

Argonne National Laboratory, with facilities in the states of Illinois and Idaho, is owned by the United States Government and operated by The University of Chicago under the provisions of a contract with the Department of Energy.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor The University of Chicago, nor any of their employees or officers, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of document authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, Argonne National Laboratory, or The University of Chicago.

Available electronically at <http://www.doe.gov/bridge>

Available for a processing fee to U.S. Department of Energy and its contractors, in paper, from:

U.S. Department of Energy
Office of Scientific and Technical Information

P.O. Box 62

Oak Ridge, TN 37831-0062

phone: (865) 576-8401

fax: (865) 576-5728

email: reports@adonis.osti.gov

Modelling of liquid metal duct and free-surface flows using CFX

S. Aleksandrova, Coventry University

S. Molokov, Coventry University

C. B. Reed, Argonne National Laboratory

Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439

Work supported by the
Office of Fusion Energy Sciences
U.S. Department of Energy
Under Contract W-31-109-ENG-38

Contents

1.	Introduction	4
2.	Software	6
3.	Solution process	7
4.	Hardware.....	9
5.	MHD flow modelling with CFX	9
5.1	Formulation	9
5.1.1	Governing equations	10
5.1.2	Dimensionless equations	11
5.1.3	Boundary conditions	12
5.1.4	Free surface flows	12
5.1.5	Surface tension	13
5.2	Implementation in CFX.....	14
5.2.1	Boundary conditions	15
5.2.2	Transient flow.....	17
5.2.3	Free-surface flow	17
5.2.4	Convergence criteria	18
5.2.5	2D flows.....	18
5.2.6	3D flows.....	20
5.2.7	Command file options.....	23
5.2.8	Additional subroutines used	49
6.	Benchmark problems	55

6.1	Duct flows.....	55
6.1.1	Shercliff solution.....	55
6.1.2	Hunt solution.....	56
6.1.3	Flow in an expansion.....	57
6.1.4	Flow in a square duct in a non-uniform transverse magnetic field.....	59
6.2	Free-surface flows.....	62
6.2.1	Spreading MHD drop.....	62
6.3	MHD jet.....	64
6.3.1	Inertialess flow in a uniform field.....	66
6.3.2	Inertial flow in a uniform field.....	66
6.3.3	Inertialess flow in a non-uniform field.....	66
7.	Discussion and conclusions.....	68
7.1	Main problems of modelling of MHD flows.....	68
7.2	Comparison of the DNS and high- <i>Ha</i> flow model.....	71
8.	Acknowledgement.....	73
9.	References.....	73
10.	List of figures.....	78
11.	Appendix 1: a sample of command file.....	85
12.	Appendix 2: a sample of source code, two-dimensional flow, MHD jet ..	87
13.	Appendix 3: a sample of source code, three-dimensional flow, Hunt solution.....	115

1. Introduction

Liquid metal free-surface flows provide an option of a renewable surface for heat absorption, removal of impurities, and eliminating the problems of erosion and thermal stresses [1], [2]. In a tokamak liquid metal flows through a strong magnetic field, which results in a magnetohydrodynamic (MHD) interaction. For a free-surface flow the MHD interaction may be even more important than for the duct flows in blankets, because the electromagnetic forces may significantly deform the free-surface and thus make it unfavourable for heat extraction. The MHD-related problems for the free-surface flows have been reviewed in [3]. Among the most important ones are the effects of nonuniform magnetic fields, inertia, surface tension, wettability and roughness of walls on both the jet/drop shape and trajectory.

The main problems for the jet divertor are shown in Figure 1 and Figure 2 [4]. Particular issues related to some of the problems listed in these figures have already been tackled (Problem 1 in [5]-[7], 4 in [8], [9], 5 in [9], 6 in [4], 7 and 10 in [9], 9 in [10]). Once main fundamental aspects for each of these sub-problems are understood, the analysis will have been performed for a particular divertor design.

The flow analysis in [4]-[8], [10] has been performed using the asymptotic model for high values of the Hartmann number, Ha , in the inertialess approximation, while in [9] inertial effects have been taken into account. The dimensionless parameters are defined in Sec. 5.1.2. In large-scale tokamaks the magnetic field is $\sim 5-10\text{T}$, which results in high or very high values of Ha and N , the interaction parameter (Table 1). Recently it has been suggested that the presence of liquid Li inside a plasma chamber might stabilize the plasma [11]. Thus a decision has been made in the US to test a free-surface device in smaller machines, such as NSTX or C-MOD [12]. In these machines the magnetic fields are lower (Table 1), and consequently the high- Ha analysis may not apply.

Therefore, the decision has been made to attempt Direct Numerical Simulation (DNS) to model flows for lower magnetic fields. It should be emphasized that compared to the high- Ha flow model, DNS for MHD flows may be considered to be in its infancy,

although some initial progress has been made ([13]-[22], [44]). A general discussion of the difficulties with DNS modelling for fusion-related MHD flows and the recommendations for further model development are given in Sec. 7.

Table 1 Dimensionless parameters and expected flow regimes for Li duct- and free-surface flows in various tokamaks (parameters have been estimated in [3], [5], [9]).

	NSTX	C-MOD	LARGE-SCALE MACHINES
Hartmann number	~50-500	~500-1000	~10³-10⁵
Interaction parameter	0.1-1	~50-10⁴	~10³-10⁵
Flow regime	Fully three-dimensional, turbulent MHD	Laminar; possibly 2-D turbulent MHD	Laminar MHD
Features	MHD effects are weak; both bending and “flattening” of the jet may not be very expressed	MHD effects are strong. However, both poloidal and radial field are about 10% of the toroidal one: bending of the jet may not be very expressed	MHD effects are very strong. Although both poloidal and radial field are about 10% of the toroidal one, bending of the jet may be significant
High-Ha model applicable?	no	yes	yes, especially for this regime
DNS applicable?	Not without turbulence models, which do not exist at present	Yes, but will require great computational power	Not with current computational facilities, even those employing supercomputers

For modelling ordinary hydrodynamic flows several commercial codes are available, such as CFX, FLUENT, FLOW-3D, etc. Although most of them do not model MHD flows automatically, they can be modified using user-defined subroutines to include the body forces, such as the Lorentz force, and couple the momentum equations with the other MHD equations.

We have decided to base the MHD code on CFX [23], which is a very flexible finite-volume code widely used in the industrial applications of MHD. The user can introduce several scalar equations and even to modify the accuracy of the numerical scheme ([23], O. Widlund, private communication).

The aims of this report are: to summarise the first steps in the development of the extension of CFX for modelling MHD flows, to present the results of testing the code on several benchmark problems, and to present the description of the code for further reference.

2. Software

The commercial fluid dynamics software package CFX 4.4 is used for solving MHD problems described below. Main features, options and commands relevant to the current study are described in Secs. 3 - 5. More details are given in [23]. CFX consists of the following modules:

- Pre-processing tools

- CFX-Build 4.4

An MSC-Patran based geometry and grid generator. It is used for creating the computational domain, specifying patches (i. e. parts of the domain where the boundary conditions will be defined – such as walls, inlets, outlets, symmetry planes, pressure boundaries etc.) and generating grid. Grid generation tools allow creation of non-uniform grids, which enables one to resolve boundary layers where needed.

- CFX-Setup

This module is used for creating the command file, containing information about fluid properties, physical models, boundary conditions and solver data.

- Fortran routines

Additional Fortran routines are used in order to implement the Magnetohydrodynamic flows and other extra features as described below.

- Solver tools

- CFX-Solver

The basis of the code is a conservative finite-volume method. All variables are defined at the centre of control volumes, and the equations are integrated over each control volume to obtain discrete equations. The complete set of equations is solved by iterative method. Pressure-correction algorithm is used to ensure mass conservation.

- Post-Processing tools

- CFX-Analyse 4.4

Graphic post-processor used for analysis and presentation of results obtained from the solver.

In addition to CFX, the following software tools have been used:

- Digital Fortran

It is used for compilation of the user subroutines that are used by the CFX solver.

- Matlab

Although CFX Analyse provides a wide range of graphic functions, sometimes Matlab is used for visualising solution.

3. Solution process

The process of problem solution in CFX consists roughly of the following stages:

- Pre-processing

-
- Problem formulation.
 - Geometry creation.

The geometry is created using CFX-Build. First, all blocks forming the computational domain are created. Then patches (two-dimensional or three-dimensional subdomains where the boundary conditions will be set) are specified. Finally, computational mesh is set up and the geometry file *.GEO containing the information about mesh points, patches and block gluing information is created. Simple geometries can be created using the command file.

- Command file.

The command file *.FC containing information about the model is created using CFX-Setup program or any text editor.

- Fortran subroutines.

Additional Fortran subroutines *.F are required in order to implement MHD model. They are also used for specifying variable boundary conditions, setting convergence criteria etc.

- Solving

All files created during the pre-processing stage are passed to the CFX-solver. It reserves the workspace, checks the validity of the information provided and runs the problem. During the solution process, some information about it can be displayed on dynamic graphs. Usually it is the residual values for each of the flow variables. It allows user to control the solution process and to see the progress of the solution.

When the calculation is finished, CFX-Solver writes output data into the dump file *.DMP. Additional information about the solution process (e.g. summary of input data, residuals) and some statistics (e.g. drag on the walls, flow rates at all inlets and outlets etc.) are written to an output file *.FO. Additional information may be

included into the dump/output file using command file options and user subroutines.

- Post-processing

Data produced by CFX-Solver can be post-processed using CFX-Analyse or any other graphical tools. In the latter case, data should be written in required format. This can be achieved by using special options in the command file or user subroutines. CFX-Analyse is the built-in post-processing tool that allows user to create 2D plots, mesh plots, streamlines, contour plots, animations etc. It also has some simple calculation tools.

4. Hardware

All three-dimensional problems have been run on a Compaq Alpha Server DS20. It features 2 processors, 500 MHz each and 768 Mb RAM. Some two-dimensional problems have been solved using Pentium-3 (1GHz) with 1 Gb RAM.

5. MHD flow modelling with CFX

5.1 Formulation

We will first discuss the set of governing equations and how these are implemented in the CFX. Since the magnetohydrodynamic features are not included in CFX by default, certain modifications are made in order to implement MHD into the CFX model. These include introduction of the Lorentz force into the momentum equation and solution of the electric potential equation using the additional scalar option available in CFX. By default, CFX deals with dimensional quantities. In order to use the non-dimensional formulation, appropriate coefficients should be introduced as discussed below. Different scales will be used for different problems, based on a particular flow, and these will be discussed separately when formulating the corresponding problem. Moreover, in the free-surface flows the dimensional model is used in some cases. Both transient flow and free-surface flow models are available in CFX.

5.1.1 Governing equations

Consider the flow of a viscous, electrically conducting, incompressible fluid subject to a strong magnetic field $\mathbf{B}_0^* = B_0 b \hat{\mathbf{y}}$. It is assumed throughout this study that the magnetic Reynolds number $Re_m = \mathbf{m} \mathbf{s} v_0 a$ is small. This is not a necessary assumption, see the discussion in [5].

Then the dimensional governing equations are the Navier-Stokes equation

$$\mathbf{r} \frac{\mathcal{D} \mathbf{v}^*}{\mathcal{D} t^*} + \mathbf{r} (\mathbf{v}^* \cdot \nabla^*) \mathbf{v}^* = -\nabla^* p^* + \mathbf{m} \nabla^{*2} \mathbf{v}^* + \mathbf{F}^*, \quad (1)$$

the Ohm's law

$$\mathbf{j}^* = \mathbf{s} (-\nabla^* f^* + \mathbf{v}^* \times \mathbf{B}^*), \quad (2)$$

and conservation of mass and current

$$\nabla^* \cdot \mathbf{v}^* = 0, \quad (3)$$

$$\nabla^* \cdot \mathbf{j}^* = 0. \quad (4)$$

In the above, t^* is time, \mathbf{v}^* is the fluid velocity, p^* is pressure, f^* is the electric potential, \mathbf{F}^* is the volumetric body force, \mathbf{j}^* is the electric current density, \mathbf{E}^* is the electric field, \mathbf{m} is the dynamic viscosity, \mathbf{s} is electric conductivity and \mathbf{r} is fluid density. In MHD problems considered here, the body force includes the Lorentz force and gravity force:

$$\mathbf{F}^* = \mathbf{j}^* \times \mathbf{B}^* + \mathbf{r} \mathbf{g}^*,$$

where \mathbf{g}^* is the gravity vector. Using expression (2) for the electric current, the body force can be rewritten as

$$\mathbf{F}^* = -\mathbf{s} \nabla^* f^* \times \mathbf{B}^* + \mathbf{s} \mathbf{v}^* \times \mathbf{v}^* \times \mathbf{B}^* + \mathbf{r} \mathbf{g}^*. \quad (5)$$

Taking divergence of Eq. (2) and using Eq. (4), an equation governing the electric potential is obtained. It reads

$$\nabla^{*2} \mathbf{f}^* = \nabla^* \cdot (\mathbf{v}^* \times \mathbf{B}^*). \quad (6)$$

Thus a set of equations (1), (3) and (6) for fluid velocity, electric potential and pressure has to be solved subject to appropriate boundary conditions. The Ohm's law (2) is used for calculation of the electric current.

5.1.2 Dimensionless equations

The characteristic values of the length, the fluid velocity, time, the electric potential and the pressure are a , v_0 , a/v_0 , av_0B_0 and $a\mathbf{S}v_0B_0^2$, respectively. Then the non-dimensional equations governing the flow are

$$N^{-1} \left[\frac{\nabla \mathbf{v}}{\nabla t} + (\mathbf{v} \cdot \nabla) \mathbf{v} \right] = -\nabla p + Ha^{-2} \nabla^2 \mathbf{v} - \nabla \mathbf{f} \times \mathbf{B} + \mathbf{v} \times \mathbf{B} \times \mathbf{B} + d \hat{\mathbf{g}}, \quad (7)$$

$$\nabla^2 \mathbf{f} = \nabla \cdot (\mathbf{v} \times \mathbf{B}), \quad \nabla \cdot \mathbf{v} = 0, \quad (8-9)$$

where $Ha = aB_0 \sqrt{\mathbf{s} / \mathbf{r} \mathbf{n}}$ is the Hartmann number that expresses the ratio of the electromagnetic to the viscous force, $N = a\mathbf{S}B_0^2 / \mathbf{r}v_0$ is the interaction parameter, which expresses the ratio of the electromagnetic to the inertial force, $d = \mathbf{r}g / \mathbf{S}v_0B_0^2$ is the parameter which expresses the ratio of the gravitational to the electromagnetic force; $\hat{\mathbf{g}}$ is the unit vector in the direction of gravity.

The electric current can be calculated from the non-dimensional form of the Ohm's law

$$\mathbf{j} = -\nabla \mathbf{f} + \mathbf{v} \times \mathbf{B}. \quad (10)$$

The velocity scale, v_0 , depends on a particular problem. If gravity is important, we select

$$v_0 = \mathbf{r}g / \mathbf{S}B_0^2, \quad (11)$$

so that $\mathbf{d} = 1$. In case of the pressure-driven flow, the average duct velocity Q/a (Q is the flow rate) is chosen as the velocity scale.

It is possible to model flows with three components of the magnetic field. In the following benchmark problems the field is supposed to have only one component in the direction of y .

5.1.3 Boundary conditions

The boundary condition for the fluid velocity at each wall is the no-slip condition

$$\mathbf{v} = 0. \quad (12)$$

For an electrically insulating wall,

$$\mathbf{j} \cdot \hat{\mathbf{n}} = \frac{\partial \mathbf{f}}{\partial n} = 0 \quad (13)$$

at the wall. Here $\hat{\mathbf{n}}$ is the normal unit vector to the wall. Additional symmetry conditions are specified for each problem if needed.

For a perfectly conducting wall,

$$\mathbf{f} = 0 \text{ at the wall.} \quad (14)$$

5.1.4 Free surface flows

CFX models free-surface flows by assuming that the surrounding medium is dynamically active. A liquid metal is surrounded by what is called "air" with certain values of density, \mathbf{r}_1 , and dynamic viscosity, μ_1 . The "air" may be vacuum if both \mathbf{r}_1 and μ_1 tend to zero.

For the coupled liquid metal-"air" system governing equations (7)-(9) (only Eqs. (7), (9) for "air" without the MHD terms) are solved for each phase with appropriate parameters. Free surface flow is modelled by volume of fluid method. In this case the boundary conditions at the free surface are satisfied automatically. Details of the two-phase model used are given in Sec. 5.2.3.

5.1.5 Surface tension

At the interface between two fluids, there is a surface tension force. Physically this force acts directly on the surface. This is hard to achieve in CFX as the surface is always smeared out in some way by the discretisation. The model used in CFX-4 is based on the Continuum Surface Force model. This models the surface tension force as a force which exists throughout the flow based upon derivatives of volume fraction, but which has the same effect overall as the surface force, even when the surface is smeared.

The model leads to an extra body force, \mathbf{F}_s^* , in the momentum equation given by:

$$\mathbf{F}_s^* = \gamma \mathbf{k}^* \nabla^* r, \quad (15)$$

where γ is the surface tension coefficient, r is the volume fraction of the first phase and \mathbf{k}^* is the surface curvature defined by:

$$\mathbf{k}^* = \frac{1}{|\hat{\mathbf{n}}|} \left(\left(\frac{\hat{\mathbf{n}}}{|\hat{\mathbf{n}}|} \cdot \nabla^* \right) \hat{\mathbf{n}} - \nabla^* \cdot \hat{\mathbf{n}} \right). \quad (16)$$

Here $\hat{\mathbf{n}} = \nabla r$ is a unit vector normal to the free surface.

In a non-dimensional form based on gravitational pressure scaling (to be used in Sec. 6.2.1 for a spreading drop problem), the surface tension body force is equal to

$$\mathbf{F}_s = Bo^{-1} \mathbf{k} \nabla r, \quad (17)$$

where Bo is the non-dimensional Bond number

$$Bo = \frac{a^2 \mathbf{r} g}{\mathbf{g}}. \quad (18)$$

If the variables are scaled as for the duct flows [9], then

$$\mathbf{F}_s = Ca^{-1} Ha^{-2} \mathbf{k} \nabla r, \quad (17)$$

where Ca is the capillary number

$$Ca = \frac{\mathbf{r}m v_0}{\mathbf{g}}.$$

This scaling will be used in Sec. 6.3 for modelling two-dimensional jet flow. For convenience, we introduce parameter

$$\mathbf{l} = Ca^{-1} Ha^{-2},$$

which will be used in calculations.

5.2 Implementation in CFX

As has been mentioned above, CFX codes are built to work with dimensional equations. Thus equations (1) and (3) can be implemented directly by setting physical properties of the fluid in the command file. The “buoyant flow” option allows accounting for the gravity force. However, the electromagnetic force has to be introduced as a source term for the momentum equation. The treatment of the electromagnetic force is discussed below.

Equation (6) can be solved treating the electric potential as a user-defined scalar governed by the steady version of a diffusion-convection equation with the source term $\nabla \cdot (\mathbf{v} \times \mathbf{B})$ calculated for each control volume.

In order to use non-dimensional formulation given by Eqs. (7)-(9), appropriate non-dimensional parameters should be used rather than actual viscosity, density etc. It follows from Eqs. (7), (1) that N^{-1} stands for fluid density while Ha^{-2} corresponds to fluid viscosity. The gravity vector for gravity-dominated problems should have length 1 instead of 9.81, which can also be defined in CFX command language.

The details of how to introduce the Lorentz force into the momentum equations and the source term into the electric potential equation are considered below separately for two-dimensional and three-dimensional flows.

Note that the two-dimensional model in CFX always assumes that the co-ordinate z is the direction in which nothing changes. In fully developed flows and flows in ducts the z co-ordinate is aligned with the axis of the duct.

5.2.1 Boundary conditions

There are several types of boundary conditions available in CFX. The following types have been used in this report: inlets, mass flow boundaries, pressure boundaries, walls. These are described below. Most of the boundary conditions can be specified in the command files. In more complicated problems, user subroutines are used.

To specify a boundary where the appropriate boundary conditions will be applied, special subdomains must be created beforehand. These are called "patches" and can be either two-dimensional or three-dimensional.

Thus, a boundary condition in CFX command file consists of a patch name where the condition holds, name of variable which is specified and the appropriate value of this variable.

Inlet boundaries

An inlet is a boundary where the values of variables are specified. Mathematically, this is known as a Dirichlet boundary condition. These values are set using the subcommand `>>INLET BOUNDARIES` in the command file. If variable profiles are required, they have to be specified using the user Fortran subroutine `USRBCS`.

Default values of the flow variables are their ambient values. By default, the ambient values are 0 and can be overridden in the command file using command `>>AMBIENT CONDITIONS`.

In problems considered in this report, the following flow variables must be set at inlet boundaries: normal velocity into domain, user scalar standing for electric potential where applicable, volume fraction in free-surface flows.

Mass flow boundaries

Mass flow boundaries are used to model inflow and outflow boundaries where the total mass flow rate into or out of the domain is known, but the detailed velocity profile is not. These mass flow rates and other information are specified using the >>FLUX and >>INFLOW VARIABLES subcommands of >>MASS FLOW BOUNDARIES, which is a subcommand of >>MODEL BOUNDARY CONDITIONS.

Mass flow boundaries are used at the entrance to the duct when the mass flow rate is known. It can also be used at the exit from the duct to model Neumann boundary conditions. In this case, fractional mass flow rate equal to 1 is specified to indicate that all outgoing fluid flows through this outlet.

If the flow is out of the domain, all variables except for pressure will satisfy Neumann boundary conditions. If the flow is into domain (negative mass flow rate is specified), Neumann boundary conditions are imposed on all flow variables except for user scalars and pressure. This, unfortunately, means that even if the velocity profile is not known and can be modelled by CFX using the specified flow rate, one still needs to specify correct values for the electric potential. It can be set in the command file if constant. Variable profiles must be set using the user routine USRBCS.

Pressure boundaries

They are used to model both inflow and outflow boundaries where the surface pressure is known, but the detailed velocity distribution is not. These can be used in duct flows to set a constant pressure drop instead of the flow rate. Fixed values of temperature and user scalars must be specified at pressure boundaries if the flow is into the solution domain. The pressure is specified at the pressure boundary using the command >>PRESSURE BOUNDARIES which is a subcommand of >>MODEL BOUNDARY CONDITIONS. Variable pressure profiles must be specified using the user subroutine USRBCS.

Wall boundaries

Default conditions at solid walls are non-slip conditions for the velocity and zero Dirichlet conditions for user scalars. Alternatively, non-zero values or shear stress can be defined for the velocity. General type of boundary conditions can be set for user scalars. Command >>WALL BOUNDARIES is used for this purpose. User subroutine USRBCS can be used to set variable boundary conditions at solid walls.

5.2.2 Transient flow

Most of the problems considered in this report are the steady-state flows. However, in some cases time-dependent formulation of the problem helps to improve problem convergence. A steady-state calculation may be considered as a transient calculation with an infinite time step. If a time accurate simulation of the flow from its initial guess to its steady state solution indicates that the flow approaches the steady state solution in a very complex manner, then it is quite likely that an attempt to reach the steady state solution in a single, large time step will overshoot the mark and never recover.

Therefore in some problems, particularly in free-surface flows, we kept the time-dependent term in Eq. (7) and run the problem until the steady solution had been reached.

5.2.3 Free-surface flow

CFX offers several models for treating two-phase flows. The homogeneous model is the most appropriate one when considering free-surface flows. This model is based on the assumption that certain solution fields of each phase are identical (e.g. velocity, pressure) whilst still solving for distinct volume fractions. Hence, individual phase continuity equations can be solved to determine the volume fractions, while individual transport equations can be summed over all phases to give a single transport equation.

5.2.4 Convergence criteria

The default convergence criterion for steady flows and transient flows with a fixed time step is a condition on the mass flow residual. It is set in the command file using subcommand MASS SOURCE TOLERANCE of the command PROGRAM CONTROL. It may be modified in a user-defined Fortran routine USRCVG. This subroutine returns a logical flag LCONVG. Setting LCONVG to .TRUE. indicates to the program that the flow is satisfactorily converged.

The MASS SOURCE TOLERANCE is not used for transient flows using the adaptive time stepping option. In this case user has more control over the convergence and divergence criteria from the command language. The relevant parameters are set using the subcommands >>CONTROL PARAMETERS and >>CONVERGENCE TESTING ON VARIABLE of the >>TRANSIENT CONTROL command. Any variable can be chosen to test for convergence, not just the mass source residual. Convergence is obtained when the residual for the tested variable for the first phase satisfies the condition $RES < MAX (RESMIN, MIN(RESMAX, RES5/REDUC))$, where RES is the residual of the corresponding variable, RESMIN is the minimum required residual value, RESMAX is the maximum allowed residual value, REDUC is the residual reduction factor and RES5 is the value of the variable's residual computed on the fifth iteration.

5.2.5 2D flows

In a two-dimensional approximation the description of the flow is considerably simplified. All quantities with the exception of f are supposed to be independent of z . If laterally the flow is confined by perfectly conducting sidewalls located at $z = \pm L^*$, which are connected through a resistor, then the resulting electric field is given. In this case, sufficiently far from the sidewalls the electric current flows in the z -direction only, while the flow may be considered two-dimensional in the (x, y) plane. If the sidewalls are not connected, no electric field is present in the fluid and therefore both E and f vanish.

Therefore, the potential equation (8) does not need to be solved. For a magnetic field $B(x)\hat{\mathbf{y}}$ the electric current reduces to one component:

$$j_z = E + uB. \quad (19)$$

The Lorentz force only acts in x -direction, and is equal to

$$F_L = -(E + uB(x))B(x), \quad (20)$$

and the resulting body force is

$$\mathbf{F} = (-EB(x) - uB^2(x) + \mathbf{d}g_x)\hat{\mathbf{x}} + \mathbf{d}g_y\hat{\mathbf{y}}, \quad (21)$$

where g_x and g_y are the corresponding components of the gravity vector; E is a given constant, which is known and may be equal to zero depending on the problem.

Thus two-dimensional problems are described by the following equations deduced from Eqs. (7)-(9):

$$N^{-1} \left[\frac{\mathcal{I}u}{\mathcal{I}t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right] = -\frac{\partial p}{\partial x} + Ha^{-2} \nabla_{xy}^2 u - (E + uB)B + \mathbf{d}g_x, \quad (22)$$

$$N^{-1} \left[\frac{\mathcal{I}v}{\mathcal{I}t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right] = -\frac{\partial p}{\partial y} + Ha^{-2} \nabla_{xy}^2 v + \mathbf{d}g_y. \quad (23)$$

Body force defined by Eq. (21) can be implemented in CFX in a straightforward way. Since all the components of the body force are defined at the centre of the control volume, the source term for each control volume in the momentum equation will be equal to \mathbf{F} multiplied by the volume of the respective cell. An example of implementing Lorentz force in case of an MHD jet is given in Appendix 2.

In order to solve free-surface flows in CFX, similar equations but without the MHD terms governing the surrounding medium (e. g. air) should also be provided.

5.2.6 3D flows

The situation is different in fully three-dimensional flows. In this case the electric potential equation (8) has to be solved which is coupled to the Navier-Stokes equation (7). There are non-zero three-dimensional electric currents induced by the magnetic field inside the fluid. These consist of the electric field $-\nabla\mathbf{f}$ and the electromotive force $\mathbf{v}\times\mathbf{B}$ (see expression (10)). The straightforward approach to the problem is to approximate the body force defined in the expression (26) at the centre of each cell and multiply this value by the volume of the cell. However, this approach fails in some cases and a more careful numerical treatment of the body force should be introduced in order to preserve global electric current conservation ([20]).

Electric currents

For fully three-dimensional flows, non-zero three-dimensional currents defined by Eq. (10) are present in the fluid. For a magnetic field $\mathbf{B} = B\hat{\mathbf{y}}$, applied in the y -direction, the three components of the electric current are

$$j_x = -\frac{\partial\mathbf{f}}{\partial x} - wB, \quad j_y = -\frac{\partial\mathbf{f}}{\partial y}, \quad j_z = -\frac{\partial\mathbf{f}}{\partial z} + uB. \quad (24)$$

Values of the electric current at cell centres can be calculated easily in CFX using actual values of the velocity stored in arrays U and W, and the utility subroutine GRADS for calculation of the gradient of the electric potential.

Electric potential equation

The electric potential satisfies Eq. (8). The right-hand side of this equation can be written as

$$\nabla \cdot (\mathbf{v}\times\mathbf{B}) = -\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}. \quad (25)$$

The straightforward approach in introducing such a source term into the electric potential equation would involve using utility subroutine GRADV for calculation of the gradients of velocity components at the cell centre and then multiplying it by the volume of the corresponding cell. In Appendix 3, such an approach is used in user subroutine USRSRC for a fully developed flow in a duct with conducting Hartmann walls and insulating parallel walls. (Note that the source term in CFX appears at the same side of the equation as the diffusion term, therefore $-\nabla \cdot (\mathbf{v} \times \mathbf{B})$ times cell volume is actually used).

Lorentz force

The Lorentz force in a vertical magnetic field has only two components:

$$F_x = -j_z B, \quad F_z = j_x B \quad (26)$$

with the electric current components determined from Eq. (24). Again, the straightforward approach illustrated in Appendix 3 is to evaluate the electric current at the cell centres as described above and to introduce it into the momentum equations.

Integral approach

However, in some cases the straightforward approach introduces considerable error. It happens especially on highly non-uniform or non-orthogonal grids. In such cases, more appropriate integration of the source term and the body force over each control volume should be used developed in [20]. It is briefly described below for orthogonal grids (from [20]).

Electric potential equation

The source term in the electric potential equation (8) is $\nabla \cdot (\mathbf{v} \times \mathbf{B})$. To specify the source term for the CFX solver, it has to be integrated over a control volume. Then

$$S \cdot V = \int_V \nabla \cdot (\mathbf{v} \times \mathbf{B}) dV = \int_A (\mathbf{v} \times \mathbf{B}) \cdot \hat{\mathbf{n}} dA. \quad (27)$$

Here V is the volume of the cell. Thus the source term can be approximated as

$$S = \frac{1}{V} \sum_f A_f (\mathbf{v} \times \mathbf{B}) \cdot \hat{\mathbf{n}}, \quad (28)$$

where f denotes the 6 faces of the control volume, A_f is the area of the corresponding face, and $\hat{\mathbf{n}}$ is the outward normal unit vector of the face. For an orthogonal grid,

$$S = \frac{1}{V} \left[A_x ((\mathbf{v} \times \mathbf{B})_{x+} - (\mathbf{v} \times \mathbf{B})_{x-}) + A_y ((\mathbf{v} \times \mathbf{B})_{y+} - (\mathbf{v} \times \mathbf{B})_{y-}) \right. \\ \left. + A_z ((\mathbf{v} \times \mathbf{B})_{z+} - (\mathbf{v} \times \mathbf{B})_{z-}) \right]. \quad (28)$$

In the above, A_x , A_y and A_z are the areas of the cell faces in x , y and z directions, respectively; $(\mathbf{v} \times \mathbf{B})_{x\pm}$, $(\mathbf{v} \times \mathbf{B})_{y\pm}$ and $(\mathbf{v} \times \mathbf{B})_{z\pm}$ are values of the vector at the faces in corresponding co-ordinate directions, + standing for face with higher x (y , z) and - standing for lower one. These values are interpolated from values at the adjacent cell centres using weight factors available in CFX.

Lorentz force and electric current density

The Lorentz force in the momentum equation (7) involves the electric current. For the problem to be consistent, the second term $\mathbf{v} \times \mathbf{B}$ must be calculated in the way similar to the source term in the potential equation. To integrate this term over a control volume, one needs its values at the cell faces. These are interpolated using the weight factors available in CFX.

Then term $\mathbf{v} \times \mathbf{B}$ at the cell centre is evaluated as

$$(\mathbf{v} \times \mathbf{B}) = \frac{1}{2} \sum_f (\mathbf{v} \times \mathbf{B})_f \cdot \hat{\mathbf{n}} / A_f. \quad (30)$$

Values of the vector $\mathbf{v} \times \mathbf{B}$ at the cell faces are obtained in the same way as in Eq. (28).

The second term of the electric current, $-\nabla f$, can be calculated directly using the subroutines provided by CFX for gradient calculation. It is modified slightly to include the case when the Neumann boundary conditions are required at the inlets/outlets. It also gives incorrect values near the walls, which should be recalculated correspondingly.

Unlike the straightforward approach discussed in the previous section, here the electric current conservation is enforced, which gives more accurate results.

5.2.7 Command file options

Command files are used in CFX for specifying fluid properties, physical models, boundary conditions and solver data. It is a text file with the following structure:

```
>>CFX4
(Subcommands and Keywords)
>>MODEL TOPOLOGY
(Subcommands and Keywords)
>>MODEL DATA
(Subcommands and Keywords)
>>SOLVER DATA
(Subcommands and Keywords)
>>CREATE GRID
(Subcommands and Keywords)
>>MODEL BOUNDARY CONDITIONS
(Subcommands and Keywords)
>>OUTPUT OPTIONS
(Subcommands and Keywords)
>>STOP
```

Hereafter a brief review of options relevant to the problems considered in this report is given.

Command CFX4

This command specifies the major flow options, the list of the additional Fortran routines used and names of variables and phases. The following subcommands are available:

>> Options

The following options are used in this study:

TWO DIMENSIONS / THREE DIMENSIONS

Specifies whether two-dimensional or three-dimensional model is used. If two dimensions option is chosen, then the grid must have only one grid step in the z-direction.

NUMBER OF PHASES

Specifies number of phases used. For a duct flow, only one phase (liquid metal) is used. For a free-surface flows, two phases are used - "air" and liquid metal.

USER SCALAR EQUATIONS

Additional scalars may be introduced in CFX. These satisfy a general diffusion/convection equation as discussed above. However, if the name of user scalar (defined as described below) starts with USRD, it is a dummy scalar and no equation is solved for it. Dummy scalars can be used for calculating additional variables, such as electric currents in three-dimensional flows, exact solutions in order to check whether the flow is fully developed at the entrance of the duct etc.

RECTANGULAR GRID / BODY FITTED GRID

Only rectangular grid has been used in this study.

LAMINAR FLOW / TURBULENT FLOW / TURBULENT FLOW WITH WILCOX MODEL

Only laminar flows have been considered in this study.

ISOTHERMAL FLOW / HEAT TRANSFER

Heat transfer option is switched on if the temperature equation needs to be solved. Only isothermal flows have been considered in this report.

COMPRESSIBLE FLOW / INCOMPRESSIBLE FLOW

Only incompressible flows have been considered in this report.

BUOYANT FLOW

Buoyancy is switched on if the flow is non-isothermal or if the gravity force should be taken into account. In this case the gravity vector and the reference density should be defined using command BUOYANCY PARAMETERS.

TRANSIENT FLOW / STEADY STATE

The transient flow option is switched on when a time-dependent flow is considered. It can also be switched on for complex steady flows if convergence difficulties are experienced. We use the transient flow option for two-phase (free surface) flows. In duct flows, the flow is considered steady.

>> User Fortran

Here the list of the CFX Fortran subroutines used in the problem is given.

>> Variable names

Here the names of the variables used in the problem can be changed. For example, standard names for user scalars are USER SCALAR N, where N is the number of the scalar. Using a special name starting with USRD turns off the solution of the transport equation for the scalar and allows one to use it to store arbitrary information. An example of using the command is:

```
U VELOCITY 'AXIAL VELOCITY'  
V VELOCITY 'VERTICAL VELOCITY'  
W VELOCITY 'AZIMUTHAL VELOCITY'  
USER SCALAR1 'USRD EXACT'  
USER SCALAR2 'USRD B'
```

>> Phase names

Using this command, one can introduce arbitrary names for the phases used in the problem. For example, in the free surface flow one can use command

```
>>PHASE NAMES
    PHASE1 'AIR '
    PHASE2 'METAL '
```

After that, names METAL and AIR can be used in all subsequent commands.

Command Model topology

This command deals with the geometry of the problem. It allows to create and to modify simple geometries, patches and grids.

For example, a single rectangular block with 250, 118 and 1 mesh divisions in the x , y and z directions, respectively, can be created by a command

```
>>CREATE BLOCK
    BLOCK NAME 'DUCT '
    BLOCK DIMENSIONS 250 118 1
```

Patches are created using command CREATE PATCH. For example, to create a two-dimensional pressure boundary at the top of the block DUCT, the following command could be used:

```
>>CREATE PATCH
    PATCH NAME 'TOP '
    BLOCK NAME 'DUCT '
    PATCH TYPE 'PRESSURE BOUNDARY '
    HIGH J
```

Here HIGH J keyword means that the patch will contain nodes of the block DUCT with the highest values of J.

When the geometry is created using the CFX-Build program, MODEL TOPOLOGY commands can be used in order to change names and types of patches. This is convenient for introducing user-defined names for patches and changing the boundary conditions (for example, at the entrance to the duct either mass flux, velocity or pressure can be given depending on the problem). The following command changes the pressure boundary defined in CFX-Build to inlet boundary. It also changes the name of the patch to ENTRANCE with is more convenient when defining the boundary conditions and using the post-processor.

```
>>MODIFY PATCH
      OLD PATCH NAME 'PRESS1 '
      NEW PATCH NAME 'ENTRANCE '
      NEW PATCH TYPE 'INLET '
```

Command Model data

This command is used to specify properties of the fluids, physical and numerical models etc.

>>*Ambient variables*

```
PHASE NAME / ALL PHASES
```

This option allows one to set the ambient variables for all phases at once or for each phase separately.

After that, the list of variables being set and the corresponding values is entered.

For example, to set the ambient medium to air, the following commands can be used:

```
>>AMBIENT VARIABLES
      PHASE NAME 'AIR '
      VOLUME FRACTION 1.0000E+00
>>AMBIENT VARIABLES
      PHASE NAME 'METAL '
      VOLUME FRACTION 0.0000E+00
```

>>Body forces

This option allows one to introduce simple body forces of type $\mathbf{F} = \mathbf{C} - R\mathbf{v}$ into the momentum equation. Here \mathbf{C} is a constant vector, and R is a diagonal matrix with constant elements; \mathbf{v} is the fluid velocity. More sophisticated body forces are defined using the user subroutine USRBF.

PATCH NAME

Specifies the name of a three-dimensional patch where the body forces will act.

PHASE NAME / ALL PHASES

This option allows one to set the body forces acting on all phases or on a particular phase. For MHD flows, the Lorentz force will be acting on the liquid metal only, since the electrical conductivity of the air is much lower than that of the metal.

BODY FORCE

This is the component of the body force independent of \mathbf{v} . In CFX the body force is represented in the following way:

$$\mathbf{F} = \mathbf{C} - R\mathbf{v} ,$$

where \mathbf{C} is the velocity-independent vector, R is the resistance constant, \mathbf{v} is the velocity. Thus, this command allows to introduce a constant body force component \mathbf{B} . If more complicated body force is used, it should be specified in subroutine USRBF.

RESISTANCE CONSTANT

Here the second component of the body force is specified, i.e. the factor in front of $-\mathbf{v}$. Again, for more complicated cases USRBF subroutine should be used instead of this command.

>>Differencing scheme

This option specifies which differencing scheme is chosen for the advection term.

ALL EQUATIONS / <variable name>

Specifies whether the differencing scheme applies to all equations or to a certain variable.

NO CONVECTION / UPWIND / HYBRID / HIGHER UPWIND / QUICK / CENTRAL / CONDIF /
CCCT / MIN-MOD / VAN LEER / SUPERBEE / NO MATRIX

This keyword chooses the corresponding differencing scheme. The "no convection" option is used, for example, for the electric potential equation where no advection takes place, or in inertialess flows for the Navier-Stokes equations.

>>Physical properties**>>Buoyancy parameters**

This command allows one to introduce gravity force. We will consider only options relevant to isothermal flows considered in this report.

PHASE NAME / ALL PHASES

Shows whether the options apply to all phases or a particular phase only.

GRAVITY VECTOR

The three components of the gravity vector are specified here. The gravity vector is the same for all phases.

>>Fluid parameters

This subcommand is for specification of fluid properties.

PHASE NAME / ALL PHASES

Shows whether the options apply to all phases or a particular phase only.

VISCOSITY

Fluid viscosity is specified by this command.

DENSITY

Fluid density is specified by this command.

>>*Multiphase parameters*>>*Multiphase models*

This command sets the multiphase models used in the simulation.

>>*Homogeneous*

This command sets all physical processes except heat transfer to be homogeneous (see description of the homogeneous model in chapter 5.2.3)

SURFACE SHARPENING ALGORITHM

This keyword specifies whether the surface sharpening algorithm is employed. It is useful in transient free-surface flows, when the interface between two fluids is sharpened by modifying the volume fraction field after each time step.

SURFACE SHARPENING LEVEL

There are four levels of surface sharpening available. The default level is level 2.

SURFACE TENSION MODEL

This keyword invokes the surface tension model. If it is not included, there is no surface tension in the problem.

SURFACE TENSION COEFFICIENT

This keyword is used to set a constant surface tension coefficient for the interface between the two phases.

WALL CONTACT ANGLE IN DEGREES

This keyword is used to set the wall contact angle as a real number between 0.0 and 180.0. This is the angle that the interface makes to the wall, which by convention, is relative to the first phase. If the wall in contact with the first phase is hydrophobic then the wall contact angle will be greater than 90° and if the wall in contact with the first phase is hydrophilic the angle will be less than 90°. The contact angle has a default of 90°.

>>*Momentum*

This command sets the model for the momentum equation to be homogeneous. It has same keyword as the >>*Homogeneous* command and therefore is not described in detail.

>>*Phase description*

PHASE NAME

Name of the phase for which the description is given.

GAS / LIQUID / SOLID

Thermodynamic phase of the fluid.

CONTINUOUS / DISPERSE

Continuity of the phase.

MODIFY EMPTY CELL VELOCITY

In flows where buoyancy and other forces act, cells with small volume fractions may attain large numerical values for velocities. The keyword **MODIFY EMPTY CELL VELOCITY** sets velocities to zero in cells with volume fractions below that specified by the keyword.

MINIMUM VOLUME FRACTION

In a multi-phase calculation, in order to avoid solving zero equations in parts of the domain where the phase is not present, the minimum value of the volume fraction for any phase is bounded below. By default, the bounding value is 1.0E-10, but this can be changed using the MINIMUM VOLUME FRACTION keyword.

>> *Scalar parameters*

>> *Diffusivities*

Specifies diffusivity coefficients in additional scalar equations. For example, in the electric potential equation (8) the coefficient in front of the diffusive term is equal to unity.

PHASE NAME / ALL PHASES

Specifies whether the diffusivity specified below applies to all phases or a certain phase.

USER SCALAR_n / ALL USER SCALARS / USER SCALARS

Specifies user scalar diffusivities. One diffusivity can be defined for a certain scalar, all user scalars, or for all scalars using a list of data.

>> *Transient parameters*

>> *Adaptive time stepping*

This option allows CFX-4 to choose the time step from within a certain range. The time step chosen will depend on how well the simulation is converging.

NUMBER OF TIME STEPS

Here an integer number is given specifying how many time steps are to be used. This includes time steps that fail because of lack of convergence.

INITIAL TIME STEP

Sets the initial time step in seconds. By default, the initial time step is taken from the restart file, if there is one.

MINIMUM TIME STEP

Allows user to set a minimum possible time step, in seconds.

MAXIMUM TIME STEP

Allows user to set a maximum possible time step, in seconds.

MULTIPLY TIME STEP BY

Sets the factor by which the time step is increased when the time step increment interval has been reached, see below.

DIVIDE TIME STEP BY

Sets the factor by which the time step is divided when a time step fails.

MINIMUM INTERVAL BETWEEN INCREMENTS

Controls how many successful time steps in a row there must be before the time step is increased.

MAXIMUM NUMBER OF CONTIGUOUS INCREMENTS

Controls the maximum number of times a time step can fail in a row before the run is stopped.

BACKWARD DIFFERENCE / CRANCK NICOLSON

This keyword invokes a fully implicit Backward Euler differencing (default) or Crank-Nicolson (central) differencing in time.

LINEAR TIME DIFFERENCING / QUADRATIC TIME DIFFERENCING

This keyword sets a first-order time-stepping method (default) or a second-order time-stepping method. It is recommended that this is not used with CRANK NICOLSON but as an alternative to it.

INITIAL TIME

This keyword used to set the initial time, in seconds. This is most often used in a restart job. The initial time is then taken from the restart file by default. Where the run is not a restart, the default initial time is zero. Using this keyword will overwrite the default.

MAXIMUM TIME

Sets the final model time, in seconds, of the transient solution. The user control over the number of time steps performed, but since the time step varies, the final solution time is not known. This keyword means that the simulation will either stop after the NUMBER OF TIME STEPS have been performed, or if the current time step is at a time which is greater than or equal to the MAXIMUM TIME.

>> *Extrapolation order*

A further advantage of adaptive time stepping is the inclusion of an extrapolation technique that improves the initial guess to the solution at each time step. This reduces the number of iterations required for convergence. The order of extrapolation is set to be first order by default, but can be changed using the >>EXTRAPOLATION ORDER command. For example, the second-order extrapolation would be recommended with second-order time differencing.

>> *Fixed time stepping*

TIME STEPS / TIME VALUES

Here the list of time steps or time values is given. For example, 10 * 0.1 means ten time steps, 0.1 seconds each. Maximum of 10000 time steps can be specified.

BACKWARD DIFFERENCE / CRANCK NICOLSON

This keyword that invokes fully implicit Backward Euler differencing (default) or Crank-Nicolson (central) differencing in time.

LINEAR TIME DIFFERENCING / QUADRATIC TIME DIFFERENCING

This keyword sets a first-order time-stepping method (default) or a second-order time-stepping method. It is recommended that this is not used with CRANK NICOLSON but as an alternative to it.

INITIAL TIME

This keyword used to set the initial time, in seconds. This is most often used in a restart job. The initial time is then taken from the restart file by default. Where the run is not a restart, the default initial time is zero. Using this keyword will overwrite the default.

>>**Set initial guess**

>>*Set constant guess*

PHASE NAME / ALL PHASES

Specifies whether the initial guess is set for all phases or for a certain phase only.

<variable name>

Specifies variables (u, v, w, pressure, volume fraction, enthalpy, user scalar n) and their values used as the initial guess.

>>Sources

This command allows one to specify extra source terms for any of the transport equations (u, v, w, pressure, volume fraction, enthalpy, user scalar n). Since these equations are obtained by integrating the appropriate conservation laws over a control volume the source terms have the form of the product of the source term and the volume of the control volume. Only constant source terms can be specified here. For more complicated cases user subroutine USRSRC should be used.

The source term for variable f is split into two parts as follows:

$$S = S_u + S_p f. \quad (31)$$

Thus, two numbers are specified for each source term.

PATCH NAME

Name of the two- or three-dimensional patches where the source term appears. In case of a two-dimensional patch, the source term is given per unit area of the corresponding cell.

PHASE NAME / ALL PHASES

Specifies whether the source term is set for all phases or for a certain phase only.

TOTAL / PER UNIT VOLUME AND MASS OF PHASE / PER UNIT MASS OF PHASE / PER UNIT VOLUME / PER UNIT MASS / PER UNIT AREA AND MASS OF PHASE / PER UNIT AREA OF PHASE / PER UNIT AREA

This keyword specifies the way the source term is defined. Details can be found in CFX user guide.

>>Title

Here a title can be given to the problem that will appear in the output file.

Command Solver data

>> False timesteps

This command allows improving problem convergence by using the false time steps method. It can be used in transient problems together with real time stepping. It can also be combined with underrelaxation method.

PHASE NAME / ALL PHASES

Specifies whether false time steps apply to all phases or a particular phase.

ALL EQUATIONS / *<variable name>*

Specifies which equations should be integrated using false time steps and the corresponding values of the false time steps.

>> Courant number false time steps

One of the difficulties of using false time steps is that knowledge of the time scales of the problem is needed: Courant number false time steps avoid that difficulty by calculating the time scale from the grid size and the local velocity field.

Different false time steps can be chosen for different equations when using Courant number false time steps. This is achieved by setting a multiple of the local Courant number time scale for each equation.

PHASE NAME / ALL PHASES

Specifies whether the following options will apply to all phases or one particular phase.

ALL EQUATIONS / *<variable name>*

By this keyword(s) the multiple of the local Courant number time scale is given for all equations or a list of variables.

IGNORE SPEED OF SOUND

By default for fully compressible flows, the code uses the sum of velocity and the local speed of sound when calculating the Courant number time scale. This can be changed to just use the velocity, as for other flow types, by using this keyword.

INITIAL FALSE TIMESTEP

When the speed of sound is included, the Courant number time scale depends upon the reciprocal of velocity. For the standard initial guess of no flow, this can lead to problems on the first iteration. There is therefore a keyword INITIAL FALSE TIMESTEP for setting a suitable value to be used just on the first iteration.

>> Equation solvers

The set of linearised difference equations for a particular variable, one equation for each control volume in the flow, is passed to a simultaneous linear equation solver, which uses an iterative solution method. The alternative methods, available for this purpose, are specified within this command.

PHASE NAME / ALL PHASES

Specifies whether the following options will apply to all phases or one particular phase.

<variable name>

Specifies the variable for which the method is set.

