TIME DISCRETIZATION STRATEGIES FOR A 3D LID-DRIVEN CAVITY BENCHMARK WITH PETSc

Damien Tromeur-Dervout

▶ **To cite this version:**

Damien Tromeur-Dervout. TIME DISCRETIZATION STRATEGIES FOR A 3D LID-DRIVEN CAVITY BENCHMARK WITH PETSc. 2018. hal-01943307v1

HAL Id: hal-01943307

https://hal.science/hal-01943307v1

Preprint submitted on 3 Dec 2018 (v1), last revised 22 Oct 2019 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright

TIME DISCRETIZATION STRATEGIES FOR A 3D LID-DRIVEN CAVITY BENCHMARK WITH PETSc

D. Tromeur-Dervout[a,b,*]

[a] *Université de Lyon, CNRS, Université Lyon 1, Institut Camille Jordan UMR 5208, 43 bd du 11 novembre 1918, F-69622 Villeurbanne-Cedex*
[b] *MAM dept of Polytech Lyon, University Lyon 1, 15 bd Latarjet, F-69622 Villeurbanne-Cedex*

Abstract

We present parallel implementations in PETSc of three time discretization strategies to solve the 3D incompressible Navier-Stokes equations in velocity-vorticity formulation applied to the lid-driven cavity with aspect ratio 3:1 with second order finite differences discretization in space. Different features of the PETSC software are investigated allowing fast prototyping of parallel numerical methods adapted to the proposed strategies. Comparisons on parallel efficiency, numerical accuracy, and flow behavior are given.

Keywords: High performance computing, Navier-Stokes, lid-driven cavity problem

Introduction

This paper is devoted to the use of the Portable, Extensible Toolkit for Scientific Computation (PETSc) for solving the 3D Navier-Stokes equations written in vorticity-velocity formulation applied to the lid-driven cavity with aspect ration 3:1. This test case is originally used by us to investigate the PETSc capabilities in the teaching of high performance computing to engineer at Polytech Lyon engineering school in the applied mathematics department. These classes come after the teaching of the Message Passing Interface library, that facilates the understanding of the PETSc data distribution and communication through MPI communicators. We show that different time

*Principal corresponding author
Email address: damien.tromeur-dervout@univ-lyon1.fr (D. Tromeur-Dervout)

discretizations of this problem can illustrate the use of linear and nonlinear solver capabilities of PETSc. The great flexibility programming of PETSc allows to investigate different solvers and preconditionner, facilitating the fast prototyping of the best numerical solvers in terms of convergence and accuracy on laboratory small computing facilities (up to 60 cores). Further investigations on the French national resources permited some comparisons of the time discretisation strategies, (up to 2048 cores), that provides non intuitive results on parallelism. We also investigate the impact of the time discretizing strategies on the flow behavior, showing that explicitation in time of the boundary conditions leads to a delay in time in the flow behavior.

The plan of this paper is as follows: section 2 describes the 3D Navier-Stokes equations governing the flow written in vorticity-velocity formulation, and the special care needed to define the boundary conditions on the vorticity. Section 3 focuses on the time discretisation and its programming counter part in the PETSc coding framework. We show three time discretizing strategies with taking implicitly more and more terms in the system of equations. Section 4 gives the results in term of parallelism of the three discretizing in time strategies while section 5 exhibits the flow behavior with respect of the time strtaegies. Section 6 concludes this paper.

1. Lid-driven cavity test case and space discretizing

1.1. Governing equations

The lid-driven cavity is a classical test case in fluid mechanics. Several papers investigate the flow behaviour with respect to the Reynolds numbers [1, 2]. For a cavity with an aspect ratio of 3:1, we expect to exhibit in the plan parallel to the flow a primary eddy at the center and three secondary eddies that rotate in the opposite way than the primary eddy. In the plan orthogonal to the flow some Taylor-Gortler (TG) structures in the flow appear after some time. The number of these TG seems to be 7 at time $T = 100$ and 9 at time $T = 200$ [3]. This structures are depicted in the figure 1.

The flow in a 3D lid-driven cavity with aspect ratio 3.1 is modelled by the incompressible Navier-Stokes equations in the velocity-vorticity ($\vec{V} - \vec{\omega}$) formulation that

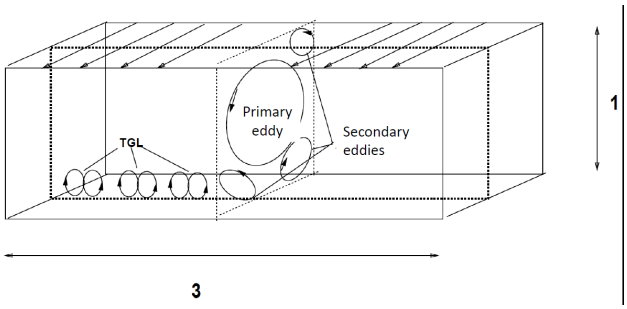

Figure 1: Lid-driven cavity with aspect ratio 3:1

follows:

$$\frac{\partial \vec{\omega}}{\partial t} - \vec{\nabla} \times (\vec{V} \times \vec{\omega}) = \frac{1}{Re} \Delta \vec{\omega} + B.C. + I.C \quad (1)$$

$$\Delta \vec{V} = -\vec{\nabla} \times \vec{\omega} + B.C. + I.C. \quad (2)$$

We have a transport equation with a time derivative for the vorticity while an elliptic equation gives the velocity. These equations have some advantage and drawback with the Navier-Stokes equations in primitive variables velocity-pressure, notably it avoids the difficult computing of the pressure that guarantees the incompressibility. The other advantages is to compute directly the vorticity for rotating flows. On the other part the definition of the Vorticity boundary condition is not naturally defined and must be deduced from the vorticity definition. The elliptic equation must be accurately solved as it guarantees the incompressibility. The other drawback is that we have to solve 6 scalar equations: 3 for the velocity and 3 for the vorticity although some mathematical manipulations can reduced the number of equation for the velocity.

1.2. Space discretizing

We use second order finite differences for the space discretizing with regular step size in the three dimensions. This leads to the standard 7 points discretizing for the Laplacian and we used centred first order derivative for the convection terms (for the forthcoming time discretizing strategies 1 and 3 and first order decentred finite differences for time discretizing strategy 2). This space discretizing gives regular data dependancies that links each points to its six neighbhoured points if they exist (see

figure 2. Our approach has the advantage to avoid the corner singularities in the computation but misses the effect on the flow of this corner singularities [4] that should be taken in account in further developments. Here we are mainly focus on the time discretizing effect on the parallelism and on the flow behavior.

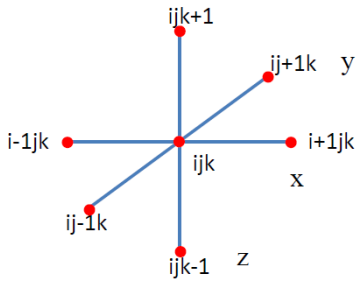

Figure 2: Data dependancies stencil.

1.3. Boundary conditions

For $t \geq 0$ we impose no-slip boundary condition for the velocity:
$\vec{V} = (V_x = 0, V_y = 0, V_z = 0)$ for $(x = \pm\frac{3}{2}, y = \pm\frac{1}{2}, z = -\frac{1}{2})$, $\vec{V} = (0,1,0)$ for $(z = +\frac{1}{2})$.

For the vorticity boundary condition we use the vorticity definition, $\vec{\nabla} \times \vec{\mathbf{V}}$ associated to the values of the velocity derivatives at the walll. For example, at the wall $z_0 = \pm\frac{1}{2}$, we have :

$$V_x \quad = \quad V_y = V_z = 0, \text{ et } \frac{\partial V_x}{\partial x} = \frac{\partial V_x}{\partial y} = \frac{\partial V_y}{\partial x} = \frac{\partial V_y}{\partial y} = \frac{\partial V_z}{\partial x} = \frac{\partial V_z}{\partial y} = 0 \qquad (3)$$

Using the definition of the vorticity we obtain:

$$z_0 = \pm\frac{1}{2} \quad : \omega_x(x,y,z_0) = -\frac{\partial V_y}{\partial z}; \quad \omega_y(x,y,z_0) = \frac{\partial V_x}{\partial z}; \quad \omega_z(x,y,z_0) = 0$$

The same considerations lead to the others vorticity boundary conditions:

$$\begin{array}{llll} x_0 = \pm\dfrac{3}{2} & : \omega_x(x_0,y,z) = 0; & \omega_y(x_0,y,z) = -\dfrac{\partial V_z}{\partial x}; & \omega_z(x_0,y,z) = \dfrac{\partial V_y}{\partial x} \\ y_0 = \pm\dfrac{1}{2} & : \omega_x(x,y_0,z) = \dfrac{\partial V_z}{\partial y}; & \omega_y(x,y_0,z) = 0; & \omega_z(x,y_0,z) = -\dfrac{\partial V_x}{\partial y} \end{array}$$

