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

Damien Tromeur-Dervout

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

D. Tromeur-Dervout[1,2]
[1] Université de Lyon, Institut Camille Jordan UMR 5208 CNRS- Lyon 1,
43 bd du 11 novembre 1918, F-69622 Villeurbanne-Cedex

[2] MAM dept of Polytech Lyon, University Lyon 1,
15 bd Latarjet, F-69622 Villeurbanne-Cedex

July 1 2019

Abstract

This work presents implementation details of three time discretization strategies to solve the 3D incompressible Navier-Stokes equations in velocity-vorticity formulation using PETSc. These time discretization strategies take more and more terms of the system of equation implicitly until to have a fully implicit system. Second order finite differences are used for the discretization in space. The target application is the lid-driven cavity of spanwise aspect ratio 3:1 at a Reynolds number Re = 3200 on uniform non-staggered grids covering all the span. Different features of the PETSC software are investigated allowing fast prototyping of parallel numerical methods adapted to the proposed time discretization strategies. Comparisons on the parallel efficiency, numerical accuracy, and flow behavior for these three time discretization strategies are given.

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

Introduction

This paper is devoted to the use of the Portable, Extensible Toolkit for Scientific Computation (PETSc) for solving the unsteady three-dimensional Navier-Stokes equations written in a conservative form of the velocity-vorticity formulation applied to the lid-driven cavity of spanwise aspect ratio 3:1 at a Reynolds number Re = 3200 on uniform non-staggered grids covering all the span.

From the numerical viewpoint, the three-dimensional flows in a cavity serve as ideal prototype non-linear problems for testing numerical codes. Geometry simplicity

and weIl defined flow structures make these flows very attractive as test cases for new numerical techniques and also provide benchmark solutions to evaluate differencing schemes and problem formulation. We use this test case to study the PETSc features in the teaching of high performance computing to the fifth year engineer students in applied mathematics at Polytech Lyon engineering school (graduate students that already having 240 European Credit Transfer and Accumulation System (ECTS) and an M1 level in the bachelor's master's doctorate system). These classes follow the teaching of the Message Passing Interface library, which facilitates the understanding of the PETSc data distribution and communication through MPI communicators.

We show how different time discretizations of this Navier-Stokes system of equations obtained with writing implicit in the time formulation more and more terms can illustrate on this problem the use of linear and nonlinear solvers features of PETSc software. Further investigations on the French national resources permitted some comparisons (up to 2048 cores) in terms of parallelism speed-up and efficiency of the time discretization strategies .We also investigated the impact of the time discretizing strategies on the flow behavior, showing that having some term explicit in time on the boundary conditions leads to a delay in time in the flow behavior. From our knowledge there no simulation of velocity-vorticity 3D formulation of Navier-Stokes with fully implicit time discretization in the literature. Some works can be found for the fully time implicit 3D lid-driven cubic cavity in primitive variables with Reynolds 1000 [1] or also [2]. Comparison between the flow behaviors in a cubical cavity and in a cavity of spanwise aspect ratio 3:1 is over the scope of this paper. We are mainly focus to see how the time discretisation strategies (semi implicit or fully implicit) with the same set order for the time and the space discretizations impact the flow.

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 discretization and its programming counterpart in the PETSc coding framework. We show three time discretizations 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 discretizations in time strategies while section 5 exhibits the flow behavior with respect of the time strategies. Section 6 concludes this paper.

1 Lid-driven cavity test case and space discretization

1.1 Governing equations

The velocity-vorticity formulation of the unsteady three-dimensional NavierStokes equations is mathematically equivalent to the primitive variables velocity-pressure formulation as demonstrated in [3] [4]. It is often chosen, as opposed to the primitive variable velocity-pressure $\vec{V} - p$ formulation, as transport is the critical physical phenomenon of unsteady viscous flows, it leads to a more natural decoupling of the governing equation by separating the spin dynamic of a fluid particle (represented by the vorticity transport equation) from its translation kinematics (represented by the elliptic velocity problem) [5] [6] and is completely independent of the pressure. Notably, as pointed

by [7], the vorticity transport equation is quasi-linear in vorticity and independent of pressure, whereas the velocity transport equation (momentum equation) is nonlinear in velocity and coupled to the pressure.

The lid-driven cavity is a classical test case in fluid mechanics. Several papers investigated the flow behavior with respect to the Reynolds numbers [8, 9]. The flow in a lid-driven cavity of spanwise aspect ratio 3:1 at a Reynolds number Re = 3200 is a challenging problem as fully transient solutions are expected to show up. The difficulties for meaningful calculations come from both space and temporal discretizations which have to be sufficiently accurate to resolve detailed structures like Taylor-Görtler-like vortices and the appropriate time development. 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-Görtler (TG) vortices in the flow appear after some time. The number of these TG vortices seems to be 7 at time $T = 100$ and 9 at time $T = 200$ [10]. This flow structures are depicted in the figure 1.

