# Optimal design of fluid flow using subproblems reduced by large eddy simulation

Argus Adrian Dunca

October 23, 2003

**Abstract**

This report presents tests of a 2-dimensional parallel PETSc code developed while I was a Givens fellow in residence at ANL, code which computes the fluid velocity and pressure in a given domain subject to given inflow and outflow boundary conditions. The code provides an approximation to the solution of the Navier-Stokes equations, a classical Large Eddy Simulation model (Smagorinsky's model) as well as a new LES model which was tested with good results(see section 4 and 5) and whose mathematical analysis is currently under study. I also wrote and tested a code to compute the total kinetic energy of the flow and the total vorticity (circulation per unit area) of the flow. Finally, I ran numerical tests to illustrate the value of using LES as a reduced order model in shape optimization.

## 1 General Methodology

To compute the fluid velocity and pressure on a domain $\Omega$ the Navier Stokes equations [2] which model the motion of an incompressible viscous, Newtonian fluid

$$\begin{aligned} u_t - \nu\Delta u + (u\cdot\nabla)u + \nabla p &= f \quad \text{in } (0,T)\times\Omega \\ \nabla\cdot u &= 0 \quad \text{in } (0,T]\times\Omega \\ u &= g \quad \text{in } (0,T]\times\partial\Omega \\ u\mid_{t=0} &= u_0 \quad \text{in } \Omega \\ \int_\Omega p\mathrm{d}x &= 0 \quad \text{in } (0,T]. \end{aligned} \tag{1}$$

need to be solved numerically.

A reference mesh is selected on the reference configuration (in our tests, the unit square). The computational mesh is the image of the reference mesh under the transformation mapping the reference domain to the physical domain. In our tests the reference mesh was selected to be uniform.

The Navier Stokes equations are discretized then in space with $P2/P1$ (Taylor-Hood) elements with a uniform mesh on the unit square leading to the semidiscrete problem :

Find $u : [0,T] \to P2 \times P2, p : [0,T] \to P1$ such that

$$\begin{array}{rcll} (u_t, v) + \nu(\nabla u, \nabla v) + ((u \cdot \nabla)u, v) + (\nabla p, v) & = & (f, v) & \text{in } (0,T) \\ (\nabla \cdot u, q) & = & 0 & \text{in } (0,T] \\ u & \approx & g & \text{in } (0,T] \times \partial\Omega \\ u\mid_{t=0} & \approx & u_0 & \text{in } \Omega \end{array} \tag{2}$$

for every $(v, q) \in P2 \times P2 \times P1$.

If $v = (v_1, .., v_k), w = (w_1, .., w_k) \in L^2(\Omega)^k$ we denote by $(v, w)$ the canonical inner product in $L^2(\Omega)^k$, that is:

$$(v, w) = \Sigma_{i=1..k} \int_\Omega v_i w_i \mathrm{d}x.$$

To discretize in time first the equations above are linearized and a variant of the Crank-Nicholson method, that uses only two linear systems per step, is used to advance in time.

At time level $t_n$ our approximation to the continuous solution is denoted by $u_n$.

At time level $t_{n+1}$ we first find $u^{(1)} \in P2 \times P2, p \in P1$ such that

$$\begin{array}{rcll} (\frac{(u^{(1)}-u_n)}{\Delta t}, v) + \nu(\nabla \frac{u^{(1)}+u_n}{2}, \nabla v) + 0.5 * ((u_n \cdot \nabla)u^{(1)}, v) + & & & \\ +0.5 * ((u_n \cdot \nabla)u_n, v) + (\nabla p, v) & = & (f, v) & \\ (\nabla \cdot u^{(1)}, q) & = & 0 & \\ u^{(1)} & \approx & g(t_{n+1}, \cdot) & \text{in } \partial\Omega \end{array} \tag{3}$$

for every $(v, q) \in P2 \times P2 \times P1$,

Next replace $u_n$ by $u^{(1)}$ when doing the linearization. and solve the linear system for $(u_{n+1}, p)$

$$\begin{array}{rcll} (\frac{(u_{n+1}-u_n)}{\Delta t}, v) + \nu(\nabla \frac{u_{n+1}+u_n}{2}, \nabla v) + 0.5 * ((u^{(1)} \cdot \nabla)u_{n+1}, v) + & & & \\ +0.5 * ((u_n \cdot \nabla)u_n, v) + (\nabla p, v) & = & (f, v) & \\ (\nabla \cdot u_{n+1}, q) & = & 0 & \\ u_{n+1} & \approx & g(t_{n+1}, \cdot) & \text{in } \partial\Omega \end{array} \tag{4}$$

Using Richardson extrapolation we can show numerically that the method is order 2 in time. Analitical verification of second order accuracy is still an open question and is currently under study. Because we do not impose the condition that the mean of the pressure be 0, the matrices of the linear sytems that are to be solved at every time step are singular. In our code in order to solve the linear system we use a nullspace reduction method [1, Chapter 4.5].

The Dirichlet boundary conditions were imposed using the command MatZeroRows of PETSC [1, page 58]. The entries corresponding to the velocity nodes on the boundary all zeros except the diagonal entry which was set to be 1.

The load vector of the linear system was modified accordingly by setting every entry corresponding to a velocity node on the boundary to be equal to the value of g at that node.

The numerical scheme outlined above attempts to compute an approximation of the fluid's velocity by discretizing the Navier Stokes equations. This approach is called direct numerical simulation(DNS) and it attempts to approximate the exact solution with respect to all scales. To approximate only flow structures of size $\delta$ or more one may use the Smagorinsky model in which the term $\nabla \cdot \nu_T \nabla u$ where $\nu_T = (C\delta)^2|0.5 * (\nabla u + \nabla^s u)|$, is substracted from the first equation of 1. The nonlinear term $\nu_T \Delta u$ from the Smagorinsky model was implemented in our code by using the same linearization technique which was used to model the nonlinear convective term $(u \cdot \nabla)u$. Another LES model which was implemented in the code makes use of differential filters to correct the computed solution at each time step. This model is explained in more detail in Chapter 4.

We have used PETSc built-in linear solver TFQMR with an additive Schwarz as a preconditioner to solve the linear system wich arise when doing the discretizations 3 and 6 .

At each time level $t_n$ the approximation $u_n$ found as above was used to compute energy and $L^2$ norm of vorticity at time $t_n$ :

$$energy\ at\ time\ t_n = \int_\Omega |u_n|^2 \mathrm{d}x$$

$$L^2\, norm\, of\, vorticity\ at\ time\ t_n = \int_\Omega |\frac{\partial u_n^1}{\partial x_2} - \frac{\partial u_n^2}{\partial x_1}|^2 \mathrm{d}x$$

where $u = (u^1, u^2)^T$. In the code the integrals above were computed by using a change of variables to go from the domain to the unit square and then looping over all elements, changing variables to set the integral on the reference element and integrating there with a quadrature formula of order 7.

Then a linearized version of the trapezoidal method, that uses only 2 linear systems per step was used to integrate in time and get total energy and total vorticity.

## 2 The Mesh

To build a finite element code which solves the Navier Stokes Equations (NSE, [2]) 1 one needs to write:

1. A routine which creates the mesh.After a call to this routine all the information regarding the mesh which is needed to build the assembly matrix and the load vector is proccesed and stored in a structure called *TheMesh*.