In order to have a second order approximation for the vorticity boundary conditions, we use a limited development of the velocity components at the viscinity of the wall of the cavity. Considering, the Poisson equations on the velocity on the wall $z = -\frac{1}{2}$, we have :

$$\frac{\partial^2 V_x}{\partial z^2} = \frac{\partial \omega_y}{\partial z}, \frac{\partial^2 V_y}{\partial z^2} = -\frac{\partial \omega_x}{\partial z}$$

Using a Taylor expansion of V_x and V_y at the viscinity of the wall $z_0 = -\frac{1}{2}$, we have:

$$V_x(x,y,-\frac{1}{2}+\Delta z) - V_x(x,y,-\frac{1}{2}) = \Delta z \frac{\partial V_x}{\partial z}(x,y,-\frac{1}{2}) + \frac{\Delta z^2}{2}\frac{\partial^2 V_x}{\partial z^2}(x,y,-\frac{1}{2}) + O(\Delta z^3)$$
$$V_y(x,y,-\frac{1}{2}+\Delta z) - V_y(x,y,-\frac{1}{2}) = \Delta z \frac{\partial V_y}{\partial z}(x,y,-\frac{1}{2}) + \frac{\Delta z^2}{2}\frac{\partial^2 V_2 y}{\partial z^2}(x,y,-\frac{1}{2}) + O(\Delta z^3)$$

Replacing the partial derivatives of V_x and V_y with respect of those of ω_x and ω_y:

$$V_x(x,y,-\frac{1}{2}+\Delta z) - V_x(x,y,-\frac{1}{2}) = \Delta z \omega_y(x,y,-\frac{1}{2}) + \frac{\Delta z^2}{2}\frac{\partial \omega_y}{\partial z}(x,y,-\frac{1}{2}) + O(\Delta z^3)$$
$$V_y(x,y,-\frac{1}{2}+\Delta z) - V_y(x,y,-\frac{1}{2}) = -\Delta z \omega_x(x,y,-\frac{1}{2}) - \frac{\Delta z^2}{2}\frac{\partial \omega_x}{\partial z}(x,y,-\frac{1}{2}) + O(\Delta z^3)$$

Discretizing $\frac{\partial \omega_y}{\partial z}$ and $\frac{\partial \omega_x}{\partial z}$ at the first order, we deduce the boundary conditions at $z = -\frac{1}{2}$:

$$\begin{aligned}
\omega_x(x,y,-\frac{1}{2}) + \omega_x(x,y,-\frac{1}{2}+\Delta z) &= -\frac{2}{\Delta z}(V_y(x,y,-\frac{1}{2}+\Delta z) - V_y(x,y,-\frac{1}{2})) \\
\omega_y(x,y,-\frac{1}{2}) + \omega_y(x,y,-\frac{1}{2}+\Delta z) &= \frac{2}{\Delta z}(V_x(x,y,-\frac{1}{2}+\Delta z) - V_x(x,y,-\frac{1}{2})) \\
\omega_z(x,y,-\frac{1}{2}) &= 0
\end{aligned}$$

We see that the boundary condition on the vorticity links the point on the boundary and the proximate point in the direction. This is important as all the flow dynamics come from the lid-driven. The other point is some delay in time between the vorticity and the velocity occurs if this boundary conditions are not treated implicitly.

2. Time discretizing strategies and their coding in PETSc

This lid-driven cavity problem has been implemented in a research parallel code, requiring 3 years man of development, and using ADI for time marching on vorticity

(second order in time) and multigrid accelerated by schur dual domain decomposition with generalized conjugate residual Krylov method for the velocity [5]. This code implementation was efficient numerically and in elapsed time in a benchmarking codes comparison[3].

A direct implementation of this code in the PETSc framework would not possible without devoting a lot of effort as there is no ADI solvers implemented in PETSc. The two difficulties would have been to define Krylov solvers associated to each direction and to associate the unknowns to each plan orthogonal to the direction. This can be an opportunity of development of PETSc.

Instead, we investigate different features of the PETSc [6, 7] parallel software to implement three time discretizing strategies for (1)-(2) that take more and more terms implicitly in the equations and boundary conditions. This benchmark constitutes the core content to teach PETSc software to engineer students in applied mathematics in few hours.

As the problem geometry is a cavity and the space discretizing is with second order finite differences then the velocity and vorticity fields can be represented by structured datas like the PETSc distributed array (DA). User can let PETSc choose how to distribute the data between the MPI processes. He has to give the dimensions in each direction, the type and the length l of the computing stencils (stencil box for data dependencies in $(i\pm l, j\pm l, k\pm l)$ or Stencil star for data dependencies in $(i\pm l, j, k), (i, j\pm l, k), (i, j, k\pm l))$) and the degree of freedom per point DOF. This DA defines the data dependancies and consequently the maximum number of coefficient entries per row in the discretizing matrices.

2.1. Strategy 1 : time explicit convection term

The first strategy solves (1) with taking the convection term explicitly and then solves (2) with taking the curl of the computed vorticity as right hand side:

$$(\mathbb{I} - \frac{\Delta t}{Re}\Delta)\vec{\omega}^{n+1} = \vec{\omega}^n + \Delta t\vec{\nabla}\times(\vec{V}^n\times\vec{\omega}^n) \quad (4)$$

$$\Delta\vec{V}^{n+1} = -\vec{\nabla}\times\vec{\omega}^{n+1}, \text{ after solving (4)} \quad (5)$$

We use a distributed array with 3 DOF and we define two Krylov Space Projec-

```
ierr = PetscInitialize(&argc,&argv,(char*)0,help);

ierr = DMDACreate3d(PETSC_COMM_WORLD,
       DM_BOUNDARY_NONE,DM_BOUNDARY_NONE,DM_BOUNDARY_NONE,
       DMDA_STENCIL_STAR,Nx,Ny,Nz,
       PETSC_DECIDE,PETSC_DECIDE,PETSC_DECIDE,
       3,1,0,0,0,&daNS);

ierr = DMDASetFieldName(daNS,0,"Vx");
ierr = DMDASetFieldName(daNS,1,"Vy");
ierr = DMDASetFieldName(daNS,2,"Vz");

ierr = DMCreateGlobalVector(daNS, &VWnm1.V);
ierr = VecSet(VWnm1.V,0.)
```

Table 1: PETSc coding to distribute the fields between processors

tion solvers KSP one for the velocity (KSPV) and one for the vorticity (KSPW). The KSP and its preconditioner PC have to be defined with the KSPSetType and PCSetType functions. The KSP is associated to the DM through the KSPSetDM function. This defines the potential matrix structure of the operator associated to the KSP. The user provides two functions, one for building the matrix (here MatrixVelocity) and for building the right hand side (here RHSVelocity) that are associated to the KSP through the KSPSetComputeOperators and KSPSetComputeRHS functions respectively.

```
ierr = KSPCreate(PETSC_COMM_WORLD,&kspV);
ierr = KSPSetDM(kspV,daNS);
ierr = KSPAppendOptionsPrefix(kspV,"V_");
ierr = KSPSetType(kspV, KSPGMRES);
ierr = KSPGetPC(kspV,&pcVelocity);
ierr = PCSetType(pcVelocity,PCHYPRE);
ierr = KSPSetComputeRHS(kspV,RHSVelocity, &VWnm1);CHKERRQ(ierr);
ierr = KSPSetComputeOperators(kspV,MatrixVelocity,NULL);
ierr = KSPSetInitialGuessNonzero(kspV,PETSC_TRUE);
ierr = KSPSetTolerances(kspV,1e-12,1e-6,1e6,50);
ierr = KSPSetFromOptions(kspV);
```

Table 2: PETSc coding of the KSP set up for velocity

The main programming effort is to define the functions giving the matrix and the

RHS. The DM is retrieved from the KSP through KSPGetDM, then DMDAGetLocalInfo gets the sizes (xm,ym,zm) of the subdomain managed by the process and the size (mx,my,mz) of the global computational domain aand also the starting point (xs,ys,zs) at the left left down corner of the subdomain.

The data are extracted from the vector by DMDAVecGetArrayDOF. Extracting the data field with its ghost points in the subdomain needs to create a local vector using DMGetLocalVector, then to extract a localArray with again DMDAVecGetArrayDOF and finally to update the gost point with neighbouring subdomains with DMGlobalToLocalBegin and DMGlobalToLocalEnd. The data of the RHS are then computed using the 4-index array [k][j][i][dof]. The data associated to the RHS have to be restore in the vector using DMDAVecRestoreArrayDOF and must be assembly with VecAssemblyBegin and VecAssemblyEnd. Local vectors have to be restore before leaving with DMRestoreLocalVector.