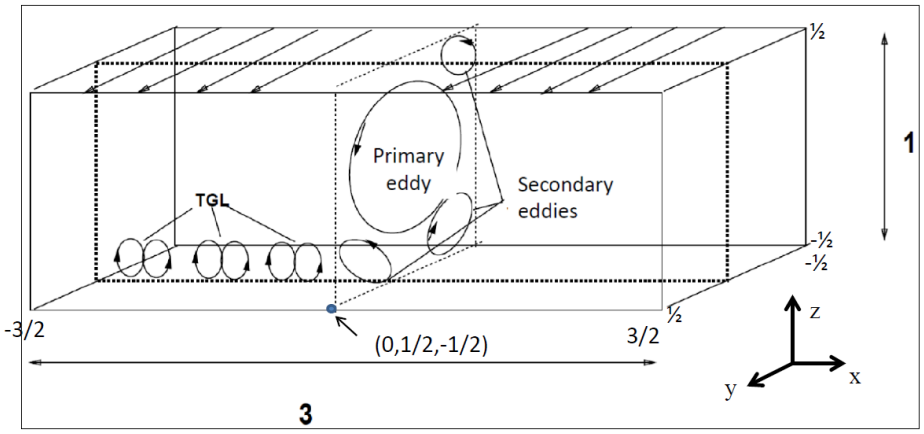

Figure 1: Lid-driven cavity with aspect ratio 3:1

The dimensionless unsteady incompressible Navier-Stokes equations in velocity-vorticity ($\vec{V} - \vec{\omega}$) conservative form are formulated as follows, neglecting body forces:

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

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

The Reynolds number is defined as $Re = \dfrac{U_\infty L}{\nu}$ where ν is the kinematic viscosity, $L = 1$ is the characteristic length of the cavity and $U_\infty = 1$ is the velocity modulus of the lid driven. We have a transport equation with a time derivative for the vorticity $\vec{\omega} \stackrel{def}{=} \vec{\nabla} \times \vec{V}$ while an elliptic equation gives the velocity. These equations have the advantages to avoid the difficult computing of the pressure that guarantees the incompressibility in

primitive variables and also to compute directly the vorticity for rotating flows. The drawback is the definition of the vorticity boundary condition that 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 six scalar equations: three for the velocity and three for the vorticity although some mathematical manipulations can reduced the number of equation for the velocity.

1.2 Space discretization

We discretize the computational domain using the same regular step size in the three dimensions. Then we approximate the differential operators using second order finite differences. This leads to the standard discretization stencil with 7 points for the Laplacian and we use centered second order discretization for the first order partial derivatives of the convection term (for the forthcoming time discretization strategies 1 and 3 and first order decentered finite differences discretization for time discretization strategy 2). This space discretization gives regular data dependencies that links each points to its six neighbor 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 [11]. This effect should be taken in account in further developments. Here, we are mainly focused on the effect of the time discretization on the parallelism efficiency of its PETSc implementation and on the flow behavior.

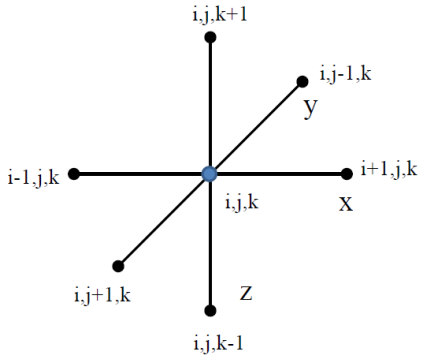

Figure 2: Stencil of the data dependancies.

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 :

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

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 other vorticity boundary conditions:

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

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

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

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

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

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

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

With 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 connects the point on the boundary and the neighbor point within the computational domain. This is an important consideration as all the flow dynamics come from the lid-driven. The other point to highlight is the introduction of a certain delay in time between the vorticity and the velocity if these boundary conditions are not implicitly treated (considering the vorticity and the velocity at the same time step).

2 Time discretization strategies and their coding in PETSc

This lid-driven cavity problem has been implemented in a research parallel code, requiring three 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 [12]. This code implementation was efficient numerically and in elapsed time in the benchmark of codes dedicated to solve this test case in the workshop[10].

A direct implementation of this code in the PETSc framework would not be 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 the tridiagonal operator in each direction of space and to split the vector of unknowns defiined on the 3D distributed mesh in a set of vectors of unknowns associated to a 2D distributed mesh for each space direction. This can be an opportunity of development of PETSc solvers.

Instead, we investigate different features of the PETSc [13, 14] parallel software to implement three time discretization 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. We must notice that the computational domain is simple with structured data. If we need to deal with more complex geometries and still use this structured data we should use some domain decomposition strategies to split the computational domain in block structured data and use some transformation mapping of the block structured meshes to fit the complex geometry. The other solution should be to work with unstructured data, and use some graph partitioning such as par-metis available in PETSc to distribute the vector components between processors.