2. A routine to build the assembled matrix and load vector of a linear system for a given mesh and numerical scheme.

3. A routine to solve the linear system that arises when doing the discretization.

The code we have developped here solves NSE on a square on which an uniform mesh is generated as shown in figure 3.

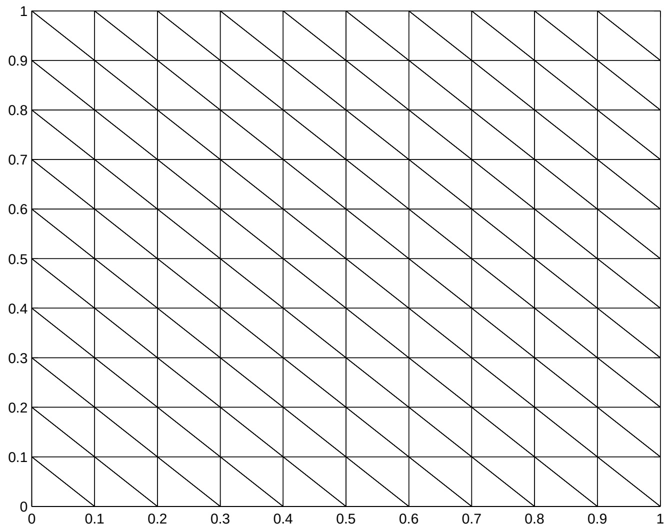

Figure 1: Uniform mesh on the unit square

Our code has the capability to solve NSE on a deformed square (the image of a square through a difeomorphism which does not make the angles in the mapped mesh too small or too big see figures 2 and 2) via a change of variables.

We give a global numbering of the velocity nodes on the mesh as seen in figure 4. In our code, we define a function *Nodeco* which for a given node

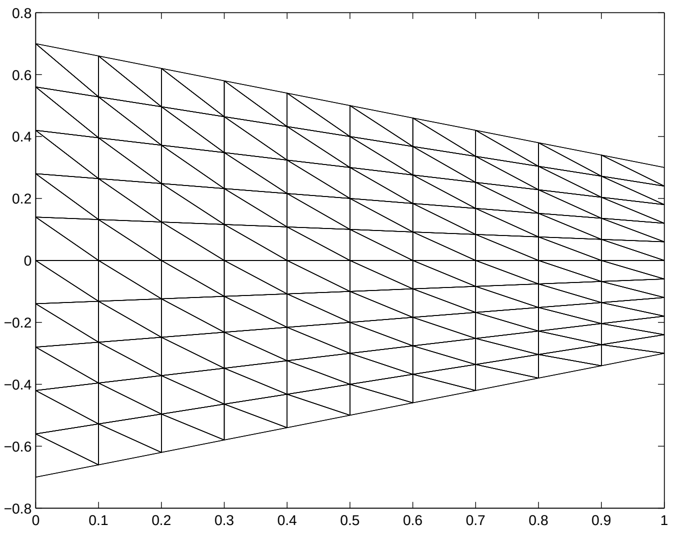

Figure 2: Uniform mesh on a trapezoid

on the mesh will return the global coordinates of that node. For example, in figure 4 above we have that *Nodeco(2)*=[0.25,0].

For parallel computation, the elements in the mesh will be distributed over the proccesors. For example if two processors are used to solve NSE on mesh on figure 4 then the first processor will own elements bellow the horizontal line y=0.5, whereas the elements above y=0.5 will be distributed to the second processor. On each processor, there will be a local numbering of the elements (see figure 5). We define a local variable called *Elnode* which will store for a given element the global indices of the velocity nodes that lie on that element. On each proccesor *Elnode* will be a double array whose number of columns is 6 and number of rows is equal to the number of elements owned by that proccesor. For the situation in figure 5, the first row of *Elnode* on the first proccesor will be [0,2,18,10,9,1] (the first three are corner points, the last 3 are middle points).

It is also neccesary to have a list of boundary nodes on each proccesor which will be used to eliminate Dirichlet boundary conditions after doing the assembly. A local variable called *boundary* will store the indices of boundary nodes on each processor. For example, in figure 5 on the first proccesor we will have *boundary*=[0,1,2,3,4,5,6,7,8,9,18,27,36,17,26,35,44].

The function *Veltopres* creates a correspondence between the index of a node as a velocity node and its index as a pressure node provided that the node is both a velocity and a pressure node. In the case of an unstructured grid this function would have to be replaced by a global numbering of the

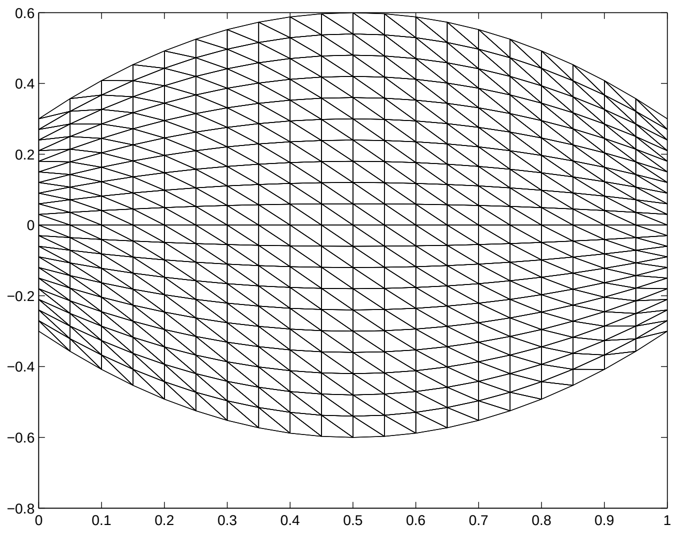

Figure 3: Uniform mesh on a deformed square

pressure nodes, a list of the coordinates of these nodes and a correspondence between the elements in the mesh and the pressure nodes on them.

## 3 The Numerical Scheme

The second routine which does the assembly is the most expensive in terms of programming. As stated in the intoduction, the NSE have to be turned into their variational formulation Find $u : [0,T] \to P2 \times P2, p : [0,T] \to P1$ such that

$$\begin{array}{rcll} (u_t, v) + \nu(\nabla u, \nabla v) + ((u \cdot \nabla)u, v) + (\nabla p, v) & = & (f, v) & \text{in } (0,T) \\ (\nabla \cdot u, q) & = & 0 & \text{in } (0,T] \\ u & \approx & g & \text{in } (0,T] \times \partial\Omega \\ u\,|_{t=0} & \approx & u_0 & \text{in } \Omega \end{array} \tag{5}$$

for every $(v,q) \in P2 \times P2 \times P1$. Afterwards, a method inspired by the Crank-Nicolson method is used to do the discretisation in time.