LINE SOLVER / STONE / BLOCK STONE / ICCG / AMG / GENERAL AMG

These keywords set one of the following iteration methods: line relaxation, full field Stone's method, block Stone's method, preconditioned conjugate gradients, algebraic Multi-grid or general version of Algebraic Multi-grid. Usually the default method chosen by CFX works satisfactory, and there is no need to change the solver.

>> Pressure correction

SIMPLE / SIMPLEC / PISO / NON ITERATIVE PISO

By default, SIMPLE algorithm (Semi-Implicit Method for Pressure-Linked Equations) is used for updating pressure and correcting velocity components in order to ensure mass conservation. SIMPLEC is a modification of SIMPLE which differs in its derivation of a simplified momentum equation. In the so-called PISO algorithm, a second pressure-correction equation is solved in order to improve the solution of the momentum equations while maintaining continuity. SIMPLE method has been used in this study.

>> Program control

MINIMUM NUMBER OF ITERATIONS

Specifies minimum number of iterations performed by solver before the convergence testing is done.

MAXIMUM NUMBER OF ITERATIONS

Specifies maximum number of iterations performed by solver.

MASS SOURCE TOLERANCE

Specifies the convergence criterion in terms of mass residual. To use other convergence criteria, subroutine USRCVG should be used.

>> Reduction factors

On each "outer" iteration step, the set of linearised difference equations for a particular variable is solved using an iterative solution method. An exact solution is not required because this is just one step in the non-linear "outer" iteration. The computational effort in obtaining a reasonable solution to the set of equations is controlled using command >> Reduction factors and >>Sweeps information.

The residual in a particular cell is the amount by which the linear equation there is not satisfied. Residual reduction factor is the amount by which the residual should reduce compared to its initial value.

PHASE NAME / ALL PHASES

Specifies whether the reduction factors are given for all phases or a particular phase.

<variable name> <factor>

Specifies a variable and a corresponding reduction factor. By default, the reduction factor is 0.1 for pressure and 0.25 for all other variables.

>> **Sweeps information**

MAXIMUM NUMBER

Specifies maximum number of inner iterations used for solving linearised equations on each "outer" iteration step.

MINIMUM NUMBER

Specifies minimum number of inner iterations used for solving linearised equations on each "outer" iteration step.

>> **Transient control**

The MASS SOURCE TOLERANCE is not used for transient flows using the adaptive time stepping option. In this case any variable can be chosen to test for convergence, not just the mass source residual. Convergence is obtained when the residual for the tested variable for the first phase satisfies the condition $RES < MAX (RESMIN, MIN(RESMAX, RES5/REDUC))$, where RES is the residual of the corresponding variable, RESMIN is the minimum required residual value, RESMAX is the maximum allowed residual value, REDUC is the residual reduction factor and RES5 is the value of the variable's residual computed on the fifth iteration.

>> Control parameters

MINIMUM RESIDUAL VALUE

Minimum required residual value for the variable specified by subcommand CONVERGENCE TESTING ON VARIABLE.

MAXIMUM RESIDUAL VALUE

Maximum allowed residual value for the variable specified by subcommand CONVERGENCE TESTING ON VARIABLE.

REDUCTION FACTOR

Residual reduction factor.

DIVERGENCE RATIO

The time step solution is assumed to have diverged if any variable's residual increases to be greater than the specified DIVERGENCE RATIO multiplied by its minimum residual value to that point. The divergence test is only performed after 5 outer iterations in order to avoid possible zero residuals that might occur initially.

>>Convergence testing on variable

<variable name>

Specifies the variable to which the convergence testing will apply.

>> Under relaxation factors

PHASE NAME / ALL PHASES

Specifies whether the command applies to all phases or a particular phase only.

ALL EQUATIONS

This keyword is used if the same underrelaxation factor is applied to all equations.

<variable name> <under-relaxation factor>

This keyword specifies under-relaxation factors for particular variables. It also may be used for under-relaxation of body forces in momentum equations (names BFX, BFY, BFZ and BPX, BPY, BPZ are used for velocity-independent and velocity-dependent components of the body force - see Sec. 5.2.8 for definitions).

Command Create grid

In most of the problems presented in this study both geometry and computational mesh are created using pre-processing tools available in CFX. However, some simple geometries are created in command file using command >>Model Topology. In this case, computational mesh should be created using command Create Grid. Only simple rectangular grids are used in this study.

>> Simple grid

BLOCK NAME

This keyword is used to specify block name on which the grid is being created.

X START

The starting value of x co-ordinate.

Y START

The starting value of y co-ordinate.

Z START

The starting value of z co-ordinate.

DX

Grid increment in x -direction.

DY

Grid increment in y -direction.

DZ

Grid increment in z -direction.

X

A list of cell vertices in x direction (can be used instead of keywords X/Y/Z START with DX/DY/DZ).

Y

A list of cell vertices in y direction (can be used instead of keywords X/Y/Z START with DX/DY/DZ).

Z

A list of cell vertices in z direction (can be used instead of keywords X/Y/Z START with DX/DY/DZ).

For example, the following command will create simple mesh with x ranging from 0 to 3 with step 0.1, y ranging from -0.5 to 0.5 with step 0.1 and z ranging from -1 to 1 with step 0.2.

```
>>SIMPLE GRID
BLOCK NAME 'BLOCK 1'
X START 0.0
Y START -0.5
Z START -1.0
DX 30 * 0.1
DY 10 * 0.1
DZ 10 * 0.2
END
```

The same mesh can also be created in the following way:

```
>>SIMPLE GRID
BLOCK NAME 'BLOCK 1'
X 0.0 TO 3.0 BY 0.1
Y -0.5 TO 0.5 BY 0.1
Z -1.0 TO 1.0 BY 0.2
END
```

Command Model boundary conditions

This command is used to specify boundary conditions. If non-constant boundary conditions are needed, these are specified in user subroutine USRBCS.

>> Inlet boundaries

PATCH NAME

This keyword specifies on which patch of type INLET boundary conditions are set.

PHASE NAME / ALL PHASES

Specifies whether same inlet boundary conditions apply to all phases, or only to a particular phase.

<variable name> <value>

Sets values for flow variables (velocities, pressure, user scalars, temperature etc.).

>> Mass flow boundaries

>> *Flux*

PHASE NAME / ALL PHASES

Specifies whether fluxes specified apply to all phases or a particular phase.

FLUXES

This keyword sets fluxes for all mass flow patches. If several mass flow patch groups are created, then several values of mass flux must be specified - one for each group of patches.

MASS FLOW SPECIFIED / FRACTIONAL MASS FLOW SPECIFIED

Specifies whether absolute value of mass flux is given, or fraction of overall mass flux into domain.

>> *Inflow variables*

This command allows user to set constant values of flow variables on mass flow boundaries.

PHASE NAME / ALL PHASES

Specifies whether command applies to all phases or a particular phase.

PATCH NAME

Name of the mass flow patch where the flow variables are specified.

VOLUME FRACTION

Volume fraction can be specified on a mass flow boundary (for example, when a jet enters the domain, negative mass flux and volume fraction = 1 can be set at the corresponding patch).

<variable name> <value>

Allows user to set values of flow variables (velocities, user scalars, temperature etc.) at a mass flow boundary. However, values for the velocity components are only used if the flow is into the domain.

>> Pressure boundaries

This command allows user to set constant values of flow variables on pressure boundaries.

PHASE NAME / ALL PHASES

Specifies whether command applies to all phases or a particular phase.

PATCH NAME

Name of the pressure patch where the flow variables are specified.

PRESSURE

Pressure is fixed on a pressure boundary. Two pressure boundaries, for example, can be used to model flow with given pressure drop.

VOLUME FRACTION

Volume fraction can be specified on a pressure boundary.

<variable name> <value>

Allows user to set values of some flow variables (user scalars, temperature etc.) at a mass flow boundary. Neumann boundary conditions, i.e. zero normal gradients, are imposed on velocity.

>> Wall boundaries

PHASE NAME / ALL PHASES

Specifies whether the command applies to all phases or one particular phase.

PATCH NAME

Name of the patch of type WALL where the boundary conditions are being set.

U (V, W) VELOCITY

Values of the three components of the velocity at the wall.

TAUX (TAUY, TAUZ)

Tangential stress at the wall.

TEMPERATURE / HEAT FLUX / TEMPERATURE ABC

Value of temperature or heat flux or general temperature boundary conditions at the wall.

USER SCALAR n / USER SCALAR n FLUX / USER SCALAR n ABC

Value of user scalar or its flux or general boundary conditions at the wall.

Command Output options

This command is used to control the way output data is produced.

By default, CFX-4 will produce an unformatted, single precision dump file at the end of the simulation. This will contain adequate information to carry out a restart from the current run. It is possible to increase the information stored in the dump file, to make it a formatted file and to change the precision of the dump file.

>> Dump file format

UNFORMATTED / FORMATTED

It is sometimes necessary to create a formatted dump file if the CFX-4 simulation is to be carried out on a different machine from the post-processing. In this case it is likely that the unformatted file will not be readable on the second machine, so a formatted file is necessary.

NUMBER OF SIGNIFICANT FIGURES

Sets the number of significant places in the output data if required.

SINGLE PRECISION / DOUBLE PRECISION

Changes the precision of output data.

>> Dump file options

At the end of the simulation CFX-4 always dumps out everything that is necessary to do a restart. It is possible to dump out any selection of variables and other data to disk at any other stage in the simulation.

PHASE NAME / ALL PHASES

Specifies whether data for all phases or a particular phase should be written into the dump file.

ITERATION

Specifies iteration number on which data should be written.

TIME STEP / EACH TIME STEP / TIME INTERVAL / TIME STEP INTERVAL

Specifies time step (time) on which data should be written. Instead, time (step) interval can be specified.

INITIAL GUESS

Specifies that the initial guess should be included into the dump file (for transient flows).

FINAL SOLUTION

Specifies that the final solution should be included into the dump file (for transient flows).

ALL VARIABLES / ALL REAL DATA

All real data consists of all the variables and properties as well as all the other real data that is required for a smooth restart to the flow simulation. Alternatively, all variables option can be used if values of flow variables are needed only.

GEOMETRY DATA / NO GEOMETRY DATA

GEOMETRY DATA consists of the grid and topological information. The geometry data is normally written to the first data group in the file, so that option NO GEOMETRY DATA can be used in order to save space. Otherwise it is included on each time step in transient calculations.

5.2.8 Additional subroutines used

Additional Fortran subroutines are used when an option cannot be specified via command file (for example, non-constant boundary conditions). All user subroutines used in a problem are written into a file M*.F (for example, m01.f). Flag IUSED should be set to 1 in each subroutine. Every subroutine should also be listed in the command file under command >>USER FORTRAN. Hereafter only features relevant to this study are listed. User has access to the following variables:

- U, V, W, P, VFRAC, T, SCAL

Contain values of velocity components, pressure, volume fraction, temperature and user scalars. All these arrays have size (NNODE, NPHASE), where NNODE is the number of all nodes, including boundary nodes and dummy nodes; NPHASE is the number of phases. The size of the scalar array is (NNODE, NPHASE, NSCAL), where NSCAL is number of user scalars.

- XP, YP, ZP, VOL, AREA, WFACT

Contain geometry information: co-ordinated of nodes, volumes of cells, areas of cell faces, weight factors.

Other data is available such as topological information etc. There are also work arrays WORK, IWORK and CWORK that allow user to reserve workspace for additional data and to pass data between subroutines.

Every cell in the computational domain can be referred to either by naming the block on which the cell resides and giving local co-ordinates (I, J, K) of the cell, or by its internal, 1D, address, which lies between 1 and NNODE (NNODE is the number of all internal grid nodes - NCELL , including dummy cells around each block, plus the number of boundary nodes placed in the centre of each patch - NBDRY).

Utility routines are available from all user subroutines. They perform such tasks as finding addresses of patches, finding variable numbers etc. The most commonly used utility routines are described below.

All user subroutines contain clearly specified user areas where the modifications can be done.

The following subroutines have been used in this study: USRBCS, USRBF, USRCVG, USRINT, USRSRC, USRTRN. A brief description of each subroutine is given below.

Utility routines

GETSCA

This subroutine is used to find the number of a user scalar within the SCAL array. The alias name (CHARACTER*24) of the scalar is passed to the subroutine, and the number of the corresponding scalar is returned.

GETVAR

In most cases, the name of the variable is passed to subroutines (the CHARACTER*6 name such as 'W ', 'VFRAC ' etc.). However, sometimes the number of variable is required. Subroutine GETVAR returns the number of variable if its name is given.

GRADS

GRADS may be used to compute the physical space gradients of a scalar variable. It creates a temporary array GRAD(NCELL,3) where values of three components of the gradient are stored for all internal cells of the computational domain.

GRADV

GRADV may be used to compute the physical space gradients of velocity. It creates three temporary arrays UGRAD, VGRAD and WGRAD of size (NCELL,3) where values of three components of the gradient for each velocity component are stored for all internal cells of the computational domain.

IPALL

This subroutine allows user to get addresses of several blocks or patches. User specifies the name of the block or patch, whether it is a block or a patch and whether the addresses of cell centres or vertices are required. In addition, the patch type has to be specified. The patch type and/or the patch or block name may be specified as '*' which means all the patch types and/or all the names. An array IPT of size NPT is returned, which contains 1D addresses of patch (block) cells.

IPREC

IPREC is used to return 1D addresses for all the centres or vertices in a rectangular group such as a block or patch. In contrast with IPALL, only one and only rectangular block or patch can be processed. The subroutine returns dimensions ILEN, JLEN and KLEN of the block/patch, and an array of corresponding 1D addresses.

Subroutine USRBCS

This subroutine enables user to specify more complicated boundary conditions. Simple boundary conditions can be set using the >>MODEL BOUNDARY CONDITIONS command.

User must set the flag IUBCSF to state whether the boundary conditions are to vary with iteration, time, or time and iteration (IUBCSF = 1, 2 or 3 respectively). Array VARBCS of size (NVAR, NPHASE, NCELL+1:NNODE) can be changed in order to set values of flow variables (NVAR is the total number of variables). User can also set coefficients in boundary conditions of general type, volume fractions, mass fluxes etc.

Two sample user FORTRAN files are given in the Appendix. Subroutine USRBCS is used to set Hartmann profile and volume fraction at the inlet for the two-dimensional MHD jet.

Subroutine USRBF

This subroutine allows user to add body force to the momentum equations. Simple body forces of this type can be included using the >>BODY FORCES command. In this study subroutine USRBF is used for including Lorentz force into momentum equations.

In order to enable the code to linearise the body force source term correctly, \mathbf{F} is expressed in the form in CFX:

$$\mathbf{F} = \mathbf{C} - R\mathbf{v} .$$

In the above \mathbf{C} is a vector, and R is a diagonal matrix; \mathbf{v} is the fluid velocity. Arrays BX, BY and BZ of size NCELL are filled with the three components of the vector \mathbf{C} for each internal cell of the computational domain. Similarly, arrays BPX, BPY and BPZ of the same size contain factors R .

A sample user FORTRAN file for two-dimensional MHD jet is given in the Appendix 1. Subroutine USRBF is used to include body force $F = -(uB + E)B$ into the momentum equation. In Appendix 2 the body force is three-dimensional as given by Eq. (5).

Subroutine USRCVG

This is a general routine as it is called at the end of each iteration. The routine allows user to set custom convergence criteria, or to modify solution parameters during the iteration history. Simple convergence criteria can be set using the command `>> SOLVER DATA`.

If the problem satisfies convergence criteria, flag `LCONVG` should be set to `.TRUE`.

In the example given in the Appendix 2 subroutine `USRCVG` sets `LCONVG = .TRUE`. if ratio of the residual to the maximum value of velocity, volume fraction and pressure is less than 10^{-7} .

Subroutine USRINT

This subroutine is used to overwrite the default initial conditions. Simple initial guess can be set using command `>>INITIAL GUESS` in the command file. Then `USRINT` can be used to overwrite or add initial conditions. Values of corresponding variables are changed directly in arrays `U`, `V`, `W` etc.

In the sample Fortran file given in the Appendix 2 subroutine `USRINT` is used to set the initial (straight) shape of the jet and velocity equal to Hartmann profile everywhere inside the jet.

Subroutine USRSRC

This subroutine allows user to intervene and change the equations; in particular to add source terms into convection-diffusion equations. In this study subroutine `USRSRC` is used for specifying the source term in the electric potential equation in three-dimensional problems. It can also be used to include body forces into the momentum equations, instead of `USRBF`. In contrast to `USRBF`, the source term in `USRSRC` must be integrated over the control volume (in simple cases, the original source per volume must be multiplied by the volume of the cell).

In the sample of three-dimensional problem in Appendix 3, the source term $S = \nabla \cdot (\mathbf{v} \times \mathbf{B})$ is added to the electric potential equation.

Subroutine USRTRN

Subroutine USRTRN is called after the initial guess and at the end of each time step. The routine can be used to monitor the calculation, or to produce special output for each time step. It may also be used at the end of the job to calculate additional quantities from the basic solution variables. In this study subroutine USRTRN is used for calculation of electric currents in three-dimensional problems and for calculation of additional data such as asymptotic or exact solutions of corresponding problems.

In the two-dimensional problem in Appendix 2 subroutine USRTRN is used to assign values of the non-uniform magnetic field to a user scalar. In the three-dimensional problem in Appendix 3 USRTRN is used to calculate electric currents and the exact Hunt solution.

6. Benchmark problems

The implementation of the CFX code for MHD flows developed here has been used to model various flows described below. Comparison with known exact and asymptotic solutions is given where possible in order to validate the implementation.

6.1 Duct flows

Fully developed flow in an arbitrary duct cross section can be computed relatively easily with CFX using mass flow- and pressure boundaries. The idea is to use a mass flow boundary at the inlet with the flow rate given and a pressure boundary with constant pressure at the outlet. If the pressure drop is given instead of the flow rate, two pressure boundaries are defined at the inlet and the outlet with fixed pressure values.

The grid in the direction coincident with the channel axis need only contain the minimum of two cells in order for the pressure drop to be correctly predicted.

6.1.1 Shercliff solution

Fully developed flow in a square channel with electrically insulating walls is used as the first sample problems. It is compared with the analytical solution given by Shercliff [24].

Both the flow geometry and the co-ordinate system are shown in Figure 3 while the grid used is shown in Figure 4. The grid is non-uniform with grid points clustered at the walls in order to resolve the Hartmann- and parallel-layers.

The results of calculations for a square duct are shown in Figure 5 - Figure 10. Excellent agreement has been achieved for both $Ha = 100$ and $Ha = 200$. Modelling the flow for $Ha = 200$ requires better resolution of the layers to achieve the same accuracy.

6.1.2 Hunt solution

For a duct with electrically insulating walls the boundary layers do not carry a significant part of the flow. However, in many cases parallel layers do carry high-velocity jets. Thus, it is important to verify the code for such flow geometries. In order to ensure that the code implementation can be used in modelling such flows with high velocities in thin boundary layers, flow in a duct with perfectly conducting Hartmann walls and perfectly conducting/insulating parallel walls has been considered (see Figure 11). The exact solutions for such flows, and for some cases of thin conducting walls, have been obtained by Hunt [25].

Since we were interested only in comparison with analytical results, it was more convenient to normalise the flow using constant pressure gradient $dp/dx = -1$ instead of a fixed flow rate.

Perfectly conducting walls

When all walls of the duct are perfectly conducting, the overvelocities in the side layers are not very high (Figure 12). The flow is almost constant in the middle of the duct with weak jets near the sidewalls. The electric potential is close to zero in the core region (Figure 13), also with peaks near the sidewalls. For $Ha = 100$ comparison with the exact solution (plotted using stars) shows very good agreement for both axial velocity and the potential (Figure 12 and Figure 13).

Perfectly conducting Hartmann walls, electrically insulated parallel walls

When the sidewalls are electrically insulating, the electric current induced in the core must vanish near the sidewalls (i. e. normal derivative of the electric potential at the wall vanishes, see Figure 14 and Figure 16). Thus the potential jump of order 1 is induced across the thin boundary layer and as a result velocity jets of order $Ha^{1/2}$ appear (see Figure 15 and Figure 17). For $Ha = 100$ (Figure 14, Figure 15) agreement with the analytical results is perfect. However, for $Ha = 200$ (Figure 16, Figure 17) some minor

difference can be noticed. It can be explained by the fact that better resolution of boundary layers is generally required for higher Hartmann numbers. Therefore, higher aspect ratio of cells in computational domain is used. Current formulation of the body force and the source term in the electric potential equation assumes that these values are constant across the cell. However, for more precise calculations these values should result from integration of corresponding equations and should use variable values on the faces of each cell (see [20]). High non-uniformity of the grid makes this inaccuracy more important, and as a result lower precision can be expected for high Hartmann numbers.

6.1.3 Flow in an expansion

As a next step we consider the two-dimensional flow in a sudden duct expansion (Figure 18). This benchmark problem is an important test case because there are sharp inner corners, which may affect the accuracy of the numerical solution. A strong, uniform, transverse magnetic field is applied in y -direction. With velocity scale $v_0 = Q/a$ (average velocity in the duct region), where Q is the flow rate and a is the length scale, governing Eqs. (7)-(9) become

$$N^{-1} \left[\frac{\mathcal{I}\mathbf{v}}{\mathcal{I}t} + (\mathbf{v} \cdot \nabla)\mathbf{v} \right] = -\nabla p + Ha^{-2} \nabla^2 \mathbf{v} - u\hat{\mathbf{x}}, \quad (38)$$

$$\nabla \cdot \mathbf{v} = 0. \quad (39)$$

In the above, $\mathbf{v} = (u(x, y), v(x, y))$ is the two-dimensional velocity field. The electric field is assumed to be zero, which corresponds to perfectly conducting, short-circuited sidewalls.

In [26] the nature of the flow in the boundary layer formed at the junction $x = 0$ in a strong magnetic field and the effect of inertia has been studied with asymptotic and numerical methods. Here we compare the results with these in [26] and perform calculations for lower values of the interaction parameter.

Inertialess flow

Consider first a symmetric 1:2 expansion. When no inertia is present, the fluid tends to flow from narrow a duct into the wider one in the shortest possible way. It is seen in Figure 19 that no separating zone is present, as expected, and the streamlines follow the form of the expansion. Upstream, the flow is fully developed with classical Hartmann profile. Near the junction, the flat Hartmann profile of the axial velocity component transforms into M-shaped profile in the direction of the field (Figure 20). This is a different effect from that causing jets in the parallel layers. Here high velocities near the points of expansion are caused by the necessity of fast flow redistribution at the junction and additional inflow into the top and the bottom parts of the wider duct. Perfect agreement between the CFX results and those in [26] have been achieved. Comparison for axial velocity component is shown in Figure 20.

Figure 21 and Figure 22 show that the flow is fully developed except for the immediate vicinity of the junction, where a parallel layer is present [26]. The graph of pressure at the top wall of the wider duct (Figure 23) also shows that apart from a very thin boundary layer at $x = 0$ the pressure is almost a straight line from the point of the expansion.

Inertial flow

Now let us include inertial effects into the problem. Asymptotic analysis [26] shows that when $N \ll Ha^{3/2}$ a layer of thickness $N^{-1/3}$ is formed at the junction where the electromagnetic force is balanced by the inertial effects. Since it still does not ensure that the non-slip conditions are satisfied, an additional, viscous, sublayer is formed at the wall $x = 0$ where the balance of inertial and viscous forces takes place.

For low values of N flow separation occurs. For example for $N = 1$ streamlines (Figure 24) show that inertia prevents the fluid from following the duct shape as in the inertialess case. It needs larger distance in x -direction to fill the whole duct. As a result, stagnant zones are formed near the corners $x = 0, y = \pm 1$ where the viscous forces become important. Consequently, less fluid flows into the "shoulders" of the expansion at $x = 0$,

and the M-shaped profile is less pronounced at the junction (Figure 25) as compared to the inertialess case.

Figure 26 shows pressure distribution along the top of the wider duct. Pressure is nearly constant near the corner where the stagnant zone is present (Figure 26). This follows by rising (adverse pressure gradient, which is the cause of separation), and then falling pressure. Even though the value of N is low, from Figure 26 and Figure 27 follows that it takes only approximately 0.7 characteristic lengths for pressure gradient to reach its fully developed value. Thus a comparatively weak field is sufficient to suppress separation in a two-dimensional duct expansion.

Flow in an asymmetric duct

Similar results hold for an asymmetric duct with expansion at one side of the duct (Figure 28). The graphs for an asymmetric 1:2 expansion are shown in Figure 29-Figure 31. A stagnant zone is formed near the upper corner (Figure 29). Qualitatively, the flow behaves in the same way as that in a symmetric expansion cut along the symmetry line $y = 0$. Thus already for $N = 1$ the so-called Coanda effect is suppressed ([27], [28], [29]).

6.1.4 Flow in a square duct in a non-uniform transverse magnetic field

As a test problem for the three-dimensional flows consider a steady, three-dimensional flow of a viscous, electrically conducting, incompressible fluid in a straight, insulating, square duct in the z -direction (Figure 32). A strong transverse magnetic field $\mathbf{B} = B(z)\hat{\mathbf{y}}$ is applied in y -direction. We use a step-like field

$$B(z) = 0.5(B_d + 1) + 0.5(B_d - 1) \tanh \mathbf{g}z, \quad (51)$$

where $\mathbf{g} = 2$ is the field gradient and $B_d = 0.5$ is the magnitude of the uniform magnetic field downstream. The profile of the magnetic field is shown in Figure 33. The flow is supposed to be inertialess. Such a flow has been considered in [30] using asymptotic and numerical methods.

The flow is governed by Eqs. (7)-(9) without the gravity term ($\mathbf{d} = 0$). The average duct velocity is chosen as the velocity scale, so that the flow rate is equal to 4.

Boundary conditions are the non-slip condition

$$\mathbf{v} = 0 \text{ at } n = 0, \quad (52)$$

and the electrically insulated walls condition

$$\frac{\partial \mathbf{f}}{\partial n} = 0 \text{ at } n = 0, \quad (53)$$

where n is the co-ordinate normal to a wall.

Far upstream and far downstream the flow is fully developed, i. e.

$$\frac{\partial \mathbf{f}}{\partial z} = 0, \quad \frac{\partial p}{\partial x} = 0 \text{ as } z \rightarrow \pm \infty. \quad (54)$$

Computational model

The length of the computational domain in z -direction should be sufficiently high to enable flow to develop. On the other hand, the cost of computation increases dramatically with increasing number of mesh points, while high aspect ratio of the cell lengths affects the precision and the convergence speed of the problem. Therefore, the length of the duct sufficient for the flow to develop far from the non-uniform field region has been estimated using several preliminary runs with low resolution. The results show that for $Ha = 200$ a duct of length equal to 16 length scales is sufficiently long for the pressure and potential to satisfy conditions (54) and to reach their fully developed values.

False time stepping has been used to reach convergent solution. False time steps equal to 0.1 for the three components of velocity and 1.0 for electric potential and pressure have been used at the beginning of the run, and then increased as the residual decreased. In order to improve convergence of the inner iteration, the reduction factor for the electric

potential has been set to 0.01, and maximum number of inner iterations has been set to 50 for the electric potential and to 100 for pressure.

Non-uniform grid has been used in all three directions. It ensures sufficient resolution of the boundary layers in x and y directions, and reduces number of the mesh points in the z -direction by clustering mesh points near the region of changes of the field at $z = 0$.

The development of the axial velocity profile is shown in Figure 34 and Figure 35. The flow profile is close to being fully developed already at $|z| = 2$ (Shercliff's flow, see [24]). In the region $|z| < 3$ three-dimensional currents circulate, which push the fluid to the sidewalls $|x| = 1$, and the M-shaped profile is formed (Figure 34 (c)-(g), Figure 35). The streamlines in the centreplane of the duct $y = 0$ are shown in Figure 36. The boundary for the three-dimensional currents may be detected from the graph for pressure (Figure 37). These currents cause transverse pressure difference in a duct cross-section. CFX is a very convenient tool for visualising the paths of the three-dimensional currents.

In various three-dimensional flows the currents are usually shown either schematically or as a projection onto the plane transverse to the magnetic field (see e. g. [5], [31], [32]). the actual paths of three-dimensional currents in the present flow for $y > 0$ are shown in Figure 38. It is seen that the current paths are very complex and involve the core, and the Hartmann- and parallel layers.

Paths shown in blue lines at $z = -3$ (loops 1) are in the fully developed flow region. The current flows in the cross-section $z = const$, being induced in the core, flowing in the parallel layers, and returning in the Hartmann layers. Closer to $z = 0$ the current path inclines in the direction of the flow (curves 2, 4). At $z = 0, y = 0$ there is a current loop that closes in the core only (curve 5). For $y > 0$ such a loop is also present, but of smaller size (curve 6). There are transitional lines, which flow around the recirculating core current zone, but involve either Hartmann or parallel layers (curves 3 and 7).

Now consider the flow for a higher Hartmann number. The results for $Ha = 200$ are shown in Figure 39 - Figure 43. Qualitatively they are similar to those for $Ha = 50$ but the three-dimensional effects are stronger.

Overall the results are in agreement with those in [30] and [31].

6.2 Free-surface flows

In free surface flows, interaction between two phases (liquid and "air") is studied. The homogeneous model available in CFX is used in this study for modelling free surface flows, and therefore equations for both media are solved. Thus, parameters of both metal and "air" should be provided. When comparing results with the asymptotic theory, finite physical properties of the surrounding air should be taken into account (see Table 2).

Table 2 Physical properties of lithium and air ([33], [3], [34])

	Density, kg/m ³	Kinematic viscosity, m ² /s	Dynamic viscosity, kg/(m·s)
Lithium	500	9.0×10^{-7}	4.5×10^{-4}
Air	1.209	1.49×10^{-5}	1.8×10^{-5}

6.2.1 Spreading MHD drop

As the first benchmark problem for the unsteady free-surface flow, we consider an inertialess, two-dimensional, gravity-dominated flow in a spreading drop of liquid metal subject to a strong vertical magnetic field (Figure 44). The reason is that for this unsteady flow an asymptotic, high- Ha solution is available ([35]).

In dimensional form, the flow is governed a by two-dimensional version of Eqs. (1) and (3) with body force

$$\mathbf{F}^* = -sB_0^2 u^* \hat{\mathbf{x}} + r\hat{\mathbf{y}}. \quad (32)$$

The boundary conditions are the non-slip condition at the solid wall

$$\mathbf{v}^* = 0 \text{ at } y^* = 0, \quad (33)$$

and the symmetry condition

$$\frac{\partial v^*}{\partial x^*} = 0, u^* = 0 \text{ at } x^* = 0. \quad (34)$$

Since the air flow around the drop should also be take into account, the following boundary conditions hold for air far from the drop:

$$\frac{\partial u^*}{\partial x^*} = 0, v^* = 0 \text{ as } x^* \rightarrow \infty, \frac{\partial v^*}{\partial y^*} = 0, u^* = 0 \text{ as } y^* \rightarrow \infty. \quad (35)$$

This means that if there is any air flow out/into the computational domain, it is fully developed.

The initial conditions for the metal are zero velocities and the following shape of the drop:

$$h^*(x^*) = a \left[1 - \frac{x^{*2}}{a^2} \right], \quad (36)$$

where the characteristic length a is equal to 0.001 m in presented calculations. Physical data for lithium and air have been used (see Table 2).

The asymptotic analysis presented in [35] shows that the shape of the drop at the time t^* is described by the following formula:

$$h^*(x^*, t^*) = \frac{a}{(1 + 6t^*/t_0)^{1/3}} \left[1 - \frac{x^{*2}}{a^2 (1 + 6t^*/t_0)^{2/3}} \right]. \quad (37)$$

The scaling of time is aSB_0^2 / rg . In the following calculations $B_0 = 0.58$ T and $a = 0.0001$ m have been chosen so that $Ha = 50$.

$$\nabla \cdot \mathbf{v} = 0. \quad (43)$$

In the above, the modified interaction parameter and the modified Hartmann number are defined as follows:

$$Ha_1 = aB_0 \sqrt{\frac{\mathbf{r}\mathbf{n}}{\mathbf{r}_1\mathbf{n}_1}} = Ha\sqrt{\bar{\mathbf{m}}}, \quad N_1 = \frac{a\mathbf{S}B_0^2}{\mathbf{r}_1\nu_0} = N\bar{\mathbf{r}}, \quad (44)$$

where \mathbf{r}_1 , \mathbf{n}_1 are the density and the kinematic viscosity of the air, respectively, and

$$\bar{\mathbf{r}} = \mathbf{r} / \mathbf{r}_1, \quad \bar{\mathbf{m}} = \mathbf{m} / \mathbf{m}_1 \quad (45)$$

are the ratios of the density and the dynamic viscosity of the metal and the air. Real physical data for air and lithium have been used, therefore

$$\bar{\mathbf{r}} = 25, \quad \bar{\mathbf{m}} = 413.56. \quad (46)$$

Since the homogeneous model is used in CFX for modelling the two-phase flow of metal and air, the velocity and pressure fields are shared by both phases (metal and air).

Boundary conditions are the non-slip condition

$$\mathbf{v} = 0 \text{ at solid walls.} \quad (47)$$

Far upstream and downstream the flow is fully developed so that

$$\frac{\partial p}{\partial x} = \text{const}, \quad v = 0 \text{ as } x \rightarrow \pm\infty. \quad (48)$$

At the entrance to the duct, the flow rate

$$Q = \int_0^1 u dy = 1 \quad (49)$$

is given.

6.3.1

First consider uniform magnetic field ($\beta = 1$) for $E = Ha$ the
 x [9] (50). The only ef
of the free surface is minor flow redistribution near the exit of the duct. In the absence of
solid walls, no viscous effects are present. Thus the velocity profile varies from the
file in the jet region far from the
exit from the duct (see 51). The pressure distribution is linear in the duct and
Figure 52 Figure 53 for the parallel layer at the
nozzle.

6.3.2

When inertia is added ($\beta = 1$), the flow in the duct and the jet in the fully developed
regions does not change. However, the character of the flow redistribution near the exit of
uct changes. Due to inertia, pressure gradient is "carried" out of the duct region into
the jet region (55). Therefore, the pressure reaches its constant value further
Figure 56 red to the inertialess case (Figure). Similarly, the
velocity development length into the jet region is larger (57). The jet thickness,

6.3.3 Inertialess flow in a non uniform field

-uniform magnetic field. The magnetic

$$B(x) = \begin{cases} 1, & \text{if } x \leq 0, \\ 1 + (B_{\text{inf}} - 1) \tanh zx & \text{if } x > 0. \end{cases} \quad (50)$$

Here B_{inf} is the induction of the uniform magnetic field as $x \rightarrow \infty$, and z is the gradient of
the magnetic field equal to 2. Since the magnetic field in the jet region is lower than in
the duct region, the Lorentz force becomes weaker. As a result, fluid accelerates and mass
conservation requires jet to shrink. The asymptotic analysis [9] shows that the thickness

of the jet changes with distance as $h(x) = B(x)$. It agrees very well with the numerical solution presented in Figure 59. The velocity in the core region increases as $1/B(x)$, also as predicted by the asymptotic theory of [9]. It is compared with the asymptotic solution in Figure 60 and a perfect agreement is seen. The discrepancy in the duct region is due to the insufficiently high value of the Hartmann number ($Ha = 200$). As Hartmann number tends to infinity, the asymptotic value $u = 1$ is expected. The streamlines near the exit region are shown in Figure 61.

7. Discussion and conclusions

both duct and free- n -dimensional flows, convergence is relatively fast, usually $n-2$ hours. Modelling the three dimensional flows takes far more time. Calculations for a three dimensional flow in a rectangular duct for Ha 96 hours to converge. This compares to about one minute of CPU time for the high Ha

Computational time for the DNS model highly increases with the growth of the Hartmann $Ha > 200$ become very expensive. This is a general feature of the DNS, not specific to the CFX. The question is for what flow conditions it is realistic to use DNS. Based on years of experience of using the high Hartmann number flow model and a short, one year experience with CFX we reached certain conclusions, which are discussed challenges in MHD modelling for various flow regimes. Then we discuss applicability of DNS and high Ha model to flows in various liquid metal systems for tokamaks.

Main problems

Generally, while modelling flows in divertors and blankets of tokamaks one faces the

1. Complex geometry of supplying/draining ducts
Nonuniform magnetic fields
- 3.
4. If walls are fully or partially conducting, global currents may circulate
5.
walls and inside the flow domain.
Deformable free surface.

Some of these issues are illustrated best in the following examples (in both cases high- Ha model has been used):

Example 1: Flow in electrically coupled radial-toroidal-radial bends [36], [37].

A system of bends with common thin conducting walls is shown in Figure 62. The electric currents induced in one bend penetrate the other, change the whole flow pattern and affect the pressure drop. Although the bends are hydraulically decoupled, the whole system must be treated simultaneously.

The position of various boundary layers is shown with the shaded areas. Some layers are being formed across the flow. If Direct Numerical Simulation (DNS) is were used to model this kind of flow, all these layers need to be resolved properly. Failure to do that would lead to wrong results both qualitative and quantitative.

Similar problem arises if insulating dividing walls end within the flow region.

Example 2: Flow in an insulating circular duct in a nonuniform field [5], [6]

The second example is that of the flow in an electrically insulating circular duct in a nonuniform magnetic field varying in the flow direction. The isolines of the core pressure in the plane transverse to the field are shown in Figure 63. There are two prominent features concerning this flow. One is the formation of the flow-induced internal layer inside the core in the region $-3 < x < 3$. The other one is very high development length (the length of the flow affected by 3-D currents). In the example above it is about 20 duct diameters. For ARIES inlet pipes the development length is about 125 duct diameters.

For this type of flow the following observations can be made:

1. Not only the boundary layers but also the internal, curvilinear layer need to be resolved numerically. In the calculations in [5], [6] at least 128 points were needed in the flow direction (using a highly nonuniform grid) and 32 points in the transverse direction to get satisfactory results.

2. The computational domain needs to be very long to account for high development length
3. Since the development length is high, the three-dimensional effects from different divertor/blanket elements (bend, manifold, jet, rivulet, draining duct) will overlap and thus the whole divertor will need to be modelled as a single piece.

The question then is what is realistic to expect from DNS with current computational facilities (supercomputers included).

Table 3 Comparison of DNS and high- Ha model

DNS	HIGH-HA MODEL
<p style="text-align: center;"><i>Advantages</i></p> <ol style="list-style-type: none"> 1. Takes into account all terms in the equations, including inertia 2. Allows treatment of complex geometries and all three components of the field 3. Commercial codes are available (CFX, FLUENT, FLOW-3D, etc.) which are relatively easy to use <p style="text-align: center;"><i>Disadvantages</i></p> <ol style="list-style-type: none"> 1. Very limited development history (almost no comparison with experiments) 2. Has never been proved to have worked for high fields (max. Ha for 3-D flows is between 100 and 500) 3. Very slow 	<ol style="list-style-type: none"> 1. Has been proved to work well for high fields: excellent agreement with the experiments within its range of applicability 2. Most knowledge on liquid-metal MHD for fusion has been obtained using this model 3. Allows treatment of complex geometries, including a three-component field. 4. Fast <ol style="list-style-type: none"> 1. Currently does not take into account inertial effects (although some attempts are being made [42], [43]) 2. Does not work for very low Hartmann numbers 3. More difficult and time consuming to modify than the commercial codes

7.2 Comparison of the DNS and high- Ha flow model

The advantages and disadvantages of DNS and high- Ha model are listed in Table 3.

If one combines Table 3 with the expected flow regimes for various machines (Table 1), one can conclude that if high- Ha model is applicable (large tokamaks and C-MOD), it is a preferred option. It is sufficiently fast to expect that a complex geometry may be modelled as one piece (maybe a major part of a divertor or a blanket). The model has been under development for over 40 years; it has been tried, understood, and compared

with the experiments for duct flows. It also gives a great insight into the effects occurring within the flow.

Concerning DNS, one can expect it to work for relatively low fields reaching the values of Ha of about 500-1000. Therefore, it is applicable for C-MOD. Calculations, however, will be costly. Indeed, perhaps the most comprehensive study of an MHD duct flow with DNS has been performed in [15]. The code has been specifically developed for the geometry studied, and thus was faster than the commercial codes. Nevertheless, to perform calculations presented in the paper required weeks of CPU time.

Applying DNS to model flows in NSTX is very problematic. The reason is that the flow for $N < 1$ is likely to be highly turbulent. This is especially so for insulating walls, for which the parameter of transition to turbulence is not N but Re/Ha [38]. Thus successful flow modelling largely depends on the adequate turbulence models. Despite some initial attempts to develop such models for MHD flow for simple geometries [20], [39]-[41], [44] understanding of turbulent MHD is years away from now. Moreover, different turbulent models may be required for different divertor elements (fully developed duct flow, bends/expansions of ducts, free-surface flows, etc.)

Taking into account what has been said in the above there is no reason to believe that within the next couple of years MHD flows for NSTX can be adequately modelled. As the flow regime in NSTX (turbulent MHD) stands alone among the other machines, from the point of view of understanding of free-surface MHD flows, more knowledge for large-scale tokamaks could be gained by testing the divertor in C-MOD.

Despite the drawbacks, CFX (as, perhaps other codes) is a very convenient tool to study MHD flows. It is very flexible and allows great degree of user control. Therefore, its development ought to be continued, but at this stage it is best not to try to overextend the range of parameters by attempting to reach highest possible values of Ha or lowest possible values of N , as it is very easy to obtain wrong results.

Let us provide an example. To obtain a steady flow CFX provides various possibilities. One is to use underrelaxation; another false time steps. For some flows for high values of

Ha calculations using underrelaxation converged to a solution, which looked very reasonable but failed the test with the exact solution (the solution in the core was not a constant; jets were of different thickness). The difference was small but increased with increasing Ha . No such problem has been observed while using false time steps. Thus flow modelling with CFX, as any other DNS code, requires a great degree of experience not only using the code itself, but with computational fluid dynamics, and above all high-field magnetohydrodynamics.

Comparison with the experiments and available analytical results is crucial at each step of the code development. DNS modelling may be considered as just one in a series of steps on the road towards understanding a particular flow. High- Ha models, exact solutions, and the physical insight must all be involved. Thus at this stage it seems to be best to use DNS to study fundamental flow problems and thus gain knowledge in the range of moderate values of Ha and N , and attempt modelling flows for C-MOD.

8. Acknowledgement

This work has been supported by the Office of Fusion Energy Sciences, U.S. Department of Energy, under Contract No. W-31-109-ENG-38.

We are grateful to Ola Widlund for sharing his experience on working with CFX and for providing subroutines for implementation of the integral formulation of the MHD model.

9. References

- [1] B. G. Karasev, A. V. Tananaev (1990) *Liquid metal fusion reactor systems, Plasma devices and operations* **1**, 11-30.
- [2] R. Mattas et al. (2000) ALPS - advanced limiter-divertor plasma-facing systems, *Fus. Tech. Eng. Des.* **49-50**, 127-134.
- [3] S. Molokov, C. B. Reed (1999) Review of free-surface MHD experiments and modeling, *Argonne National Laboratory Report, ANL/TD/TM99-08*.

-
- [4] S. Molokov, I. Cox, C. B. Reed (2001) Theoretical investigation of liquid metal MHD free surface flows for ALPS, *Fusion Technology* **39**, 880-884.
- [5] S. Molokov, C. B. Reed (2002) Parametric study of the liquid metal flow in a straight insulated circular duct in a strong nonuniform magnetic field, *Fusion Science and Technology*, (to be published).
- [6] S. Molokov, C. B. Reed (2001) Liquid metal flow in an insulated circular duct in a strong non-uniform magnetic field, Part 1: Flow in a straight duct and the benchmark problem, *Argonne National Laboratory Report, ANL/TD/TM01-18*.
- [7] S. Molokov, C. B. Reed (2001) Liquid metal flow in an insulated circular duct in a strong non-uniform magnetic field, Part 2: Inclination of the field gradient to the duct axis, *Argonne National Laboratory Report, ANL/TD/TM01-19*.
- [8] S. Molokov, C. B. Reed (2001) Liquid-metal jet flow in a strong uniform magnetic field, *Argonne National Laboratory Report, ANL/TD/TM01-21*.
- [9] S. Molokov, C. B. Reed (2002) Flow of a two-dimensional Liquid Metal jet in a Strong Magnetic field, *Argonne National Laboratory Report, ANL/TD/TM02-29*.
- [10] S. Molokov, C. B. Reed (2000) Fully developed magnetohydrodynamic flow in a rivulet, *Argonne National Laboratory Report, ANL/TD/TM00-12*.
- [11] J.N. Brooks, T.D. Rognlien, D.N. Ruzic, J.P. Allain (2001) Erosion/redeposition analysis of lithium-based liquid surface divertors, *Journal of Nuclear Materials* **290-293**, 185-190.
- [12] M. Ulrickson (2001) Applications of Free Surface Liquids for Fusion Plasma Facing Components, *Int. Seminar on Electromagnetic control of Liquid Metal Processes, Coventry, UK, June 27-29 2001*.
- [13] A. Sterl (1990) Numerical simulation of liquid-metal flows in rectangular ducts, *J. Fluid Mech.* **216**, 161-191.

-
- [14] L. Lenhart (1994) Magnetohydrodynamik in Rechteckgeometrien, *Wissenschaftliche Berichte FZKA 5317, Forschungszentrum Karlsruhe*.
- [15] B. Mück, C. Günter, U. Müller, L. Bühler (2000) Three-dimensional flows in rectangular ducts with internal obstacles, *J. Fluid Mech.* **418**, 265-295.
- [16] L. Leboucher (1999) Monotone scheme and boundary conditions for finite volume simulation of magnetohydrodynamic internal flows at high Hartmann number, *J. Comput. Phys.* **150**, 181-198.
- [17] I. Di Piazza, L. Bühler (1999) Numerical simulations of Buoyant Magnetohydrodynamic flows using the CFX code, *Forschungszentrum Karlsruhe Report FZKA 6354*.
- [18] I. Di Piazza, M. Ciofalo (2002) MHD free convection in a liquid-metal filled cubic enclosure. I. Differential heating, *Int. J. Heat and Mass Transfer* **45**, 1477-1492.
- [19] I. Di Piazza, M. Ciofalo (2002) MHD free convection in a liquid-metal filled cubic enclosure. II. Internal heating, *Int. J. Heat and Mass Transfer* **45**, 1493-1511.
- [20] O. Widlund (2000) Modelling of magnetohydrodynamic turbulence, *Technical Report ISRN KTH/MEK/TR--99/11--SE, TRITA-MEK*.
- [21] T. Tagawa, R. Moreau, G. Authié (2000) Buoyant flow in long vertical enclosures in the presence of a strong horizontal magnetic field, *Proceeding of the 4th Pamir International conference, France*, 153-158.
- [22] A. Y. Ying *et al.* (2001) MHD and heat transfer issues and characteristics for Li free surface flows under NSTX conditions, *Fusion Technology* **39**, 739-745.
- [23] CFX On-Line Help, AEA Technology Engineering Software Ltd.

-
- [24] J. A. Shercliff (1953) Steady motion of conductive fluids in pipes under transverse magnetic fields, *Proc. Cambr. Philos. Soc.* **49**, 136-144.
- [25] J. C. R. Hunt (1965) Magnetohydrodynamic flow in rectangular ducts, *J. Fluid Mech.* **21**, 577-590.
- [26] S. Molokov (2000) Two-dimensional parallel layers at high Ha, Re and N, *Proceeding of the 4th Pamir International conference, France*, 153-158.
- [27] N. Alleborn, K. Nandakumar, H. Raszillier, F. Durst (1997) Further contributions on the two-dimensional flow in a sudden expansion, *J. Fluid Mech.* **330**, 169-188.
- [28] W. Cherdron, F. Durst, J. H. Whitelaw (1978) Asymmetric flows and instabilities in symmetric ducts with sudden expansions, *J. Fluid Mech.* **84**, 13-31.
- [29] F. Durst, J. C. F. Pereira, C. Tropea (1993) The plane symmetric sudden-expansion flow at low Reynolds numbers, *J. Fluid Mech.* **248**, 567-581.
- [30] C. C. Sellers, J. S. Walker (1999) Liquid-metal flow in an electrically insulated rectangular duct with a non-uniform magnetic field, *Int. J. Engineering Science* **37**, 541-552
- [31] J. S. Walker, B. F. Picologlou (1995) Liquid-metal flow in an insulated rectangular expansion with a strong transverse magnetic field, *J. Fluid Mech.* **305**, 111-126.
- [32] R. Holroyd, J. S. Walker (1989) A theoretical study of the effects of wall conductivity, non-uniform magnetic fields and variable-area ducts on liquid metal flows at high Hartmann number, *J. Fluid Mech.* **84**, 471-495.
- [33] CFX Material properties database, AEA Technology Engineering Software Ltd.
- [34] C. J. Smithells, E. A. Brandes (1976) Metals reference book, *Butterworths*: London & Boston.

