CHAPTER

Computer Implementation of the CBS Algorithm* 15

15.1 Introduction

In this chapter we consider some essential steps in the computer implementation of the CBS algorithm on structured or unstructured finite element grids. Only linear triangular elements are used and the description given here is intended for a two-dimensional version of the program. The program source and user manual along with several solved problems are available at the website

http://www.zetacomp.com

free of charge. Readers also will find documentation of the code and other details on this website including new results and corrections.

The program can be used to solve the following different categories of fluid mechanics problems:

1. Compressible viscous and inviscid flow problems
2. Incompressible viscous flows
3. Incompressible flows with heat transfer

With further simple modifications, many other problems such as turbulent flows, solidification, mass transfer, free surfaces, etc., can be solved. The procedures presented are largely based on the theory presented in Chapter 3. The program is written in Fortran and it is assumed that the reader is familiar with coding and finite element procedures discussed in this book [1,2].

We call the present program CBSflow since it is based on the CBS algorithm discussed in Chapter 3 of this volume. We prefer to keep the compressible and incompressible flow codes separate to avoid any confusion. However an experienced programmer can incorporate both parts into a single code without much difficulty. Each program listing is accompanied by some model problems which may be used to validate an installation. In addition to the model inputs to programs, a complete user manual is available to describe each part of the program in detail. Any error reported by users will be corrected and the program will be continuously updated by the authors.

*Dr Rhodri Bevan's assistance in writing and improving the code is gratefully acknowledged.

Finite Element Meth Fluid Dynam, 7E, Volume 3. http://dx.doi.org/10.1016/B978-1-85617-635-4.00015-7

The modules are (1) the data input module with preprocessing; (2) the solution module; and (3) the output module. The CBSflow program contains the source statements to solve transient Navier-Stokes (or Euler-Stokes) equations iteratively. Here there are many possibilities such as fully explicit forms, semi-implicit forms, quasi-implicit forms, and fully implicit forms as presented in Chapter 3. We concentrate mainly on the first two forms which require small memory and simple solution procedures compared to other forms.

In both the compressible and incompressible flow codes, only nondimensional equations are used. The reader is referred to the appropriate chapters (Chapters 3, 4, and 5) for different nondimensional parameters.

In Section 15.2 we describe the essential features of data input to the program. Here either structured or unstructured meshes can be used to divide the problem domain into finite elements. Section 15.3 explains how the steps of the CBS algorithm are implemented. In that section, we briefly remark on the options available for shock capturing, various methods of time stepping, and different procedures for equation solving. In Section 15.4, the output generated by the program and postprocessing procedures are considered.

15.2 The data input module

This part of the program is the starting point of a calculation where the input data for the solution module are prepared. Here an appropriate input file is opened and the data are read from it. The mesh generators are provided separately. By suitable coupling, the reader can implement various adaptive procedures as discussed in Chapters 4 and 6. Either structured or unstructured mesh data can be given as input to the program. Readers are referred to the user manual for further details.

15.2.1 Mesh data: Nodal coordinates and connectivity

Once the nodal coordinates and connectivity of a finite element mesh are available from a mesh generator, they are allotted to appropriate arrays. The coordinates are allotted to $coord(i, j)$ with i defining the appropriate Cartesian coordinates $x_1(i = 1)$ and $x_2(i = 2)$ and j defining the global node number. Similarly the global node numbers describing the connectivity are allotted to an array $intma(k, l)$. Here k is the local node number and l is the global element number. It should be noted that the material code normally used in heat conduction and stress analysis is not used but can be introduced if necessary.

15.2.2 Boundary data

In CBSflow we mostly use the edges to store the information on boundary conditions. Some situations require boundary nodes (e.g., pressure specified in a single node) and in such cases corresponding node numbers are supplied to the solution module.

15.2.3 Other necessary data and flags

In addition to the mesh data and boundary information, the user needs to input a few more parameters used in flow calculations. For example, compressible flow computations need the values of nondimensional parameters such as the Mach number, Reynolds number, Prandtl number, etc. Here the reader may consult the nondimensional equations and parameters discussed in Section 3.1, Chapter 3, and in Chapter 4. Further details are available in the user manual.

