work that has been performed through this project.

A total of six graphs are presented in the coming pages. Figure 4.1 and 4.2 show the evolution of the error as the iterations progress. Then Figure 4.3 to 4.6 show C_P plots for four different sections of the wing. Following these graphs an analysis of the results will be performed and some additional remarks on the factors influencing the global quality of the results will be made.

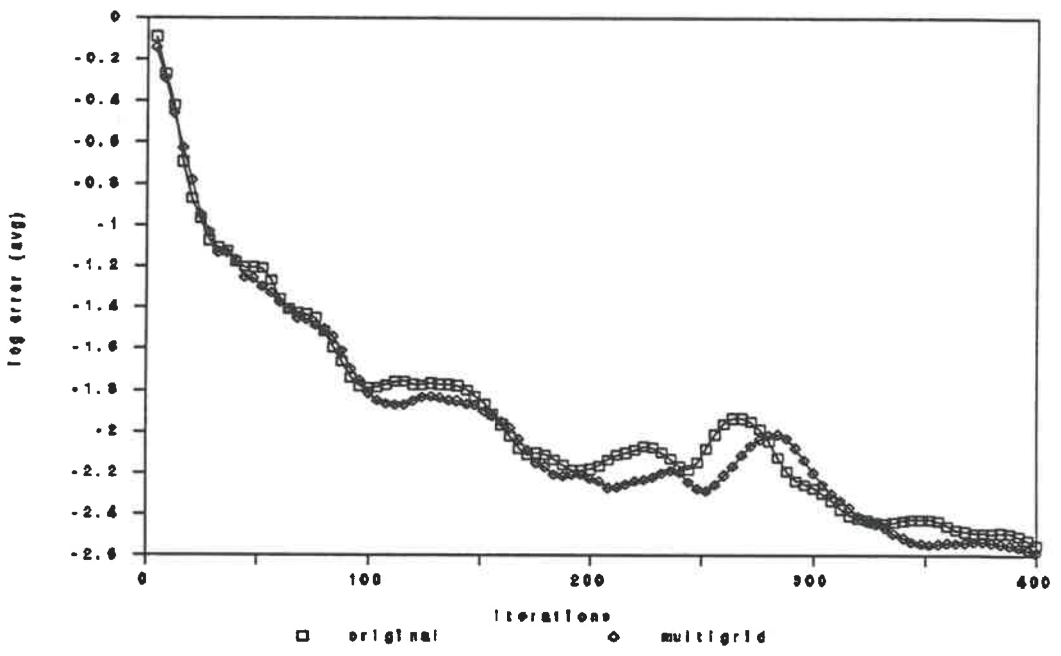


Figure 4.1 Convergence history (average error)

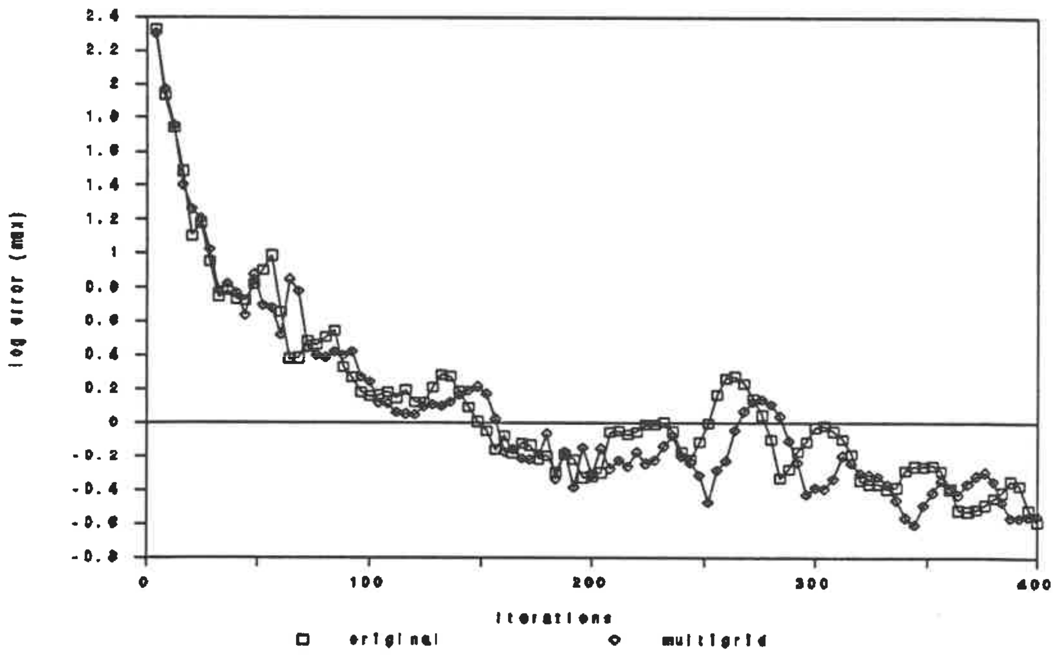


Figure 4.2 Convergence history (maximum error)

