Appendix A UIFlow2D Documentation

Overview

UIFlow2D is a computer code which simulates the flow of turbulent, isothermal or reacting flows in 2-D complex geor
UIFlow2D is based on a finite volume discretization of the time mean steady governing equations in body fitted coord
(Thompson, et. al., 1974). The discretization is based on the use of a colocated grid for velocities and pressure (Rodi,
1987). Use of a colocated grid facilitates solution of flow problems in reverse flow geometries (e.g. reverse flow annul
combustor) and avoids the problem of curvature terms encountered when solving for grid oriented velocities. UIFlow2
the unconditionally stable hybrid differencing scheme.

UIFlow2D incorporates a multi-grid solution technique to solve the coupled system of equations governing fluid flow.
UIFlow2D currently uses a FMG-FAS acceleration technique to enhance the convergence rate of the calculation. At p
the continuity and momentum equations are efficiently solved using a V cycle. Additional scalar equations are solved
single grid fashion on the finest grid (Vanka, et. al., 1989).

UIFlow2D incorporates the standard k-e turbulence model with wall functions in order to simulate turbulent flows (Lau
and Spalding, 1974). A turbulent diffusion flame model based on the assumptions of infinitely fast chemistry and equa
diffusivities is incorporated in UIFlow2D (Bilger, 1980). UIFlow2D is also capable of simulating turbulent premixed fla
The model for premixed flames is based on a one step description of kinetics (Westbrook and Dryer, 1981) with a
combination Arrhenius rate / eddy break up expression used for the time mean reaction rate (see Magnussen and Hje
1976 for EBU expression). UIFlow2D is currently capable of solving for only low Mach number flames which are adia
The current version of UIFlow2D does not incorporate models for radiation or liquid fuel sprays.

What Equations Does UIFlow2D Solve ?

UIFlow2D solves a discrete form of the governing equations in general curvilinear coordinates (x,h). A description of t
geometry in terms of body fitted coordinates facilitates the accurate solution of flows in complex geometries such as a
found in combustion chambers. The form of the general conservation equation in curvilinear coordinates is given belo
the case of a transformation from axisymmetric (z,r)

coordinates to general (x,h) coordinates. For additional information on coordinate transformations, see Fletcher, 1988

$$\f(\partial,\partial\xi)\,(\rho U\varphi) + \f(\partial,\partial\eta)\,(\rho V\varphi) = \f(\partial,\partial\xi)\,\b\lc\[\rc\](\Gamma\, q_{11}\,\f(\partial\varphi,\partial\xi)) + \f(\partial,\partial\eta)\,\b\lc\[\rc\](\Gamma\, q_{22}\,\f(\partial\varphi,\partial\eta)) + \f(\partial,\partial\xi)\,\b\lc\[\rc\](\Gamma\, q_{12}\,\f(\partial\varphi,\partial\eta)) + \f(\partial,\partial\eta)\,\b\lc\[\rc\](\Gamma\, q_{21}\,\f(\partial\varphi,\partial\xi)) + r\,|\,J\,|\,S^{\varphi}\,(\xi,\eta)$$

1.2.1)

$$U = r\,(\,ur_{\eta} - vz_{\eta}\,)$$

(1.2.2)

$$V = r\,(\,vz_{\xi} - ur_{\xi}\,)$$

(1.2.3)

$$q_{11} = \f(r,|\,J\,|)\,(\,r_{\eta}^{2} + z_{\eta}^{2}\,)$$

(1.2.4)

$$q_{22} = \f(r,|\,J\,|)\,(\,r_{\xi}^{2} + z_{\xi}^{2}\,)$$

(1.2.5)

$$q_{21} = q_{12} = -\,\f(r,|\,J\,|)\,(r_{\xi}r_{\eta} + z_{\xi}z_{\eta}\,)$$

(1.2.6)

$$|\,J\,| = (\,z_{\xi}r_{\eta} - z_{\eta}r_{\xi}\,)$$

(1.2.7)

The governing equations of fluid flow may be written in terms of a general conservation equation as given above (for r information, see Patankar, 1980). The source terms and diffusive exchange coefficients associated with any given var are given in Table 1, where S\s(f,TR) is the transformation source term and includes the cross derivative diffusion term

Table 1: Values of diffusive exchange coefficient and source term for variables used in simulation of turbulent reacting flow.