As the problem geometry is a cavity and the space discretization is with second order finite differences then the velocity and vorticity fields can be represented by structured data like the PETSc distributed array (DA). This is done by the DMDACreate3D function of Table 1. The 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 dependencies and consequently the maximum number of coefficient entries per row in the discretization matrices.

```
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

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 Projection

```
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

solvers KSP one for the velocity (KSPV) and one for the vorticity (KSPW) (see Table 2). 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 (see Table 2).

```
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 3: Part of the PETSc coding for the RHS vorticity function

The main programming effort is to define the functions giving the matrix and the RHS (see Table 3). 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 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 ghost point with neighbor 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 been restored before leaving with DMRestoreLocalVector.

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

```
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 4: PETSc coding of the time loop for strategy 1

solvers associated to different preconditioners. It appears that the best pair KSP/PC for the elliptic equation is the GMRES [15, 16] associated to the HYPRE Algebraic multigrid [17] preconditioner that converges in 5 (respectively 3) iterations per time step for the velocity (respectively the vorticity) for the $192 \times 64 \times 64$ mesh.

As both operators for the vorticity solution and the velocity solution do not depend on the time iteration, we also studied the choice of KSP composed by KSPPREONLY where only the PC is applied with the choice of Gauss factorizaton with the SuperLU_dist package [18]. This is done simply by passing the "-ksp_type preonly" option in the execution order.

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

```
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

KSPSetComputeOperators function to update the matrix vorticity before KSPSolve associated to the vorticity equation (see Table 5).

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.

```
  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

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 dependencies 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.

We also tested several SNES solver the two candidates have been newtonls (for

Newton based nonlinear solver that uses a line search [19]), and ngmres (for Nonlinear Generalized Minimum Residual method [20] [21] that combines m previous solutions into a minimum-residual solution by solving a small linearized optimization problem at each iteration, very similar to the Anderson mixing [22]). While newtonls was performing better to reach a relative tolerance of 1e-6 ,ngmres with up to 300 previous solutions was prefered to reach a relative tolerance of 1e-8 as we will discuss in the flow behavior result.

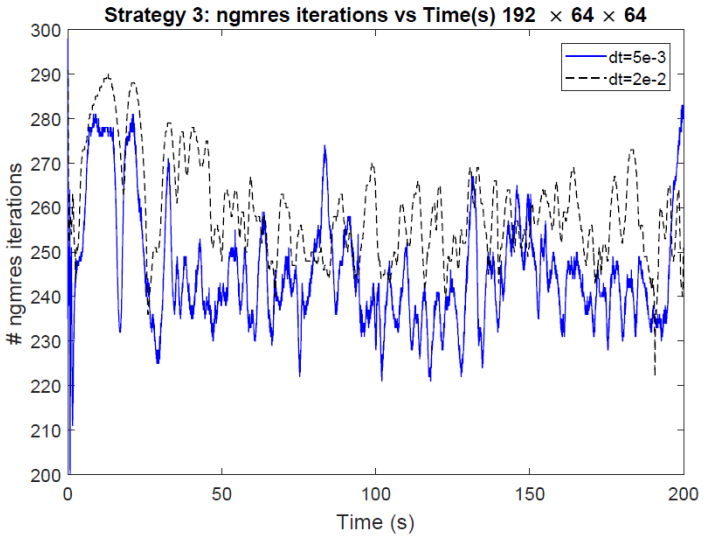

Figure 3: Comparison of the number of ngmres iterations with respect to the simulaton Time (s) for the $192 \times 64 \times 64$ mesh and for two time steps $dt = 0.005$ (full line) and $dt = 0.02$ (dash line)

Figure 3 gives the comparison of the number of ngmres iterations with respect to the simulaton Time (s) for the $192 \times 64 \times 64$ mesh and for two time steps $dt = 0.005$ and $dt = 0.02$. We see that for the two simulations the number of ngmres oscillates with quite the same behavior until $T = 40$. The mean of the number of iteration is 245.19 (respectively 259.17) for $dt = 0.005$ (respectively $dt = 0.02$). We see that the computational effort for ngmres is slightly greater with the time step $dt = 0.02$ than for $dt = 0.005$ but not in the ratio of 4 that the no limitation on the CFL condition allows.

3 Parallelism results

3.1 Computational resources and time measurement

Table 7 presents the characteristics of computational resources used to obtain the parallelism results. We dispose of a local computational resource 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 resource 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 much 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
Resource	laboratory	national (cines)
	∞ hours	50000 hours
Use	speed-up tests	speed-up tests
	flow behavior	

Table 7: Computing resources characteristics

```
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 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 has to declare PetscLogStage and use the PetscLogStagePush(stage) and PetscLogStagePop(); before and after the portion of code that must be monitored 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.

```
---------------------------------------------- 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 observation 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