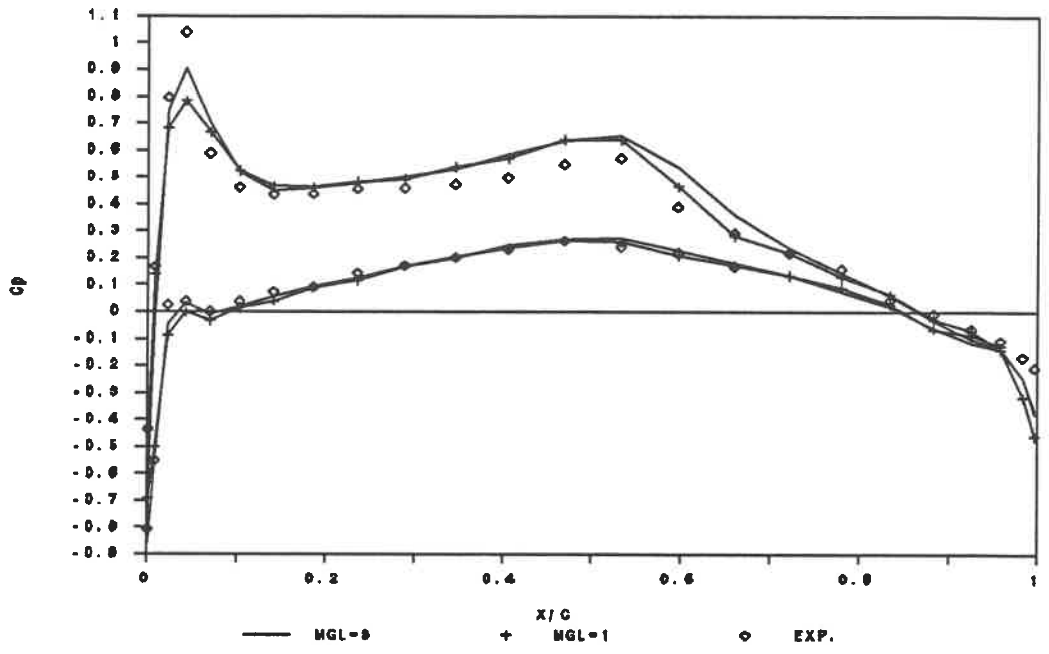


Figure 4.3 C_p distribution at 20% half span

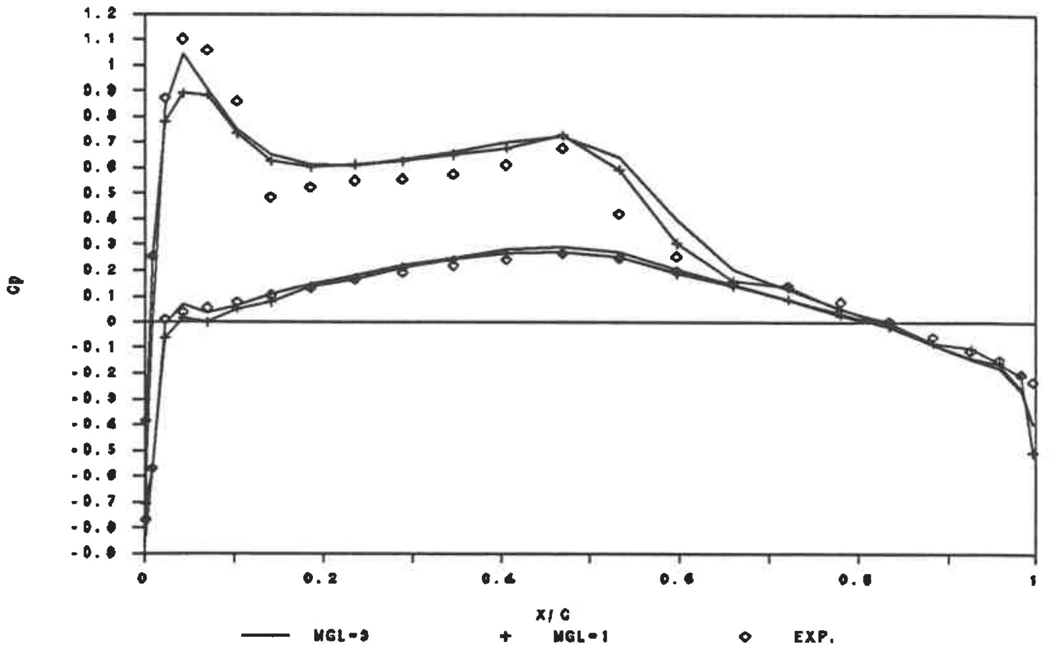


Figure 4.4 C_p distribution at 44% half span

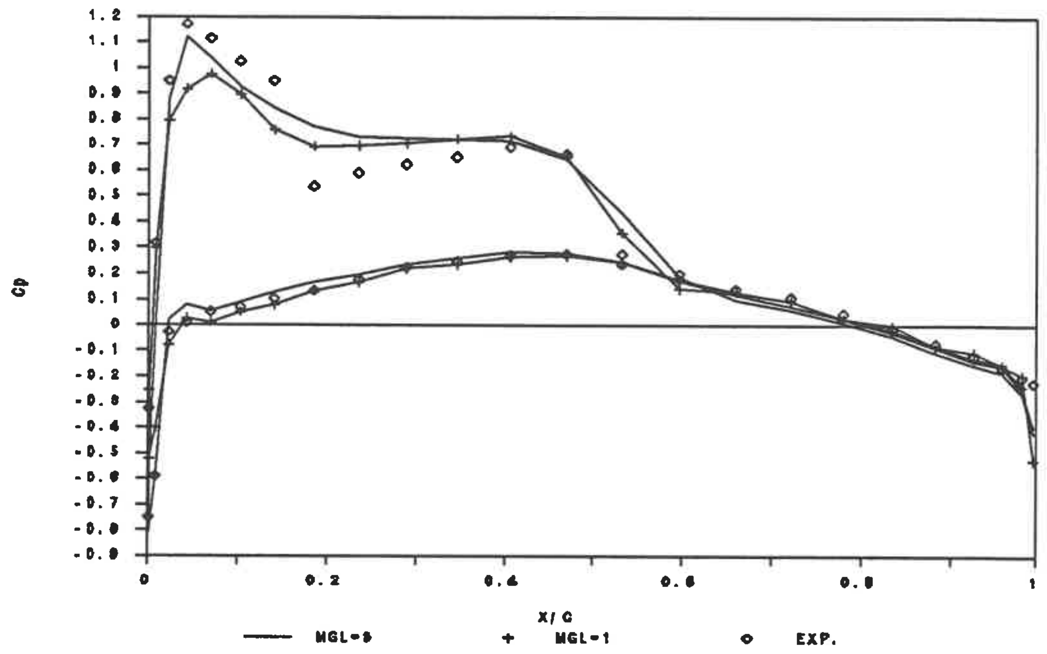


Figure 4.5 C_p distribution at 65% half span

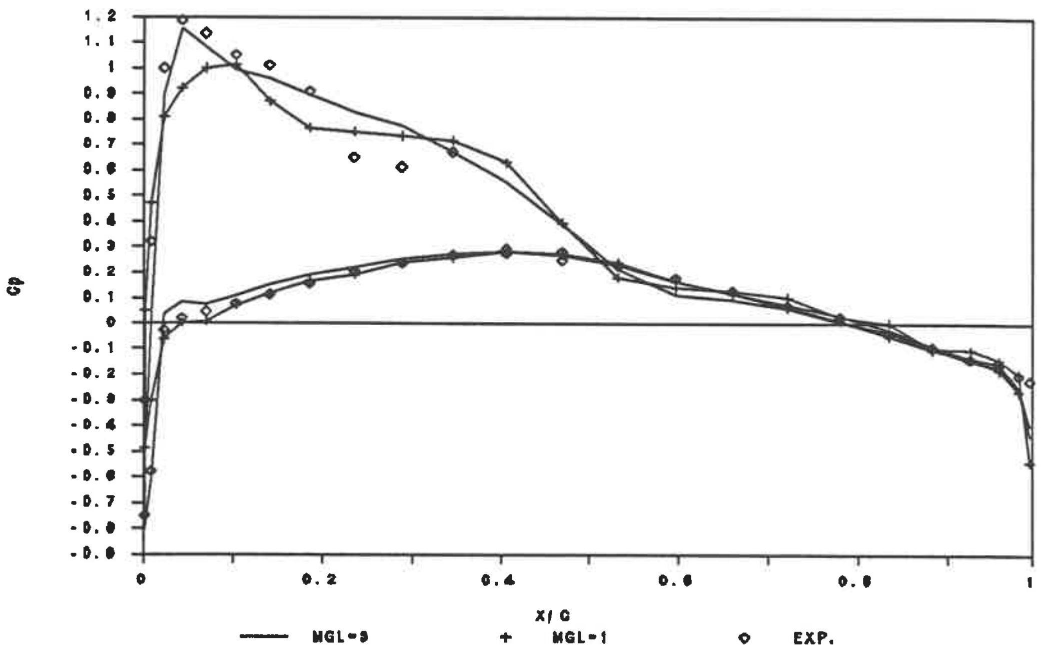


Figure 4.6 C_p distribution at 80% half span

4.2 DISCUSSION

The convergence rates presented in fig. 4.1 and 4.2 show a little improvement of convergence with multigrid when compared to original results obtained without multigrid. This is especially visible on the averaged error curve. It is also observed that the lapse between the curves is increasing with the iterations. This last statement maybe questionable because all along the convergence process it may be observed that the curves are crossing each other at various points. The justification for this assertion relies on the fact that further the convergence progresses larger are the period for which the multigrid has a lower error. Actual lapses between the curves during these periods seems also larger.

The pressure coefficient curves are given for four sections taken from the centre of the wing to the tip. The relative position of these sections (in half-span %) are : 20%, 44%, 65% and 80%. All experimental results were originally presented by Schmitt and Charpin (6), and results without multigrid (noted MGL=1) are extracted from Saha's thesis (1). It is observed that results with and without multigrid are behaving similarly, even if discrepancies are obvious as we move toward the tip of the wing. It is acceptable to assess that both solutions are of equivalent quality. For all sections the difference between solutions is associated to the prediction of the shock and of the pressure peak on the upper surface of the wing. The prediction using multigrid provides a much better representation of the pressure peak but tends to smoothen all shocks. This comparison is sufficient to conclude that multigrid process do not affect the quality of the results, in fact it even improves them a little.

The noticed differences between the two approaches might be explained either by a different choice of the second order dissipation constant or simply by a different total number of iterations.

If figure 4.1 is compared to figure 2.3 (convergence rates for common multigrid scheme) it is obvious that the multigrid scheme studied during this project is far less efficient than the "standard" (and complex) multigrid approach used by most programs. The present method could possibly provide better results if a finer grid and an higher multigrid level were used. It also seems that this method could be of some use if a very high number of iterations were to be performed.

CONCLUSION

The use of a simplified approach for 3-D calculations is not as interesting as it was for the resolution of 2-D cases. The main difference is the fact that it is inexpensive to run a large number of iterations in 2-D compared to 3-D. The cost of a single iteration on the 48x16x16 3-D grid is around ten computer dollars. Therefore, it is not realistic to run on a regular basis, at least not on Ecole Polytechnique's main frame, cases for much finer grids or for over a thousand iterations, it would cost a fortune.