The Crank-Nicholson method is [3]: At time $t_{n+1}$ find $u_{n+1} \in P2 \times P2, p \in$

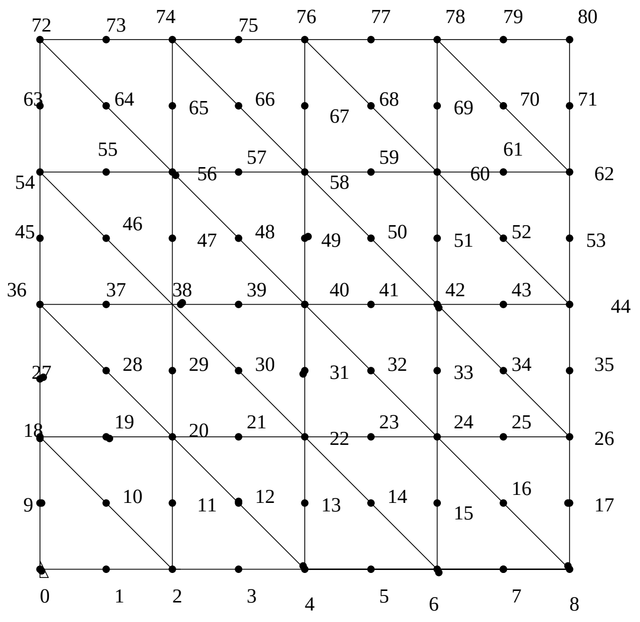

Figure 4: Global numbering of the velocity nodes on mesh shown on figure 4

$P1$ such that

$$\begin{array}{rcl} (\frac{(u_{n+1}-u_n)}{\Delta t}, v) + \nu(\nabla \frac{u_{n+1}+u_n}{2}, \nabla v) + & & \\ +0.5 * ((u_{n+1}\cdot\nabla)u_{n+1}, v) + 0.5 * ((u_n\cdot\nabla)u_n, v) + (\nabla p, v) & = & (f, v) \\ (\nabla\cdot u_{n+1}, q) & = & 0 \\ u_{n+1} & \approx & g(t_{n+1},\cdot) \ \text{ in } \partial\Omega \end{array} \tag{6}$$

for every $(v, q) \in P2 \times P2 \times P1$.

This system of nonlinear equations is solved by two iterations of the fixed point iteration presented in the introduction. The predictor $u^{(1)}$ is expanded on the canonical basis of the finite element space $P2 \times P2 \times P1$

$$u^{(1)} = \sum c_i^{n+1}\varphi_i$$

and the coefficients $c_i^{(1)}$ have to be found. When replacing $u^{(1)}$ by formula above, the following linear system arises

$$\sum B(\varphi_i, \varphi_j)c_j^{(1)} = F(\varphi_i)$$

where

$$B^*((u,p),(v,q)) = \frac{1}{\Delta t}(u,v) + 0.5*\nu*(\nabla u, \nabla v) + 0.5*((u_n\cdot\nabla)u, v) + (\nabla p, v) + (q, \nabla u)$$

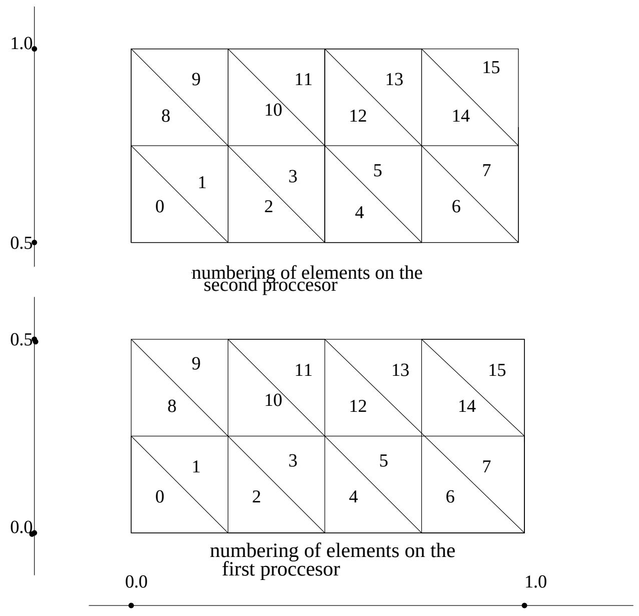
1.0
9
11
13
15
8
10
12
14
1
3
5
7
0
2
4
6
0.5
numbering of elements on the second proccesor
0.5
9
11
13
15
8
10
12
14
1
3
5
7
0
2
4
6
0.0
numbering of elements on the first proccesor
0.0
1.0

Figure 5: Distribution and local numbering of elements on two proccesors

and

$$F(v) = \frac{(u_n, v)}{\Delta t} - 0.5\nu(\nabla u_n, \nabla v) + 0.5(f^{n+1} + f^n, v).$$

The integral $B(\varphi_i, \varphi_j)$ is evaluated separatelly on each element in the support of $\varphi_i\varphi_j$ using a change of variables to the reference element and integrating there with a quadrature formula of order 7.

## 4 Convergence and Stability of the numerical scheme.

In order to study the convergence and stability of the code we have picked an example where the exact solution of the NSE is

$$(u_1, u_2) = (e^x + e^{-t} sin(y), -e^x y + e^{-t}).$$

The Reynolds number was chosen to be 2000 and we have computed the right hand side of the Navier-Stokes equations accordingly.

On the unit square $(0, 1) \times (0, 1)$ we have solved the NSE with time step $dt = 0.25$ and mesh size $h = 0.25$ and the $H^1$ -norm and $L^2$-norm of the error $u_{exact} - u_{approx}$ were measured at time 2.5 doing 10 time-steps.Then the time step $dt$ and mesh size $h$ were halved and 20 time steps were computed. The $H^1$ -norm and $L^2$-norm of the error $u_{exact} - u_{approx}$ were thus computed at time 2.5. The same procedure was applied twice and on the right column of the table 1 the quotient of two consecutive errors is shown.

Table 1: $H^1$ convergence of the numerical scheme.

| $dt$ | $h$ | iterations | $H^1$-error | Quotient |
|---|---|---|---|---|
| 0.25 | 0.25 | 10 | 0.204387 | |
| 0.125 | 0.125 | 20 | 0.0260867 | 7.8349 |
| 0.0625 | 0.0625 | 40 | 0.00550811 | 4.7361 |
| 0.03125 | 0.03125 | 80 | 0.00113084 | 4.8708 |

Table 2: $L^2$ convergence of the numerical scheme.

| $dt$ | $h$ | iterations | $H^1$-error | Quotient |
|---|---|---|---|---|
| 0.25 | 0.25 | 10 | 0.00716448 | |
| 0.125 | 0.125 | 20 | 0.000530275 | 13.5108 |
| 0.0625 | 0.0625 | 40 | 6.35028e-05 | 8.3504 |
| 0.03125 | 0.03125 | 80 | 5.30697e-06 | 11.965 |

In the case of the $H^1$ -norm we see that as time-step and mesh size are halved the error decreases by a factor of 4 so the convergence is quadratic. In the case of $L^2$ -norm as time-step and mesh size are halved the error decreases by a factor of 8 so the convergence is cubic.

The stability of the numerical scheme was studied by picking the exact solution of NSE and Reynolds number as in example 1 above, fixing the time-step to $dt = 0.25$, choosing the mesh size equal to 0.03125 and measuring $L^2$ -norm of the error at various time-levels. Table 3 shows that the $L^2$ -norm of the error is always less than $7 \cdot 10^{-6}$ According to the test above the method is stable. An analytical proof of the stability of the method is under study.

Table 3: Stability test of the numerical scheme.

| time level | $L^2$-norm of error |
|---|---|
| 0 | 4.64529e-06 |
| 1 | 6.08724e-06 |
| 2 | 6.29856e-06 |
| 3 | 6.46518e-06 |
| 4 | 6.58639e-06 |
| 5 | 6.64484e-06 |
| ............... | .............. |
| 10 | 6.28822e-06 |
| 11 | 6.20186e-06 |
| 20 | 5.7965e-06 |
| 30 | 5.58357e-06 |
| 40 | 5.40848e-06 |
| 50 | 5.32966e-06 |
| 60 | 5.3021e-06 |
| 70 | 5.29949e-06 |
| 79 | 5.30697e-06 |

## 5 The differential filter and the implementation of LES.

The idea of LES is to try to compute not the entire flow, for which computational resources could be insufficient, but only flow structures that exceed a certaib spatial scale $\delta$. To that end, we need to build a tool that, for a given velocity field $u$, will compute an average of it $\overline{u}$. Such an operation will preserve all the large-scale flow structures in u and will filter out small structures and oscilations in the solution $u$ due to a too coarse mesh.