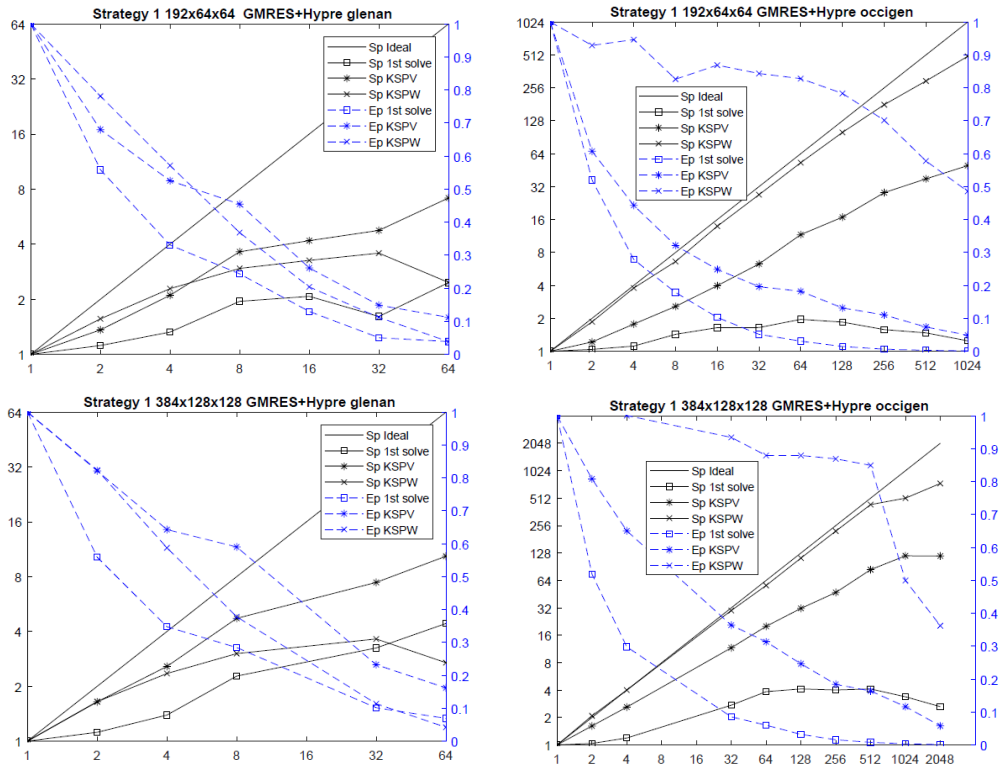

Figure 4: Speed-up (strong scaling) of strategy 1 on glenan (left) and occigen (right) for domain sizes $192 \times 64 \times 64$ (top) and $384 \times 128 \times 128$

Figure 4 gives the speed-up for strategy 1 for two meshes $192 \times 64 \times 64$ and $384 \times 128 \times 128$ 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 $192 \times 64 \times 64$ mesh and 741,86 for 2048 processes for the $384 \times 128 \times 128$ 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

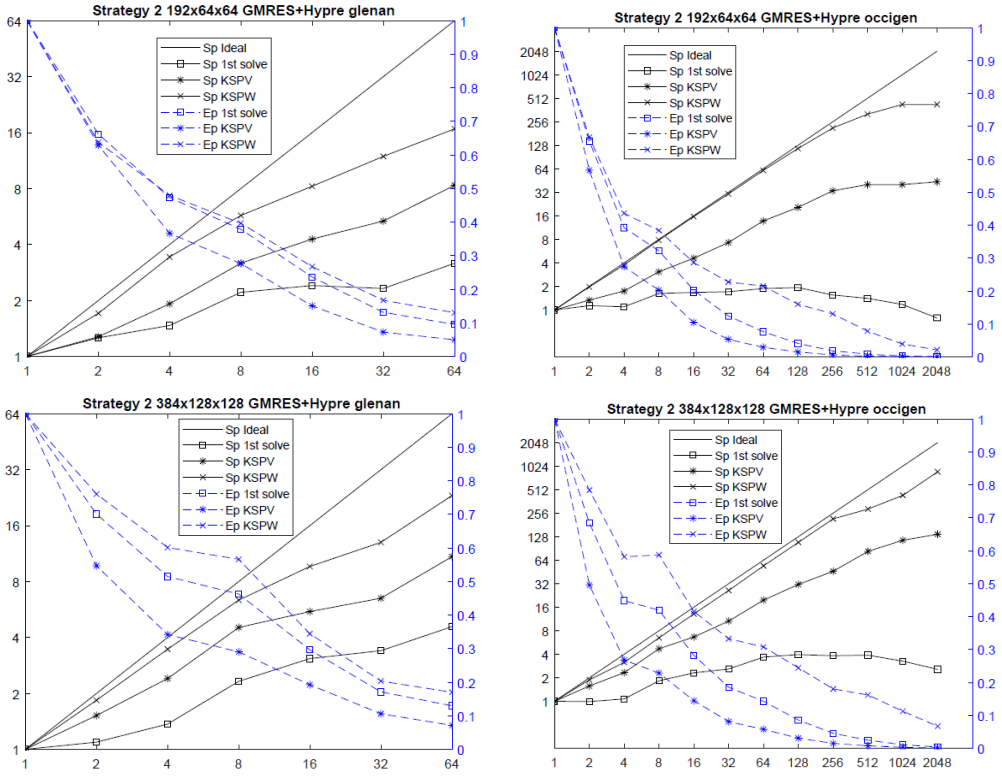

Figure 5: Speed-up (strong scaling) of strategy 2 on glenan (left) and occigen (right) for domain sizes $192 \times 64 \times 64$ (top) and $384 \times 128 \times 128$

Figure 5 gives the speed-up for strategy 2 for two meshes $192 \times 64 \times 64$ and $384 \times 128 \times 128$ 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 6 gives the speed-up for the strategy 3 on occigen for a mesh of $192 \times 64 \times 64$ with Nonlinear GMRES with up to 300 previous solutions. A speed-up of is obtained for 1024 CPUs. We see that the SNES's initialization (Snes init), that corresponds to the