-
- [35] S. Molokov (2002) Evolution of free surface disturbance in a strong magnetic field, *Proceeding of the 5th Pamir International conference, France, September 2002*.
- [36] S. Molokov, R. Stieglitz (1995) Liquid-metal flow in a system of electrically coupled U-bends in a strong uniform magnetic field. *J. Fluid Mech.* **299**, 73-95.
- [37] R. Stieglitz, S. Molokov (1997) Experimental study of magnetohydrodynamic flows in electrically coupled bends, *J. Fluid Mech.* **343**, 1-28.
- [38] J. Sommeria, R. Moreau (1982) Why, how, and when, MHD turbulence becomes two-dimensional, *J. Fluid Mech.* **118**, 507-518.
- [39] S. Cuevas, P. F. Picologlou, J. S. Walker, G. Talmage (1997) Liquid-metal MHD flow in rectangular ducts with thin conducting or insulating walls: laminar and turbulent solutions, *Int. J. Engineering Science* **35**, 485-503.
- [40] S. Cuevas, P. F. Picologlou, J. S. Walker, G. Talmage. T. Q. Hua (1997) Heat transfer in laminar and turbulent liquid-metal MHD flows, *Int. J. Engineering Science* **35**, 505-514.
- [41] S. Smolentsev et al. (2002) Application of the "K-epsilon" model to open channel flows in a magnetic field, *Int. J. Eng. Science* **40**, 693-711.
- [42] L. Bühler (1996) Instabilities in quasi-two-dimensional magnetohydrodynamic flows, *J. Fluid Mech.* **326**, 125-150.
- [43] A. Potherat, J. Sommeria, R. Moreau (2000) An effective two-dimensional model for MHD flows with transverse magnetic field, *J. Fluid Mech.* **424**, 75-100.
- [44] D. Lee, H. Choi (2001) Magnetohydrodynamic turbulent flow in a channel at low magnetic Reynolds number, *J. Fluid Mech.* **439**, 367-394.

10. List of figures

- Figure 1** MHD problems in for the upper parts of the divertor.
- Figure 2** MHD problems in for the lower part of the divertor.
- Figure 3** Shercliff solution. Geometry and co-ordinate system.
- Figure 4** Shercliff solution. Grid used: (a) for $Ha = 100$; (b) for $Ha = 200$.
- Figure 5** Shercliff solution. Electric potential in the plane $y = 0$ for a square duct and for $Ha = 100$ (numerical solution - solid lines; exact solution - crosses).
- Figure 6** Shercliff solution. Axial velocity in the plane $x = 0$ for a square duct and for $Ha = 100$ (numerical solution - solid lines; exact solution - crosses).
- Figure 7** Shercliff solution. Axial velocity in the plane $y = 0$ for a square duct and for $Ha = 100$ (numerical solution - solid lines; exact solution - crosses).
- Figure 8** Shercliff solution. Electric potential at $x = 0$ for a square duct and for $Ha = 200$ (numerical solution - solid lines; exact solution - crosses).
- Figure 9** Shercliff solution. Axial velocity in the plane $y = 0$ for a square duct and for $Ha = 200$ (numerical solution - solid lines; exact solution - crosses).
- Figure 10** Shercliff solution. Axial velocity in the plane $x = 0$ for a square duct and for $Ha = 200$ (numerical solution - solid lines; exact solution - crosses).

-
- Figure 11** **Hunt solution. Geometry and co-ordinate system.**
- Figure 12** **Hunt solution for a square duct with perfectly conducting walls. Axial velocity in the plane $y = 0$ for $Ha = 100$ (numerical solution - solid lines; exact solution - stars).**
- Figure 13** **Hunt solution for a square duct with perfectly conducting walls. Electric potential in the plane $y = 0$ for $Ha = 100$ (numerical solution - solid lines; exact solution - stars).**
- Figure 14** **Hunt solution for a square duct with perfectly conducting Hartmann walls and electrically insulating parallel walls. Electric potential in the plane $y = 0$ for $Ha = 100$ (numerical solution - solid lines; exact solution - stars).**
- Figure 15** **Hunt solution for a square duct with perfectly conducting Hartmann walls and electrically insulating parallel walls. Axial velocity in the plane $y = 0$ for $Ha = 100$ (numerical solution - solid lines; exact solution - stars).**
- Figure 16** **Hunt solution for a square duct with perfectly conducting Hartmann walls and electrically insulating parallel walls. Electric potential in the plane $y = 0$ for $Ha = 200$ (numerical solution - solid lines; exact solution - stars).**
- Figure 17** **Hunt solution for a square duct with perfectly conducting Hartmann walls and electrically insulating parallel walls. Axial velocity in the plane $y = 0$ for $Ha = 200$ (numerical solution - solid lines; exact solution - stars).**
- Figure 18** **Flow in a duct with a 1:2 symmetric expansion in a transverse magnetic field.**

-
- Figure 19** Inertialess flow in a duct with an expansion. Streamlines for $Ha = 200$.
- Figure 20** Inertialess flow in a duct with an expansion. Velocity profiles in the duct ($x = -3$) and at the junction ($x = 0$) for $Ha = 200$. Solid lines - CFX numerical solution, stars - numerical solution obtained by a different method ([26]).
- Figure 21** Inertialess flow in a duct with an expansion. Core velocity ($y = 0$) for $Ha = 200$.
- Figure 22** Inertialess flow in a duct with an expansion. Pressure distribution for $Ha = 200$.
- Figure 23** Inertialess flow in a duct with an expansion. Pressure at the top of the wider duct ($y = 1$) for $Ha = 200$.
- Figure 24** Inertial flow in a duct with an expansion. Streamlines for $Ha = 200$, $N = 1$.
- Figure 25** Inertial flow in a duct with an expansion. Velocity profiles in the duct ($x = -3$) and at the junction ($x = 0$) for $Ha = 200$, $N = 1$.
- Figure 26** Inertial flow in a duct with an expansion. Pressure at the top of the wider duct ($y = 1$) for $Ha = 200$, $N = 1$.
- Figure 27** Inertial flow in a duct with an expansion. Pressure distribution for $Ha = 200$, $N = 1$.
- Figure 28** Flow in an asymmetric duct with an expansion in a transverse magnetic field.

-
- Figure 29** Inertial flow in an asymmetric duct with an expansion. Streamlines for $Ha = 200, N = 1$.
- Figure 30** Inertial flow in an asymmetric duct with an expansion. Pressure distribution for $Ha = 200, N = 1$.
- Figure 31** Inertial flow in an asymmetric duct with an expansion. Pressure at the top of the wider duct ($y = 1$) for $Ha = 200, N = 1$.
- Figure 32** Flow in a square duct in a non-uniform transverse magnetic field.
- Figure 33** Flow in a square duct in a non-uniform transverse magnetic field. Magnetic field versus axial co-ordinate.
- Figure 34** Axial velocity profiles for inertialess flow in a square duct in a non-uniform transverse magnetic field. $Ha = 50$. Line $y = 0$ and (a) $z = -6$; (b) $z = -2$; (c) $z = -1$; (d) $z = -0.5$; (e) $z = 0$; (f) $z = 0.5$; (g) $z = 1$; (h) $z = 2$; (i) $z = 6$.
- Figure 35** Axial velocity profiles in the inertialess flow in a square duct in a non-uniform transverse magnetic field for $Ha = 50$.
- Figure 36** Streamlines in the inertialess flow in a square duct in a non-uniform transverse magnetic field in the plane $y = 0$ for $Ha = 50$.
- Figure 37** Pressure in the inertialess flow in a square duct in a non-uniform transverse magnetic field on the central line of the duct $x = y = 0$ (broken line) and near the wall $x = y = 1$ (solid line). $Ha = 50$.
- Figure 38** Electric current lines in the inertialess flow in a square duct in a non-uniform transverse magnetic field for $Ha = 50$.

-
- Figure 39** Axial velocity profiles in the inertialess flow in a square duct in a non-uniform transverse magnetic field in the plane $y = 0$ for $Ha = 200$.
- Figure 40** Streamlines in the inertialess flow in a square duct in a non-uniform transverse magnetic field in the plane $y = 0$ for $Ha = 200$.
- Figure 41** Pressure in the inertialess flow in a square duct in a non-uniform transverse magnetic field on the central line of the duct $x = y = 0$ (broken line) and near the wall $x = y = 1$ (solid line). $Ha = 200$.
- Figure 42** Electric current lines in the inertialess flow in a square duct in a non-uniform transverse magnetic field in the plane $y = 0$ for $Ha = 200$.
- Figure 43** Pressure variation in the inertialess flow in a square duct in a non-uniform transverse magnetic field in the plane $y = 0$ for $Ha = 200$.
- Figure 44** Liquid metal drop in a strong, vertical magnetic field.
- Figure 45** Liquid metal drop in a strong, vertical magnetic field after 0.185 s. Solid line - asymptotic solution.
- Figure 46** Liquid metal drop in a strong, vertical magnetic field. Horizontal velocity for both phases ("air" and liquid metal). Solid line - drop surface (asymptotic solution).
- Figure 47** Liquid metal drop in a strong, vertical magnetic field. Vertical velocity for both phases ("air" and liquid metal). Solid line - drop surface (asymptotic solution).
- Figure 48** Liquid metal drop in a strong, vertical magnetic field. Pressure for both phases ("air" and liquid metal). Solid line - drop surface (asymptotic solution).

-
- Figure 49** Liquid metal jet in a strong, transverse magnetic field.
- Figure 50** Liquid metal jet in a strong, transverse magnetic field. Variation of jet thickness in a uniform field for $E = -1$, $Ha = 200$.
- Figure 51** Liquid metal jet in a strong, transverse magnetic field. Velocity profile in a uniform field for $E = -1$, $Ha = 200$. Velocity in the duct (solid line) and in the jet region (stars).
- Figure 52** Liquid metal jet in a strong, transverse magnetic field. Variation of pressure in a uniform field for $E = -1$, $Ha = 200$.
- Figure 53** Liquid metal jet in a strong, transverse magnetic field. Variation of pressure in a uniform field at $y = 0$ for $E = -1$, $Ha = 200$.
- Figure 54** Liquid metal jet in a strong, transverse magnetic field. Velocity in the core in a uniform field at $y = 0$ for $E = -1$, $Ha = 200$.
- Figure 55** Liquid metal jet in a strong, transverse magnetic field. Variation of pressure in a uniform field for $E = -1$, $Ha = 200$, $N = 1$.
- Figure 56** Liquid metal jet in a strong, transverse magnetic field. Variation of pressure in a uniform field at $y = 0$ for $E = -1$, $Ha = 200$, $N = 1$.
- Figure 57** Liquid metal jet in a strong, transverse magnetic field. Velocity in the core in a uniform field at $y = 0$ for $E = -1$, $Ha = 200$, $N = 1$.
- Figure 58** Liquid metal jet in a strong, transverse magnetic field. Variation of jet thickness for $E = -1$, $Ha = 200$, $B_\infty = 0.5$, $z = 2$. Colour map represents the numerical solution, the solid black line shows the asymptotic solution.

-
- Figure 59** Liquid metal jet in a strong, transverse magnetic field. Variation of pressure for $E = -1$, $Ha = 200$, $B_\infty = 0.5$, $z = 2$.
- Figure 60** Liquid metal jet in a strong, transverse magnetic field. Core velocity in the jet ($y = 0$) for $E = -1$, $Ha = 200$, $B_\infty = 0.5$, $z = 2$. Solid line corresponds to the numerical solution, stars to the asymptotic solution.
- Figure 61** Liquid metal jet in a strong, transverse magnetic field. Streamlines in the jet for $E = -1$, $Ha = 200$, $B_\infty = 0.5$, $z = 2$.
- Figure 62** Flow in electrically coupled U-bends(from [36]).
- Figure 63** Flow in a circular insulating duct in a nonuniform magnetic field. Projection of lines of constant pressure onto the plane transverse to the field. The field is out of the plane of the figure; it varies between $x = -1$ and $x = 1$. Variable x is in the flow direction. Variable z is in the direction transverse to the magnetic field (duct axis is at $z = 0$). Hartmann and Roberts layers are not shown. Here $Ha=7000$. (from [5]).

11. Appendix 1: a sample of command file

```
>>CFX4
  >>SET LIMITS
    LARGE
    TOTAL INTEGER WORK SPACE 5000000
    TOTAL CHARACTER WORK SPACE 2000
    TOTAL REAL WORK SPACE 70000000
    MAXIMUM NUMBER OF BLOCKS 10
    MAXIMUM NUMBER OF PATCHES 100
    MAXIMUM NUMBER OF INTER BLOCK BOUNDARIES 20
  >>OPTIONS
    TWO DIMENSIONS
    RECTANGULAR GRID
    CARTESIAN COORDINATES
    LAMINAR FLOW
    ISOTHERMAL FLOW
    INCOMPRESSIBLE FLOW
    TRANSIENT FLOW
    USER SCALAR EQUATIONS 3
    NUMBER OF PHASES 2
  >>USER FORTRAN
    USRBCS
    USRBF
    USRCVG
    USRINT
    USRTRN
  >>VARIABLE NAMES
    U VELOCITY 'U VELOCITY'
    V VELOCITY 'V VELOCITY'
    W VELOCITY 'W VELOCITY'
    PRESSURE 'PRESSURE'
    VOLUME FRACTION 'VOLUME FRACTION'
    DENSITY 'DENSITY'
    VISCOSITY 'VISCOSITY'
    USER SCALAR1 'USRD B'
    USER SCALAR2 'USRD EXACT U'
    USER SCALAR3 'USRD EXACT H'
  >>PHASE NAMES
    PHASE1 'AIR'
    PHASE2 'METAL'
  >>MODEL TOPOLOGY
  >>MODIFY PATCH
    OLD PATCH NAME 'INLET'
    NEW PATCH NAME 'ENTRANCE'
    NEW PATCH TYPE 'INLET'
  >>MODIFY PATCH
    OLD PATCH NAME 'OUTLET'
    NEW PATCH NAME 'EXIT'
    NEW PATCH TYPE 'MASS FLOW BOUNDARY'
  >>MODIFY PATCH
    OLD PATCH NAME 'USER3D DUCT'
    NEW PATCH NAME 'ALLDUCT'
    NEW PATCH TYPE 'USER3D'
  >>MODEL DATA
```

```
>>AMBIENT VARIABLES
  PHASE NAME 'AIR'
  VOLUME FRACTION 1.0000E+00
>>AMBIENT VARIABLES
  PHASE NAME 'METAL'
  VOLUME FRACTION 0.0000E+00
>>DIFFERENCING SCHEME
  ALL EQUATIONS 'NO CONVECTION'
>>TITLE
  PROBLEM TITLE 'MHD JET'
>>PHYSICAL PROPERTIES
  >>FLUID PARAMETERS
    PHASE NAME 'AIR'
    VISCOSITY 1.000E-06
    DENSITY 1.0000E+00
  >>FLUID PARAMETERS
    PHASE NAME 'METAL'
    VISCOSITY 2.5000E-05
    DENSITY 1.0000E+00
  >>MULTIPHASE PARAMETERS
    >>PHASE DESCRIPTION
      PHASE NAME 'AIR'
      GAS
      CONTINUOUS
    >>PHASE DESCRIPTION
      PHASE NAME 'METAL'
      LIQUID
      CONTINUOUS
    >>MULTIPHASE MODELS
      >>MOMENTUM
        HOMOGENEOUS
        SURFACE SHARPENING ALGORITHM
  >>TRANSIENT PARAMETERS
    >>FIXED TIME STEPPING
      TIME STEPS 100* 1.00000E+00
>>SOLVER DATA
  >>PROGRAM CONTROL
    MAXIMUM NUMBER OF ITERATIONS 1000
    MINIMUM NUMBER OF ITERATIONS 100
  >>UNDER RELAXATION FACTORS
    U VELOCITY 8.000E-01
    V VELOCITY 8.000E-01
    PRESSURE 1.0000E+00
    VOLUME FRACTION 1.0000E+00
    VISCOSITY 1.0000E+00
>>MODEL BOUNDARY CONDITIONS
  >>INLET BOUNDARIES
    PHASE NAME 'AIR'
    PATCH NAME 'ENTRANCE'
    VOLUME FRACTION 0.0000E+00
  >>INLET BOUNDARIES
    PHASE NAME 'METAL'
    PATCH NAME 'ENTRANCE'
    VOLUME FRACTION 1.0000E+00
  >>PRESSURE BOUNDARIES
    PHASE NAME 'METAL'
    PATCH NAME 'PRESS FORCED'
```

```

    VOLUME FRACTION 0.0000E+00
>>PRESSURE BOUNDARIES
    PHASE NAME 'AIR'
    PATCH NAME 'PRESS FORCED'
    VOLUME FRACTION 1.0000E+00
>>OUTPUT OPTIONS
    >>DUMP FILE OPTIONS
        ALL PHASES
        EACH TIME STEP
        INITIAL GUESS
        FINAL SOLUTION
        ALL REAL DATA
        NO GEOMETRY DATA
>>STOP

```

12. Appendix 2: a sample of source code, two-dimensional flow, MHD jet

```

    SUBROUTINE USRINT(U,V,W,P,VFRAC,DEN,VIS,TE,ED,RS,T,H,RF,SCAL,CONV,
+                   XC,YC,ZC,XP,YP,ZP,VOL,AREA,VPOR,ARPOR,WFACT,
+                   DISWAL,IPT,IBLK,IPVERT,IPNODN,IPFACN,IPNODF,
+                   IPNODE,IPFACB,WORK,IWORK,CWORK)
C#####
C#####
C      Program for setting initial data at the first time step
C#####
C#####
C
C*****
C
C      UTILITY SUBROUTINE FOR USER-SUPPLIED INITIAL FIELD.
C
C*****
C
C      THIS SUBROUTINE IS CALLED BY THE FOLLOWING SUBROUTINE
C      CUSR  INIT
C
C*****
C      CREATED
C      13/06/90  ADB
C      MODIFIED
C      07/08/91  IRH   NEW STRUCTURE
C      10/09/91  IRH   CORRECTION TO IUSED
C      26/09/91  IRH   ALTER ARGUMENT LIST
C      01/10/91  DSC   REDUCE COMMENT LINE GOING OVER COLUMN 72.
C      03/10/91  IRH   CORRECT COMMENTS
C      28/01/92  PHA   UPDATE CALLED BY COMMENT, ADD RF ARGUMENT,
C                     CHANGE LAST DIMENSION OF RS TO 6 AND IVERS TO 2
C      03/06/92  PHA   ADD PRECISION FLAG AND CHANGE IVERS TO 3
C      08/02/93  NSW   REMOVE REDUNDANT COMMENTS
C      23/11/93  CSH   EXPLICITLY DIMENSION IPVERT ETC.
C      03/02/94  PHA   CHANGE FLOW3D TO CFDS-FLOW3D, REMOVE COMMA
C                     FROM BEGINNING OF DIMENSION STATEMENT
C      03/03/94  FHW   CORRECTION OF SPELLING MISTAKE
C      09/08/94  NSW   CORRECT SPELLING
C                     MOVE 'IF(IUSED.EQ.0) RETURN' OUT OF USER AREA
C      19/12/94  NSW   CHANGE FOR CFX-F3D
C      30/01/95  NSW   INCLUDE NEW EXAMPLE
C      02/07/97  NSW   UPDATE FOR CFX-4

```

```

C
C*****
C
C   SUBROUTINE ARGUMENTS
C
C   U       - U COMPONENT OF VELOCITY
C   V       - V COMPONENT OF VELOCITY
C   W       - W COMPONENT OF VELOCITY
C   P       - PRESSURE
C   VFRAC   - VOLUME FRACTION
C   DEN     - DENSITY OF FLUID
C   VIS     - VISCOSITY OF FLUID
C   TE      - TURBULENT KINETIC ENERGY
C   ED      - EPSILON
C   RS      - REYNOLD STRESSES
C   T       - TEMPERATURE
C   H       - ENTHALPY
C   RF      - REYNOLD FLUXES
C   SCAL    - SCALARS (THE FIRST 'NCONC' OF THESE ARE MASS FRACTIONS)
C   CONV    - CONVECTION COEFFICIENTS
C   XC      - X COORDINATES OF CELL CORNERS
C   YC      - Y COORDINATES OF CELL CORNERS
C   ZC      - Z COORDINATES OF CELL CORNERS
C   XP      - X COORDINATES OF CELL CENTRES
C   YP      - Y COORDINATES OF CELL CENTRES
C   ZP      - Z COORDINATES OF CELL CENTRES
C   VOL     - VOLUME OF CELLS
C   AREA    - AREA OF CELLS
C   VPOR    - POROUS VOLUME
C   ARPOR   - POROUS AREA
C   WFACT   - WEIGHT FACTORS
C   DISWAL  - DISTANCE OF CELL CENTRE FROM WALL
C
C   IPT     - 1D POINTER ARRAY
C   IBLK    - BLOCK SIZE INFORMATION
C   IPVERT  - POINTER FROM CELL CENTERS TO 8 NEIGHBOURING VERTICES
C   IPNODN  - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING CELLS
C   IPFACN  - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING FACES
C   IPNODF  - POINTER FROM CELL FACES TO 2 NEIGHBOURING CELL CENTERS
C   IPNODB  - POINTER FROM BOUNDARY CENTERS TO CELL CENTERS
C   IPFACB  - POINTER FROM BOUNDARY CENTERS TO BOUNDARY FACES
C
C   WORK    - REAL WORKSPACE ARRAY
C   IWORK   - INTEGER WORKSPACE ARRAY
C   CWORK   - CHARACTER WORKSPACE ARRAY
C
C   SUBROUTINE ARGUMENTS PRECEDED WITH A '*' ARE ARGUMENTS THAT MUST
C   BE SET BY THE USER IN THIS ROUTINE.
C
C   LOGICAL VARIABLE LRDISK IN COMMON BLOCK IOLOGC INDICATES WHETHER
C   THE RUN IS A RESTART AND CAN BE USED SO THAT INITIAL INFORMATION
C   IS ONLY SET WHEN STARTING A RUN FROM SCRATCH.
C
C   NOTE THAT OTHER DATA MAY BE OBTAINED FROM CFX-4 USING THE
C   ROUTINE GETADD, FOR FURTHER DETAILS SEE THE VERSION 4
C   USER MANUAL.
C
C*****
C   DOUBLE PRECISION U
C   DOUBLE PRECISION V
C   DOUBLE PRECISION W
C   DOUBLE PRECISION P
C   DOUBLE PRECISION VFRAC

```

```

DOUBLE PRECISION DEN
DOUBLE PRECISION VIS
DOUBLE PRECISION TE
DOUBLE PRECISION ED
DOUBLE PRECISION RS
DOUBLE PRECISION T
DOUBLE PRECISION H
DOUBLE PRECISION RF
DOUBLE PRECISION SCAL
DOUBLE PRECISION CONV
DOUBLE PRECISION XC
DOUBLE PRECISION YC
DOUBLE PRECISION ZC
DOUBLE PRECISION XP
DOUBLE PRECISION YP
DOUBLE PRECISION ZP
DOUBLE PRECISION VOL
DOUBLE PRECISION AREA
DOUBLE PRECISION VPOR
DOUBLE PRECISION ARPOR
DOUBLE PRECISION WFACT
DOUBLE PRECISION DISWAL
DOUBLE PRECISION WORK
DOUBLE PRECISION TIME
DOUBLE PRECISION DT
DOUBLE PRECISION DTINVF
DOUBLE PRECISION TPARM
DOUBLE PRECISION FULL
DOUBLE PRECISION EMPTY
LOGICAL LDEN, LVIS, LTURB, LTEMP, LBUOY, LSCAL, LCOMP, LRECT, LCYN, LAXIS,
+       LPOROS, LTRANS
LOGICAL LRDISK, LWDISK
C
CHARACTER*(*) CWORK
C
C+++++ USER AREA 1 ++++++
C---- AREA FOR USERS EXPLICITLY DECLARED VARIABLES
DOUBLE PRECISION Ha, RO, SIGMA, AA, VISC, B0, ParInterC, YY, VEL
C+++++ END OF USER AREA 1 ++++++
C
COMMON /ALL/NBLOCK, NCELL, NBDRY, NNODE, NFACE, NVERT, NDIM,
+       /ALLWRK/NRWS, NIWS, NCWS, IWRFRE, IWIFRE, IWCFRE, /ADDIMS/NPHASE,
+       NSCAL, NVAR, NPROP, NDVAR, NDPROP, NDXNN, NDGEOM, NDCOEF, NILIST,
+       NRLIST, NTOPOL, /CHKUSR/IVERS, IUCALL, IUSED, /DEVICE/NREAD,
+       NWRITE, NRDISK, NWDISK, /IDUM/ILEN, JLEN, /IOLOGC/LRDISK, LWDISK,
+       /LOGIC/LDEN, LVIS, LTURB, LTEMP, LBUOY, LSCAL, LCOMP, LRECT, LCYN,
+       LAXIS, LPOROS, LTRANS, /MLTGRD/MLEVEL, NLEVEL, ILEVEL,
+       /SGLDBL/IFLGPR, ICHKPR, /TRANSI/NSTEP, KSTEP, MF, INCORE,
+       /TRANSR/TIME, DT, DTINVF, TPARM
C
C+++++ USER AREA 2 ++++++
C---- AREA FOR USERS TO DECLARE THEIR OWN COMMON BLOCKS
C   THESE SHOULD START WITH THE CHARACTERS 'UC' TO ENSURE
C   NO CONFLICT WITH NON-USER COMMON BLOCKS
C
C+++++ END OF USER AREA 2 ++++++
C
DIMENSION U(NNODE, NPHASE), V(NNODE, NPHASE), W(NNODE, NPHASE),
+       P(NNODE, NPHASE), VFRAC(NNODE, NPHASE), TE(NNODE, NPHASE),
+       ED(NNODE, NPHASE), RS(NNODE, NPHASE, 6), T(NNODE, NPHASE),
+       H(NNODE, NPHASE), RF(NNODE, NPHASE, 4),
+       SCAL(NNODE, NPHASE, NSCAL), DEN(NNODE, NPHASE),
+       VIS(NNODE, NPHASE), CONV(NFACE, NPHASE)

```

```

      DIMENSION XC(NVERT),YC(NVERT),ZC(NVERT),XP(NNODE),YP(NNODE),
+             ZP(NNODE),VOL(NCELL),AREA(NFACE,3),VPOR(NCELL),
+             ARPOR(NFACE,3),WFACT(NFACE),DISWAL(NCELL)
      DIMENSION IPT(*),IBLK(5,NBLOCK),IPVERT(NCELL,8),IPNODN(NCELL,6),
+             IPFACN(NCELL,6),IPNODF(NFACE,4),IPNODB(NBDRY,4),
+             IPFACB(NBDRY)
      DIMENSION IWORK(NIWS),WORK(NRWS),CWORK(NCWS)
C
C+++++ USER AREA 3 ++++++
C---- AREA FOR USERS TO DIMENSION THEIR ARRAYS
C
C---- AREA FOR USERS TO DEFINE DATA STATEMENTS
C
C+++++ END OF USER AREA 3 ++++++
C
C---- STATEMENT FUNCTION FOR ADDRESSING
      IP(I,J,K) = IPT((K-1)*ILEN*JLEN+ (J-1)*ILEN+I)
C
C----VERSION NUMBER OF USER ROUTINE AND PRECISION FLAG
C
      IVERS = 3
      ICHKPR = 2
C
C+++++ USER AREA 4 ++++++
C---- TO USE THIS USER ROUTINE FIRST SET IUSED=1
C
      IUSED = 1
C
C+++++ END OF USER AREA 4 ++++++
C
      IF (IUSED.EQ.0) RETURN
C
C---- FRONTEND CHECKING OF USER ROUTINE
      IF (IUCALL.EQ.0) RETURN
C
C+++++ USER AREA 5 ++++++
C
C---- AREA FOR INITIALISING VARIABLES U,V,W,P,VFRAC,TE,ED,RS,T,SCAL
C      ONLY.
C
C---- EXAMPLE 1 (SET TEMPERATURE TO 300.0 EVERYWHERE)
C
C      USE IPALL TO FIND 1D ADDRESSES OF ALL CELL CENTRES
C      CALL IPALL(' ',' ','BLOCK','CENTRES',IPT,NPT,CWORK,IWORK)
C
C      DO 100 IPHASE = 1, NPHASE
C      LOOP OVER ALL INTERIOR CELLS
C      DO 110 I=1,NPT
C      USE ARRAY IPT TO GET ADDRESS
C      INODE=IPT(I)
C      T(INODE,IPHASE)=300.0
C 110 CONTINUE
C 100 CONTINUE
C
C---- END OF EXAMPLE 1
C
C---- EXAMPLE 2 (SET FIRST SCALAR TO 0.5 EVERYWHERE IF STARTING RUN
C      FROM SCRATCH, BUT NOT ON A RESTART).
C
C      IF(.NOT.LRDISK) THEN
C
C      USE IPALL TO FIND 1D ADDRESSES OF ALL CELL CENTRES
C      CALL IPALL(' ',' ','BLOCK','CENTRES',IPT,NPT,CWORK,IWORK)

```

```

C
C   DO 100 IPHASE = 1, NPHASE
C   LOOP OVER ALL INTERIOR CELLS
C     DO 110 I=1,NPT
C   USE ARRAY IPT TO GET ADDRESS
C     INODE=IPT(I)
C     SCAL(INODE,IPHASE,1)=0.5
C 110   CONTINUE
C 100 CONTINUE
C
C   END IF
C
C---- END OF EXAMPLE 2
C
C----TO SET UP THE INITIAL FIELD FOR REFERENCE EXAMPLE 29
C
C#####
C   Setting initial data:
C   shape and velocities of the jet
C#####

C#####
C   If program restarts from previous dump file (LRDISK=.TRUE.),
C   no initial data is set
C#####

      IF(.NOT.LRDISK) THEN

      AA      = 1.D+00

      Ha      = 200.D0

      SIGMA   = 3.3434D+06
      VISC    = 9.D-07
      RO      = 500.D+00
      B0      = DSQRT(RO*VISC/SIGMA)*Ha/AA
      ParInter=Ha**2.D0*VISC/AA

      write(6,*) '##### B0 = ',B0
      write(6,*) '##### N = ',ParInter

      FULL = 1.0D0
      EMPTY = 1.D-10

C#####
C   First, velocities are set to zero everywhere and
C   all domain full of "air" (phase 1)
C#####

      CALL IPALL(' ',' ','BLOCK','CENTRES',IPT,NPT,CWORK,IWORK)

      DO 101 K = 1,NPT
         INODE = IPT(K)
         VFRAC(INODE,1) = FULL
         VFRAC(INODE,2) = EMPTY
         U(INODE,1) = 0.D0
         V(INODE,1) = 0.D0
         U(INODE,2) = 0.D0
         V(INODE,2) = 0.D0
      101 CONTINUE

```

```

C#####
C      On patch ALLDUCT (straight central subdomain)
C      volume fraction of liquid metal (phase 2) is set to 1
C      and Hartmann profile for velocity
C#####

      CALL IPALL('ALLDUCT','*','PATCH','CENTRES',IPT,NPT,CWORK,IWORK)

DO 102 K = 1,NPT
  INODE = IPT(K)
  VFRAC(INODE,1) = EMPTY
  VFRAC(INODE,2) = FULL
  YY = YP(INODE)/AA
  IF ((YY.LE.1.D0).AND.(YY.GE.-1.D0)) THEN
    VEL = Ha/(Ha-tanh(Ha))*(1.D0-DEXP(Ha*(YY-1.D0)))*
+      (1.D0+DEXP(-2.D0*Ha*YY))/(1.D0+DEXP(-2.D0*Ha))
  ELSE
    VEL = 0.D0
  END IF
  U(INODE,1) = VEL
  U(INODE,2) = VEL
102 CONTINUE

      ENDIF

C
C+++++ END OF USER AREA 5 +++++
C
      RETURN

C
      END
      SUBROUTINE USRBF(IPHASE,BX,BY,BZ,BPX,BPY,BPZ,U,V,W,P,VFRAC,DEN,
+      VIS,TE,ED,RS,T,H,RF,SCAL,XP,YP,ZP,VOL,AREA,VPOR,
+      ARPOR,WFACT,IPT,IBLK,IPVERT,IPNODN,IPFACN,IPNODEF,
+      IPNODEB,IPFACB,WORK,IWORK,CWORK)
C#####
C#####
C      Program for setting body forces
C#####
C#####
C
C*****
C
C      UTILITY SUBROUTINE FOR USER-SUPPLIED BODY FORCES
C
C      >>> IMPORTANT <<<
C      >>> <<<
C      >>> USERS MAY ONLY ADD OR ALTER PARTS OF THE SUBROUTINE WITHIN <<<
C      >>> THE DESIGNATED USER AREAS <<<
C
C*****
C
C      THIS SUBROUTINE IS CALLED BY THE FOLLOWING SUBROUTINES
C      BFCAL
C
C*****
C      CREATED
C      24/01/92 ADB
C      MODIFIED
C      03/06/92 PHA ADD PRECISION FLAG AND CHANGE IVERS TO 2
C      23/11/93 CSH EXPLICITLY DIMENSION IPVERT ETC.
C      03/02/94 PHA CHANGE FLOW3D TO CFDS-FLOW3D
C      03/03/94 FHW CORRECTION OF SPELLING MISTAKE

```

```

C      23/03/94  FHW  EXAMPLES COMMENTED OUT
C      09/08/94  NSW  CORRECT SPELLING
C      MOVE 'IF(IUSED.EQ.0) RETURN' OUT OF USER AREA
C      19/12/94  NSW  CHANGE FOR CFX-F3D
C      31/01/97  NSW  EXPLAIN USAGE IN MULTIPHASE FLOWS
C      02/07/97  NSW  UPDATE FOR CFX-4
C
C*****
C
C      SUBROUTINE ARGUMENTS
C
C      IPHASE - PHASE NUMBER
C
C      * BX      - X-COMPONENT OF VELOCITY-INDEPENDENT BODY FORCE
C      * BY      - Y-COMPONENT OF VELOCITY-INDEPENDENT BODY FORCE
C      * BZ      - Z-COMPONENT OF VELOCITY-INDEPENDENT BODY FORCE
C      * BPX     -
C      * BPY     - COMPONENTS OF LINEARISABLE BODY FORCES.
C      * BPZ     -
C
C      N.B. TOTAL BODY-FORCE IS GIVEN BY:
C
C      X-COMPONENT = BX + BPX*U
C      Y-COMPONENT = BY + BPY*V
C      Z-COMPONENT = BZ + BPZ*W
C
C      U      - U COMPONENT OF VELOCITY
C      V      - V COMPONENT OF VELOCITY
C      W      - W COMPONENT OF VELOCITY
C      P      - PRESSURE
C      VFRAC  - VOLUME FRACTION
C      DEN    - DENSITY OF FLUID
C      VIS    - VISCOSITY OF FLUID
C      TE     - TURBULENT KINETIC ENERGY
C      ED     - EPSILON
C      RS     - REYNOLD STRESSES
C      T      - TEMPERATURE
C      H      - ENTHALPY
C      SCAL   - SCALARS (THE FIRST 'NCONC' OF THESE ARE MASS FRACTIONS)
C      XP     - X COORDINATES OF CELL CENTRES
C      YP     - Y COORDINATES OF CELL CENTRES
C      ZP     - Z COORDINATES OF CELL CENTRES
C      VOL    - VOLUME OF CELLS
C      AREA   - AREA OF CELLS
C      VPOR   - POROUS VOLUME
C      ARPOR  - POROUS AREA
C      WFACT  - WEIGHT FACTORS
C
C      IPT    - 1D POINTER ARRAY
C      IBLK   - BLOCK SIZE INFORMATION
C      IPVERT - POINTER FROM CELL CENTERS TO 8 NEIGHBOURING VERTICES
C      IPNODN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING CELLS
C      IPFACN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING FACES
C      IPNODEF - POINTER FROM CELL FACES TO 2 NEIGHBOURING CELL CENTERS
C      IPNODEB - POINTER FROM BOUNDARY CENTERS TO CELL CENTERS
C      IPFACB - POINTER FROM BOUNDARY CENTERS TO BOUNDARY FACES
C
C      WORK   - REAL WORKSPACE ARRAY
C      IWORK  - INTEGER WORKSPACE ARRAY
C      CWORK  - CHARACTER WORKSPACE ARRAY
C
C      SUBROUTINE ARGUMENTS PRECEDED WITH A '*' ARE ARGUMENTS THAT MUST
C      BE SET BY THE USER IN THIS ROUTINE.

```

```

C
C NOTE THAT OTHER DATA MAY BE OBTAINED FROM CFX-4 USING THE
C ROUTINE GETADD, FOR FURTHER DETAILS SEE THE VERSION 4
C USER MANUAL.
C
C*****
C
C      DOUBLE PRECISION BX
C      DOUBLE PRECISION BY
C      DOUBLE PRECISION BZ
C      DOUBLE PRECISION BPX
C      DOUBLE PRECISION BPY
C      DOUBLE PRECISION BPZ
C      DOUBLE PRECISION U
C      DOUBLE PRECISION V
C      DOUBLE PRECISION W
C      DOUBLE PRECISION P
C      DOUBLE PRECISION VFRAC
C      DOUBLE PRECISION DEN
C      DOUBLE PRECISION VIS
C      DOUBLE PRECISION TE
C      DOUBLE PRECISION ED
C      DOUBLE PRECISION RS
C      DOUBLE PRECISION T
C      DOUBLE PRECISION H
C      DOUBLE PRECISION RF
C      DOUBLE PRECISION SCAL
C      DOUBLE PRECISION XP
C      DOUBLE PRECISION YP
C      DOUBLE PRECISION ZP
C      DOUBLE PRECISION VOL
C      DOUBLE PRECISION AREA
C      DOUBLE PRECISION VPOR
C      DOUBLE PRECISION ARPOR
C      DOUBLE PRECISION WFACT
C      DOUBLE PRECISION WORK
C      DOUBLE PRECISION SMALL
C      DOUBLE PRECISION SORMAX
C      DOUBLE PRECISION TIME
C      DOUBLE PRECISION DT
C      DOUBLE PRECISION DTINVF
C      DOUBLE PRECISION TPARM
C      LOGICAL LDEN, LVIS, LTURB, LTEMP, LBUOY, LSCAL, LCOMP, LRECT, LCYN, LAXIS,
+      LPOROS, LTRANS
C
C      CHARACTER*(*) CWORK
C
C+++++++ USER AREA 1 ++++++++
C---- AREA FOR USERS EXPLICITLY DECLARED VARIABLES
C      DOUBLE PRECISION Ha, AA, UFORCE, GAM, E, EFIELD, BFIELD
C
C+++++++ END OF USER AREA 1 ++++++++
C
C      COMMON /ALL/NBLOCK, NCELL, NBDY, NNODE, NFACE, NVERT, NDIM,
+      /ALLWRK/NRWS, NIWS, NCWS, IWRFRE, IWIFRE, IWCFRE, /ADDIMS/NPHASE,
+      NSCAL, NVAR, NPROP, NDVAR, NDPROP, NDXNN, NDGEOM, NDCOEF, NILIST,
+      NRLIST, NTOPOL, /CHKUSR/IVERS, IUCALL, IUSED, /DEVICE/NREAD,
+      NWRITE, NRDISK, NWDISK, /IDUM/ILEN, JLEN, /LOGIC/LDEN, LVIS,
+      LTURB, LTEMP, LBUOY, LSCAL, LCOMP, LRECT, LCYN, LAXIS, LPOROS,
+      LTRANS, /MLTGRD/MLEVEL, NLEVEL, ILEVEL, /SGLDBL/IFLGPR, ICHKPR,
+      /SPARM/SMALL, SORMAX, NITER, INDPRI, MAXIT, NODREF, NODMON,
+      /TRANSI/NSTEP, KSTEP, MF, INCORE, /TRANSR/TIME, DT, DTINVF, TPARM
C

```

```

C+++++ USER AREA 2 ++++++
C---- AREA FOR USERS TO DECLARE THEIR OWN COMMON BLOCKS
C      THESE SHOULD START WITH THE CHARACTERS 'UC' TO ENSURE
C      NO CONFLICT WITH NON-USER COMMON BLOCKS
C
C+++++ END OF USER AREA 2 ++++++
C
      DIMENSION BX(NCELL),BY(NCELL),BZ(NCELL),BPX(NCELL),BPY(NCELL),
+          BPZ(NCELL)
C
      DIMENSION U(NNODE,NPHASE),V(NNODE,NPHASE),W(NNODE,NPHASE),
+          P(NNODE,NPHASE),VFRAC(NNODE,NPHASE),DEN(NNODE,NPHASE),
+          VIS(NNODE,NPHASE),TE(NNODE,NPHASE),ED(NNODE,NPHASE),
+          RS(NNODE,NPHASE,*),T(NNODE,NPHASE),H(NNODE,NPHASE),
+          RF(NNODE,NPHASE,4),SCAL(NNODE,NPHASE,NSCAL)
C
      DIMENSION XP(NNODE),YP(NNODE),ZP(NNODE),VOL(NCELL),AREA(NFACE,3),
+          VPOR(NCELL),ARPOR(NFACE,3),WFACT(NFACE),IPT(*),
+          IBLK(5,NBLOCK),IPVERT(NCELL,8),IPNODN(NCELL,6),
+          IPFACN(NCELL,6),IPNODF(NFACE,4),IPNODB(NBDRY,4),
+          IPFACB(NBDRY),IWORK(*),WORK(*),CWORK(*)
C
C+++++ USER AREA 3 ++++++
C---- AREA FOR USERS TO DIMENSION THEIR ARRAYS
C
C---- AREA FOR USERS TO DEFINE DATA STATEMENTS
C
C+++++ END OF USER AREA 3 ++++++
C
C---- STATEMENT FUNCTION FOR ADDRESSING
      IP(I,J,K) = IPT((K-1)*ILEN*JLEN+ (J-1)*ILEN+I)
C
C----VERSION NUMBER OF USER ROUTINE AND PRECISION FLAG
C
      IVERS = 2
      ICHKPR = 2
C
C+++++ USER AREA 4 ++++++
C---- TO USE THIS USER ROUTINE FIRST SET IUSED=1
C
      IUSED = 1
C
C+++++ END OF USER AREA 4 ++++++
C
      IF (IUSED.EQ.0) RETURN
C
C---- FRONTEND CHECKING OF USER ROUTINE
      IF (IUCALL.EQ.0) RETURN
C
C+++++ USER AREA 5 ++++++
C
C THIS ROUTINE IS ENTERED REPEATEDLY FOR EACH PHASE IN A MULTIPHASE
C CALCULATION. BODY FORCES CAN BE SET FOR A PARTICULAR PHASE USING
C THE VARIABLE IPHASE. EG. IF (IPHASE.EQ.2) WOULD ALLOW BODY FORCES
C FOR THE SECOND PHASE.
C
C----ADD USER-DEFINED BODY FORCES.
C
C----EXAMPLE 1: LOCALISED MOMENTUM SOURCE, EG. PROPELLOR.
C
C----USE IPREC TO FIND ADDRESSES
C
      CALL IPREC('DUCT','BLOCK','CENTRES',IPT

```

```

C      +          , ILEN, JLEN, KLEN, CWORK, IWORK )
C
C      IST = ILEN/2 + 1
C      IFN = IST
C      JST = 1
C      JFN = JLEN/2
C
C      SMOM = 10.0
C      DO 103 K = 1, KLEN
C          DO 102 J = JST, JFN
C              DO 101 I = IST, IFN
C                  INODE = IP(I, J, K)
C                  BX(INODE) = BX(INODE) + SMOM
C 101          CONTINUE
C 102          CONTINUE
C 103 CONTINUE
C
C-----EXAMPLE 2: LOCALISED RESISTANCE
C
C-----USE IPREC TO FIND ADDRESSES
C
C      CALL IPREC('DUCT', 'BLOCK', 'CENTRES', IPT
C      +          , ILEN, JLEN, KLEN, CWORK, IWORK )
C
C      IST = ILEN/4 + 1
C      IFN = 3*ILEN/4
C      JST = 1
C      JFN = JLEN
C
C      RESIST = 1.0E+2
C      DO 203 K = 1, KLEN
C          DO 202 J = JST, JFN
C              DO 201 I = IST, IFN
C                  INODE = IP(I, J, K)
C                  BPX(INODE) = BPX(INODE) - RESIST
C                  BPY(INODE) = BPY(INODE) - RESIST
C                  BPZ(INODE) = BPZ(INODE) - RESIST
C 201          CONTINUE
C 202          CONTINUE
C 203 CONTINUE
C
C-----EXAMPLE 3: LOCALISED RESISTANCES (DISCONTINUOUS CHANGE)
C
C-----USE IPREC TO FIND ADDRESSES
C
C      CALL IPREC('DUCT', 'BLOCK', 'CENTRES', IPT
C      +          , ILEN, JLEN, KLEN, CWORK, IWORK )
C
C      IST1 = ILEN/4 + 1
C      IFN1 = IST1 + ILEN/4 - 1
C      IST2 = IFN1 + 1
C      IFN2 = ILEN - 1
C
C      DO 313 K = 1, KLEN
C          DO 312 J = 1, JLEN
C
C              RESIST = 1.0
C              DO 311 I = IST1, IFN1
C                  INODE = IP(I, J, K)
C                  BPX(INODE) = BPX(INODE) - RESIST
C                  BPY(INODE) = BPY(INODE) - RESIST
C                  BPZ(INODE) = BPZ(INODE) - RESIST
C 311          CONTINUE

```

```

C
C      RESIST = 10.0
C      DO 321 I = IST2,IFN2
C          INODE = IP(I,J,K)
C          BPX(INODE) = BPX(INODE) - RESIST
C          BPY(INODE) = BPY(INODE) - RESIST
C          BPZ(INODE) = BPZ(INODE) - RESIST
C 321      CONTINUE
C
C 312      CONTINUE
C 313      CONTINUE

C#####
C      Setting body force for phase 2 (liquid metal)
C#####

      IF (IPHASE.EQ.2) THEN

      E      = -1.D0
      GAM    = 2.D0

      CALL IPALL(' ',' ','BLOCK', 'CENTRES', IPT,NPT,CWORK,IWORK)

      DO 203 K = 1, NPT
          INODE = IPT(K)

          IF (XP(INODE).GT.0.D0) THEN
              BFIELD=1.D0-0.5D0*DTANH(GAM*XP(INODE))
          ELSE
              BFIELD = 1.D0
          ENDIF

          UFORCE      = -BFIELD**2.D0

C#####
C      In all internal cells body force is set equal to
C      Fx = UFORCE * U - E * BFIELD
C#####

          BPX(INODE) = BPX(INODE) + UFORCE
          BX(INODE)  = BX(INODE) - E*BFIELD
      203 CONTINUE

      ENDIF

C
C+++++ END OF USER AREA 5 +++++
C
      RETURN
C
      END
      SUBROUTINE USRCVG(U,V,W,P,VFRAC,DEN,VIS,TE,ED,RS,T,H,RF,SCAL,XP,
+          YP,ZP,VOL,AREA,VPOR,ARPOR,WFACT,CONV,IPT,IBLK,
+          IPVERT,IPNODN,IPFACN,IPNODF,IPNODB,IPFACB,CMETH,
+          MNSL,MXSL,RDFC,RESOR,URFVAR,LCONVG,WORK,IWORK,
+          CWORK)
C#####
C#####
C      Program for setting initial data at the first time step
C#####
C#####
C
C*****
C

```

```

C   THIS SUBROUTINE ALLOWS USERS TO MONITOR CONVERGENCE, ALTER
C   UNDER RELAXATION FACTORS, REDUCTION FACTORS ETC
C   AND WRITE SOLUTION DATA AS A FUNCTION OF ITERATION
C
C   >>> IMPORTANT                                     <<<
C   >>>                                               <<<
C   >>> USERS MAY ONLY ADD OR ALTER PARTS OF THE SUBROUTINE WITHIN <<<
C   >>> THE DESIGNATED USER AREAS                     <<<
C
C*****
C
C   THIS SUBROUTINE IS CALLED BY THE FOLLOWING SUBROUTINE
C       CUSR  CVGTST
C
C*****
C   CREATED
C       09/12/88  ADB
C   MODIFIED
C       08/08/91  IRH  NEW STRUCTURE
C       03/09/91  IRH  ADD CONV TO ARGUMENT LIST
C       23/09/91  IRH  ADD USEFUL COMMON BLOCKS
C       29/11/91  PHA  UPDATE CALLED BY COMMENT, ADD RF ARGUMENT,
C                   CHANGE LAST DIMENSION OF RS TO 6 AND IVERS TO 2
C       03/06/92  PHA  ADD PRECISION FLAG AND CHANGE IVERS TO 3
C       07/07/92  IRH  CORRECT EXAMPLE
C       23/11/93  CSH  EXPLICITLY DIMENSION IPVERT ETC.
C       03/02/94  PHA  CHANGE FLOW3D TO CFDS-FLOW3D
C       03/03/94  FHW  CORRECTION OF SPELLING MISTAKE
C       22/08/94  NSW  MOVE 'IF(IUSED.EQ.0) RETURN' OUT OF USER AREA
C       19/12/94  NSW  CHANGE FOR CFX-F3D
C       25/03/96  NSW  CORRECT MAXIMUM VELOCITY EXAMPLE
C       02/07/97  NSW  UPDATE FOR CFX-4
C
C*****
C
C   SUBROUTINE ARGUMENTS
C
C   U       - U COMPONENT OF VELOCITY
C   V       - V COMPONENT OF VELOCITY
C   W       - W COMPONENT OF VELOCITY
C   P       - PRESSURE
C   VFRAC   - VOLUME FRACTION
C   DEN     - DENSITY OF FLUID
C   VIS     - VISCOSITY OF FLUID
C   TE      - TURBULENT KINETIC ENERGY
C   ED      - EPSILON
C   RS      - REYNOLD STRESSES
C   T       - TEMPERATURE
C   H       - ENTHALPY
C   RF      - REYNOLD FLUXES
C   SCAL    - SCALARS (THE FIRST 'NCONC' OF THESE ARE MASS FRACTIONS)
C   XP      - X COORDINATES OF CELL CENTRES
C   YP      - Y COORDINATES OF CELL CENTRES
C   ZP      - Z COORDINATES OF CELL CENTRES
C   VOL     - VOLUME OF CELLS
C   AREA    - AREA OF CELLS
C   VPOR    - POROUS VOLUME
C   ARPOR   - POROUS AREA
C   WFACT   - WEIGHT FACTORS
C
C   IPT     - 1D POINTER ARRAY
C   IBLK    - BLOCK SIZE INFORMATION
C   IPVERT  - POINTER FROM CELL CENTERS TO 8 NEIGHBOURING VERTICES