The tool we developped in this work is an elliptic differential filter. The filter is applied by solving an auxiliary problem, an elliptic second order PDE, on the same mesh on which the flow is solved. The auxiliary problem is: Find $\overline{u}$ such that

$$\begin{array}{c} -\delta^2\Delta\overline{u}+\overline{u}=u \quad \text{in } \Omega \\ \overline{u}=u\text{in } \partial\Omega\ . \end{array} \tag{7}$$

The advantage of using differential filters instead of other filtering techniques such as convolution is that it is very cheap computationally.

One needs to assembly the stiffness matrix of the PDE (7) only one time at first time step and then solve at each time-step a linear system whose matrix is already known.

Figure 6 shows the average of the velocity field in figure 10 after the differential filter is applied to it. The mesh size is 0.05 and we picked $\delta^2 = 0.0005$.

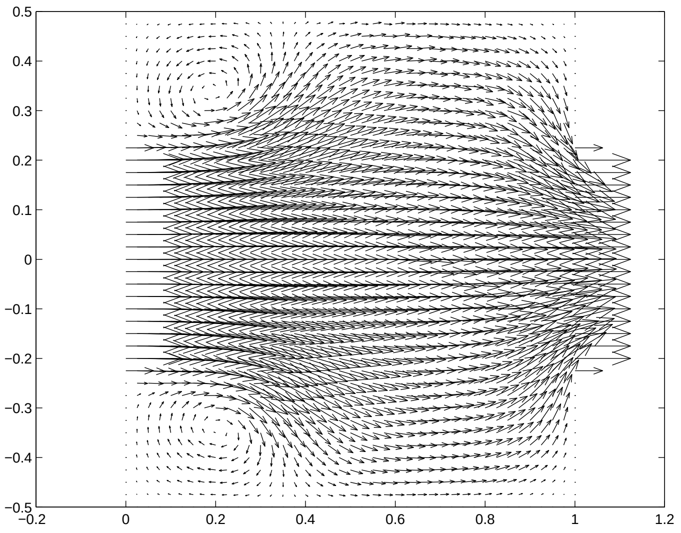

Figure 6: The average of the flow shown on figure 10

In the code the large eddies were simulated by adjusting the velocity field $u_n$ computed at each time level $t_n$. When doing DNS (direct numerical simulation) only $u_n$ was used to compute $u_{n+1}$. In the case of LES, before computing $u_{n+1}$,the average $\overline{u_n}$ is computed and then in the places where $u_n$ and $\overline{u_n}$ differ substantially, we replace $u_n$ by $\overline{u_n}$. When the difference $u_n - \overline{u_n}$ is very small ($|u_n - \overline{u_n}| < \gamma||u_n||_\infty$ where $\gamma$ is determined by experiment) we do not modify $u_n$. When computing $u_{n+1}$ this new $u_n$ is used. This method was tested with good results that are shown in the next section.

## 6 A comparison between DNS and LES .

We consider the domain $(0,1)\times(-0.5,0.5)$ and we set up Dirichlet boundary conditions as shown in the Figure 7. The inflow boundary condition is set to be equal to $\varphi$, where the shape of $\varphi$ is shown in 8, when $t > 1$ and $t\varphi$ when $0 \leq t \leq 1$. The outflow boundary condition is the same as inflow boundary condition. The flow is solved with DNS at Re=2000 with $dt = 0.05$ and mesh

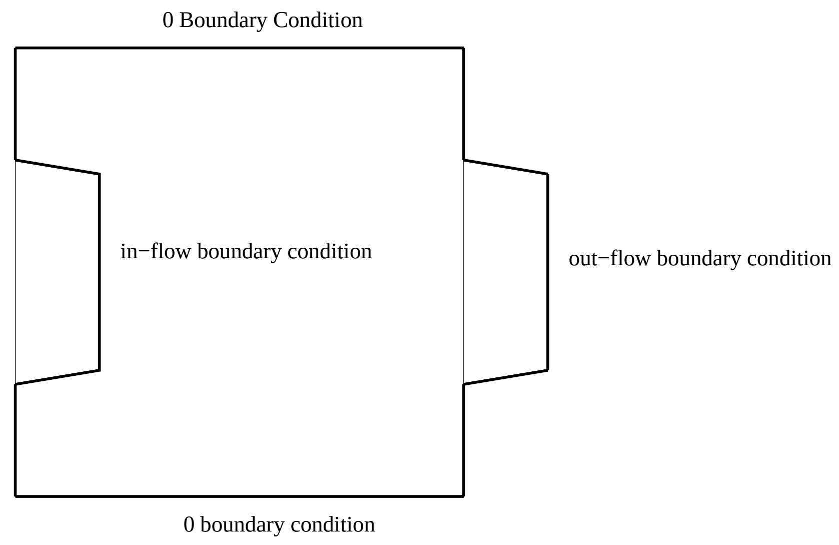

Figure 7: The domain (0,1)x(-0.5,0.5) and the Dirichlet B.C on it

size $h = 0.05$ . We show the flow after 1 second (see Figure 10) and after 5 seconds (see Figure 11).

It can be seen from the pictures above that this is not a good approximation of the exact flow because after 1 second there is a lot of oscilation on the computed solution in the outflow region and after 5 seconds no structure shows up on the upper part of the container. To obtain a better approximation the mesh is refined to $h = 0.025$ and the flow is solved again. The following figures show the computed flow after 1 second (see Figure 12) and after 5 seconds (see Figure 13). Small vortices are forming where the flow enters the container then they move horizontally and in the end they are absorbed by the two big vortices where the fluid flows out of the container.

In conclusion a DNS on a mesh $20 \times 20$ would not solve the problem and a DNS on a mesh $40 \times 40$ would give a good approximation of the flow but at the same time would require more computational effort to solve the problem. Instead of trying to compute this flow very accuratelly we will solve with LES on a mesh $20 \times 20$ to compute with accuracy only big structures of this flow. Figures 14 and 15 show the flow computed with LES after 1 second (see Figure 14) and after 5 seconds (see Figure 15). In the experiment above we picked $\gamma = 0.07$ and $\delta^2 = 0.0005$. As it is seen in the figures above the two vortices are well recovered and there are no significant oscilations in the computed solution.