This project demonstrates that the simplified multigrid method proposed (in 2-D) by Saha (1), can be applied in 3-D and can provide some improvement in the convergence rate. Results also demonstrates that the improvements generated are not sufficient to justify an extensive use of this Three dimensional model. It would be more appropriate, for 3-D calculations, to implement another multigrid scheme which would be more complex, but a lot more efficient. Some other convergence accelerators, such as implicit smoothing, could also be of some interest and could be studied as for an eventual application in the Saha's 3-D Euler solver.

The lack of a 3-D flow visualization program on the CMS system, or on any other available system in the Mechanical Engineering Department, at the time of development of this project, cause some unnecessary difficulties. It can only be suggested here, that this type of flow visualization program should be acquired by the Department in order to provide a powerful tool for future work on 3-D flow solvers.

REFERENCES

1. Saha S., "Computation of Transonic Flow About Aerofoils and Wings", PhD Thesis, Dept. of Aerospace Engineering, Indian Institute of Technology, Kharagpur, 1989.

2. Jameson A., "Solution of the Euler Equations for Two Dimensional Transonic Flow by a Multigrid Method", MAE Report No. 1613, 1983; Applied Math and Computation, No. 13, 1983.

3. Jameson A., Schmidt W., Turkel E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes", AIAA paper 81-1259, June 1981.

4. Jameson A., "Transonic Aerofoil Calculations Using the Euler Equations", Numerical Methods in Aeronautical Fluid Dynamics, Edited by P.L. Roe, Academic Press, 1983.

5. Brandt A., "Multi-Level Adaptive Computations in Fluid Dynamics", AIAA paper, Williamsburg, Virginia, 1979.

6. Schmitt V., Charpin F., "Pressure Distributions on the ONERA-M6 Wing at Transonic Mach Number", Experimental Data Base for the Computer Program Assessment, AGARD AR-138, May 1979.

APPENDIX A

Comparison of interpolation models

1.5 (15.40)	2.5 (13.30)	2.0 (11.20)	1.5 (9.10)	1.0 (11.20)	0.8 (13.30)
4.0 (12.76)	5.5 (11.02)	4.0 (9.28)	3.5 (7.54)	3.0 (9.28)	2.5 (11.02)
6.0 (10.56)	7.5 (9.12)	6.0 (7.68)	5.5 (6.24)	5.0 (7.68)	4.5 (9.12)
7.0 (8.80)	8.5 (7.60)	7.5 (6.40)	7.0 (5.20)	7.0 (6.40)	6.5 (7.60)
7.5 (7.48)	9.5 (6.46)	8.5 (5.44)	7.5 (4.42)	8.0 (5.44)	7.5 (6.46)
7.5 (6.60)	10.0 (5.70)	9.0 (4.80)	8.0 (3.90)	8.5 (4.80)	7.0 (5.70)

Figure A.1 Fine grid with initial value of parameter

3.201 (52.48)	2.682 (37.12)	1.724 (44.80)
7.150 (36.08)	6.458 (25.52)	5.638 (30.80)
8.680 (26.24)	8.286 (18.56)	7.827 (22.40)

Figure A.2 Coarse grid accumulated and averaged values

1.755	2.586	2.122	1.641	1.144	0.939
4.028	5.299	3.984	3.503	3.018	2.531
6.024	7.257	5.934	5.454	4.973	4.254
7.080	8.314	7.420	6.945	6.896	6.453
7.820	9.291	8.408	7.517	7.903	7.487
7.877	9.787	8.935	8.026	8.416	7.124