Then for strategies 1 one time step consists to solve the linear system with KSPSolve and to extract the solution from KSP with KSPGetSolution. We tested sev-

```
for ( it=1; it< nbiter; it++) {

  PetscPrintf(PETSC_COMM_WORLD,"it: %d \n",it);

  ierr = KSPSolve(kspV,NULL,NULL);
  ierr = KSPGetSolution(kspV, &VWnm1.V);

  ierr = KSPSolve(kspW,NULL,NULL);
  ierr = KSPGetSolution(kspW,&VWnm1.W);
}
```

Table 3: PETSc coding of the time loop for strategy 1

eral KSP solvers associated to different preconditioners. It appears that the best pair KSP/PC for the elliptic equation is the GMRES [8, 9] associated to the HYPRE [10] preconditionner that converges in 5 (respectively 3) iterations per time step for the velocity (respectively the vorticity) for the $192 \times 64 \times 64$ mesh.

```
PetscErrorCode RHSVorticity(KSP ksp,Vec b, void *ctx) {
  DM                  da;
  DMDALocalInfo info;
  Vec xlocalV, xlocalW;
  PetscScalar    ****arrayRHS, ****arrayV,  ****arrayW;
  SolutionNS *NS = (SolutionNS*) ctx;
  PetscInt       i,j,k,l,zdebut,zfin,ydebut,yfin,xdebut,xfin,rank;
  PetscScalar    Hx,Hy,Hz;
  PetscFunctionBeginUser;
  KSPGetDM(ksp,&da);
  DMDAGetLocalInfo(da, &info);
  Hx   = ((PetscReal)(info.mx-1)/2)/3.;
  Hy   = (PetscReal)(info.my-1)/2;
  Hz   = (PetscReal)(info.mz-1)/2;

  DMDAVecGetArrayDOF(da, b, &arrayRHS);
  DMGetLocalVector(da,&xlocalW);
  DMGlobalToLocalBegin(da,NS->W,INSERT_VALUES,xlocalW);
  DMGlobalToLocalEnd(da,NS->W,INSERT_VALUES,xlocalW);
  DMDAVecGetArrayDOF(da,xlocalW,&arrayW);
  DMGetLocalVector(da,&xlocalV);
  DMGlobalToLocalBegin(da,NS->V,INSERT_VALUES,xlocalV);
  DMGlobalToLocalEnd(da,NS->V,INSERT_VALUES,xlocalV);
  DMDAVecGetArrayDOF(da,xlocalV,&arrayV);

  if (info.zs==0) { // vorticity boundary condition for the top
    for (j=info.ys; j<info.ys+info.ym; j++) {
     for (i=info.xs; i<info.xs+info.xm; i++) {
        arrayRHS[info.zs][j][i][0] =
             -2*2*Hz*(arrayV[1][j][i][1]-arrayV[0][j][i][1]);
        arrayRHS[info.zs][j][i][1] =
             2*2*Hz*(arrayV[1][j][i][0]-arrayV[0][j][i][0]);
        arrayRHS[info.zs][j][i][2] = 0;    }  }
   }
//  other boundary condition and RHS (arrayW) computations
  DMDAVecRestoreArrayDOF(da, b, &arrayRHS);
  VecAssemblyBegin(b);
  VecAssemblyEnd(b);

  DMDAVecRestoreArrayDOF(da,xlocalW,&arrayW);
  DMRestoreLocalVector(da,&xlocalW);
  DMDAVecRestoreArrayDOF(da,xlocalV,&arrayV);
  DMRestoreLocalVector(da,&xlocalV);
  }
```

Table 4: Part of the PETSc coding for the RHS vortiticity function

2.2. Strategy 2 : semi-implicit convection term

Strategy 2 solves (1)-(2) with taking the convection term semi-implicitly:

$$\begin{pmatrix} (\mathbb{I} - \frac{\Delta t}{Re}\Delta). + \Delta t \nabla \times (\vec{\mathbf{V}}^n \times .) & 0 \\ \nabla \times . & \Delta. \end{pmatrix} \begin{pmatrix} \vec{\omega}^{n+1} \\ \vec{V}^{n+1} \end{pmatrix} = \begin{pmatrix} \vec{\omega}^n \\ 0 \end{pmatrix} \tag{6}$$

Let us notice that for strategy 2, we still use the boundary condition for the vorticity using the $\vec{\mathbf{V}}^n$ as we split the solution of (6) in two stages, the solution of the velocity and then the solution for the vorticity.

The PETSc implementation of strategy 2 involves two distributed array (DA) with three degree of freedom (DOF) with two Krylov solvers (KSP) with updating a part of the matrix vorticity at each time step. The vorticity boundary condition would have been taken implicitly if we used a DA with six DOF and on KSP taking in account the six equations.

One time step for strategy 2 differs from those of strategy 1 with calling KSPSetComputeOperators function to update the matrix vorticity before KSPSolve associated to the vorticity equation (see Table 5).

```
for ( it=1; it< nbiter; it++) {

  PetscPrintf(PETSC_COMM_WORLD,"it: %d \n",it);

  ierr = KSPSolve(kspV,NULL,NULL);
  ierr = KSPGetSolution(kspV, &VWnm1.V);

  ierr = KSPSetComputeOperators(kspW,MatrixVorticity,&VWnm1);
  ierr = KSPSolve(kspW,NULL,NULL);
  ierr = KSPGetSolution(kspW,&VWnm1.W);
}
```

Table 5: PETSc coding of the time loop for strategy 2

Again the best pair KSP/PC for strategy 2 is the GMRES/HYPRE pair , that converges in 5 (respectively 3) iterates per time step for the velocity (respectively the vorticity) for the $192 \times 64 \times 64$ mesh and in 6 (respectively 4) iterates per time step for the velocity (respectively the vorticity) for the $384 \times 128 \times 128$.

2.3. Strategy 3 : totaly implicit formulation

The third strategy solves (1)-(2) totally implicitly:

$$\begin{pmatrix} (\mathbb{I} - \frac{\Delta t}{Re}\Delta)(\vec{\omega}^{n+1} - \Delta t\vec{\nabla} \times (\vec{\mathbf{V}}^{n+1} \times \vec{\omega}^{n+1}) - \vec{\omega}^{n} \\ \Delta\vec{\mathbf{V}}^{n+1} + \nabla \times \vec{\omega}^{n+1} \end{pmatrix} = \vec{0} \qquad (7)$$

In this strategy the boundary condition for the vorticity is totally implicit as well as the convection term as the six equations are solved together.

For strategy 3 the DM has six DOF and we create a SNES (system of Nonlinear Equations Solver) object. This DM is associated to the SNES through the command SNESSetDM. Then the user has to provide the function to minimize, implementing the six discretized equations and boundary conditions with SNESSetFunction. As the SNES may needs the Jacobian of the function to minimize we can provide the jacobian to the SNES with the SNESSetJacobian function or we can let the Jacobian be evaluated with differentiating with providing a Index Set coloring associated to the data dependancies in the function to minimize in order to optimize the number of call to the function. This is done by the MatFDColoringCreate creating a MatFDColoring object that has been associated to the SNES through the SNESSetJacobian command.We also can change the KSP from the SNES through SNESGetKSP. One time step iteration consists in applying SNESSolve and then get de solution with SNESGetSolution.

```
  DMSetMatType(daNS,MATAIJ);
  DMCreateMatrix(daNS,&J);

  MatFDColoring  matfdcoloring;
  ISColoring iscoloring;
  DMCreateColoring(daNS,IS_COLORING_GLOBAL,&iscoloring);
  MatFDColoringCreate(J,iscoloring,&matfdcoloring);
  MatFDColoringSetType(matfdcoloring,MATMFFD_DS);
  MatFDColoringSetFunction(matfdcoloring,(PetscErrorCode
      (*)(void))FunctionNS3DVW,&VWnm1);
  MatFDColoringSetFromOptions(matfdcoloring);
  MatFDColoringSetUp(J,iscoloring,matfdcoloring);
   ISColoringDestroy(&iscoloring);


SNESCreate(PETSC_COMM_WORLD,&snes);
SNESSetDM(snes,daNS);
SNESSetFunction(snes,NULL,FunctionNS3DVW,(void*)&VWnm1);
SNESSetJacobian(snes,J,J,SNESComputeJacobianDefaultColor,matfdcoloring);
SNESSetComputeInitialGuess(snes,FormInitialGuess,&VWnm1);

SNESGetKSP(snes,&kspV);          KSPSetType(kspV, KSPGMRES);
KSPGetPC(kspV,&pcVelocity);   PCSetType(pcVelocity,PCHYPRE);
KSPSetTolerances(kspV,1e-4,1e-4,1e6,50);
KSPSetFromOptions(kspV);
SNESSetFromOptions(snes);
ierr = SNESSetUp(snes);

for (it=1; it< nbiter; it++) {
PetscPrintf(PETSC_COMM_WORLD,"it: %d \n",it);

ierr = SNESSolve(snes,NULL,NULL);
ierr = SNESGetSolution(snes,&VWnm1.VW);
}
```

Table 6: PETSc coding of the SNES setup for strategy 3 and the time loop associated

We also tested several SNES solver the two candidates have been newtonls for line search Newton [11] and anderson mixing [12]. While newtonls was performing better to reach a relative tolerance of 1e-6 , anderson mixing with 300 fixed-point algorithm iterations was prefered to reach a relative tolerance of 1e-8 as we will discuss in the flow behavior result. From our knownledge there no simultation of velocity-vorticity

3D formualtion of Navier-Stokes with fully implicit time discretizing in the literature. Some works can be found for the fully time implicit 3D lid-driven cubic cavity in primitive variable with Reynold 1000 [13] or also [14].