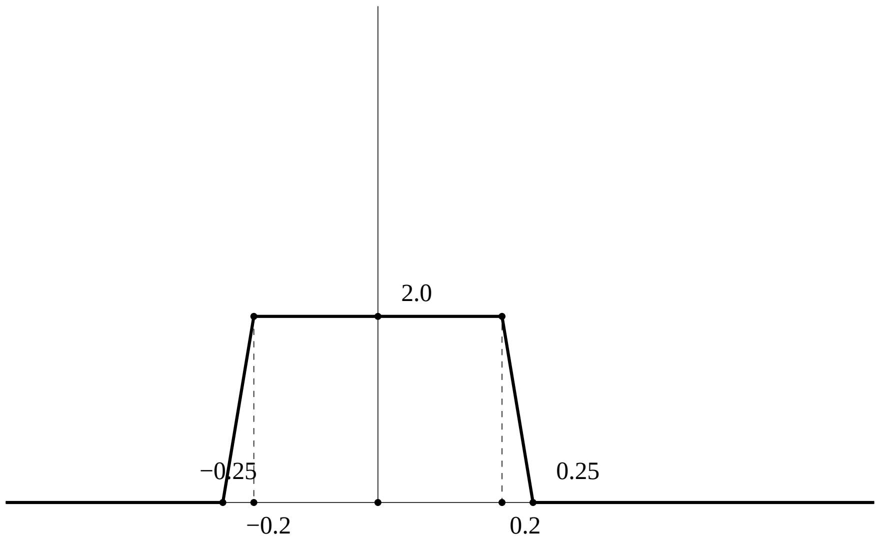

Figure 8: The function $\varphi$

## 7 The comparison of energy and vorticity computed with LES and DNS

The comparison of energy and vorticity computed with LES and DNS was carried out in the following setting. The unit square is deformed as a function of two parameters $\alpha$ and $\beta$ while keeping the area of the domain equal to 1 as shown in figure 9.

We use Dirichlet boundary conditions which are defined the same way as in the previous problem. The inflow boundary condition is set to be equal to $\varphi$ as shown in 8 when $t > 1$ and $t\varphi(x, y)$ when $0 \leq t \leq 1$. The outflow boundary condition is the same as the inflow boundary condition. Everywhere else the boundary conditions are 0. The flow was solved with DNS at Re=2000 with $dt = 0.05$, mesh size$h = 0.025$ with 200 time steps and with LES with $dt = 0.05$, mesh size $h = 0.05$ $\gamma = 0.07$ ,$\delta^2 = 0.0005$ and the same number of iterations for various values of the parameters $\alpha$ and $\beta$. Then total energy and vorticity during the first 10 seconds were computed and the results are shown in the Tables 4 and 5

We see that the kinetic energy and vorticity of the big structures in the flow are approximated well with the LES model disscused in previous two sections.

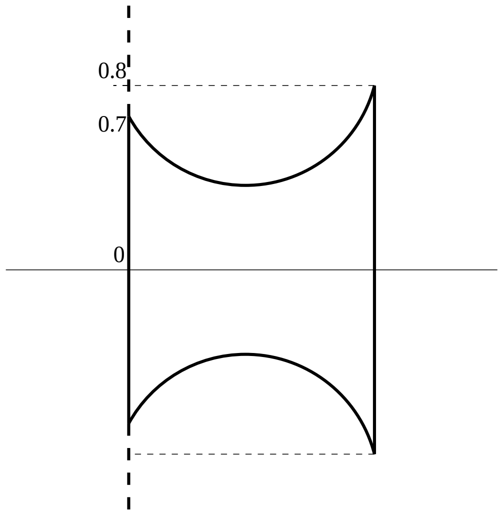

Figure 9: $\alpha = 0.7, \beta = 0.8$

## 8 Conclusions

We have constructed a computational environment for the simulation of the Navier-Stokes equations describing the motion of an incompressible fluid flow in two dimensions. The method is implemented in PETSC [1] and can thus be executed on any parallel architecture running MPI. We demonstrate numerically that the method is second-order in space and time and that it is stable at fairly high Reynolds numbers, although it solves only two linear systems per time step. In addition, we have designed an efficient way of doing large eddy simulation with our code, by the use of an elliptic differential filter. We show that our LES approach captures the important characteristics of the fluid flow (such as kinetic energy and vorticity) while taking significantly less computational effort than a DNS that produces a result of a similar quality over the large spatial scales. In the near future, we plan to analyze rigurously the properties of our numerical scheme and interface our solver with an optimization approach.

## 9 Acknowledgement

I thank the Division of Educational Programs at Argonne National Laboratory and the U.S. Dept. of Energy under Contract W-31-109-ENG-38 that have provided the support for the author of this report, through a Givens fellowship.

Table 4: Comparison of energy computed with LES and DNS.

| $\alpha$ | $\beta$ | kinetic energy with LES | kinetic energy with DNS |
|---|---|---|---|
| 0.25 | 0.25 | 15.7033 | 16.2235 |
| 0.25 | 0.5 | 15.6198 | 16.1618 |
| 0.25 | 0.75 | 15.3528 | 15.8571 |
| 0.25 | 1.0 | 15.0686 | 15.3465 |
| 0.25 | 1.25 | 18.2731 | 17.0209 |
| 0.5 | 0.25 | 15.7818 | 16.1674 |
| 0.5 | 0.5 | 15.9302 | 16.3165 |
| 0.5 | 0.75 | 15.7961 | 16.0797 |
| 0.5 | 1.0 | 14.849 | 15.5623 |
| 0.5 | 1.25 | 16.1781 | 15.1901 |

I also thank the members of the Mathematics and Computer Science Division at Argonne National Laboratory, especially my advisors Dr.Mihai Anitescu and Dr.Traian Iliescu, for their kind and enthusiast support throughout the duration of this project.

## References

[1] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.3, Argonne National Laboratory, 2002.

[2] Alexandre J. Chorin and Jerrold E. Marsden. *A Mathematical Introduction to Fluid Dynamics*. Springer-Verlag, 1992.

[3] Charles A. Hall and Thomas A. Porsching. *Numerical Analysis of Partial Differential Equations*. Prentice-Hall, 1990.

Table 5: Comparison of vorticity computed with LES and DNS.

| $\alpha$ | $\beta$ | vorticity with LES | vorticity with DNS |
|---|---|---|---|
| 0.25 | 0.25 | 696.974 | 701.785 |
| 0.25 | 0.5 | 726.861 | 707.405 |
| 0.25 | 0.75 | 713.274 | 688.725 |
| 0.25 | 1.0 | 749.095 | 678.744 |
| 0.25 | 1.25 | 1401 | 1271.49 |
| 0.5 | 0.25 | 725.818 | 774.544 |
| 0.5 | 0.5 | 765.063 | 780.091 |
| 0.5 | 0.75 | 757.065 | 766.581 |
| 0.5 | 1.0 | 722.276 | 667.836 |
| 0.5 | 1.25 | 905.446 | 685.388 |