φ	Γ_{eff}	S^{φ}
1	0	0
\o(u,~)	$\mu+\mu_T$	S\s(u,TR) - (a_{11} \f(∂P,∂ξ) + a_{21} \f(∂P,∂η))
\o(v,~)	$\mu+\mu_T$	S\s(v,TR) - (a_{12} \f(∂P,∂ξ) + a_{22} \f(∂P,∂η)) + |J| (- 2Γ_{eff} \f(\o(v,~),r^2) + \f(\o(ρ,-)\o(w,~)^2,r))
\o(w,~)	$\mu+\mu_T$	S\s(w,TR) - \b\lc\[\rc\](\f(ρ\o(v,~)\o(w,~),r) - Γ_{eff}\f(\o(w,~),r^2) - \f(\o(w,~),r) \f(∂Γ_{eff},∂r)) | J |
k	μ + \f(μ_T,Pr $_k$)	(P_k - ρε) | J | + S\s(k,TR)
ε	μ + \f(μ_T,Pr $_\varepsilon$)	\f(ε,k) (C_1 P_k - C_2 ρε) | J | + S\s(ε,TR)
\o(f,~)	μ + \f(μ_T,Pr $_f$)	S\s(f,TR)
\o(g,~)	μ + \f(μ_T,Pr $_g$)	$C_{g1}\mu_T$ \b\lc\{\rc\}(\b\lc\[\rc\](\f(∂f,∂z))^2 + \b\lc\[\rc\](\f(∂f,∂r))^2) | J | - \f(C_{g2}ρgε,k) | J | + S\s(g,TR)
\o(Y,~)$_{FU}$	μ + \f(μ_T,Pr $_Y$)	\x\to(\o(ω,˙)) | J | + S\s(Y,TR)
\o(h,~)$_o$	μ + \f(μ_T,Pr $_{ho}$)	S\s(h_o,TR) - | J | \i\su(i=1,N,\x\to(\o(ω,˙))$_i$ h\s(o,i))

1.3. How Does UIFlow2D Solve The Flow Equations ?

UIFlow2D is based on a decoupled or segregated solution (Cope, 1991) of the governing flow equations. The discrete form of the u momentum equation is solved first, followed by the solution of the v momentum equation. The discrete continuity equation is written as a pressure correction equation (Patankar, 1980). The solution of the pressure correction equation is followed by an update of pressure, velocities, and mass fluxes.

The solution of all scalar quantities such as k, e, etc. is obtained next. The calculation of thermodynamic and transport properties completes one iteration. Due to the coupled nature of the equations to be solved, numerous sweeps through this sequence of equation solving must be performed. Typical single grid schemes can require thousands of iterations before a converged solution is obtained.

UIFlow2D uses a FMG-FAS multi-grid scheme to accelerate the convergence rate of the calculation (Brandt, 1977). I current version of UIFlow2D , the continuity and momentum equations are solved in a V cycle. The scalar equations a solved in a single grid manner on the finest grid. One iteration on the finest grid thus involves performing one sweep o cycle, followed by successive solutions of each scalar equation. A flowchart of the calculation procedure used by UIFl is given in Figure 1.