Several flags for boundary conditions, shock capture, etc., need to be given as inputs. For a complete list of such flags, the reader is referred to the user manual and program listing www.zetacomp.com.

15.2.4 Preliminary subroutines and checks

A few preliminary subroutines are called before the start of the time iteration loop. Establishing the surface normals, element area (for direct integration), mass matrix calculation and lumping, and some allocation subroutines are necessary before starting the time loop.

15.3 Solution module

The solution module primarily contains the following steps (explicit time stepping):

```
pre-processing
do iter = 1, number of time steps
   call alotim ! allot appropriate time step value
   call shock  ! calculate shock capturing viscosity
   call step1  ! intermediate momentum
   call step2  ! calculate density/pressure
   call step3  ! correct momentum
   call energy ! energy equation
   call press  ! relate density and pressure using energy
   call bound  ! apply boundary conditions
   call check  ! check steady state criterion
enddo !iter
post-processing
```

The time iteration is carried out over the steps of the CBS algorithm and over many other subroutines such as the local time step and shock capture calculations. As mentioned, the energy can be calculated after the velocity correction. However, for a fully explicit form of solution, the energy equation can be solved in step 1 along with the intermediate momentum variable if preferred.

Most of the routines within the time loop are further branched into several other subroutines. For instance, convection and diffusion are treated using separate routines within each step.

15.3.1 Time step

In general, three different ways of establishing the time steps are possible. In problems where only the steady state is of importance, so-called "local time stepping" is used (Chapter 3). Here a local time step at each and every nodal point is calculated and used in the computation.

When we seek accurate transient solution of any problem, the so-called "minimum step" value is used. Here the minimum of all local time step values is calculated and used in the computation.

Another and less frequently used option is that of giving a "fixed" user-prescribed time step value. Selection of such a quantity needs considerable experience from solving several flow problems.

The time loop starts with a subroutine where the above-mentioned time step options are available (`alotim`). If the last option of the user-specified fixed time step is used, the local time steps are not calculated.

15.3.2 Shock capture

The CBS algorithm introduces naturally some terms to stabilize the oscillations generated by the convective acceleration. However, for compressible high-speed flows, these terms are not sufficient to suppress the oscillations in the vicinity of shocks and some additional artificial viscosity terms need to be added (Chapter 7). We have given two different forms of artificial viscosities based on the second derivative of pressure in the program. Another possibility is to use anisotropic shock capturing based on the residual of individual equations solved. However we have not used the second alternative in the program as the second derivative–based procedures give quite satisfactory results for all high-speed flow problems.

In the first method implemented, we need to calculate a pressure switch (Chapter 7) from the nodal pressure values. We calculate the inner nodal switch as (Fig. 15.1)

$$S_1 = \frac{|4p_1 - p_2 - p_3 - p_4 - p_5|}{|p_1 - p_2| + |p_1 - p_3| + |p_1 - p_4| + |p_1 - p_5|} \tag{15.1}$$

and for the boundary node we calculate

$$S_1 = \frac{|5p_1 - 2p_2 - p_3 - 2p_4|}{2|p_1 - p_2| + |p_1 - p_3| + 2|p_1 - p_4|} \tag{15.2}$$

The nodal quantities calculated in a manner described above are averaged over elements.

In the next option available in the code, the second derivative of pressure is calculated from the smoothed nodal pressure gradients (Chapter 4) by averaging. Other approximations to the second derivative of pressure are described in Chapter 4. The user can employ those methods to approximate the second derivative of pressure if desired.

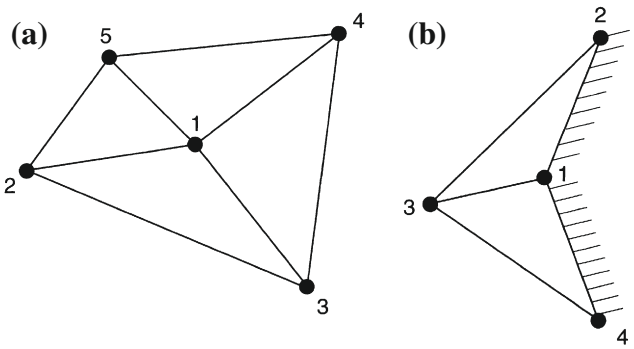

FIGURE 15.1

Typical element patches: (a) interior node; (b) boundary node.