3. Parallelism results

3.1. Computational ressources and time measurment

Table 7 presents the characteristics of computational ressources used to obtain the parallelism results. We dispose of a local computational ressource that allows us to define the methodology and to obtain the flow behavior and some results on parallel efficiency in a shared memory environment. The national allocated computing ressource occigen is finite 50000 hours and is used to obtain parallel speed-up in an dedicated environment with up to date software framework by an expert dedicated team. The results on occigen will use as mush as possible the load balancing between memory and CPU with using, when is possible, not all CPUs of each node.

Computer	glenan	occigen
Model	Dell PE920	bulx DLC
Processor	Xeon	Xeon
	E7-4880v2	E5-2690v3
	4x15C 2.5Ghz	1x12C 2.6Ghz
Memory/Node	512Gb	110Gb
Network	shared memory	Infinityband FDR
Ressource	laboratory	national (cines)
	∞ hours	50000 hours
Use	speed-up tests	speed-up tests
	flow behavior	

Table 7: Computing ressources characteristics

PETSc provides a smart mechanism to monitor the performance with a command line option *-log_view* that provides counters on events defined by the user. The user have

to declare PetscLogStage and use the PetscLogStagePush(stage) and PetscLogStage-Pop(); before and after the portion of code that must be monitorzed as illustrated in table 8. Then when the code terminates, the log of selected events with their time measurement are displayed as shown in table 9.

```
PetscLogStage stage3, stage4;
PetscLogStageRegister("KSPV solve",&stage3);
PetscLogStageRegister("KSPW solve",&stage4);
for ( it=itsave+1; it< itsave+nbiter; it++) {
PetscPrintf(PETSC_COMM_WORLD,"it: %d \n",it);
PetscLogStagePush(stage3);
ierr = ComputeRHS(&VW, &RHS);
ierr = KSPSolve(kspV,NULL,NULL);
ierr = KSPGetSolution(kspV, &VWnm1.V);
PetscLogStagePop();
PetscLogStagePush(stage4);
ierr = KSPSolve(kspW,NULL,NULL);
ierr = KSPGetSolution(kspW,&VWnm1.W);
PetscLogStagePop();}
```

Table 8: Coding of the event log for parallel performances

```
---------------------------------------------- PETSc Performance Summary: ----------------------------------------------

./nsVW on a arch-linux2-c-opt named glenan with 64 processors, by dtromeur Fri Mar  2 16:44:07 2018
Using Petsc Release Version 3.5.4, May, 23, 2015

                         Max       Max/Min        Avg      Total
Time (sec):           4.644e+01      1.00008   4.644e+01
Objects:              1.190e+02      1.00000   1.190e+02
Flops:                2.081e+08      1.01142   2.072e+08  1.326e+10
Flops/sec:            4.481e+06      1.01147   4.460e+06  2.855e+08
MPI Messages:         7.120e+02      1.66745   6.060e+02  3.879e+04
MPI Message Lengths:  8.730e+06      1.76995   1.190e+04  4.615e+08
MPI Reductions:       4.820e+02      1.00000

Summary of Stages:   ----- Time ------  ----- Flops -----  --- Messages ---  -- Message Lengths --  -- Reductions --
                        Avg     %Total     Avg     %Total   counts   %Total     Avg         %Total   counts   %Total
 0:      Main Stage: 2.4242e-02   0.1%  0.0000e+00   0.0%  1.620e+02   0.4%  3.246e+00        0.0%  0.000e+00   0.0%
 1:  Initialization: 5.7360e-02   0.1%  0.0000e+00   0.0%  1.632e+03   4.2%  4.824e+01        0.4%  4.300e+01   8.9%
 2: Init Saved Fields: 4.8056e-06   0.0%  0.0000e+00   0.0%  0.000e+00   0.0%  0.000e+00        0.0%  0.000e+00   0.0%
 3:        Init KSP: 1.5372e-03   0.0%  0.0000e+00   0.0%  0.000e+00   0.0%  0.000e+00        0.0%  0.000e+00   0.0%
 4:     first solve: 2.5315e+01  54.5%  1.6297e+09  12.3%  5.168e+03  13.3%  1.445e+03       12.1%  1.410e+02  29.3%
 5:      KSPV solve: 1.9861e+01  42.8%  7.1637e+09  54.0%  1.714e+04  44.2%  5.779e+03       48.6%  1.800e+02  37.3%
 6:      KSPW solve: 1.1827e+00   2.5%  4.4644e+09  33.7%  1.469e+04  37.9%  4.623e+03       38.9%  1.170e+02  24.3%
 7:       SaveFinal: 1.4156e-07   0.0%  0.0000e+00   0.0%  0.000e+00   0.0%  0.000e+00        0.0%  0.000e+00   0.0%
------------------------------------------------------------------------------------------------------------------------
```

Table 9: PETSc log of the code for 10+1(first) time steps of strategy 1 on glenan for the mesh $192 \times 64 \times 64$

We made the constation that the first solve always takes more elapse time, this why we separated the first solve from other iterations in the time measurement.

3.2. Speed-up for strategy 1

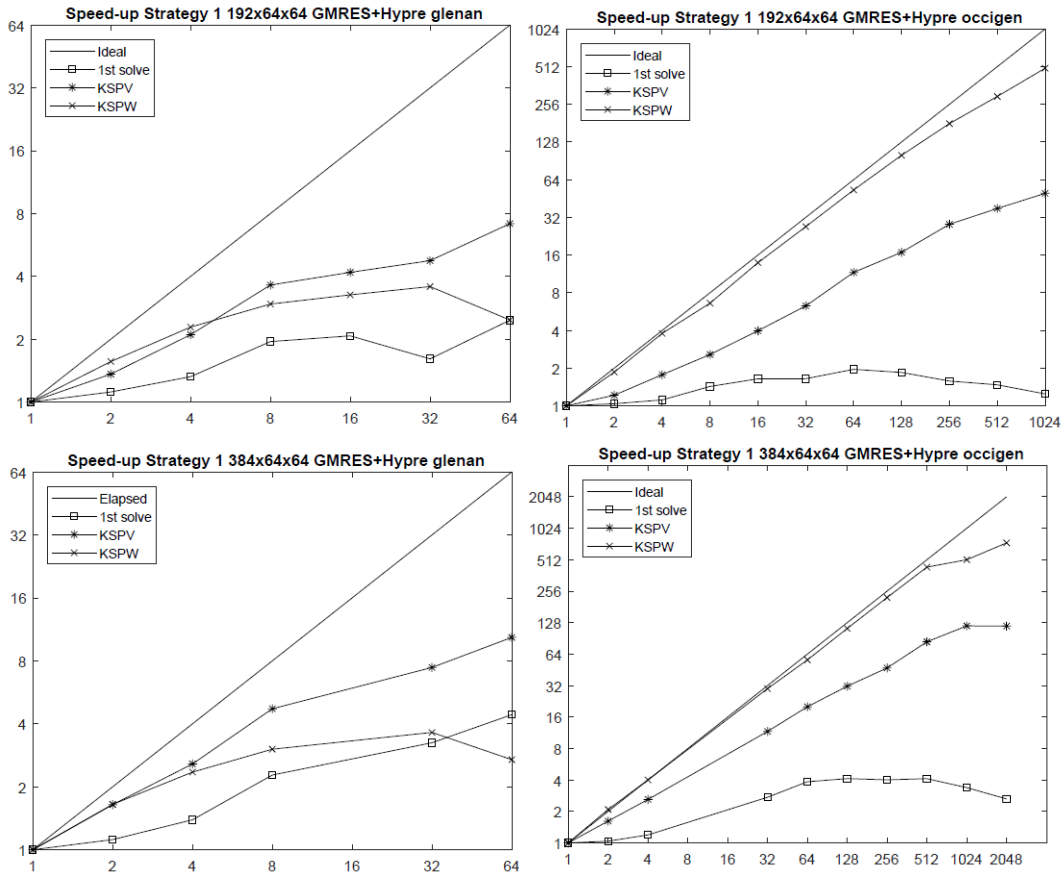

Figure 3: Speed-up (strong scaling) of strategy 1 on glenan (left) and occigen (right) for domain sizes 192×64 *times*64 (top) and 384×128 *times*128

Figure 3 gives the speed-up for strategy 1 for two meshes 192×64 *times*64 and 384×128 *times*128 on glenan and occigen with the GMRES+Hypre solver. We see that the speed-up on glenan is limited to 8 for 64 threads while the KSPW scales better on occigen with a speed-up of 512 for 1024 processes for the 192×64 *times*64 mesh and 741,86 for 2048 processes for the 384×128 *times*128 mesh. We also note that the KSPW scales better than the KSPV on occigen due to the nature of the operator nearby the identity compared to the Laplacian operator, that should give less complex

algebraic multigrid projection and restriction operators and so more efficient in term of parallelism preconditioner. Nevertheless the speed-up on glenan for the KSPW is strongly degraded on the $384 \times 64 \times 64$ with nearby 4 for 32 threads. This degradation is less sensible on occigen.

