Examensarbete Nr 95/6

CHALMERS TEKNISKA HÖGSKOLA
Institutionen för Thermo- och Fluiddynamik

CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Thermo- and Fluid Dynamics

COMPUTATIONAL MODELING OF 2D Hill FLOWS

by

Jérôme RENARD and Dominique GRESSER

Examensarbete i Termo- och Fluiddynamik

Göteborg, Juni 1995

Abstract

The present work has been focused on computational method involved in the modeling of a 2D turbulent flow. The flow configuration consists of a channel with a hill.

The dependency of the results upon three factors have been studied. Three differents discretization scheme (Hybrid, Van Leer and QUICK) have been used, as well as six meshes presenting different levels of refinement.

Six different turbulence models have been considered, they are: The standard $k-\varepsilon$ model, a Two layer $k-\varepsilon$ model, a Low-Re $k-\varepsilon$ model, the $k-\omega$ model, a Low-Re $k-\omega$ model, and finally a Two layer second-order moment closure model.

A fourth factor that has no direct influence on the results but determines the computations time and therefore their costs, is the solving procedure used to compute the linearized equations. New general purpose solvers have been implemented, and applied either to solve a single variable, or two interdependent variables.

Acknowledgments

This work has been carried out during our one-year stay at Chalmers University of Technology. During this year, we have spent 50 percent of our time following courses, and the rest of the time we have been working on the project work presented in this report.

This year, partly financed by ERAMUS and Conseil Régional de Lorraine stands for our fourth year of studies out of five, at ESSTIN (Ecole Supérieure des Sciences et Technologies de l'ingénieur de Nancy).

We first would like to thank our supervisor Dr Lars Davidson for his help and guidance throughout this work.

Special thanks to Sven Perzon for his unvaluable contribution, support and patience under our flow of questions.

Our regards go to our "room mate" Gang Zhou, who helped us a lot with his wise advices and his knowledge.

We would like to express our gratitude to all members of the Department of Thermo- and Fluid Dynamics: they really created a friendly atmosphere which contributed greatly to our work and our happiness in Sweden.

Finally, we are particulary grateful to the "french team" Pierre-Yves, Jean-Marc, Cyril, Gerard, Jean, 'Godasse' and Bruno for their support.

Introduction

The present work is a part of an on-going NUTEK project at the Department of Thermo and Fluid Dynamics, in cooperation with Volvo Data, on turbulence modeling of separated flow along curved surfaces. It has also been presented at the *ERCOFTAC Workshop on Data Bases and Testing of Calculation Methods for Turbulent Flows*, April 3-7, 1995, University of Karlsruhe, Karlsruhe, Germany (test case 2a, 2D Hill Flow).

In a constant wish to improve the turbulence modeling of complex flow situations, different workshops have been set up to study test-cases, geometrically simple configurations allowing complex flow behaviors.

The present test-case, a channel flow with separation, involves streamlines curvatures. Turbulence models based on the eddy-viscosity concept usually predict inaccurrately or fail to predict such situation.

The rapid increase of the computers capacities, along with the constant decrease of their cost, led to an important developement of the computational modeling procedures: more equations are solved, on larger computation domains...Some powerful numerical methods have been developed to solve the flow quantities equations. This includes new solvers, but also new concepts such as the coupled resolution of strongly inter-dependent variables.

Contents

List of Figures

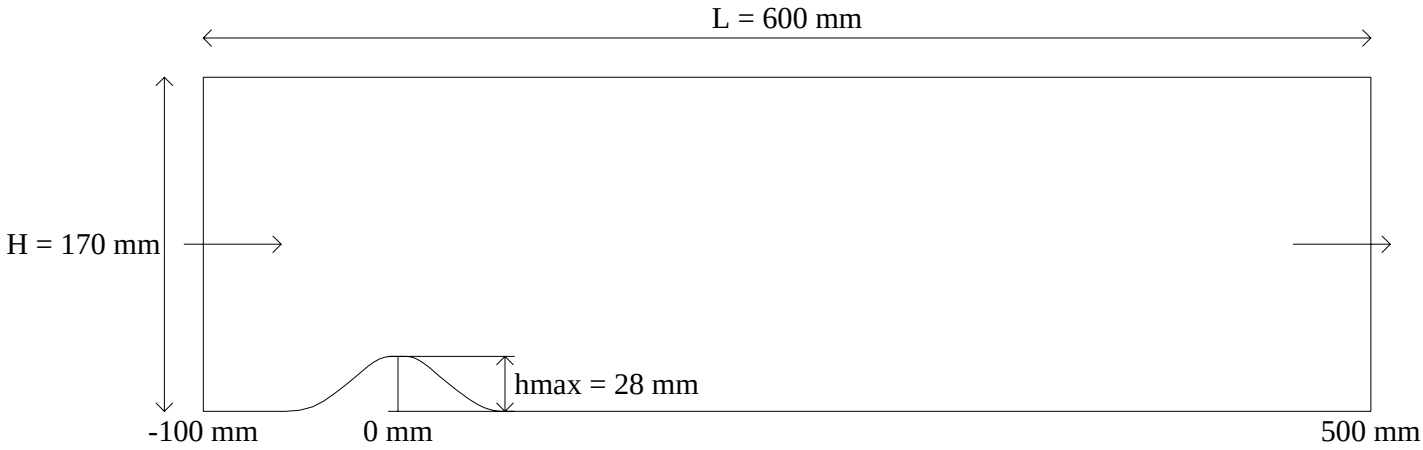


Figure 1: Configuration of the hill

1 Flow description

The flow configuration consists of a channel with a hill, located on the bottom of the channel.

geometry
The channel height is $H = 170$ mm.
The hill height is $h_{max} = 28$ mm.
The channel length should be, at least, 600 mm.
The top of the hill is placed 100 mm after the inlet.
The hill profile, a quadratic function based shape, is taken from experimental data.

flow parameters
The fluid is water.
Kinematic viscosity: $\nu = 1.0 \cdot 10^{-6}$ m^2/s.
Density: $\rho = 1000$ kg/m^3.
Centreline mean velocity at inlet: $U_0 = 2.147$ m/s.

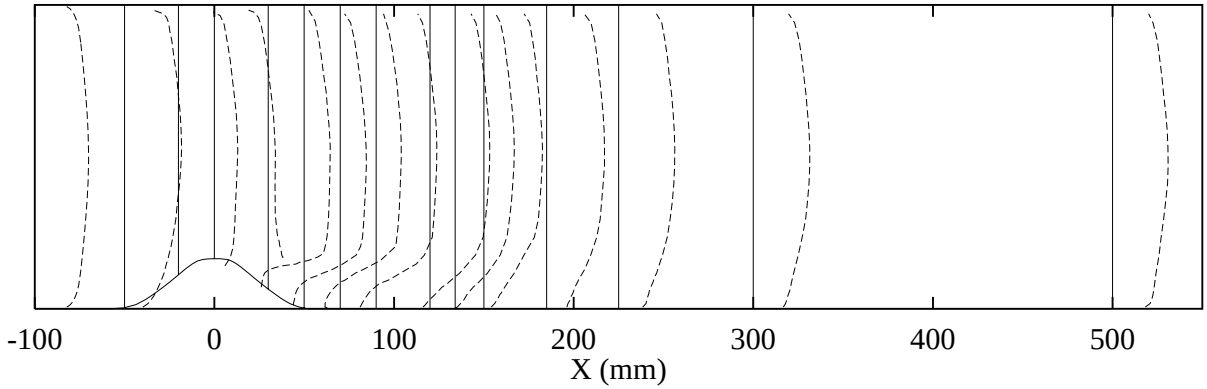
$$\text{Reynolds number: } R_e = \frac{U_0 h_{max}}{\nu} \approx 60\,000.$$

1.1 Experimental results

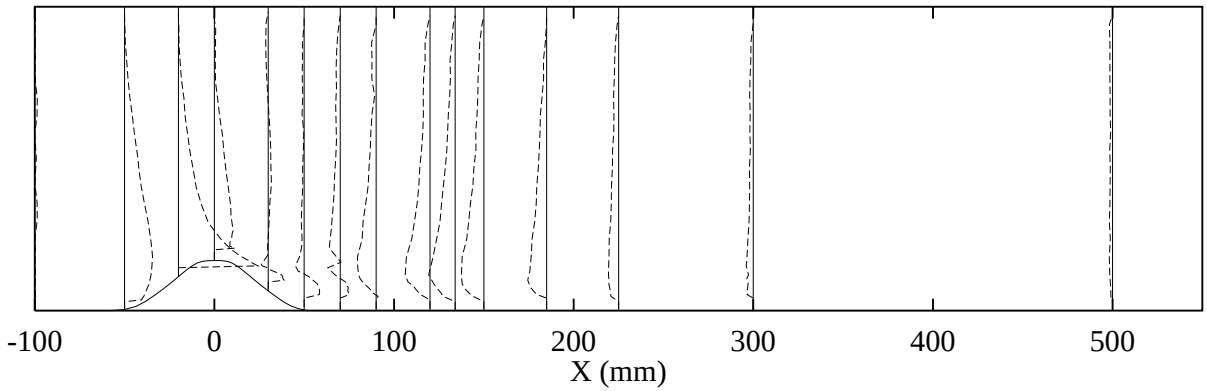
The experimental results (mean and fluctuating velocities) are available at 14 locations including the inlet. The measurements were carried out by Almeida *et al.* [1] using a laser-Doppler velocimeter.

The different plots of figure 2 show the experimental results. Only U, V and k profiles are plotted here, but $\overline{u^2}$, $\overline{v^2}$ and $\overline{uv}$ are also available.

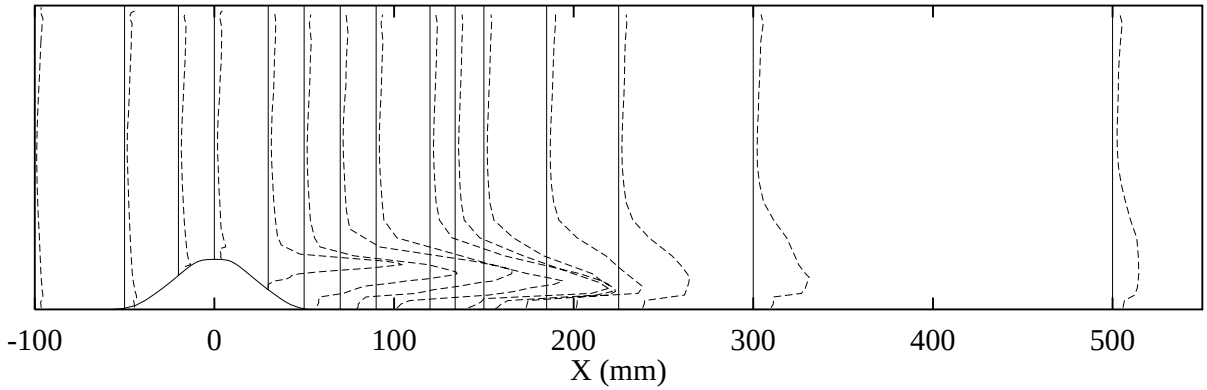
After the top of the hill the flow separates from the wall. A recirculation zone occurs from $x/h_{max} = 0.43$ (≈ 12 mm), and extends up to $x/h_{max} = 4.82$ (≈ 135 mm).



U velocity profile



V velocity profile



Turbulent kinetic energy profile

Figure 2: Experimental Results

2 Numerical methodology

The code used for this work is CALC-BFC [2], it has been developed at the department for research applications.

2.1 CALC-BFC

CALC-BFC (Boundary Fitted Coordinate) is a general three-dimensional finite volume code for three-dimensional geometries. It uses Cartesian velocity components, and collocated variables stored in the middle of the control volumes. Linear interpolation is used when a variable is needed at a face of a control volume.

The convective terms can be discretized using different schemes (see section 2.2 below). The diffusive terms are discretized using central differences (second-order accuracy). A pressure correction equation is obtained using the SIMPLEC algorithm, on a non-staggered grid arrangement.

CALC-BFC solves the flow quantities separately, in the following order:

- mean velocities U, V, W.
- pressure correction equation.
- turbulent quantities (k, ε, Reynolds stresses).
- other quantities (Temperature, concentrations ...)

Different solvers are available: TDMA (Tri-Diagonal Matrix Algorithm), SIP (Stone's strongly Implicit Procedure) and all the solvers available in the SLAP routines (see section 2.6).

Currently, U, V, k, ε, and the Reynolds stresses are computed using TDMA, while the pressure correction equation is solved using SIP or conjugated gradients algorithm.

A coupled resolution of k- and ε-equation, using different general purpose solvers implemented in SLAP, has been developed during this thesis work (see section 2.6.3).

2.2 Discretization schemes

Three different schemes available in CALC-BFC have been used to discretize the convective terms.

Hybrid by Spalding (1972).
This first order accurate (marginally second order) bounded scheme is mainly used to obtain rough results, that provides a good set of initial values to restart the computation with much slower high order schemes.

The Hybrid scheme is a combination of first order upwind and central differencing schemes. For the east face of a control volume, it is expressed as:

$$\Phi_e = \begin{cases} f_x\Phi_E + (1-f_x)\Phi_P & \text{if } |Re_{\delta x}| \le 2 \\ \left\{ \begin{array}{ll} \Phi_P & \text{if } U_e \ge 0 \\ \Phi_E & \text{if } U_e < 0 \end{array} \right\} & \text{if } |Re_{\delta x}| > 2 \end{cases}$$

Where f_x is a weighting function proportional to distances.

The Hybrid scheme has been used to discretize the k- and ε-equation where the accuracy of the scheme does not affect so much the computation (RSM also uses Hybrid to discretize the convective terms of the stresses equations).

Van-Leer (1974) is basically a first-order upwind scheme corrected when Φ-gradient changes are relatively small. The discretization error is therefore of second-order accuracy. This scheme is bounded.

It is defined as

$$\Phi_e = \begin{cases} \text{if } U_e \ge 0 & \begin{cases} \Phi_P & \text{if } |\Phi_E - 2\Phi_P + \Phi_W| \ge |\Phi_E - \Phi_W| \\ \Phi_P + \dfrac{(\Phi_E - \Phi_P)(\Phi_P - \Phi_W)}{\Phi_E - \Phi_W} & \text{otherwise} \end{cases} \\ \text{if } U_e < 0 & \begin{cases} \Phi_E & \text{if } |\Phi_P - 2\Phi_E + \Phi_{EE}| \ge |\Phi_P - \Phi_{EE}| \\ \Phi_E + \dfrac{(\Phi_P - \Phi_E)(\Phi_E - \Phi_P)}{\Phi_{EE} - \Phi_P} & \text{otherwise} \end{cases} \end{cases}$$

The Van-Leer scheme has also been applied on k- and ε-equations instead of the Hybrid scheme, slowing down the computation, without significant improvement in the result (see section 3.2).

QUICK by Leonard (1979) is a third-order accurate scheme.
It is based on a quadratic polynomial interpolation, using two nodes upstream, and one node downstream. For a uniform grid, this gives:

$$\Phi_e = \begin{cases} -\frac{1}{8}\Phi_W + \frac{3}{4}\Phi_P + \frac{3}{8}\Phi_E & \text{if } U_e \ge 0 \\ \frac{3}{8}\Phi_P + \frac{3}{4}\Phi_E - \frac{1}{8}\Phi_{EE} & \text{if } U_e < 0 \end{cases}$$

QUICK is an unbounded scheme, and thus it cannot be used for k- and ε-equations.

2.3 Meshes

Six different grids have been used (see figure 3), all created by Sven Perzon, using the ICEM-CFD mesh generator.

- High Re model meshes: Four meshes (74×25, 79×49, 122×50, and 148×50 interior cells) are to be used with models based on wall functions.

  To be valid, the wall functions, and therefore the nodes closest to the wall have to be located in the logarithmic region, $30 \leq y^+ \leq 100$. Figure 4 shows that y^+ lies in this interval only for two of the four meshes (namely 78×49 and 122×50). One can already expresses doubts concerning the accuracy of the results that can be obtained using the two remaining meshes.

- Low-Re model meshes: Two meshes (64×50, and 128×100 cells) are designed for the low-Re models. The grids are highly refined close to the walls, since the viscous sub-layer has to be solved. It is important to have several nodes in this layer, to be able to catch the high gradients existing there. It usually requires a first node around $y^+ = 1$.

  As shown in figure 5, the condition is respected for the north wall. On the south wall, a peak in y^+ is encountered, around $x = 0$, at the top of the hill where high velocity gradients exist. This deficiency is corrected on the finest mesh, where the number of cells in the y-direction is doubled.

The different refinement levels available in each "family" allow to choose between the accuracy of the results, and the computation cost. Fast computations are desirable during the development phase, while the accuracy is – of course – required for the final results.

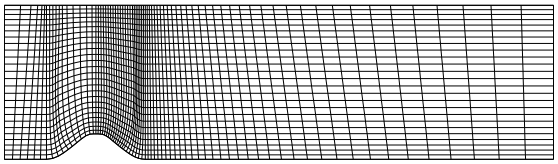
2.4 Boundary conditions

Four different types of boundary condition have been applied in this case, namely: inlet, outlet, symmetry and wall boundary conditions, as shown in figure 6.

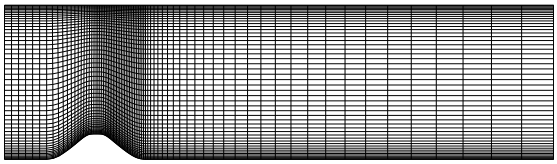
Inlet

- U, V are interpolated from the experimental data.
- k is computed using the interpolated values of the stresses $k = \frac{1}{2}\left(\overline{u^2} + 2\overline{v^2}\right)$. $\overline{w^2}$ is not available, therefore it is approximated by $\overline{v^2}$.
- ε is taken as $\varepsilon = \max\left(-\overline{uv}\dfrac{\partial U}{\partial y}, \dfrac{k^{3/2}{C_\mu}^{3/4}}{\ell_m}\right)$ with $\ell_m = \min(\kappa y, \lambda\delta)$. This assumption has been checked through a separated computation of a linear channel flow, where all the quantities were fixed but ε, which was solved (see figure 7).
- The Reynolds stress model requires inlet values for the extra variables ($\overline{u^2}$, $\overline{v^2}$, $\overline{uv}$). They are set according to the experiments using linear interpolation.

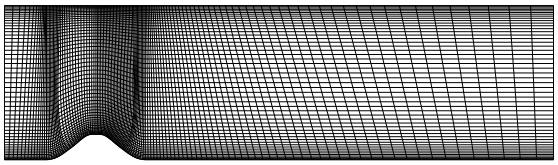
Wall function model meshes.



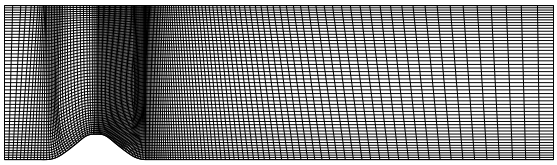
74 × 25 cells



78 × 49 cells

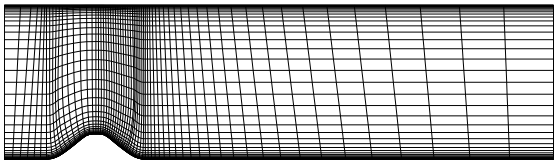


122 × 50 cells

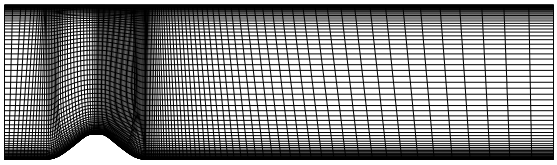


148 × 50 cells

Low-Reynolds number model meshes.



64 × 50 cells



128 × 100 cells

Figure 3: The different mesh grids used.

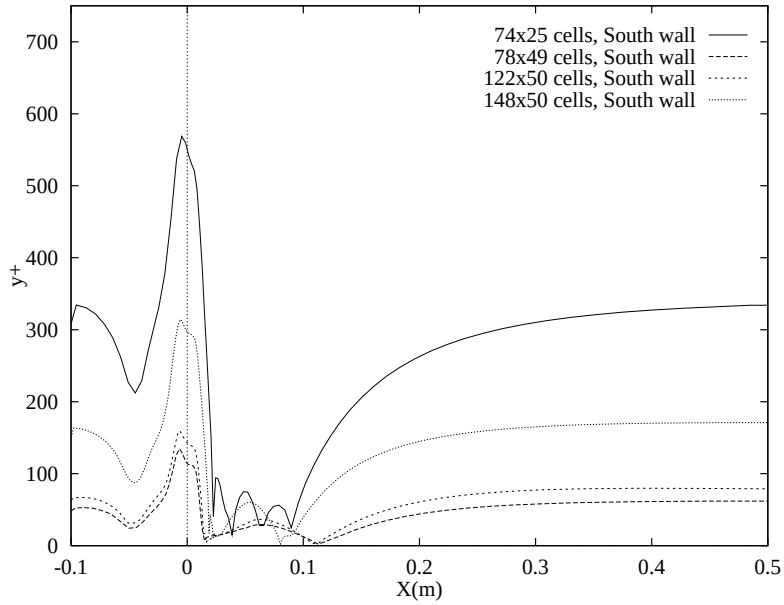


Figure 4: y^+ for two of the High-Reynolds number model meshes.

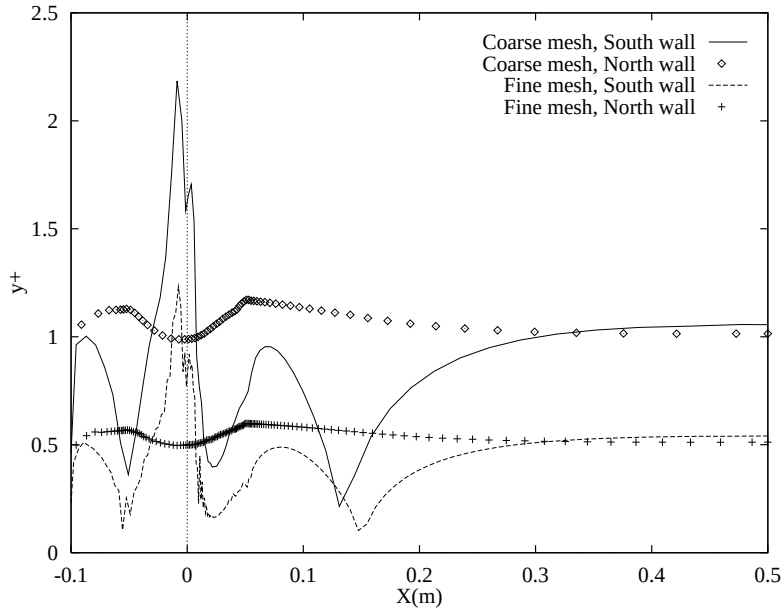


Figure 5: y^+ for the two Low-Reynolds number model meshes.