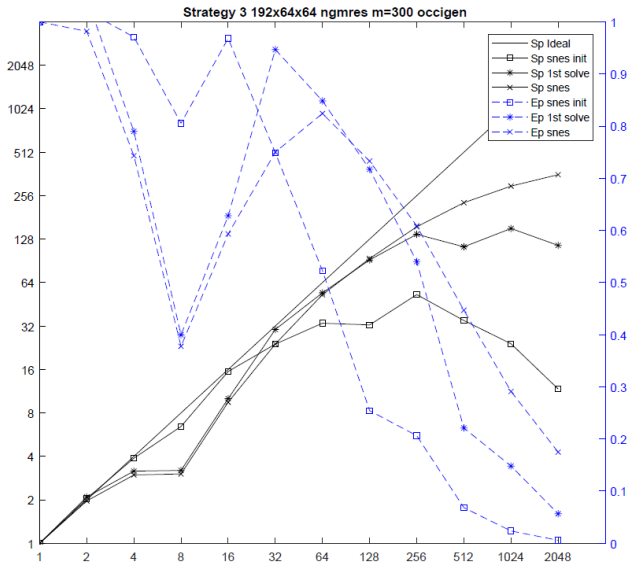

Figure 6: Speed-up for strategy 3 with ngmres and $192 \times 64 \times 64$

buliding of the data structures for the snes, scales until 64 and the scaling performances are degraded after. As this initialization time is small compared to the time associated to the number of time iterates, this result is not significant. We still spearate the first solve to the other iterations as it takes more time. The SNES solving scales well until 64 processes on occigen, and a speed-up of 298 (357 respectively) is obtained on 1024 processes (respectively 2048 processes).

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 74.76 times more time than strategy 2 on one process. This ratio drops to 20 on 256 processes. Strategy 3 still reduces the elapsed time after 256 processes.

- 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 decentered discretization of the convection 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 256 processors the strategy 3 becomes reachable as there is no CFL condition on the time step.

P	S1	S1 LU	S2	S3	comments
1	12.61	-	12.59	941.25	too much even with no CFL
2	9.97	32.89	9.09	479.35	
4	**6.69**	20.48	**6.78**	316.63	S1 vs S2 similar (<>tol)
8	4.56	10.77	3.82	311.45	
16	2.91	8.50	2.52	99.18	
32	1.83	4.54	1.58	39.20	
64	0.99	4.16	0.83	17.84	
128	0.68	2.37	0.56	10.03	
256	0.40	2.42	0.34	6.04	S3 reachable with no CFL
512	0.30	1.83	0.28	4.11	
1024	0.23	5.05	0.28	3.15	
2048	-	-	0.26	2.63	

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

To be complete, we must notice that the elapsed time per point and time step is around $1.60e^{-6}$ to $3e^{-7}$ for strategy 2 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 [10].

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 accuracy 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 $y = 1/6$ where the Taylor-Gortler vortices should appear. We extract the distributed component ω_y 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 Underresolved problem for strategy 3

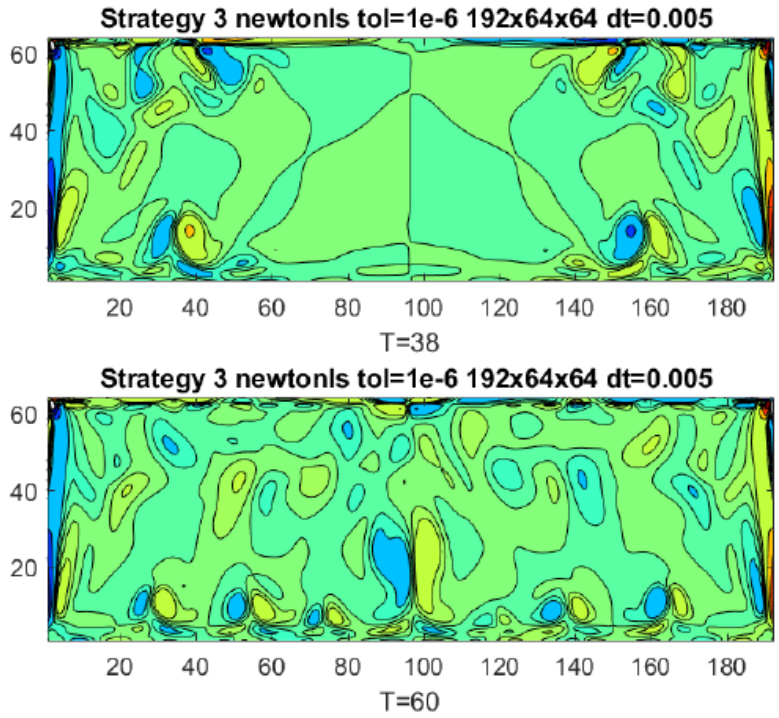

Figure 7: Effect on the undersolve time step solution on the flow behavior: up the Newton LS algorithm for strategy 3 with a $tol = 1e^{-6}$ at $T = 38$ and bottom the lost of the flow symmetry at $T = 60$.