3.3. Speed-up for strategy 2

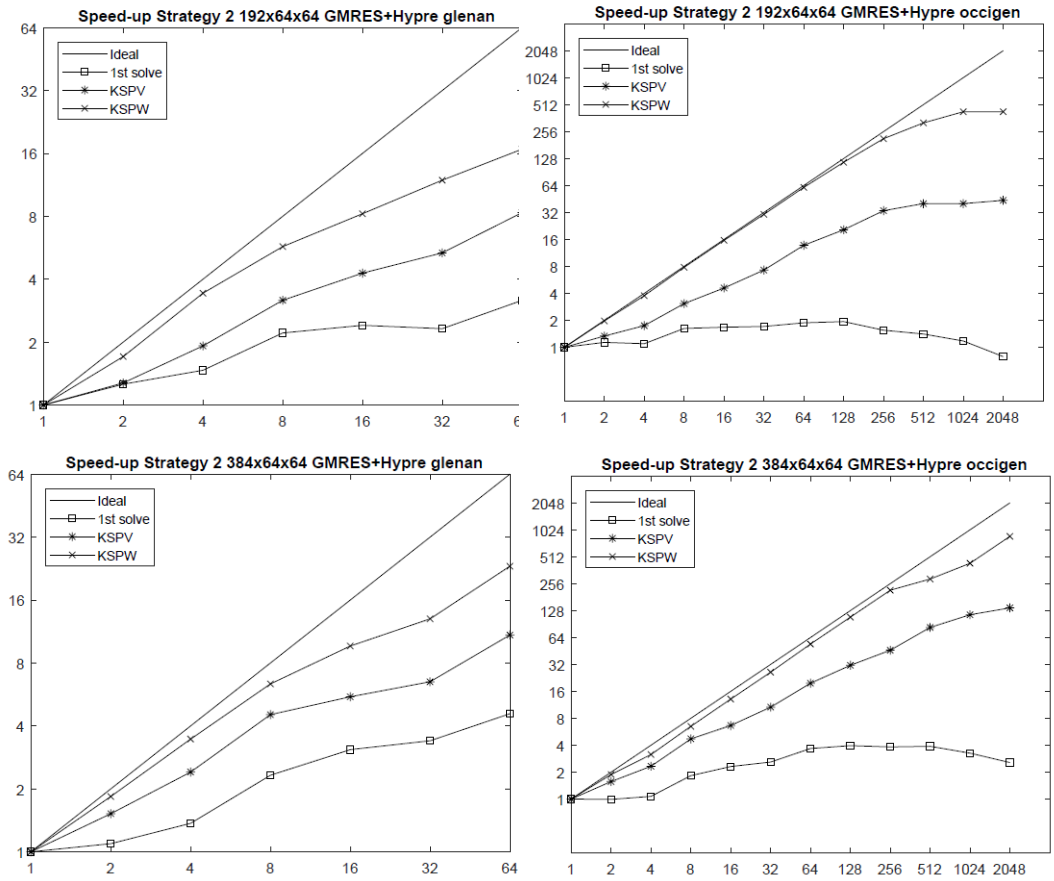

Figure 4: Speed-up (strong scaling) of strategy 2 on glenan (left) and occigen (right) for domain sizes 192×64 *times*64 (top) and 384×128 *times*128

Figure 4 gives the speed-up for strategy 2 for two meshes 192×64 *times*64 and 384×128 *times*128 on glenan and on occigen with the GMRES+Hypre solver. Again

the speed-up is better for the KSPW than for KSPV, even if the stagnation of the speed-up comes more early with the introduction of convective term in the vorticity operator that leads to an operator with the diagonal dominance reinforced . A speed-up of 426,67 for 1024 processes is reach on the $192 \times 64 \times 64$ on occigen for KSPW. This speed-up stagnates on 2048 processes. The speed-up is again around 786,36 for 2048 processes on occigen for the $384 \times 128 \times 128$ mesh.

3.4. Speed-up for strategy 3

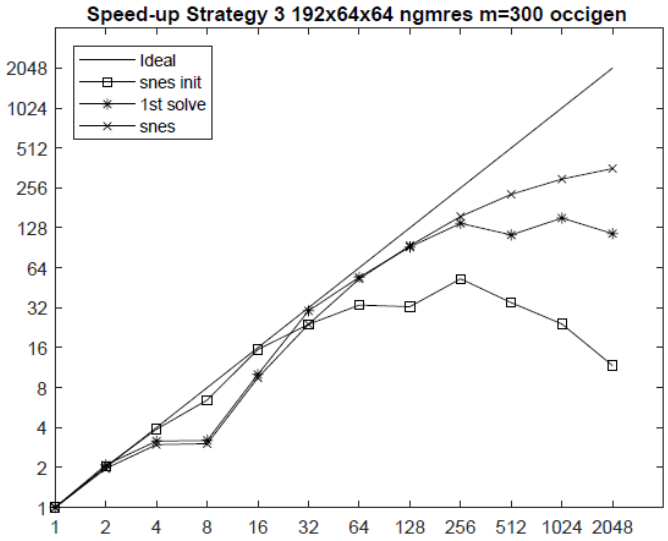

Figure 5: Speed-up for strategy 3 with ngmres and $192 \times 64 \times 64$

Figure 5 gives the speed-up for the strategy 3 on occigen for a mesh of $192 \times 64 \times 64$ withAnderson mixing with 300 fixed-point iterations. A speed-up of is obtained for 1024 CPUs. We see that the SNES's initialisation scales until 64 and the scaling performances is degrated after. As this initialisation time is small compared to the time associated to the number of time iterates, this result is not significant. The SNES solving scales well until 64 porcesses on occigen, and a speed-up of 298 (357 respectively) is obtained on 1024 processes (respectively 2048 processes).

P	S1	S1 LU	S2	S3	comments
1	12.61	-	12.59	137.6	too much even with no CFL
2	9.97	32.89	9.09	65.8	
4	**6.69**	20.48	**6.78**	43.55	S1 vs S2 similar (<>tol)
8	4.56	10.77	3.82	43.04	
16	2.91	8.50	2.52	13.70	
32	1.83	4.54	1.58	4.54	S3 start to win on S1 LU
64	0.99	4.16	0.83	2.53	
128	0.68	2.37	0.56	1.50	
256	0.40	2.42	0.34	1.00	S3 competitive with no CFL
512	0.30	1.83	0.28	1.22	
1024	0.23	5.05	0.28	0.91	
2048	-	-	0.26	1.18	

Table 10: Time per iteration: occigen dt=0.005 192x64x64

3.5. Elapsed time comparison for the three strategies

Table 10 gives the elapse time per iteration for all the strategies for the $192 \times 64 \times 64$ mesh with $dt = 0.005$ on occigen computer.

- The strategy 1 with superLU didn't run onto occigen due to lack of memory with 110Gb. Strategies 1 and 2 take quite the same time on one processor while strategy 3 took 11 times more time on 1 processor.
- Then strategy 2 is quite always better than strategy 1 even if the matrix for the vorticity change. This is quite surprising. This may due to the decentred discretizing of the convetion term for the vorticity equation that leads to more diagonal dominant matrix than for the Laplacian and consequently has a better Krylov convergence. The velocity equations are solved identically in both strategies.
- For 32 processors the strategy 3 starts to win over the strategy 1 with superLU direct solving.

- For 256 processors the strategy 3 becomes competitive with strategy 1 as there is no CFL condition on the time step.