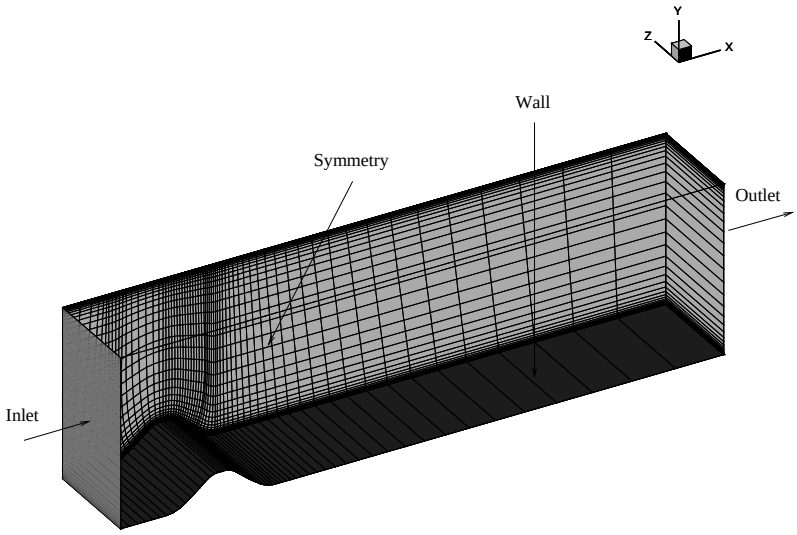


Figure 6: Sketch locating the different boundary conditions (the upper wall and front symmetry are removed to enhance readability).

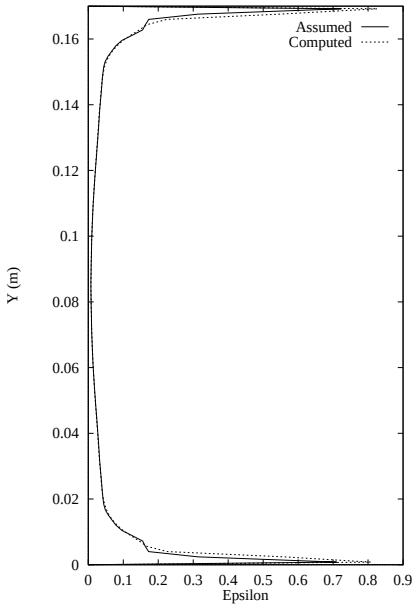


Figure 7: Comparison between ε-assumed, and ε-solved over a straight channel.

Outlet Zero exit gradient is assumed for all variables ($\Phi = U$, V, p, k, ε (or ω), and the Reynolds stresses).

$$\frac{\partial \Phi}{\partial x} = 0$$

Symmetry boundary CALC-BFC is a 3D code used here to study a 2D case, therefore symmetry boundary conditions have to be applied to the low and high faces of each cell. They are:

- The normal velocity W is set to 0.
- The remaining variables are set to satisfy $\dfrac{\partial \Phi}{\partial n} = 0$.

Wall boundary Different conditions have been applied to the wall, according to each model specifications. It can be "wall functions" computed on the node closest to the wall, or a set of equations applied to a near-wall region. Each case will be discussed along with the corresponding model.

2.5 Turbulence models

The turbulence models are based on the Reynolds decomposition; an instantaneous quantity can be decomposed into an averaged, and a fluctuating part.

$$\tilde{\Phi} = \Phi + \phi$$

The instantaneous velocities can be replaced by their mean and fluctuating parts, into the instantaneous momentum equations. By integrating this equation in time, one can obtain the time-averaged momentum equation.

$$\frac{\partial(\rho U_i U_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\right) - \frac{\partial\left(\rho\overline{u_i u_j}\right)}{\partial x_j}$$

Six (only three in 2D) new unknowns, $-\rho\overline{u_i u_j}$, also called the Reynolds stresses, arised from this last equation.
An exact equation for the turbulent kinetic energy $k = \frac{1}{2}(\overline{u^2} + \overline{v^2} + \overline{w^2})$ is derived by subtracting the time-averaged momentum equation from the instantaneous momentum equation and multiply the result by $u_i/2$. This leads to:

$$\underbrace{\frac{\partial(\rho U_j k)}{\partial x_j}}_{C} = \underbrace{-\frac{\partial}{\partial x_j}\left(\overline{u_j p} + \rho\overline{u_j k}\right)}_{D} \underbrace{-\rho\overline{u_i u_j}\frac{U_i}{\partial x_j}}_{P} \underbrace{-\mu\overline{\frac{\partial u_i}{\partial x_j}\frac{\partial u_i}{\partial x_j}}}_{\varepsilon}$$

Again, some new unknowns such as pressure-diffusion ($\overline{u_j p}$), triple correlations ($\overline{u_j k}$) and dissipation ε appeared.
The new set of equations is no longer closed, and thus some assumptions have to be introduced to model, or approximate their behavior, using only known variables.

2.5.1 $k-\varepsilon$ model

The standard $k-\varepsilon$ model is widely used to predict the influence of turbulence on the mean flow. In this two equations model a k-equation, modeled from the exact k-equation, simulates the Reynolds stresses using the eddy-viscosity concept developed by Boussinesq.

$$-\rho\overline{u_i u_j} = \mu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) - \frac{2}{3}\delta_{ij}\rho k$$

By dimensional analysis, μ_t can be expressed as a function of k and ε

$$\mu_t = C_\mu \rho \frac{k^2}{\varepsilon}$$

The dissipation rate of turbulent kinetic energy ε, is also solved using a transport equation derived by analogy with the k-equation.
The model itself is defined by the following set of equations

$$\frac{\partial}{\partial x_j}(\rho U_j k) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right] + P_k - \rho\varepsilon \tag{1}$$

$$\frac{\partial}{\partial x_j}(\rho U_j \varepsilon) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon}{\partial x_j} \right] + \frac{\varepsilon}{k}(C_{\varepsilon 1} P_k - C_{\varepsilon 2}\rho\varepsilon) \tag{2}$$

Where the generation term P_k is

$$P_k = \mu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) \frac{\partial U_i}{\partial x_j} \tag{3}$$

The different constants are

$$C_\mu = 0.09 \qquad C_{\varepsilon 1} = 1.44 \qquad C_{\varepsilon 2} = 1.92 \qquad \sigma_k = 1 \qquad \sigma_\varepsilon = 1.3 \tag{4}$$

Wall boundary This model uses wall functions to simulate the behavior of the flow close to the wall.
The wall shear stress is obtained by calculating the viscosity at the nodes adjacent to the wall from the log-law. The viscosity used in momentun equations is prescribed at the nodes adjacent to the wall (index P) as follows:

$$\tau_w = \mu_t \frac{\partial U}{\partial n} \approx \mu_t \frac{U_P}{n}$$

$$\tau_w = \rho U_\star^2$$

Equating these two equations leads to $\mu_t \dfrac{U_P}{n} = \rho U_\star^2$, and therefore $\mu_t = \dfrac{U_\star}{U_P}\rho U_\star n$ The law of the wall reads

$$\frac{U_P}{U_\star} = \frac{1}{\kappa}\ln(E n^+)$$

- The viscosity is set as $\mu_t = \dfrac{\rho U_\star n \kappa}{\ln(En^+)}$
- The turbulent kinetic energy is set as $k = {C_\mu}^{-1/2} {U_\star}^2$
- The energy dissipation rate is $\varepsilon = \dfrac{U_\star^3}{\kappa n}$

Where $U_\star$ is computed iteratively by the relation $\dfrac{U_P}{U_\star} = \dfrac{1}{\kappa}\ln(En^+) = \dfrac{1}{\kappa}\ln\left(E\dfrac{nU_\star}{\nu}\right)$

2.5.2 Two-layer $k-\varepsilon$ model

A two-layer model separates the flow into two parts: the mean flow region where the standard $k-\varepsilon$ model is solved, and a near-wall region is computed using a different model that resolves the boundary layer accurately.

Near the walls, the one-equation model by Wolfshtein, modified by Chen and Patel [3] is used. The k-equation is solved, with a new dissipation term based on the eddy-viscosity assumption.

$$\frac{\partial}{\partial x_j}(\rho U_j k) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + P_k - \rho\varepsilon \tag{5}$$

The dissipation term and the turbulent viscosity are

$$\varepsilon = k^{3/2}/\ell_\varepsilon \qquad\qquad \mu_t = C_\mu \rho \sqrt{k}\,\ell_\mu \tag{6}$$

The turbulent length scales are prescribed as

$$\ell_\varepsilon = c_\ell n[1-\exp(-R_n/A_\varepsilon)] \qquad\qquad \ell_\mu = c_\ell n[1-\exp(-R_n/A_\mu)] \tag{7}$$

Where the turbulent Reynolds number R_n and the constants are

$$R_n = \sqrt{k}\,n/\nu \qquad A_\varepsilon = 2c_\ell \qquad A_\mu = 70 \qquad c_\ell = \kappa {C_\mu}^{-3/4} \tag{8}$$

n is the normal distance to the wall. The one-equation model is applied near the walls, for $R_n \leq 250$.

Boundary conditions The natural boundary conditions can be applied. At the walls,

$$U = 0$$

$$V = 0$$

$$k = 0$$

ε does not need any boundary conditions, since it is not solved in the near-wall region.

Implementation The separation between the near-wall region and the main flow region has to be computed for each wall; it can take different forms:

- Arbitrary matching line: A preselected grid line can be used. This procedure provides a poor accuracy, but is very stable.
- Averaged matching line: A grid line can be chosen and located at the average thickness of the near wall region. The accuracy is increased compared to the previous solution, since it follows the flow behavior during the iterative resolution. But the geometrical changes of the boundary layer along the wall are still not considered.
- Exact matching line: The "ideal" matching line (a highly broken line) can also be applied, but it is very unstable, mainly due to the steep changes of the variable (here, ε or μ) across the separation. When this step is smoothed, by averaging the value of the last cell with its neighbor, the procedure is relatively stable. This should of course give the best results.

To avoid non-physical boundary layer thicknesses during the first iterations, which would lead to the divergence of the iterative resolution, the matching line has to be arbitrarily fixed. When the flow is "formed", a more accurate procedure can be used. Both second and third approximations have been tested; the best results are obtained from the "Exact matching line".

During the computation, the boundary layer thickness changes, as the solution converges; therefore, the matching line has to be recomputed dynamically to catch this evolution. Updating the matching line is an important source of instability, so it is computed only every, say 10 to 50 iterations.

When the matching line is computed, ε and μ_t are calculated using eq. 6 in the near-wall region. To prevent ε from being solved in the near wall region during the solving procedure, the source term S_u is set to (great $\times\ \varepsilon$) while S_P is set to ($-$great). The discretized equation then becomes

$$\underbrace{\left(\sum a_{NB} - S_P\right)}_{\approx \text{ great}} \Phi_P = \sum a_{NB}\Phi_{NB} + \underbrace{S_U}_{\text{great} \times \Phi_{fixed}}$$

$$\text{i.e} \quad \Phi_P = \Phi_{fixed}$$

2.5.3 Low-Re $k-\varepsilon$ model

A Low Reynolds-number model developed by Lien-Leschziner [4] has been applied. It introduces damping functions that affect the flow only close to the walls. These functions modify the turbulent viscosity, the production and the dissipation terms of the ε-equation. The k-equation remains unchanged in shape, but not the ε-equation

$$\frac{\partial}{\partial x_j}(\rho U_j k) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + P_k - \rho\varepsilon \tag{9}$$

$$\frac{\partial}{\partial x_j}(\rho U_j \varepsilon) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + \frac{\varepsilon}{k}\left(C_{\varepsilon 1} f_{\varepsilon 1} P_k - \left(C_{\varepsilon 2} f_{\varepsilon 2} + \mathrm{Yap}\right)\rho\varepsilon\right) \tag{10}$$

The turbulent viscosity is now defined as

$$\mu_t = C_\mu f_\mu \rho \frac{k^2}{\varepsilon} \tag{11}$$

The damping functions introduced are defined as follows

$$f_\mu = \left[\frac{1-\exp(-0.016 n_\star)}{1-\exp(-0.263 n_\star)}\right] \qquad f_{\varepsilon 1} = \left(1 + \frac{P_k'}{P_k}\right) \qquad f_{\varepsilon 2} = \left[1 - 0.3\exp(-{R_T}^2)\right] \tag{12}$$

$$\text{Yap's length scale correction} = -0.83\left(\frac{k^{3/2}/\varepsilon}{2.44n} - 1\right)\left(\frac{k^{3/2}/\varepsilon}{2.44n}\right)^2 \tag{13}$$

Where P_k', $n_\star$ and R_T are

$${P_k}' = \frac{1.92[1 - 0.3\exp(-R_T^2)]k^{3/2}}{3.53n[1-\exp(-0.263 n_\star)]}\exp(-0.00222{n_\star}^2) \tag{14}$$

$$n_\star = n\sqrt{k}/\nu \qquad R_T = k^2/\nu\varepsilon \tag{15}$$

Wall boundary The following values are prescribed at the node nearest to the wall:

$$U = 0$$

$$V = 0$$

$$k = 0$$

$$\varepsilon = \frac{0.75{P_k}'}{1 - 0.3\exp(-R_T^2)}$$

Implementation Practically, the damping functions are introduced through the arbitrary constant of the standard $k-\varepsilon$ model: C_μ, $C_{\varepsilon 1}$, $C_{\varepsilon 2}$. Therefore, they have to be stored in a new array.

2.5.4 $k-\omega$ model

The $k-\omega$ model introduced by Wilcox [5] replaces the ε transport equation by an ω-equation which can be referred to as the specific dissipation rate, or the rate of dissipation of turbulence per unit energy.

$$\frac{\partial}{\partial x_j}(\rho U_j k) = \frac{\partial}{\partial x_j}\left[(\mu + \mu_t\sigma_\star)\frac{\partial k}{\partial x_j}\right] + P_k - \beta_\star \rho\omega k \tag{16}$$

$$\frac{\partial}{\partial x_j}(\rho U_j \omega) = \frac{\partial}{\partial x_j}\left[(\mu + \mu_t\sigma)\frac{\partial \omega}{\partial x_j}\right] + \gamma\frac{\omega}{k}P_k - \beta\rho\omega^2 \tag{17}$$

The turbulent viscosity is defined, using non-dimensional analysis, by

$$\mu_t = \gamma_\star \frac{\rho k}{\omega} \tag{18}$$

The constants are

$$\sigma = 0.5 \qquad \sigma_\star = 0.5 \qquad \beta = \frac{3}{40} \qquad \beta_\star = \frac{9}{100} \qquad \gamma = \frac{5}{9} \qquad \gamma_\star = 1 \tag{19}$$

The relation between ω and ε can be found as

$$\omega = \frac{\varepsilon}{\beta_\star k}$$

Wall boundary

$$k = \frac{U_\star^2}{\sqrt{\beta_\star}}$$

$$\omega = \frac{U_\star}{\sqrt{\beta_\star}\kappa n}$$

Implementation The easiest way to implement ω transport equation is to transform the ε-equation. Only the source terms have to be changed, since the convective and the diffusive terms remains identical.

Initial values are usually set to small (10^{-10}) by CALC-BFC. It is a very bad initial guess for ω, since it would lead to $\mu_t \approx 1000\frac{10^{-10}}{10^{-10}} \approx 1000$ at the first iteration ! Therefore, ω has to be initialized to a value in the range 10^{-3} to 1.

2.5.5 Low-Re $k-\omega$ model

Basically, this model [5] is a two layer model. In the near-wall region, only one equation for k is solved; ω is fixed. Eq. 21 is nothing else than the exact solution to eq. 17 in the limit $n^+ \to 0$. By defining ω, the goal is just to guarantee the numerical accuracy.

$$\frac{\partial}{\partial x_j}(\rho U_j k) = \frac{\partial}{\partial x_j}\left[(\mu + \mu_t \sigma_\star)\frac{\partial k}{\partial x_j}\right] + P_k - \beta_\star \rho \omega k \tag{20}$$

$$\omega = \frac{6\nu_w}{\beta n^2} \tag{21}$$

Wall boundary k is defined using natural boundary conditions, i.e:

$$k = 0$$

Implementation A matching line splitting the computation domain in two regions has to be computed; the same procedure as for the two-layer $k-\varepsilon$ model has been applied, see section 2.5.2. The separation between the near wall region and the outer region is chosen to fit the condition $n^+ \leq 2.5$.

2.5.6 Two-layer Reynolds stress model

The stationary and incompressible form of the exact Reynolds stresses transport equations can be found to be:

$$\begin{aligned} \underbrace{\rho U_k \frac{\partial \overline{u_i u_j}}{\partial x_k}}_{C_{ij}} &= \underbrace{-\rho\overline{u_i u_k}\frac{\partial U_j}{\partial x_k} - \rho\overline{u_j u_k}\frac{\partial U_i}{\partial x_k}}_{P_{ij}} + \underbrace{\overline{p\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)}}_{\Phi_{ij}} \\ &\quad \underbrace{-\frac{\partial}{\partial x_k}\left(\rho\overline{u_i u_j u_k} + \overline{pu_j}\delta_{ik} + \overline{pu_i}\delta_{jk} - \mu\frac{\partial \overline{u_i u_j}}{\partial x_k}\right)}_{D_{ij}} \\ &\quad \underbrace{-2\mu\overline{\frac{\partial u_i}{\partial x_k}\frac{\partial u_j}{\partial x_k}}}_{\varepsilon_{ij}} \end{aligned} \tag{22}$$

The RSM code used for this work has been written by Stefan Jansson [10], during his Ph. D work. It has been updated to fit the newer version of CALC-BFC used here, and modified to this test case.

More details regarding the different approximations raised, and the implementation can be found in [11], [12] and [10]

Near the walls, the one-equation model by Chen and Patel described in 2.5.2 is used.

2.6 Implementation of new solvers

The limits of TDMA and SIP solvers

- The Tri-Diagonal-Matrix Algorithm [8] is a very powerful and convenient equation solver since it requires a rather small computer storage. However, it is time consuming, since it has to scan the computation domain line by line.
- The Strongly Implicit Solver [9], one of the most popular iterative solvers used in CFD, is based on a recursive algorithm and is therefore not vectorizable in a straightforward manner.

These two solvers cannot be used for advanced procedures, such as coupled resolutions or unstructured meshes. A general solver, suitable for any types of matrixes, and not only a three or seven diagonals arrangements, is therefore needed.

The SLAP 2.0 library (Sparse Linear Algebra Package) provides a large choice of such solvers and different preconditioning procedures.

2.6.1 The Slap library

The solvers proposed in this library can be used independently with two different preconditioned iterative methods, which are:

- The Diagonal Scaling (SD).
- The Incomplete Factorization (LU in most of the cases).

The following table reviews briefly the solvers of interest.

	Preconditioners	
Solvers	Diagonal Scaling	incomplete factorization (LU)
Conjugate Gradient (symmetric)	SSDCG	SSICCG
Biconjugate Gradient	SSDBCG	SSLUBC
Biconjugate Gradient Square	SSDCGS	SSLUCS
Generalized Minimal Residual	SSDGMR	SSLUGM

For more details about these different solvers, see [6] and [7]
The SLAP routines use a common data structure, allowing two different formats for the input data. The Triad-Format is easy to implement, but is also more CPU consuming than the complicated Column-Format.

The SLAP Triad-Format The non-zero elements are stored in a vector A. Two other vectors (IA and JA) are used to locate each element.
The following 5×5 matrix containing 11 non-zero elements, is used to give an example of the data storage.

$$\begin{bmatrix} 11 & 12 & 0 & 0 & 15 \\ 21 & 22 & 0 & 0 & 0 \\ 0 & 0 & 33 & 0 & 35 \\ 0 & 0 & 0 & 44 & 0 \\ 51 & 0 & 53 & 0 & 55 \end{bmatrix}$$

The SLAP Triad format is in this case

$$\begin{array}{rcl} A & = & [\ 11\ \ 21\ \ 51\ \ 22\ \ 12\ \ 33\ \ 53\ \ 44\ \ 15\ \ 35\ \ 55\ ] \\ IA & = & [\ 5\ \ 1\ \ 1\ \ 3\ \ 1\ \ 5\ \ 5\ \ 2\ \ 3\ \ 4\ \ 2\ ] \\ JA & = & [\ 1\ \ 2\ \ 1\ \ 3\ \ 5\ \ 3\ \ 5\ \ 2\ \ 5\ \ 4\ \ 1\ ] \end{array}$$

The implementation of this procedure is straightforward.

The SLAP Column-format Basically, SLAP Column-Format uses three vectors: one containing the data (A), and two considered as pointers (IA and JA).
The original matrix is scanned column by column. For each column, the diagonal element is considered at first, and is followed by the remaining non zero elements, going down the column.
IA holds the row index of each element.
JA acts as a pointer on A, indicating the first element of a column (i.e the diagonal element).
Using the previous example, the new arrangement is:

$$\begin{array}{rcl} A & = & [\ 11\ \ 21\ \ 51\ \ 22\ \ 12\ \ 33\ \ 53\ \ 44\ \ 15\ \ 35\ \ 55\ ] \\ IA & = & [\ 1\ \ 2\ \ 5\ \ 2\ \ 1\ \ 3\ \ 5\ \ 4\ \ 5\ \ 1\ \ 3\ ] \\ JA & = & [\ 1\ \ 4\ \ 6\ \ 8\ \ 9\ \ 12\ ] \end{array}$$

One can first notice that the memory requirements are smaller using this procedure, but the real gain is CPU time, since the SLAP routines are using this common data storage, no translation step is required.

2.6.2 Implementation of the SLAP Column-Format in CALC-BFC

The figure 8 presents, for a 3D case, the set of linearized equations that have to be solved. As explained before, a_P have to be stored first in A, the remaining elements are then processed, going down the column. The coefficients are therefore stored in the following order: a_P, a_H, a_N, a_E, a_W, a_S, and a_L. This sequence has to be looped over the whole computation domain, i.e $(NI-2)(NJ-2)(NK-2)$ times.

The location of the different coefficients in the matrix can be easily related to the corresponding a_P coefficient. Assume that a_P is spotted by a variable $i2$ in IA, according to figure 8, a_W is then pointed by $(i2+1)$ and a_E by $(i2-1)$; a_S is located by $(i2+NI-2)$, a_L by $(i2+(NI-2)(NJ-2))$ and so on.

Of course, the off-diagonal elements, along a column, are related to the a_P coefficients of other nodes (a_W in column i is related to $a_{P(i+1,j,k)}$). Their locations have to be converted between the vectorized arrangement into the 3D matrix coordinate system i, j, k.

Some coefficients must not appear in the matrix (in a 2D case, the terms a_H and a_L have no meaning). A deep analysis of figure 8 helps to understand which cells may involve such situation.
a_H has no sense when k=2. On the same principle, a_L, a_N, a_S, a_E, a_W, respectively disappear when $k = Nkm1$, $j = 2$, $j = Njm1$, $i = 2$, and $i = Nim1$.
When these coefficients are not in the matrix, they have to be added to the right hand side of the equation (the source term). The source term related to an a_P term is stored using the symmetry of the different coefficients. If, for example, a_E is zero in the column containing the diagonal element $a_{P(i,j,k)}$, then the west coefficient located on its line ($a_{W(i,j,k)}$) must also be zero; $a_{W(i,j,k)}\Phi_{i-1,j,k}$ is, therefore, added to the source term. This procedure is applied to a_N relatively to a_S, a_L relatively to a_H, and vice-versa.