```

```

C      IPNODN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING CELLS
C      IPFACN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING FACES
C      IPNODF - POINTER FROM CELL FACES TO 2 NEIGHBOURING CELL CENTERS
C      IPNODB - POINTER FROM BOUNDARY CENTERS TO CELL CENTERS
C      IPFACB - POINTER FROM BOUNDARY CENTERS TO BOUNDARY FACES
C
C      CMETH  - SOLUTION METHOD
C      MNLSL  - MINIMUM NUMBER OF SWEEPS
C      MXSL   - MAXIMUM NUMBER OF SWEEPS
C      RDFC   - REDUCTION FACTORS REQUIRED
C      RESOR  - NON LINEAR RESIDUALS
C      URFVAR - UNDER RELAXATION FACTORS
C      * LCONVG - LOGICAL CONVERGENCE FLAG
C
C      WORK   - REAL WORKSPACE ARRAY
C      IWORK  - INTEGER WORKSPACE ARRAY
C      CWORK  - CHARACTER WORKSPACE ARRAY
C
C      SUBROUTINE ARGUMENTS PRECEDED WITH A '*' ARE ARGUMENTS THAT MUST
C      BE SET BY THE USER IN THIS ROUTINE.
C
C      NOTE THAT OTHER DATA MAY BE OBTAINED FROM CFX-4 USING THE
C      ROUTINE GETADD, FOR FURTHER DETAILS SEE THE VERSION 4
C      USER MANUAL.
C
C*****
C
      DOUBLE PRECISION U
      DOUBLE PRECISION V
      DOUBLE PRECISION W
      DOUBLE PRECISION P
      DOUBLE PRECISION VFRAC
      DOUBLE PRECISION DEN
      DOUBLE PRECISION VIS
      DOUBLE PRECISION TE
      DOUBLE PRECISION ED
      DOUBLE PRECISION RS
      DOUBLE PRECISION T
      DOUBLE PRECISION H
      DOUBLE PRECISION RF
      DOUBLE PRECISION SCAL
      DOUBLE PRECISION XP
      DOUBLE PRECISION YP
      DOUBLE PRECISION ZP
      DOUBLE PRECISION VOL
      DOUBLE PRECISION AREA
      DOUBLE PRECISION VPOR
      DOUBLE PRECISION ARPOR
      DOUBLE PRECISION WFACT
      DOUBLE PRECISION CONV
      DOUBLE PRECISION RDFC
      DOUBLE PRECISION RESOR
      DOUBLE PRECISION URFVAR
      DOUBLE PRECISION WORK
      DOUBLE PRECISION SMALL
      DOUBLE PRECISION SORMAX
      DOUBLE PRECISION TIME
      DOUBLE PRECISION DT
      DOUBLE PRECISION DTINVF
      DOUBLE PRECISION TPARM
      LOGICAL LDEN, LVIS, LTURB, LTEMP, LBUOY, LSCAL, LCOMP, LRECT, LCYN, LAXIS,
+      LPOROS, LTRANS
      LOGICAL LCONVG

```

```

C
      CHARACTER*(*) CMETH,CWORK
C
C+++++ USER AREA 1 ++++++
C---- AREA FOR USERS EXPLICITLY DECLARED VARIABLES
C
C+++++ END OF USER AREA 1 ++++++
C
      COMMON /ALL/NBLOCK,NCELL,NBDRY,NNODE,NFACE,NVERT,NDIM,
+           /ALLWRK/NRWS,NIWS,NCWS,IWRFRE,IWIFRE,IWCFRE,/ADDIMS/NPHASE,
+           NSCAL,NVAR,NPROP,NDVAR,NDPROP,NDXNN,NDGEOM,NDCOEF,NILIST,
+           NRLIST,NTOPOL,/CHKUSR/IVERS,IUCALL,IUSED,/DEVICE/NREAD,
+           NWRITE,NRDISK,NWDISK,/IDUM/ILEN,JLEN,/LOGIC/LDEN,LVIS,
+           LTRANS,/MLTGRD/MLEVEL,NLEVEL,ILEVEL,/RESID/IRESID,NRESID,
+           /SGLDBL/IFLGPR,ICHPKR,/SPARM/SMALL,SORMAX,NITER,INDPRI,
+           MAXIT,NODREF,NODMON,/TRANSI/NSTEP,KSTEP,MF,INCORE,
+           /TRANR/TIME,DT,DTINV,TPARM
C
C+++++ USER AREA 2 ++++++
C---- AREA FOR USERS TO DECLARE THEIR OWN COMMON BLOCKS
C      THESE SHOULD START WITH THE CHARACTERS 'UC' TO ENSURE
C      NO CONFLICT WITH NON-USER COMMON BLOCKS
C
C---- COMMON BLOCK FOR EXAMPLE IN USER AREA 6
C      COMMON /UC1/ VELOLD
C
C+++++ END OF USER AREA 2 ++++++
C
      DIMENSION CMETH(NVAR,NPHASE),MNSL(NVAR,NPHASE),MXSL(NVAR,NPHASE),
+           RDFC(NVAR,NPHASE),RESOR(NVAR,NPHASE),URFVAR(NVAR,NPHASE)
      DIMENSION U(NNODE,NPHASE),V(NNODE,NPHASE),W(NNODE,NPHASE),
+           P(NNODE,NPHASE),VFRAC(NNODE,NPHASE),DEN(NNODE,NPHASE),
+           VIS(NNODE,NPHASE),TE(NNODE,NPHASE),ED(NNODE,NPHASE),
+           RS(NNODE,NPHASE,6),T(NNODE,NPHASE),H(NNODE,NPHASE),
+           RF(NNODE,NPHASE,4),SCAL(NNODE,NPHASE,NSCAL)
      DIMENSION XP(NNODE),YP(NNODE),ZP(NNODE),VOL(NCELL),AREA(NFACE,3),
+           VPOR(NCELL),ARPOR(NFACE,3),WFACT(NFACE),
+           CONV(NFACE,NPHASE),IPT(*),IBLK(5,NBLOCK),
+           IPVERT(NCELL,8),IPNODN(NCELL,6),IPFACN(NCELL,6),
+           IPNODF(NFACE,4),IPNODB(NBDRY,4),IPFACB(NBDRY),IWORK(*),
+           WORK(*),CWORK(*)
C
C+++++ USER AREA 3 ++++++
C---- AREA FOR USERS TO DIMENSION THEIR ARRAYS
C
C---- AREA FOR USERS TO DEFINE DATA STATEMENTS
C
C+++++ END OF USER AREA 3 ++++++
C
C---- STATEMENT FUNCTION FOR ADDRESSING
      IP(I,J,K) = IPT((K-1)*ILEN*JLEN+ (J-1)*ILEN+I)
C
C----VERSION NUMBER OF USER ROUTINE AND PRECISION FLAG
C
      IVERS = 3
      ICHKPR = 2
C
C+++++ USER AREA 4 ++++++
C---- TO USE THIS USER ROUTINE FIRST SET IUSED=1
C
      IUSED = 1
C

```

```

C+++++ END OF USER AREA 4 ++++++
C
C      IF (IUSED.EQ.0) RETURN
C
C----- FRONTEND CHECKING OF USER ROUTINE
C      IF (IUCALL.EQ.0) RETURN
C
C+++++ USER AREA 5 ++++++
C
C----- EXAMPLE: (TEST ON MAX OF MASS RESIDUAL AND ENTHALPY RESIDUAL)
C
C      CALL GETVAR('USRCVG','P      ',IPRES)
C      CALL GETVAR('USRCVG','H      ',IH)
C
C      URESM=0.0
C      DO 10 IPHASE=1,NPHASE
C      URESM=MAX(URES,RESOR(IPRES,IPHASE),RESOR(IH,IPHASE))
C 10   CONTINUE
C
C      LCONVG = URESM .LT. 1.0E-5
C
C-----END OF EXAMPLE
C
C+++++ END OF USER AREA 5 ++++++
C
C+++++ USER AREA 6 ++++++
C
C----- EXAMPLE: MONITOR CHANGE IN MAXIMUM VELOCITY
C              ADJUST UNDER RELAXATION ACCORDINGLY
C
C      VELMAX=0.0
C      DO 20 IPHASE=1,NPHASE
C      USE IPALL TO FIND 1D ADDRESSES OF ALL CELL CENTRES
C      CALL IPALL('*', '*', 'BLOCK', 'CENTRES', IPT, NPT, CWORK, IWORK)
C      LOOP OVER ALL CELL CENTRE LOCATIONS IN FLOW DOMAIN
C      DO 30 I=1, NPT
C      USE ARRAY IPT TO GET ADDRESS
C      INODE=IPT(I)
C      VELMAX=MAX(VELMAX, ABS(U(INODE, IPHASE)), ABS(V(INODE, IPHASE)),
C      +          ABS(W(INODE, IPHASE)))
C 30   CONTINUE
C 20   CONTINUE
C
C      IF (NITER.GT.1) THEN
C      DVEL=(VELMAX-VELOLD)/VELMAX
C      URFMIN=0.01
C      URFMAX=0.8
C      URF=(1.0-DVEL)*URFMAX+DVEL*URFMIN
C      WRITE(NWRITE,100)NITER,DVEL,URF
C      CALL GETVAR('USRCVG','U      ',IU)
C      CALL GETVAR('USRCVG','V      ',IV)
C      CALL GETVAR('USRCVG','W      ',IW)
C      DO 40 IPHASE=1,NPHASE
C      URFVAR(IU,IPHASE)=URF
C      URFVAR(IV,IPHASE)=URF
C      URFVAR(IW,IPHASE)=URF
C 40   CONTINUE
C      ENDIF
C
C      VELOLD=VELMAX
C
C-----END OF EXAMPLE

```

```

C#####
C      Setting convergence criterion
C#####

C#####
C      Convergence test will first be performed after 150th iteration
C#####

      IF (NITER.GT.150) THEN

      CALL GETVAR('USRCVG','P      ',IPRES)
      CALL GETVAR('USRCVG','U      ',IU)
      CALL GETVAR('USRCVG','V      ',IV)
      CALL GETVAR('USRCVG','VFRAC ',IVFRAC)

C#####
C      Maximum values of flow variables are calculated
C#####

      VELMAX=0.0D0
      VFRACMAX=0.0D0
      PMAX=0.0D0
      DO 20 IPHASE=1,NPHASE
        CALL IPALL('*','*','BLOCK','CENTRES',IPT,NPT,CWORK,IWORK)
        DO 30 I=1,NPT
          INODE=IPT(I)
          VELMAX=MAX(VELMAX,ABS(U(INODE,IPHASE)),ABS(V(INODE,IPHASE)))
          VFRACMAX=MAX(VFRACMAX,ABS(VFRAC(INODE,IPHASE)))
          PMAX=MAX(PMAX,ABS(P(INODE,IPHASE)))
        30 CONTINUE
      20 CONTINUE

C#####
C      Maximum values of residuals of flow variables are calculated
C#####

      VELRESM=0.0D0
      VFRACRESM = 0.0D0
      PRESM = 0.0D0
      DO 10 IPHASE=1,NPHASE
        VELRESM=MAX(VELRESM,RESOR(IU,IPHASE),RESOR(IV,IPHASE))
        VFRACRESM=MAX(VFRACRESM,RESOR(IVFRAC,IPHASE))
        PRESM=MAX(PRESM,RESOR(IPRES,IPHASE))
      10 CONTINUE

C#####
C      If ratio of RES/MAX. VALUE is less than 1E-7, problem converged
C#####

      LCONVG = (VELRESM/VELMAX.LT.1.0E-7).AND.(PRESM/PMAX.LT.1.0E-7)
+           .AND.(VFRACRESM/VFRACMAX.LT.1.0E-7)

      IF (LCONVG) THEN
        write(6,*) '##### Converged! #####'
        write(6,*) '##### VMAX = ',VELMAX,' PMAX = ',PMAX,
+           ' VFRACMAX = ',VFRACMAX
        write(6,*) '##### VRES = ',VELRESM,' PRES = ',PRESM,
+           ' VFRACRES = ',VFRACRESM
      ENDIF

      ENDIF
C*****

```

```

C
C+++++ END OF USER AREA 6 ++++++
C
      RETURN
C
      END
      SUBROUTINE USRBCS (VARBCS, VARAMB, A, B, C, ACND, BCND, CCND, IWGVEL,
+                       NDVWAL, FLOUT, NLABEL, NSTART, NEND, NCST, NCEN, U, V, W,
+                       P, VFRAC, DEN, VIS, TE, ED, RS, T, H, RF, SCAL, XP, YP, ZP,
+                       VOL, AREA, VPOR, ARPOR, WFACT, IPT, IBLK, IPVERT,
+                       IPNODN, IPFACN, IPNODF, IPNODB, IPFACB, WORK, IWORK,
+                       CWORK)
C#####
C#####
C      Program for setting boundary conditions
C#####
C#####
C
C*****
C
C      USER ROUTINE TO SET REALS AT BOUNDARIES.
C
C      >>> IMPORTANT                                     <<<
C      >>>                                             <<<
C      >>> USERS MAY ONLY ADD OR ALTER PARTS OF THE SUBROUTINE WITHIN <<<
C      >>> THE DESIGNATED USER AREAS                               <<<
C
C*****
C
C      THIS SUBROUTINE IS CALLED BY THE FOLLOWING SUBROUTINE
C      CUSR  SRLIST
C
C*****
C      CREATED
C      30/11/88 ADB
C      MODIFIED
C      08/09/90 ADB  RESTRUCTURED FOR USER-FRIENDLINESS.
C      10/08/91 IRH  FURTHER RESTRUCTURING ADD ACND BCND CCND
C      22/09/91 IRH  CHANGE ICALL TO IUCALL + ADD /SPARM/
C      10/03/92 PHA  UPDATE CALLED BY COMMENT, ADD RF ARGUMENT,
C                   CHANGE LAST DIMENSION OF RS TO 6 AND IVERS TO 2
C      03/06/92 PHA  ADD PRECISION FLAG AND CHANGE IVERS TO 3
C      30/06/92 NSW  INCLUDE FLAG FOR CALLING BY ITERATION
C                   INSERT EXTRA COMMENTS
C      03/08/92 NSW  MODIFY DIMENSION STATEMENTS FOR VAX
C      21/12/92 CSH  INCREASE IVERS TO 4
C      02/08/93 NSW  INCORRECT AND MISLEADING COMMENT REMOVED
C      05/11/93 NSW  INDICATE USE OF FLOUT IN MULTIPHASE FLOWS
C      23/11/93 CSH  EXPLICITLY DIMENSION IPVERT ETC.
C      01/02/94 NSW  SET VARIABLE POINTERS IN WALL EXAMPLE.
C                   CHANGE FLOW3D TO CFDS-FLOW3D.
C                   MODIFY MULTIPHASE MASS FLOW BOUNDARY TREATMENT.
C      03/03/94 FHW  CORRECTION OF SPELLING MISTAKE
C      02/07/94 BAS  SLIDING GRIDS - ADD NEW ARGUMENT IWGVEL
C                   TO ALLOW VARIANTS OF TRANSIENT-GRID WALL BC
C                   CHANGE VERSION NUMBER TO 5
C      09/08/94 NSW  CORRECT SPELLING
C                   MOVE 'IF(IUSED.EQ.0) RETURN' OUT OF USER AREA
C      19/12/94 NSW  CHANGE FOR CFX-F3D
C      02/02/95 NSW  CHANGE COMMON /IMFBMP/
C      02/06/97 NSW  MAKE EXAMPLE MORE LOGICAL
C      02/07/97 NSW  UPDATE FOR CFX-4
C      08/09/98 NSW  CORRECT SIZE OF WALL ARRAY IN COMMENT

```

```

C      22/05/00 NSW INITIALISE IUBCSF
C
C*****
C
C      SUBROUTINE ARGUMENTS
C
C      VARBCS - REAL BOUNDARY CONDITIONS
C      VARAMB - AMBIENT VALUE OF VARIABLES
C      A      - COEFFICIENT IN WALL BOUNDARY CONDITION
C      B      - COEFFICIENT IN WALL BOUNDARY CONDITION
C      C      - COEFFICIENT IN WALL BOUNDARY CONDITION
C      ACND   - COEFFICIENT IN CONDUCTING WALL BOUNDARY CONDITION
C      BCND   - COEFFICIENT IN CONDUCTING WALL BOUNDARY CONDITION
C      CCND   - COEFFICIENT IN CONDUCTING WALL BOUNDARY CONDITION
C      IWGVEL - USAGE OF INPUT VELOCITIES (0 = AS IS,1 = ADD GRID MOTION)
C      NDVWAL - FIRST DIMENSION OF ARRAY IWGVEL
C      FLOUT  - MASS FLOW/FRACTIONAL MASS FLOW
C      NLABEL - NUMBER OF DISTINCT OUTLETS
C      NSTART - ARRAY POINTER
C      NEND   - ARRAY POINTER
C      NCST   - ARRAY POINTER
C      NCEN   - ARRAY POINTER
C      U      - U COMPONENT OF VELOCITY
C      V      - V COMPONENT OF VELOCITY
C      W      - W COMPONENT OF VELOCITY
C      P      - PRESSURE
C      VFRAC  - VOLUME FRACTION
C      DEN    - DENSITY OF FLUID
C      VIS    - VISCOSITY OF FLUID
C      TE     - TURBULENT KINETIC ENERGY
C      ED     - EPSILON
C      RS     - REYNOLD STRESSES
C      T      - TEMPERATURE
C      H      - ENTHALPY
C      RF     - REYNOLD FLUXES
C      SCAL   - SCALARS (THE FIRST 'NCONC' OF THESE ARE MASS FRACTIONS)
C      XP     - X COORDINATES OF CELL CENTRES
C      YP     - Y COORDINATES OF CELL CENTRES
C      ZP     - Z COORDINATES OF CELL CENTRES
C      VOL    - VOLUME OF CELLS
C      AREA   - AREA OF CELLS
C      VPOR   - POROUS VOLUME
C      ARPOR  - POROUS AREA
C      WFACT  - WEIGHT FACTORS
C
C      IPT    - 1D POINTER ARRAY
C      IBLK   - BLOCK SIZE INFORMATION
C      IPVERT - POINTER FROM CELL CENTERS TO 8 NEIGHBOURING VERTICES
C      IPNODN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING CELLS
C      IPFACN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING FACES
C      IPNODF - POINTER FROM CELL FACES TO 2 NEIGHBOURING CELL CENTERS
C      IPNOB  - POINTER FROM BOUNDARY CENTERS TO CELL CENTERS
C      IPFACB - POINTER TO NODES FROM BOUNDARY FACES
C
C      WORK   - REAL WORKSPACE ARRAY
C      IWORK  - INTEGER WORKSPACE ARRAY
C      CWORK  - CHARACTER WORKSPACE ARRAY
C
C      SUBROUTINE ARGUMENTS PRECEDED WITH A '*' ARE ARGUMENTS THAT MUST
C      BE SET BY THE USER IN THIS ROUTINE.
C
C      NOTE THAT OTHER DATA MAY BE OBTAINED FROM CFX-4 USING THE
C      ROUTINE GETADD, FOR FURTHER DETAILS SEE THE VERSION 4

```

```

C   USER MANUAL.
C
C*****
      DOUBLE PRECISION VARBCS
      DOUBLE PRECISION VARAMB
      DOUBLE PRECISION A
      DOUBLE PRECISION B
      DOUBLE PRECISION C
      DOUBLE PRECISION ACND
      DOUBLE PRECISION BCND
      DOUBLE PRECISION CCND
      DOUBLE PRECISION FLOUT
      DOUBLE PRECISION U
      DOUBLE PRECISION V
      DOUBLE PRECISION W
      DOUBLE PRECISION P
      DOUBLE PRECISION VFRAC
      DOUBLE PRECISION DEN
      DOUBLE PRECISION VIS
      DOUBLE PRECISION TE
      DOUBLE PRECISION ED
      DOUBLE PRECISION RS
      DOUBLE PRECISION T
      DOUBLE PRECISION H
      DOUBLE PRECISION RF
      DOUBLE PRECISION SCAL
      DOUBLE PRECISION XP
      DOUBLE PRECISION YP
      DOUBLE PRECISION ZP
      DOUBLE PRECISION VOL
      DOUBLE PRECISION AREA
      DOUBLE PRECISION VPOR
      DOUBLE PRECISION ARPOR
      DOUBLE PRECISION WFACT
      DOUBLE PRECISION WORK
      DOUBLE PRECISION SMALL
      DOUBLE PRECISION SORMAX
      DOUBLE PRECISION TIME
      DOUBLE PRECISION DT
      DOUBLE PRECISION DTINVF
      DOUBLE PRECISION TPARM
      LOGICAL LDEN, LVIS, LTURB, LTEMP, LBUOY, LSCAL, LCOMP, LRECT, LCYN, LAXIS,
+       LPOROS, LTRANS
C
      CHARACTER*(*) CWORK
C
C+++++++ USER AREA 1 ++++++++
C---- AREA FOR USERS EXPLICITLY DECLARED VARIABLES
      DOUBLE PRECISION AA, EMPTY, FULL, HA, YY, VEL
C
C+++++++ END OF USER AREA 1 ++++++++
C
      COMMON /ALL/NBLOCK, NCELL, NBDRY, NNODE, NFACE, NVERT, NDIM,
+       /ALLWRK/NRWS, NIWS, NCWS, IWRFRE, IWIFRE, IWCFRE, /ADDIMS/NPHASE,
+       NSCAL, NVAR, NPROP, NDVAR, NDPROP, NDXNN, NDGEOM, NDCOEF, NILIST,
+       NRLIST, NTOPOL, /BCSOUT/IFLOUT, /CHKUSR/IVERS, IUCALL, IUSED,
+       /DEVICE/NREAD, NWRITE, NRDISK, NWDISK, /IDUM/ILEN, JLEN,
+       /IMFBMP/IMFBMP, JMFMBMP, /LOGIC/LDEN, LVIS, LTURB, LTEMP, LBUOY,
+       LSCAL, LCOMP, LRECT, LCYN, LAXIS, LPOROS, LTRANS, /MLTGRD/MLEVEL,
+       NLEVEL, ILEVEL, /SGLDBL/IFLGPR, ICHKPR, /SPARM/SMALL, SORMAX,
+       NITER, INDPRI, MAXIT, NODREF, NODMON, /TRANSI/NSTEP, KSTEP, MF,
+       INCORE, /TRANSR/TIME, DT, DTINVF, TPARM, /UBCSFL/IUBCSF
C

```

```

C+++++ USER AREA 2 ++++++
C---- AREA FOR USERS TO DECLARE THEIR OWN COMMON BLOCKS
C   THESE SHOULD START WITH THE CHARACTERS 'UC' TO ENSURE
C   NO CONFLICT WITH NON-USER COMMON BLOCKS
C
C+++++ END OF USER AREA 2 ++++++
C
      DIMENSION VARBCS(NVAR,NPHASE,NCELL+1:NNODE),VARAMB(NVAR,NPHASE),
+          A(4+NSCAL,NPHASE,NSTART:*),B(4+NSCAL,NPHASE,NSTART:*),
+          C(4+NSCAL,NPHASE,NSTART:*),FLOUT(*),ACND(NCST:*),
+          BCND(NCST:*),CCND(NCST:*),IWGVEL(NDVWAL,NPHASE)
C
      DIMENSION U(NNODE,NPHASE),V(NNODE,NPHASE),W(NNODE,NPHASE),
+          P(NNODE,NPHASE),VFRAC(NNODE,NPHASE),DEN(NNODE,NPHASE),
+          VIS(NNODE,NPHASE),TE(NNODE,NPHASE),ED(NNODE,NPHASE),
+          RS(NNODE,NPHASE,6),T(NNODE,NPHASE),H(NNODE,NPHASE),
+          RF(NNODE,NPHASE,4),SCAL(NNODE,NPHASE,NSCAL)
C
      DIMENSION XP(NNODE),YP(NNODE),ZP(NNODE),VOL(NCELL),AREA(NFACE,3),
+          VPOR(NCELL),ARPOR(NFACE,3),WFACT(NFACE),IPT(*),
+          IBLK(5,NBLOCK),IPVERT(NCELL,8),IPNODN(NCELL,6),
+          IPFACN(NCELL,6),IPNODF(NFACE,4),IPNOB(NBDRY,4),
+          IPFACB(NBDRY),IWORK(*),WORK(*),CWORK(*)
C
C+++++ USER AREA 3 ++++++
C---- AREA FOR USERS TO DIMENSION THEIR ARRAYS
C
C---- AREA FOR USERS TO DEFINE DATA STATEMENTS
C
C+++++ END OF USER AREA 3 ++++++
C
C---- STATEMENT FUNCTION FOR ADDRESSING
      IP(I,J,K) = IPT((K-1)*ILEN*JLEN+ (J-1)*ILEN+I)
C
C----VERSION NUMBER OF USER ROUTINE AND PRECISION FLAG
C
      IVERS = 5
      ICHKPR = 2
C
C+++++ USER AREA 4 ++++++
C---- TO USE THIS USER ROUTINE FIRST SET IUSED=1
C   AND SET IUBCSF FLAG:
C   BOUNDARY CONDITIONS NOT CHANGING           IUBCSF=0
C   BOUNDARY CONDITIONS CHANGING WITH ITERATION IUBCSF=1
C   BOUNDARY CONDITIONS CHANGING WITH TIME      IUBCSF=2
C   BOUNDARY CONDITIONS CHANGING WITH TIME AND ITERATION IUBCSF=3
C
      IUSED = 1
      IUBCSF = 0
C
C+++++ END OF USER AREA 4 ++++++
C
      IF (IUSED.EQ.0) RETURN
C
C---- FRONTEND CHECKING OF USER ROUTINE
      IF (IUCALL.EQ.0) RETURN
C
C+++++ USER AREA 5 ++++++
C
C---- AREA FOR SETTING VALUES AT INLETS, PRESSURE BOUNDARIES
C   AND OUTLETS. (NOTE THAT THE MASS FLOW AT OUTLETS IS
C   SPECIFIED IN USER AREA 7)
C

```

```

C     IF USING A REYNOLDS STRESS OR FLUX MODEL, NOTE THAT AT INLETS
C     IT IS IMPORTANT THAT THE USER SETS ALL COMPONENTS OF THE
C     REYNOLDS STRESS AND FLUX AND THE TURBULENT KINETIC ENERGY
C     AS WELL AS THE ENERGY DISSIPATION RATE.
C
C     SET THE VALUES IN VARBCS(NVAR,NPHASE,ILEN,JLEN,KLEN)
C
C---- EXAMPLE: SETTING A LINEAR T PROFILE ON INLET PATCH 'ENTRANCE'
C           LEAVE OTHER VARIABLES AS SET IN COMMAND LANGUAGE
C
C-- INTERROGATE GETVAR FOR VARIABLE NUMBERS.
C
C     CALL GETVAR('USRBCS','T      ',IT)
C
C SET IPHS = 1 FOR SINGLE PHASE FLOW.
C
C     IPHS = 1
C
C USE IPREC TO FIND ADDRESSES
C
C     CALL IPREC('ENTRANCE','PATCH','CENTRES',IPT,ILEN,JLEN,KLEN,
C +           CWORK,IWORK)
C
C     XMAX=2.0
C     XMIN=1.0
C     TMAX=300.0
C     TMIN=250.0
C LOOP OVER PATCH
C     DO 103 K = 1, KLEN
C         DO 102 J = 1, JLEN
C             DO 101 I = 1, ILEN
C USE STATEMENT FUNCTION IP TO GET ADDRESSES
C             INODE = IP(I,J,K)
C SET VARBCS
C             F=(XP(INODE)-XMIN)/(XMAX-XMIN)
C             VARBCS(IT,IPHS,INODE) = F*TMAX + (1.0-F)*TMIN
C 101     CONTINUE
C 102     CONTINUE
C 103     CONTINUE
C
C----END OF EXAMPLE

C#####
C     Setting Hartmann profile at the entrance to the duct
C#####

C     CALL GETVAR('USRBCS','U      ',IU)
C     CALL GETVAR('USRBCS','VFRAC ',IVFRAC)
C     CALL IPALL('ENTRANCE','*','PATCH','CENTRES',IPT,NPT,CWORK,IWORK)

C     IPHS = 1

C     AA      = 1.D+00
C     HA      = 200.0D0

C     FULL = 1.0D0
C     EMPTY = 1.D-10

C     CALL IPALL('ENTRANCE','*','PATCH','CENTRES',IPT
C +           ,NPT,CWORK,IWORK)

C     DO 110 K = 1, NPT

```

```

      INODE = IPT(K)
      YY    = YP(INODE)/AA

      IF ((YY.LE.1.D0).AND.(YY.GE.-1.D0)) THEN

          VEL = Ha/(Ha-tanh(Ha))*(1.D0-DEXP(Ha*(YY-1.D0))*
+          (1.D0+DEXP(-2.D0*Ha*YY))/(1.D0+DEXP(-2.D0*Ha)))
      ELSE
          VEL = 0.D0
      END IF

C#####
C  Volume fraction of metal is set to 1, velocity to Hartmann profile
C#####

      VARBCS(IVFRAC,1,INODE) = EMPTY
      VARBCS(IVFRAC,2,INODE) = FULL
      VARBCS(IU,1,INODE) = VEL
      VARBCS(IU,2,INODE) = VEL

      110 CONTINUE

C*****

C
C+++++ END OF USER AREA 5 +++++
C
C+++++ USER AREA 6 +++++
C
C---- AREA FOR SETTING VALUES AT WALLS
C
C      USE A(4+NSCAL,NPHASE,NNODE)
C      WHERE NSCAL = NO. OF SCALARS, AND NPHASE = NO. OF PHASES.
C
C      THE CONVENTION FOR VARIABLE NUMBERS IS DIFFERENT IN THIS ROUTINE
C      FROM THAT IN THE REST OF THE PROGRAM. IT IS:
C
C      IU = 1, IV = 2 , IW = 3, IT = 4, IS = 5
C
C---- EXAMPLE: SETTING FREE SLIP BOUNDARY CONDITIONS AT ALL WALLS
C              AND SETTING T=300.0 AND SCALAR1 AND SCALAR2 =0.0
C              ON WALL1. SET T=400.0 ON CONDUCTING SOLID BOUNDARY WALL2
C
C-- SET POINTERS
C
C      IU = 1
C      IV = 2
C      IW = 3
C      IT = 4
C      IS = 5
C
C-- SET IPHS = 1 FOR SINGLE PHASE FLOW.
C
C      IPHS = 1
C
C      USE IPALL TO FIND 1D ADDRESSES OF A GROUP OF PATCH CENTRES
C
C      CALL IPALL(' ', 'WALL', 'PATCH', 'CENTRES', IPT,NPT,CWORK,IWORK)
C
C      LOOP OVER GROUP OF PATCHES
C      DO 200 I=1,NPT
C      USE ARRAY IPT TO GET ADDRESS

```

```

C      INODE=IPT(I)
C      A(IU,IPHS,INODE) = 0.0
C      B(IU,IPHS,INODE) = 1.0
C      C(IU,IPHS,INODE) = 0.0
C
C      A(IV,IPHS,INODE) = 0.0
C      B(IV,IPHS,INODE) = 1.0
C      C(IV,IPHS,INODE) = 0.0
C
C      A(IW,IPHS,INODE) = 0.0
C      B(IW,IPHS,INODE) = 1.0
C      C(IW,IPHS,INODE) = 0.0
C 200  CONTINUE
C
C  USE IPREC TO FIND ADDRESSES OF SINGLE PATCH
C
C      CALL IPREC('WALL1','PATCH','CENTRES',IPT,ILEN,JLEN,KLEN,
C      +          CWORK,IWORK)
C  LOOP OVER PATCH
C      DO 203 K = 1, KLEN
C          DO 202 J = 1, JLEN
C              DO 201 I = 1, ILEN
C  USE STATEMENT FUNCTION IP TO GET ADDRESSES
C          INODE = IP(I,J,K)
C
C          A(IT,IPHS,INODE) = 1.0
C          B(IT,IPHS,INODE) = 0.0
C          C(IT,IPHS,INODE) = 300.0
C
C          A(IS,IPHS,INODE) = 1.0
C          B(IS,IPHS,INODE) = 0.0
C          C(IS,IPHS,INODE) = 0.0
C
C          A(IS+1,IPHS,INODE) = 1.0
C          B(IS+1,IPHS,INODE) = 0.0
C          C(IS+1,IPHS,INODE) = 0.0
C
C 201      CONTINUE
C 202      CONTINUE
C 203      CONTINUE
C
C  USE IPALL TO FIND 1D ADDRESSES OF A GROUP OF PATCH CENTRES
C
C      CALL IPALL('WALL2','*','PATCH','CENTRES',IPT,NPT,CWORK,IWORK)
C
C  LOOP OVER GROUP OF PATCHES
C      DO 300 I=1,NPT
C  USE ARRAY IPT TO GET ADDRESS
C          INODE=IPT(I)
C          ACND(INODE) = 1.0
C          BCND(INODE) = 0.0
C          CCND(INODE) = 400.0
C 300  CONTINUE
C
C-----END OF EXAMPLE
C
C+++++END OF USER AREA 6 +++++
C
C+++++ USER AREA 7 +++++
C
C----- DEFINE FLOW AT OUTLETS (MASS FLOW BOUNDARIES)
C      (TO TEMPERATURES AND SCALARS AT MASS FLOW BOUNDARIES USE

```

```

C      USER AREA 5)
C
C      SET PARAMETER IFLOUT:
C      IFLOUT = 1 ==> MASS FLOW SPECIFIED AT LABELLED OUTLETS.
C      IFLOUT = 2 ==> FRACTIONAL MASS FLOW SPECIFIED AT LABELLED OUTLETS
C      IFLOUT = 2
C
C      SET OUTLET FLOW RATES:
C      FLOUT(LABEL) = MASS FLOW OUT OF OUTLETS LABELLED LABEL (IFLOUT=1).
C      FLOUT(LABEL) = FRACTIONAL MASS FLOW OUT OF OUTLETS LABELLED LABEL
C                      (IFLOUT=2).
C      FOR MULTIPHASE FLOWS IT IS NECESSARY TO SET
C      EITHER
C                      FLOUT(LABEL) = TOTAL MASS FLOW
C                      IFLOUT = 1
C                      IMFBMP = 0
C      OR
C                      FLOUT(LABEL + (IPHASE-1)*NLABEL) FOR EACH PHASE
C                      IFLOUT = 1 OR 2
C                      IMFBMP = 1
C
C---- EXAMPLE: EQUIDISTRIBUTION OF FRACTIONAL MASS FLOW AMONGST OUTLETS
C
C      IFLOUT=2
C      FRAC = 1.0 / MAX( 1.0, FLOAT(NLABEL) )
C      DO 300 ILABEL = 1, NLABEL
C          FLOUT(ILABEL) = FRAC
C300  CONTINUE
C
C----END OF EXAMPLE
C
C+++++ END OF USER AREA 7 +++++
C
C      RETURN
C
C      END
C      SUBROUTINE USRTRN(U,V,W,P,VFRAC,DEN,VIS,TE,ED,RS,T,H,RF,SCAL,XP,
C      +                YP,ZP,VOL,AREA,VPOR,ARPOR,WFACT,CONV,IPT,IBLK,
C      +                IPVERT,IPNODN,IPFACN,IPNODF,IPNODB,IPFACB,WORK,
C      +                IWORK,CWORK)
C#####
C#####
C      Program for calculation additional data sets
C#####
C#####
C*****
C
C      USER SUBROUTINE TO ALLOW USERS TO MODIFY OR MONITOR THE SOLUTION AT
C      THE END OF EACH TIME STEP
C      THIS SUBROUTINE IS CALLED BEFORE THE START OF THE RUN AS WELL AS AT
C      THE END OF EACH TIME STEP
C
C      >>> IMPORTANT                                     <<<
C      >>>                                               <<<
C      >>> USERS MAY ONLY ADD OR ALTER PARTS OF THE SUBROUTINE WITHIN <<<
C      >>> THE DESIGNATED USER AREAS                     <<<
C
C*****
C
C      THIS SUBROUTINE IS CALLED BY THE FOLLOWING SUBROUTINES
C      CUSR  TRNMOD
C

```

```

C*****
C   CREATED
C     27/04/90   ADB
C   MODIFIED
C     05/08/91   IRH   NEW STRUCTURE
C     01/10/91   DSC   REDUCE COMMENT LINE GOING OVER COLUMN 72.
C     29/11/91   PHA   UPDATE CALLED BY COMMENT, ADD RF ARGUMENT,
C                       CHANGE LAST DIMENSION OF RS TO 6 AND IVERS TO 2
C     05/06/92   PHA   ADD PRECISION FLAG AND CHANGE IVERS TO 3
C     03/07/92   DSC   CORRECT COMMON MLTGRD.
C     23/11/93   CSH   EXPLICITLY DIMENSION IPVERT ETC.
C     03/02/94   PHA   CHANGE FLOW3D TO CFDS-FLOW3D
C     22/08/94   NSW   MOVE 'IF(IUSED.EQ.0) RETURN' OUT OF USER AREA
C     19/12/94   NSW   CHANGE FOR CFX-F3D
C     02/07/97   NSW   UPDATE FOR CFX-4
C     02/07/99   NSW   INCLUDE NEW EXAMPLE FOR CALCULATING FLUX OF A
C                       SCALAR AT A PRESSURE BOUNDARY
C
C*****
C   SUBROUTINE ARGUMENTS
C
C     U       - U COMPONENT OF VELOCITY
C     V       - V COMPONENT OF VELOCITY
C     W       - W COMPONENT OF VELOCITY
C     P       - PRESSURE
C     VFRAC   - VOLUME FRACTION
C     DEN     - DENSITY OF FLUID
C     VIS     - VISCOSITY OF FLUID
C     TE      - TURBULENT KINETIC ENERGY
C     ED      - EPSILON
C     RS      - REYNOLD STRESSES
C     T       - TEMPERATURE
C     H       - ENTHALPY
C     RF      - REYNOLD FLUXES
C     SCAL    - SCALARS (THE FIRST 'NCONC' OF THESE ARE MASS FRACTIONS)
C     XP     - X COORDINATES OF CELL CENTRES
C     YP     - Y COORDINATES OF CELL CENTRES
C     ZP     - Z COORDINATES OF CELL CENTRES
C     VOL     - VOLUME OF CELLS
C     AREA   - AREA OF CELLS
C     VPOR   - POROUS VOLUME
C     ARPOR  - POROUS AREA
C     WFACT  - WEIGHT FACTORS
C     CONV   - CONVECTION COEFFICIENTS
C
C     IPT    - 1D POINTER ARRAY
C     IBLK   - BLOCK SIZE INFORMATION
C     IPVERT - POINTER FROM CELL CENTERS TO 8 NEIGHBOURING VERTICES
C     IPNODN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING CELLS
C     IPFACN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING FACES
C     IPNODEF - POINTER FROM CELL FACES TO 2 NEIGHBOURING CELL CENTERS
C     IPNODEB - POINTER FROM BOUNDARY CENTERS TO CELL CENTERS
C     IPFACB - POINTER FROM BOUNDARY CENTERS TO BOUNDARY FACESS
C
C     WORK   - REAL WORKSPACE ARRAY
C     IWORK  - INTEGER WORKSPACE ARRAY
C     CWORK  - CHARACTER WORKSPACE ARRAY
C
C   SUBROUTINE ARGUMENTS PRECEDED WITH A '*' ARE ARGUMENTS THAT MUST
C   BE SET BY THE USER IN THIS ROUTINE.
C
C   NOTE THAT OTHER DATA MAY BE OBTAINED FROM CFX-4 USING THE

```

```

C  ROUTINE GETADD, FOR FURTHER DETAILS SEE THE VERSION 4
C  USER MANUAL.
C
C*****
C
C
C      DOUBLE PRECISION U
C      DOUBLE PRECISION V
C      DOUBLE PRECISION W
C      DOUBLE PRECISION P
C      DOUBLE PRECISION VFRAC
C      DOUBLE PRECISION DEN
C      DOUBLE PRECISION VIS
C      DOUBLE PRECISION TE
C      DOUBLE PRECISION ED
C      DOUBLE PRECISION RS
C      DOUBLE PRECISION T
C      DOUBLE PRECISION H
C      DOUBLE PRECISION RF
C      DOUBLE PRECISION SCAL
C      DOUBLE PRECISION XP
C      DOUBLE PRECISION YP
C      DOUBLE PRECISION ZP
C      DOUBLE PRECISION VOL
C      DOUBLE PRECISION AREA
C      DOUBLE PRECISION VPOR
C      DOUBLE PRECISION ARPOR
C      DOUBLE PRECISION WFACT
C      DOUBLE PRECISION CONV
C      DOUBLE PRECISION WORK
C      DOUBLE PRECISION SMALL
C      DOUBLE PRECISION SORMAX
C      DOUBLE PRECISION DTUSR
C      DOUBLE PRECISION TIME
C      DOUBLE PRECISION DT
C      DOUBLE PRECISION DTINVF
C      DOUBLE PRECISION TPARM
C      LOGICAL LDEN, LVIS, LTURB, LTEMP, LBUOY, LSCAL, LCOMP, LRECT, LCYN, LAXIS,
C      +      LPOROS, LTRANS
C
C      CHARACTER*(*) CWORK
C
C***** USER AREA 1 *****
C---- AREA FOR USERS EXPLICITLY DECLARED VARIABLES
C      DOUBLE PRECISION BFIELD, GAM
C
C***** END OF USER AREA 1 *****
C
C      COMMON /ALL/NBLOCK, NCELL, NBDRY, NNODE, NFACE, NVERT, NDIM,
C      +      /ALLWRK/NRWS, NIWS, NCWS, IWRFRE, IWIFRE, IWCFRE, /ADDIMS/NPHASE,
C      +      NSCAL, NVAR, NPROP, NDVAR, NDPROP, NDXNN, NDGEOM, NDCEOF, NILIST,
C      +      NRLIST, NTOPOL, /CHKUSR/IVERS, IUCALL, IUSED, /CONC/NCONC,
C      +      /DEVICE/NREAD, NWRITE, NRDISK, NWDISK, /IDUM/ILEN, JLEN,
C      +      /LOGIC/LDEN, LVIS, LTURB, LTEMP, LBUOY, LSCAL, LCOMP, LRECT, LCYN,
C      +      LAXIS, LPOROS, LTRANS, /MLTGRD/MLEVEL, NLEVEL, ILEVEL,
C      +      /SGLDBL/IFLGPR, ICHKPR, /SPARM/SMALL, SORMAX, NITER, INDPRI,
C      +      MAXIT, NODREF, NODMON, /TIMUSR/DTUSR, /TRANSI/NSTEP, KSTEP, MF,
C      +      INCORE, /TRANSR/TIME, DT, DTINVF, TPARM
C
C***** USER AREA 2 *****
C---- AREA FOR USERS TO DECLARE THEIR OWN COMMON BLOCKS
C      THESE SHOULD START WITH THE CHARACTERS 'UC' TO ENSURE

```

```

C      NO CONFLICT WITH NON-USER COMMON BLOCKS
C
C+++++ END OF USER AREA 2 ++++++
C
      DIMENSION U(NNODE,NPHASE),V(NNODE,NPHASE),W(NNODE,NPHASE),
+           P(NNODE,NPHASE),VFRAC(NNODE,NPHASE),DEN(NNODE,NPHASE),
+           VIS(NNODE,NPHASE),TE(NNODE,NPHASE),ED(NNODE,NPHASE),
+           RS(NNODE,NPHASE,6),T(NNODE,NPHASE),H(NNODE,NPHASE),
+           RF(NNODE,NPHASE,4),SCAL(NNODE,NPHASE,NSCAL)
      DIMENSION XP(NNODE),YP(NNODE),ZP(NNODE),VOL(NCELL),AREA(NFACE,3),
+           VPOR(NCELL),ARPOR(NFACE,3),WFACT(NFACE),
+           CONV(NFACE,NPHASE),IPT(*),IBLK(5,NBLOCK),
+           IPVERT(NCELL,8),IPNODN(NCELL,6),IPFACN(NCELL,6),
+           IPNODF(NFACE,4),IPNODB(NBDRY,4),IPFACB(NBDRY),IWORK(*),
+           WORK(*),CWORK(*)
      DIMENSION SGNWL(6)
C
C+++++ USER AREA 3 ++++++
C---- AREA FOR USERS TO DIMENSION THEIR ARRAYS
C
C---- AREA FOR USERS TO DEFINE DATA STATEMENTS
C
C+++++ END OF USER AREA 3 ++++++
C
      DATA SGNWL/1.0D0,1.0D0,1.0D0,-1.0D0,-1.0D0,-1.0D0/
C
C---- STATEMENT FUNCTION FOR ADDRESSING
      IP(I,J,K) = IPT((K-1)*ILEN*JLEN+ (J-1)*ILEN+I)
C
C----VERSION NUMBER OF USER ROUTINE AND PRECISION FLAG
C
      IVERS = 3
      ICHKPR = 2
C
C+++++ USER AREA 4 ++++++
C---- TO USE THIS USER ROUTINE FIRST SET IUUSED=1
C
      IUUSED = 1
C
C+++++ END OF USER AREA 4 ++++++
C
      IF (IUUSED.EQ.0) RETURN
C
C---- FRONTEND CHECKING OF USER ROUTINE
      IF (IUCALL.EQ.0) RETURN
C
C+++++ USER AREA 5 ++++++
C
C---- EXAMPLE (SET TIME INCREMENT FOR NEXT TIME STEP)
C
      DTUSR = 0.1
C
C---- END OF EXAMPLE
C
C---- EXAMPLE (CALCULATE FLUX OF FIRST SCALAR AT A PRESSURE BOUNDARY)
C
      IPHASE = 1
      FLUX = 0.0
C USE IPALL TO FIND ADDRESSES OF BOUNDARY NODES ON PATCH PRESS1
      CALL IPALL('PRESS1','PRESS','PATCH','CENTRES'
C      +           ,IPT,NPT,CWORK,IWORK)
C LOOP OVER ALL BOUNDARY NODES
      DO 300 I=1,NPT

```

```

C USE ARRAY IPT TO GET ADDRESS
C   INODE = IPT(I)
C   IBDRY = INODE - NCELL
C   IFACE = IPFACE( IBDRY)
C   INDUM = IPNOB( IBDRY,2)
C   NWL   = IPNOB( IBDRY,4)
C   FLUX  = FLUX
C   +    + SGNWL(NWL)*CONV( IFACE, IPHASE)*SCAL( INDUM,IPHASE,1)
C 300 CONTINUE
C
C---- END OF EXAMPLE
C#####
C       Calculating additional data:
C       shape and velocities of the jet (asymptotic solution)
C       in all internal cells
C#####
      CALL GETSCA( 'USRD B', IB,CWORK)
      CALL GETSCA( 'USRD EXACT U', IW,CWORK)
      CALL GETSCA( 'USRD EXACT H', IH,CWORK)

      GAM = 2.D0

      CALL IPALL( '*', '*', 'BLOCK', 'CENTRES', IPT
+              , NPT, CWORK, IWORK)
      DO 110 K = 1, NPT

         INODE = IPT(K)

         IF (XP(INODE).GT.0.D0) THEN
            BFIELD=1.D0-0.5D0*DTANH(GAM*XP( INODE))
         ELSE
            BFIELD = 1.D0
         ENDIF

         SCAL( INODE, 1, IW)=1.D0/BFIELD
         SCAL( INODE, 1, IB)=BFIELD

         SCAL( INODE, 2, IW)=1.D0/BFIELD
         SCAL( INODE, 2, IB)=BFIELD

         IF (YP(INODE).GT.BFIELD) THEN
            SCAL( INODE, 1, IH)=0.D0
            SCAL( INODE, 2, IH)=0.D0
         ELSE
            SCAL( INODE, 1, IH)=1.D0
            SCAL( INODE, 2, IH)=1.D0
         ENDIF

      110 CONTINUE

C#####
C       Calculating additional data:
C       shape and velocities of the jet (asymptotic solution)
C       on all patches
C#####
      CALL IPALL( '*', '*', 'PATCH', 'CENTRES', IPT
+              , NPT, CWORK, IWORK)
      DO 210 K = 1, NPT

         INODE = IPT(K)

         IF (XP(INODE).GT.0.D0) THEN