To be complete, we must notice that the elapsed time per point and time step is around $1.15e^{-6}$ to $3e^{-7}$ that is not impressive compared to the elapsed time of the optimized research code with ADI and multigrid and domain decomposition done on the CRAY YMP 25 years ago with 2.510^{-5} s per point and time step for a $81 \times 41 \times 41$ mesh [3]. Nevertheless the programming effort was low (once understanding the PETSc philosophy of implementation); we also let PETSc deciding of the data distribution and the equations have been solved with a better precision at each time step.

4. Flow behavior

For the flow behavior comparison between the three strategies, we restrict ourselves to compare the flow behavior in the $x-z$ plane at $y = 2/3$ where the Taylor-Gortler vortices should appear. We extract the distributed component Ω_2 from the distributed mesh through MPI implementation although we could use some vecGather function from the PETSc library. The figures have been generated with matlab contourf function for isovalues equal to $[-10,-5,-3,-1,-0.5,0,0.5,1,3,5,10]$

4.1. Undersolve problem for strategy 3

Figure 6 exhibits a problem that occuref for the strategy 3 when the solver a newtonls with a $rtol = 1e-6$ was used. It presents the flow behavior at $T = 38$ that keeps its symmetry while it looses its symmetry for $T = 60$. This is due to some undersolve solution at some time after $T = 38$. We advocate that as the equations are symetrics as the boundary conditions also, we should keep the symmetry of the flow. The problem was solved by changing the solver with the ngmres (Anderson mixing with 300 fixed-point iterations) that allows to solve the problem with a $rtol = 1e-8$.

Figure 6 gives the flow behavior for $T = 38$ and $T = 60$ for strategy 3 with newtonls solver with a tolerance of $1e-6$. We see that the flow looses its symmetry due to an undersolving during some time step. This why we easely changed the solver with ngmres with 300 fixed-point iterations in order to reach a tolerance of $1e^{-8}$.

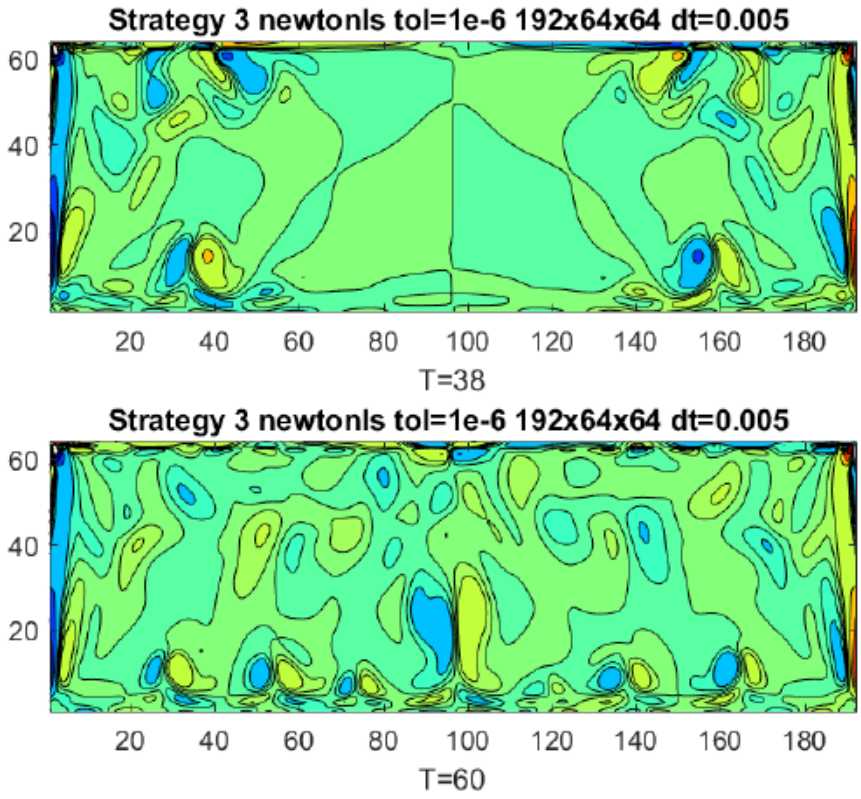

Figure 6: Effect on the undersolve time step solution on the flow behavior: up the Newton LS algorithm for strategy 3 with a $tol = 1e^{-6}$ at $T = 38$ and bottom the lost of the flow symmetry at $T = 60$.

4.2. Flow behavior comparison for the three strategies

Figures 7 , 8, and 9 compare the flow behavior for the three strategies at time $T = \{20.28, 45.00, 73.48\}$ for the mesh $192 \times 64 \times 64$ and a time step value of $dt = 0.005$ (which is imposed by the CFL condition of strategy 1). The solvers used are superlu+dist, gmres+hypre and ngmres 300 (Anderson mixing) for strategies 1, 2 and 3 respectively.

Figure 7 exhibits some quite similar results for strategy 1 and strategy 2 with slighty more strong vortice in position i=45 and k=50. The difference coming from the implicitation of the convection term as the treatment of the vorticity boundary conditions are unchanged. For the strategy 3, it shows that the vortices attached to the boundaries are

more strong, and the flow is slighty in advance compared to the two other strategies as the vortice in position i=20, k=55 is already attached to the vortice on the boundary. We conclude that the non implicitation of the convection term and with more impact the non implicitation of the vorticity boundary conditions introduces a delay on the flow.

The flow behavior at T=45 on Figure 8 still exhibits the quite same flow structure for the three strategy, with more strong vortices for the strategy 3. The flow keeps its symmetry. This is also the case for Figure 9 where the strategy 3 exhibits more defined vortices structures.

4.3. Flow behavior comparison with respect to the time step for strategy 3

Figure 10 gives the flow behavior at time $T = 76$ for the strategy 3 with the ngmres solver for two time steps $dt = 2.10^{-2}$ and $dt = 5.10^{-3}$. We see a good agreement between the two computations and see the advantage of strategy 3 that has no CFL condition. The limitation for the time step is only due to physics consideration, in order to catch the right dynamics and we see that we can take a 4 times greater time step for this flow behavior computation.

5. Conclusions

This paper shows the use of different functionnalities of PETSc to solve the lid-driven cavity problem with aspect ratio 3:1 for Reynolds 3200 from a semi-implicit to a fully implicit in time formulations. We notably exhibits that the implicitation of boundary condition leads to stronger vortices much more that the implicitation of the convective term. This conclusion is somewhat previsible as all the dynamics of the flow comes from the boundaries. The PETSc library permit us to tests different solver and to choose the solver and its parameters the best adapted for the computation. It was somewhat surprising in a first approach that the implicitation of the convective term gives better results on the parallelism for the solver associated to the vorticity. This is mainly due to the HYPRE preconditioner behavior, that creates algebraically the restriction and projection operators, which performs well for the operator where we reinforce the diagonal dominance.

The fully implicit formulation has the advantage to not have CFL condition and allows us to violate at least 4 times this CFL of the strategy 1 with keeping the right flow behavior. One output of this work, is to demonstrates that the fully implicit in time strategy can be competitive (reachable) with a sufficient number of processes. Nevertheless, we must mention that we could provide to the PETSc library with the KSPRegister instruction our owner implementation of gmres where we can stored the directions of descent from different consecutive time steps in a cumulative Krylov space since the operator for strategy 1 does not change. Then we can initialize the new time step solution with projecting the new right hand side with respect to this Krylov space [5, 15, 16]. The discretizing could also be improve by several way, with at least better space discretizing with compact schemes [17] for example and better discretizing in time. The perspective of this work could be to go further in the contrÃť le of the error on the solution with using for example the BDF schemes with adaptative time step of the SUNDIALS [18] library.

[1] S. Bhaumik, T. K. Sengupta, A new velocityâĂŞvorticity formulation for direct numerical simulation of 3d transitional and turbulent flows, Journal of Computational Physics 284 (2015) 230 – 260. doi:https://doi.org/10.1016/j.jcp.2014.12.030.
URL http://www.sciencedirect.com/science/article/pii/S002199911400847X

[2] G. Guj, F. Stella, A vorticity-velocity method for the numerical solution of 3d incompressible flows, Journal of Computational Physics 106 (2) (1993) 286 – 298. doi:https://doi.org/10.1016/S0021-9991(83)71108-3.
URL http://www.sciencedirect.com/science/article/pii/S0021999183711083

[3] D. Tromeur-Dervout, T. P. Loc, Parallelization via Domain Decomposition Techniques of Multigrid and ADI Solvers for Navier-Stokes Equations, Notes on Numerical Fluid Mechanics, Numerical Simulation of 3D Incompressible Unsteady Viscous Laminar Flows 36 (1992) 107–118, iSBN 3-528-07636-4.