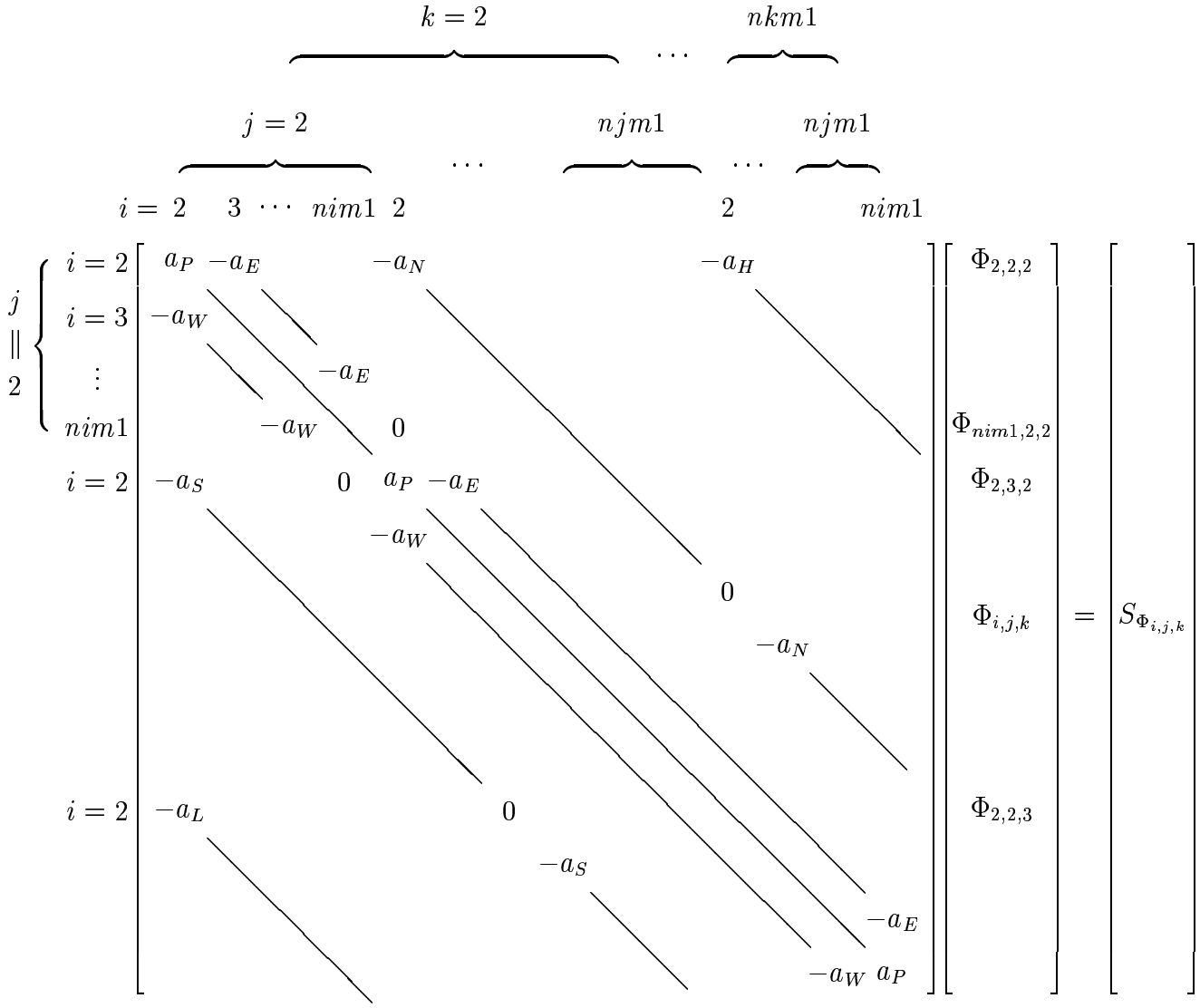
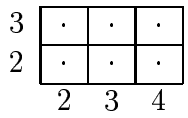


Figure 8: Structure of a general matrix storage

An example Consider a very small 2D mesh ($Nim1 = 4$, $Njm1 = 3$).



A is then a 6×6 matrix as shown below:

$$A = \begin{bmatrix} a_{P(2,2)} & -a_{E(2,2)} & & -a_{N(2,2)} & & \\ -a_{W(3,2)} & a_{P(3,2)} & -a_{E(3,2)} & & -a_{N(3,2)} & \\ & -a_{W(4,2)} & a_{P(4,2)} & 0 & & -a_{N(4,2)} \\ -a_{S(2,3)} & & 0 & a_{P(2,3)} & -a_{E(2,3)} & \\ & -a_{S(3,3)} & & -a_{W(3,3)} & a_{P(3,3)} & -a_{E(3,3)} \\ & & -a_{S(4,3)} & & -a_{W(4,3)} & a_{P(4,3)} \end{bmatrix}$$

To store this matrix using the SLAP Column-Format, the following procedure have to be performed.

First column ($i = 2$, $j = 2$, $i2 = 1$)

- Diagonal element:
  $A(1) = a_P(i, j) = a_P(2, 2)$
  $IA(1) = 1$

- The north term does not appear in this first column ($j = 2$)
  $a_S(2, 2)\Phi_{2,1}$ is added to $S_{\Phi_{2,2}}$

- No east term either ($i = 2$),
  $a_W(2, 2)\Phi_{1,2}$ is added to $S_{\Phi_{2,2}}$

- West term:
  $A(2) = -a_W(i + 1, j) = -a_W(3, 2)$
  $IA(2) = i2 + 1 = 2$

- South term:
  $A(3) = -a_S(i, j + 1) = -a_S(2, 3)$
  $IA(3) = i2 + (NI - 2) = 4$

$JA(1) = 1$
$JA(2) = JA(1)+$ number of coefficients in the current column $= 4$.

Second column ($i = 3$, $j = 2$, $i2 = 2$)

- Diagonal element:
  $A(4) = a_P(3, 2)$
  $IA(4) = i2 = 2$

- The north term is still not present in the column,
  $a_S(3,2)\Phi_{3,1}$ is added to $S_{\Phi_{3,2}}$

- East term:
  $A(5) = -a_E(i-1,j) = -a_E(2,2)$
  $IA(5) = i2 - 1 = 1$

- West term:
  $A(6) = -a_W(4,2)$
  $IA(6) = i2 + 1 = 3$

- South term:
  $A(7) = -a_S(3,3)$
  $IA(7) = i2 + (NI - 2) = 2 + 3 = 5$

$JA(3) = JA(2) + 4 = 8$

Third column $(i = 4,\ j = 2,\ i2 = 3)$

- Diagonal element:
  $A(8) = a_P(4,2)$
  $IA(8) = i2 = 3$

- The north term is, again, not present this column $(j = 2)$
  $a_S(4,2)\Phi_{4,1}$ is added to $S_{\Phi_{4,2}}$

- East term:
  $A(9) = -a_E(3,2)$
  $IA(9) = i2 - 1 = 2$

- The west term is also removed $(i = nim1)$
  $a_E(4,2)\Phi_{5,2}$ is added to $S_{\Phi_{4,2}}$

- South term:
  $A(10) = -a_S(4,3)$
  $IA(10) = i2 + (NI - 2) = 3 + 3 = 6$

$JA(4) = JA(3) + 3 = 11$

Fourth column $(i = 2,\ j = 3,\ i2 = 4)$

- Diagonal element:
  $A(11) = a_P(2,3)$
  $IA(11) = i2 = 4$

- North coefficient:
  $A(12) = -a_N(2,2)$
  $IA(12) = i2 - (NI - 2) = 4 - 3 = 1$

- The east term is not present $(i = 2)$
  $a_W(2,3)\Phi_{1,3}$ is added to $S_{\Phi_{2,3}}$
  and so on ...

For more details, one could refer to the source code, in appendix C.

2.6.3 Coupled resolution of the k- and ε-equations

Using the standard $k - \varepsilon$ model, the transport equations for the k- and ε-equations have the following form:

$$C_k = D_k + \underbrace{\frac{\partial U_i}{\partial x_j}\left(\frac{\partial U_i}{\partial x_j} + \frac{U_j}{\partial x_i}\right) C_\mu \rho \frac{k^2}{\varepsilon}}_{S_k^k \times k} \underbrace{-\rho\varepsilon}_{S_\varepsilon^k \times \varepsilon}$$

$$C_\varepsilon = D_\varepsilon + \underbrace{\frac{\partial U_i}{\partial x_j}\left(\frac{\partial U_i}{\partial x_j} + \frac{U_j}{\partial x_i}\right) C_{\varepsilon 1} C_\mu \rho k}_{S_k^\varepsilon \times k} \underbrace{-C_{\varepsilon 2}\rho\frac{\varepsilon^2}{k}}_{S_\varepsilon^\varepsilon \times \varepsilon}$$

Each equation holds a source (or sink) term proportional to k, and another one connected to ε.

When the usual non-coupled resolution is applied, the source terms can either be computed using the known values for k and ε (fully explicit), or be expressed in terms of the variable-dependent source term (semi-implicit).

The coupled resolution of k and ε, using a general purpose solver provided by SLAP, allows to develop "fully" implicit source terms, using the unknown values of k and ε.

The coupling procedure does usually not improve the stability of the iterative resolution, since it introduces more implicit terms, but can substantially increase the convergence rate, when the coupled equations are strongly inter-dependent. That may not be true with $k - \varepsilon$ equations, where only four different terms can be coupled:

$$\begin{aligned}
S_k^k &= \frac{\partial U_i}{\partial x_j}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) C_\mu \rho \frac{k}{\varepsilon} \\
S_\varepsilon^k &= -\rho \\
S_\varepsilon^\varepsilon &= -C_{\varepsilon 2}\rho\frac{\varepsilon}{k} \\
S_k^\varepsilon &= \frac{\partial U_i}{\partial x_j}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) C_{\varepsilon 1} C_\mu \rho
\end{aligned}$$

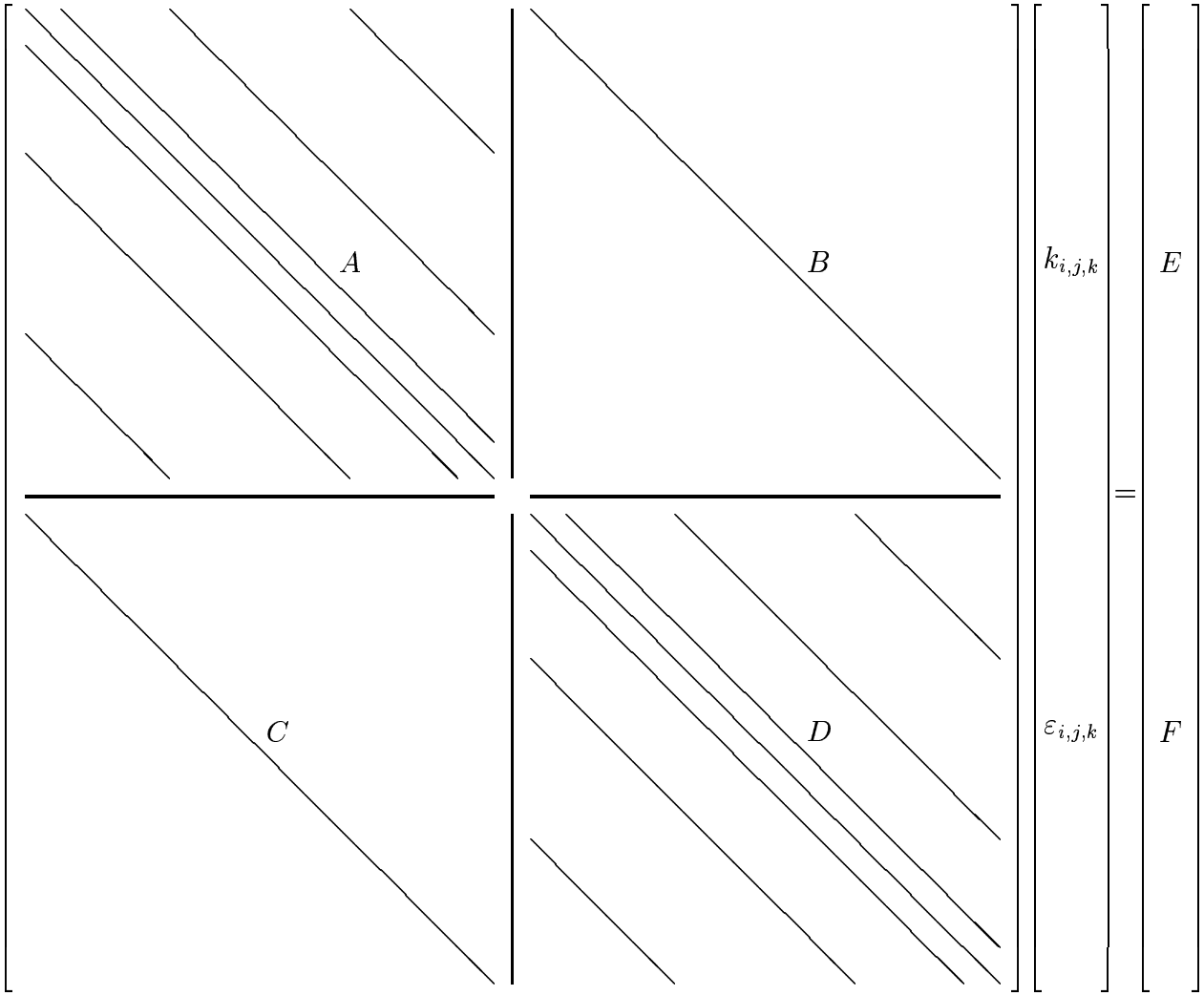


Figure 9: Structure of the matrix storage when 2 variables are coupled

Implementation in CALC-BFC The discretized form of the k and ε transport equations is:

$$k \text{ equation}: \quad a_P k_P = \Sigma a_{nb}^k k_{nb} + S_U^k + S_P^k k_P$$

$$\varepsilon \text{ equation}: \quad a_P \varepsilon_P = \Sigma a_{nb}^\varepsilon \varepsilon_{nb} + S_U^\varepsilon + S_P^\varepsilon \varepsilon_P$$

Where the source terms are

$$S_U^k = S_k^k \times k \qquad S_P^k = S_\varepsilon^k \frac{\varepsilon}{k}$$

$$S_U^\varepsilon = S_k^\varepsilon \times k \qquad S_P^\varepsilon = S_\varepsilon^\varepsilon$$

One may notice that S_k^k is not coupled in the k-equation: S_k^k is always positive, substracted to the main diagonal coefficient it would reduce the diagonal dominancy, and therefore reduce the convergence rate.

The coupled resolution leads to:

$$k \text{ equation}: \quad a_P k_P = \Sigma a_{nb}^k k_{nb} + S_U^k + S_{kP}^k k_P + S_{\varepsilon P}^k \varepsilon_P$$

$$\varepsilon \text{ equation}: \quad a_P \varepsilon_P = \Sigma a_{nb}^\varepsilon \varepsilon_{nb} + S_U^\varepsilon + S_{kP}^\varepsilon k_P + S_{\varepsilon P}^\varepsilon \varepsilon_P$$

Where the new source terms are:

$$(A) \quad S_{kP}^k = 0 \qquad (B) \quad S_{\varepsilon P}^k = S_\varepsilon^k$$

$$(C) \quad S_{kP}^\varepsilon = S_k^\varepsilon \qquad (D) \quad S_{\varepsilon P}^\varepsilon = S_\varepsilon^\varepsilon$$

$$(E) \quad S_U^k = S_k^k \times k \qquad (F) \quad S_U^\varepsilon = 0$$

Arrangement of k- and ε-equation in the coupled matrix The matrix filling procedure is mainly based upon the general SLAP Column-Format. The algorithm follows:

- The first block A is stored as usual along with its constant source terms.
- Some space has to be saved in each column to store the coupling terms C. Dummy values are just added to A, and located through IA, $(ni-2)(nj-2)(nk-2)$ cells under the main diagonal element.
- While the second coupling term B is stored, memory has to be reserved to provide the storage of the main ε coefficients (D). A and D have similar shape, we can therefore predict where the coefficients should be stored in A and IA.
- D is stored using the usual procedure.
- C is set.

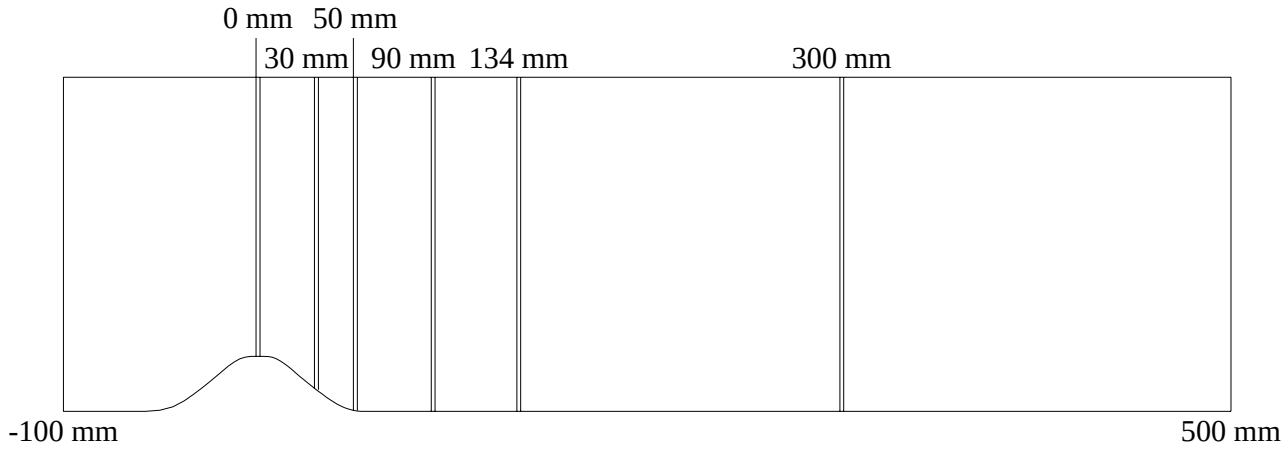


Figure 10: Location of the selected profiles

3 Results analysis

Different factors have been studied.

The mean velocities and turbulent kinetic energy profiles are plotted at six selected locations, see figure 10.

- One on the top of the hill ($x = 0$ mm), where the fluid reaches its maximum velocity.
- Two on the back side of the hill ($x = 30$ mm and $x = 50$ mm), in the recirculation zone.
- Three after the hill: the first one is located at $x = 90$ mm, the second at $x = 134$ mm, where the recirculation zone ends, and the last is chosen far downstream ($x = 300$ mm) where the flow may have recovered the shape of a fully developed turbulent flow.

The recirculation zone is visualized using streamlines plots. The length of the recirculation zone has also been used as a criterion.

3.1 Influence of the code

Considering an identical mesh, discretization schemes, turbulent model, and boundary conditions, two different codes should provide very similar results (ideally, there should display no difference at all!).

Sven Perzon [13] worked at the same test case, using the commercial software CFDS-FLOW3D (AEA technology, Harwell Laboratory, UK). One of the turbulence models he

used was the standard $k-\varepsilon$ model. The comparison shown below is based on computations carried out with a fine mesh (122×50 cells). Both computations used QUICK to discretize the convective terms for the mean velocities, and Hybrid for the turbulent quantities. The inlet and outlet boundary conditions are also identical. On the wall, different wall functions have been applied.

Contrary to what one could expect, the results, shown in figures 11, 12, and 13, present major differences. Such disparity between the results is likely to be introduced by the different wall functions applied.

Of course, the numerical accuracy of each code affects the results, but not in the same order.

3.2 Influence of the discretization scheme

The computations presented (see figures 14, 15, and 16) have been carried out using the Low Reynolds number model on the finest mesh available (128×100 interior cells).

Hybrid shows significantly different results, globally less accurate than high-order schemes. It is interesting to notice that the numerical errors introduced along with Hybrid tends sometimes (mainly on k profiles figure 16, $x = 30$, 50 mm) to overcome the model lacks and gives better results than the other schemes.

One can also notice that the gains brought by Van Leer, when it is applied on the turbulent quantities are almost negligeable. The efficiency is even lower using Van Leer, since it increases significantly the computation cost.

3.3 Influence of the mesh

High-Re models The results shown in figures 17, 18, and 19 have been computed using the standard $k-\varepsilon$ model. The discretization schemes used is QUICK for the mean flow quantities and Van Leer for the turbulent quantities.

Significant differences can be found between these four meshes (see figure 3)

74×25 and 148×50: The results obtained using these two meshes are inaccurate. The errors seem to be introduced when the wall functions are not applied correctly, since for these two meshes, y^+ does not lie within the right limits. The coarser mesh has been derived from the finer one, with half cells in x- and y- direction, it is therefore not surprising to get similar results.

78×49 gives very good results, despite the relatively low quantity of cells in x-direction, and the orthogonality problems that can occur, close to the hill (these orthogonality problems might balance the errors introduced by the coarse grid, and lead to the present good results).

122×50 is well refined both in x- and y- direction, the nodes closest to the wall is accurately located within the y^+ interval. It is therefore not surprising to obtain these good results.

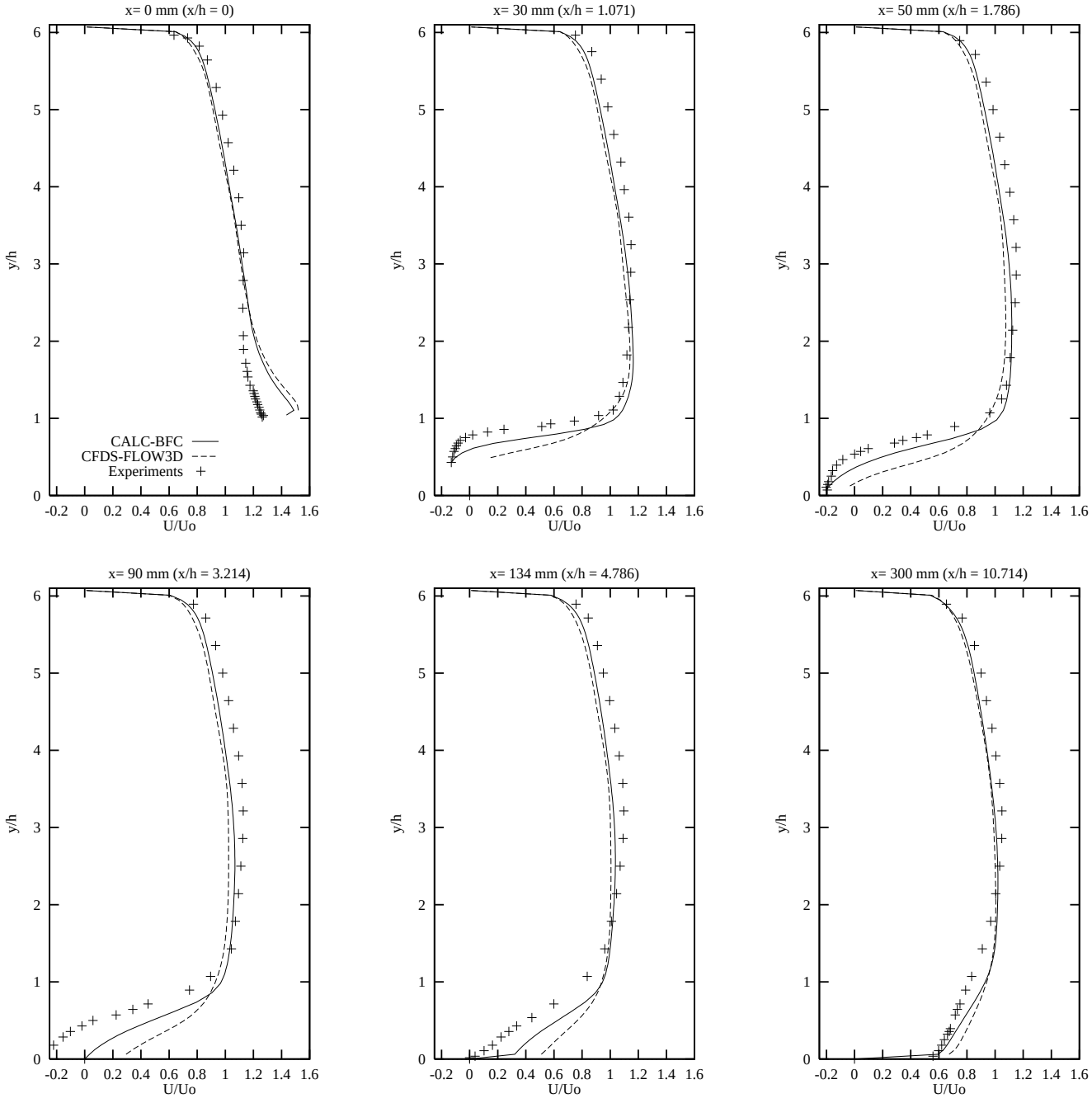


Figure 11: U mean velocity profiles with two different codes.