Figure A.3 Resulting values scheme New1 without mixing

1.755	2.595	2.112	1.659	1.126	0.939
3.954	5.259	3.939	3.470	2.986	2.457
6.098	7.302	5.984	5.496	4.998	4.597
7.051	8.313	7.407	6.943	6.877	6.412
7.848	9.312	8.420	7.539	7.917	7.508
7.877	9.795	9.339	8.034	8.408	7.124

Figure A.4 Resulting values scheme New1 with mixing

1.819	2.716	2.186	1.705	1.235	1.030
4.091	5.353	4.029	3.523	3.085	2.621
5.968	7.221	5.907	5.429	4.968	4.502
7.024	8.264	7.379	6.910	6.878	6.432
7.681	9.249	8.370	7.481	8.878	7.442
7.840	9.749	9.298	7.989	8.388	7.094

Figure A.5 Resulting values scheme New2 without mixing

1.787	2.627	2.144	1.691	1.172	0.984
3.956	5.283	3.963	3.497	3.001	2.502
6.070	7.284	5.952	5.488	4.996	4.588
7.023	8.286	7.390	6.925	6.867	6.401
7.829	9.291	8.400	7.521	7.904	7.496
7.859	9.775	9.306	8.016	8.394	7.109

Figure A.6 Resulting values scheme New2 with mixing

1.678	2.472	2.026	1.544	1.032	0.605
4.014	5.259	3.951	3.469	2.989	2.460
6.096	7.288	5.962	5.480	4.991	4.537
7.156	8.352	7.454	6.975	6.915	6.447
7.600	9.279	8.410	7.514	7.909	7.416
7.910	9.879	9.451	9.120	8.532	7.201

Figure A.7 Resulting values from scheme New3

Table A.1 Parameter conservation over one multigrid cycle

	Sum	Abs. diff.	Rel. diff.
Initial values	1498.1	0.0	0.0
New1 no mix.	1494.5	-3.6	0.24%
New1 with mix.	1495.7	-2.4	0.16%
New2 no mix.	1500.8	+2.7	0.18%
New2 with mix.	1487.5	-0.6	0.04%
New3	1492.2	-5.9	0.39%