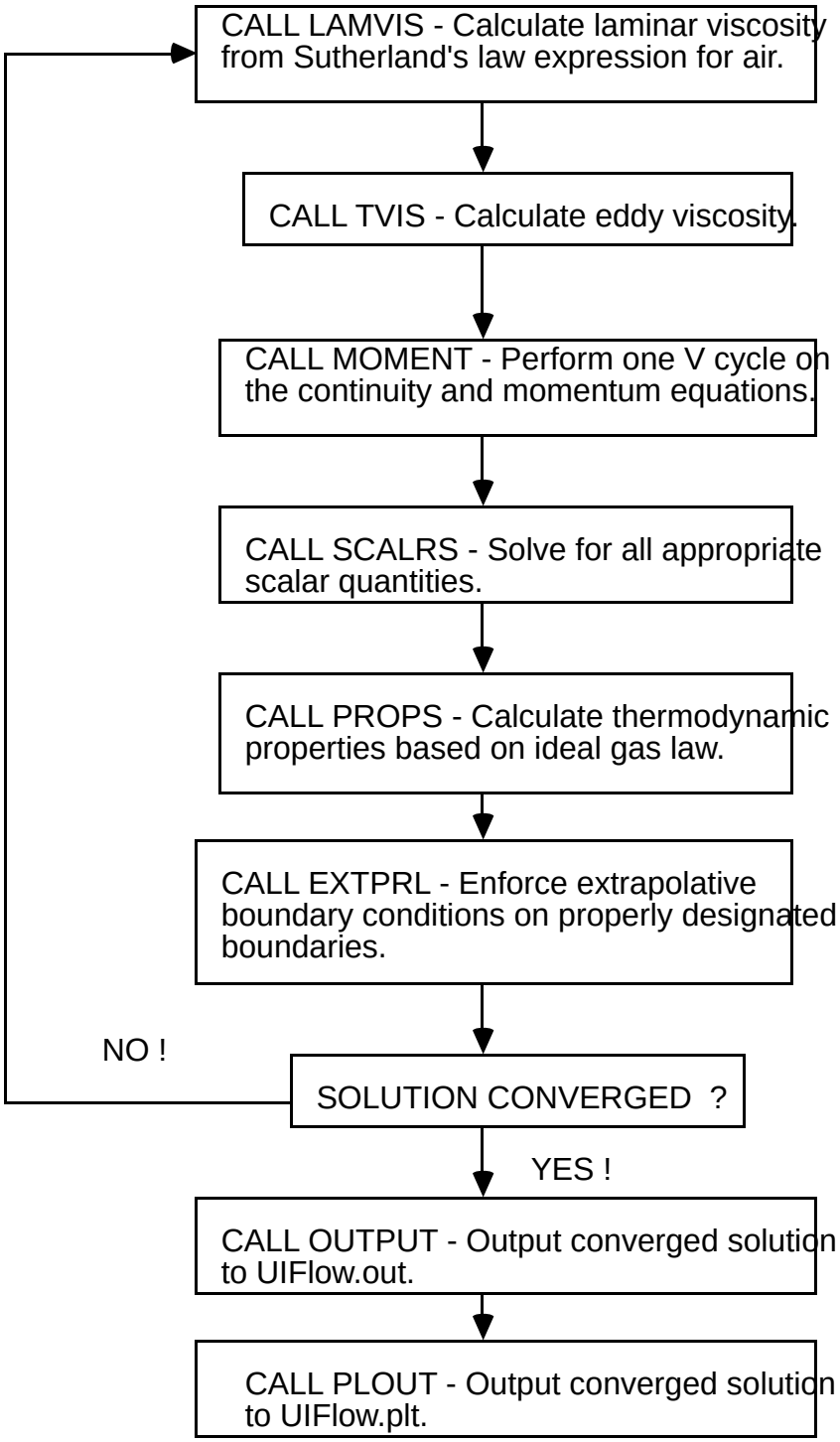

Figure 1: Flowchart of solution routine used by UIFlc

1.4. Using UIFlow2D

UIFlow2D requires five (5) files in order to compile and execute. UIFlow2D requires three (3) files for its execution:

1) UIFlow (Executable file, i.e. compiled version of code).

2) UIFlow.In (Input file which describes the flow problem).

3) UIFlow.Grid (Grid file).

Two (2) additional files are required in order to compile UIFlow2D :

1) UIFlow.com (Common block for program).