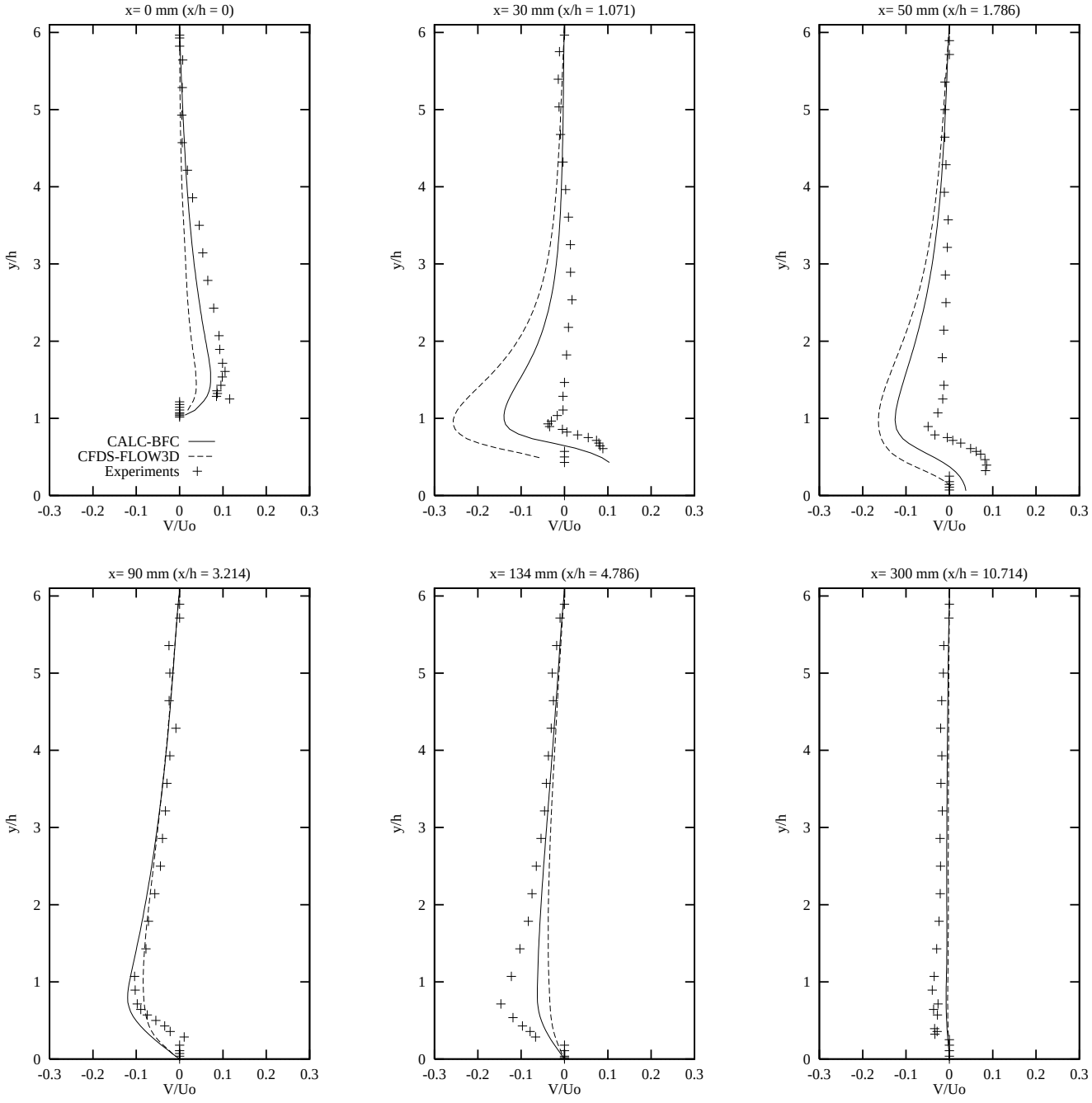


Figure 12: V mean velocity profiles with two different codes.

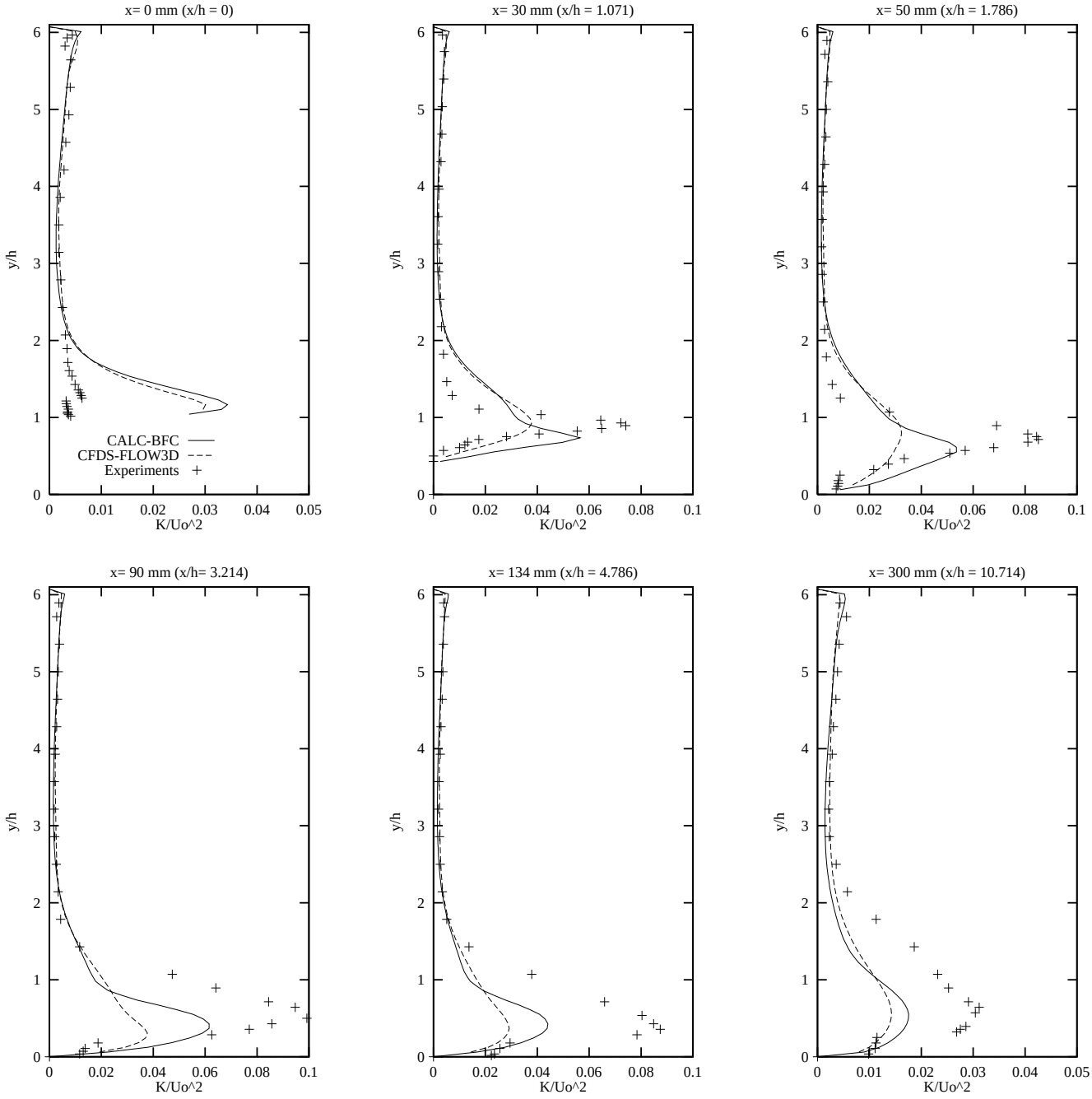


Figure 13: Turbulent kinetic energy profiles with two different codes.

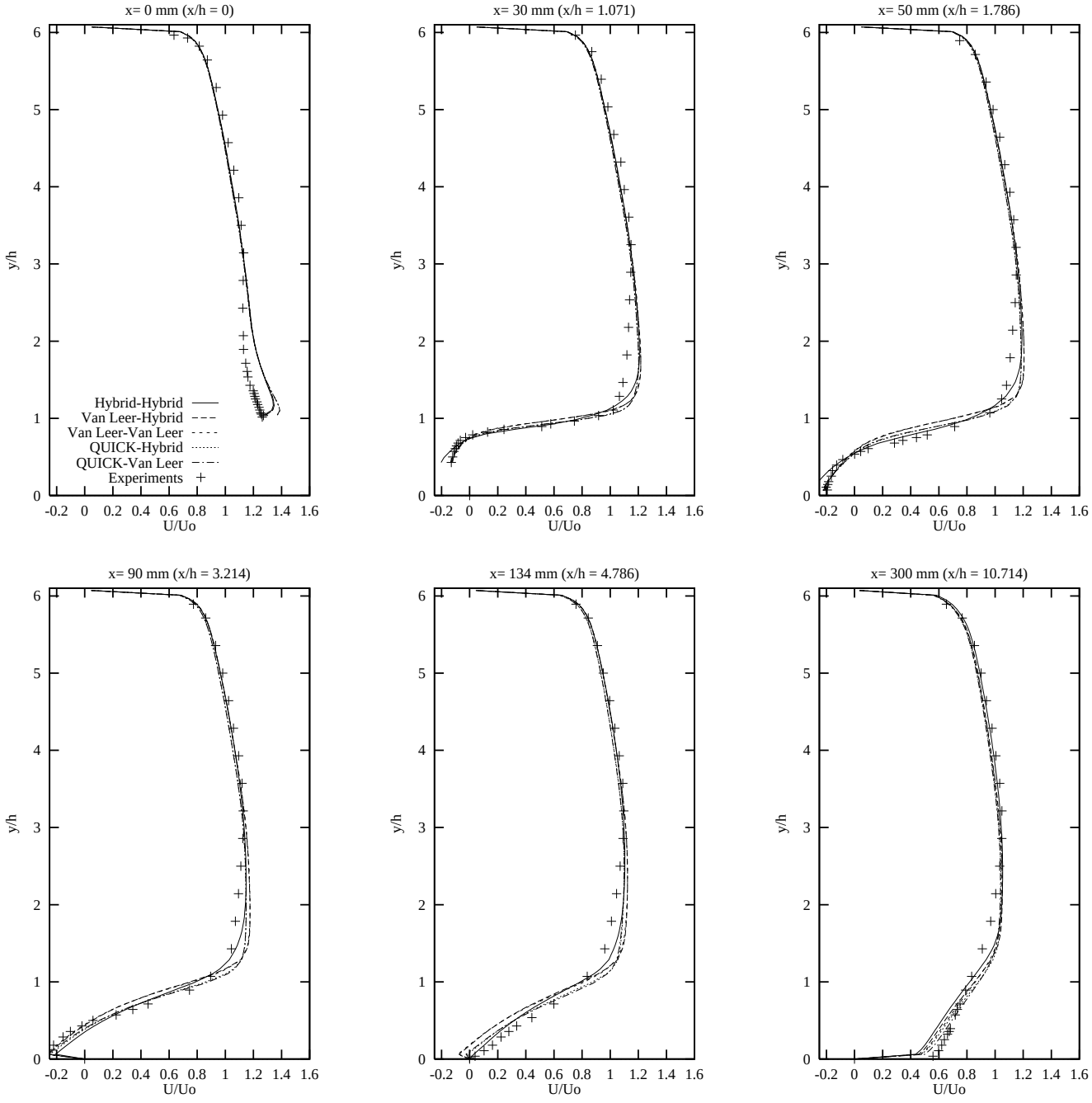


Figure 14: U mean velocity profiles with the different schemes combinations.

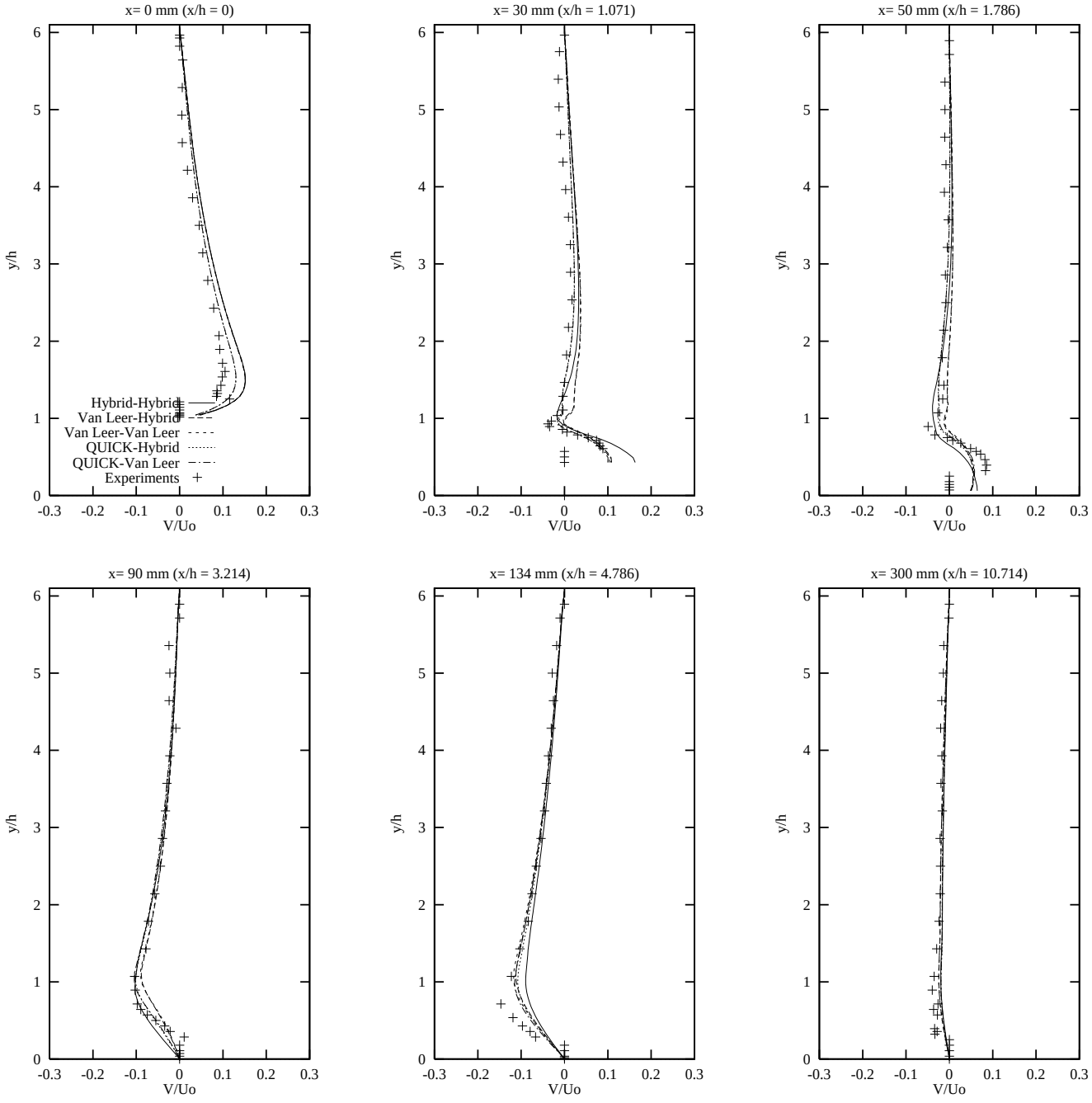


Figure 15: V mean velocity profiles with the different schemes combinations.

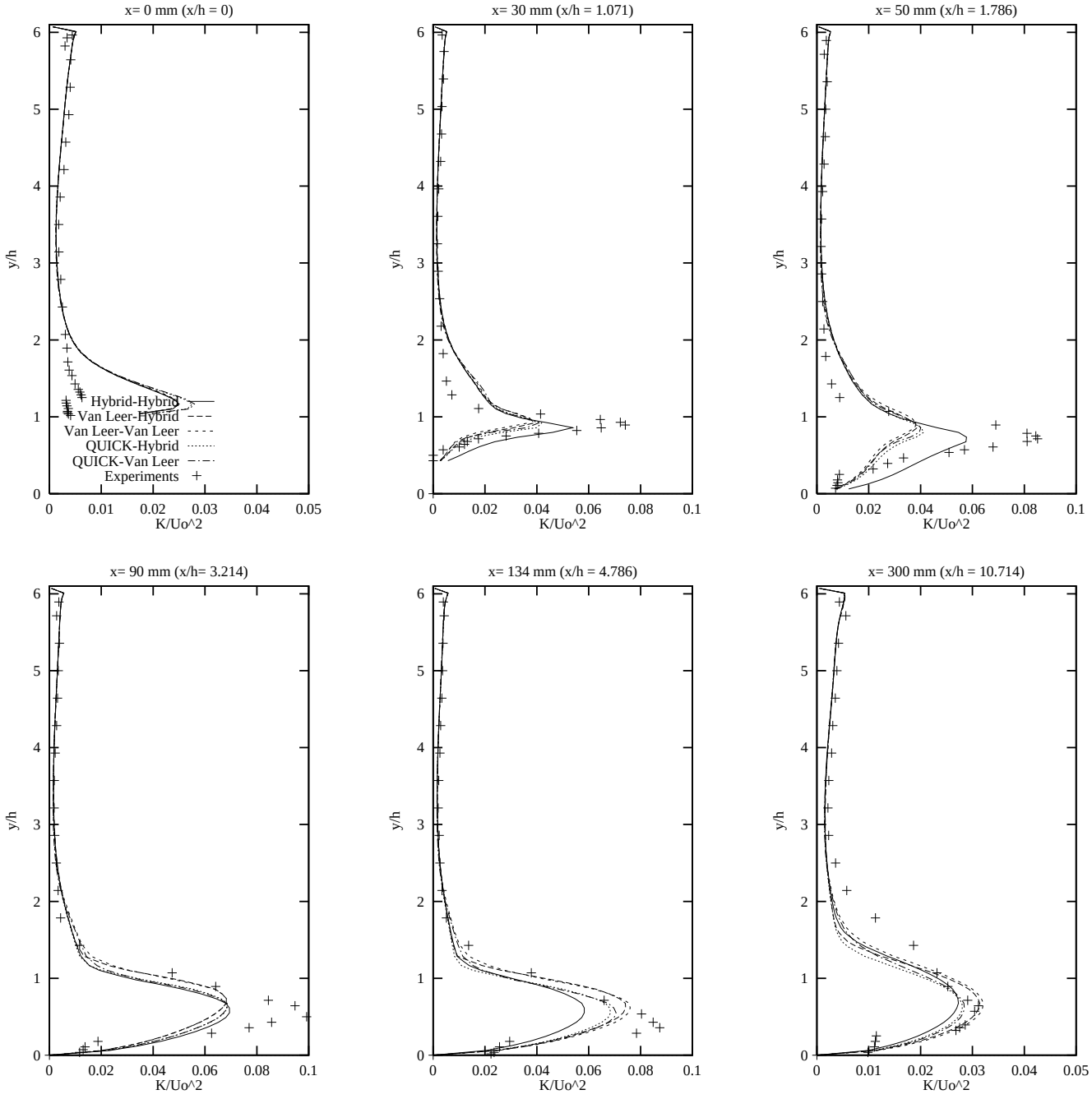


Figure 16: Turbulent kinetic energy profiles with the different schemes combinations.

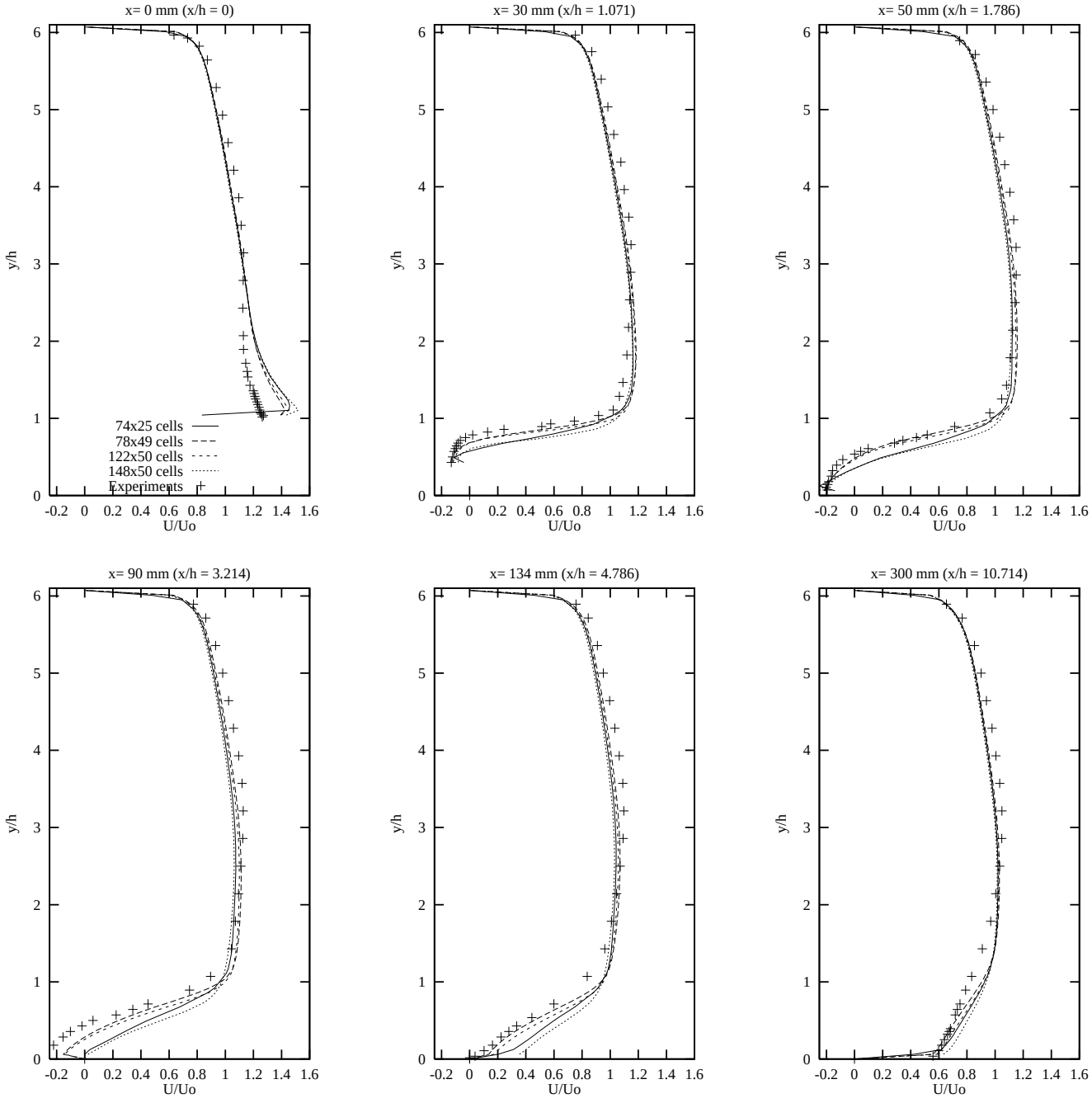


Figure 17: U mean velocity profiles with the different meshes.