Figure 7 exhibits a problem that occurred 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 loses its symmetry for $T = 60$. This is due to some underresolved solution at some time after $T = 38$. We advocate that as the equations are symmetric as the boundary conditions also, we should keep the symmetry of the flow. The problem was solved by changing the solver with the ngmres with up to 300 previous solutions that allows to solve the problem with a $rtol = 1e-8$.

Figure 7 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 loses its symmetry due to an underresolving 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 $1e^{-8}$.

4.2 Flow behavior comparison for the three strategies

Figures 8 , 9, and 10 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 for strategies 1, 2 and 3 respectively.

Figure 8 exhibits some quite similar results for strategy 1 and strategy 2 with slightly more strong vortex in position i=45 and k=50. The difference coming from taking the vorticity term implicit in time in the convection term but the treatment of the vorticity boundary conditions are unchanged between the two strategies. For the strategy 3, it shows that the vortices attached to the boundaries are more strong, and the flow is slightly in advance compared to the two other strategies as the vortex in position i=20, k=55 is already attached to the vortex on the boundary. We conclude that taking explicit in time the convection term and with more impact taking explicit in time the right hand side of the vorticity boundary conditions introduce a delay on the flow.

The flow behavior at T=45 on Figure 9 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 10 where the strategy 3 exhibits more defined vortices structures.

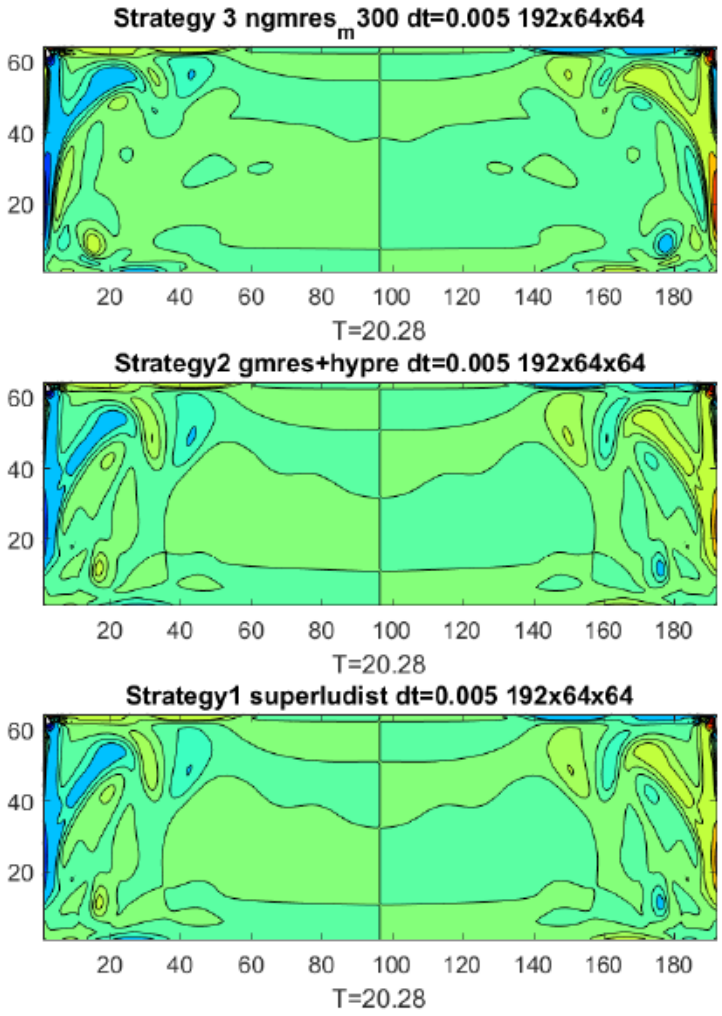

Figure 8: Comparison of the flow behavior for the three strategies (strategy 3 up, strategy 2 middle, strategy 1 bottom) at time t=20.