2) UIFlow.indx (Index file)

Prior to executing the code, the problem description must be given in UIFlow.In. A description of flow parameters found in UIFlow.In is given in the appendix along with some sample input files. An appropriate grid must also be generated and stored in UIFlow.Grid.

To facilitate easy use of UIFlow2D , a pre-processor has been developed by NCSA at the University of Illinois, Urbana Champaign. The pre-processor allows the user to create an input file and grid by simply making selections from vario menus and by drawing a grid. No file editing is therefore required. Section 2 of this manual gives a complete descripti the pre-processor.

1.5. What Output Does UIFlow2D Give?

UIFlow2D generates two (2) output files. UIFlow.out presents the converged values of all desired flow variables. It als a convergence history of the calculation and shows the flow parameters which were used in solving the flow problem. UIFlow.plt gives the converged values of u, v, and w in the format appropriate for use with Plot3d software developed NASA Ames. In addition, an HDF file containing raster images readable by NCSA Image is produced. Other flow varia can also be printed in this format.

1.6. Solving Convergence Problems in UIFlow2D

UIFlow2D incorporates several procedures which improve its rate of convergence. UIFlow2D is not foolproof howeve may at times exhibit oscillatory convergence or it may even diverge. The following advice should prove useful in resol convergence problems.

1. Check the input file (UIFlow.In). If the data in the input file is not self consistent, divergence will result. For example, if one specifies that there are two segments on the x-minus boundary and only one set of

data is given, then insufficient information is being supplied to the code. Also, the number of cells used in the x and y directions must be consistent with the number of cells found in the grid file (UIFlow.Grid). If the pre-processor is used to generate the input file, then problems caused by an improper input file will be minimized.

2. If the convergence history of UIFlow2D exhibits a diverging oscillatory nature (i.e. the residuals oscillate between high and low values but increase with iteration number), then the specified relaxation factors may be too high. Under NO circumstances should the relaxation factor exceed a value of 1.0. Divergence will be the result. Also, relaxation below 0.3 for all variables is an indication that the convergence problem does not involve underrelaxation.

   A relaxation of 0.6 on velocities, 0.4 on pressure, and 0.7 - 0.8 on k and e has been found to work well for most problems. For reacting flow problems, a density relaxation factor of 0.5 - 0.6 and a temperature relaxation of 0.6 - 0.7 are most likely appropriate. As a general rule, a lower relaxation factor is required for flows at higher Reynolds numbers and for grids with many cells.

3. If the solution diverges very early in the solution, then the problem may be the initially estimated flow parameters. This problem is most pronounced for reacting flows, and especially the premixed flame calculations. In general, the estimated flow variables should be as close as possible to the expected solution. In other words, if one is calculating a premixed flame, a guessed temperature field of 300 K is not a good choice. A temperature of approximately 1500 K is closer to the temperatures expected. In some respects, guessing the temperature is analogous to "igniting" the flow. A spark of only 300 K would not ignite a fuel-air mixture. A guessed temperature of 300 K will not result in reaction if an Arrhenius rate expression is used to describe the reaction rate.

   For turbulent flow calculations, the information given in Nallasamy, 1987 may be used to prescibe initial estimates of k and e. Use of a small length scale (0.03 R) leads to an estimate of e which may be somewhat large. This has been observed to be beneficial to the convergence rate in that the turbulent kinetic energy is "damped" early on in the calculation procedure.

4. The grid used in calculating the flow may also lead to convergence difficulties. If the prescribed outlet boundary is placed within a recirculation region, divergence will result. The outflow boundary condition specifies the flow as being "fully-developed" . If the flow is recirculating at the outflow plane, a fully developed flow does not exist, and problems in solving the flow equations will result. Also, regions of high gradient (i.e. shear layers, flame surfaces, etc.) should be sufficiently resolved. Resolution of recirculation regions is particularly important. Insufficient resolution of flow recirculation may result in the flow being predicted as recirculatory at the outflow plane.