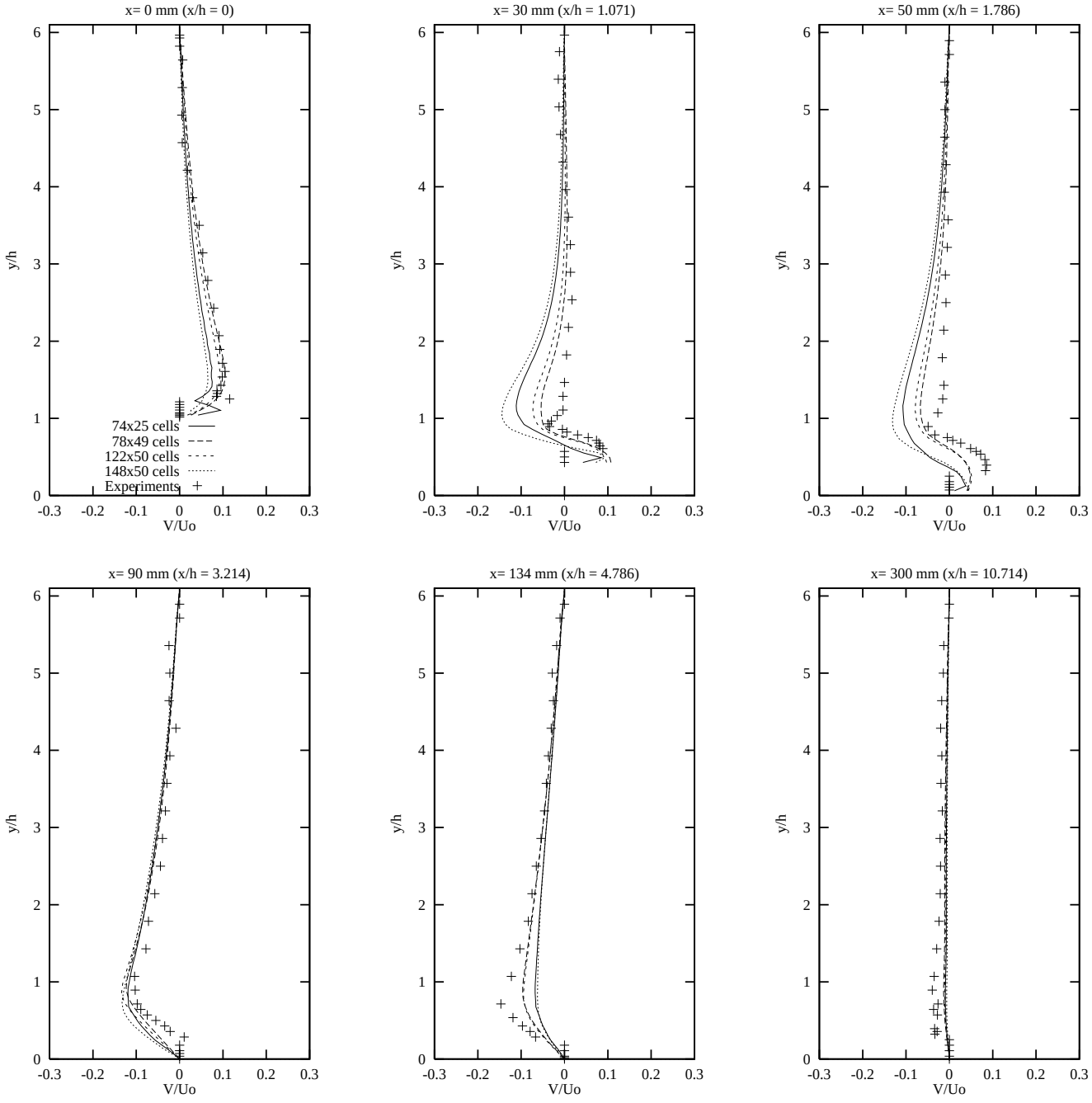


Figure 18: V mean velocity profiles with the different meshes.

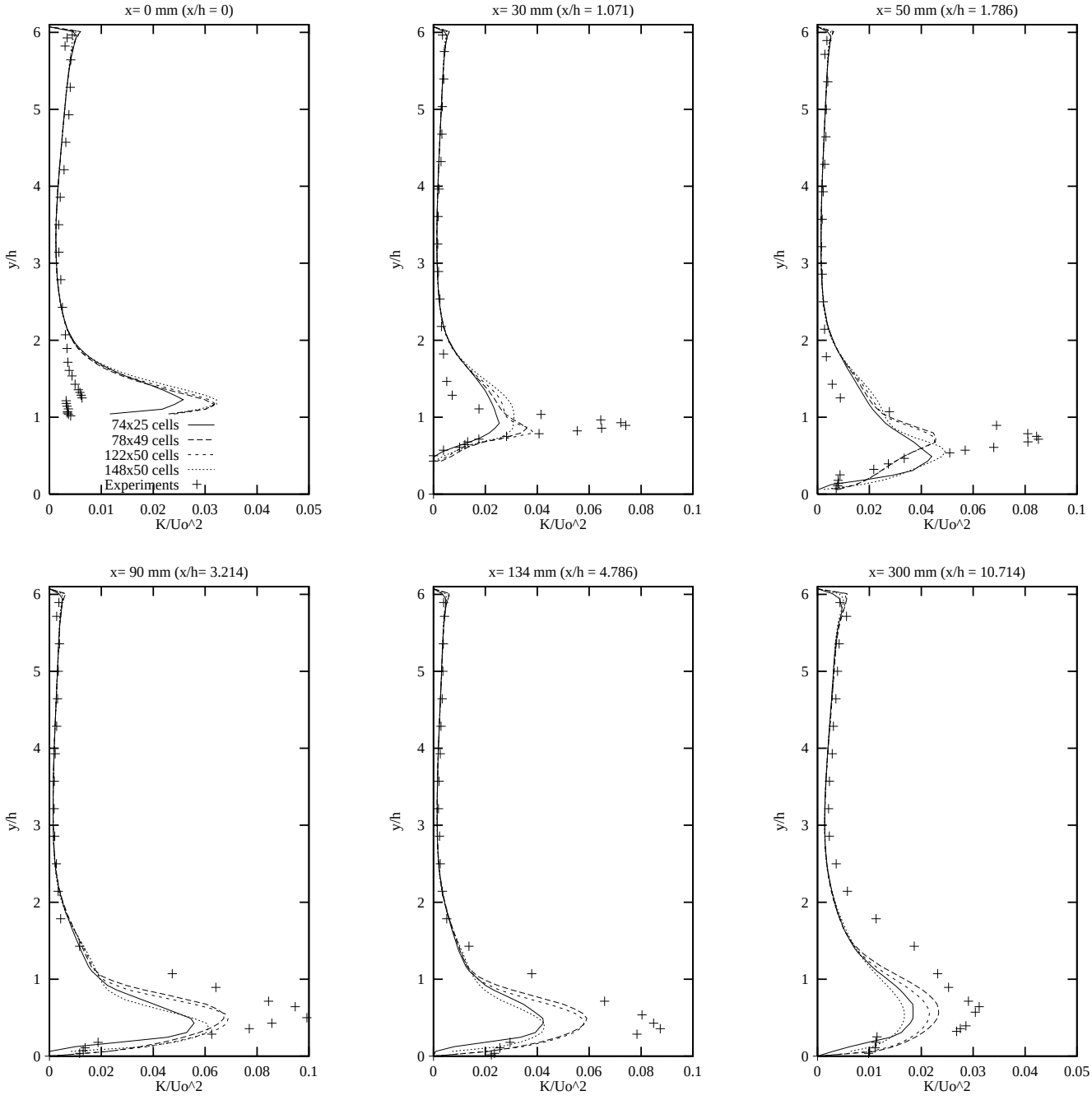


Figure 19: Turbulent kinetic energy profiles with the different meshes.

Low-Re models Figures 20, 21, and 22 show the results of computations with the two layer $k - \varepsilon$ model, using QUICK and Van Leer as discretization schemes.

Again, the coarser mesh is derived from the finest one. The differences between the two meshes are very small, and an increase the number of cells would not improve significantly the results.

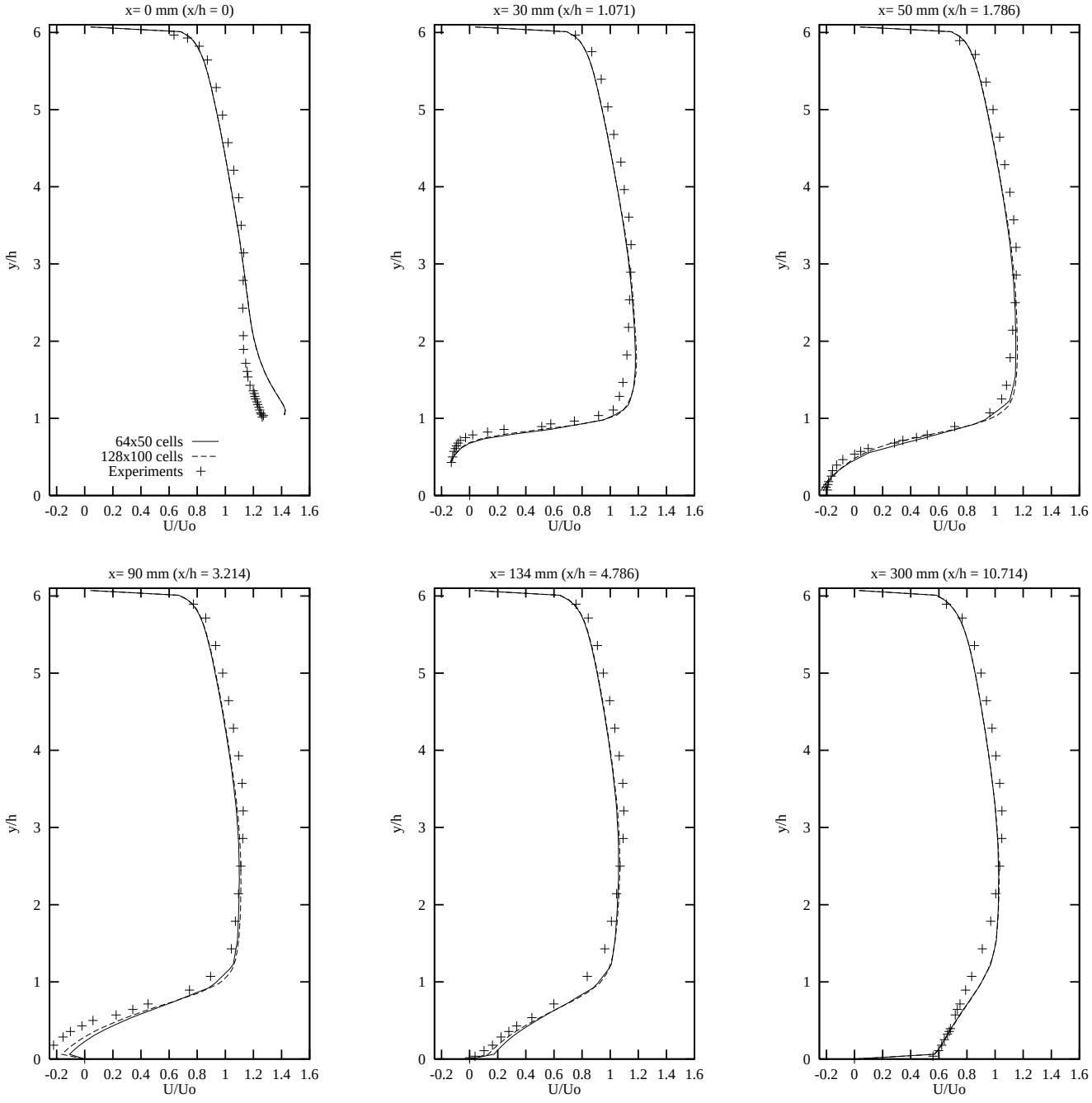


Figure 20: U mean velocity profiles with the different low Reynolds meshes.

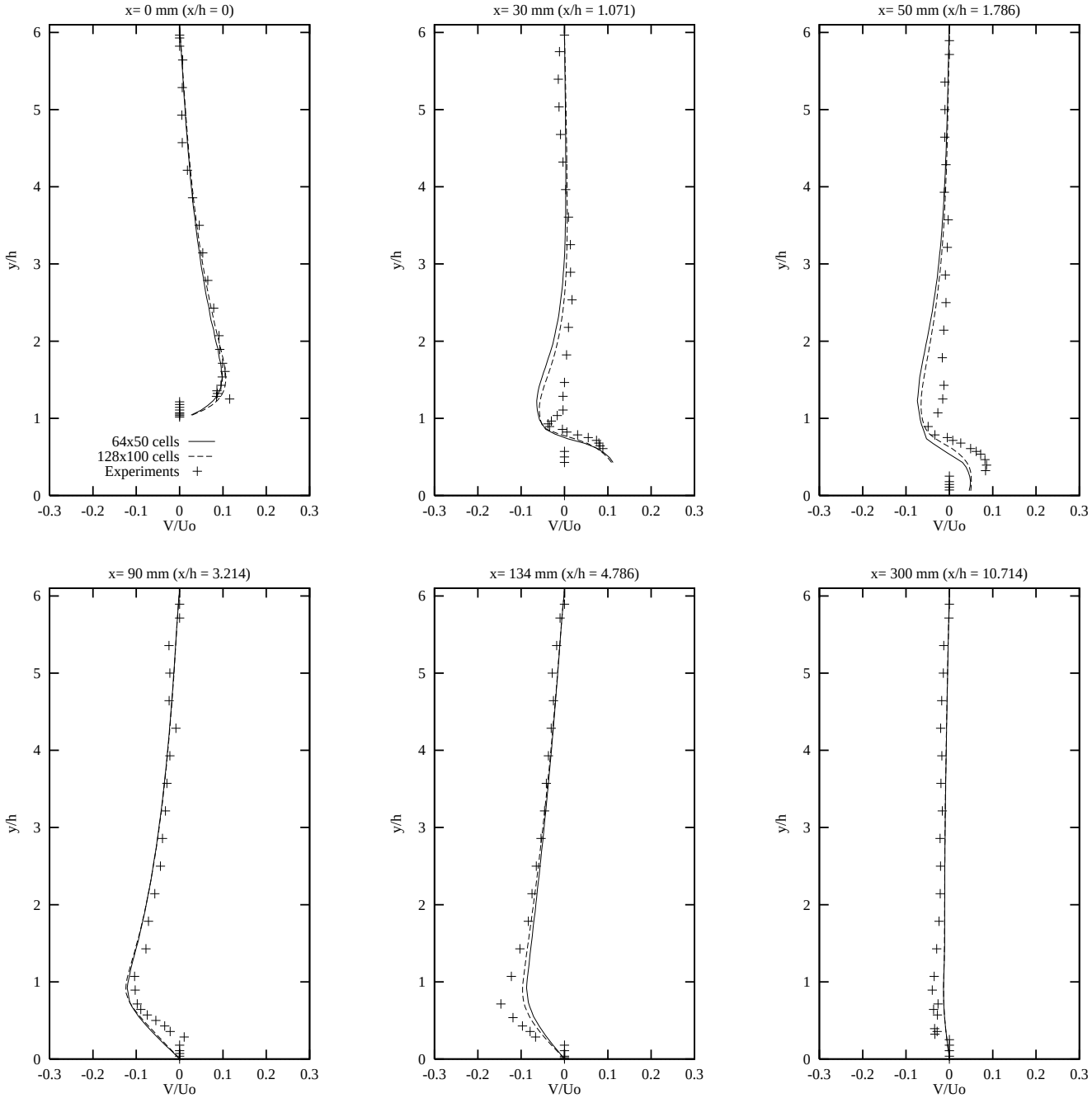


Figure 21: V mean velocity profiles with the different low Reynolds meshes.

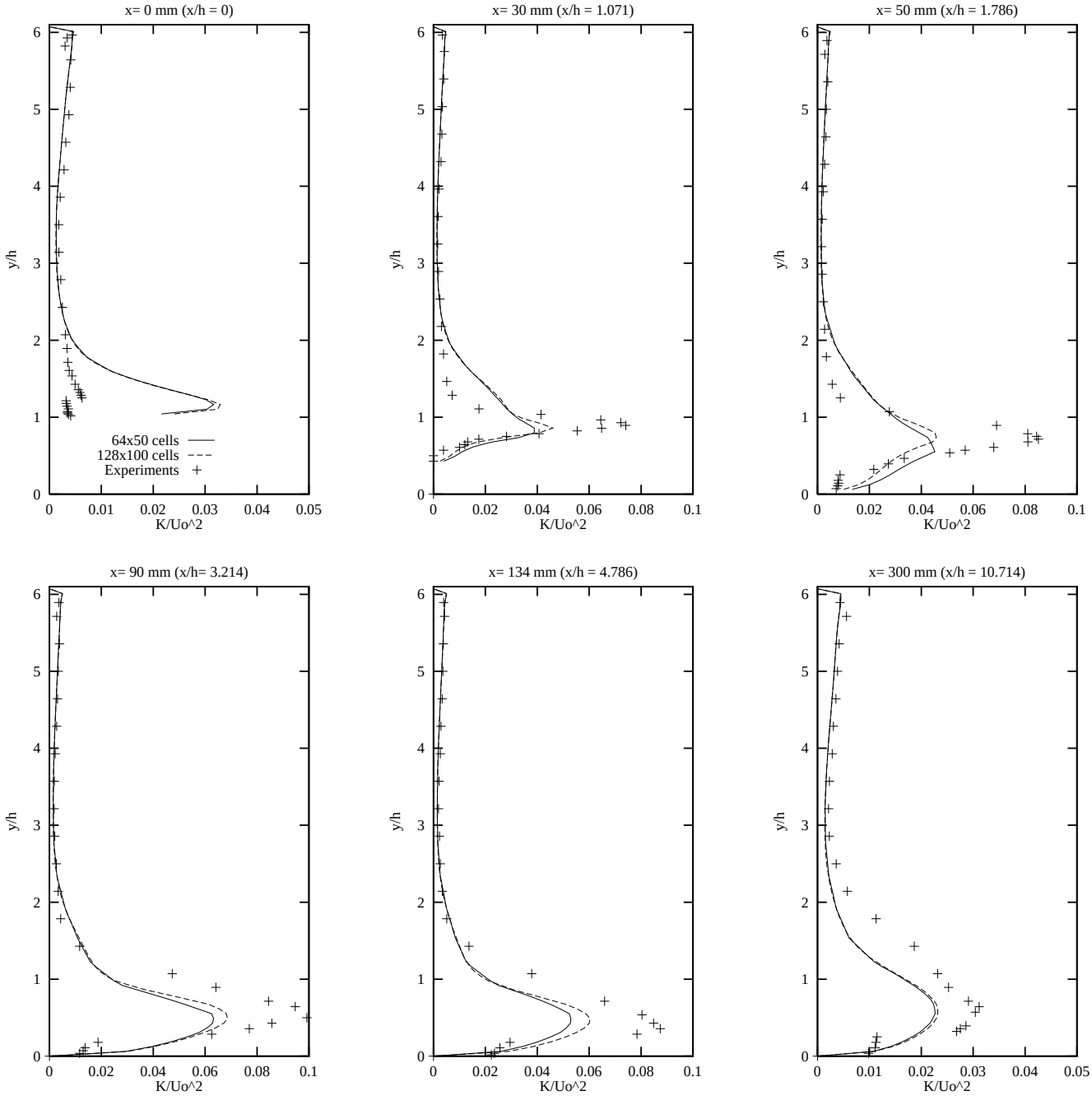


Figure 22: Turbulent kinetic energy profiles with the different low Reynolds meshes.

3.4 Influence of the turbulence model

The computation results displayed here have been computed using a fine mesh (122×50 for High-Re models, and 128×100 for Low-Re models). The discretization schemes are always QUICK on the mean flow quantities, and Van Leer on the turbulent quantities.

The implementation of the Reynolds Stress Model was "new" when this report was written. The code might not be "bug-free", and therefore, the results must be considered very carefully. The very long and fat recirculation zone tends to confirm that there is a bug somewhere in the RSM code.

A rough analysis based upon the shape and the length of the recirculation zone brings the main ends, and is improved by a careful study of the velocities (and k) profiles.

	Separation point		Reattachement point	
	x/h	x (mm)	x/h	x (mm)
$k-\varepsilon$	0.535	15	4.035	113
$k-\omega$	0.607	17	3.517	98.5
Two layer $k-\varepsilon$	0.356	9.97	3.904	109.3
Low-Re $k-\varepsilon$	0.287	8.03	5.378	150.5
Low-Re $k-\omega$	0.242	6.77	6.540	183.1
Two layer RSM	*0.275*	*7.7*	*7.47*	*209.1*
Experiments	0.428	12	4.821	135

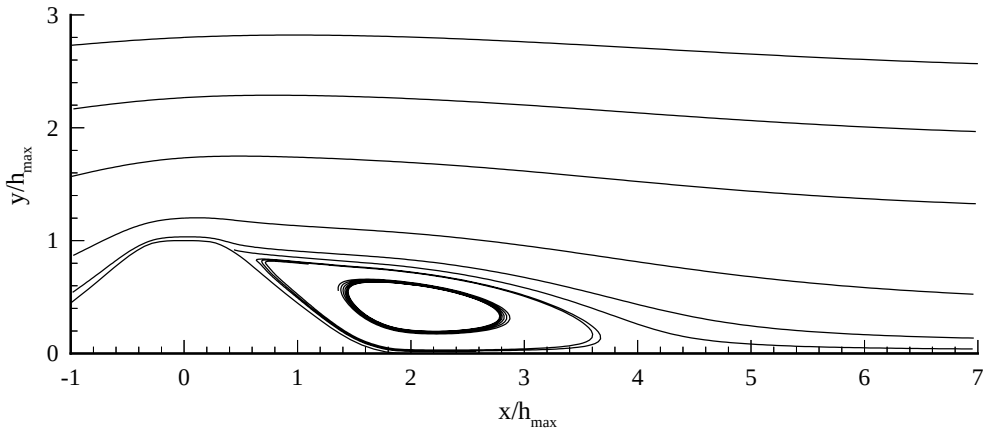
High-Re models Both High-Re $k-\varepsilon$ and $k-\omega$ models (see figure 23) predict too small recirculation zones; in each case, it starts late, while the reattachment occurs too early (20 to 35 mm before the experimental reattachement point).