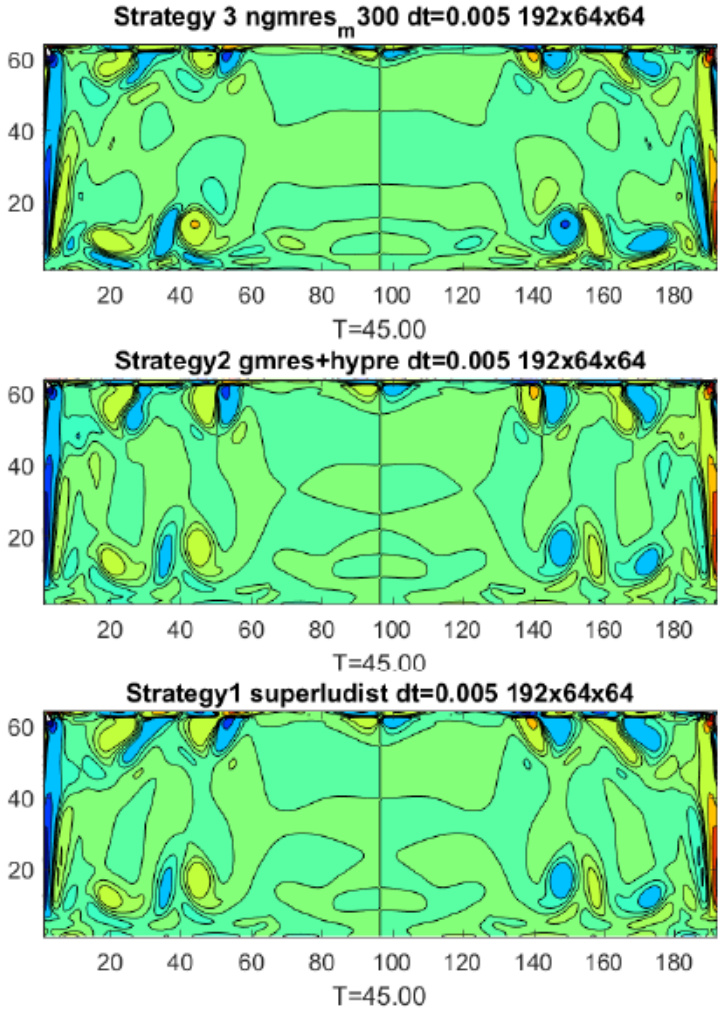

Figure 9: Comparison of the flow behavior for the three strategies (strategy 3 up, strategy 2 middle, strategy 1 bottom) at time t=45.

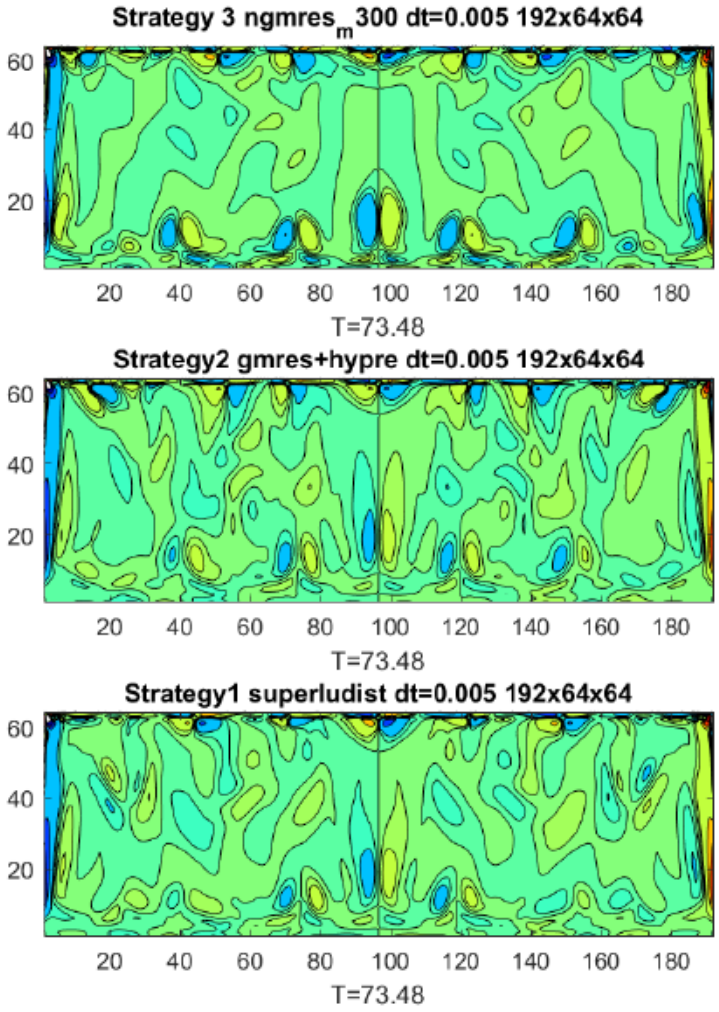

Figure 10: Comparison of the flow behavior for the three strategies (strategy 3 up, strategy 2 middle, strategy 1 bottom) at time t=74.

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

Figure 11 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.