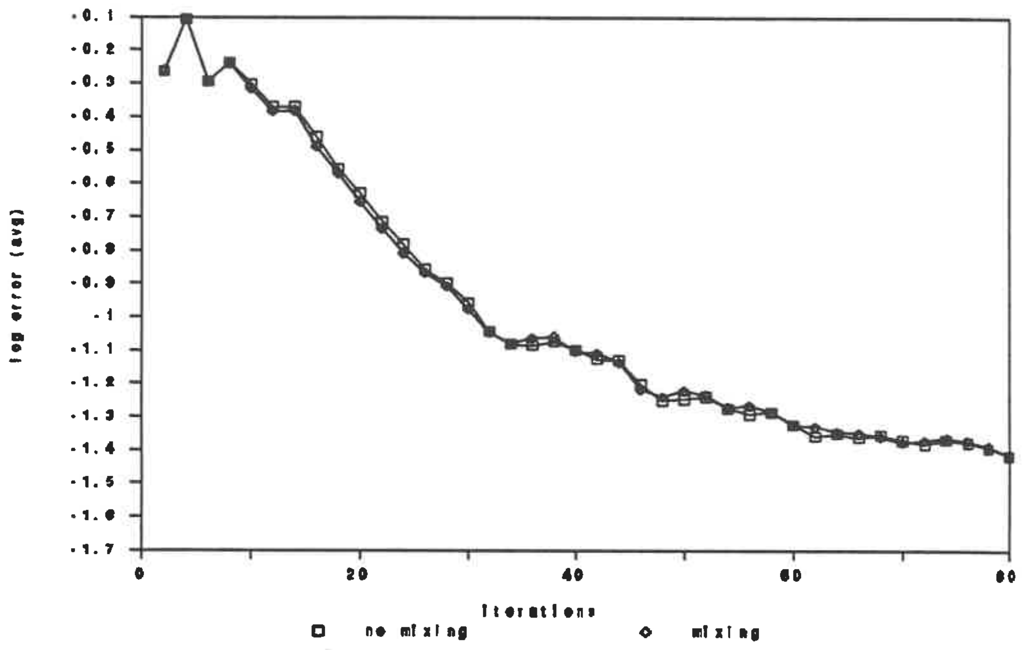


Figure A.8 Average error with or without mixing

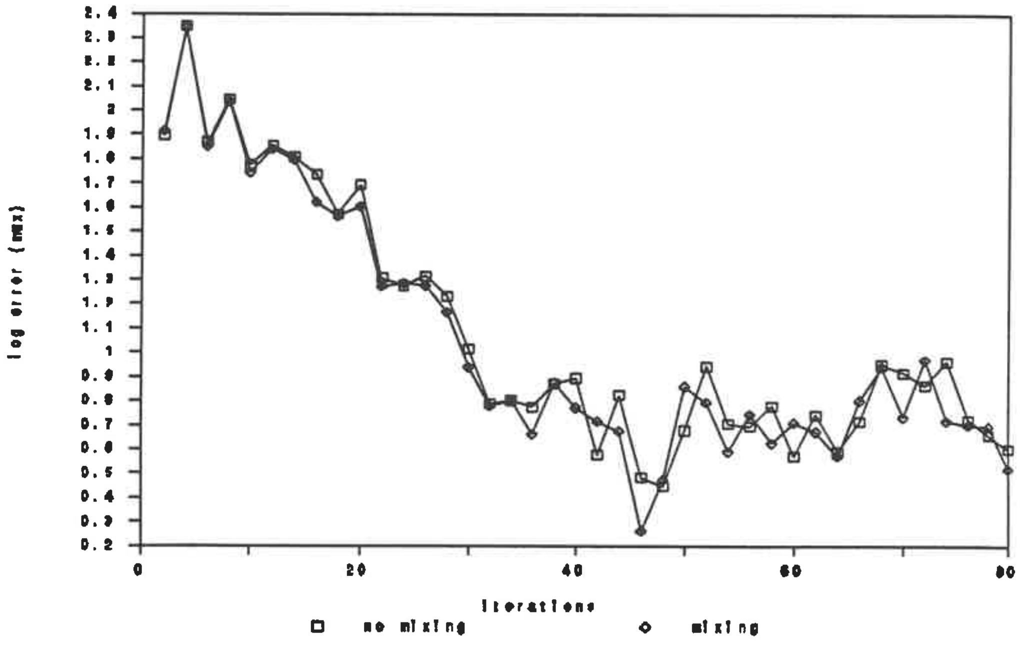


Figure A.9 Maximum error with or without mixing

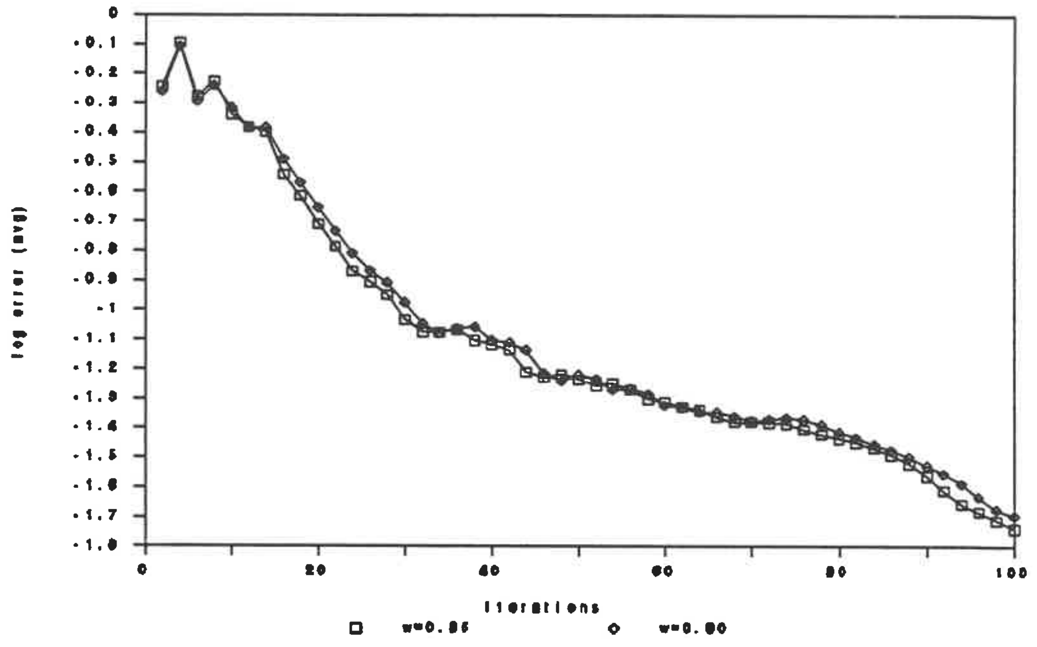


Figure A.10 Average error for 2 different w

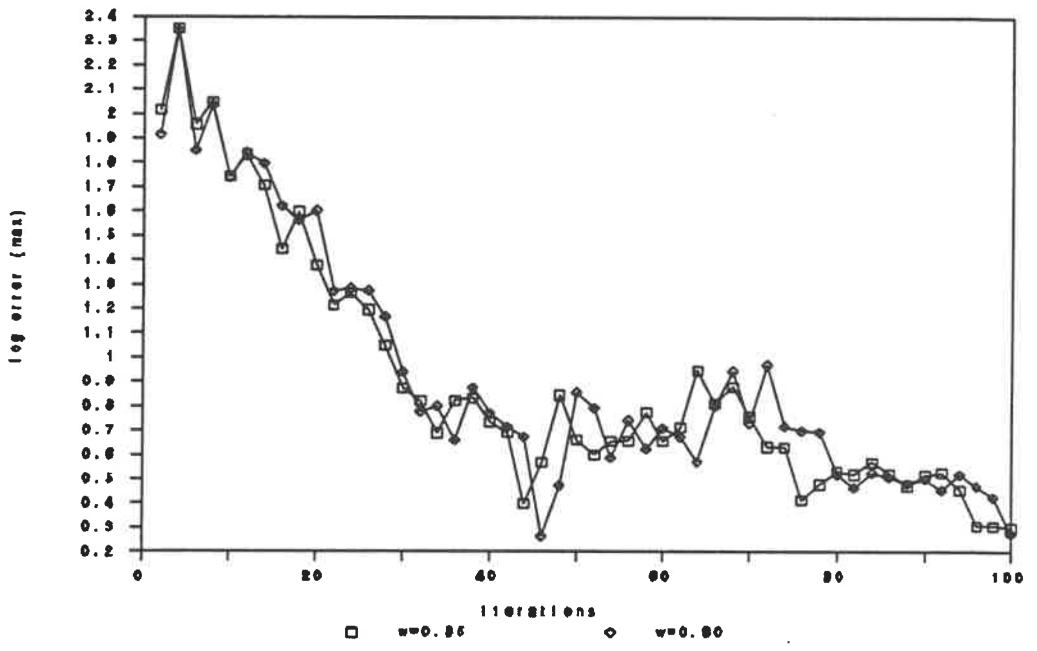


Figure A.11 Maximum error for 2 different w

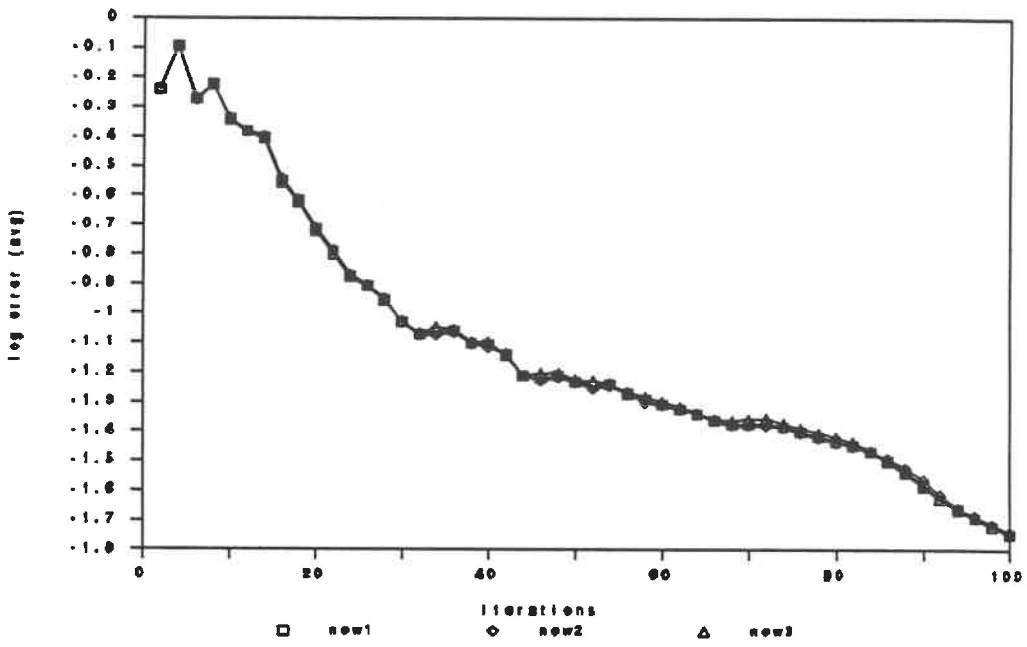


Figure A.12 Average error for the 3 inter. schemes

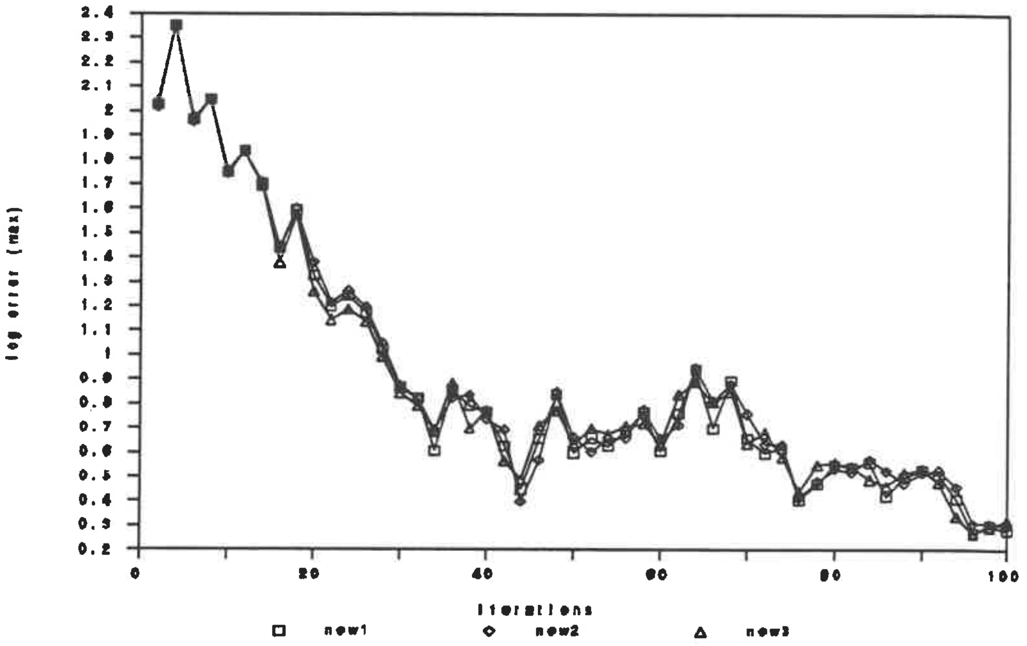


Figure A.13 Maximum error for the 3 inter. schemes

APPENDIX B

Listing of subroutine COARSE

```
C   *****************************************************************
C   *                                                               *
C   *    Subroutine : COARSE                                        *
C   *                                                               *
C   *    CALCULATION OF VOLUMES ON COARSER GRID LEVELS              *
C   *                                                               *
C   *    prepared by : Stephane Major                               *
C   *    DATE        : 20/08/91                                     *
C   *                                                               *
C   *****************************************************************

      SUBROUTINE COARSE

      PARAMETER ( NXI  = 48, NETA  = 16, NKHI  = 16)
      PARAMETER ( NXI1 = 49, NETA1 = 17, NKHI1 = 17)
      PARAMETER ( NXIC = 24, NETAC =  8, NKHIC =  8, NLEV = 2)

      COMMON/COM2/ALPHA,PINF,UINF,RHOINF,VINF,WINF,NI,NJ,
     1            NK,KTIP,C1,C2,C4,H0,J0,KHIGH,OMEG
      COMMON/GRID/X(NXI1,NETA1,NKHI1),Y(NXI1,NETA1,NKHI1),
     1            Z(NXI1,NETA1,NKHI1)
      COMMON/VOLM/DELV(NXI,NETA,NKHI),DVC(NXIC,NETAC,NKHIC,NLEV)
      COMMON/CNTL/MGL

      DIMENSION XC(NXIC+1,NETAC+1,NKHIC+1,NLEV),
     1          YC(NXIC+1,NETAC+1,NKHIC+1,NLEV),