```

```

      BFIELD=1.D0-0.5D0*DTANH(GAM*XP(INODE))
    ELSE
      BFIELD = 1.D0
    ENDIF

    SCAL(INODE,1,IW)=1.D0/BFIELD
    SCAL(INODE,1,IB)=BFIELD

    SCAL(INODE,2,IW)=1.D0/BFIELD
    SCAL(INODE,2,IB)=BFIELD

    IF (YP(INODE).GT.BFIELD) THEN
      SCAL(INODE,1,IH)=0.D0
      SCAL(INODE,2,IH)=0.D0
    ELSE
      SCAL(INODE,1,IH)=1.D0
      SCAL(INODE,2,IH)=1.D0
    ENDIF

210 CONTINUE

C*****
C
C+++++++ END OF USER AREA 5 ++++++++
C
      RETURN
C
      END

13. Appendix 3: a sample of source code, three-dimensional
    flow, Hunt solution

      SUBROUTINE USRBF(IPHASE,BX,BY,BZ,BPX,BPY,BPZ,U,V,W,P,VFRAC,DEN,
+          VIS,TE,ED,RS,T,H,RF,SCAL,XP,YP,ZP,VOL,AREA,VPOR,
+          ARPOR,WFACT,IPT,IBLK,IPVERT,IPNODN,IPFACN,IPNODEF,
+          IPNODEB,IPFACB,WORK,IWORK,CWORK)C
C#####
C#####
C      Program for setting body forces
C#####
C#####
C*****C
C      UTILITY SUBROUTINE FOR USER-SUPPLIED BODY FORCES
C      >>> IMPORTANT
C      >>>
C      >>> USERS MAY ONLY ADD OR ALTER PARTS OF THE SUBROUTINE WITHIN
C      >>> THE DESIGNATED USER AREAS
C
C*****
C
C      THIS SUBROUTINE IS CALLED BY THE FOLLOWING SUBROUTINES
C      BFCAL
C
C*****
C      CREATED
C      24/01/92 ADB
C      MODIFIED
C      03/06/92 PHA ADD PRECISION FLAG AND CHANGE IVERS TO 2

```

```

C      23/11/93  CSH  EXPLICITLY DIMENSION IPVERT ETC.
C      03/02/94  PHA  CHANGE FLOW3D TO CFDS-FLOW3D
C      03/03/94  FHW  CORRECTION OF SPELLING MISTAKE
C      23/03/94  FHW  EXAMPLES COMMENTED OUT
C      09/08/94  NSW  CORRECT SPELLING
C                      MOVE 'IF(IUSED.EQ.0) RETURN' OUT OF USER AREA
C      19/12/94  NSW  CHANGE FOR CFX-F3D
C      31/01/97  NSW  EXPLAIN USAGE IN MULTIPHASE FLOWS
C      02/07/97  NSW  UPDATE FOR CFX-4
C
C*****
C
C      SUBROUTINE ARGUMENTS
C
C      IPHASE - PHASE NUMBER
C
C      * BX      - X-COMPONENT OF VELOCITY-INDEPENDENT BODY FORCE
C      * BY      - Y-COMPONENT OF VELOCITY-INDEPENDENT BODY FORCE
C      * BZ      - Z-COMPONENT OF VELOCITY-INDEPENDENT BODY FORCE
C      * BPX     -
C      * BPY     - COMPONENTS OF LINEARISABLE BODY FORCES.
C      * BPZ     -
C
C      N.B. TOTAL BODY-FORCE IS GIVEN BY:
C
C      X-COMPONENT = BX + BPX*U
C      Y-COMPONENT = BY + BPY*V
C      Z-COMPONENT = BZ + BPZ*W
C
C      U        - U COMPONENT OF VELOCITY
C      V        - V COMPONENT OF VELOCITY
C      W        - W COMPONENT OF VELOCITY
C      P        - PRESSURE
C      VFRAC    - VOLUME FRACTION
C      DEN      - DENSITY OF FLUID
C      VIS      - VISCOSITY OF FLUID
C      TE       - TURBULENT KINETIC ENERGY
C      ED       - EPSILON
C      RS       - REYNOLD STRESSES
C      T        - TEMPERATURE
C      H        - ENTHALPY
C      SCAL     - SCALARS (THE FIRST 'NCONC' OF THESE ARE MASS FRACTIONS)
C      XP       - X COORDINATES OF CELL CENTRES
C      YP       - Y COORDINATES OF CELL CENTRES
C      ZP       - Z COORDINATES OF CELL CENTRES
C      VOL      - VOLUME OF CELLS
C      AREA     - AREA OF CELLS
C      VPOR     - POROUS VOLUME
C      ARPOR    - POROUS AREA
C      WFACT    - WEIGHT FACTORS
C
C      IPT      - 1D POINTER ARRAY
C      IBLK     - BLOCK SIZE INFORMATION
C      IPVERT   - POINTER FROM CELL CENTERS TO 8 NEIGHBOURING VERTICES
C      IPNODN   - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING CELLS
C      IPFACN   - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING FACES
C      IPNODEF  - POINTER FROM CELL FACES TO 2 NEIGHBOURING CELL CENTERS
C      IPNODEB  - POINTER FROM BOUNDARY CENTERS TO CELL CENTERS
C      IPFACB   - POINTER FROM BOUNDARY CENTERS TO BOUNDARY FACES
C
C      WORK     - REAL WORKSPACE ARRAY
C      IWORK    - INTEGER WORKSPACE ARRAY
C      CWORK    - CHARACTER WORKSPACE ARRAY

```

```

C
C   SUBROUTINE ARGUMENTS PRECEDED WITH A '*' ARE ARGUMENTS THAT MUST
C   BE SET BY THE USER IN THIS ROUTINE.
C
C   NOTE THAT OTHER DATA MAY BE OBTAINED FROM CFX-4 USING THE
C   ROUTINE GETADD, FOR FURTHER DETAILS SEE THE VERSION 4
C   USER MANUAL.
C
C*****
C
C   DOUBLE PRECISION BX
C   DOUBLE PRECISION BY
C   DOUBLE PRECISION BZ
C   DOUBLE PRECISION BPX
C   DOUBLE PRECISION BPY
C   DOUBLE PRECISION BPZ
C   DOUBLE PRECISION U
C   DOUBLE PRECISION V
C   DOUBLE PRECISION W
C   DOUBLE PRECISION P
C   DOUBLE PRECISION VFRAC
C   DOUBLE PRECISION DEN
C   DOUBLE PRECISION VIS
C   DOUBLE PRECISION TE
C   DOUBLE PRECISION ED
C   DOUBLE PRECISION RS
C   DOUBLE PRECISION T
C   DOUBLE PRECISION H
C   DOUBLE PRECISION RF
C   DOUBLE PRECISION SCAL
C   DOUBLE PRECISION XP
C   DOUBLE PRECISION YP
C   DOUBLE PRECISION ZP
C   DOUBLE PRECISION VOL
C   DOUBLE PRECISION AREA
C   DOUBLE PRECISION VPOR
C   DOUBLE PRECISION ARPOR
C   DOUBLE PRECISION WFACT
C   DOUBLE PRECISION WORK
C   DOUBLE PRECISION SMALL
C   DOUBLE PRECISION SORMAX
C   DOUBLE PRECISION TIME
C   DOUBLE PRECISION DT
C   DOUBLE PRECISION DTINVF
C   DOUBLE PRECISION TPARM
C   LOGICAL LDEN, LVIS, LTURB, LTEMP, LBUOY, LSCAL, LCOMP, LRECT, LCYN, LAXIS,
C   +       LPOROS, LTRANS
C
C   CHARACTER*(*) CWORK
C
C+++++++ USER AREA 1 ++++++++
C---- AREA FOR USERS EXPLICITLY DECLARED VARIABLES
C
C+++++++ END OF USER AREA 1 ++++++++
C
C   COMMON /ALL/NBLOCK, NCELL, NBDRY, NNODE, NFACE, NVERT, NDIM,
C   +       /ALLWRK/NRWS, NIWS, NCWS, IWRFRE, IWIFRE, IWCFRE, /ADDIMS/NPHASE,
C   +       NSCAL, NVAR, NPROP, NDVAR, NDPROP, NDXNN, NDGEOM, NDCOEF, NILIST,
C   +       NRLIST, NTOPOL, /CHKUSR/ IVERS, IUCALL, IUSED, /DEVICE/NREAD,
C   +       NWRITE, NRDISK, NWDISK, /IDUM/ ILEN, JLEN, /LOGIC/LDEN, LVIS,
C   +       LTURB, LTEMP, LBUOY, LSCAL, LCOMP, LRECT, LCYN, LAXIS, LPOROS,
C   +       LTRANS, /MLTGRD/MLEVEL, NLEVEL, ILEVEL, /SGLDBL/IFLGPR, ICHKPR,
C   +       /SPARM/SMALL, SORMAX, NITER, INDPRI, MAXIT, NODREF, NODMON,

```

```

      +          /TRANSI/NSTEP,KSTEP,MF,INCORE,/TRANSR/TIME,DT,DTINVF,TPARM
C
C+++++++ USER AREA 2 ++++++++
C---- AREA FOR USERS TO DECLARE THEIR OWN COMMON BLOCKS
C      THESE SHOULD START WITH THE CHARACTERS 'UC' TO ENSURE
C      NO CONFLICT WITH NON-USER COMMON BLOCKS
C
C+++++++ END OF USER AREA 2 ++++++++
C
      DIMENSION BX(NCELL),BY(NCELL),BZ(NCELL),BPX(NCELL),BPY(NCELL),
      +          BPZ(NCELL)
C
      DIMENSION U(NNODE,NPHASE),V(NNODE,NPHASE),W(NNODE,NPHASE),
      +          P(NNODE,NPHASE),VFRAC(NNODE,NPHASE),DEN(NNODE,NPHASE),
      +          VIS(NNODE,NPHASE),TE(NNODE,NPHASE),ED(NNODE,NPHASE),
      +          RS(NNODE,NPHASE,*),T(NNODE,NPHASE),H(NNODE,NPHASE),
      +          RF(NNODE,NPHASE,4),SCAL(NNODE,NPHASE,NSCAL)
C
      DIMENSION XP(NNODE),YP(NNODE),ZP(NNODE),VOL(NCELL),AREA(NFACE,3),
      +          VPOR(NCELL),ARPOR(NFACE,3),WFACT(NFACE),IPT(*),
      +          IBLK(5,NBLOCK),IPVERT(NCELL,8),IPNODN(NCELL,6),
      +          IPFACN(NCELL,6),IPNODF(NFACE,4),IPNOB(NBDRY,4),
      +          IPFACB(NBDRY),IWORK(*),WORK(*),CWORK(*)
C
C+++++++ USER AREA 3 ++++++++
C---- AREA FOR USERS TO DIMENSION THEIR ARRAYS
C
C---- AREA FOR USERS TO DEFINE DATA STATEMENTS
C
C+++++++ END OF USER AREA 3 ++++++++
C
C---- STATEMENT FUNCTION FOR ADDRESSING
      IP(I,J,K) = IPT((K-1)*ILEN*JLEN+ (J-1)*ILEN+I)
C
C----VERSION NUMBER OF USER ROUTINE AND PRECISION FLAG
C
      IVERS = 2
      ICHKPR = 2
C
C+++++++ USER AREA 4 ++++++++
C---- TO USE THIS USER ROUTINE FIRST SET IUSED=1
C
      IUSED = 1
C
C+++++++ END OF USER AREA 4 ++++++++
C
      IF (IUSED.EQ.0) RETURN
C
C---- FRONTEND CHECKING OF USER ROUTINE
      IF (IUCALL.EQ.0) RETURN
C
C+++++++ USER AREA 5 ++++++++
C
C      THIS ROUTINE IS ENTERED REPEATEDLY FOR EACH PHASE IN A MULTIPHASE
C      CALCULATION. BODY FORCES CAN BE SET FOR A PARTICULAR PHASE USING
C      THE VARIABLE IPHASE. EG. IF (IPHASE.EQ.2) WOULD ALLOW BODY FORCES
C      FOR THE SECOND PHASE.
C
C----ADD USER-DEFINED BODY FORCES.
C
C----EXAMPLE 1: LOCALISED MOMENTUM SOURCE, EG. PROPELLOR.
C
C----USE IPREC TO FIND ADDRESSES

```

```

C
C   CALL IPREC('DUCT','BLOCK','CENTRES',IPT
C +           ,ILEN,JLEN,KLEN,CWORK,IWORK)
C
C   IST = ILEN/2 + 1
C   IFN = IST
C   JST = 1
C   JFN = JLEN/2
C
C   SMOM = 10.0
C   DO 103 K = 1, KLEN
C     DO 102 J = JST, JFN
C       DO 101 I = IST,IFN
C         INODE = IP(I,J,K)
C         BX(INODE) = BX(INODE) + SMOM
C 101     CONTINUE
C 102     CONTINUE
C 103     CONTINUE
C
C-----EXAMPLE 2: LOCALISED RESISTANCE
C
C-----USE IPREC TO FIND ADDRESSES
C
C   CALL IPREC('DUCT','BLOCK','CENTRES',IPT
C +           ,ILEN,JLEN,KLEN,CWORK,IWORK)
C
C   IST = ILEN/4 + 1
C   IFN = 3*ILEN/4
C   JST = 1
C   JFN = JLEN
C
C   RESIST = 1.0E+2
C   DO 203 K = 1, KLEN
C     DO 202 J = JST, JFN
C       DO 201 I = IST,IFN
C         INODE = IP(I,J,K)
C         BPX(INODE) = BPX(INODE) - RESIST
C         BPY(INODE) = BPY(INODE) - RESIST
C         BPZ(INODE) = BPZ(INODE) - RESIST
C 201     CONTINUE
C 202     CONTINUE
C 203     CONTINUE
C
C-----EXAMPLE 3: LOCALISED RESISTANCES (DISCONTINUOUS CHANGE)
C
C-----USE IPREC TO FIND ADDRESSES
C
C   CALL IPREC('DUCT','BLOCK','CENTRES',IPT
C +           ,ILEN,JLEN,KLEN,CWORK,IWORK)
C
C   IST1 = ILEN/4 + 1
C   IFN1 = IST1 + ILEN/4 - 1
C   IST2 = IFN1 + 1
C   IFN2 = ILEN - 1
C
C   DO 313 K = 1, KLEN
C     DO 312 J = 1, JLEN
C
C       RESIST = 1.0
C       DO 311 I = IST1,IFN1
C         INODE = IP(I,J,K)
C         BPX(INODE) = BPX(INODE) - RESIST
C         BPY(INODE) = BPY(INODE) - RESIST

```

```

C          BPZ(INODE) = BPZ(INODE) - RESIST
C 311      CONTINUE
C
C          RESIST = 10.0
C          DO 321 I = IST2,IFN2
C            INODE = IP(I,J,K)
C            BPX(INODE) = BPX(INODE) - RESIST
C            BPY(INODE) = BPY(INODE) - RESIST
C            BPZ(INODE) = BPZ(INODE) - RESIST
C 321      CONTINUE
C
C 312      CONTINUE
C 313      CONTINUE
C
C***** ADDED BY S. A. *****
C#####
C          Calculating gradients of the electric potential
C#####

          CALL GETVAR('USRBF','SCAL ',IVAR)
          IPHASE = 1

          CALL SETWRK('USRBF','WORK ', 'GRADT ',3*NCELL,JGRADT)
          CALL GRADS('USRBF','SCAL ',IVAR,IPHASE,SCAL(1,1,1)
+                ,WORK(JGRADT),XP,YP,ZP,VOL,AREA
+                ,IBLK,IPVERT,IPNODN,IPFACN,IPNODF,IPNODB
+                ,IPFACB,WORK,IWORK,CWORK)

          CALL IPALL('*','*', 'BLOCK', 'CENTRES',IPT
+                ,NPT,CWORK,IWORK)

          DO 203 K = 1, NPT
            INODE = IPT(K)

C#####
C          In all internal cells body force is set equal to
C          Fx = dFi/dz - 1 * U
C          Fz = -dFi/dx - 1 * W
C#####

          BX(INODE) = BX(INODE) + WORK(JGRADT+2*NCELL+INODE-1)
          BPX(INODE) = BPX(INODE) + (-1.0D+0)
          BZ(INODE) = BZ(INODE) + (-WORK(JGRADT+INODE-1))
          BPZ(INODE) = BPZ(INODE) + (-1.0D+0)
203      CONTINUE
          CALL DELWRK('USRBF','WORK ', 'GRADT ')
C*****
C+++++ END OF USER AREA 5 +++++
C
          RETURN
C
          END
          SUBROUTINE USRSRC(IEQN,ICALL,CNAME,CALIAS,AM,SP,SU,CONV,U,V,W,P,
+                VFRAC,DEN,VIS,TE,ED,RS,T,H,RF,SCAL,XP,YP,ZP,VOL,
+                AREA,VPOR,ARPOR,WFACT,IPT,IBLK,IPVERT,IPNODN,
+                IPFACN,IPNODF,IPNODB,IPFACB,WORK,IWORK,CWORK)
C#####
C          Program for adding source term for electric potential
C#####
C#####
C

```

```

C*****
C
C   UTILITY SUBROUTINE FOR USER-SUPPLIED SOURCES
C
C   >>> IMPORTANT                                     <<<
C   >>>                                               <<<
C   >>> USERS MAY ONLY ADD OR ALTER PARTS OF THE SUBROUTINE WITHIN <<<
C   >>> THE DESIGNATED USER AREAS                     <<<
C
C*****
C
C   THIS SUBROUTINE IS CALLED BY THE FOLLOWING SUBROUTINES
C   CUSR SCDF SCDS SCED SCENRG SCHF SCMOM SCPCE SCSCAL
C   SCTE SCVF
C
C*****
C   CREATED
C   08/03/90 ADB
C   MODIFIED
C   04/03/91 ADB ALTERED ARGUMENT LIST.
C   28/08/91 IRH NEW STRUCTURE
C   28/09/91 IRH CHANGE EXAMPLE + ADD COMMON BLOCKS
C   10/02/92 PHA UPDATE CALLED BY COMMENT, ADD RF ARGUMENT,
C               CHANGE LAST DIMENSION OF RS TO 6 AND IVERS TO 2
C   03/06/92 PHA ADD PRECISION FLAG AND CHANGE IVERS TO 3
C   23/11/93 CSH EXPLICITLY DIMENSION IPVERT ETC.
C   07/12/93 NSW INCLUDE CONV IN ARGUMENT LIST AND CHANGE IVERS
C               TO 4
C   03/02/94 PHA CHANGE FLOW3D TO CFDS-FLOW3D
C   03/03/94 FHW CORRECTION OF SPELLING MISTAKE
C   08/03/94 NSW CORRECT SPELLING
C   09/08/94 NSW CORRECT SPELLING.
C               MOVE 'IF(IUSED.EQ.0) RETURN' OUT OF USER AREA.
C               INCLUDE COMMENT ON MASS SOURCES.
C   19/12/94 NSW CHANGE FOR CFX-F3D
C   02/07/97 NSW UPDATE FOR CFX-4
C
C*****
C
C   SUBROUTINE ARGUMENTS
C
C   IEQN - EQUATION NUMBER
C   ICALL - SUBROUTINE CALL
C   CNAME - EQUATION NAME
C   CALIAS - ALIAS OF EQUATION NAME
C   AM - OFF DIAGONAL MATRIX COEFFICIENTS
C   SU - SU IN LINEARISATION OF SOURCE TERM
C   SP - SP IN LINEARISATION OF SOURCE TERM
C   CONV - CONVECTION COEFFICIENTS
C   U - U COMPONENT OF VELOCITY
C   V - V COMPONENT OF VELOCITY
C   W - W COMPONENT OF VELOCITY
C   P - PRESSURE
C   VFRAC - VOLUME FRACTION
C   DEN - DENSITY OF FLUID
C   VIS - VISCOSITY OF FLUID
C   TE - TURBULENT KINETIC ENERGY
C   ED - EPSILON
C   RS - REYNOLD STRESSES
C   T - TEMPERATURE
C   H - ENTHALPY
C   RF - REYNOLD FLUXES
C   SCAL - SCALARS (THE FIRST 'NCONC' OF THESE ARE MASS FRACTIONS)

```

```

C     XP     - X COORDINATES OF CELL CENTRES
C     YP     - Y COORDINATES OF CELL CENTRES
C     ZP     - Z COORDINATES OF CELL CENTRES
C     VOL    - VOLUME OF CELLS
C     AREA   - AREA OF CELLS
C     VPOR   - POROUS VOLUME
C     ARPOR  - POROUS AREA
C     WFACT  - WEIGHT FACTORS
C
C     IPT    - 1D POINTER ARRAY
C     IBLK   - BLOCK SIZE INFORMATION
C     IPVERT - POINTER FROM CELL CENTERS TO 8 NEIGHBOURING VERTICES
C     IPNODN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING CELLS
C     IPFACN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING FACES
C     IPNODF - POINTER FROM CELL FACES TO 2 NEIGHBOURING CELL CENTERS
C     IPNODB - POINTER FROM BOUNDARY CENTERS TO CELL CENTERS
C     IPFACB - POINTER FROM BOUNDARY CENTERS TO BOUNDARY FACES
C
C     WORK   - REAL WORKSPACE ARRAY
C     IWORK  - INTEGER WORKSPACE ARRAY
C     CWORK  - CHARACTER WORKSPACE ARRAY
C
C     SUBROUTINE ARGUMENTS PRECEDED WITH A '*' ARE ARGUMENTS THAT MUST
C     BE SET BY THE USER IN THIS ROUTINE.
C
C     NOTE THAT WHEN USING MASS SOURCES, THE FLOWS THROUGH MASS FLOW
C     BOUNDARIES ARE UNCHANGED. THE USER SHOULD THEREFORE INCLUDE AT
C     LEAST ONE PRESSURE BOUNDARY FOR SUCH A CALCULATION.
C
C     NOTE THAT OTHER DATA MAY BE OBTAINED FROM CFX-4 USING THE
C     ROUTINE GETADD, FOR FURTHER DETAILS SEE THE VERSION 4
C     USER MANUAL.
C
C*****
C
C     DOUBLE PRECISION AM
C     DOUBLE PRECISION SP
C     DOUBLE PRECISION SU
C     DOUBLE PRECISION CONV
C     DOUBLE PRECISION U
C     DOUBLE PRECISION V
C     DOUBLE PRECISION W
C     DOUBLE PRECISION P
C     DOUBLE PRECISION VFRAC
C     DOUBLE PRECISION DEN
C     DOUBLE PRECISION VIS
C     DOUBLE PRECISION TE
C     DOUBLE PRECISION ED
C     DOUBLE PRECISION RS
C     DOUBLE PRECISION T
C     DOUBLE PRECISION H
C     DOUBLE PRECISION RF
C     DOUBLE PRECISION SCAL
C     DOUBLE PRECISION XP
C     DOUBLE PRECISION YP
C     DOUBLE PRECISION ZP
C     DOUBLE PRECISION VOL
C     DOUBLE PRECISION AREA
C     DOUBLE PRECISION VPOR
C     DOUBLE PRECISION ARPOR
C     DOUBLE PRECISION WFACT
C     DOUBLE PRECISION WORK
C     DOUBLE PRECISION SMALL

```

```

DOUBLE PRECISION SORMAX
DOUBLE PRECISION TIME
DOUBLE PRECISION DT
DOUBLE PRECISION DTINVF
DOUBLE PRECISION TPARM
LOGICAL LDEN,LVIS,LTURB,LTEMP,LBOUY,LSCAL,LCOMP,LRECT,LCYN,LAXIS,
+       LPOROS,LTRANS
C
CHARACTER*(*) CWORK
CHARACTER CNAME*6,CALIAS*24
C
C+++++ USER AREA 1 ++++++
C---- AREA FOR USERS EXPLICITLY DECLARED VARIABLES
DOUBLE PRECISION THESRC
C
C+++++ END OF USER AREA 1 ++++++
C
COMMON /ALL/NBLOCK,NCELL,NBDRY,NNODE,NFACE,NVERT,NDIM,
+       /ALLWRK/NRWS,NIWS,NCWS,IWRFRE,IWIFRE,IWCFRE,/ADDIMS/NPHASE,
+       NSCAL,NVAR,NPROP,NDVAR,NDPROP,NDXNN,NDGEOM,NDCOEF,NILIST,
+       NRLIST,NTOPOL,/CHKUSR/IVERS,IUCALL,IUSED,/DEVICE/NREAD,
+       NWRITE,NRDISK,NWDISK,/IDUM/ILEN,JLEN,/LOGIC/LDEN,LVIS,
+       LTURB,LTEMP,LBOUY,LSCAL,LCOMP,LRECT,LCYN,LAXIS,LPOROS,
+       LTRANS,/MLTGRD/MLEVEL,NLEVEL,ILEVEL,/SGLDBL/IFLGPR,ICHPKPR,
+       /SPARM/SMALL,SORMAX,NITER,INDPRI,MAXIT,NODREF,NODMON,
+       /TRANSI/NSTEP,KSTEP,MF,INCORE,/TRANSR/TIME,DT,DTINVF,TPARM
C
C+++++ USER AREA 2 ++++++
C---- AREA FOR USERS TO DECLARE THEIR OWN COMMON BLOCKS
C   THESE SHOULD START WITH THE CHARACTERS 'UC' TO ENSURE
C   NO CONFLICT WITH NON-USER COMMON BLOCKS
C
C+++++ END OF USER AREA 2 ++++++
C
DIMENSION AM(NCELL,6,NPHASE),SP(NCELL,NPHASE),SU(NCELL,NPHASE),
+       CONV(NFACE,NPHASE)
C
DIMENSION U(NNODE,NPHASE),V(NNODE,NPHASE),W(NNODE,NPHASE),
+       P(NNODE,NPHASE),VFRAC(NNODE,NPHASE),DEN(NNODE,NPHASE),
+       VIS(NNODE,NPHASE),TE(NNODE,NPHASE),ED(NNODE,NPHASE),
+       RS(NNODE,NPHASE,6),T(NNODE,NPHASE),H(NNODE,NPHASE),
+       RF(NNODE,NPHASE,4),SCAL(NNODE,NPHASE,NSCAL)
C
DIMENSION XP(NNODE),YP(NNODE),ZP(NNODE),VOL(NCELL),AREA(NFACE,3),
+       VPOR(NCELL),ARPOR(NFACE,3),WFACT(NFACE),IPT(*),
+       IBLK(5,NBLOCK),IPVERT(NCELL,8),IPNODN(NCELL,6),
+       IPFACN(NCELL,6),IPNODF(NFACE,4),IPNODB(NBDRY,4),
+       IPFACB(NBDRY),IWORK(*),WORK(*),CWORK(*)
C
C+++++ USER AREA 3 ++++++
C---- AREA FOR USERS TO DIMENSION THEIR ARRAYS
C
C---- AREA FOR USERS TO DEFINE DATA STATEMENTS
C
C+++++ END OF USER AREA 3 ++++++
C
C---- STATEMENT FUNCTION FOR ADDRESSING
      IP(I,J,K) = IPT((K-1)*ILEN*JLEN+ (J-1)*ILEN+I)
C
C----VERSION NUMBER OF USER ROUTINE AND PRECISION FLAG
C
      IVERS = 4
      ICHKPR = 2

```

```

C
C+++++ USER AREA 4 ++++++
C---- TO USE THIS USER ROUTINE FIRST SET IUSED=1
C
      IUSED = 1
C
C+++++ END OF USER AREA 4 ++++++
C
      IF (IUSED.EQ.0) RETURN
C
C---- FRONTEND CHECKING OF USER ROUTINE
      IF (IUCALL.EQ.0) RETURN
C
C---- ADD TO SOURCE TERMS
      IF (ICALL.EQ.1) THEN
C
C+++++ USER AREA 5 ++++++
C
C---- EXAMPLE (HEAT SOURCE) ADD 100W PER UNIT VOLUME IN BLOCK
C      'BLOCK-NUMBER-2'
C
C      USE IPREC TO FIND ADDRESSES
C
C      CALL IPREC('BLOCK-NUMBER-2','BLOCK','CENTRES',IPT,ILEN,JLEN,KLEN,
C      +          CWORK,IWORK)
C
C      FIND VARIABLE NUMBER FOR ENTHALPY
C      CALL GETVAR('USRSRC','H',IVAR)
C      IF ENTHALPY EQUATION ADD SOURCE TERMS
C      IF (IVAR.EQ.IEQN) THEN
C      LOOP OVER PATCH
C      DO 103 K = 1, KLEN
C      DO 102 J = 1, JLEN
C      DO 101 I = 1, ILEN
C      USE STATEMENT FUNCTION IP TO GET ADDRESSES
C      INODE = IP(I,J,K)
C      ADD HEAT SOURCE
C      SU(INODE,1)=SU(INODE,1)+100.0*VOL(INODE)
C 101    CONTINUE
C 102    CONTINUE
C 103    CONTINUE
C      ENDIF
C
C---- END OF EXAMPLE

C***** ADDED BY S. A. *****

C#####
C      Getting numer of the user scalar
C#####

      CALL GETVAR('USRSRC','SCAL',IVAR)
      IPHASE = 1

      IF (IVAR.EQ.IEQN) THEN

C#####
C      If it is electric potential equation, calculate source
C      First, gradients of velocities are calculated
C#####

      CALL SETWRK('USRSRC','WORK','UGRAD',3*NCELL,JUGRAD)
      CALL SETWRK('USRSRC','WORK','VGRAD',3*NCELL,JVGRAD)

```

```

CALL SETWRK('USRSRC','WORK ','WGRAD ',3*NCELL,JWGRAD)

CALL GRADV('USRSRC',IPHASE,U(1,IPHASE),V(1,IPHASE)
+       ,W(1,IPHASE),WORK(JUGRAD),WORK(JVGRAD)
+       ,WORK(JWGRAD),XP,YP,ZP,VOL,AREA
+       ,IBLK,IPVERT,IPNODN,IPFACN,IPNODF,IPNODEB
+       ,IPFACB,WORK,IWORK,CWORK)
CALL IPALL('*','*', 'BLOCK', 'CENTRES', IPT, NPT, CWORK, IWORK)

DO 200 I=1,NPT
  INODE=IPT(I)
C#####
C      At all internal cells the source term is equal to
C      S = - V * (-dW/dx + dU/dz)
C#####
      THESRC =
+      (-WORK(JWGRAD+INODE-1))
+      +WORK(JUGRAD+2*NCELL+INODE-1)
      SU(INODE,1)=SU(INODE,1)-VOL(INODE)*THESRC
200 CONTINUE
CALL DELWRK('USRSRC','WORK ','UGRAD ')

ENDIF

C*****
C+++++++ END OF USER AREA 5 ++++++++
      END IF
C
C---- OVERWRITE SOURCE TERMS
      IF (ICALL.EQ.2) THEN
C
C+++++++ USER AREA 6 ++++++++
C
C---- EXAMPLE (HEAT SOURCE) OVERWRITE WITH 100W PER UNIT VOLUME IN
      ALL INTERIOR CELLS
C
C      CALL GETVAR('USRSRC','H ','IVAR)
C
C      IF (IVAR.EQ.IEQN) THEN
C      USE IPALL TO FIND 1D ADDRESSES OF ALL CELL CENTRES
C      CALL IPALL('*','*', 'BLOCK', 'CENTRES', IPT, NPT, CWORK, IWORK)
C      LOOP OVER ALL INTERIOR CELLS
C      DO 200 I=1,NPT
C      USE ARRAY IPT TO GET ADDRESS
C      INODE=IPT(I)
C      OVERWRITE SOURCE TERMS
C      SU(INODE,1)=100.0*VOL(INODE)
C 200 CONTINUE
C      ENDIF
C
C---- END OF EXAMPLE
C+++++++ END OF USER AREA 6 ++++++++
C
      END IF
C
      RETURN
C
      END
SUBROUTINE USRTRN(U,V,W,P,VFRAC,DEN,VIS,TE,ED,RS,T,H,RF,SCAL,XP,
+       YP,ZP,VOL,AREA,VPOR,ARPOR,WFACT,CONV,IPT,IBLK,
+       IPVERT,IPNODN,IPFACN,IPNODF,IPNODEB,IPFACB,WORK,
+       IWORK,CWORK)

```

```

C#####
C#####
C      Program for calculation additional data sets
C#####
C#####
C
C*****
C
C      USER SUBROUTINE TO ALLOW USERS TO MODIFY OR MONITOR THE SOLUTION AT
C      THE END OF EACH TIME STEP
C      THIS SUBROUTINE IS CALLED BEFORE THE START OF THE RUN AS WELL AS AT
C      THE END OF EACH TIME STEP
C
C      >>> IMPORTANT                                     <<<
C      >>>                                               <<<
C      >>> USERS MAY ONLY ADD OR ALTER PARTS OF THE SUBROUTINE WITHIN <<<
C      >>> THE DESIGNATED USER AREAS                       <<<
C
C*****
C      THIS SUBROUTINE IS CALLED BY THE FOLLOWING SUBROUTINES
C      CUSR  TRNMOD
C
C*****
C      CREATED
C      27/04/90  ADB
C      MODIFIED
C      05/08/91  IRH  NEW STRUCTURE
C      01/10/91  DSC  REDUCE COMMENT LINE GOING OVER COLUMN 72.
C      29/11/91  PHA  UPDATE CALLED BY COMMENT, ADD RF ARGUMENT,
C                   CHANGE LAST DIMENSION OF RS TO 6 AND IVERS TO 2
C      05/06/92  PHA  ADD PRECISION FLAG AND CHANGE IVERS TO 3
C      03/07/92  DSC  CORRECT COMMON MLTGRD.
C      23/11/93  CSH  EXPLICITLY DIMENSION IPVERT ETC.
C      03/02/94  PHA  CHANGE FLOW3D TO CFDS-FLOW3D
C      22/08/94  NSW  MOVE 'IF(IUSED.EQ.0) RETURN' OUT OF USER AREA
C      19/12/94  NSW  CHANGE FOR CFX-F3D
C      02/07/97  NSW  UPDATE FOR CFX-4
C      02/07/99  NSW  INCLUDE NEW EXAMPLE FOR CALCULATING FLUX OF A
C                   SCALAR AT A PRESSURE BOUNDARY
C
C*****
C      SUBROUTINE ARGUMENTS
C
C      U      - U COMPONENT OF VELOCITY
C      V      - V COMPONENT OF VELOCITY
C      W      - W COMPONENT OF VELOCITY
C      P      - PRESSURE
C      VFRAC  - VOLUME FRACTION
C      DEN    - DENSITY OF FLUID
C      VIS    - VISCOSITY OF FLUID
C      TE     - TURBULENT KINETIC ENERGY
C      ED     - EPSILON
C      RS     - REYNOLD STRESSES
C      T      - TEMPERATURE
C      H      - ENTHALPY
C      RF     - REYNOLD FLUXES
C      SCAL  - SCALARS (THE FIRST 'NCONC' OF THESE ARE MASS FRACTIONS)
C      XP    - X COORDINATES OF CELL CENTRES
C      YP    - Y COORDINATES OF CELL CENTRES
C      ZP    - Z COORDINATES OF CELL CENTRES
C      VOL   - VOLUME OF CELLS

```

```

C     AREA   - AREA OF CELLS
C     VPOR   - POROUS VOLUME
C     ARPOR  - POROUS AREA
C     WFACT  - WEIGHT FACTORS
C     CONV   - CONVECTION COEFFICIENTS
C
C     IPT    - 1D POINTER ARRAY
C     IBLK   - BLOCK SIZE INFORMATION
C     IPVERT - POINTER FROM CELL CENTERS TO 8 NEIGHBOURING VERTICES
C     IPNODN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING CELLS
C     IPFACN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING FACES
C     IPNODEF - POINTER FROM CELL FACES TO 2 NEIGHBOURING CELL CENTERS
C     IPNODEB - POINTER FROM BOUNDARY CENTERS TO CELL CENTERS
C     IPFACB - POINTER FROM BOUNDARY CENTERS TO BOUNDARY FACES
C
C     WORK   - REAL WORKSPACE ARRAY
C     IWORK  - INTEGER WORKSPACE ARRAY
C     CWORK  - CHARACTER WORKSPACE ARRAY
C
C     SUBROUTINE ARGUMENTS PRECEDED WITH A '*' ARE ARGUMENTS THAT MUST
C     BE SET BY THE USER IN THIS ROUTINE.
C
C     NOTE THAT OTHER DATA MAY BE OBTAINED FROM CFX-4 USING THE
C     ROUTINE GETADD, FOR FURTHER DETAILS SEE THE VERSION 4
C     USER MANUAL.
C
C*****
C
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C     DOUBLE PRECISION U
C     DOUBLE PRECISION V
C     DOUBLE PRECISION W
C     DOUBLE PRECISION P
C     DOUBLE PRECISION VFRAC
C     DOUBLE PRECISION DEN
C     DOUBLE PRECISION VIS
C     DOUBLE PRECISION TE
C     DOUBLE PRECISION ED
C     DOUBLE PRECISION RS
C     DOUBLE PRECISION T
C     DOUBLE PRECISION H
C     DOUBLE PRECISION RF
C     DOUBLE PRECISION SCAL
C     DOUBLE PRECISION XP
C     DOUBLE PRECISION YP
C     DOUBLE PRECISION ZP
C     DOUBLE PRECISION VOL
C     DOUBLE PRECISION AREA
C     DOUBLE PRECISION VPOR
C     DOUBLE PRECISION ARPOR
C     DOUBLE PRECISION WFACT
C     DOUBLE PRECISION CONV
C     DOUBLE PRECISION WORK
C     DOUBLE PRECISION SMALL
C     DOUBLE PRECISION SORMAX
C     DOUBLE PRECISION DTUSR
C     DOUBLE PRECISION TIME
C     DOUBLE PRECISION DT
C     DOUBLE PRECISION DTINVF
C     DOUBLE PRECISION TPARM
C     DOUBLE PRECISION SGNWL
C     LOGICAL LDEN, LVIS, LTURB, LTEMP, LBUOY, LSCAL, LCOMP, LRECT, LCYN, LAXIS,

```

```

      +          LPOROS,LTRANS
C
      CHARACTER*(*) CWORK
C
C+++++ USER AREA 1 ++++++
C---- AREA FOR USERS EXPLICITLY DECLARED VARIABLES
      DOUBLE PRECISION SUMU, SUMP, ALFAN, CN, BETAN, GAMMAN, SGN
      DOUBLE PRECISION CS1, CS2, SN1, SN2, CH1, CH2, SH1, SH2
      DOUBLE PRECISION CSG, CHB, DN, EN, KN, DPN
      DOUBLE PRECISION FRAC1, FRAC2, FRAC3, FRAC4
      DOUBLE PRECISION E1, E2, E3, E4, EB, UN, PNC
C+++++ END OF USER AREA 1 ++++++
C
      COMMON /ALL/NBLOCK,NCELL,NBDRY,NNODE,NFACE,NVERT,NDIM,
+          /ALLWRK/NRWS,NIWS,NCWS,IWRFRE,IWLFRE,IWCFRE,/ADDIMS/NPHASE,
+          NSCAL,NVAR,NPROP,NDVAR,NDPROP,NDXNN,NDGEOM,NDCOEF,NILIST,
+          NRLIST,NTOPOL,/CHKUSR/IVERS,IUCALL,IUSED,/CONC/NCONC,
+          /DEVICE/NREAD,NWRITE,NRDISK,NWDISK,/IDUM/ILEN,JLEN,
+          /LOGIC/LDEN,LVIS,LTURB,ITEMP,LBUOY,LSCAL,LCOMP,LRECT,LCYN,
+          LAXIS,LPOROS,LTRANS,/MLTGRD/MLEVEL,NLEVEL,ILEVEL,
+          /SGLDBL/IFLGPR,ICHPR,/SPARM/SMALL,SORMAX,NITER,INDPRI,
+          MAXIT,NODREF,NODMON,/TIMUSR/DTUSR,/TRANSI/NSTEP,KSTEP,MF,
+          INCORE,/TRANSR/TIME,DT,DTINVF,TPARM
C
C+++++ USER AREA 2 ++++++
C---- AREA FOR USERS TO DECLARE THEIR OWN COMMON BLOCKS
C      THESE SHOULD START WITH THE CHARACTERS 'UC' TO ENSURE
C      NO CONFLICT WITH NON-USER COMMON BLOCKS
C
C+++++ END OF USER AREA 2 ++++++
C
      DIMENSION U(NNODE,NPHASE),V(NNODE,NPHASE),W(NNODE,NPHASE),
+          P(NNODE,NPHASE),VFRAC(NNODE,NPHASE),DEN(NNODE,NPHASE),
+          VIS(NNODE,NPHASE),TE(NNODE,NPHASE),ED(NNODE,NPHASE),
+          RS(NNODE,NPHASE,6),T(NNODE,NPHASE),H(NNODE,NPHASE),
+          RF(NNODE,NPHASE,4),SCAL(NNODE,NPHASE,NSCAL)
      DIMENSION XP(NNODE),YP(NNODE),ZP(NNODE),VOL(NCELL),AREA(NFACE,3),
+          VPOR(NCELL),ARPOR(NFACE,3),WFACT(NFACE),
+          CONV(NFACE,NPHASE),IPT(*),IBLK(5,NBLOCK),
+          IPVERT(NCELL,8),IPNODN(NCELL,6),IPFACN(NCELL,6),
+          IPNODF(NFACE,4),IPNODB(NBDRY,4),IPFACB(NBDRY),IWORK(*),
+          WORK(*),CWORK(*)
      DIMENSION SGNWL(6)
C
C+++++ USER AREA 3 ++++++
C---- AREA FOR USERS TO DIMENSION THEIR ARRAYS
C
C---- AREA FOR USERS TO DEFINE DATA STATEMENTS
C
C+++++ END OF USER AREA 3 ++++++
C
      DATA SGNWL/1.0D0,1.0D0,1.0D0,-1.0D0,-1.0D0,-1.0D0/
C
C---- STATEMENT FUNCTION FOR ADDRESSING
      IP(I,J,K) = IPT((K-1)*ILEN*JLEN+ (J-1)*ILEN+I)
C
C----VERSION NUMBER OF USER ROUTINE AND PRECISION FLAG
C
      IVERS = 3
      ICHKPR = 2
C
C+++++ USER AREA 4 ++++++
C---- TO USE THIS USER ROUTINE FIRST SET IUSED=1

```

```

C
C      IUSED = 1
C
C+++++ END OF USER AREA 4 +++++
C
C      IF (IUSED.EQ.0) RETURN
C
C---- FRONTEND CHECKING OF USER ROUTINE
C      IF (IUCALL.EQ.0) RETURN
C
C+++++ USER AREA 5 +++++
C
C---- EXAMPLE (SET TIME INCREMENT FOR NEXT TIME STEP)
C
C      DTUSR = 0.1
C
C---- END OF EXAMPLE
C
C---- EXAMPLE (CALCULATE FLUX OF FIRST SCALAR AT A PRESSURE BOUNDARY)
C
C      IPHASE = 1
C      FLUX = 0.0
C      USE IPALL TO FIND ADDRESSES OF BOUNDARY NODES ON PATCH PRESS1
C      CALL IPALL('PRESS1','PRESS','PATCH','CENTRES'
C      +          ,IPT,NPT,CWORK,IWORK)
C      LOOP OVER ALL BOUNDARY NODES
C      DO 300 I=1,NPT
C      USE ARRAY IPT TO GET ADDRESS
C      INODE = IPT(I)
C      IBDRY = INODE - NCELL
C      IFACE = IPFACB(IBDRY)
C      INDUM = IPNODEB(IBDRY,2)
C      NWL = IPNODEB(IBDRY,4)
C      FLUX = FLUX
C      +          + SGNWL(NWL)*CONV(IFACE,IPHASE)*SCAL(INDUM,IPHASE,1)
C 300 CONTINUE
C
C---- END OF EXAMPLE

C***** ADDED BY S. A. *****
C#####
C          If not the first iteration,
C          additional scalars are calculated
C#####

C      IF (NITER.GT.1) THEN

C      IPHASE = 1
C      CALL GETVAR('USRTRN','SCAL ',IT)

C#####
C          Calculate gradient of the electric potential
C#####

C      CALL SETWRK('USRTRN','WORK ', 'GRADT ', 3*NCELL, JGRADT)
C      CALL GRADS('USRTRN','SCAL ', IT, IPHASE, SCAL(1,1,1)
C      +          , WORK(JGRADT), XP, YP, ZP, VOL, AREA
C      +          , IBLK, IPVERT, IPNODN, IPFACN, IPNODF, IPNODB
C      +          , IPFACB, WORK, IWORK, CWORK)
C      CALL IPALL('*','*', 'BLOCK', 'CENTRES', IPT
C      +          , NPT, CWORK, IWORK)

C      DO 203 K = 1, NPT

```

```

      INODE=IPT(K)
C#####
C      Calculate electric current in x and y directions
C#####
      SCAL( INODE,1,2)=-WORK(JGRADT+INODE-1)-W( INODE,1)
      SCAL( INODE,1,3)=-WORK(JGRADT+NCELL+INODE-1)
203 CONTINUE
      CALL DELWRK('USRTRN','WORK ','GRADT ')

C#####
C      Exact solution for electric potential and velocity W
C      (Hunt solution)
C#####

      CALL GETVAR('USRTRN','SCAL ',IT)
      CALL GETSCA('USRD EXACT POT',IEP,CWORK)
      CALL GETSCA('USRD EXACT W',IW,CWORK)

      IPHS = 1

      AA      = 1.0D0
      DD      = 1.0D0
      HA      = 200.0D0
      NTERMS  = 10000
      PI      = 3.14159265358979D0

      CALL IPALL('*','*','BLOCK','CENTRES',IPT
+             ,NPT,CWORK,IWORK)
      DO 110 K = 1, NPT
          INODE = IPT(K)
          X      = -XP(INODE)
          YY     = YP(INODE)
          SUMU   = 0.0D0
          SUMB   = 0.0D0
          SUMP   = 0.0D0
          SGN    = 1.0D0
          DO 210 N = 0,NTERMS
              ALFAN = (DFLOAT(N)+0.5D0)*PI/AA
              EN     = 2.0D0*SGN/AA/ALFAN/(ALFAN**2.+Ha**2.)
              BETAN  = DSQRT( ALFAN/2.0D0*(ALFAN+DSQRT(ALFAN**2.+Ha**2.)) )
              GAMMAN = DSQRT( ALFAN/2.0D0*(-ALFAN+DSQRT(ALFAN**2.+Ha**2.)) )
              IF ((BETAN*(X+DD).LT.20.0D0)
*              .AND.(-BETAN*(X-DD).LT.20.0D0)) THEN
                  CS1 = DCOS( GAMMAN*(X+DD))
                  CS2 = DCOS( GAMMAN*(X-DD))
                  SN1 = DSIN( GAMMAN*(X+DD))
                  SN2 = DSIN( GAMMAN*(X-DD))
                  CH1 = DCOSH( BETAN*(X+DD))
                  CH2 = DCOSH( BETAN*(X-DD))
                  SH1 = DSINH( BETAN*(X+DD))
                  SH2 = DSINH( BETAN*(X-DD))
                  CSG = DCOS(2.0D0*GAMMAN*DD)
                  CHB = DCOSH(2.0D0*BETAN*DD)
                  CN = CH1*CS2 + CH2*CS1
                  DN = -SH1*SN2 - SH2*SN1
                  KN = CHB + CSG
                  DPN = (BETAN*(SH2*SN1-SH1*SN2)+GAMMAN*(CH1*CS2-CH2*CS1))
*                  / (BETAN**2.+GAMMAN**2.)
                  FRAC1 = (CN - HA/ALFAN*DN)/KN
                  FRAC3 = DPN*(Ha/ALFAN+ALFAN/HA)/KN
              ELSE
                  E1 = DEXP( -BETAN*(X+DD) )

```

```

E2 = DEXP( BETAN*(X-DD) )
E3 = DEXP( -2.D0*BETAN*(X+DD) )
E4 = DEXP( 2.D0*BETAN*(X-DD) )
EB = DEXP( -2.D0*BETAN*DD)
CS1 = DCOS( GAMMAN*(X+DD))
CS2 = DCOS( GAMMAN*(X-DD))
SN1 = DSIN( GAMMAN*(X+DD))
SN2 = DSIN( GAMMAN*(X-DD))
CSG = DCOS(2.D0*GAMMAN*DD)

CN = 0.5D0*(E2*CS2*(1.D0+E3)+E1*CS1*(E4+1.D0))
DN = 0.5D0*(-E2*SN2*(1.D0-E3)-E1*SN1*(E4-1.D0))
KN = 0.5D0*(1.D0+EB**2.) + EB*CSG
DPN = (-BETAN*( E2*SN2*(1.D0+E3)+E1*SN1*(E4+1.D0) )
*      + GAMMAN*(E2*CS2*(1.D0-E3)+E1*CS1*(E4-1.D0) ) )
*      / (BETAN**2.+GAMMAN**2.) / 2.D0

FRAC1 = (CN - Ha/ALFAN*DN)/KN
FRAC3 = DPN*(Ha/ALFAN+ALFAN/Ha)/KN
ENDIF
UN = EN*(1.D0 - FRAC1)
PN = EN*(FRAC3)
SUMU = SUMU + UN*DCOS( ALFAN*YY )
SUMP = SUMP + PN*DCOS( ALFAN*YY )
SGN = -SGN
210 CONTINUE
VEL = SUMU*Ha**2.
POT = SUMP*Ha**2.
SCAL(INODE,1,IEP) = POT
SCAL(INODE,1,IW) = VEL

110 CONTINUE
ENDIF
write(6,*) '##### Subroutine USRTRN finished'
C*****

C+++++++ END OF USER AREA 5 ++++++++
C
RETURN

END

SUBROUTINE USRBCS(VARBCS,VARAMB,A,B,C,ACND,BCND,CCND,IWGVEL,
+ NDVWAL,FLOUT,NLABEL,NSTART,NEND,NCST,NCEN,U,V,W,
+ P,VFRAC,DEN,VIS,TE,ED,RS,T,H,RF,SCAL,XP,YP,ZP,
+ VOL,AREA,VPOR,ARPOR,WFACT,IPT,IBLK,IPVERT,
+ IPNODN,IPFACN,IPNODF,IPNODB,IPFACB,WORK,IWORK,
+ CWORK)
C#####
C#####
C Program for setting boundary conditions
C#####
C#####
C*****
C
C USER ROUTINE TO SET REALS AT BOUNDARIES.
C
C >>> IMPORTANT <<<
C >>> <<<
C >>> USERS MAY ONLY ADD OR ALTER PARTS OF THE SUBROUTINE WITHIN <<<
C >>> THE DESIGNATED USER AREAS <<<