However, the results obtained using the standard $k-\varepsilon$ model are more accurate than using the standard $k-\omega$ model. The figures 25, 26, and 27 displaying the velocity and turbulent kinetic energy profiles tend to strengthen this first conclusion.

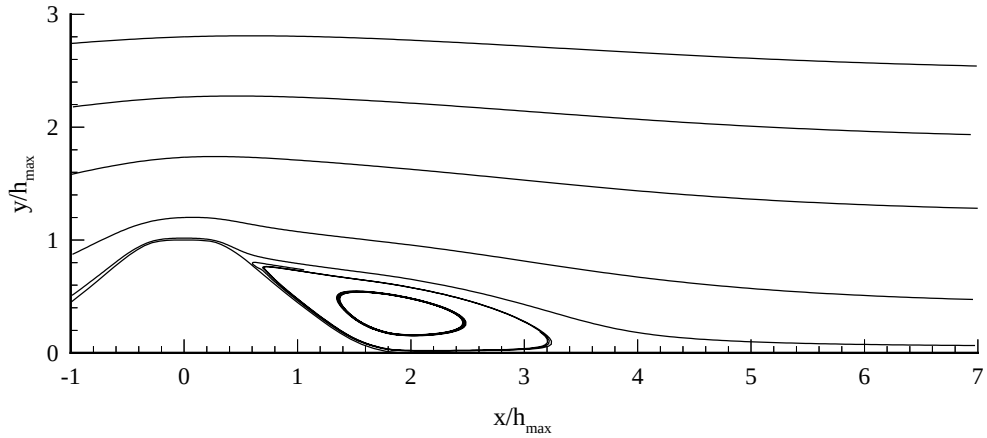
Low-Re models The results obtained using low Reynolds number models show important discrepancies.

- The Two layer $k-\varepsilon$ model predicts a relatively small bubble, 20% shorter than in the experiments.
- The Low-Re $k-\varepsilon$ model predicts the bubble length closest to the experiments. The separation point is early.
- The Low-Re $k-\omega$ model predicts a far too big recirculation zone, it starts far too earlier, and ends far too late.

The results are shown in figures 28, 29, and 30.

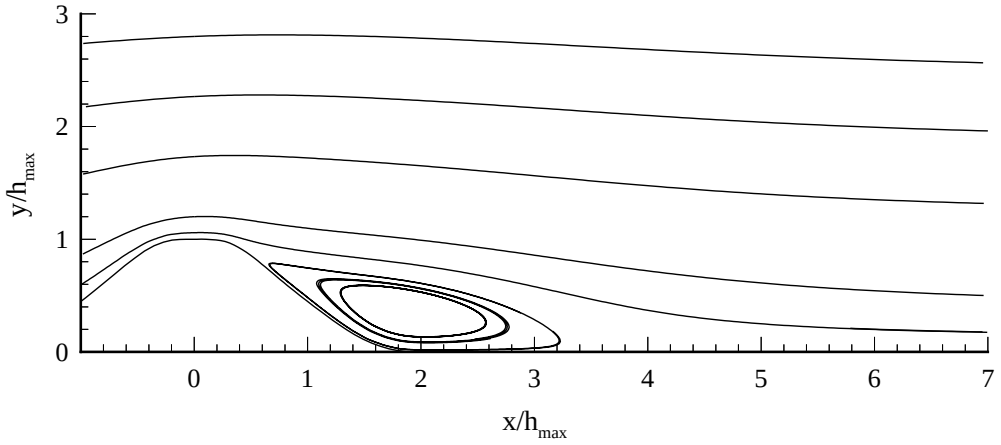


$k - \varepsilon$ model

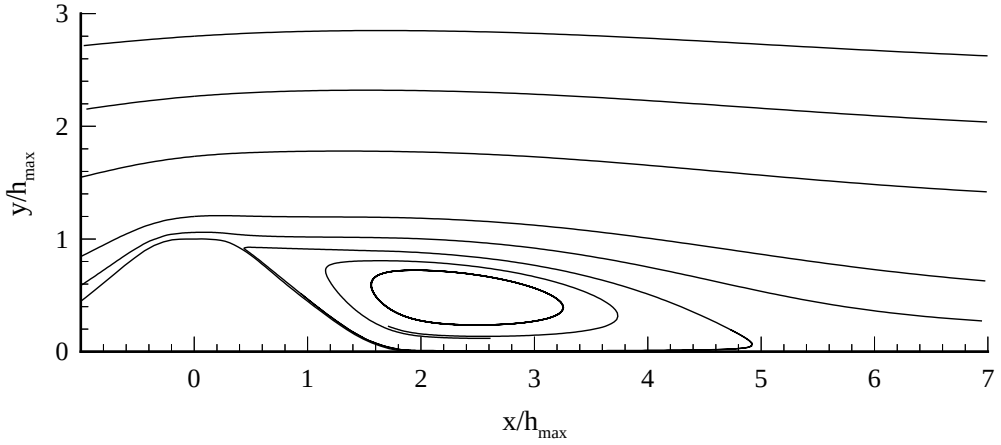


$k - \omega$ model

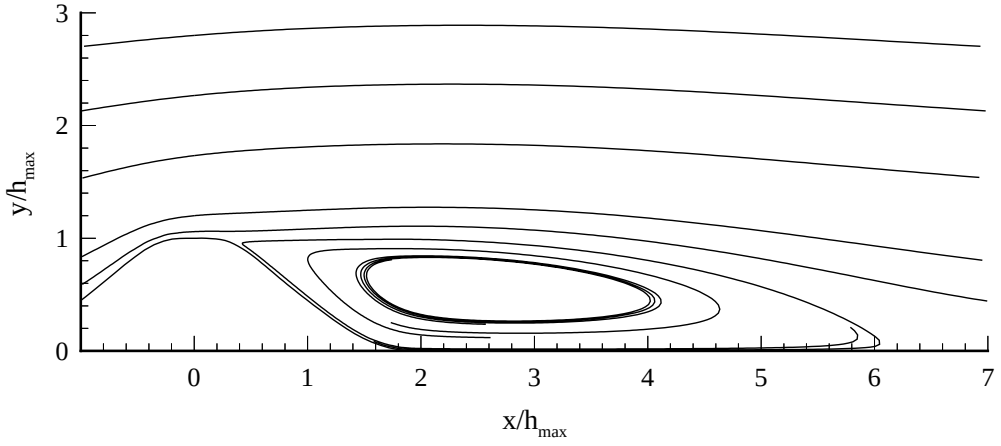
Figure 23: Streamlines plot of the recirculation zone, using High-Re models



Two layer $k - \varepsilon$ model



Low Re $k - \varepsilon$ model



Low Re $k - \omega$ model

Figure 24: Streamlines plot of the recirculation zone, using Low-Re models

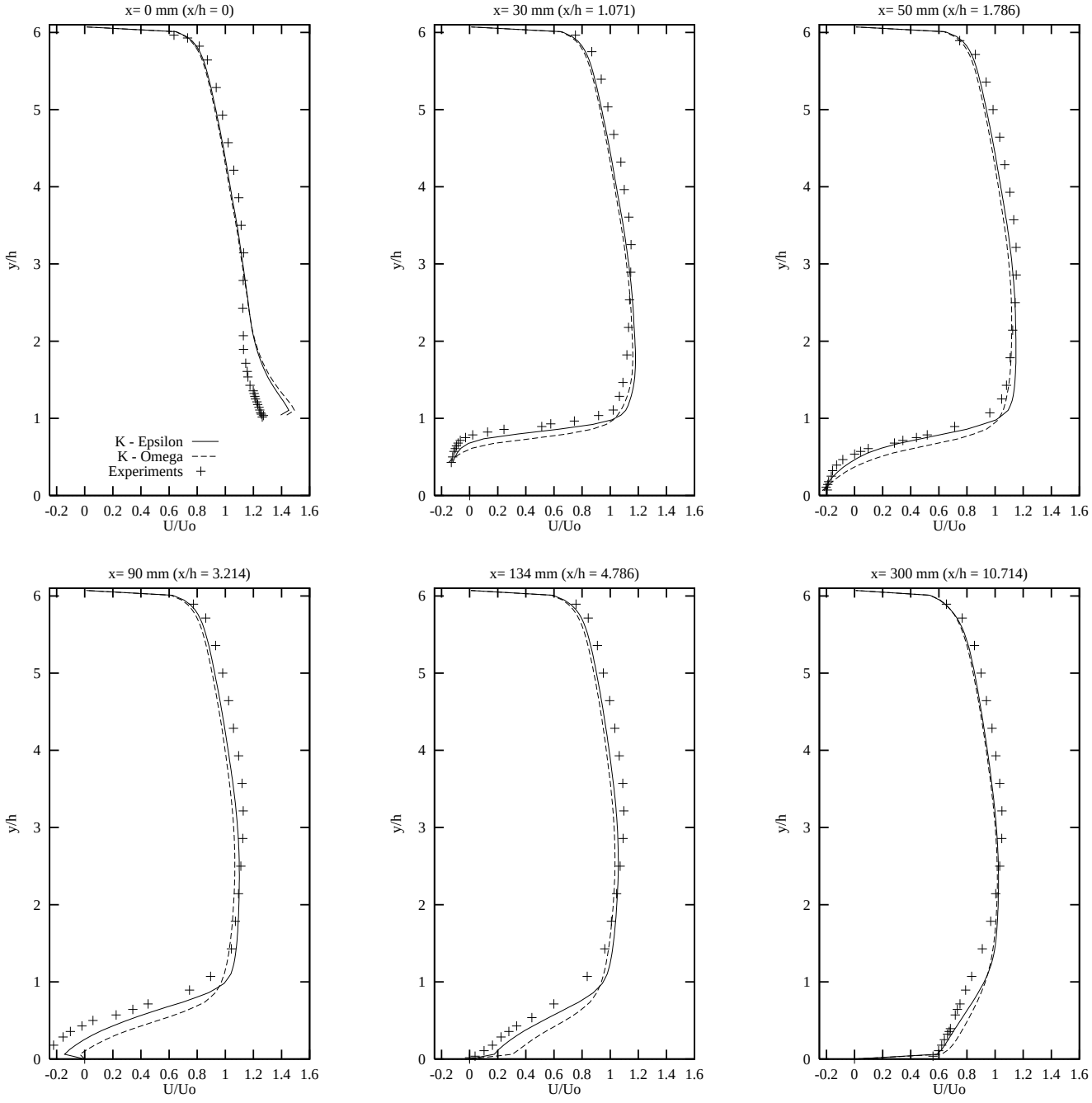


Figure 25: U mean velocity profiles with the different wall function based models.

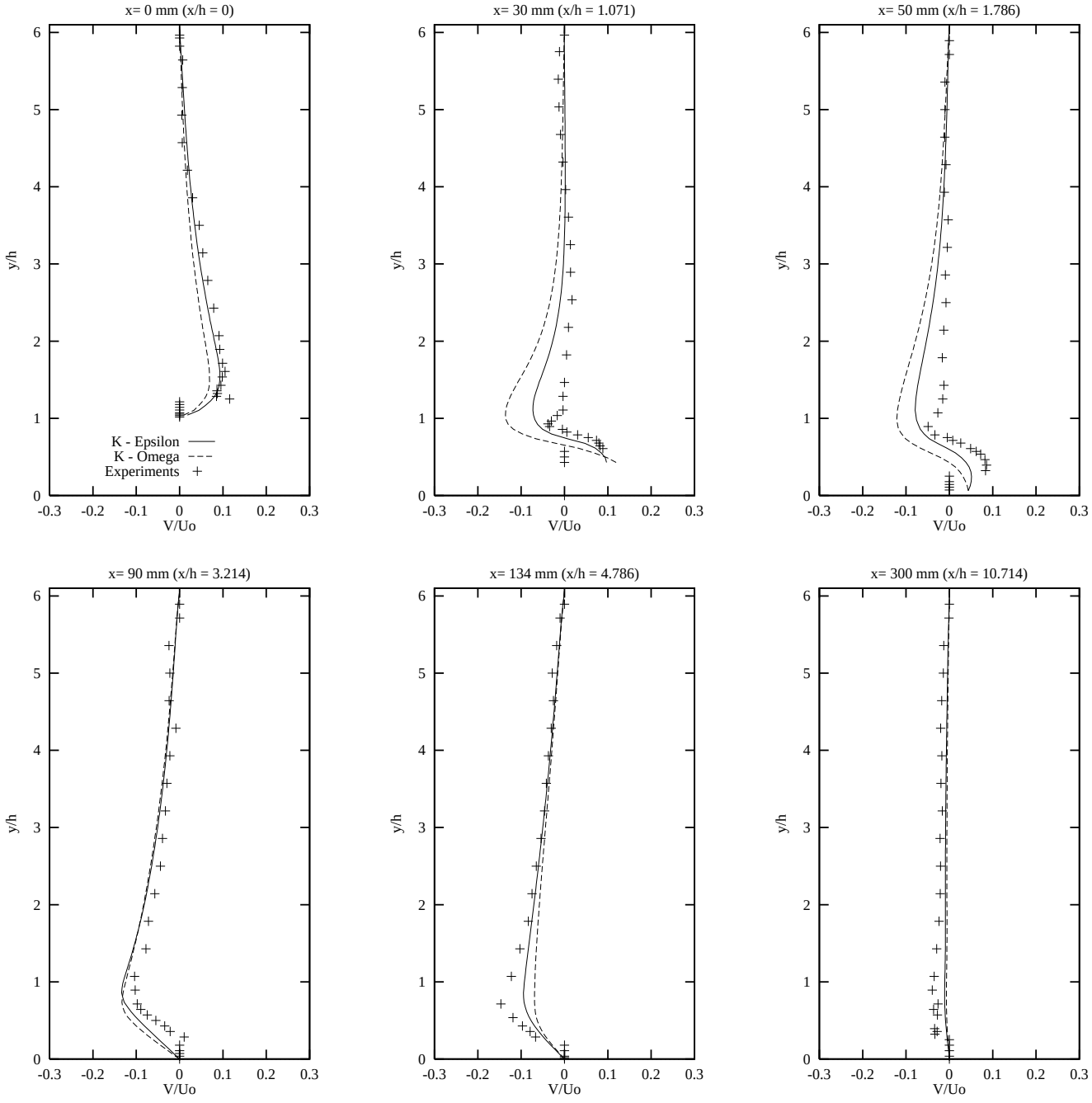


Figure 26: V mean velocity profiles with the different wall function based models.

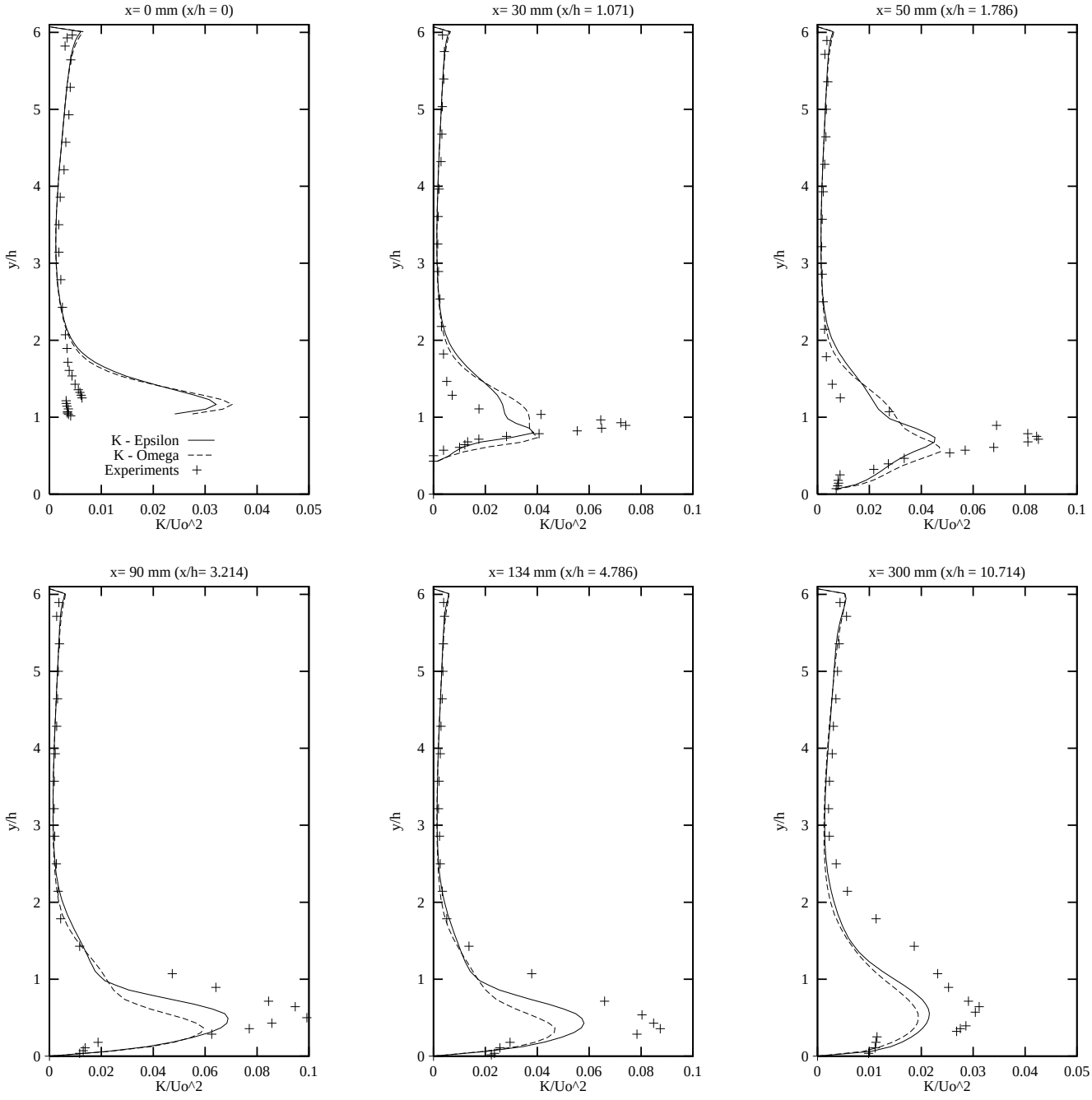


Figure 27: Turbulent kinetic energy profiles with the different wall function based models.

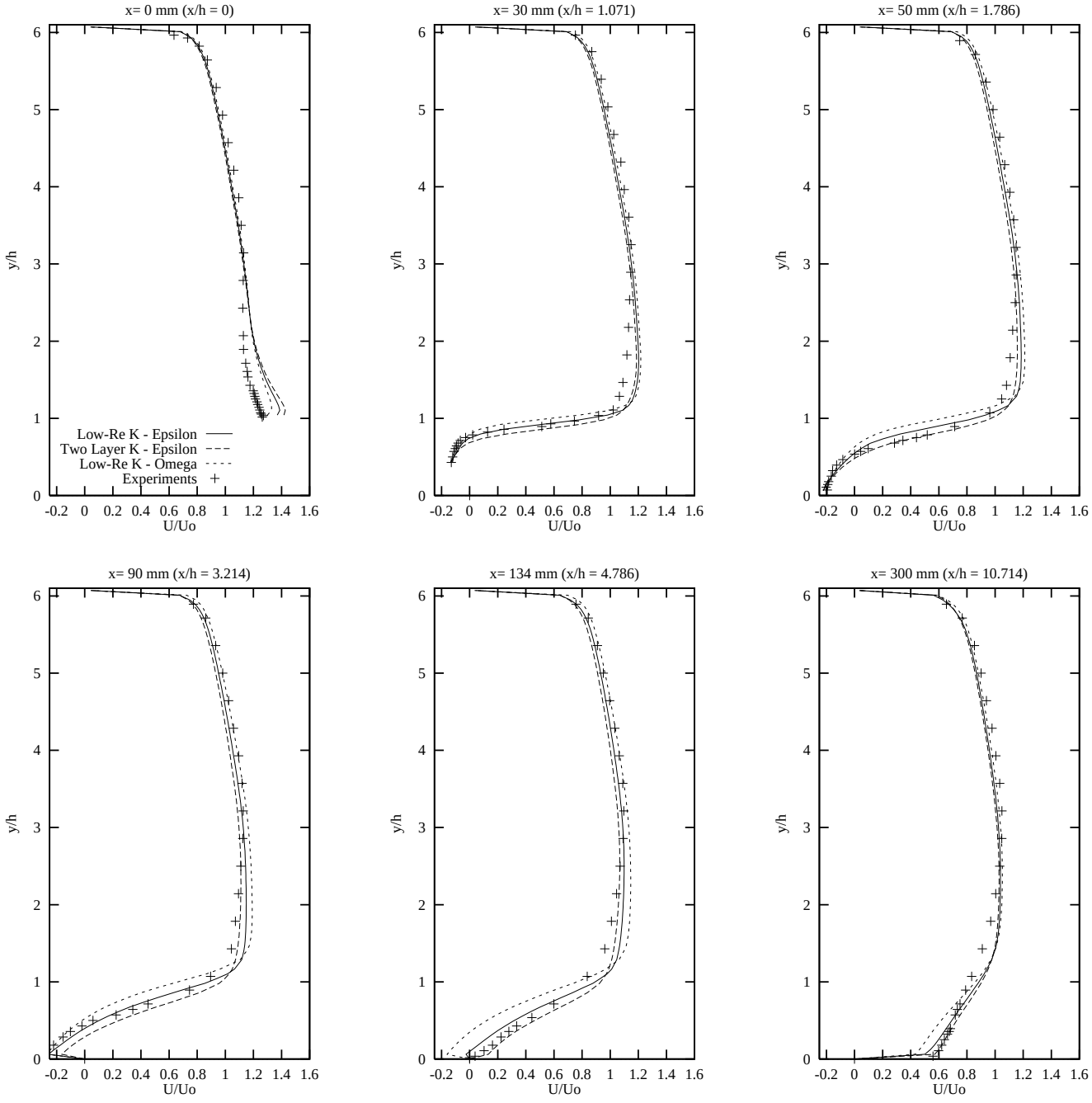


Figure 28: U mean velocity profiles with the different low Reynolds number models.