     2          ZC(NXIC+1,NETAC+1,NKHIC+1,NLEV)

C  ****  LEVEL COUNTER INCREMENT ****

      LEV = 0
  100 LEV = LEV + 1

C ****   DEFINITION OF COARSE GRID NODES   ****

      kk = 0
      do 40 k=1,nk+1,2
      kk = kk + 1
      jj = 0
      do 40 j=1,nj+1,2
      jj = jj + 1
      ii = 0
      do 40 i=1,ni+1,2
      ii = ii + 1
```

```
          xc(ii,jj,kk,lev) = x(i,j,k)
          yc(ii,jj,kk,lev) = y(i,j,k)
          zc(ii,jj,kk,lev) = z(i,j,k)

   40 continue

C ****   REDUCTION OF NI, NJ, AND NK   ****

        NI   = NI / 2
        NJ   = NJ / 2
        NK   = NK / 2

C ****   EXCHANGE OF VALUES BETWEEN THE 2 GRID LEVELS   ****

        do 50 k=1,nk+1
        do 50 j=1,nj+1
        do 50 i=1,ni+1
          a              = x(i,j,k)
          x(i,j,k)       = xc(i,j,k,lev)
          xc(i,j,k,lev)  = a
          b              = y(i,j,k)
          y(i,j,k)       = yc(i,j,k,lev)
          yc(i,j,k,lev)  = b
          c              = z(i,j,k)
          z(i,j,k)       = zc(i,j,k,lev)
          zc(i,j,k,lev)  = c
   50 continue

        DO 55 K=1,NK
        DO 55 J=1,NJ
        DO 55 I=1,NI
          DVC(I,J,K,LEV)  = DELV(I,J,K)
   55 CONTINUE
        CALL VOLUME
        DO 60 K=1,NK
        DO 60 J=1,NJ
        DO 60 I=1,NI
          A               = DVC(I,J,K,LEV)
          DVC(I,J,K,LEV)  = DELV(I,J,K)
          DELV(I,J,K)     = A
   60 CONTINUE

        IF (LEV.LT.MGL-1) GOTO 100
```

```
C    ****   SET BACK NI, NJ AND NK TO ORIGINAL VALUES   ****

         DO 70 LEVEL=1,MGL-1
           NI = NI * 2
           NJ = NJ * 2
           NK = NK * 2
   70    CONTINUE

C    ****    UPDATE OF X, Y AND Z OF FINE GRID    ****

         DO 80 K=1,(NK/2)+1
         DO 80 J=1,(NJ/2)+1
         DO 80 I=1,(NI/2)+1
           X(I,J,K)  = XC(I,J,K,1)
           Y(I,J,K)  = YC(I,J,K,1)
           Z(I,J,K)  = ZC(I,J,K,1)
   80    CONTINUE

         Return
         End
```