```

```

C
C*****
C
C THIS SUBROUTINE IS CALLED BY THE FOLLOWING SUBROUTINE
C   CUSR  SRLIST
C
C*****
C   CREATED
C     30/11/88  ADB
C   MODIFIED
C     08/09/90  ADB  RESTRUCTURED FOR USER-FRIENDLINESS.
C     10/08/91  IRH  FURTHER RESTRUCTURING ADD ACND BCND CCND
C     22/09/91  IRH  CHANGE ICALL TO IUCALL + ADD /SPARM/
C     10/03/92  PHA  UPDATE CALLED BY COMMENT, ADD RF ARGUMENT,
C                   CHANGE LAST DIMENSION OF RS TO 6 AND IVERS TO 2
C     03/06/92  PHA  ADD PRECISION FLAG AND CHANGE IVERS TO 3
C     30/06/92  NSW  INCLUDE FLAG FOR CALLING BY ITERATION
C                   INSERT EXTRA COMMENTS
C     03/08/92  NSW  MODIFY DIMENSION STATEMENTS FOR VAX
C     21/12/92  CSH  INCREASE IVERS TO 4
C     02/08/93  NSW  INCORRECT AND MISLEADING COMMENT REMOVED
C     05/11/93  NSW  INDICATE USE OF FLOUT IN MULTIPHASE FLOWS
C     23/11/93  CSH  EXPLICITLY DIMENSION IPVERT ETC.
C     01/02/94  NSW  SET VARIABLE POINTERS IN WALL EXAMPLE.
C                   CHANGE FLOW3D TO CFDS-FLOW3D.
C                   MODIFY MULTIPHASE MASS FLOW BOUNDARY TREATMENT.
C     03/03/94  FHW  CORRECTION OF SPELLING MISTAKE
C     02/07/94  BAS  SLIDING GRIDS - ADD NEW ARGUMENT IWGVEL
C                   TO ALLOW VARIANTS OF TRANSIENT-GRID WALL BC
C                   CHANGE VERSION NUMBER TO 5
C     09/08/94  NSW  CORRECT SPELLING
C                   MOVE 'IF(IUSED.EQ.0) RETURN' OUT OF USER AREA
C     19/12/94  NSW  CHANGE FOR CFX-F3D
C     02/02/95  NSW  CHANGE COMMON /IMFBMP/
C     02/06/97  NSW  MAKE EXAMPLE MORE LOGICAL
C     02/07/97  NSW  UPDATE FOR CFX-4
C     08/09/98  NSW  CORRECT SIZE OF WALL ARRAY IN COMMENT
C     22/05/00  NSW  INITIALISE IUBCSF
C
C*****
C
C   SUBROUTINE ARGUMENTS
C
C   VARBCS - REAL BOUNDARY CONDITIONS
C   VARAMB - AMBIENT VALUE OF VARIABLES
C   A      - COEFFICIENT IN WALL BOUNDARY CONDITION
C   B      - COEFFICIENT IN WALL BOUNDARY CONDITION
C   C      - COEFFICIENT IN WALL BOUNDARY CONDITION
C   ACND   - COEFFICIENT IN CONDUCTING WALL BOUNDARY CONDITION
C   BCND   - COEFFICIENT IN CONDUCTING WALL BOUNDARY CONDITION
C   CCND   - COEFFICIENT IN CONDUCTING WALL BOUNDARY CONDITION
C   IWGVEL - USAGE OF INPUT VELOCITIES (0 = AS IS,1 = ADD GRID MOTION)
C   NDVWAL - FIRST DIMENSION OF ARRAY IWGVEL
C   FLOUT  - MASS FLOW/FRACTIONAL MASS FLOW
C   NLABEL - NUMBER OF DISTINCT OUTLETS
C   NSTART - ARRAY POINTER
C   NEND   - ARRAY POINTER
C   NCST   - ARRAY POINTER
C   NCEN   - ARRAY POINTER
C   U      - U COMPONENT OF VELOCITY
C   V      - V COMPONENT OF VELOCITY
C   W      - W COMPONENT OF VELOCITY
C   P      - PRESSURE

```

```

C   VFRAC - VOLUME FRACTION
C   DEN   - DENSITY OF FLUID
C   VIS   - VISCOSITY OF FLUID
C   TE    - TURBULENT KINETIC ENERGY
C   ED    - EPSILON
C   RS    - REYNOLD STRESSES
C   T     - TEMPERATURE
C   H     - ENTHALPY
C   RF    - REYNOLD FLUXES
C   SCAL  - SCALARS (THE FIRST 'NCONC' OF THESE ARE MASS FRACTIONS)
C   XP    - X COORDINATES OF CELL CENTRES
C   YP    - Y COORDINATES OF CELL CENTRES
C   ZP    - Z COORDINATES OF CELL CENTRES
C   VOL   - VOLUME OF CELLS
C   AREA  - AREA OF CELLS
C   VPOR  - POROUS VOLUME
C   ARPOR - POROUS AREA
C   WFACT - WEIGHT FACTORS
C
C   IPT   - 1D POINTER ARRAY
C   IBLK  - BLOCK SIZE INFORMATION
C   IPVERT - POINTER FROM CELL CENTERS TO 8 NEIGHBOURING VERTICES
C   IPNODN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING CELLS
C   IPFACN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING FACES
C   IPNODF - POINTER FROM CELL FACES TO 2 NEIGHBOURING CELL CENTERS
C   IPNODB - POINTER FROM BOUNDARY CENTERS TO CELL CENTERS
C   IPFACB - POINTER TO NODES FROM BOUNDARY FACES
C
C   WORK  - REAL WORKSPACE ARRAY
C   IWORK - INTEGER WORKSPACE ARRAY
C   CWORK - CHARACTER WORKSPACE ARRAY
C
C   SUBROUTINE ARGUMENTS PRECEDED WITH A '*' ARE ARGUMENTS THAT MUST
C   BE SET BY THE USER IN THIS ROUTINE.
C
C   NOTE THAT OTHER DATA MAY BE OBTAINED FROM CFX-4 USING THE
C   ROUTINE GETADD, FOR FURTHER DETAILS SEE THE VERSION 4
C   USER MANUAL.
C
C*****
C   IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C   DOUBLE PRECISION VARBCS
C   DOUBLE PRECISION VARAMB
C   DOUBLE PRECISION A
C   DOUBLE PRECISION B
C   DOUBLE PRECISION C
C   DOUBLE PRECISION ACND
C   DOUBLE PRECISION BCND
C   DOUBLE PRECISION CCND
C   DOUBLE PRECISION FLOUT
C   DOUBLE PRECISION U
C   DOUBLE PRECISION V
C   DOUBLE PRECISION W
C   DOUBLE PRECISION P
C   DOUBLE PRECISION VFRAC
C   DOUBLE PRECISION DEN
C   DOUBLE PRECISION VIS
C   DOUBLE PRECISION TE
C   DOUBLE PRECISION ED
C   DOUBLE PRECISION RS
C   DOUBLE PRECISION T
C   DOUBLE PRECISION H
C   DOUBLE PRECISION RF

```

```

DOUBLE PRECISION SCAL
DOUBLE PRECISION XP
DOUBLE PRECISION YP
DOUBLE PRECISION ZP
DOUBLE PRECISION VOL
DOUBLE PRECISION AREA
DOUBLE PRECISION VPOR
DOUBLE PRECISION ARPOR
DOUBLE PRECISION WFACT
DOUBLE PRECISION WORK
DOUBLE PRECISION SMALL
DOUBLE PRECISION SORMAX
DOUBLE PRECISION TIME
DOUBLE PRECISION DT
DOUBLE PRECISION DTINVF
DOUBLE PRECISION TPARM
LOGICAL LDEN, LVIS, LTURB, LTEMP, LBUOY, LSCAL, LCOMP, LRECT, LCYN, LAXIS,
+       LPOROS, LTRANS
C
C       CHARACTER*(*) CWORK
C
C+++++ USER AREA 1 ++++++
C---- AREA FOR USERS EXPLICITLY DECLARED VARIABLES
DOUBLE PRECISION SUMU, SUMP, ALFAN, CN, BETAN, GAMMAN, SGN
DOUBLE PRECISION CS1, CS2, SN1, SN2, CH1, CH2, SH1, SH2
DOUBLE PRECISION CSG, CHB, DN, EN, KN, DPN
DOUBLE PRECISION FRAC1, FRAC2, FRAC3, FRAC4
DOUBLE PRECISION E1, E2, E3, E4, EB, UN, PNC
C+++++ END OF USER AREA 1 ++++++
C
COMMON /ALL/NBLOCK, NCELL, NBDRY, NNODE, NFACE, NVERT, NDIM,
+       /ALLWRK/NRWS, NIWS, NCWS, IWRFRE, IWLFRE, IWCFRE, /ADDIMS/NPHASE,
+       NSCAL, NVAR, NPROP, NDVAR, NDPROP, NDXNN, NDGEOM, NDCOEF, NILIST,
+       NRLIST, NTOPOL, /BCSOUT/IFLOUT, /CHKUSR/IVERS, IUCALL, IUSED,
+       /DEVICE/NREAD, NWRITE, NRDISK, NWDISK, /IDUM/ILEN, JLEN,
+       /IMFBMP/IMFBMP, JMFMBMP, /LOGIC/LDEN, LVIS, LTURB, LTEMP, LBUOY,
+       LSCAL, LCOMP, LRECT, LCYN, LAXIS, LPOROS, LTRANS, /MLTGRD/MLEVEL,
+       NLEVEL, ILEVEL, /SGLDBL/IFLGPR, ICHKPR, /SPARM/SMALL, SORMAX,
+       NITER, INDPRI, MAXIT, NODREF, NODMON, /TRANSI/NSTEP, KSTEP, MF,
+       INCORE, /TRANSR/TIME, DT, DTINVF, TPARM, /UBCSFL/IUBCSF
C
C+++++ USER AREA 2 ++++++
C---- AREA FOR USERS TO DECLARE THEIR OWN COMMON BLOCKS
C       THESE SHOULD START WITH THE CHARACTERS 'UC' TO ENSURE
C       NO CONFLICT WITH NON-USER COMMON BLOCKS
C
C+++++ END OF USER AREA 2 ++++++
C
DIMENSION VARBCS(NVAR, NPHASE, NCELL+1:NNODE), VARAMB(NVAR, NPHASE),
+       A(4+NSCAL, NPHASE, NSTART:*), B(4+NSCAL, NPHASE, NSTART:*),
+       C(4+NSCAL, NPHASE, NSTART:*), FLOUT(*), ACND(NCST:*),
+       BCND(NCST:*), CCND(NCST:*), IWGVEL(NDVWAL, NPHASE)
C
DIMENSION U(NNODE, NPHASE), V(NNODE, NPHASE), W(NNODE, NPHASE),
+       P(NNODE, NPHASE), VFRAC(NNODE, NPHASE), DEN(NNODE, NPHASE),
+       VIS(NNODE, NPHASE), TE(NNODE, NPHASE), ED(NNODE, NPHASE),
+       RS(NNODE, NPHASE, 6), T(NNODE, NPHASE), H(NNODE, NPHASE),
+       RF(NNODE, NPHASE, 4), SCAL(NNODE, NPHASE, NSCAL)
C
DIMENSION XP(NNODE), YP(NNODE), ZP(NNODE), VOL(NCELL), AREA(NFACE, 3),
+       VPOR(NCELL), ARPOR(NFACE, 3), WFACT(NFACE), IPT(*),
+       IBLK(5, NBLOCK), IPVERT(NCELL, 8), IPNODN(NCELL, 6),
+       IPFACN(NCELL, 6), IPNODF(NFACE, 4), IPNODB(NBDRY, 4),

```

```

+          IPFACB(NBDRY),IWORK(*),WORK(*),CWORK(*)
C
C+++++ USER AREA 3 ++++++
C---- AREA FOR USERS TO DIMENSION THEIR ARRAYS
C
C---- AREA FOR USERS TO DEFINE DATA STATEMENTS
C
C+++++ END OF USER AREA 3 ++++++
C
C---- STATEMENT FUNCTION FOR ADDRESSING
      IP(I,J,K) = IPT((K-1)*ILEN*JLEN+ (J-1)*ILEN+I)
C
C----VERSION NUMBER OF USER ROUTINE AND PRECISION FLAG
C
      IVERS = 5
      ICHKPR = 2
C
C+++++ USER AREA 4 ++++++
C---- TO USE THIS USER ROUTINE FIRST SET IUSED=1
C      AND SET IUBCSF FLAG:
C      BOUNDARY CONDITIONS NOT CHANGING           IUBCSF=0
C      BOUNDARY CONDITIONS CHANGING WITH ITERATION IUBCSF=1
C      BOUNDARY CONDITIONS CHANGING WITH TIME     IUBCSF=2
C      BOUNDARY CONDITIONS CHANGING WITH TIME AND ITERATION IUBCSF=3
C
      IUSED = 1
      IUBCSF = 0
C
C+++++ END OF USER AREA 4 ++++++
C
      IF (IUSED.EQ.0) RETURN
C
C---- FRONTEND CHECKING OF USER ROUTINE
      IF (IUCALL.EQ.0) RETURN
C
C+++++ USER AREA 5 ++++++
C
C---- AREA FOR SETTING VALUES AT INLETS, PRESSURE BOUNDARIES
C      AND OUTLETS. (NOTE THAT THE MASS FLOW AT OUTLETS IS
C      SPECIFIED IN USER AREA 7)
C
C      IF USING A REYNOLDS STRESS OR FLUX MODEL, NOTE THAT AT INLETS
C      IT IS IMPORTANT THAT THE USER SETS ALL COMPONENTS OF THE
C      REYNOLDS STRESS AND FLUX AND THE TURBULENT KINETIC ENERGY
C      AS WELL AS THE ENERGY DISSIPATION RATE.
C
C      SET THE VALUES IN VARBCS(NVAR,NPHASE,ILEN,JLEN,KLEN)
C
C---- EXAMPLE: SETTING A LINEAR T PROFILE ON INLET PATCH 'ENTRANCE'
C      LEAVE OTHER VARIABLES AS SET IN COMMAND LANGUAGE
C
C-- INTERROGATE GETVAR FOR VARIABLE NUMBERS.
C
C      CALL GETVAR('USRBCS','T',IT)
C
C      SET IPHS = 1 FOR SINGLE PHASE FLOW.
C
C      IPHS = 1
C
C      USE IPREC TO FIND ADDRESSES
C
C      CALL IPREC('ENTRANCE','PATCH','CENTRES',IPT,ILEN,JLEN,KLEN,
C      +          CWORK,IWORK)

```

```

C
C   XMAX=2.0
C   XMIN=1.0
C   TMAX=300.0
C   TMIN=250.0
C   LOOP OVER PATCH
C     DO 103 K = 1, KLEN
C       DO 102 J = 1, JLEN
C         DO 101 I = 1, ILEN
C   USE STATEMENT FUNCTION IP TO GET ADDRESSES
C     INODE = IP(I,J,K)
C   SET VARBCS
C     F=(XP(INODE)-XMIN)/(XMAX-XMIN)
C     VARBCS(IT,IPHS,INODE) = F*TMAX + (1.0-F)*TMIN
C 101   CONTINUE
C 102   CONTINUE
C 103   CONTINUE
C
C----END OF EXAMPLE
C
C***** ADDED BY S. A. *****

      CALL GETVAR('USRBCS','SCAL ',IT)
      CALL GETVAR('USRBCS','W      ',IW)
      IPHS = 1

      AA      = 1.0D0
      DD      = 1.0D0
      HA      = 200.0D0
      NTERMS  = 10000
      PI      = 3.14159265358979D0

C#####
C      At patch entrance of type inlet velocity profile
C      and electric potential are set
C#####

      CALL IPALL('ENTRANCE','*', 'PATCH', 'CENTRES', IPT
+             ,NPT,CWORK,IWORK)
      DO 110 K = 1, NPT
        INODE = IPT(K)
        X      = -XP(INODE)
        YY     = YP(INODE)
        SUMU   = 0.D0
        SUMB   = 0.D0
        SUMP   = 0.D0
        SGN    = 1.D0
        DO 210 N = 0, NTERMS
          ALFAN = (DFLOAT(N)+0.5D0)*PI/AA
          EN    = 2.D0*SGN/AA/ALFAN/(ALFAN**2.+Ha**2.)
          BETAN = DSQRT( ALFAN/2.D0*(ALFAN+DSQRT(ALFAN**2.+Ha**2.)) )
          GAMMAN = DSQRT( ALFAN/2.D0*(-ALFAN+DSQRT(ALFAN**2.+Ha**2.)) )
          IF ((BETAN*(X+DD).LT.20.D0)
*          .AND.(-BETAN*(X-DD).LT.20.D0)) THEN
            CS1 = DCOS( GAMMAN*(X+DD))
            CS2 = DCOS( GAMMAN*(X-DD))
            SN1 = DSIN( GAMMAN*(X+DD))
            SN2 = DSIN( GAMMAN*(X-DD))
            CH1 = DCOSH( BETAN*(X+DD))
            CH2 = DCOSH( BETAN*(X-DD))
            SH1 = DSINH( BETAN*(X+DD))
            SH2 = DSINH( BETAN*(X-DD))
            CSG = DCOS(2.D0*GAMMAN*DD)

```

```

      CHB = DCOSH(2.D0*BETAN*DD)
      CN = CH1*CS2 + CH2*CS1
      DN = -SH1*SN2 - SH2*SN1
      KN = CHB + CSG
      DPN = (BETAN*(SH2*SN1-SH1*SN2)+GAMMAN*(CH1*CS2-CH2*CS1))
*          / (BETAN**2.+GAMMAN**2.)
      FRAC1 = (CN - HA/ALFAN*DN)/KN
      FRAC3 = DPN*(Ha/ALFAN+ALFAN/HA)/KN
ELSE
      E1 = DEXP( -BETAN*(X+DD) )
      E2 = DEXP( BETAN*(X-DD) )
      E3 = DEXP( -2.D0*BETAN*(X+DD) )
      E4 = DEXP( 2.D0*BETAN*(X-DD) )
      EB = DEXP( -2.D0*BETAN*DD)
      CS1 = DCOS( GAMMAN*(X+DD))
      CS2 = DCOS( GAMMAN*(X-DD))
      SN1 = DSIN( GAMMAN*(X+DD))
      SN2 = DSIN( GAMMAN*(X-DD))
      CSG = DCOS(2.D0*GAMMAN*DD)

      CN = 0.5D0*(E2*CS2*(1.D0+E3)+E1*CS1*(E4+1.D0))
      DN = 0.5D0*(-E2*SN2*(1.D0-E3)-E1*SN1*(E4-1.D0))
      KN = 0.5D0*(1.D0+EB**2.) + EB*CSG
      DPN = (-BETAN*( E2*SN2*(1.D0+E3)+E1*SN1*(E4+1.D0) )
*          + GAMMAN*(E2*CS2*(1.D0-E3)+E1*CS1*(E4-1.D0) ) )
*          / (BETAN**2.+GAMMAN**2.)/2.D0

      FRAC1 = (CN - HA/ALFAN*DN)/KN
      FRAC3 = DPN*(Ha/ALFAN+ALFAN/HA)/KN
ENDIF
      UN = EN*(1.D0 - FRAC1)
      PN = EN*(FRAC3)
      SUMU = SUMU + UN*DCOS( ALFAN*YY )
      SUMP = SUMP + PN*DCOS( ALFAN*YY )
      SGN = -SGN
210  CONTINUE
      VEL = SUMU*Ha**2.
      POT = SUMP*Ha**2.
      VARBCS(IT,1,INODE) = POT
      VARBCS(IW,1,INODE) = VEL

110  CONTINUE

      write(6,*) '##### Subroutine USRBCS finished'
C*****
C+++++ END OF USER AREA 5 ++++++
C
C+++++ USER AREA 6 ++++++
C
C---- AREA FOR SETTING VALUES AT WALLS
C
C      USE A(4+NSCAL,NPHASE,NNODE)
C      WHERE NSCAL = NO. OF SCALARS, AND NPHASE = NO. OF PHASES.
C
C      THE CONVENTION FOR VARIABLE NUMBERS IS DIFFERENT IN THIS ROUTINE
C      FROM THAT IN THE REST OF THE PROGRAM. IT IS:
C
C      IU = 1, IV = 2 , IW = 3, IT = 4, IS = 5
C
C---- EXAMPLE: SETTING FREE SLIP BOUNDARY CONDITIONS AT ALL WALLS
C              AND SETTING T=300.0 AND SCALAR1 AND SCALAR2 =0.0
C              ON WALL1. SET T=400.0 ON CONDUCTING SOLID BOUNDARY WALL2

```

```
C
C-- SET POINTERS
C
C     IU = 1
C     IV = 2
C     IW = 3
C     IT = 4
C     IS = 5
C
C-- SET IPHS = 1 FOR SINGLE PHASE FLOW.
C
C     IPHS = 1
C
C USE IPALL TO FIND 1D ADDRESSES OF A GROUP OF PATCH CENTRES
C
C     CALL IPALL('*', 'WALL', 'PATCH', 'CENTRES', IPT, NPT, CWORK, IWORK)
C
C LOOP OVER GROUP OF PATCHES
C     DO 200 I=1, NPT
C USE ARRAY IPT TO GET ADDRESS
C     INODE=IPT(I)
C     A(IU, IPHS, INODE) = 0.0
C     B(IU, IPHS, INODE) = 1.0
C     C(IU, IPHS, INODE) = 0.0
C
C     A(IV, IPHS, INODE) = 0.0
C     B(IV, IPHS, INODE) = 1.0
C     C(IV, IPHS, INODE) = 0.0
C
C     A(IW, IPHS, INODE) = 0.0
C     B(IW, IPHS, INODE) = 1.0
C     C(IW, IPHS, INODE) = 0.0
C 200 CONTINUE
C
C USE IPREC TO FIND ADDRESSES OF SINGLE PATCH
C
C     CALL IPREC('WALL1', 'PATCH', 'CENTRES', IPT, ILEN, JLEN, KLEN,
C +           CWORK, IWORK)
C LOOP OVER PATCH
C     DO 203 K = 1, KLEN
C     DO 202 J = 1, JLEN
C     DO 201 I = 1, ILEN
C USE STATEMENT FUNCTION IP TO GET ADDRESSES
C     INODE = IP(I, J, K)
C
C     A(IT, IPHS, INODE) = 1.0
C     B(IT, IPHS, INODE) = 0.0
C     C(IT, IPHS, INODE) = 300.0
C
C     A(IS, IPHS, INODE) = 1.0
C     B(IS, IPHS, INODE) = 0.0
C     C(IS, IPHS, INODE) = 0.0
C
C     A(IS+1, IPHS, INODE) = 1.0
C     B(IS+1, IPHS, INODE) = 0.0
C     C(IS+1, IPHS, INODE) = 0.0
C
C 201 CONTINUE
C 202 CONTINUE
C 203 CONTINUE
C
C USE IPALL TO FIND 1D ADDRESSES OF A GROUP OF PATCH CENTRES
C
```

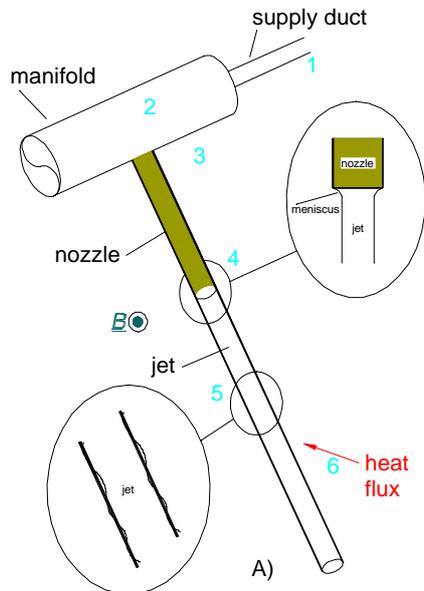
```

C      CALL IPALL('WALL2','*', 'PATCH', 'CENTRES', IPT, NPT, CWORK, IWORK)
C
C      LOOP OVER GROUP OF PATCHES
C      DO 300 I=1, NPT
C      USE ARRAY IPT TO GET ADDRESS
C      INODE=IPT(I)
C      ACND(INODE) = 1.0
C      BCND(INODE) = 0.0
C      CCND(INODE) = 400.0
C 300   CONTINUE
C
C-----END OF EXAMPLE
C
C+++++ END OF USER AREA 6 +++++
C
C
C+++++ USER AREA 7 +++++
C
C----- DEFINE FLOW AT OUTLETS (MASS FLOW BOUNDARIES)
C      (TO TEMPERATURES AND SCALARS AT MASS FLOW BOUNDARIES USE
C      USER AREA 5)
C
C      SET PARAMETER IFLOUT:
C      IFLOUT = 1 ==> MASS FLOW SPECIFIED AT LABELLED OUTLETS.
C      IFLOUT = 2 ==> FRACTIONAL MASS FLOW SPECIFIED AT LABELLED OUTLETS
C      IFLOUT = 2
C
C      SET OUTLET FLOW RATES:
C      FLOUT(LABEL) = MASS FLOW OUT OF OUTLETS LABELLED LABEL (IFLOUT=1).
C      FLOUT(LABEL) = FRACTIONAL MASS FLOW OUT OF OUTLETS LABELLED LABEL
C                      (IFLOUT=2).
C      FOR MULTIPHASE FLOWS IT IS NECESSARY TO SET
C      EITHER
C
C                      FLOUT(LABEL) = TOTAL MASS FLOW
C                      IFLOUT = 1
C                      IMFBMP = 0
C      OR
C
C                      FLOUT(LABEL + (IPHASE-1)*NLABEL) FOR EACH PHASE
C                      IFLOUT = 1 OR 2
C                      IMFBMP = 1
C
C----- EXAMPLE: EQUIDISTRIBUTION OF FRACTIONAL MASS FLOW AMONGST OUTLETS
C
C      IFLOUT=2
C      FRAC = 1.0 / MAX( 1.0, FLOAT(NLABEL) )
C      DO 300 ILABEL = 1, NLABEL
C          FLOUT(ILABEL) = FRAC
C 300   CONTINUE
C
C-----END OF EXAMPLE
C
C+++++ END OF USER AREA 7 +++++
C
      RETURN
C
      END

```

Figure 1 MHD problems in for the upper parts of the divertor.

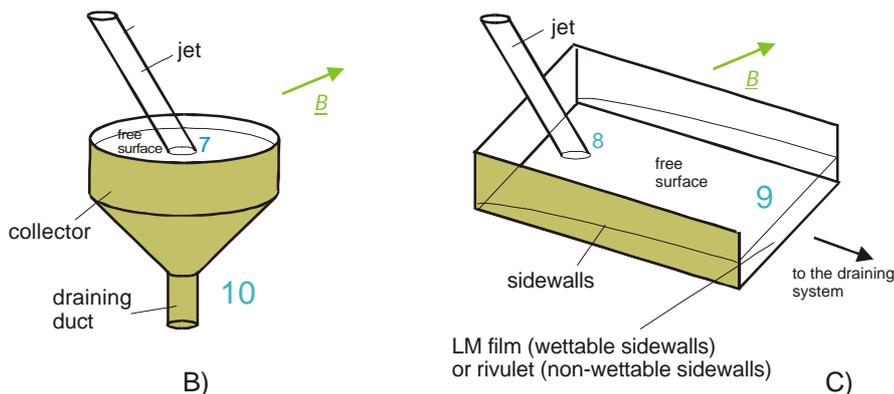
Main Thermal, Fluid, and MHD Problems Associated With a Jet Divertor



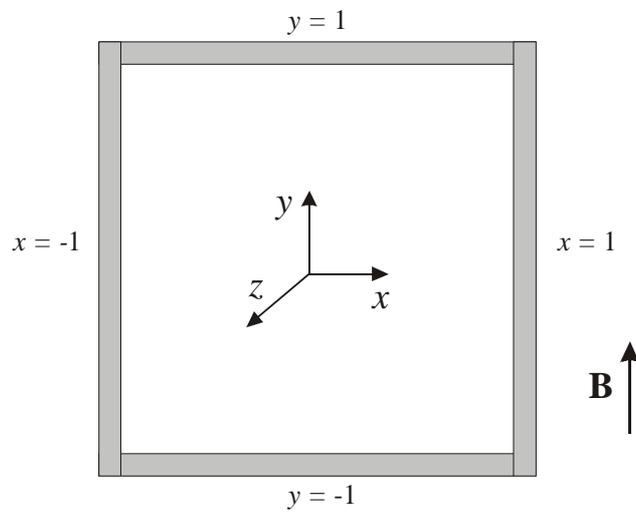
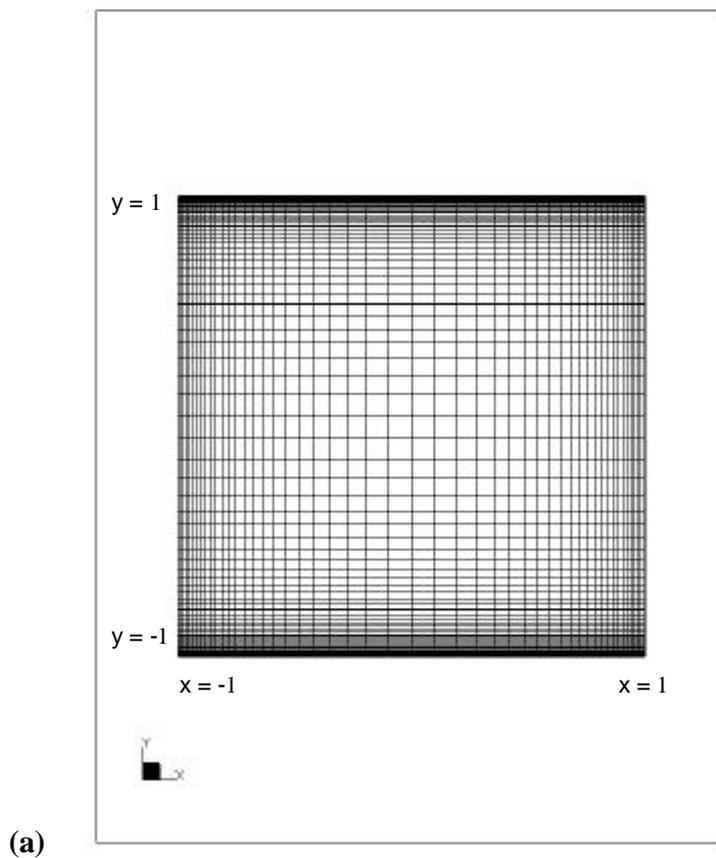
1. **Pressure drop in the supplying duct due to a nonuniform field and bending of the duct**
2. **Transition from duct flow to manifold flow (the manifold problem)**
3. **Transition from manifold flow to nozzle flow**
4. **Transition from duct flow to jet flow (the nozzle problem) and the meniscus effect**
5. **Non-uniform field effects and jet stability**
6. **Heat transfer analysis, including thermocapillary convection**

Figure 2 MHD problems in for the lower part of the divertor.

Main Thermal, Fluid, and MHD Problems Associated With a Jet Divertor (cont.)



7. **Impact of a jet on a liquid metal surface**
8. **Impact of a jet on a solid wall**
9. **LM film or rivulet**
10. **The problem of draining**

Figure 3 Shercliff solution. Geometry and co-ordinate system.**Figure 4** Shercliff solution. Grid used: (a) for $Ha = 100$; (b) for $Ha = 200$.

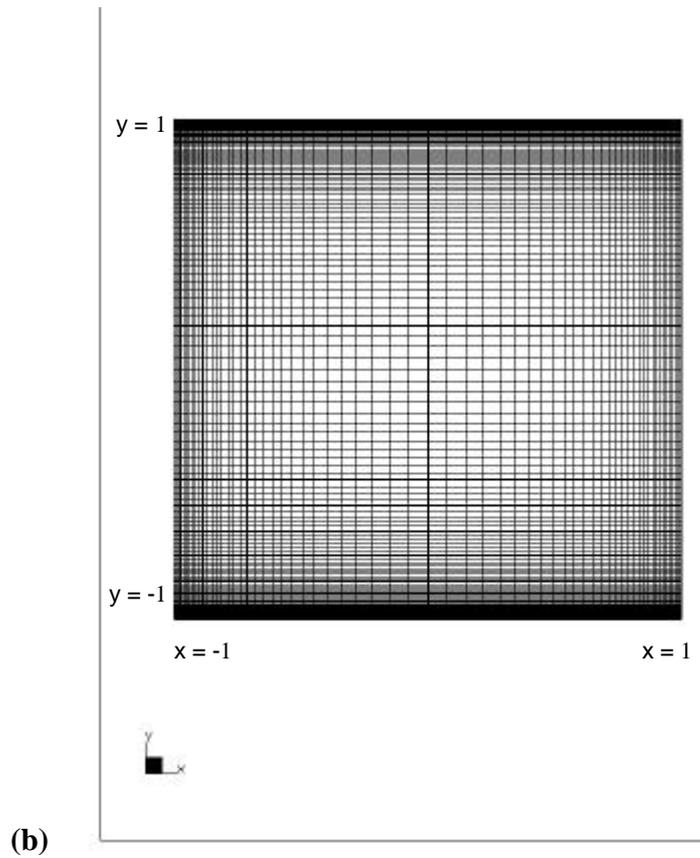


Figure 5 Shercliff solution. Electric potential in the plane $y = 0$ for a square duct and for $Ha = 100$ (numerical solution - solid lines; exact solution - crosses).

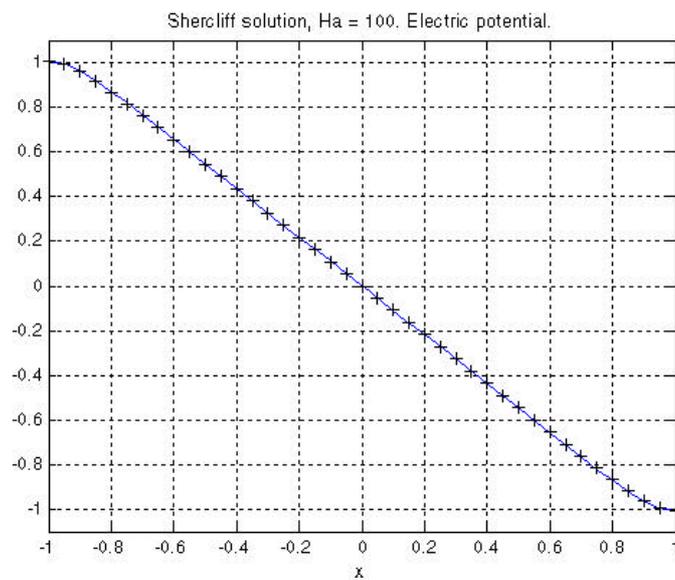


Figure 6 Shercliff solution. Axial velocity in the plane $x = 0$ for a square duct and for $Ha = 100$ (numerical solution - solid lines; exact solution - crosses).

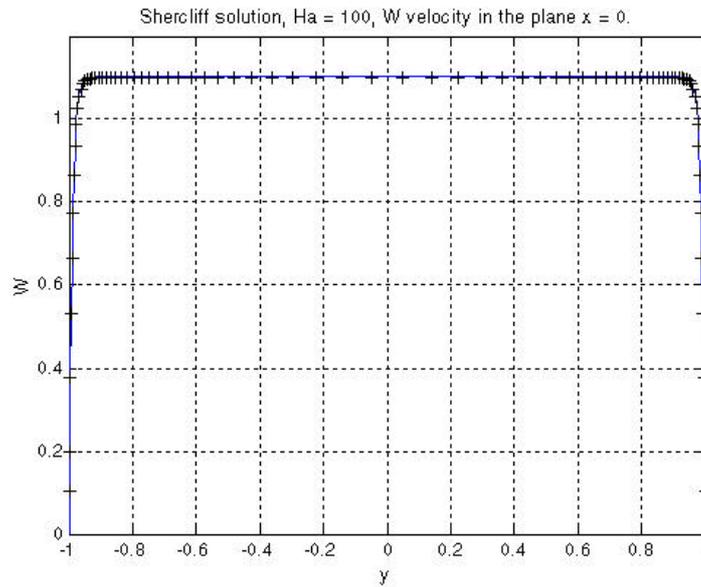


Figure 7 Shercliff solution. Axial velocity in the plane $y = 0$ for a square duct and for $Ha = 100$ (numerical solution - solid lines; exact solution - crosses).

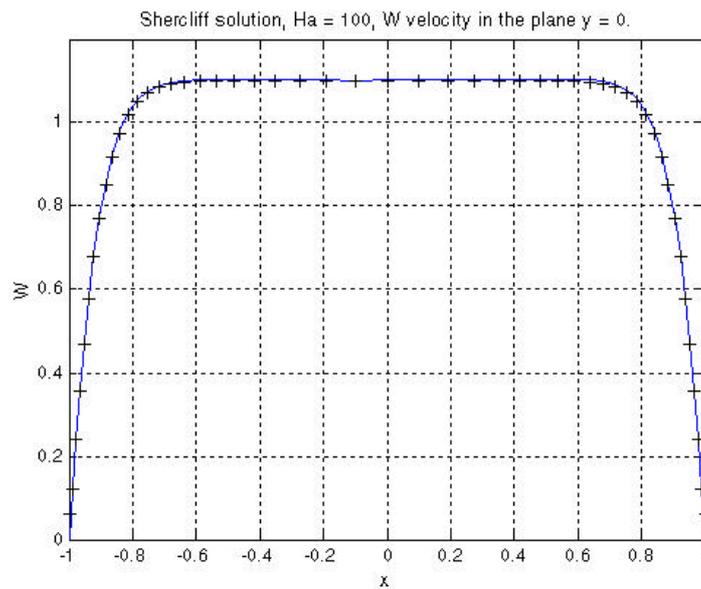


Figure 8 Shercliff solution. Electric potential at $x = 0$ for a square duct and for $Ha = 200$ (numerical solution - solid lines; exact solution - crosses).

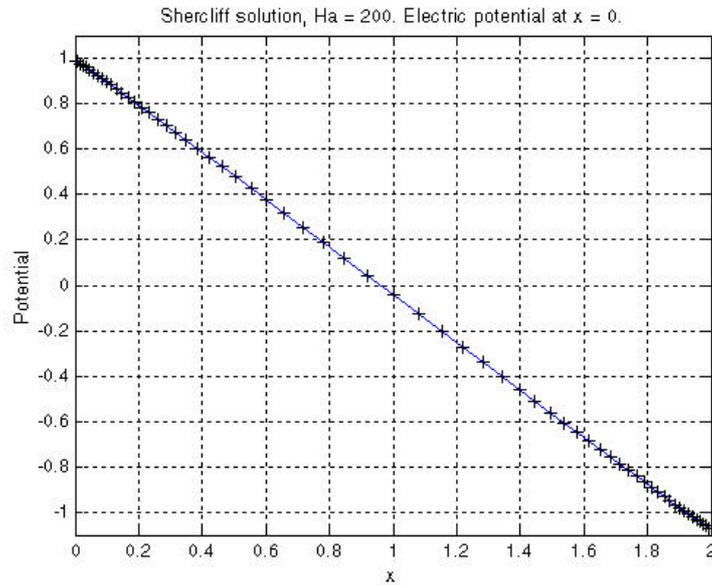


Figure 9 Shercliff solution. Axial velocity in the plane $y = 0$ for a square duct and for $Ha = 200$ (numerical solution - solid lines; exact solution - crosses).

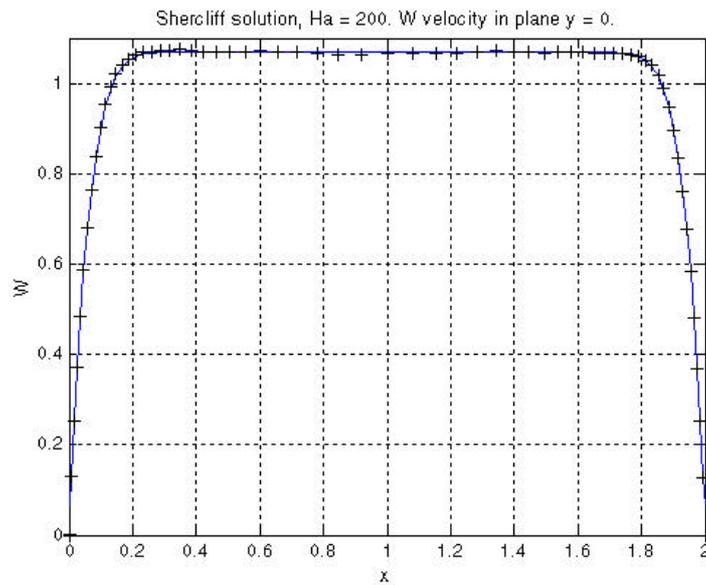


Figure 10 Shercliff solution. Axial velocity in the plane $x = 0$ for a square duct and for $Ha = 200$ (numerical solution - solid lines; exact solution - crosses).

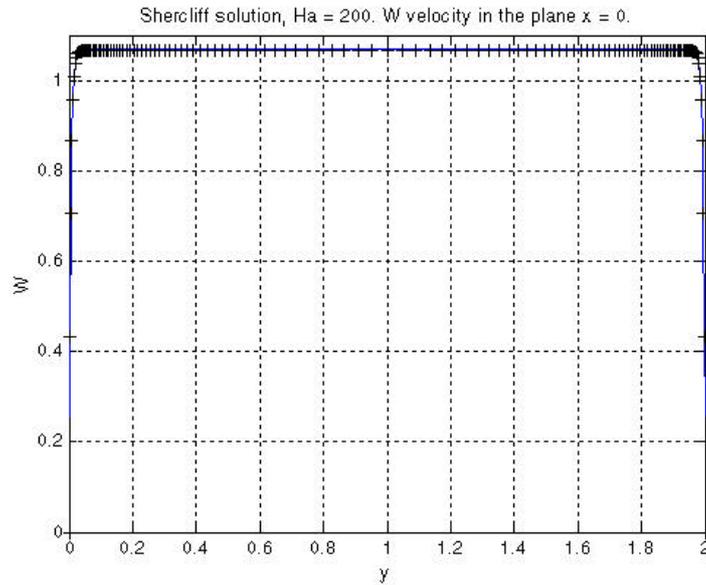


Figure 11 Hunt solution. Geometry and co-ordinate system.

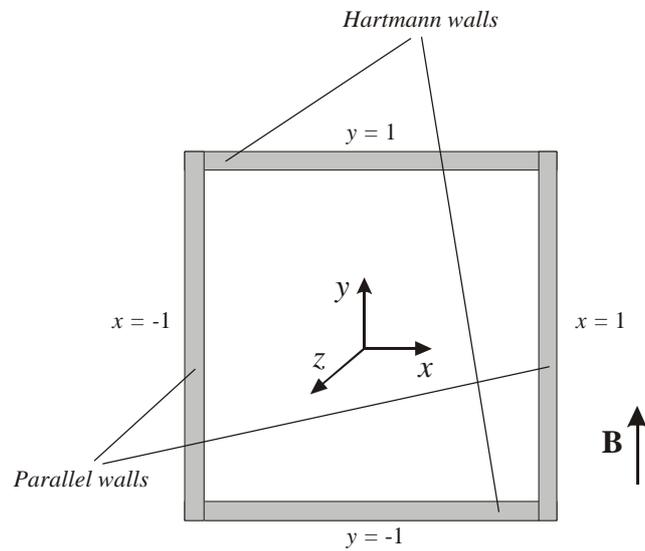


Figure 12 Hunt solution for a square duct with perfectly conducting walls. Axial velocity in the plane $y = 0$ for $Ha = 100$ (numerical solution - solid lines; exact solution - stars).

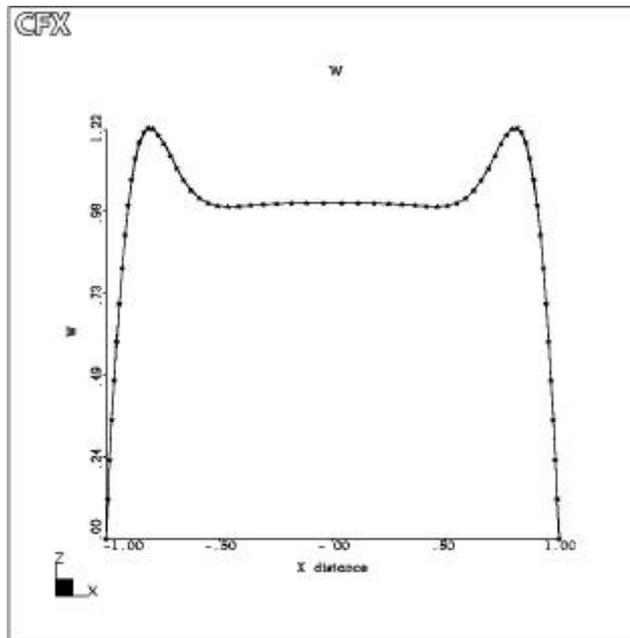


Figure 13 Hunt solution for a square duct with perfectly conducting walls. Electric potential in the plane $y = 0$ for $Ha = 100$ (numerical solution - solid lines; exact solution - stars).

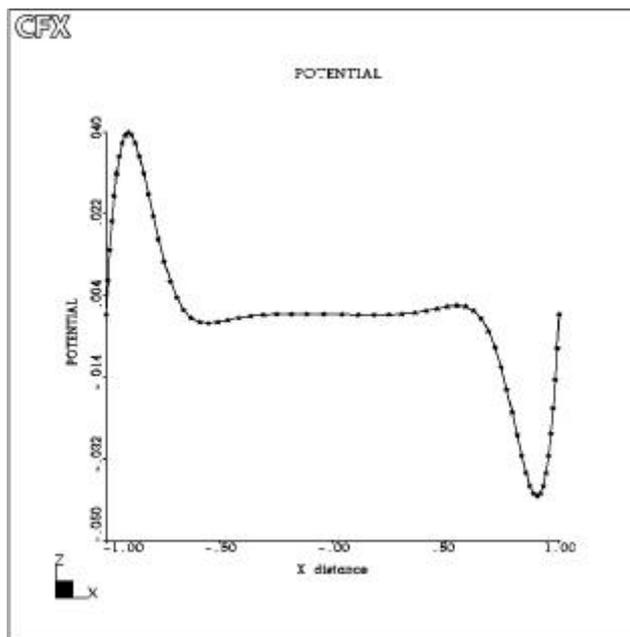


Figure 14 Hunt solution for a square duct with perfectly conducting Hartmann walls and electrically insulating parallel walls. Electric potential in the plane $y = 0$ for $Ha = 100$ (numerical solution - solid lines; exact solution - stars).

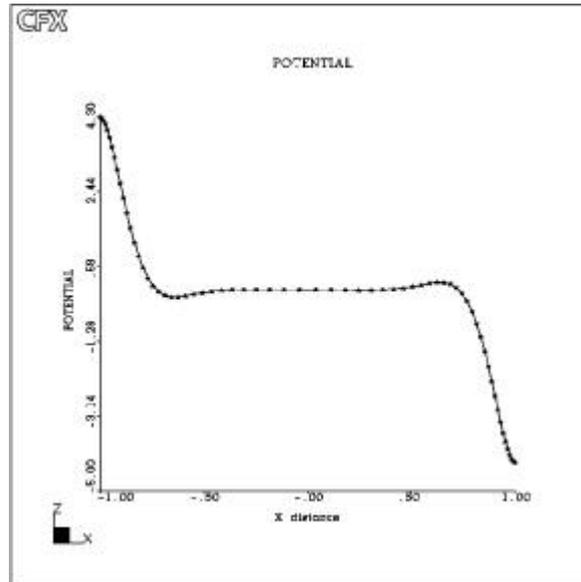


Figure 15 Hunt solution for a square duct with perfectly conducting Hartmann walls and electrically insulating parallel walls. Axial velocity in the plane $y = 0$ for $Ha = 100$ (numerical solution - solid lines; exact solution - stars).

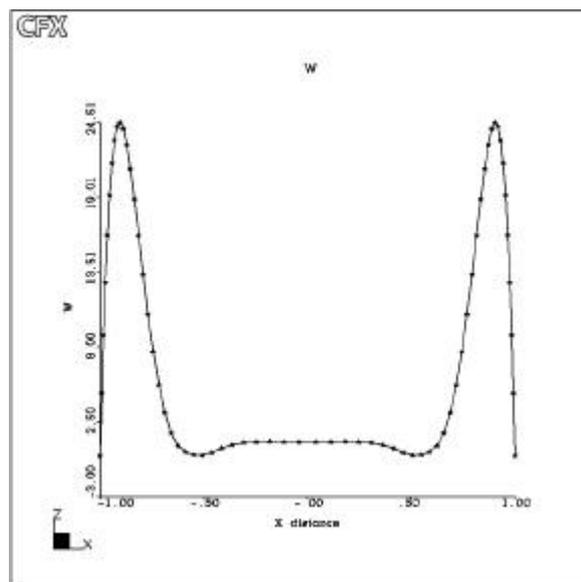


Figure 16 Hunt solution for a square duct with perfectly conducting Hartmann walls and electrically insulating parallel walls. Electric potential in the plane $y = 0$ for $Ha = 200$ (numerical solution - solid lines; exact solution - stars).