5. Use of integral mass adjustments (kadj = 1) may impede the convergence rate for reacting flow calculations and for flows exhibiting significant recirculation (e.g. high swirl number flow). The calculation procedure is convergent without use of integral mass adjustments. Mass adjustments are only used to enhance the convergence rate. If they do not enhance convergence, or if it is suspected that they impede convergence, simply do not use integral mass adjustments.

6. If the grid used in the calculation is highly non-orthogonal (included angle between grid lines less than 30 degrees), then additional terms may be needed in the pressure correction equation. Use of knorth = 1 includes these terms. It has been found that for most internal flow geometries, these terms are not needed. If the additional terms are included, the convergence rate will be enhanced but more computational effort will be expended in evaluating these non-orthogonal terms. In general, a lower relaxation factor on pressure is required for highly non-orthogonal grids if the additional terms are NOT included.

7. Performing more sweeps through the ADI for any given flow variable should help the convergence. In general, more iterations are required on the pressure correction equation than on other equations (10 - 20 iterations should suffice).

8. Under NO circumstances is an outflow segment to be specified along with any other segment at a given boundary. For example, if the x-plus boundary is the boundary at which outflow occurs, then ALL of the boundary must be an outflow. A wall segment could not be specified at the x-plus boundary along with the outflow.

References

Bilger, R. W. (1980). "Turbulent Flows with Nonpremixed Reactants." in Turbulent Reacting Flows. eds. P. A. Libby and F. A. Williams. pp. 65-113. Springer-Verlag. Berlin.

Brandt, Achi. (1977). "Multi-Level Adaptive Solutions to Boundary-Value Problems." Mathematics of Computation. Vol. 31. No. 138. pp. 333-390.

Cope, Wm. Kevin. (1991). On the Numerical Simulation of Turbulent Flows in Complex Geometries. M.S. Thesis. University of Illinois, Urbana-Champaign. Dept. of Mechanical and Industrial Engineering.

Fletcher, C. A. J. (1988). Computational Techniques for Fluid Dynamics. Vol. 2. Springer-Verlag. Berlin.

Launder, B. E. and Spalding, D. B. (1974). "The Numerical Computation of Turbulent Flows." Computer Methods in Applied Mechanics and Engineering. Vol. 3. pp. 269-289.

Magnussen, B. F. and Hjertager, B. H. (1976). "On Mathematical Modelling of Turbulent Combustion with Special Emphasis on Soot Formation and Combustion." Sixteenth Symposium (International) on Combustion. pp. 719-729.

Nallasamy, M. (1987). "Turbulence Models and Their Applications to the Prediction of Internal Flows: A Review." Computers and Fluids. Vol. 15. No. 2. pp. 151-194.

Patankar, S. V. (1980). Numerical Heat Transfer and Fluid Flow. Hemisphere Publishing Company.

Rodi, W., Majumdar, S., and Schonung, B. (1987). "Finite Volume Methods for Two Dimensional Incompressible Flows with Complex Boundaries." Paper presented at the 8th International Conference on Computing Methods in Applied Sciences and Engineering. Versailles, France. December 14-17, 1987.

Thompson, Joe F., Thames, Frank C., and Mastin, C. Wayne (1974). "Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies." Journal of Computational Physics. Vol. 15. pp. 299-319.

Vanka, S. P., Krazinski, J. L., and Nejad, A. S. (1989). "Efficient Computational Tool for Ramjet Combustor Research." Journal of Propulsion and Power. Vol. 5. No. 4. pp. 431-437.

Westbrook, Charles K. and Dryer, Frederick L. (1981). "Simplified Reaction Mechanisms for the Oxidation of Hydrocarbon Fuels in Flames." Combustion Science and Technology. Vol. 27. pp. 31-43.

Glossary of Variables

UIFlow.In is an input file required by UIFlow2D for its execution. UIFlow.In provides a description of the problem to be solved. Information regarding boundary conditions, flow type (reacting or isothermal), relaxation factors, etc. is included in UIFlow.In. Given below is a complete description of the variables which are to be defined in UIFlow.In.