[4] O. Botella, R. Peyret, Benchmark spectral results on the lid-driven cavity flow, Computers & Fluids 27 (4) (1998) 421 – 433. doi:https://doi.org/10.1016/S0045-7930(98)00002-4.
URL http://www.sciencedirect.com/science/article/pii/S0045793098000024

[5] D. Tromeur-Dervout, Résolution des Equations de Navier-Stokes en Frmulation Vitesse-Tourbillon sur Systèmes Multiprocesseurs à Mémoire Distribuée, Ph.D. thesis, University Paris 6 / ONERA (January 1993).

[6] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11 - Revision 3.8, Argonne National Laboratory (2017).
URL http://www.mcs.anl.gov/petsc

[7] S. Balay, W. D. Gropp, L. C. McInnes, B. F. Smith, Efficient management of parallelism in object oriented numerical software libraries, in: E. Arge, A. M. Bruaset, H. P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhäuser Press, 1997, pp. 163–202.

[8] Y. SAAD, M. SCHULTZ, GMRES - A Generalized Minimal RESidual algorithm for solving nonsymmetric linear systems, SIAM Journal on SCIentific and Statistical Computing 7 (3) (1986) 856–869. doi:{10.1137/0907058}.

[9] Y. Kuznetsov, Numerical methods in subspaces, in: G. I. Marchuk (Ed.), Vychislitel'nye Processy i Sistemy II, Nauka, Moscow, 1985, pp. 265–350.

[10] R. Falgout, U. Yang, hypre: A library of high performance preconditioners, in: Sloot, P and Tan, CJK and Dongarra, JJ and Hoekstra, AG (Ed.), COMPUTATIONAL SCIENCE-ICCS 2002, PT III, PROCEEDINGS, Vol. 2331 of LECTURE NOTES IN COMPUTER SCIENCE, Univ Amsterdam, Sect Computat Sci; SHARCNET, Canada; Univ Tennessee, Dept Comp Sci; Power Comp &

Commun BV; Elsevier Sci Publ; Springer Verlag; HPCN Fdn; Natl Supercomp Facilities; Sun Microsyst Inc; Queens Univ, Sch Comp Sci, 2002, pp. 632–641, International Conference on Computational Science, AMSTERDAM, NETHERLANDS, APR 21-24, 2002.

[11] P. Deuflhard, Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-23899-4_2.
URL https://doi.org/10.1007/978-3-642-23899-4_2

[12] H. Walker, P. Ni, Anderson acceleration for fixed-point iterations, SIAM Journal on Numerical Analysis 49 (4) (2011) 1715–1735. arXiv:https://doi.org/10.1137/10078356X, doi:10.1137/10078356X.
URL https://doi.org/10.1137/10078356X

[13] H. C. Elman, V. E. Howle, J. N. Shadid, R. S. Tuminaro, A parallel block multi-level preconditioner for the 3d incompressible navierâĂŞstokes equations, Journal of Computational Physics 187 (2) (2003) 504 – 523. doi:https://doi.org/10.1016/S0021-9991(03)00121-9.
URL http://www.sciencedirect.com/science/article/pii/S0021999103001219

[14] D. Loghin, A. J. Wathen, Schur complement preconditioners for the navierâĂŞstokes equations, International Journal for Numerical Methods in Fluids 40 (3âĂŘ4) (2002) 403–412. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.296, doi:10.1002/fld.296.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.296

[15] J. Erhel, F. Guyomarc'h, An augmented conjugate gradient method for solving consecutive symmetric positive definite linear systems, SIAM Journal on Matrix Analysis and Applications 21 (4) (2000) 1279–1299. arXiv:https://doi.org/10.1137/S0895479897330194, doi:10.1137/S0895479897330194.
URL https://doi.org/10.1137/S0895479897330194

[16] D. Tromeur-Dervout, Y. Vassilevski, Choice of initial guess in iterative solution of series of systems arising in fluid flow simulations, Journal of Computational Physics 219 (1) (2006) 210 – 227. doi:https://doi.org/10.1016/j.jcp.2006.03.014.
URL http://www.sciencedirect.com/science/article/pii/S0021999106001483

[17] S. K. Lele, Compact finite difference schemes with spectral-like resolution, Journal of Computational Physics 103 (1) (1992) 16 – 42. doi:https://doi.org/10.1016/0021-9991(92)90324-R.
URL http://www.sciencedirect.com/science/article/pii/002199919290324R

[18] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, C. S. Woodward, Sundials: Suite of nonlinear and differential/algebraic equation solvers, ACM Transactions on Mathematical Software (TOMS) 31 (3) (2005) 363–396.

Acknowledgements

This work was granted access to the HPC resources of CINES under the allocation1 2018-AP01061040 made by GENCI (Grand Equipement National de Calcul Intensif). Authors thank also the Center for the Development of Parallel Scientific Computing (CDCSP) of University of Lyon 1 for providing local computing resources to design the methodology.

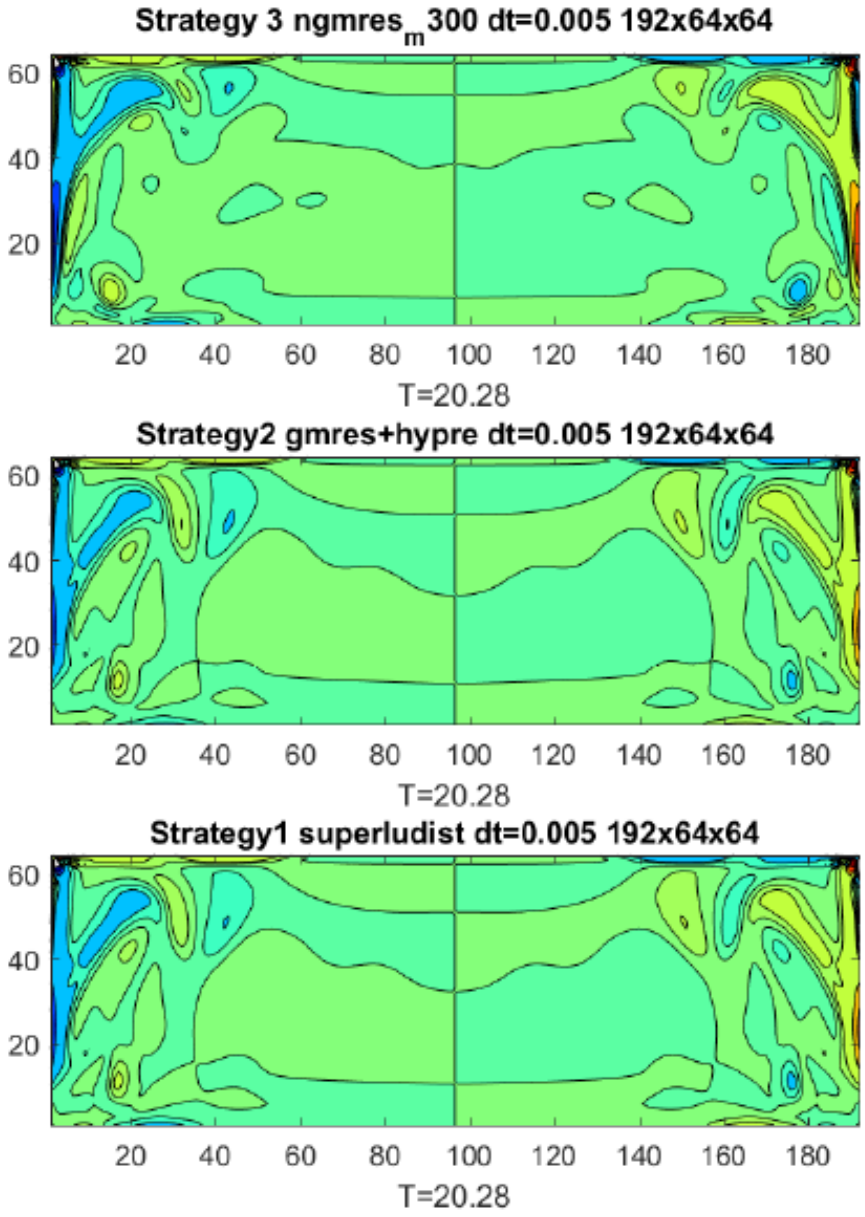

Figure 7: Comparison of the flow behavior for the three strategies (strategy 3 up, strategy 2 middle, strategy 1 bottom) at time t=20.