15.3.3 CBS algorithm: Steps

In the subroutine `stepl` we calculate the temperature-dependent viscosity at the beginning according to Sutherland's relation (see Chapter 7). The averaged viscosity values over each element are used in the diffusion terms of the momentum equation and dissipation terms of the energy equation. The diffusion, convective and stabilization terms are integrated over elements and assembled appropriately to the RHS vector. The integration is carried out directly. Finally the RHS vector is divided by the lumped mass matrices and the values of intermediate momentum variables are established.

In step 2, in explicit form, the density/pressure values are calculated. The subroutine `step2` is used for this purpose. Here the option of using different values of θ_1 and θ_2 is available. In explicit form θ_2 is identically equal to zero and θ_1 varies between 0.5 and 1.0. For compressible flow computations, the semi-implicit form with θ_2 greater than zero has little advantage over the fully explicit form. For this reason we have not given the semi-implicit form for compressible flow problems in the program.

The third step is the one where the intermediate momentum variables are corrected to get the real values of the intermediate momentum. In all three steps, mass matrices are lumped if the fully explicit form of the algorithm is used. As mentioned in earlier chapters, this is the best way to accelerate the steady-state solution along with local time stepping. However, in problems where transient solutions are of importance, either a mass matrix correction as given in Chapter 2 or simultaneous solution using a consistent mass matrix may be necessary. The dual time stepping approach used in Chapter 3 may also be used.

15.3.4 Boundary conditions

As explained before, the boundary edges are stored along with the elements to which they belong. Also in the same array `iside(i, j)` the flags necessary to inform the solution module about the type of boundary conditions are stored. In this array

$i = 1, 2$ correspond to the node numbers of any boundary side of an element, $i = 3$ indicates the element to which the particular edge belongs, and $i = 4$ is the flag which indicates the type of boundary condition (a complete list is given in the user manual available at the web page). Here j is the boundary edge number.

15.3.5 Solution of simultaneous equations: Semi-implicit form

The simultaneous equations need to be solved for the semi-implicit form of the CBS algorithm. Two types of solvers are provided. The first one is a banded solver which is effective when structured meshes are used. For this the half-bandwidth is necessary in order to proceed further. The second solver is a conjugate gradient solver. The latter can be used to solve both structured and unstructured meshes. The details of procedures for solving simultaneous equations can be found in Ref. [3].

15.3.6 Different forms of energy equation

In compressible flow computations only the fully conservative form of all equations ensures correct position of shocks. Thus in the compressible flow code, the energy equation is solved in its conservative form with the variable being the energy. However for incompressible flow computations, the energy equation can be written in terms of the temperature variable and the dissipation terms can be neglected. In general for compressible flows, Eq. (3.14) is used, and Eq. (4.6) is used for incompressible flow problems.

15.3.7 Convergence to steady state

The residuals (difference between the current and previous time step values of parameters) of all equations are checked at every few user-prescribed number of iterations. If the required convergence (steady state) is achieved, the program stops automatically. The aimed residual value is prescribed by the user. The program calculates the L_2 norm of the residual of each variable over the domain. The user can use them to fix the required accuracy.

15.4 Output module

If the imposed convergence criteria are satisfied then the output is written into a separate file. The user can modify the output according to the requirements of the postprocessor employed. The formats for Tecplot [4] and ParaView [5] are provided.

References

[1] I. Smith, D.V. Griffiths, Programming the Finite Element Method, Wiley, Chichester, 1998.
[2] D.R. Willé, Advanced Scientific Fortran, Wiley, Chichester, 1995.

[3] O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, The Finite Element Method: Its Basis and Fundamentals, seventh ed., Elsevier, Oxford, 2013.
[4] Tecplot, Inc., Bellevue, Washington, Tecplot: Visualization software. <www.tecplot.com>.
[5] Sandia National Laboratories, Paraview: Visualization software. <www.paraview.org>.