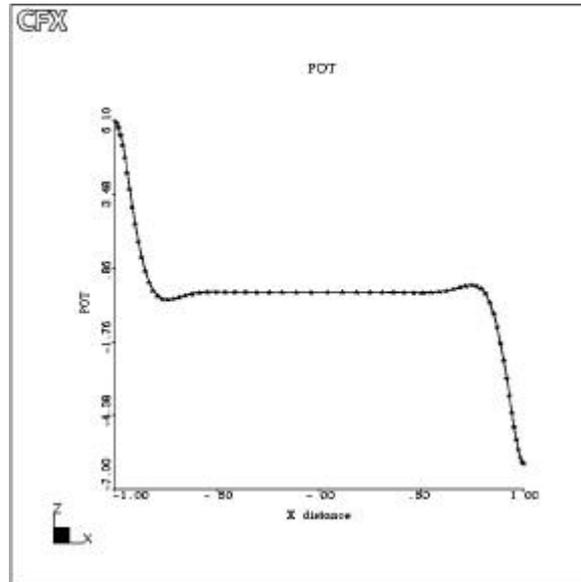


Figure 17 Hunt solution for a square duct with perfectly conducting Hartmann walls and electrically insulating parallel walls. Axial velocity in the plane $y = 0$ for $Ha = 200$ (numerical solution - solid lines; exact solution - stars).

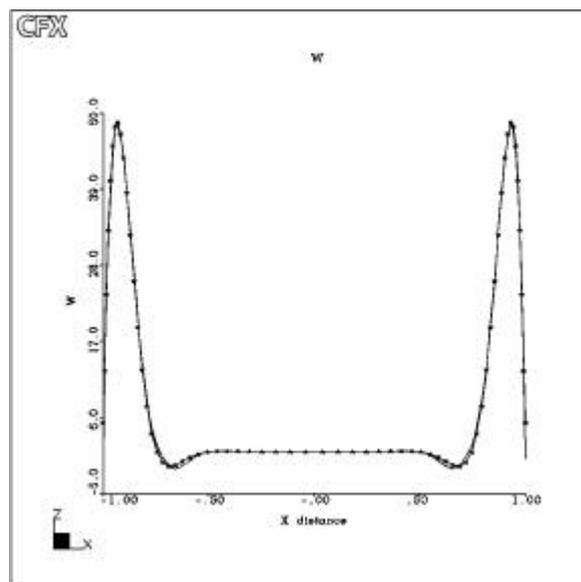


Figure 18 Flow in a duct with a 1:2 symmetric expansion in a transverse magnetic field.

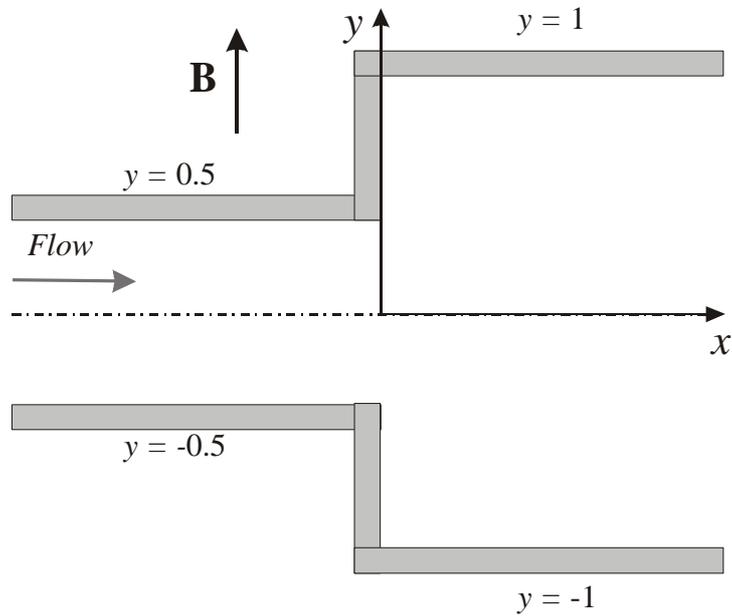


Figure 19 Inertialess flow in a duct with an expansion. Streamlines for $Ha = 200$.

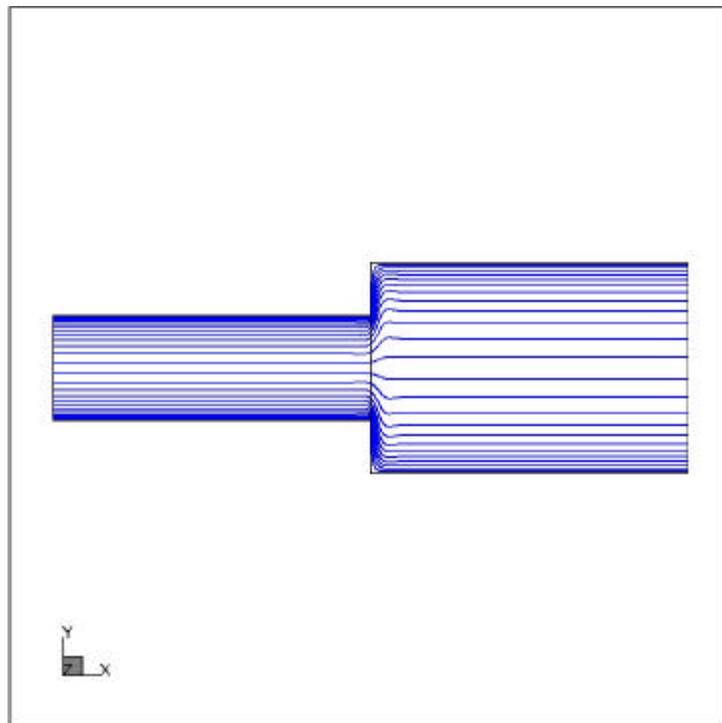


Figure 20 Inertialess flow in a duct with an expansion. Velocity profiles in the duct ($x = -3$) and at the junction ($x = 0$) for $Ha = 200$. Solid lines - CFX numerical solution, stars - numerical solution obtained in ([26]).

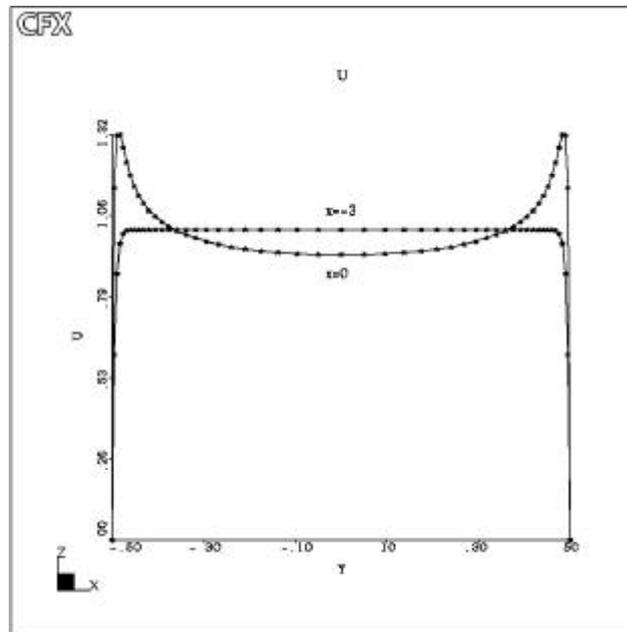


Figure 21 Inertialess flow in a duct with an expansion. Core velocity ($y = 0$) for $Ha = 200$.

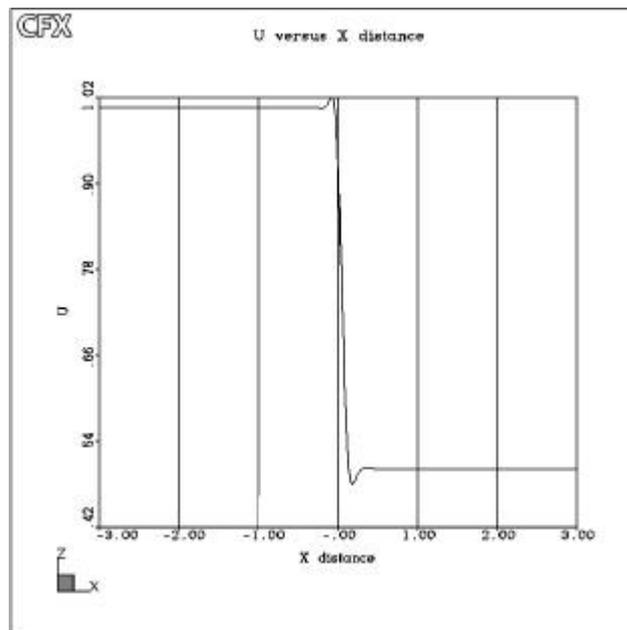


Figure 22 Inertialess flow in a duct with an expansion. Pressure distribution for $Ha = 200$.

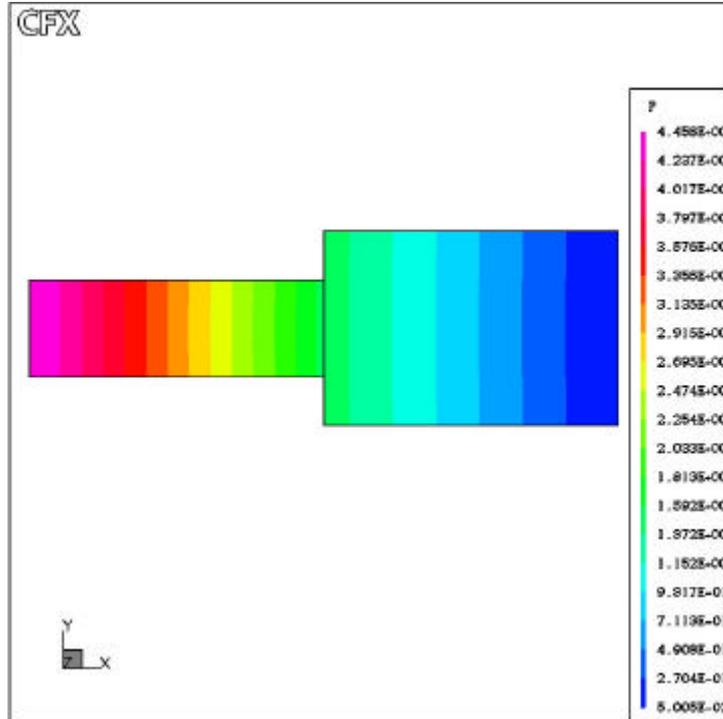


Figure 23 Inertialess flow in a duct with an expansion. Pressure at the top of the wider duct ($y = 1$) for $Ha = 200$.

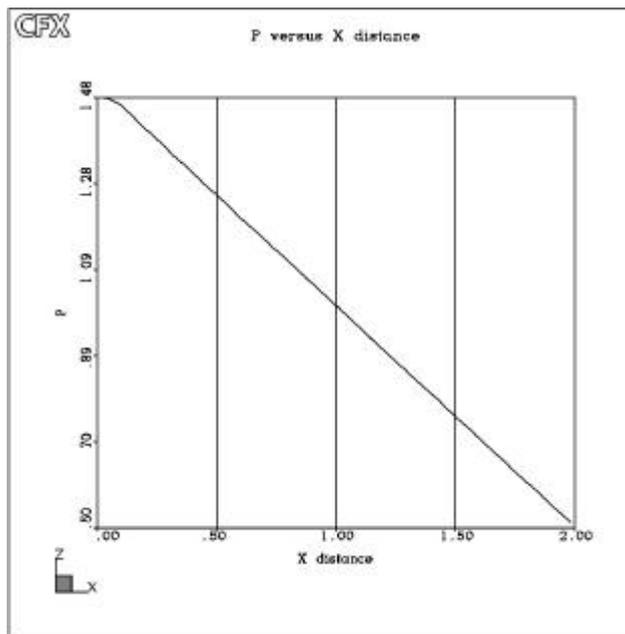


Figure 24 Inertial flow in a duct with an expansion. Streamlines for $Ha = 200$, $N = 1$.

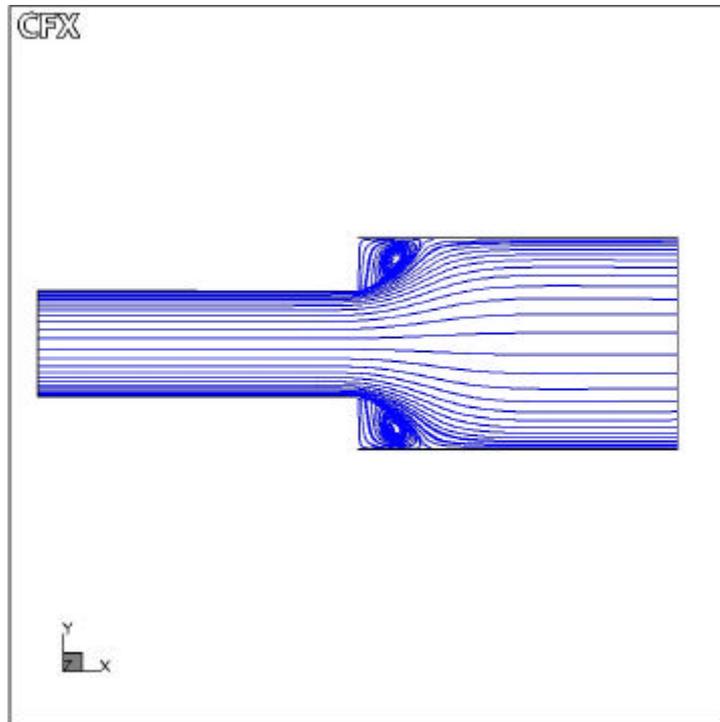


Figure 25 Inertial flow in a duct with an expansion. Velocity profiles in the duct ($x = -3$) and at the junction ($x = 0$) for $Ha = 200$, $N = 1$.

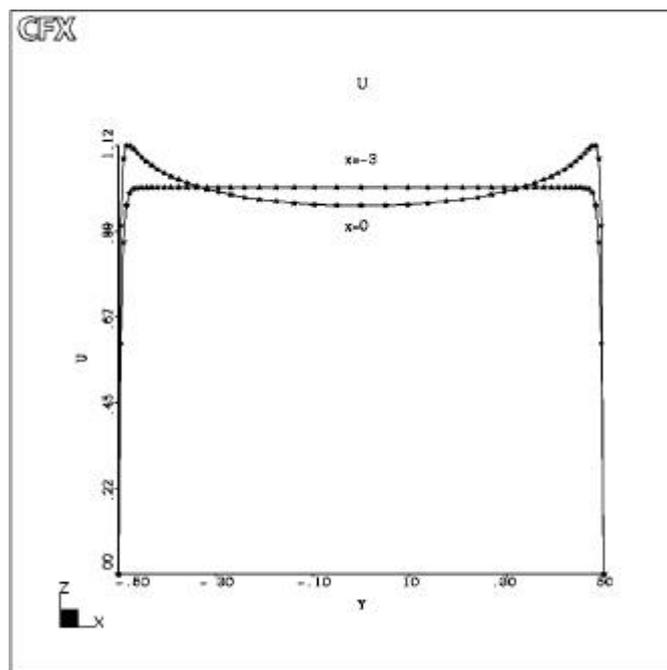


Figure 26 Inertial flow in a duct with an expansion. Pressure at the top of the wider duct ($y = 1$) for $Ha = 200, N = 1$.

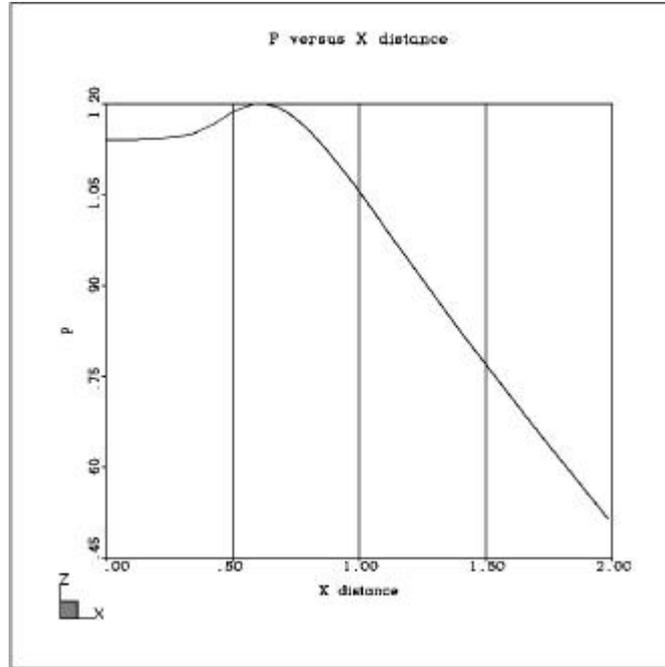


Figure 27 Inertial flow in a duct with an expansion. Pressure distribution for $Ha = 200, N = 1$.

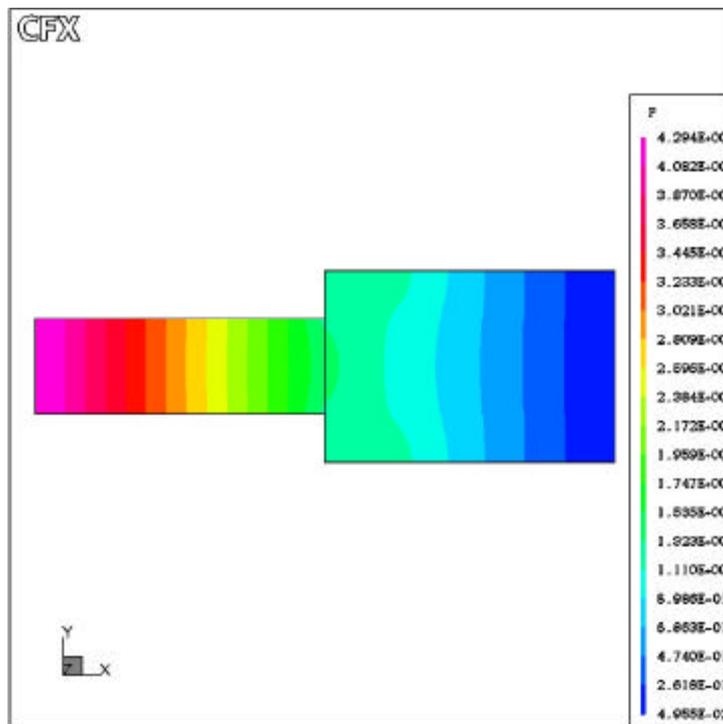


Figure 28 Flow in an asymmetric duct with an expansion in a transverse magnetic field.

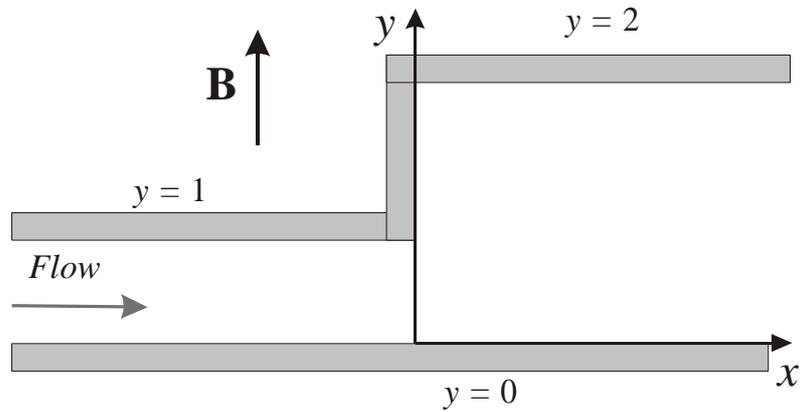


Figure 29 Inertial flow in an asymmetric duct with an expansion. Streamlines for $Ha = 200, N = 1$.

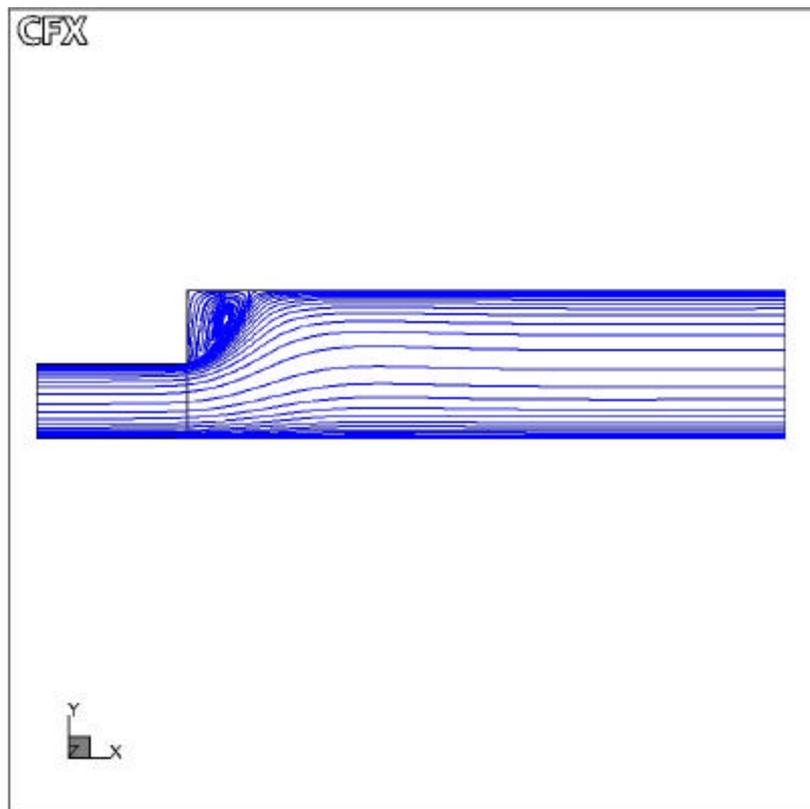


Figure 30 Inertial flow in an asymmetric duct with an expansion. Pressure distribution for $Ha = 200, N = 1$.

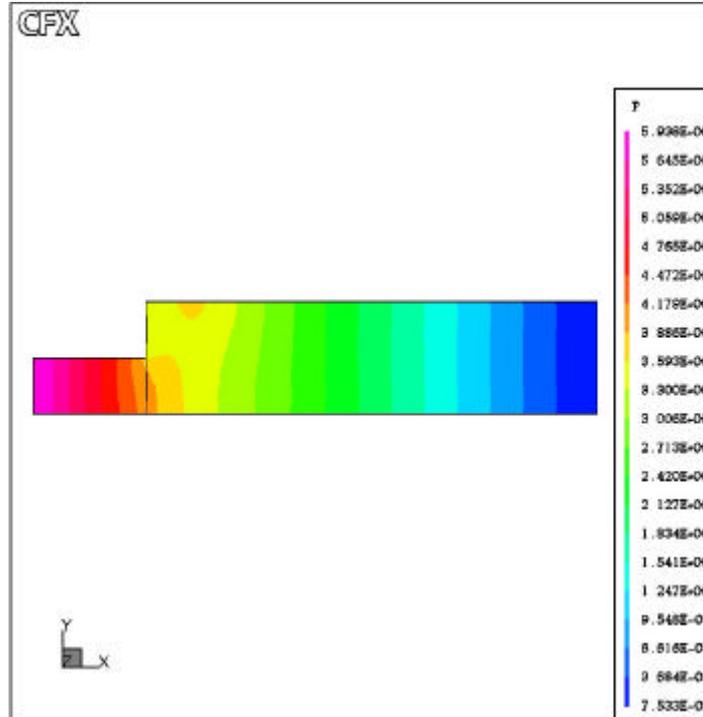


Figure 31 Inertial flow in an asymmetric duct with an expansion. Pressure at the top of the wider duct ($y = 1$) for $Ha = 200, N = 1$.

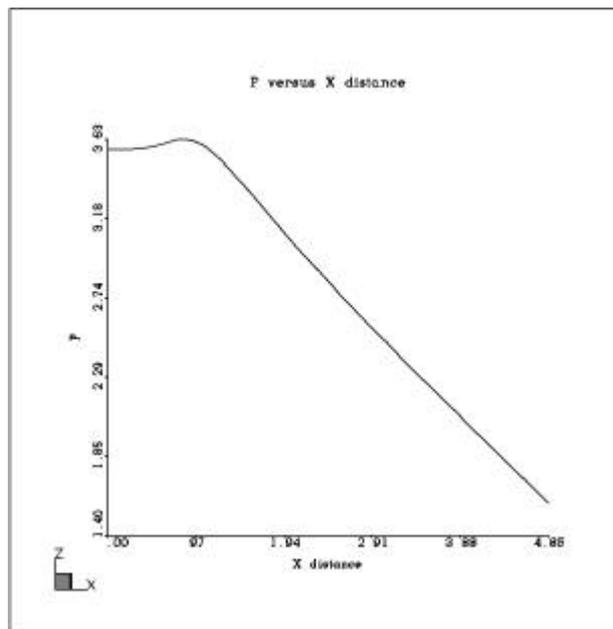


Figure 32 Flow in a square duct in a non-uniform transverse magnetic field.

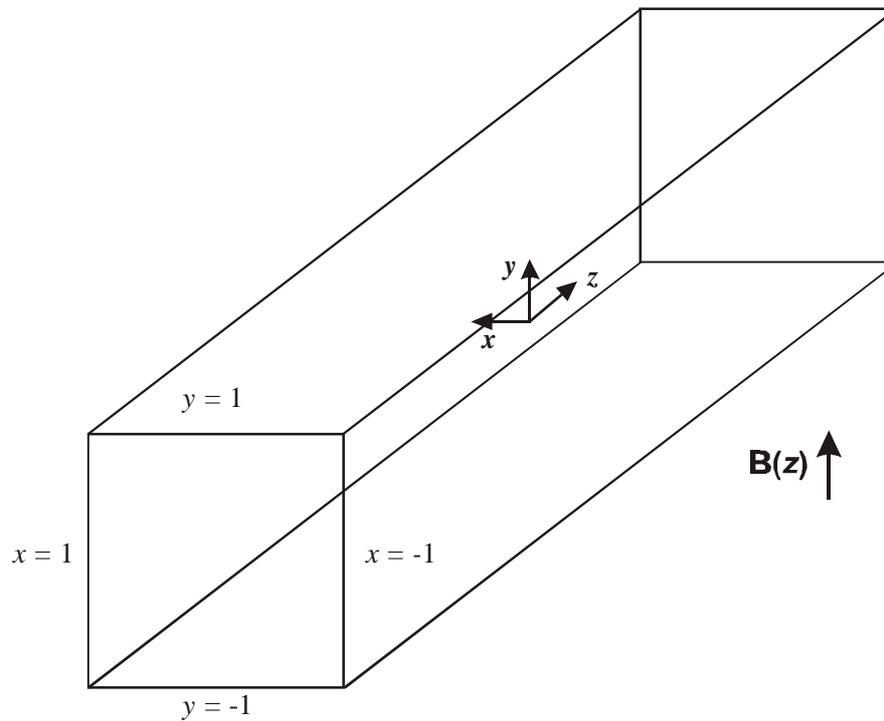


Figure 33 Flow in a square duct in a non-uniform transverse magnetic field.
Magnetic field versus axial co-ordinate.

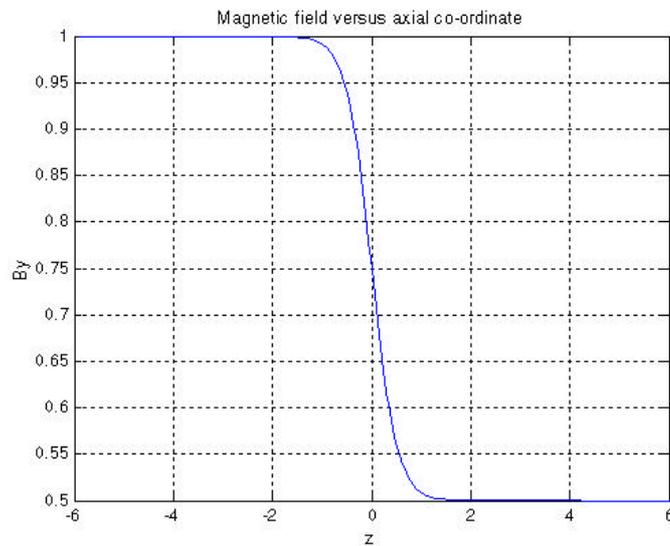


Figure 34 Axial velocity profiles for inertialess flow in a square duct in a non-uniform transverse magnetic field. $Ha = 50$. Line $y = 0$ and (a) $z = -6$; (b) $z = -2$; (c) $z = -1$; (d) $z = -0.5$; (e) $z = 0$; (f) $z = 0.5$; (g) $z = 1$; (h) $z = 2$; (i) $z = 6$.

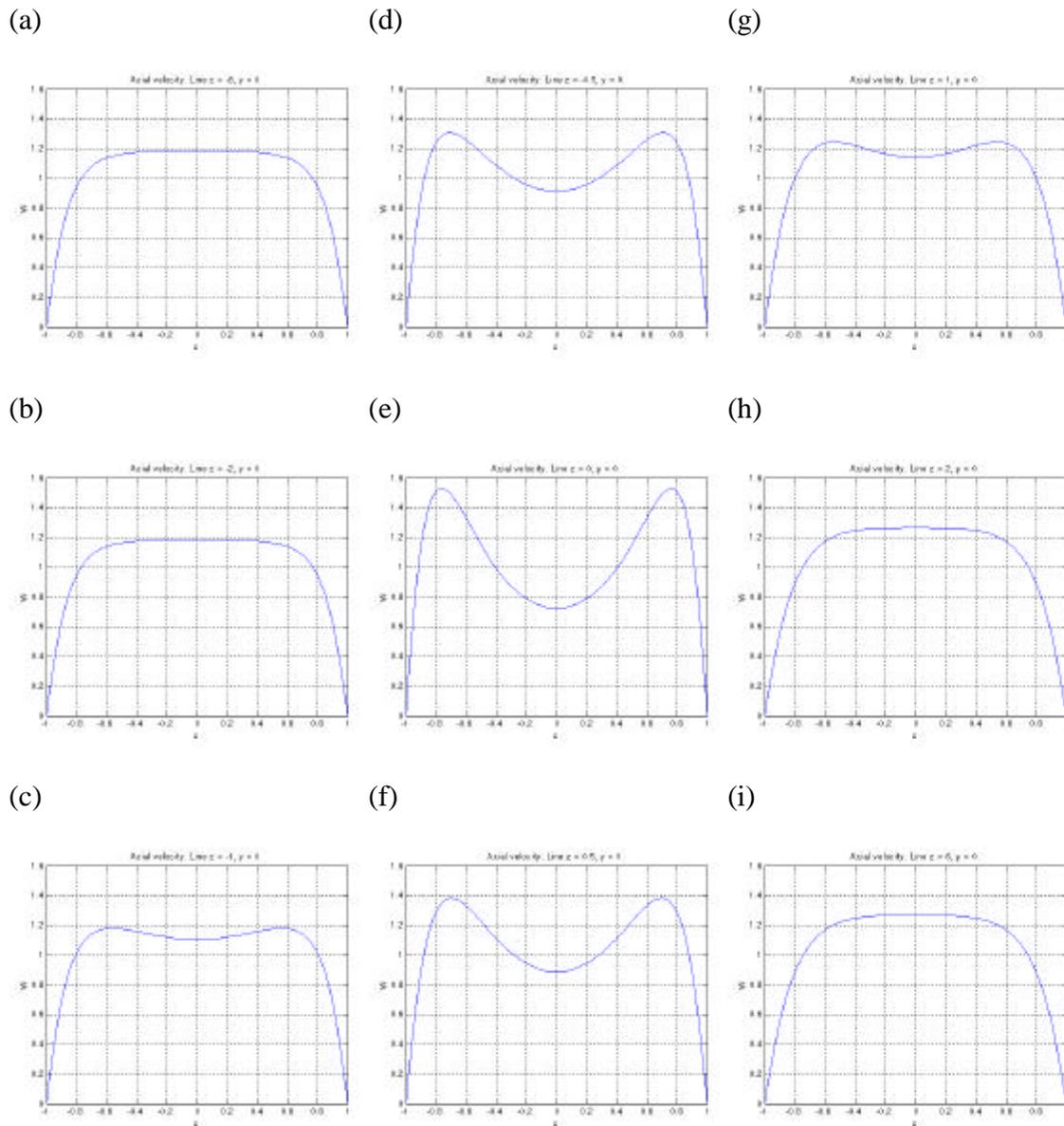


Figure 35 Axial velocity profiles in the inertialess flow in a square duct in a non-uniform transverse magnetic field for $Ha = 50$.

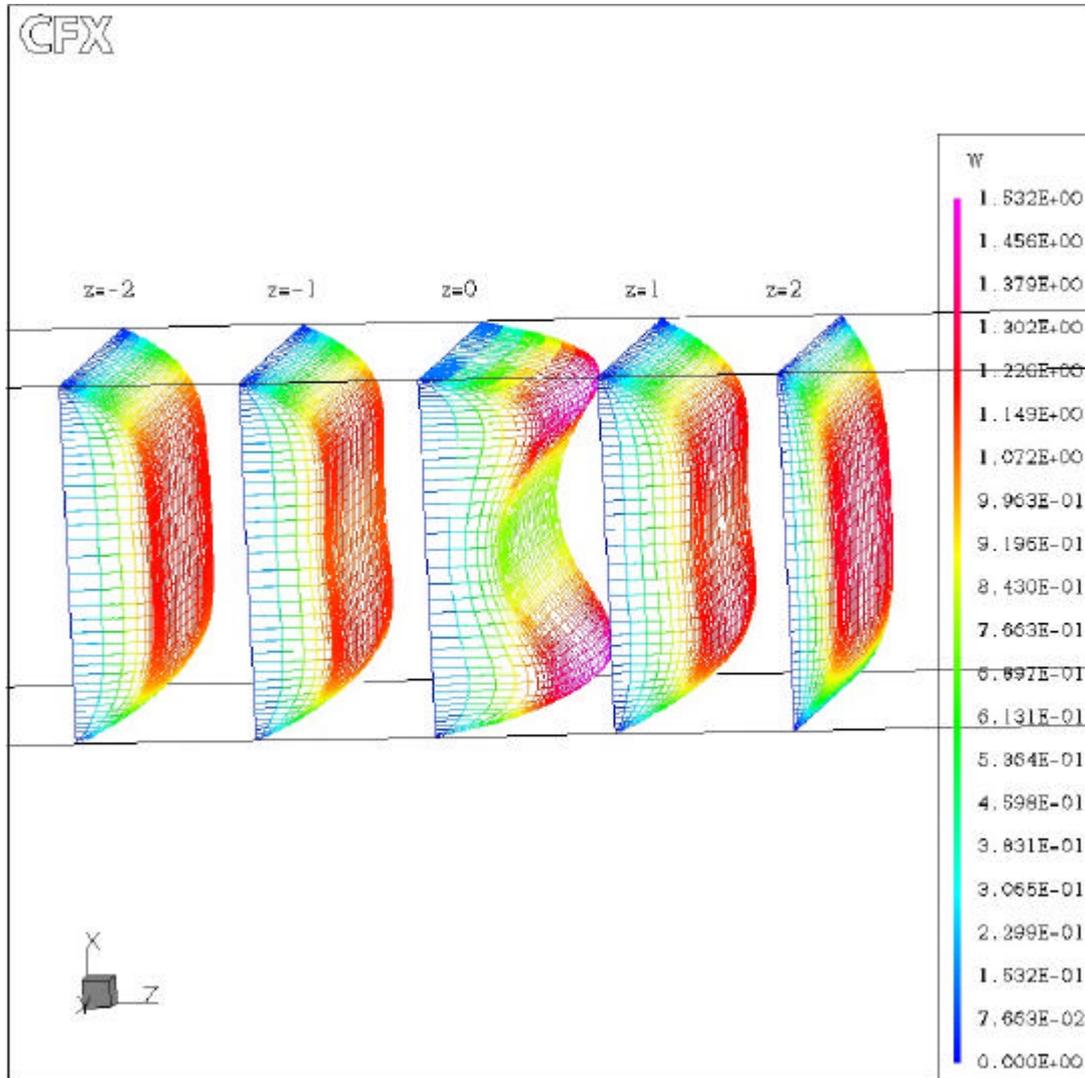


Figure 36 Streamlines in the inertialess flow in a square duct in a non-uniform transverse magnetic field in the plane $y = 0$ for $Ha = 50$.

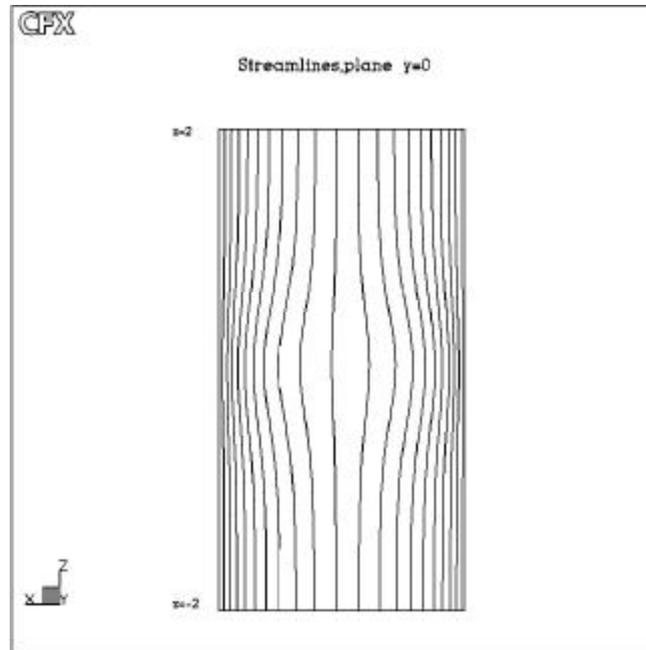


Figure 37 Pressure in the inertialess flow in a square duct in a non-uniform transverse magnetic field on the central line of the duct $x = y = 0$ (broken line) and near the wall $x = y = 1$ (solid line). $Ha = 50$.

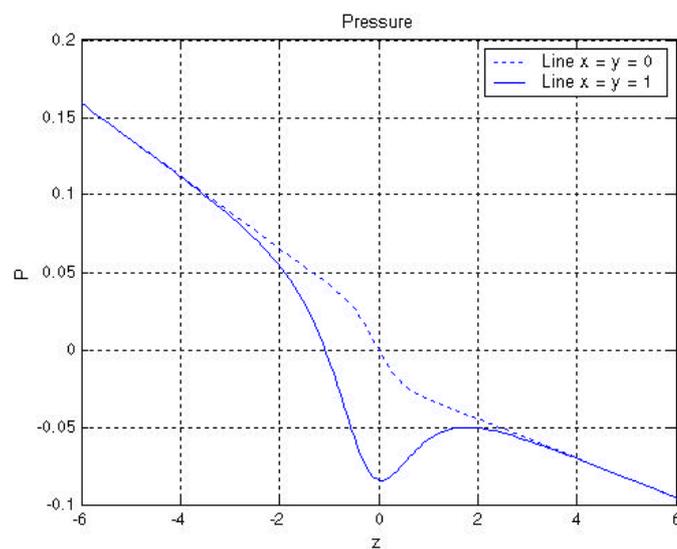


Figure 38 Electric current lines in the inertialess flow in a square duct in a non-uniform transverse magnetic field for $Ha = 50$.

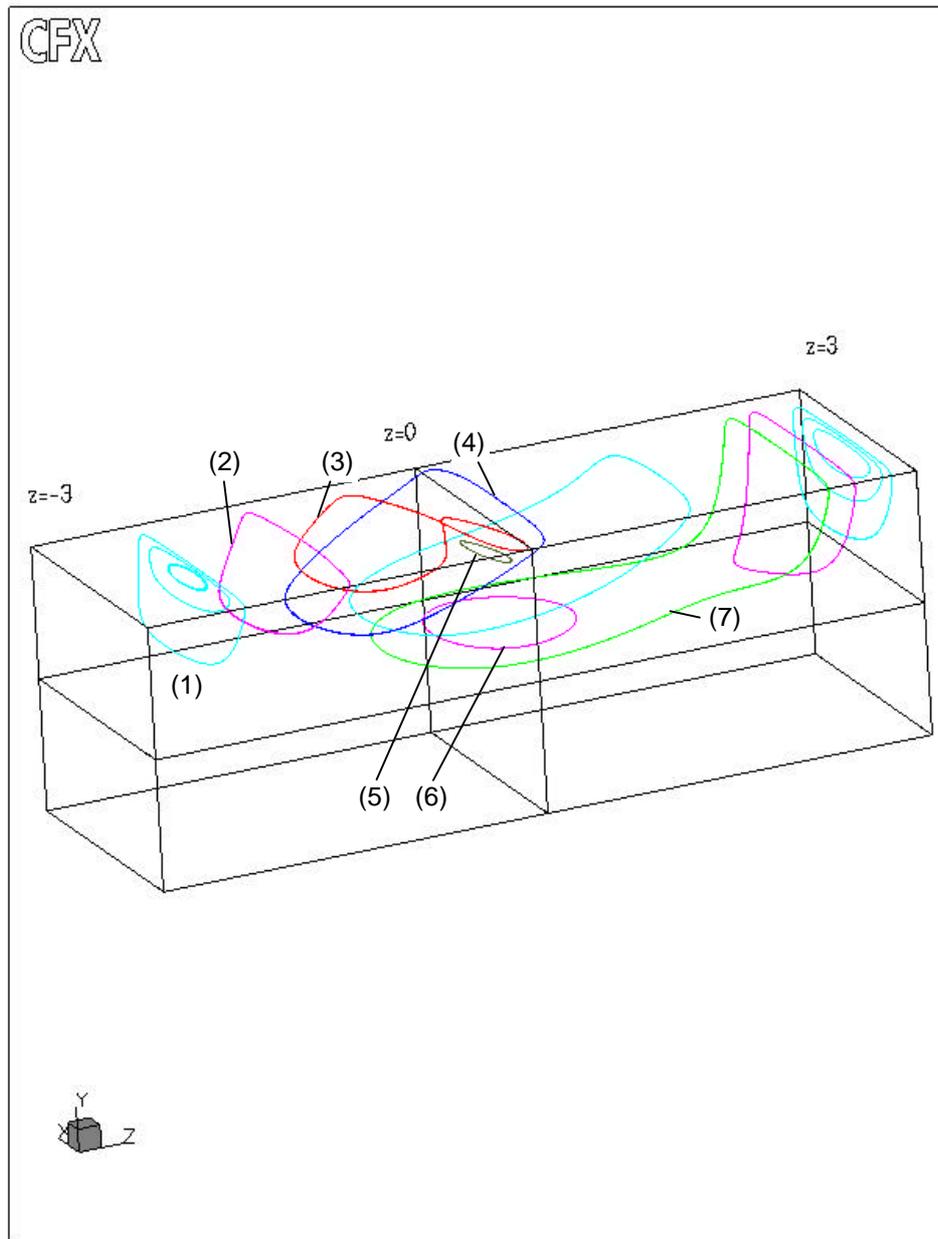


Figure 39 Axial velocity profiles in the inertialess flow in a square duct in a non-uniform transverse magnetic field in the plane $y = 0$ for $Ha = 200$.

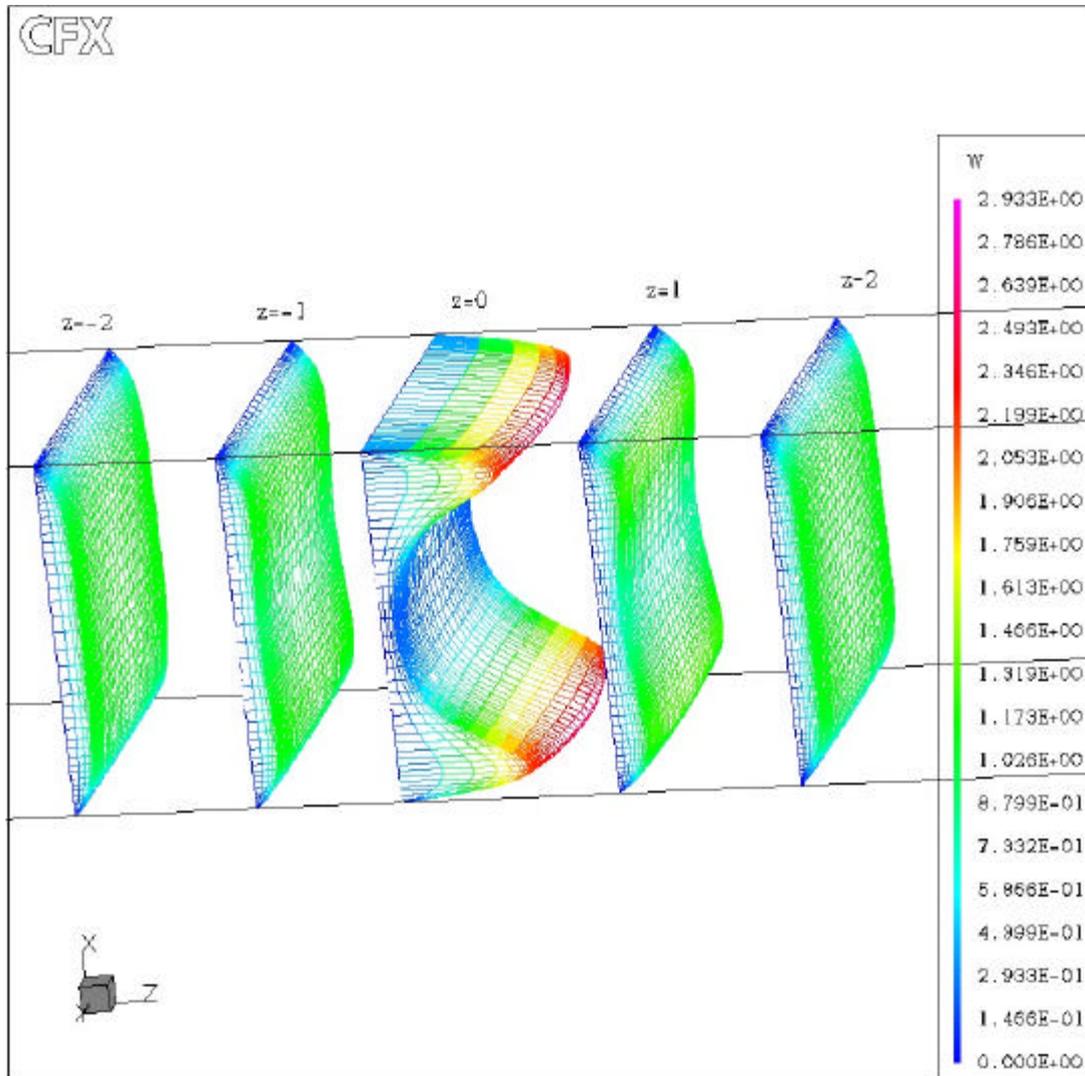


Figure 40 Streamlines in the inertialess flow in a square duct in a non-uniform transverse magnetic field in the plane $y = 0$ for $Ha = 200$.

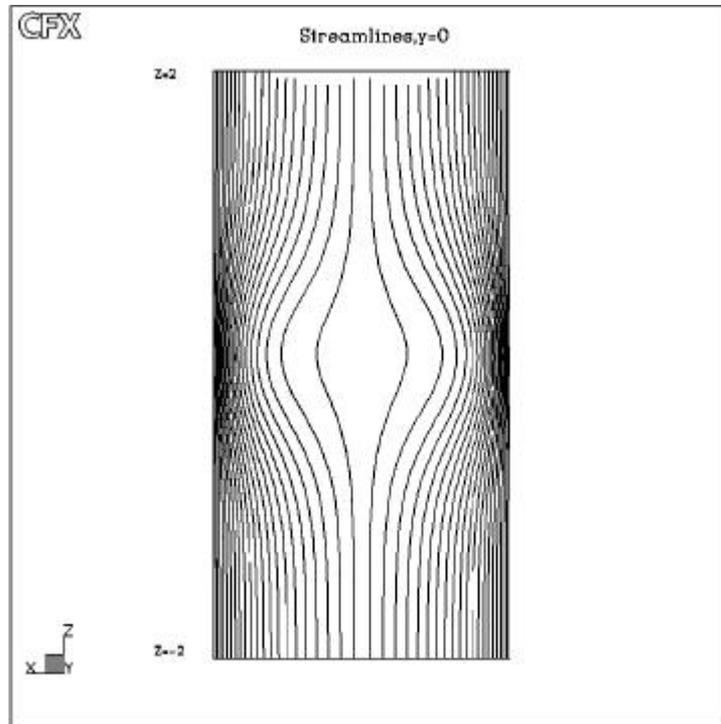


Figure 41 Pressure in the inertialess flow in a square duct in a non-uniform transverse magnetic field on the central line of the duct $x = y = 0$ (broken line) and near the wall $x = y = 1$ (solid line). $Ha = 200$.