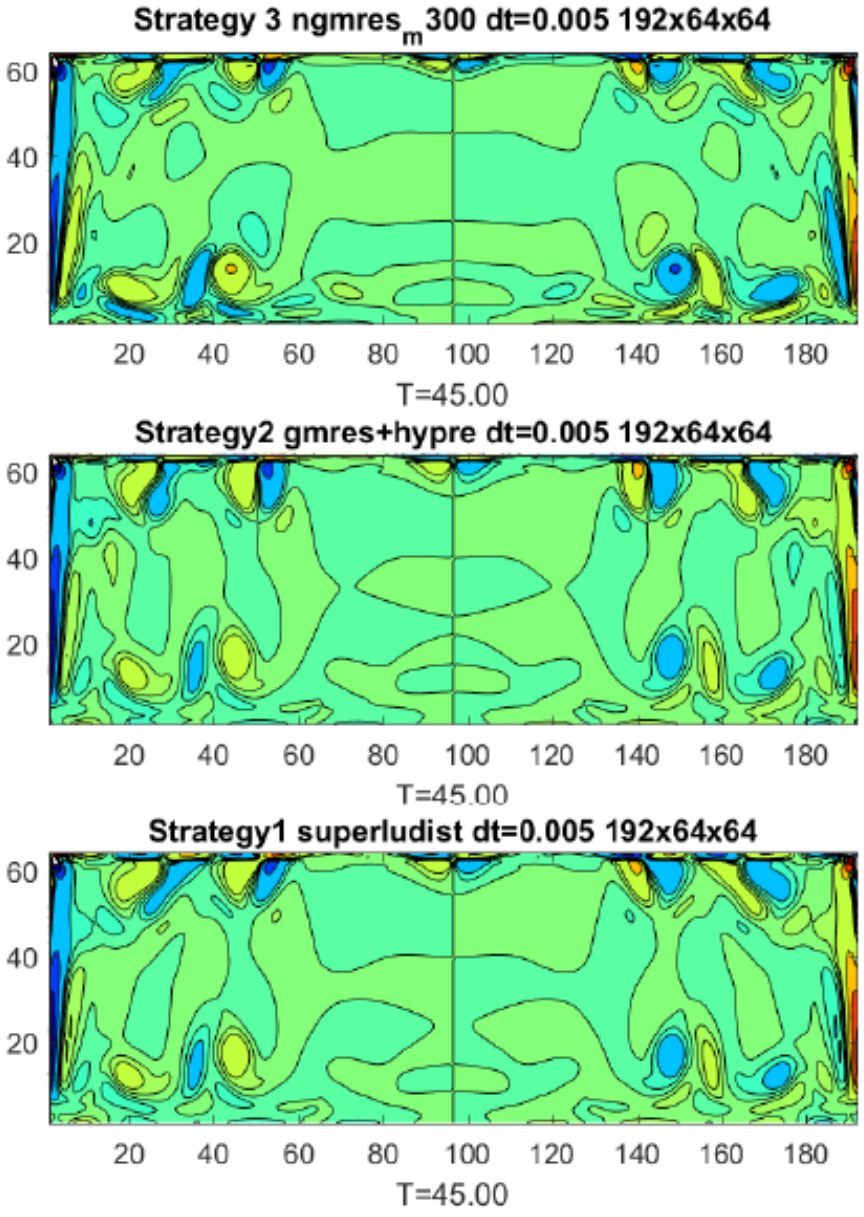

Figure 8: Comparison of the flow behavior for the three strategies (strategy 3 up, strategy 2 middle, strategy 1 bottom) at time t=45.

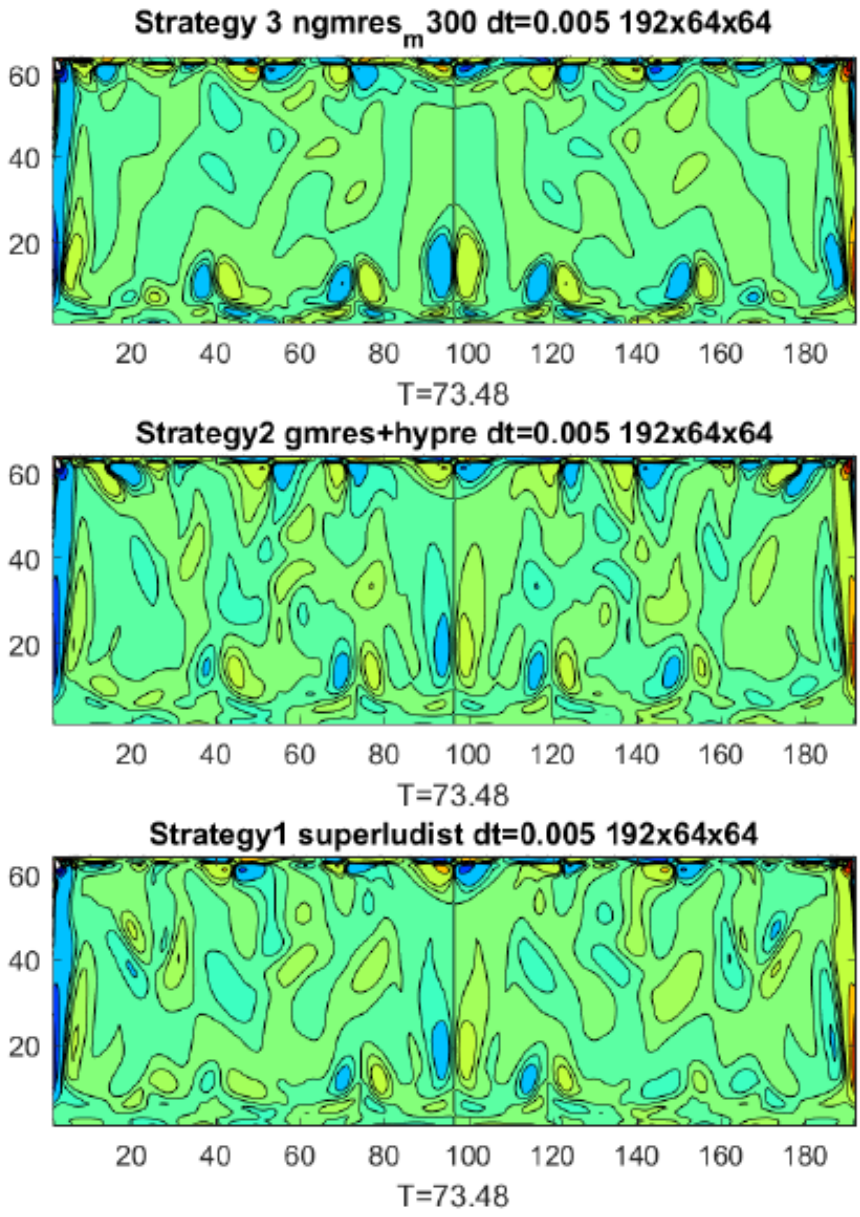

Figure 9: Comparison of the flow behavior for the three strategies (strategy 3 up, strategy 2 middle, strategy 1 bottom) at time t=74.

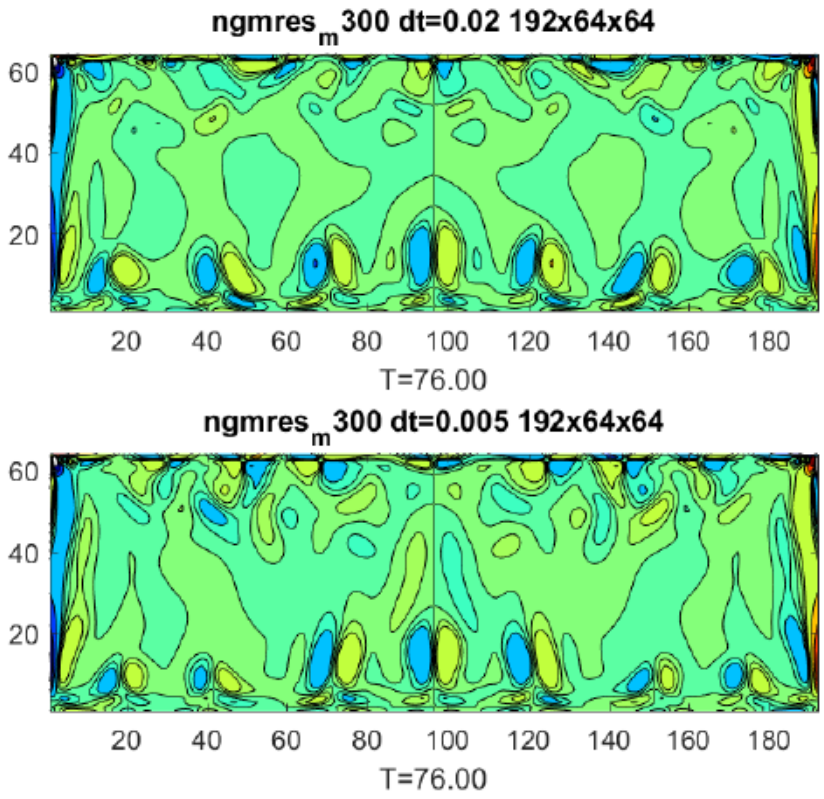

Figure 10: Comparison for strategy 3 (with no CFL time step limitation) for two time steps $dt = 2.10^{-2}$ (top) and $dt = 5.10^{-3}$ (bottom).