APPENDIX C

Listing of subroutine MULTI

```
C    ****************************************************************
C    *                                                              *
C    *    SUBROUTINE : MULTI                                        *
C    *                                                              *
C    *    CORRECTION OF P(W) USING THE MULTI-GRID ACCELERATOR       *
C    *                                                              *
C    *    PREPARED BY : STEPHANE MAJOR                              *
C    *    DATE        : 23/11/91                                    *
C    *                                                              *
C    ****************************************************************

      SUBROUTINE MULTI

      PARAMETER ( NXI  = 48, NETA  = 16, NKHI  = 16)
      PARAMETER ( NXI1 = 49, NETA1 = 17, NKHI1 = 17)
      PARAMETER ( NXIC = 24, NETAC =  8, NKHIC =  8, NLEV = 2)

      COMMON/COM2/ALPHA,PINF,UINF,RHOINF,VINF,WINF,NI,NJ,
     1            NK,KTIP,C1,C2,C4,H0,J0,KHIGH,OMEG
      COMMON/COM7/DELW(4,NXI,NETA,NKHI)
      COMMON/GRID/X(NXI1,NETA1,NKHI1),Y(NXI1,NETA1,NKHI1),
     1            Z(NXI1,NETA1,NKHI1)
      COMMON/VOLM/DELV(NXI,NETA,NKHI),DVC(NXIC,NETAC,NKHIC,NLEV)
      COMMON/TIME/DT(NXI,NETA,NKHI)
      COMMON/CNTL/MGL

      REAL  PW(4,NXIC,NETAC,NKHIC,NLEV),PWI(4,NXI1,NETA1,NKHI1)
      REAL  PWF(4,NXI,NETA,NKHI)

      LEV   = 0
  100 LEV = LEV + 1

C   ****  RESIDUAL COLLECTION   ****

      KK = 0
      DO 10 K=1,NK-1,2
      KK = KK + 1
      JJ = 0
      DO 10 J=1,NJ-1,2
      JJ = JJ + 1
      II = 0
      DO 10 I=1,NI-1,2
      II = II + 1
      DO 10 N=1,4
      IF (LEV.EQ.1) THEN

       PW1 = DELV(I,J,K)  * DELW(N,I,J,K)
       PW1 = PW1 + DELV(I+1,J,K)  * DELW(N,I+1,J,K)
       PW1 = PW1 + DELV(I,J+1,K)  * DELW(N,I,J+1,K)
       PW1 = PW1 + DELV(I,J,K+1)  * DELW(N,I,J,K+1)
       PW1 = PW1 + DELV(I+1,J+1,K)  * DELW(N,I+1,J+1,K)
       PW1 = PW1 + DELV(I+1,J,K+1)  * DELW(N,I+1,J,K+1)
       PW1 = PW1 + DELV(I,J+1,K+1)  * DELW(N,I,J+1,K+1)
       PW1 = PW1 + DELV(I+1,J+1,K+1)  * DELW(N,I+1,J+1,K+1)
       PW(N,II,JJ,KK,1)  = PW1 / DVC(II,JJ,KK,1)

      ELSE

       PW1 = DVC(I,J,K,LEV-1)  * PW(N,I,J,K,LEV-1)
       PW1 = PW1 + DVC(I+1,J,K,LEV-1)  * PW(N,I+1,J,K,LEV-1)
       PW1 = PW1 + DVC(I,J+1,K,LEV-1)  * PW(N,I,J+1,K,LEV-1)
       PW1 = PW1 + DVC(I,J,K+1,LEV-1)  * PW(N,I,J,K+1,LEV-1)
       PW1 = PW1 + DVC(I+1,J+1,K,LEV-1)  * PW(N,I+1,J+1,K,LEV-1)
       PW1 = PW1 + DVC(I+1,J,K+1,LEV-1)  * PW(N,I+1,J,K+1,LEV-1)
       PW1 = PW1 + DVC(I,J+1,K+1,LEV-1)  * PW(N,I,J+1,K+1,LEV-1)
       PW1 = PW1 + DVC(I+1,J+1,K+1,LEV-1)  * PW(N,I+1,J+1,K+1,LEV-1)
       PW(N,II,JJ,KK,LEV)  = PW1 / DVC(II,JJ,KK,LEV)

      ENDIF
   10 CONTINUE

C ****  UPDATE OF CONTROL PARAMETERS  ****

      NI = NI / 2
      NJ = NJ / 2
      NK = NK / 2
      KTIP = KTIP / 2

      IF (LEV.LT.MGL-1) GOTO 100

C ****  INTERPOLATION TOWARD THE FINEST GRID LEVEL  ****

      LEV = MGL
  200 LEV = LEV - 1

      NI  = NI * 2
      NJ  = NJ * 2
      NK  = NK * 2
      KTIP = KTIP * 2

      DO 75 N=1,4
      KK = 0
      DO 40 K=2,NK,2
      KK = KK + 1
      JJ = 0
      DO 40 J=2,NJ,2
      JJ = JJ + 1
      II = 0
      DO 40 I=2,NI,2
      II = II + 1
        PWI(N,I,J,K) = PW(N,II,JJ,KK,LEV)
   40 CONTINUE

      DO 50 K=2,NK,2
      DO 50 J=2,NJ,2
        PWI(N,1,J,K)    = 0.5*(PWI(N,2,J,K) + PWI(N,NI,J,K))
        PWI(N,NI+1,J,K) = PWI(N,1,J,K)
      DO 50 I=3,NI-1,2
        PWI(N,I,J,K) = 0.5*(PWI(N,I-1,J,K) + PWI(N,I+1,J,K))
   50 CONTINUE

      DO 60 K=2,NK,2
      DO 60 I=1,NI+1
        PWI(N,I,1,K)    = PWI(N,I,2,K)
        PWI(N,I,NJ+1,K) = PWI(N,I,NJ,K)
      DO 60 J=3,NJ-1,2
        PWI(N,I,J,K) = 0.5*(PWI(N,I,J-1,K) + PWI(N,I,J+1,K))
   60 CONTINUE

      DO 70 I=1,NI+1
      DO 70 J=1,NJ+1
        PWI(N,I,J,1)    = PWI(N,I,J,2)
        PWI(N,I,J,NK+1) = PWI(N,I,J,NK)
      DO 70 K=3,NK-1,2
        PWI(N,I,J,K) = 0.5*(PWI(N,I,J,K-1) + PWI(N,I,J,K+1))
   70 CONTINUE

      DO 75 K=KTIP+1,NK+1
      DO 75 I=1,NI+1
        PWI(N,I,1,K)      = 0.5*(PWI(N,I,1,K) + PWI(N,NI+2-I,1,K))
   75 CONTINUE
```

```
C   ***   AVERAGE 8 CORNERS OF FINER CELLS   ***

      DO 90 N=1,4
      DO 90 K=1,NK
      DO 90 J=1,NJ
      DO 90 I=1,NI
        PWF(N,I,J,K) = PWI(N,I,J,K)      + PWI(N,I+1,J,K)     +
     @                 PWI(N,I,J+1,K)    + PWI(N,I,J,K+1)     +
     @                 PWI(N,I+1,J+1,K)  + PWI(N,I+1,J,K+1)   +
     @                 PWI(N,I,J+1,K+1)  + PWI(N,I+1,J+1,K+1)
        PWF(N,I,J,K) = PWF(N,I,J,K) / 8.0
   90 CONTINUE

C   ***   50 - 50 MIXING WITH COARSE GRID RESIDUAL   ***

      DO 92 N=1,4
      KK = 0
      DO 92 K=1,NK-1,2
      KK = KK + 1
      JJ = 0
      DO 92 J=1,NJ-1,2
      JJ = JJ + 1
      II = 0
      DO 92 I=1,NI-1,2
      II = II + 1
        PWF(N,I,J,K)  = .5*(PWF(N,I,J,K)  + PW(N,II,JJ,KK,LEV))
        PWF(N,I+1,J,K)  = .5*(PWF(N,I+1,J,K)  + PW(N,II,JJ,KK,LEV))
        PWF(N,I,J+1,K)  = .5*(PWF(N,I,J+1,K)  + PW(N,II,JJ,KK,LEV))
        PWF(N,I,J,K+1)  = .5*(PWF(N,I,J,K+1)  + PW(N,II,JJ,KK,LEV))
        PWF(N,I+1,J+1,K)  = .5*(PWF(N,I+1,J+1,K)  + PW(N,II,JJ,KK,LEV))
        PWF(N,I,J+1,K+1)  = .5*(PWF(N,I,J+1,K+1)  + PW(N,II,JJ,KK,LEV))
        PWF(N,I+1,J,K+1)  = .5*(PWF(N,I+1,J,K+1)  + PW(N,II,JJ,KK,LEV))
        PWF(N,I+1,J+1,K+1)  = .5*(PWF(N,I+1,J+1,K+1)+PW(N,II,JJ,KK,LEV))
   92 CONTINUE

C   ***   AVERAGING OF INTERPOLATED RESIDUALS   ***

      DO 94 N=1,4
      DO 94 K=2,NK-1
      DO 94 J=2,NJ-1
      DO 94 I=2,NI-1
       PWF(N,I,J,K)  = PWF(N,I,J,K)/2.0 +
     @ (PWF(N,I+1,J,K)+ PWF(N,I-1,J,K)+ PWF(N,I,J+1,K)+
     @  PWF(N,I,J-1,K)+ PWF(N,I,J,K+1)+ PWF(N,I,J,K-1))/20.0 +
     @ (PWF(N,I,J+1,K+1)+ PWF(N,I,J-1,K+1)+ PWF(N,I,J+1,K-1)+
     @  PWF(N,I,J-1,K-1)+ PWF(N,I+1,J,K+1)+ PWF(N,I-1,J,K+1)+
     @  PWF(N,I+1,J,K-1)+ PWF(N,I-1,J,K-1)+ PWF(N,I+1,J+1,K)+
     @  PWF(N,I-1,J+1,K)+ PWF(N,I+1,J-1,K)+ PWF(N,I-1,J-1,K))/80.0 +
     @ (PWF(N,I+1,J+1,K+1)+ PWF(N,I-1,J+1,K+1)+ PWF(N,I+1,J-1,K+1)+
     @  PWF(N,I+1,J+1,K-1)+ PWF(N,I-1,J-1,K+1)+ PWF(N,I-1,J+1,K-1)+
     @  PWF(N,I+1,J-1,K-1)+ PWF(N,I-1,J-1,K-1))/160.0
   94 CONTINUE

C  ***  CORRECTIONS ON COARSE LEVEL GRID  ***

      IF (LEV.GT.1) THEN
        DO 96 N=1,4
        DO 96 K=1,NK
        DO 96 J=1,NJ
        DO 96 I=1,NI
        PW(N,I,J,K,LEV-1) = (OMEG * PW(N,I,J,K,LEV-1)) +
     @                      ((1.0 - OMEG) * PWF(N,I,J,K))
   96   CONTINUE
      ENDIF

      IF (LEV.GT.1) GOTO 200

C  ***  CORRECTIONS ON FINE LEVEL GRID  ***

      DO 98 N=1,4
      DO 98 K=1,NK-1
      DO 98 J=1,NJ-1
      DO 98 I=1,NI
        DELW(N,I,J,K) = (OMEG * DELW(N,I,J,K)) +
     @                  ((1.0 - OMEG) * PWF(N,I,J,K))
   98 CONTINUE

      RETURN
      END
```