VARIABLE	DESCRIPTION
klam	**klam** specifies whether the flow is laminar or turbulent. Specify: klam = **1** if flow is **laminar** klam = 0 if flow is turbulent
kcomp	**kcomp** specifies whether the flow is incompressible or compressible (currently this option is inactive). Specify: kcomp = **1** if **compressible** kcomp = 0 if incompressible
kswrl	**kswrl** specifies a swirling or non-swirling flow. Specify: kswrl = **1** if **swirl** is involved kswrl = 0 if no swirl is involved
kpgrid	**kpgrid** specifies whether the grid being used was generated from the pre-processor. Specify: kpgrid = **1** if **pre-processor** grid kpgrid = 0 if other
model	**model** specifies the type of problem being solved. Specify: model = **0 isothermal** and **incompressible**. model = 1 compressible air flow. model = 2 premixed flame. model = 3 diffusion flame.
kfuel	**kfuel** specifies the type of fuel which is being burned. Specify: kfuel = **0** for **propane** kfuel = 1 for methane kfuel = 2 for "town gas"
knorth	**knorth** specifies whether the grid used is highly non-orthogonal. If **knorth** = 1, then additional terms are added to the pressure correction equation so as to account for the non-orthogonality. Specify: knorth = **1** for highly **non-orthogonal** grid. knorth = 0 for orthogonal grid.

kplax — **kplax** designates whether the flow geometry is planar symmetric or axisymmetric
Specify: kplax = **1** for **planar** symmetry
kplax = 0 for axisymmetric

kadj — **kadj** indicates whether integral mass adjustments are to be used.
Specify: kadj = **1** for **mass adjustments**
kadj = 0 for no adjustments

ngrid — **ngrid** specifies the number of grid levels to be used in the multi-grid solution
Specify: ngrid = 1, 2, 3, 4, or 5

ncelx — **ncelx** designates the total number of **cells** on the finest grid which are used in the x direction. The value given for ncelx **must** be consistent with the number of cells generated in the x direction by the grid generator.

ncely — **ncely** designates the total number of **cells** on the finest grid which are used in the y direction. The value given for ncely **must** be consistent with the number of cells generated in the y direction by the grid generator.

nsxm — **nsxm** designates the number of different boundary types found on the x - minus (left) boundary.
Specify: nsxm = Any integer from 1 to 5 depending on the problem.

kbxm — **kbxm** is an array which specifies the boundary type for any given segment. As such, a value of **kbxm** must be given for each segment.
Specify: kbxm = 1 for a wall
kbxm = 2 for an inlet
kbxm = 3 for a symmetry boundary
kbxm = 4 for a ZERO derivative outlet condition

jfxm, jlxm — **jfxm** and **jlxm** are the numbers of the first cell at a given boundary type and the last cell at a given boundary type, respectively. For example, consider that 20 cells span the x - minus boundary and two (2) segments are present on this boundary. If the first 10 cells correspond to an inlet, then jfxm = 1 and jlxm = 10 for the inlet segment and jfxm = 11, jlxm = 20 for the other segment.

ubxm — **ubxm** is the value of the u component of velocity at a particular x - minus segment.