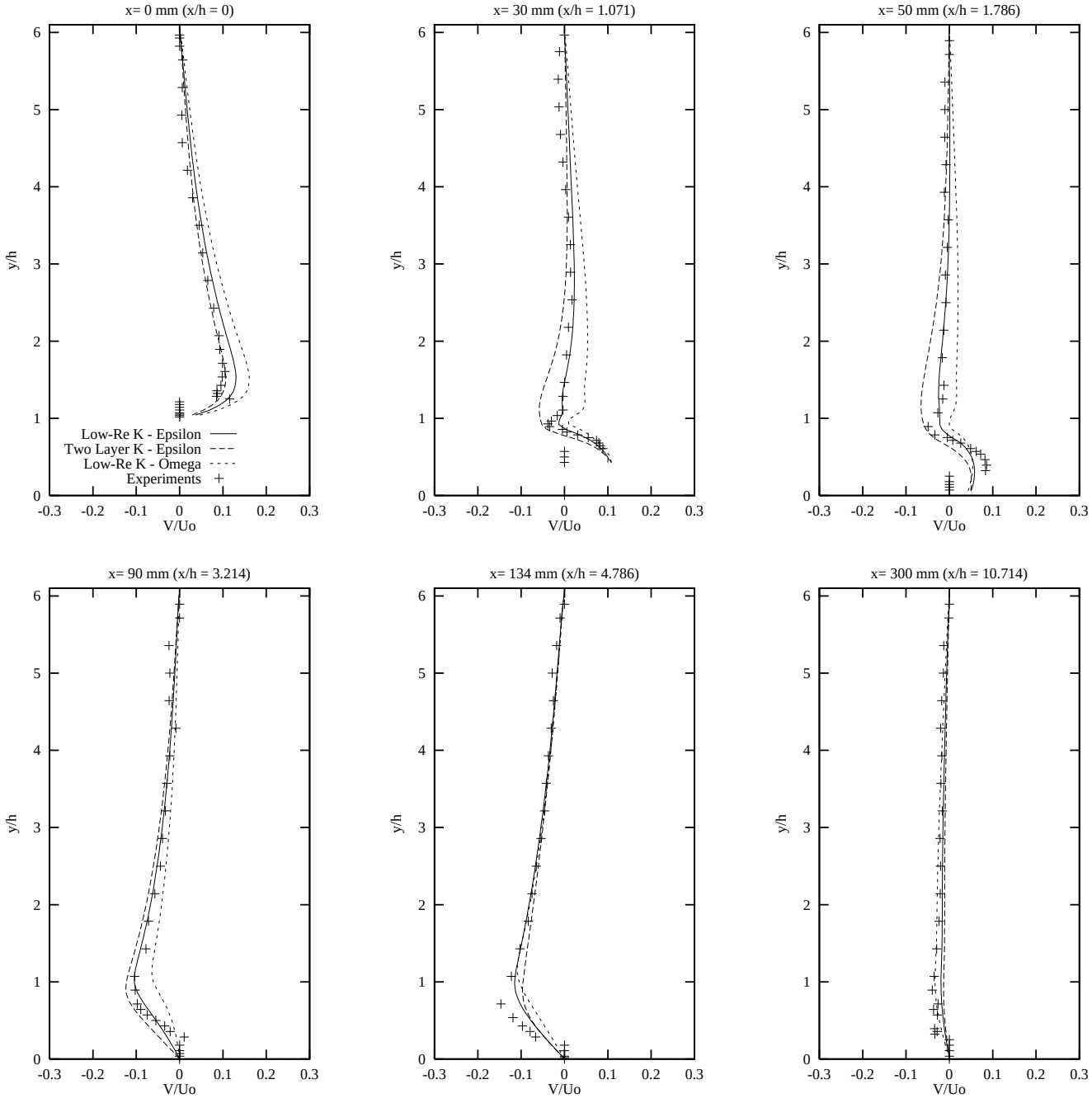


Figure 29: V mean velocity profiles with the different low Reynolds number models.

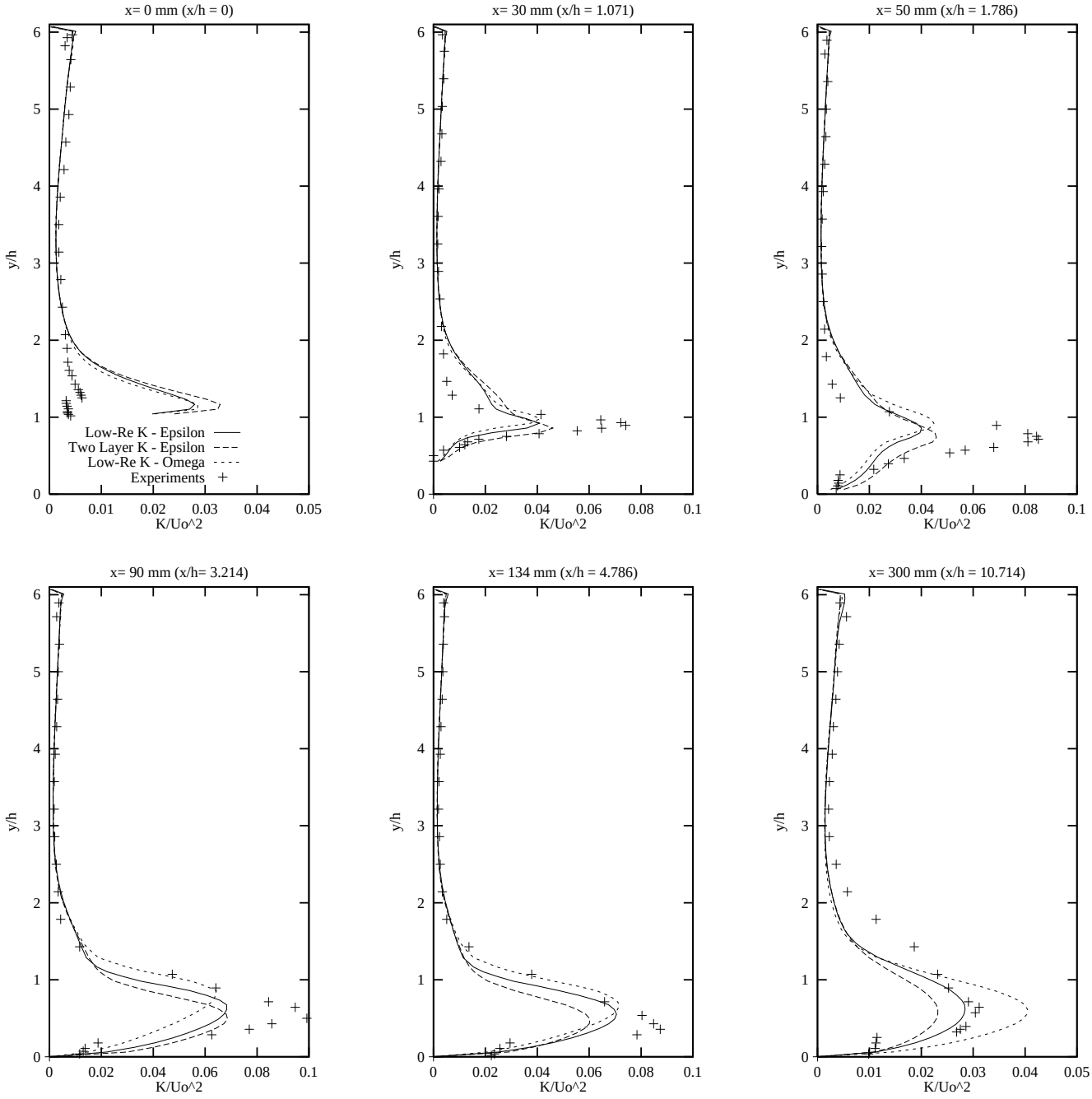


Figure 30: Turbulent kinetic energy profiles with the different low Reynolds number models.

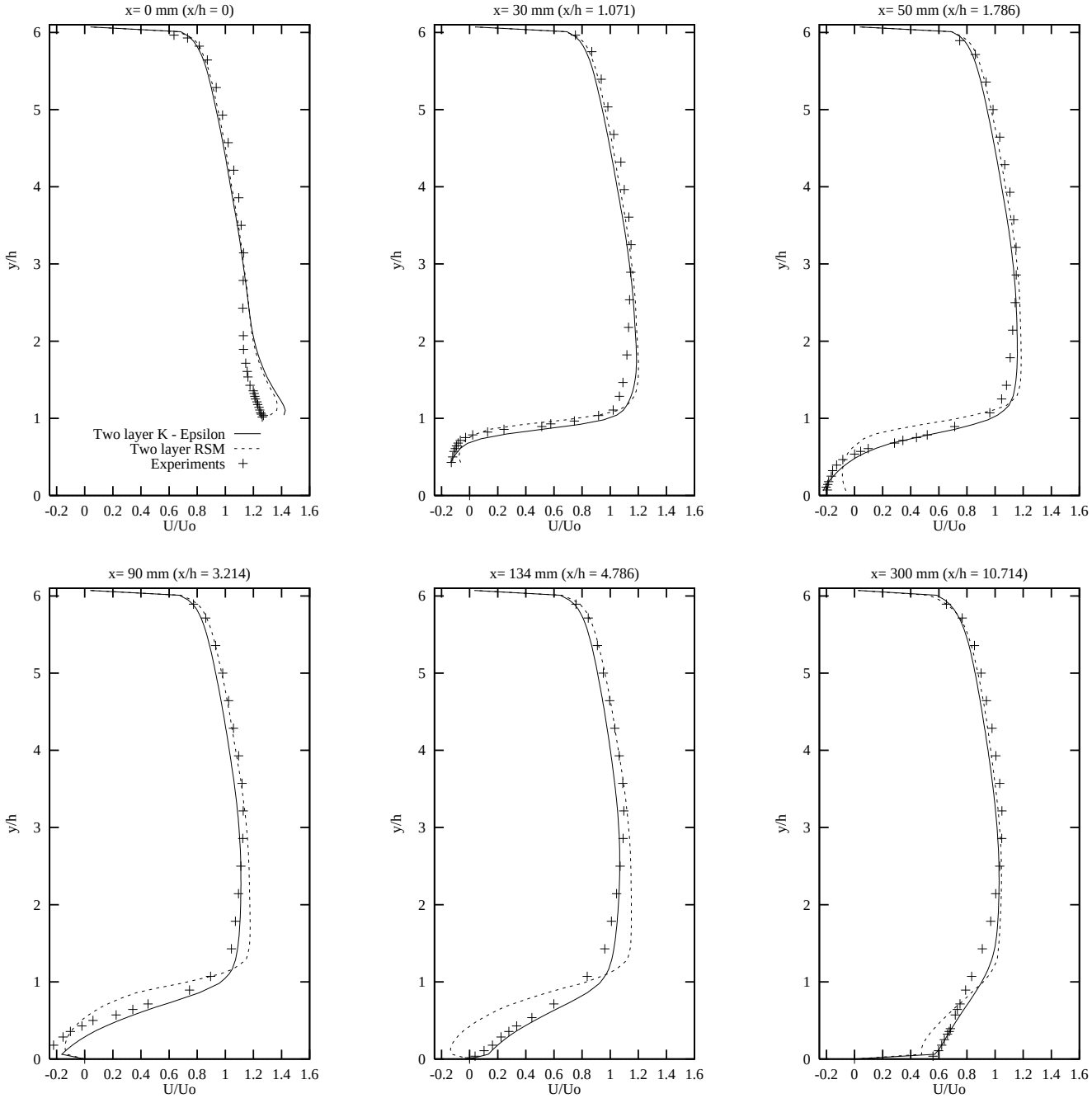


Figure 31: U mean velocity profiles with RSM.

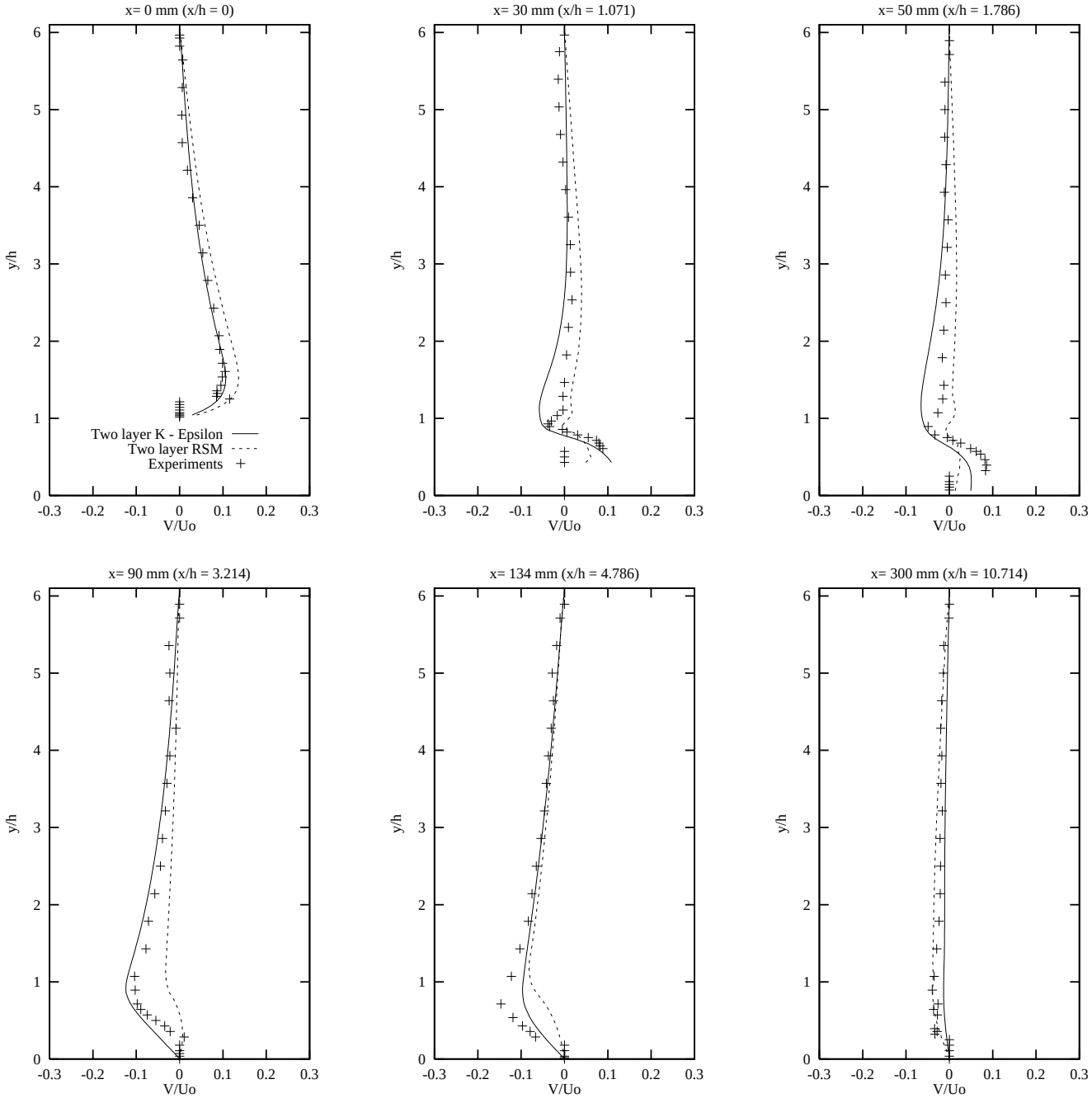


Figure 32: V mean velocity profiles with RSM.

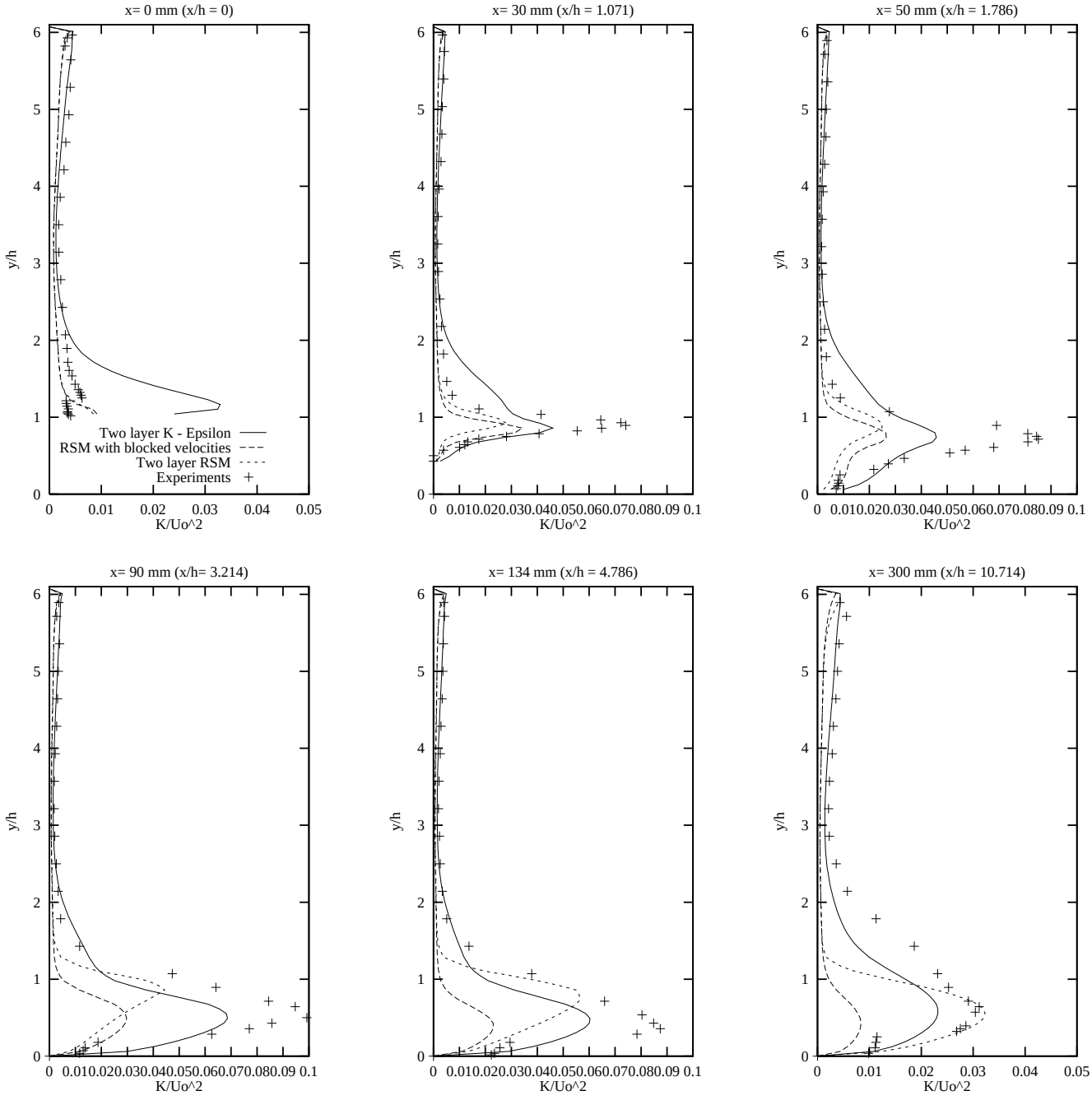


Figure 33: Turbulent kinetic energy profiles with RSM.

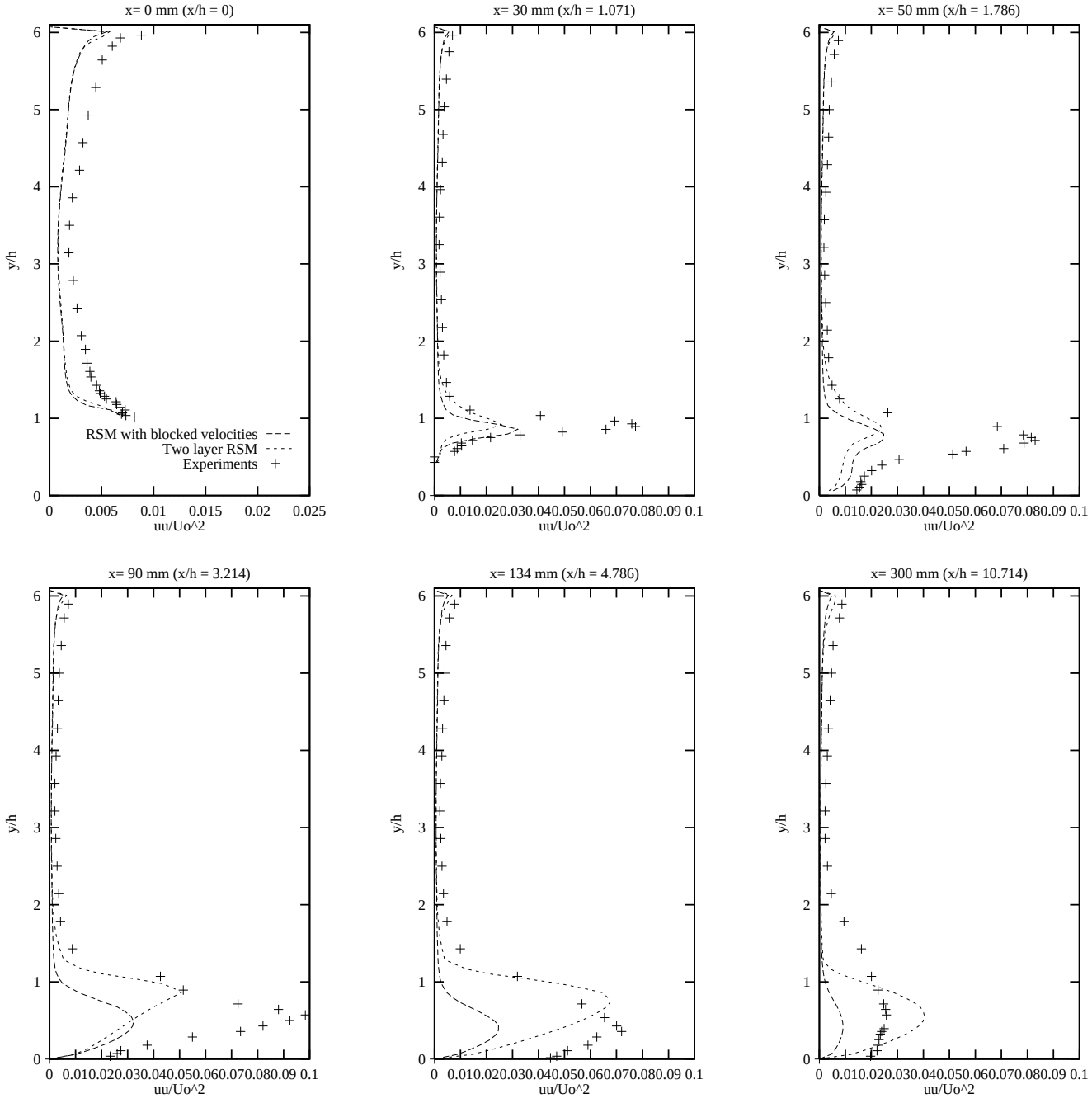


Figure 34: $\overline{u^2}$ Reynolds stress profiles with RSM.