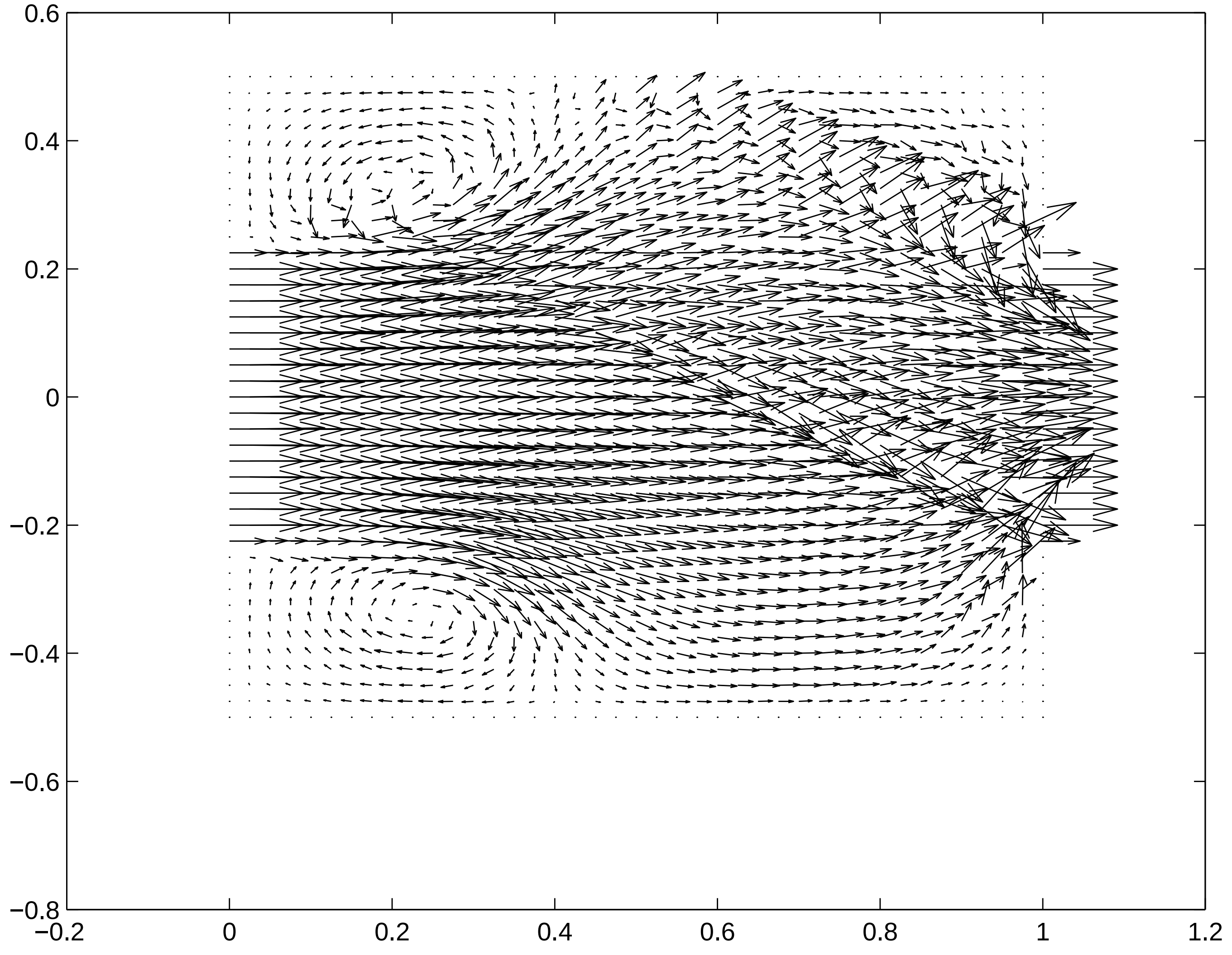

Figure 10: Flow computed with DNS after 1 second on mesh 20 × 20

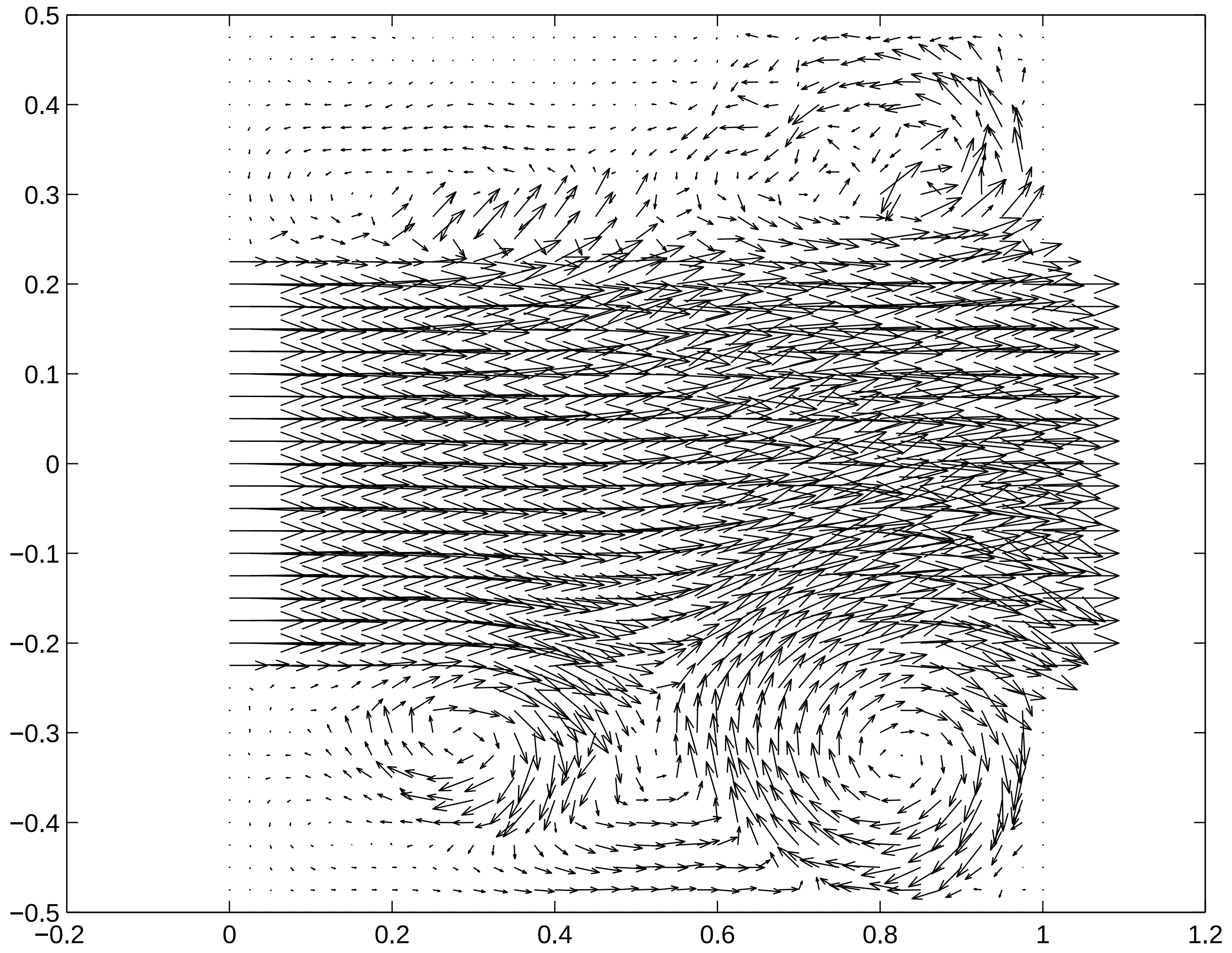

Figure 11: Flow computed with DNS after 5 seconds on mesh 20 × 20