vbxm	**vbxm** is the value of the v component of velocity at a particular x - minus segment.
wbxm	**wbxm** is the value of the w component of velocity at a particular x - minus segment.
vscxm	**vscxm** is the value of the molecular viscosity at a particular x - minus segment. This value does **not** need to be prescribed. UIFlow2D will prescribe a value based on the temperature.
txm	**txm** is the value of the static temperature at a particular x - minus segment.
rhxm	**rhxm** is the value of the fluid density at a particular x - minus segment.
fxm	**fxm** is the value of the mixture fraction at a particular x - minus segment.
gxm	**gxm** is the value of the concentration fluctuation at a particular x - minus segment.
tkxm	**tkxm** is the value of the turbulent kinetic energy at a particular x - minus segment.
tdxm	**tdxm** is the value of the turbulent dissipation rate (e) at a particular x - minus segment.
fuxm	**fuxm** is the value of the fuel mass fraction at a particular x - minus segment.
co2xm	**co2xm** is the value of the carbon dioxide mass fraction at a particular x - minus segment.
h2oxm	**h2oxm** is the value of the water mass fraction at a particular x - minus segment.
o2xm	**o2xm** is the value of the oxygen mass fraction at a particular x - minus segment.
wmxm	**wmxm** is the value of the mixture molecular weight at a particular x - minus segment.
nsxp, kbxp, jfc, jlc, ubxp, vbxp, wbxp, vscxp, txp, rhxp, fxp, gxp, tkxp,tdxp, fuxp, co2xp,h2oxp, o2xp, wmxp	Values at x - plus (right) boundary segment.
nsym, kbym, ifc, ilc, ubym, vbym, wbym, vscym, tym, rhym, fym, gym, tkym, tdym, fuym, co2ym, h2oym, o2ym, wmym	Values at y - minus (bottom) boundary segment

nsyp, kbyp, ifc, ilc, ubyp, vbyp, wbyp, vscyp, typ, rhyp, fyp, gyp, tkyp, tdyp, fuyp, co2yp, h2oyp, o2yp, wmyp	Values at y - plus (top) boundary segment
ugs	Initial estimate of the u component of velocity.
vgs	Initial estimate of the v component of velocity.
wgs	Initial estimate of the w component of velocity.
rhgs	Initial estimate of the fluid density.
tgs	Initial estimate of the fluid temperature.
tkgs	Initial estimate of the turbulent kinetic energy (k).
tdgs	Initial estimate of the turbulent dissipation rate (e).
fgs	Initial estimate of the mixture fraction (f).
ggs	Initial estimate of the turbulent concentration fluctuation (g).
fugs	Initial estimate of the fuel mass fraction.
tfuel	The temperature of the inlet fuel stream. This parameter is important for simulation of adiabatic diffusion flames.
tair	The temperature of the inlet air stream. This parameter is important for simulation of adiabatic diffusion flames.

For the variables **prl, prt, relx, nswp**, and **iprint**, different values exist for different variables. Each variable has been assigned a number according to the following list:

Variable	Number of variable (nv)
u	1
v	2
P	3
w	4
h,T	5
k	6
e	7
f	8
fu	9
g	10
r	11

prl — **prl** is the laminar Prandtl number array. Eleven (11) values of this parameter are to be specified. One value for each dependent variable solved by UIFlow2D.

prt — **prt** is the turbulent Prandtl number array. Eleven (11) values of this parameter are to be specified. One value for each dependent variable solved by UIFlow2D.

relx — **relx** is the relaxation parameter array. Eleven (11) values of this parameter are to be specified. One value for each dependent variable solved by UIFlow2D.

nswp — **nswp** is the variable which specifies the number of ADI sweeps to perform on any given variable. Eleven (11) values of this parameter are to be specified. One value for each dependent variable solved by UIFlow2D.

iprint — **iprint** is the array which designates whether a given variable is to be printed. Twelve values of this parameter must be supplied. The twelfth value corresponds to mass fractions relevant to the premixed flame simulation. If iprint(12) = 1, then these mass fractions will be printed.
Specify: iprint = **1** for **printing** the variable
iprint = 0 for no printing

maxitn — **maxitn** specifies the maximum number of allowable iterations which will be performed before program execution is terminated.

rin — **rin** is the offset radius specified for axisymmetric problems. **rin** is the distance from the y-minus boundary to the actual symmetry axis.

pref — **pref** is the reference pressure for the system. For low mach number flows, this pressure determines the thermodynamic state of the fluid.

vscty — **vscty** is the molecular viscosity of the fluid. For isothermal flows, this value is **not** calculated in the program. For variable density flows, subroutined lamvis will calculate this quantity as a function of temperature.

tolr **tolr** is the prescribed tolerance for declaring a converged solution. A value of $1x10^{-4}$ requires that the mass residual on the finest grid be reduced by four orders of magnitude before convergence is obtained.