APPENDIX D

User's Guide to E3NEW1 FORTRAN

D.1 FIRST RUN

To execute the new program, the user must be connected to the CMS system on Ecole Polytechnique main frame. The active CMS directory must contain the following files:

```
ITER MULTI
INPUT MULTI
E3NEW1 FORTRAN
WING49 GRID        (or grid file name at user's choice)
```

For a first run the ITER MULTI file must contain three zeros that will be read in free format by the program. The grid file contains a sequence of spatial coordinates as specified in section 3.1.1. The INPUT MULTI file contains all the external information necessary to determine the case run conditions. All values are given on a single line and are separated by at least a blank space (read in free format). The order in which the values must be provided is :

Alpha, Mach, Ni, Nj, Nk, Ktip, C1, C2, C4, Inflow, J0, Khigh, Nots, Ncycle, Mgl, Omeg.

Complete description of these variables is provided in section 3.1.1 and 3.2.1. As an example, the INPUT MULTI file used to run the Cp comparison case is :

```
3.06 0.84  48 16 16 8  1.8 2.0 1.0  3 16 16  2 200  3 0.85
```

To execute the main program, the user must make sure that E3NEW1 is compiled using the FORTVS2 command. To start, the program must be loaded using the LOAD command, then started using the START command. The first thing to do is specify the name of the grid file. Then the program engages the computation and gives a feedback by printing, on the screen, the number of the last iteration completed.

Once the execution is over the user will have access to four new, or refreshed, output files, which are thoroughly described in section 3.1.3.

D.2 SEQUENCE RUN

If the results for a given case are not satisfying, it is possible to continue the iterations further. After a computation, the ITER MULTI file is modified, it now contains the total number of iterations that were performed up to now and the average and maximum errors associated to this number of iterations. In order to continue the computation the program will read this information from ITER MULTI and will toggle the program to read the RESULTS MULTI file generated at the end of the previous run. The final results of the previous run are now the initial values used for the new run. The grid must be the same, and modifying values in the INPUT MULTI is not recommended, but not prohibited. The sequence run is, in any other aspects, identical to a first run.