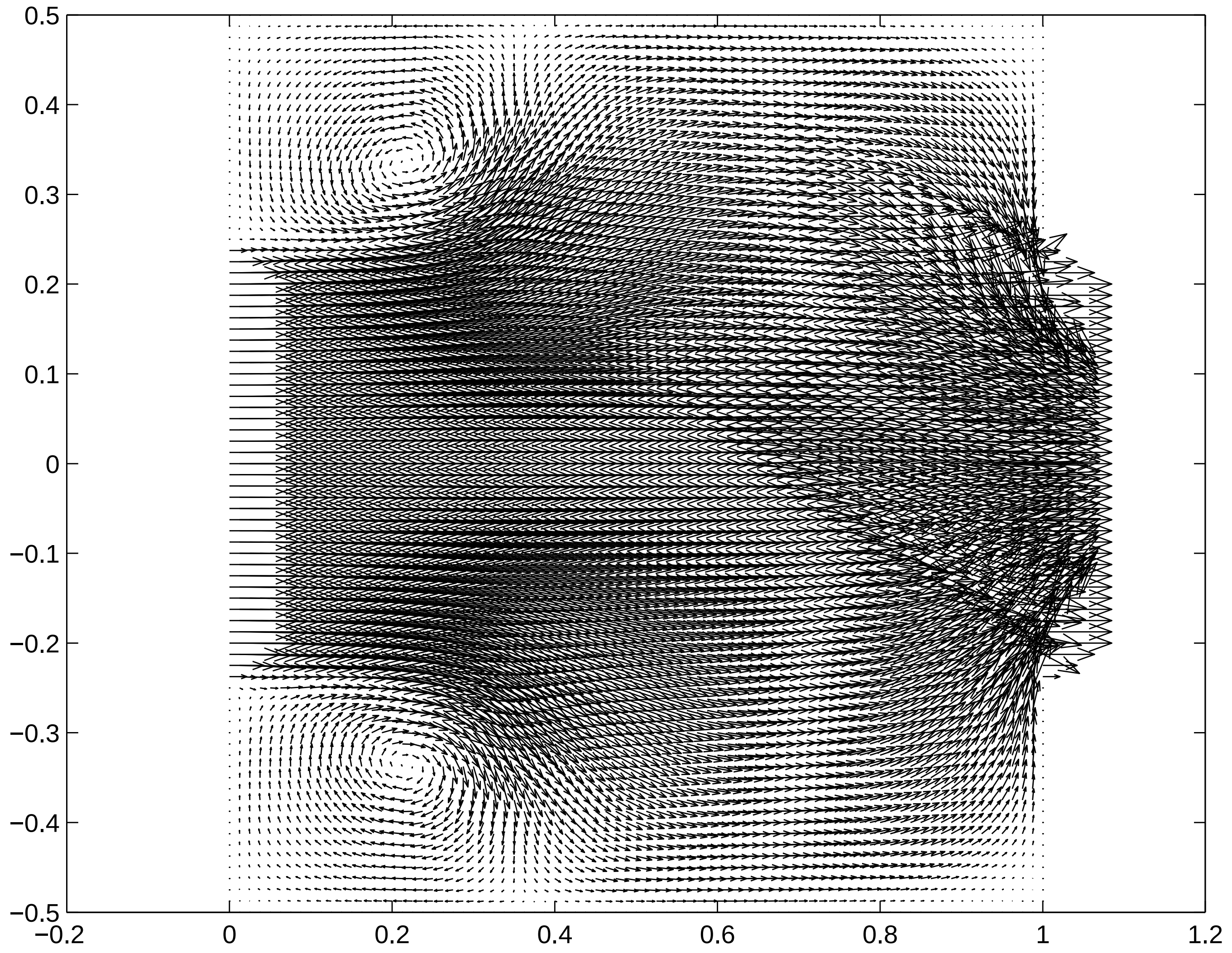

Figure 12: Flow computed with DNS after 1 second on mesh 40 × 40

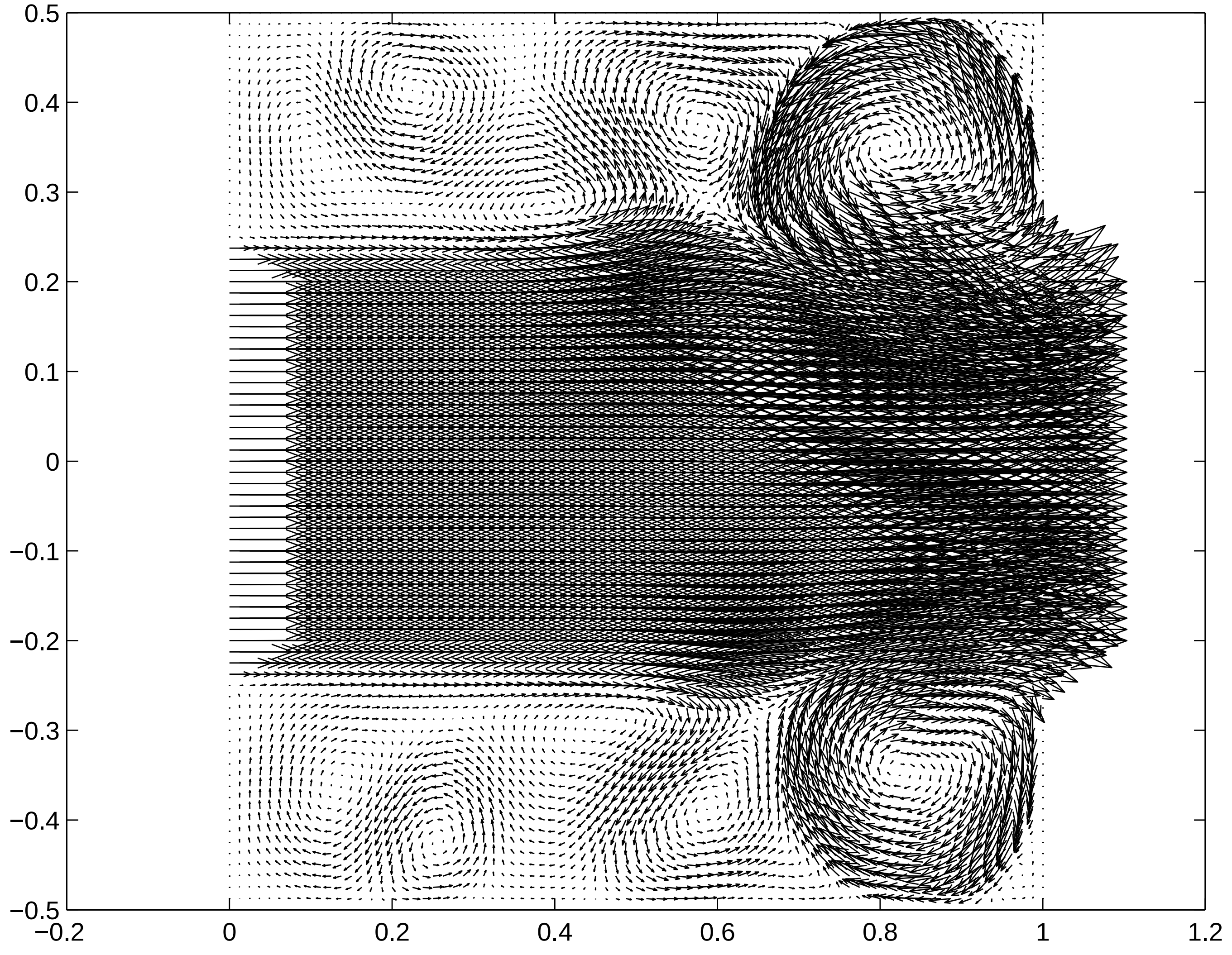

Figure 13: Flow computed with DNS after 5 seconds on mesh 40 $\times$ 40