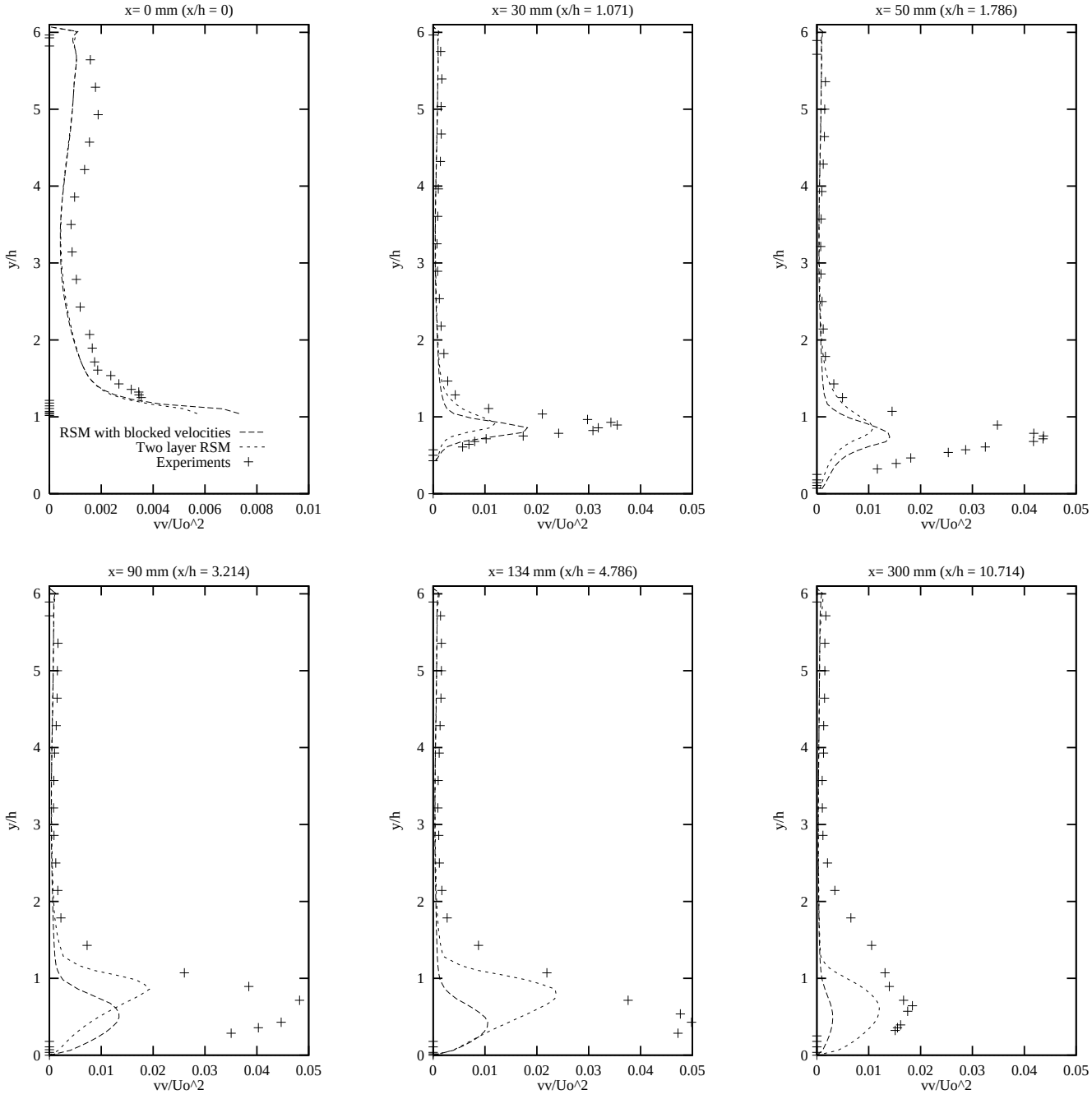


Figure 35: $\overline{v^2}$ Reynolds stress profiles with RSM.

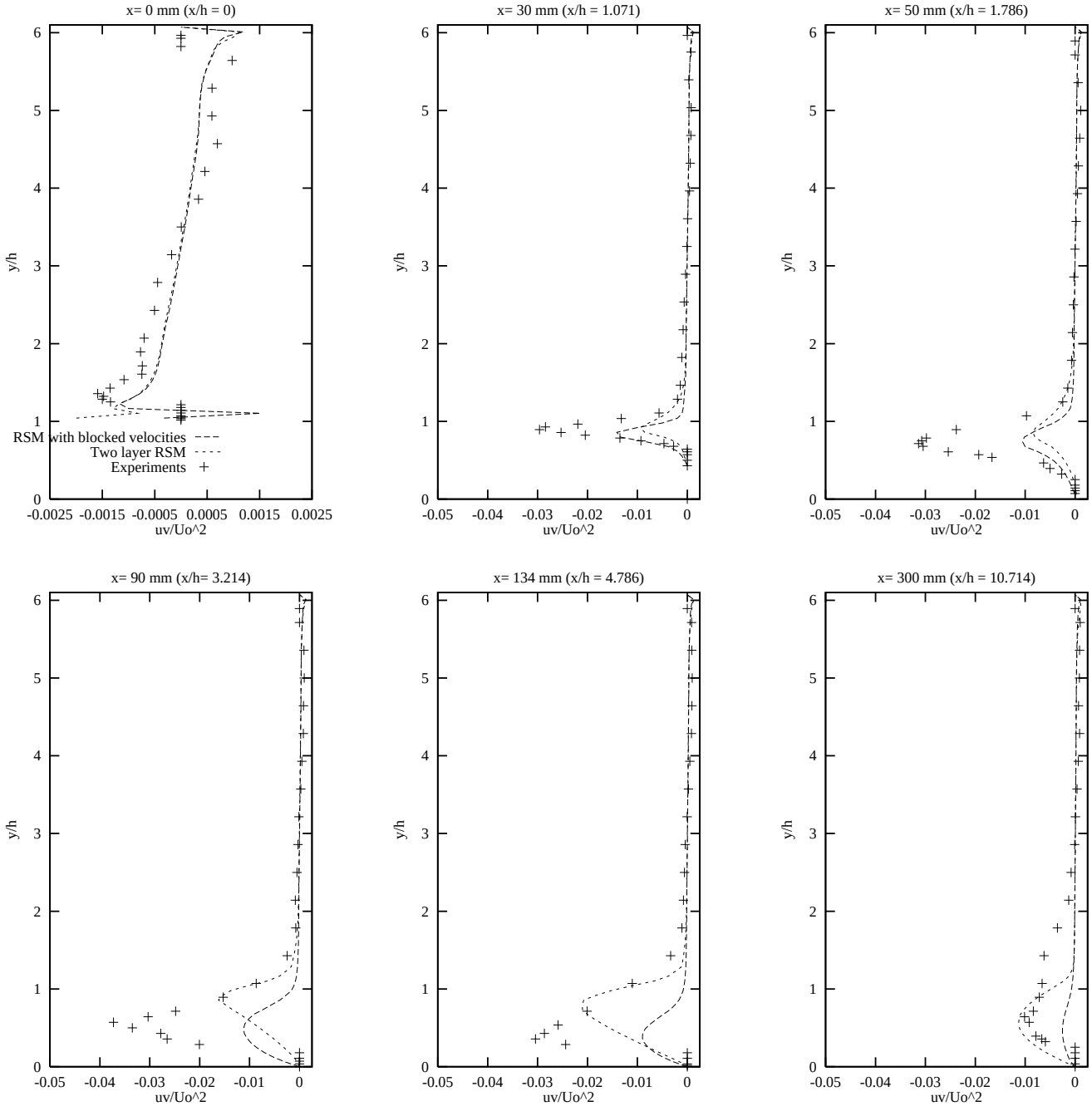


Figure 36: $\overline{uv}$ Reynolds stress profiles with RSM.

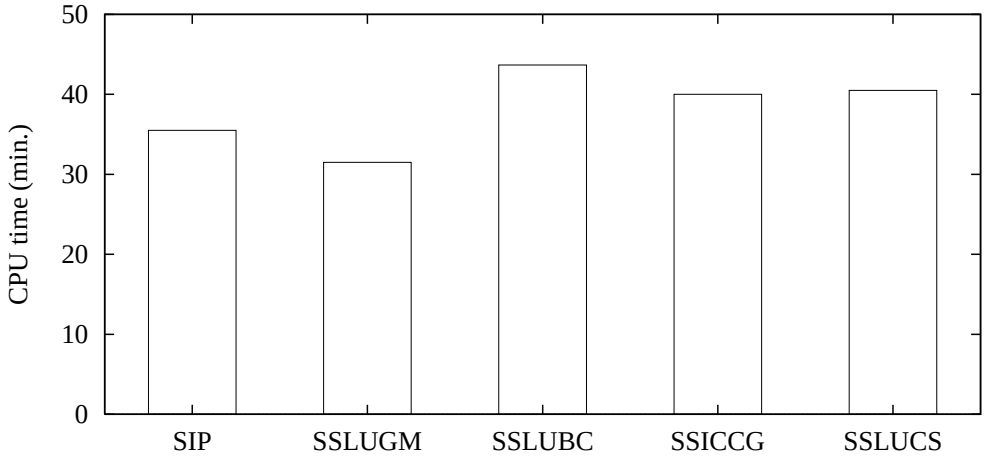


Figure 37: Computation time applying different solvers on the continuity equation.

3.5 Compared performances using different solving procedures

To test the different solvers, the Low-Re $k-\varepsilon$ model has been chosen along with the coarse mesh (64×50), to save computation time.

- Continuity equation.

  In fact, it is worth paying a special attention to this problem since the correction of the pressure is used by the three different velocities.

  It should be noticed here that all methods require more CPU time than SIP, except with the Generalized Minimal RESidual solver which required less iterations and which converged fatser.

- $k-\varepsilon$ coupled

  A change in the relaxation factors for k and ε was required to achieve convergence. A common value of 0.22 seemed to be an optimum one. Nevertheless one solver: SSDBCG did not work at all. SSLUGM was used on the Pressure.

  Note that SIP stands for comparison purposes, and is not coupled at all.

  All solvers shown here required the same amount of iterations, and about comparable CPU time as for the computation of the Pressure correction Equation, which makes sense since the residuals on k and ε are not really a limiting factor.

- Finer mesh (128×100 cells)
  The results are not as good as expected: the solver which gave the best results in

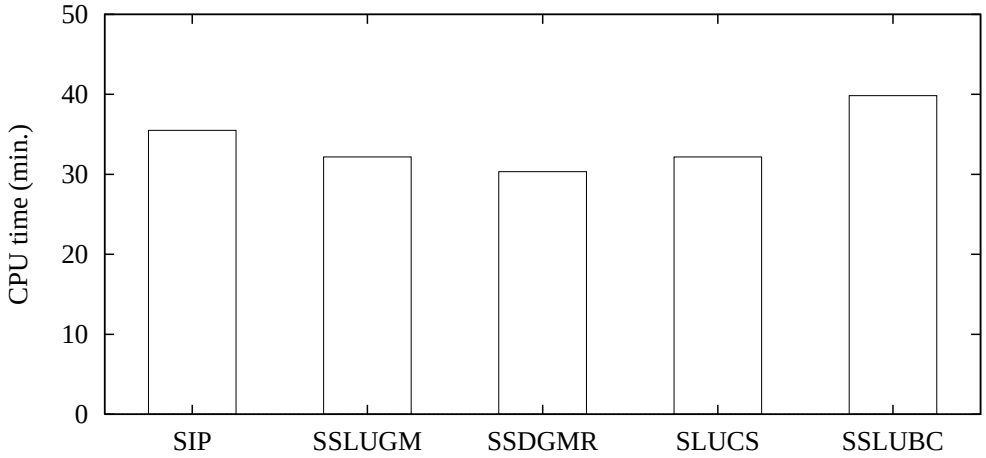


Figure 38: Computation time applying different solvers on the coupled resolution of k and ε.

the other cases (SSLUGM) does not converge for the continuity equation! A solver specialized on symmetric matrixes is then required.

The following results were found:

Configuration	CPU time	Iterations
Standard case (SIP)	4h	3140
Continuity equation, using **Conjugated Gradient (Incomplete Choleski)**	6h40	2500
$k-\varepsilon$ **coupled**, using Generalized Minimum residuals (Incomplete LU)	4h40	3400

The number of iterations needed to solve the continuity equation has been reduced, using the conjugated gradient algorithm, but since the iterations are slower, the gains disappear.

Concerning $k-\varepsilon$ coupled equation, one can notice that more iterations are required; it tends to show that this procedure is more unstable.

Conclusion

All the factors studied here had a relative influence on the results; only few combinations may surely lead to accurate results.

- Discretization schemes:
  Hybrid should only be used to provide sets of initial values. The use of an high order scheme on the turbulent quantities seems useless.

- Meshes:
  The results depend strongly on the mesh resolution, but also on the mesh refinements. Special care must be taken when the mesh is generated.

- Model:
  High-Re models are here less accurate, and difficult to use, since y^+ must lie in a very small interval.

  Low-Re models give much better results, but are also much more CPU consuming (a Low-Re model requires a very refined mesh close to the walls, and therefore much more cells).

  RSM seems very promising, and will become more and more important, in the future.

- Solvers:
  No significant improvements have been noticed using a general purpose solver on the pressure correction equation.

  The concept of coupled resolution of inter-dependent quantities seems also very promising. The coupled resolution studied here involved only two coupling terms. Numerical simulations of buoyant flows may be significantly improved by coupling the mean velocities along with the temperature and the pressure correction (i.e five variables coupled).

References

[1] M. V. ALMEIDA, D.F.G. DURAU and M.V. HEITOR, Wake Flows Behind Two-Dimensional Hills, *Exp. Thermal and Fluid Science*, Vol. 7, pp. 87-101, 1993.

[2] L. DAVIDSON and B. FARHANIEH, CALC-BFC: a Finite-Volume Code Employing Collocated Variable Arrangement and Cartesian Velocity Components for Computation of Fluid Flow and Heat Transfer in Complex Three-Dimensional Geometries', *Publication 91/14*, Dept. of Thermo- and Fluid Dynamics, Chalmers University of Technology, Göteborg, Sweden, 1991.

[3] H.C. CHEN and V.C. PATEL, Near-Wall Turbulence Models for Complex Flows Including Separation, *AIAA J.*, Vol. 26, pp. 641-648, 1988.

[4] F.S. LIEN and M.A. LESCHZINER, Assessment of Turbulence-Transport Models Including Non-Linear RNG Eddy-Viscosity Formulation and Second-Moment Closure for Flow Over a Backward-Facing Step, *Computers & Fluids*, Vol. 23, pp. 983-1004, 1994.

[5] D.C. WILCOX, Reassessment of the Scale-Determining Equation for Advanced Turbulence Models, *AIAA J.*, Vol. 26, pp. 1299-1309, 1988.

[6] J.C. NASH, *Compact Numerical Methods for Computers: Linear algebra and function minimization* – 2nd ed., Adam Hilger, Bristol, England, 1990.

[7] G. STRANG, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Cambridge, USA, 1986.

[8] S.V. PATANKAR, *Numerical Heat Transfer and Fluid Flow*, Mc Graw-Hill, Washington, 1980.

[9] H.-J. LEISTER and M. PERIC, Vectorized Strongly Implicit Solving Procedure for Seven-Diagonal Coefficient matrix, Lehrstuhl für Strömungsmechanik, Universität Erlangen-Nürnberg, Erlangen, Germany.

[10] S. JANSSON, Turbulence Modeling of Flows Related to Wall-Cooling Applications, *PhD Thesis*, Dept. of Thermo- and Fluid Dynamics, Chalmers University of Technology, Göteborg, Sweden, 1994.

[11] B. E. LAUNDER and N. SHIMA, Second-Moment Closure for the Near-Wall Sublayer: Development and Application, *AIAA J.*, Vol. 27, pp. 1319-1325, 1989.

[12] B. FARHANIEH, L. DAVIDSON and B. SUNDEN, Employment of Second-Moment Closure for Calculation of Turbulent Recirculating Flows in Complex Geometry with Collocated Variable Arrangement, *Int. Journal for Numerical Methods in Fluids*, Vol. 16, pp. 525-544, 1993.

[13] S. PERZON, L. DAVIDSON and M. RAMNEFORS, Reynolds Stress Modeling of Flow Separation on Curved Surfaces. A Status Report, *Internal Report Nr 95/12*, Dept. of Thermo- and Fluid Dynamics, Chalmers University of Technology, Göteborg, Sweden, April 1995.

Appendix

A List of computations

The following abbreviations are used:
h = Hybrid
v = Van-leer
q = QUICK
The first letter indicates the scheme used on the mean velocities, while the second indicates the scheme used to discretize the turbulent quantities.

Wall function based models

Mesh:	74 × 25 cells					78 × 49 cells				
Scheme:	h-h	v-h	v-v	q-h	q-v	h-h	v-h	v-v	q-h	q-v
basic $k-\varepsilon$	×				×	×				×
basic $k-\omega$	×				×					

Mesh:	122 × 50 cells					148 × 50 cells				
Scheme:	h-h	v-h	v-v	q-h	q-v	h-h	v-h	v-v	q-h	q-v
basic $k-\varepsilon$	×			×	×	×			×	×
basic $k-\omega$	×				×	×			×	×

Low-Re models

Mesh:	64 × 50 cells					128 × 100 cells				
Scheme:	h-h	v-h	v-v	q-h	q-v	h-h	v-h	v-v	q-h	q-v
Low-Re $k-\varepsilon$	×			×	×	×		×	×	×
Two layer $k-\varepsilon$	×			×	×	×		×		×
Low-Re $k-\omega$										×
Two layer RSM	×					×		×		

B Automatized processes and tools

Data handling task, one of the most repeated procedure, can be highly automatized thanks to script files and macro commands.

This approach provide interesting features, such as:

- Reduce the amount of work.
- Save time (users and CPU time).
- Avoid some file handling mistakes.

Here are the three main scripts used during this work:

showres plots the residual values evolution. A key element to appreciate the convergence level of a computation.

This macro simply extracts the residual values for each variable, and plots the data obtained using GNUPLOT.

probe uses tec-plot to extract automatically a linear probe, and save it in ASCII format. The data-file obtained contains the values for all the variables available, interpolated on the chosen line.

The data can be plotted with GNUPLOT or MATLAB to compare the simulation results with experiments or with other results (or both).

build This script has been used to submit all the data to the workshop. The ER-COFTAC Workshop required a specific format for the computation results. The "heavy" structure used is in fact very convenient, and allow a complete automatization of the data conversion.

Main features of the format specifications:

- non-dimensionned data.
- identification of the data files using a generic name and location.
- only 2 columns of data in each file.

Data submitted by the script: velocities and turbulent quantities profiles at all the required locations, in only one operation (12×3 files generated, 12×6 using a Reynolds stress model).

Script description:

1) create a temporary non-dimensionned tecplot data file.
2) extract temporary probe files.
3) convert the probe files to ascii.
4) extract the usefull data from the probe, to fully identified temporary files.
5) build the final data files with data and comments, using the name generation procedure.

C SLAP Column-Format code

```
       subroutine slapsolver(nphi)
c*****************************************
c this routine transform the calc matrixes *
c into slap column format                  *
c*****************************************

       include 'common.f'
       parameter(n=(it-2)*(jt-2)*(kt-2),nworki=250000,nworkr=250000)
       dimension b(n),x(n),a(7*n), rwork(nworkr)
       dimension iwork(nworki),ia(7*n),ja(n+1)
       nr=(ni-2)*(nj-2)*(nk-2)


c      i1=position of ant element in a
c      i2=position of the element p

       i1=1
       i2=0
       ja(1)=1
       do 10 k=2,nkm1
         do 10 j=2,njm1
          do 10 i=2,nim1
           i2=i2+1

           a(i1)=ap(i,j,k)
           ia(i1)=i2
           b(i2)=su(i,j,k)
           i1=i1+1

           if (k.ne.2) then
              a(i1)=-ah(i,j,k-1)
              ia(i1)=i2-(ni-2)*(nj-2)
              i1=i1+1
           else
               b(i2)=b(i2)+al(i,j,k)*phi(i,j,k-1,nphi)
           endif


           if (j.ne.2) then
               a(i1)=-an(i,j-1,k)
                ia(i1)=i2-ni+2
```

```
            i1=i1+1
          else
            b(i2)=b(i2)+as(i,j,k)*phi(i,j-1,k,nphi)
          endif


          if (i.ne.2) then
            a(i1)=-ae(i-1,j,k)
            ia(i1)=i2-1
            i1=i1+1
          else
            b(i2)=b(i2)+aw(i,j,k)*phi(i-1,j,k,nphi)
          endif

          if (i.ne.nim1) then
            a(i1)=-aw(i+1,j,k)
            ia(i1)=i2+1
            i1=i1+1
          else
            b(i2)=b(i2)+ae(i,j,k)*phi(i+1,j,k,nphi)
          endif

          if (j.ne.njm1) then
            a(i1)=-as(i,j+1,k)
            ia(i1)=i2+ni-2
            i1=i1+1
          else
            b(i2)=b(i2)+an(i,j,k)*phi(i,j+1,k,nphi)
          endif

          if (k.ne.nkm1) then
            a(i1)=-al(i,j,k+1)
            ia(i1)=i2+(ni-2)*(nj-2)
            i1=i1+1
          else
            b(i2)=b(i2)+al(i,j,k)*phi(i,j,k+1,nphi)
          endif

        ja(i2+1)=i1

      x(i2)=phi(i,j,k,nphi)
10    continue
```

```
        nelt=i1-1
        itol=2
        tol=1.e-5
        if (nphi.eq.pp) then
            itol=0
            tol=0.5
c     tol=0.5 seems to be an optimum value. if you decrease it, you will
c     make more work on PP (then decreasing the total number of
c     iterations, but losing much time in the Slap routine)
c     increasing this value may make the system diverge

        endif
        isym=0
        itmax=nsweep(nphi)
        iunit=6
        nsave=10
        lenw=(7+nsave)*n+nelt+(nsave+3)*nsave+2
        leniw=7*n+11+nelt

c for PP, a solver for symmetric systems is recommanded for fine meshes
c          SSLUGM can be used on coarse meshes
c for other variables, SSDCGS seems to be the best choice
c                    SSDCGS > SSDGMR > SSLUCS > SSLUGM

        if (nphi.eq.pp) then
        call sslugm(nr,b,x,nelt,ia,ja,a,isym,nsave,itol,tol,itmax,ite,
     1             err,ierr,iunit,rwork,lenw,iwork,leniw)

        else
        call ssdcgs(nr,b,x,nelt,ia,ja,a,isym,itol,tol,itmax,ite,
     1             err,ierr,iunit,rwork,lenw,iwork,leniw)

        endif
        j2=0
        do 20 k=2,nkm1
          do 20 j=2,njm1
            do 20 i=2,nim1
              j2=j2+1
 20           phi(i,j,k,nphi)=x(j2)

c------ min & max values of PHI
      do 310 i=1,ni
      do 310 j=1,nj
```

```
      do 310 k=1,nk
        phi(i,j,k,nphi)=min(max(phi(i,j,k,nphi),phimin(nphi)),
     .                                          phimax(nphi))
310   continue

        return
        end
```