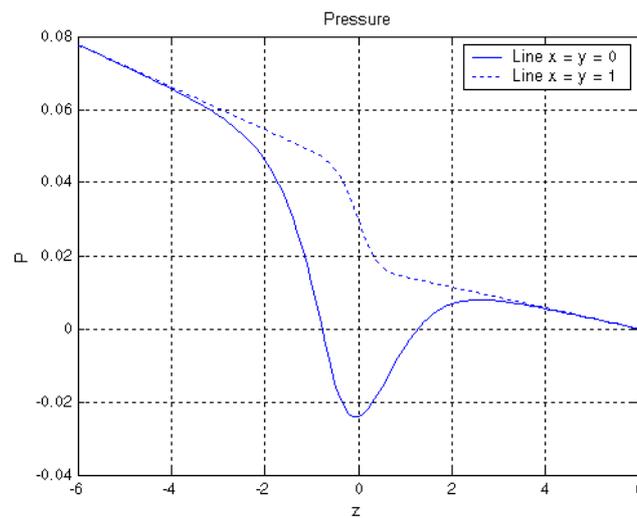


Figure 42 Electric current lines in the inertialess flow in a square duct in a non-uniform transverse magnetic field in the plane $y = 0$ for $Ha = 200$.

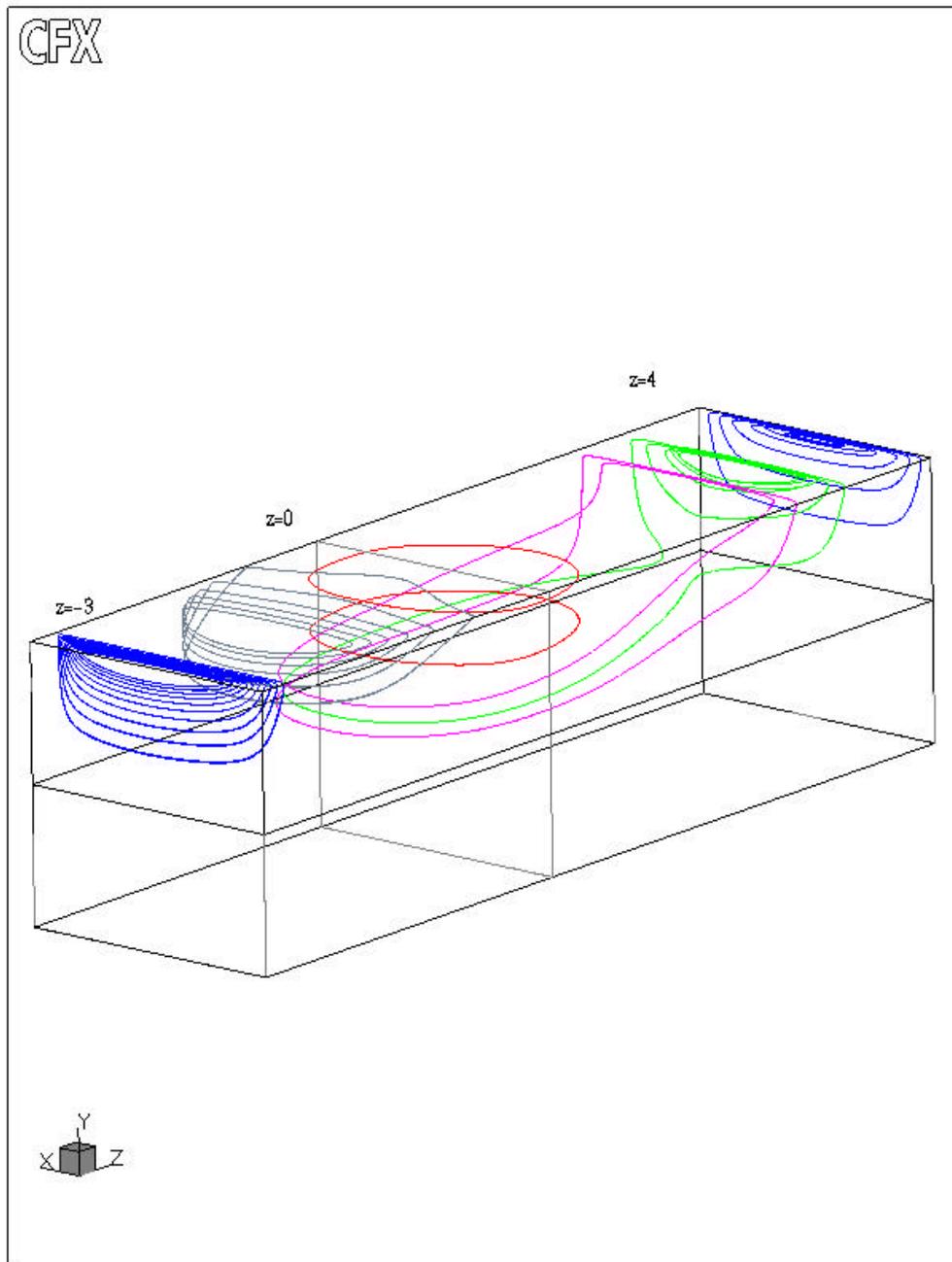


Figure 43 Pressure variation in the inertialess flow in a square duct in a non-uniform transverse magnetic field in the plane $y = 0$ for $Ha = 200$.

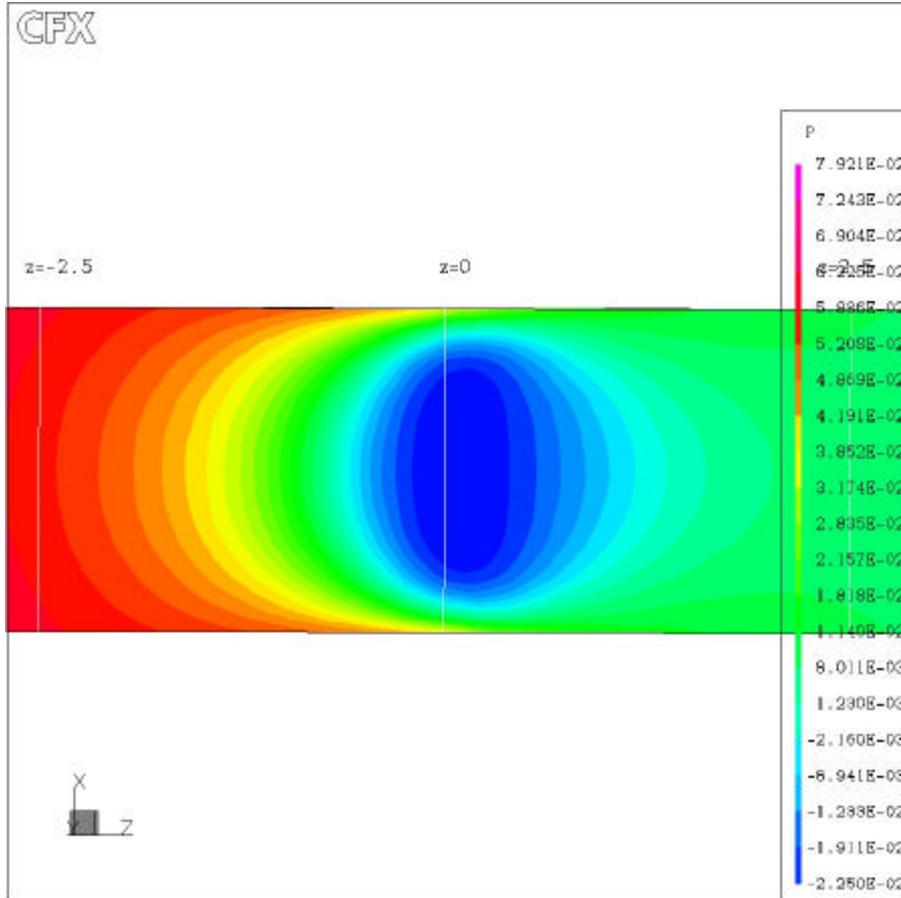


Figure 44 Liquid metal drop in a strong, vertical magnetic field.

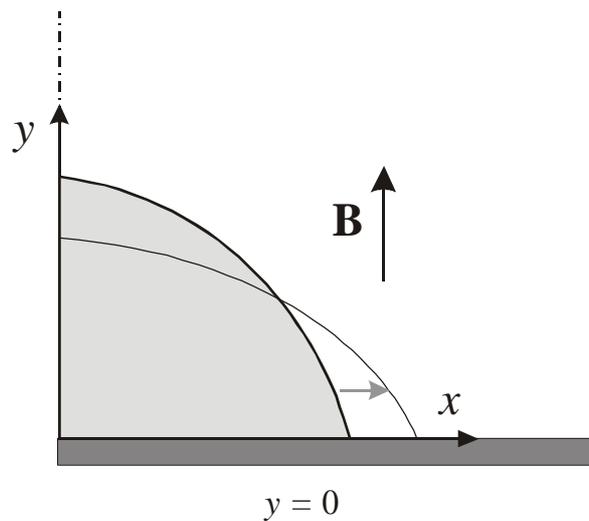


Figure 45 Liquid metal drop in a strong, vertical magnetic field after 0.185 s.
Solid line - asymptotic solution.

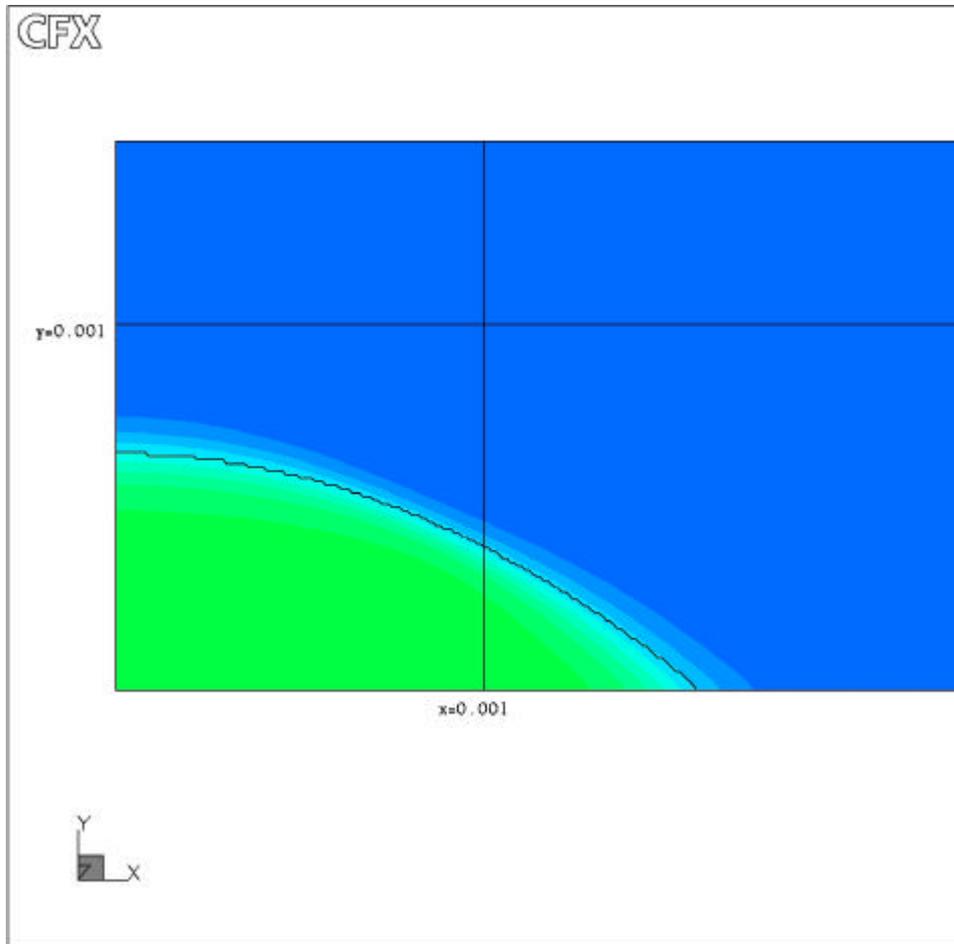


Figure 46 Liquid metal drop in a strong, vertical magnetic field. Horizontal velocity for both phases ("air" and liquid metal). Solid line - drop surface (asymptotic solution).

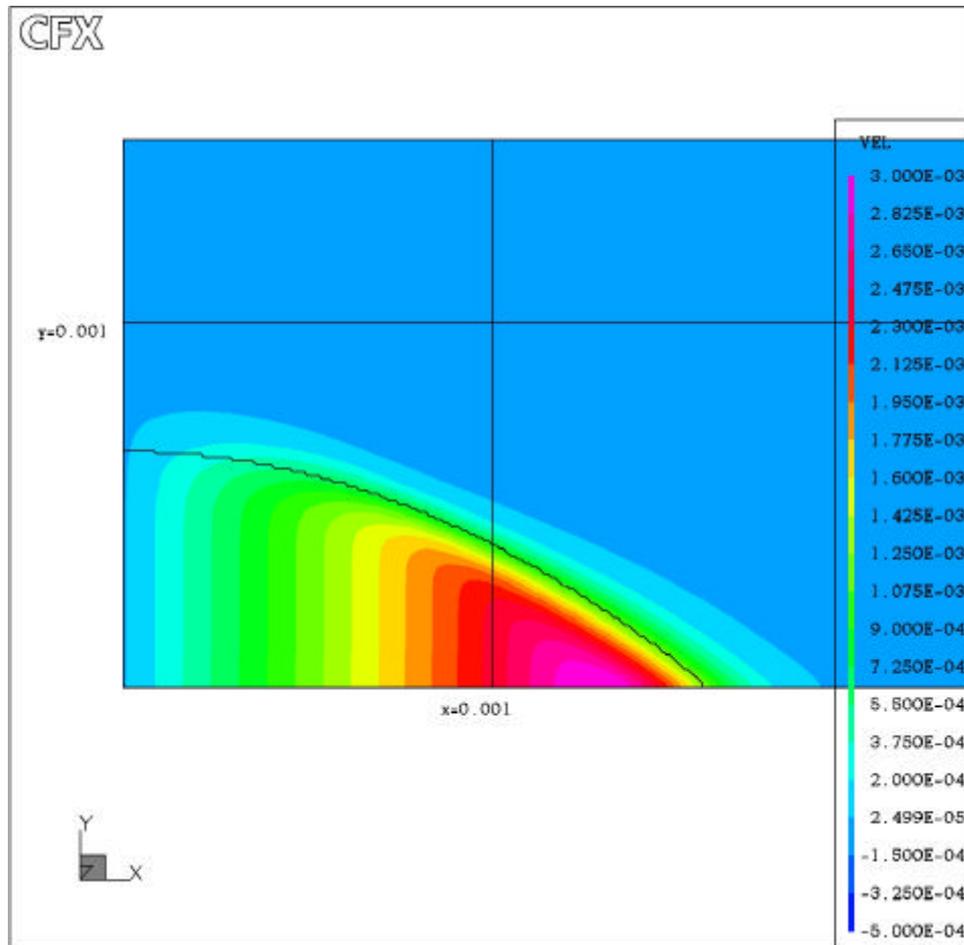


Figure 47 Liquid metal drop in a strong, vertical magnetic field. Vertical velocity for both phases ("air" and liquid metal). Solid line - drop surface (asymptotic solution).

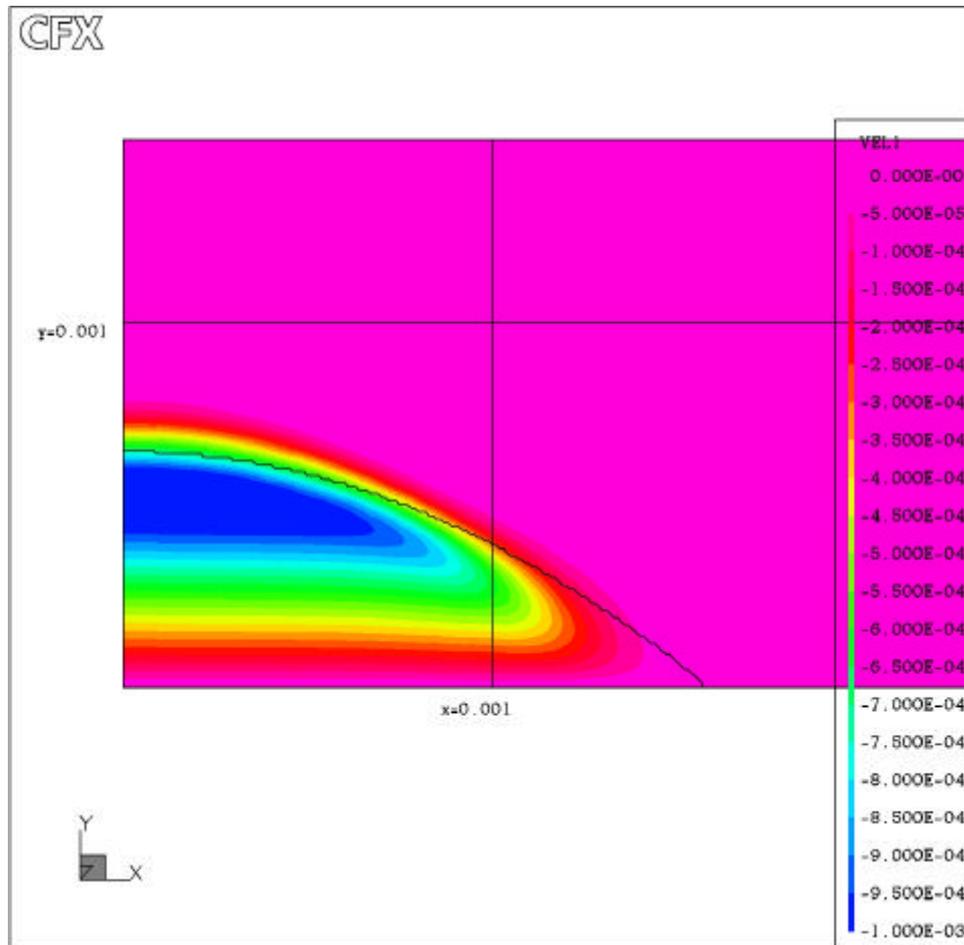


Figure 48 Liquid metal drop in a strong, vertical magnetic field. Pressure for both phases ("air" and liquid metal). Solid line - drop surface (asymptotic solution).

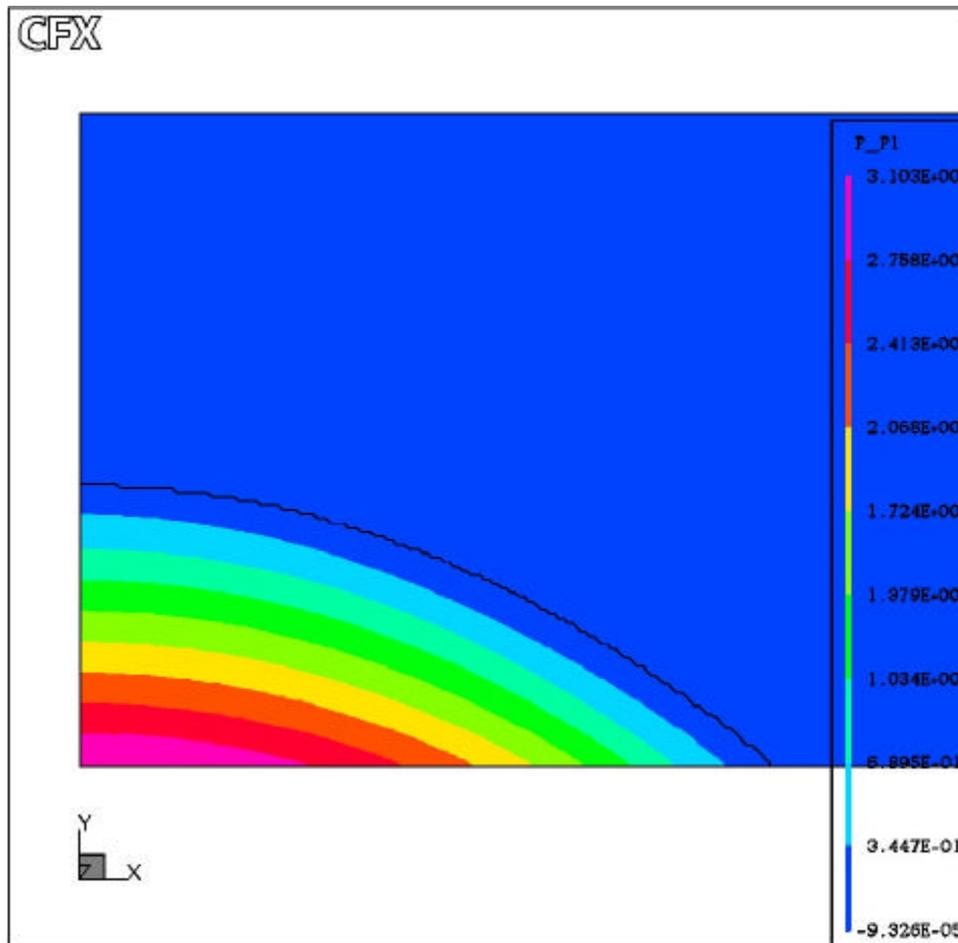


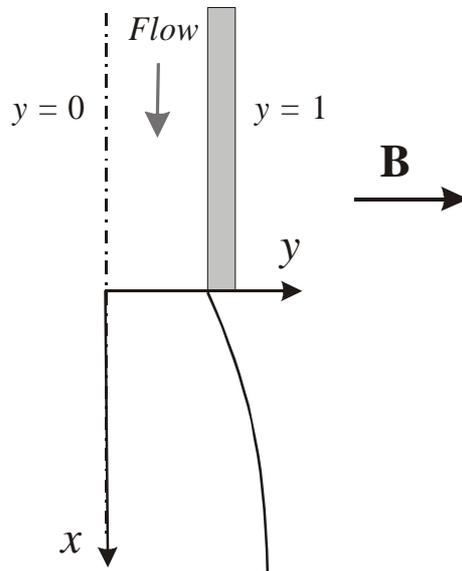
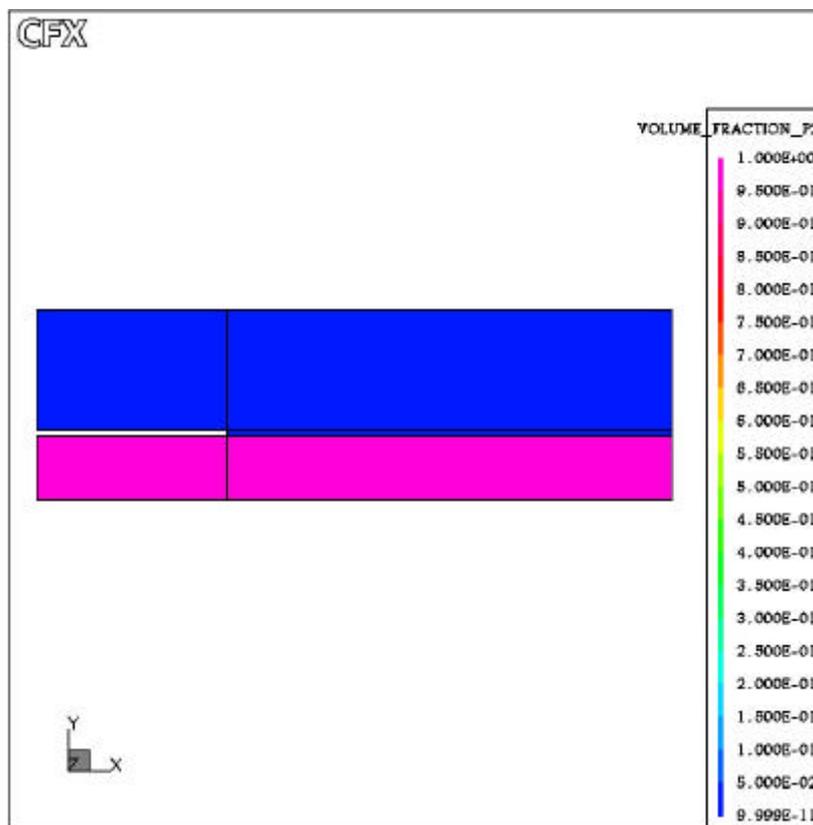
Figure 49 Liquid metal jet in a strong, transverse magnetic field.**Figure 50** Liquid metal jet in a strong, transverse magnetic field. Variation of jet thickness in a uniform field for $E = -1$, $Ha = 200$.

Figure 51 Liquid metal jet in a strong, transverse magnetic field. Velocity profile in a uniform field for $E = -1$, $Ha = 200$. Velocity in the duct (solid line) and in the jet region (stars).

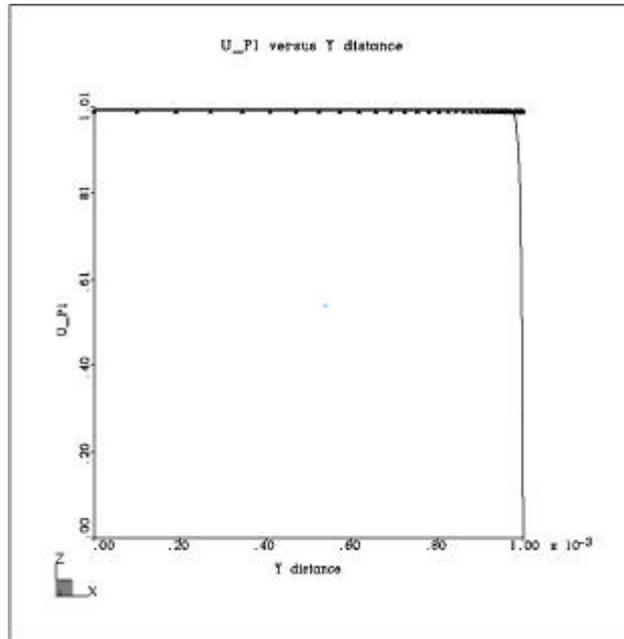


Figure 52 Liquid metal jet in a strong, transverse magnetic field. Variation of pressure in a uniform field for $E = -1$, $Ha = 200$.

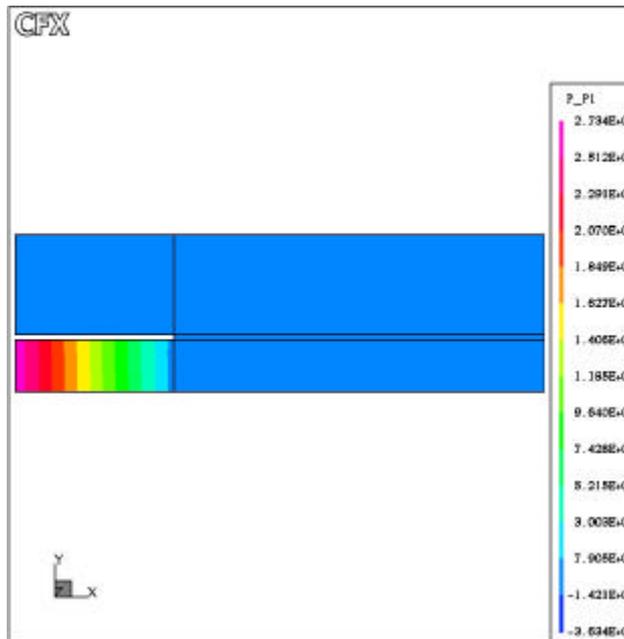


Figure 53 Liquid metal jet in a strong, transverse magnetic field. Variation of pressure in a uniform field at $y = 0$ for $E = -1$, $Ha = 200$.

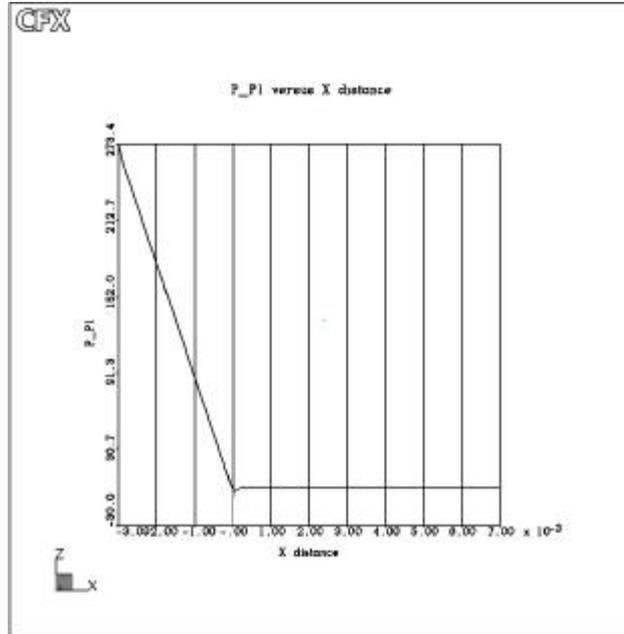


Figure 54 Liquid metal jet in a strong, transverse magnetic field. Velocity in the core in a uniform field at $y = 0$ for $E = -1$, $Ha = 200$.

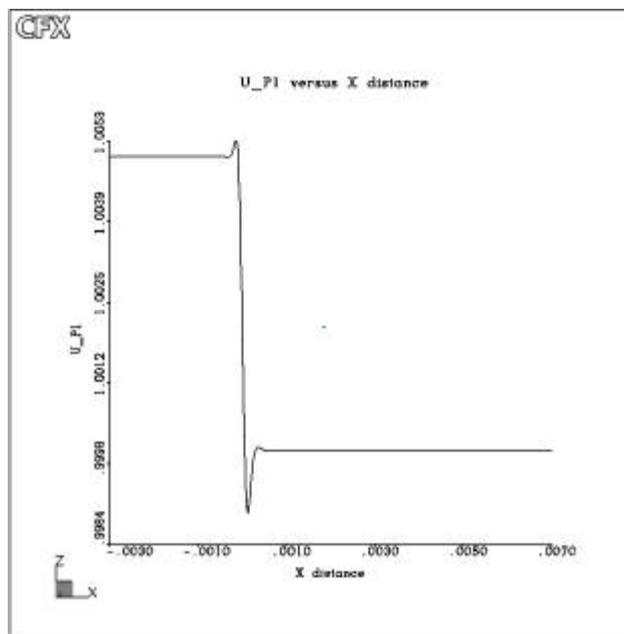


Figure 55 Liquid metal jet in a strong, transverse magnetic field. Variation of pressure in a uniform field for $E = -1$, $Ha = 200$, $N = 1$.

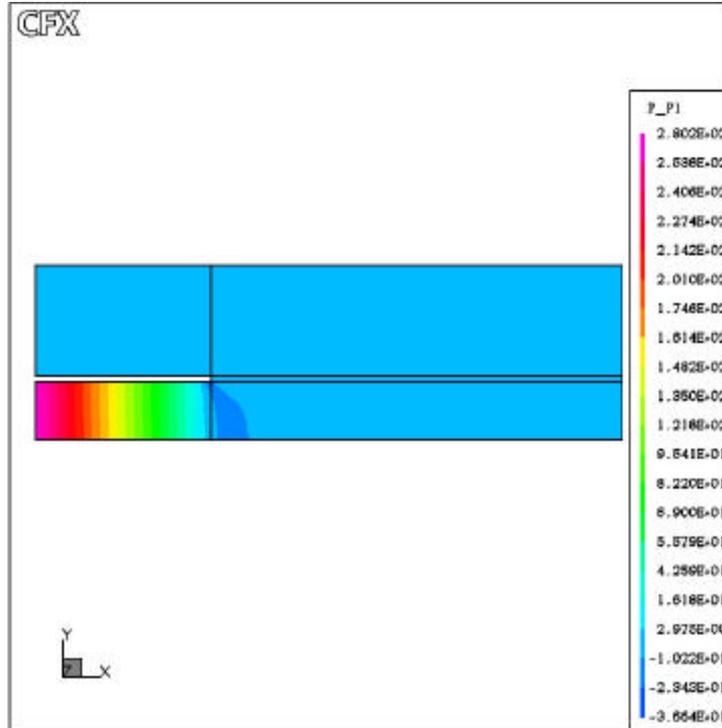


Figure 56 Liquid metal jet in a strong, transverse magnetic field. Variation of pressure in a uniform field at $y = 0$ for $E = -1$, $Ha = 200$, $N = 1$.

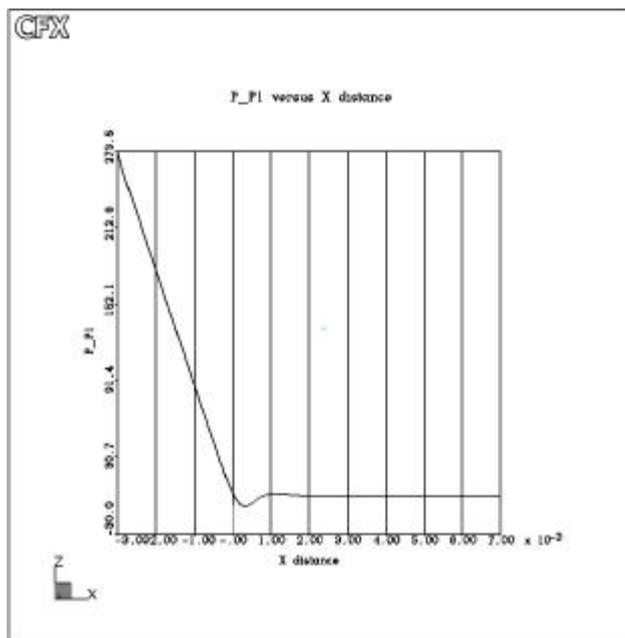


Figure 57 Liquid metal jet in a strong, transverse magnetic field. Velocity in the core in a uniform field at $y = 0$ for $E = -1$, $Ha = 200$, $N = 1$.

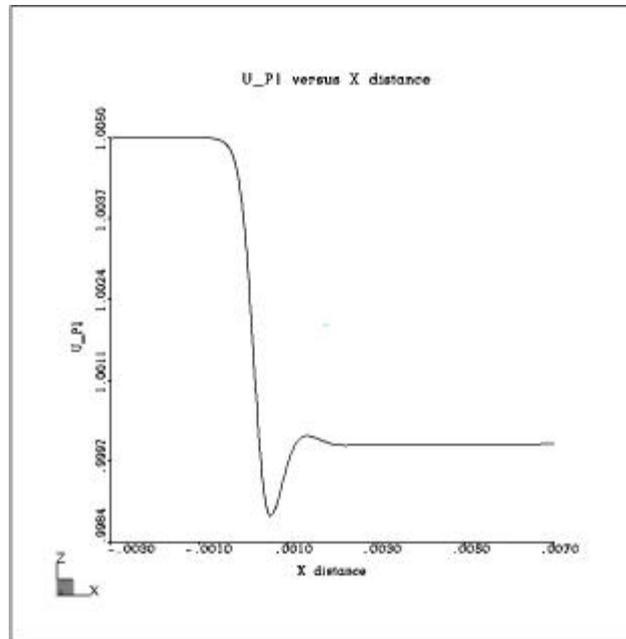


Figure **Liquid metal jet in a strong, transverse magnetic field. Variation of jet thickness for $E = -1$, $Ha = 200$, $B_\infty = 0.5$, $z = 2$. Colour map represents the numerical solution, the solid black line shows the asymptotic solution.**

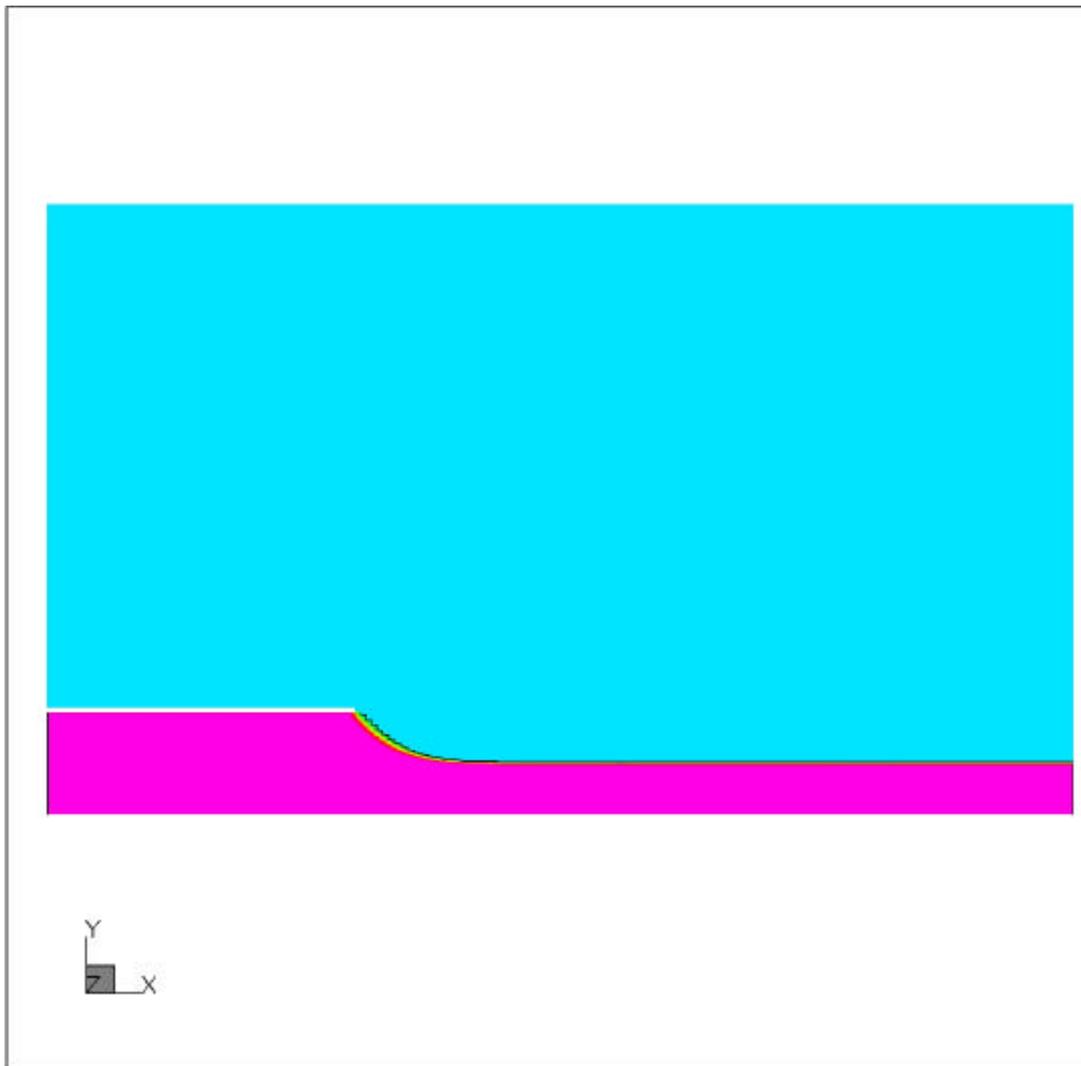


Figure 59 Liquid metal jet in a strong, transverse magnetic field. Variation of pressure for $E = -1$, $Ha = 200$, $B_\infty = 0.5$, $z = 2$.

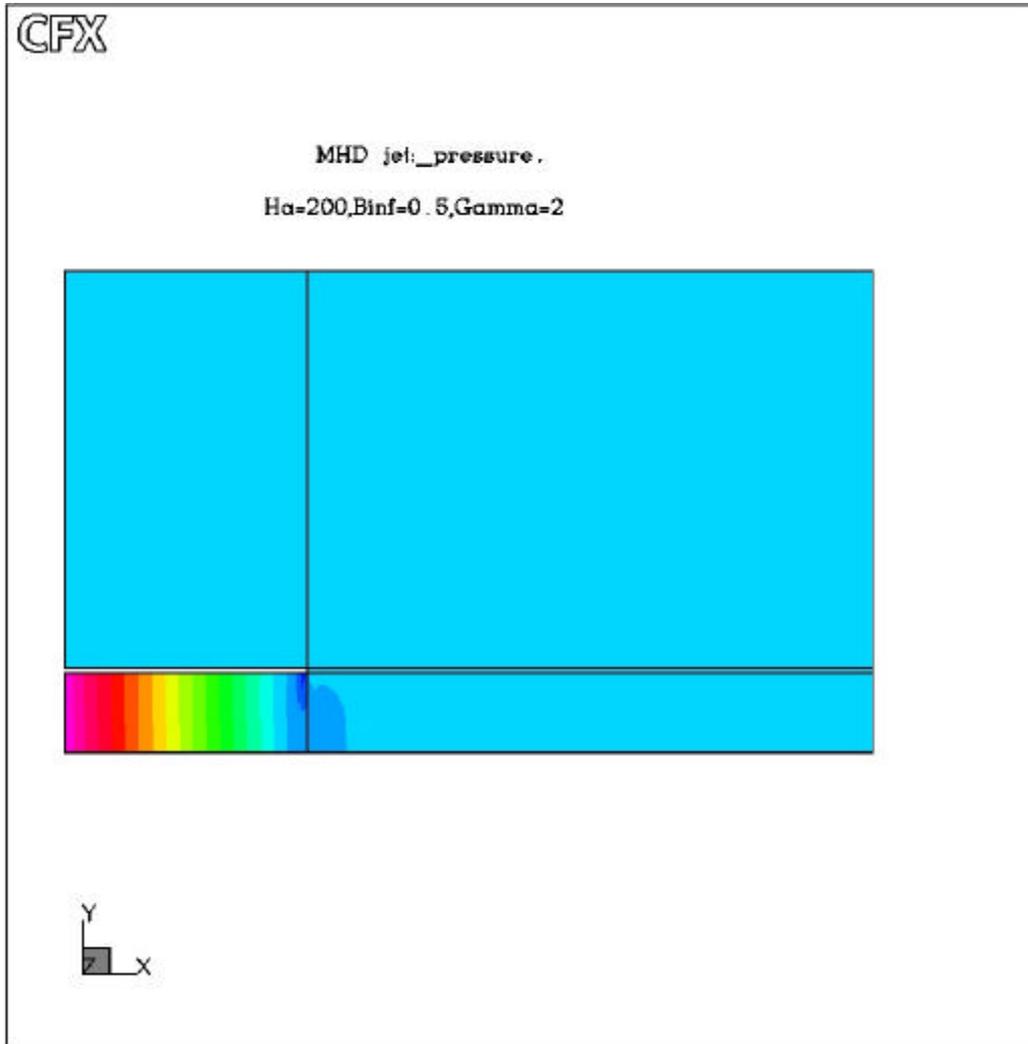


Figure 60 Liquid metal jet in a strong, transverse magnetic field. Core velocity in the jet ($y = 0$) for $E = -1$, $Ha = 200$, $B_\infty = 0.5$, $z = 2$. Solid line corresponds to the numerical solution, stars to the asymptotic solution.

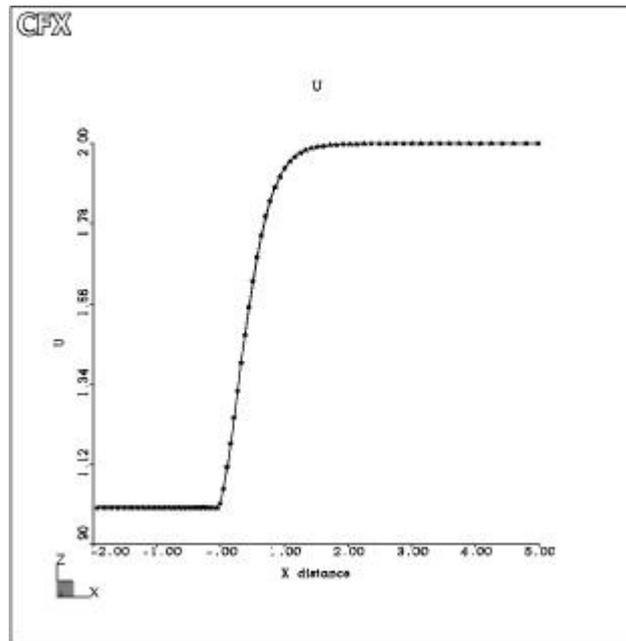


Figure 61 Liquid metal jet in a strong, transverse magnetic field. Streamlines in the jet for $E = -1$, $Ha = 200$, $B_\infty = 0.5$, $z = 2$.

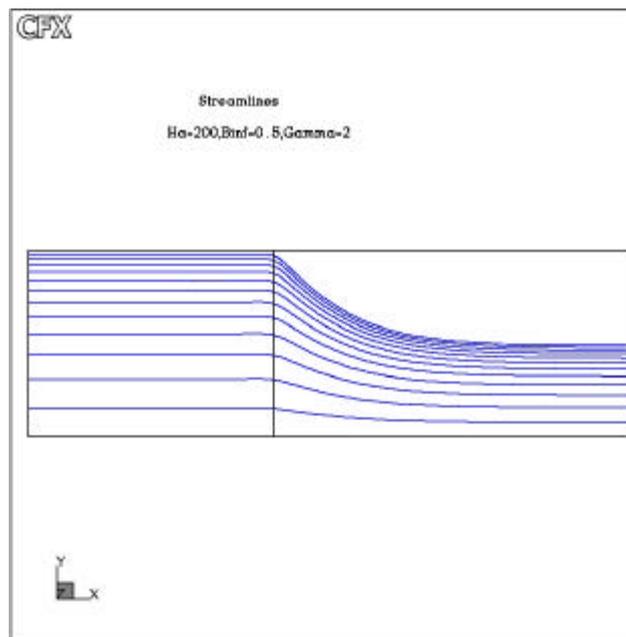


Figure 62 Flow in electrically coupled U-bends(from [36]).

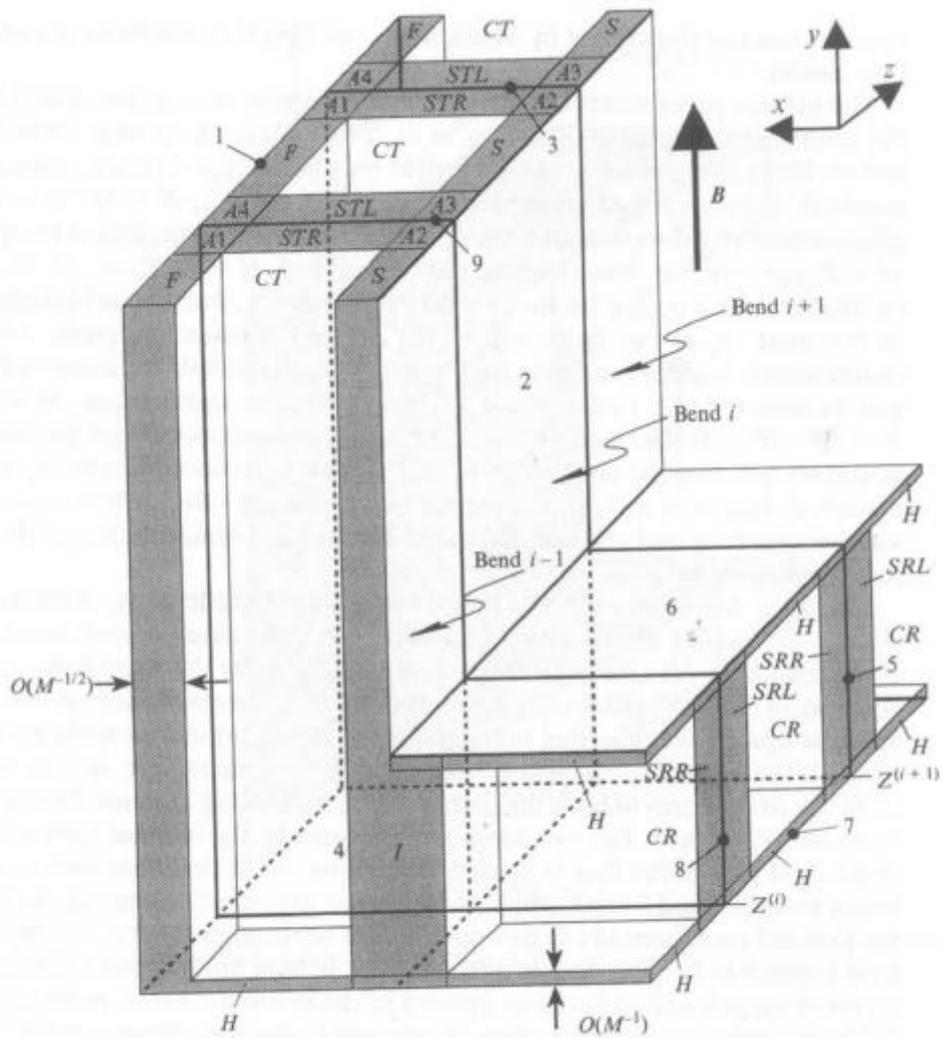


FIGURE 2. Flow subregions for $M \gg 1$ and wall numbers of bend i for $y < l$. The walls are: blanket first and second walls (1, 2); sidewalls of the toroidal (3, 9) and the radial (5, 8) ducts; top and bottom of the radial duct (6, 7); bottom of the toroidal duct (4). Flow subregions are: CR, CT – cores of the radial and toroidal ducts, H – the Hartmann layers, F, S, I, SRL, SRR, STL, STR – the parabolic layers, A1–A4 – the corner layers.

Figure 63 Flow in a circular insulating duct in a nonuniform magnetic field. Projection of lines of constant pressure onto the plane transverse to the field. The field is out of the plane of the figure; it varies between $x = -1$ and $x = 1$. Variable x is in the flow direction. Variable z is in the direction transverse to the magnetic field (duct axis is at $z = 0$). Hartmann and Roberts layers are not shown. Here $Ha = 7000$. (from [5]).