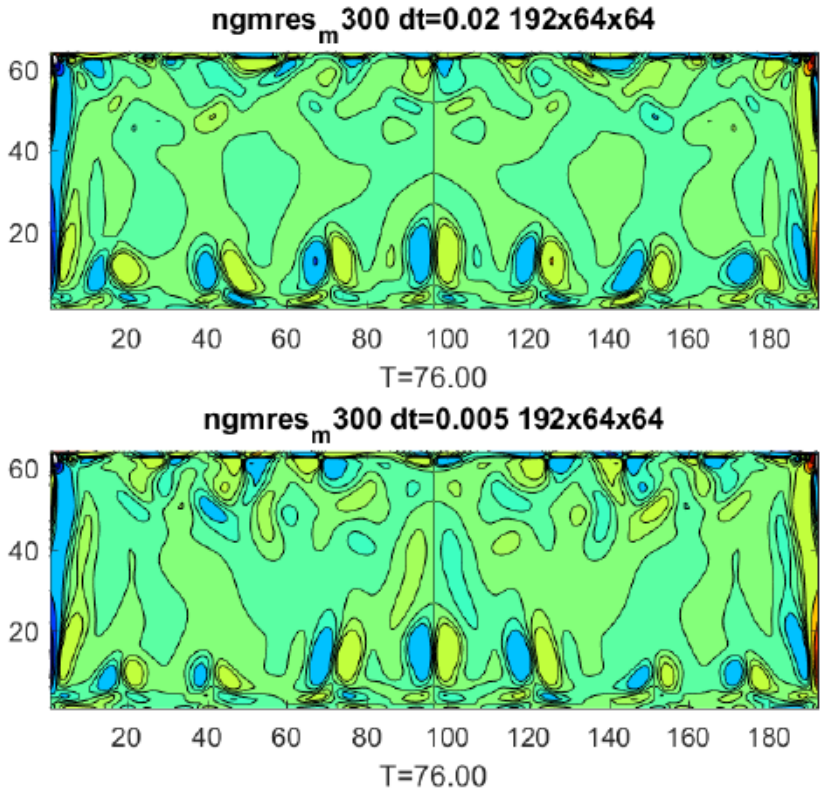

Figure 11: Comparison for strategy 3 (with no CFL time step limitation) for two time steps $dt = 2.10^{-2}$ (top) and $dt = 5.10^{-3}$ (bottom).

5 Conclusions

This paper shows the use of different functionalities 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 writing all the terms of the

vorticity boundary condition at time step t_{n+1} leads to stronger vortices much more than writing all the terms involved in the convective term at same time step t_{n+1}. This conclusion is somewhat foreseeable as all the dynamics of the flow comes from the boundaries. The PETSc library permits us to tests different solvers and to choose the solver and its parameters the best adapted for the computation. It was somewhat surprising in a first approach that taking the convective term implicit 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 [12, 23, 24]. We now have a non-trivial PETSc base test case that can allow us to develop non-linear acceleration methods by further entering in the PETSc coding to define our own SNES solvers, using for example singular value decomposition of previous time step solutions to accelerate the Newton [24]. The discretization could also be improve by several way, with at least better space discretization with compact schemes [25] for example and better discretization in time. The perspective of this work could be also to go further in controling the error on the solution with using for example BDF schemes with adaptive time step of the SUNDIALS [26] library.

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). Author thanks 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.

References

[1] H. C. Elman, V. E. Howle, J. N. Shadid, R. S. Tuminaro, A parallel block multi-level preconditioner for the 3D incompressible Navier-Stokes 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

[2] D. Loghin, A. J. Wathen, Schur complement preconditioners for the Navier-Stokes equations, International Journal for Numerical Methods in Fluids 40 (3-4) (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

[3] O. Daube, J. Guermond, A. Sellier, On the Velocity-Vorticity formulation of the Navier-Stokes equations in incompressible flow, COMPTES RENDUS DE L ACADEMIE DES SCIENCES SERIE II 313 (4) (1991) 377–382.

[4] C. Speziale, On the advantages of the Vorticity Velocity formulation of the equations of fluids-dynamics, JOURNAL OF COMPUTATIONAL PHYSICS 73 (2) (1987) 476–480. doi:{10.1016/0021-9991(87)90149-5}.

[5] T. Gatski, Review of incompressible fluid-flow computations using the Vorticiy Velocity formulation, APPLIED NUMERICAL MATHEMATICS 7 (3) (1991) 227–239. doi:{10.1016/0168-9274(91)90035-X}.

[6] G. A. Osswald, K. N. Ghia, U. Ghia, Direct method for solution of three-dimensional unsteady incompressible navier-stokes equations, in: D. L. Dwoyer, M. Y. Hussaini, R. G. Voigt (Eds.), 11th International Conference on Numerical Methods in Fluid Dynamics, Springer Berlin Heidelberg, Berlin, Heidelberg, 1989, pp. 454–461.

[7] Y. Huang, U. Ghia, G. A. Osswald, K. N. Ghia, Velocity-Vorticity Simulation of Unsteady 3-D Viscous Flow within a Driven Cavity, Vieweg+Teubner Verlag, Wiesbaden, 1992, pp. 54–66. doi:10.1007/978-3-663-00221-5_7.
URL https://doi.org/10.1007/978-3-663-00221-5_7

[8] S. Bhaumik, T. K. Sengupta, A new velocity-vorticity 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

[9] 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

[10] 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.

[11] 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

[12] 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).

[13] 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

[14] 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.

[15] 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}.

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

[17] 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.

[18] X. S. Li, J. W. Demmel, SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems, ACM Trans. Mathematical Software 29 (2) (2003) 110–140.

[19] 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

[20] C. Oosterlee, T. Washio, Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows, SIAM JOURNAL ON SCIENTIFIC COMPUTING 21 (5) (2000) 1670–1690, 5th Copper Mountain Meeting on Itera-

tive Methods, COPPER MOUNTAIN, ALASKA, APR, 1998. doi:{10.1137/S1064827598338093}.

[21] P. R. Brune, M. G. Knepley, B. F. Smith, X. Tu, Composing Scalable Nonlinear Algebraic Solvers, SIAM REVIEW 57 (4) (2015) 535–565. doi:{10.1137/130936725}.

[22] 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

[23] 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

[24] 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

[25] 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

[26] 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.