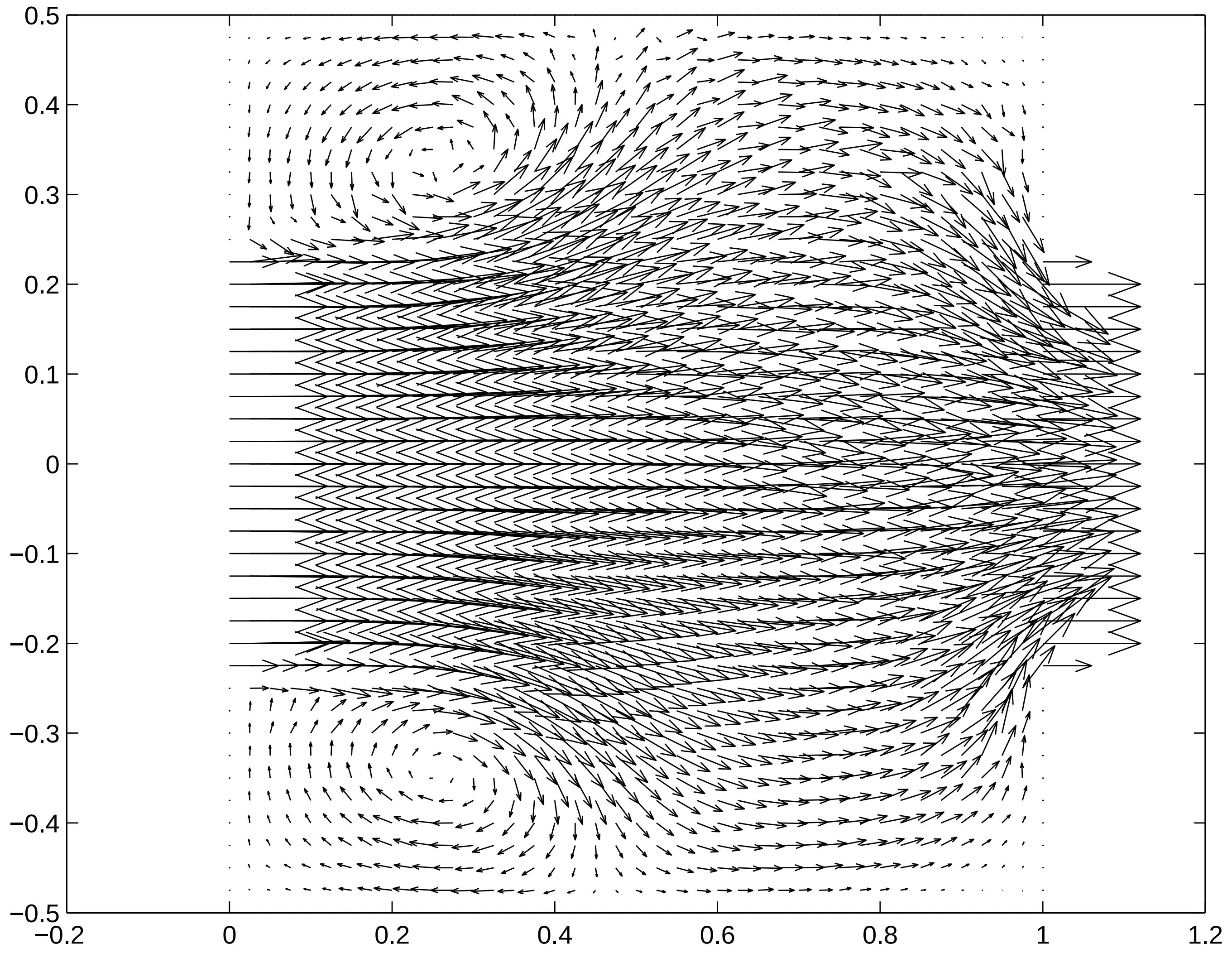

Figure 14: Flow computed with LES after 1sec on mesh 20x20

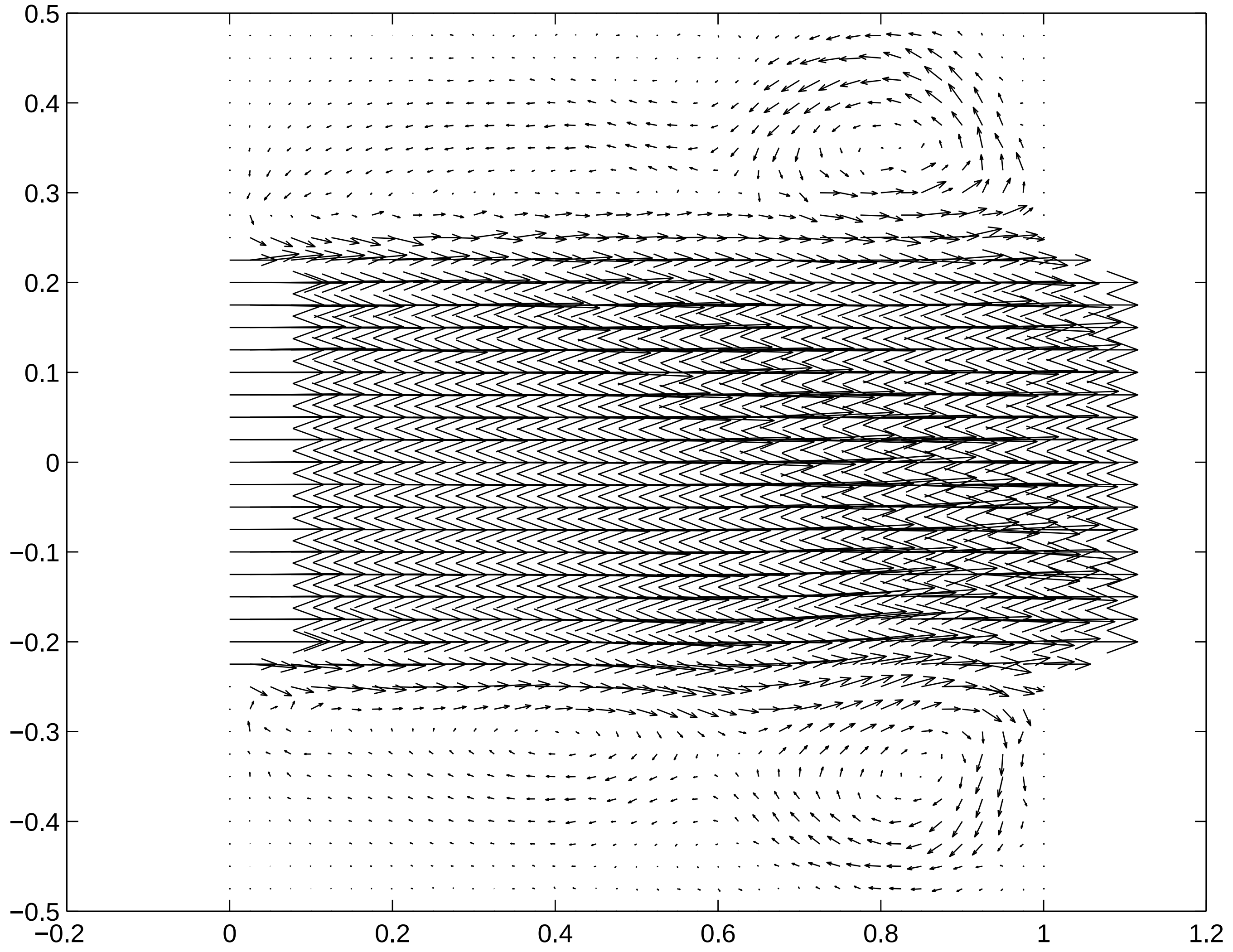

Figure 15: Flow computed with LES after 5 seconds on mesh 20x20