

18. On the Interface and Two-Level Preconditioners in Newton-Schwarz Method

Daniel Lee¹

Introduction

This paper is concerned with parallel computation in solving the convection-diffusion equation and the incompressible Navier-Stokes equation via Newton-Schwarz method, a nonlinear domain decomposition (DD) method. Various preconditioners are investigated here. An interface problem is tackled as a preconditioner for nonlinear block Jacobi DD approach, with an optional fine level interface problem solved as further preconditioner. Also a (global) coarse level preconditioner is considered. Examined also is the relaxation type preconditioner. Such preconditioned nonlinear DD methods exhibit impressive improvement over the basic non-preconditioned parallel Newton-Jacobi method.

A general review on Newton-Schwarz method is [GEMT98]. Our setup has the advantages of both the overlapped and the nonoverlap DD approach. The subdomain variables form a (nonoverlap) partition of the whole global system of equations. Solving the interface problem is regarded as a preconditioner to all subproblems. We note that the interface variables were excluded from subproblems in [LC98].

We describe in later sections the problem and solution procedure, the test cases and results, and a brief conclusion.

Solution Procedure and Newton-Schwarz

Following previous work [LC98], boundary-fitted cell-center type finite volumes with collocated grids were assumed for geometry discretization. Test problems ([Wan89]) are considered in integral form, involving properly defined numerical fluxes. All the definite integrals are further discretized ([Lee99], [JYL99]) as weighted averages involving the primitive (and the flux) variables at neighboring cells. To double-check our numerical observation, we solved both the coupled system, consisting of the continuity and the momentum equations, and also in a decoupled way (PISO, [Iss85]) for the system consisting of the momentum and the Pressure-Poisson equation. We tested also the Burgers equation. The discrete global nonlinear system is decomposed into partition of (nonoverlap) blocks of equations. Basic Parallel Newton-Jacobi (PNJ) method can be then carried out accordingly.

Interface Preconditioner

We regard an interface preconditioner as solving the interface problem on the interface B , before each nonlinear block Jacobi iteration. We prepose the following for further discussion.

¹National Center for High-Performance Computing, Taiwan, R.O.C., c00dle00@nchc.gov.tw

Procedure IPPNJ (Interface-Preconditioned Parallel Newton-Jacobi) :

Do While $\{|X^{new} - X^{old}| \geq \text{global-tolerance}\}$

Step 1 : Solve the discrete algebraic nonlinear system $F_i(x) = 0, x \in \Omega_i$
for all subdomain problems in parallel.

Step 2a : Set up the interface problem, with information communicated
from each subdomain.

Step 2b : Solve the nonlinear system for the interface problem.
 $F_B(x) = 0, x \in B.$

Step 2c : Update via communication the interior boundary conditions
for all subproblems with the interface variables just solved.

End Do

The interface problem is relatively small in size, easier to solve by an approximate matrix-free Newton-GMRes method [BS90], and the solution is supposed to offer more accurate interior boundary condition at the interface variables in an efficient way. This is the spirit of IPPNJ. That is, acceleration on (only) the interface variables yields a preconditioner for subsequent DD iterations. In implementation the interface preconditioner is squeezed into after and before two consecutive block Jacobi iterations. We mention that, in our setup, an interface-preconditioned Newton-Schwarz method corresponds actually to a mixing of nonlinear analogue of Schwarz type and Schur-complement type linear DD methods.

Preconditioned Block Newton-Schwarz Procedure

For general application, we recast the discrete system as $\Phi(u) = rhs$, where $\Phi = (\Phi_1, \dots, \Phi_{n_d})^T$, $u = (u_1, \dots, u_{n_d})^T$, with $u_i \in R^{d_s} \equiv X_s$. Here s denotes a subdomain (and a subproblem), d_s denotes the dimension of subproblem (on subdomains). We assume equal size of the subproblems for simplicity. The space X_s is therefore where a solution to the discrete subproblem resides. Consider the subproblems $\widetilde{\Phi}_i(\widetilde{u}_i) = \widetilde{rhs}_i$ and $J(\Phi_i, u_i) = \frac{\partial \Phi_i}{\partial u_i}$. We assume regularity of such portion of the global Jacobian. We describe a more general version of previous procedure in details with these notation, based on partition of nonoverlap subdomains and some certain order of the equations and variables.

Procedure PPNJ (Preconditioned Parallel Newton-Jacobi) :

Do while (global convergence achieved or maximum DD iteration exceeded)

1. *Do $i = 1, \dots, n_d$ (in parallel)*
 - a. *set \tilde{u}_i by u and canonical projection*
 - b. *set \widetilde{rhs}_i*
 - c. *obtain an approximated solution to $\widetilde{\Phi}_i(\tilde{u}_i) = \widetilde{rhs}_i$*
 - d. *evaluate for local convergence the residual $\tilde{r}_i := \widetilde{rhs}_i - \widetilde{\Phi}_i(\tilde{u}_i)$*
 - e. *evaluate for local convergence the difference $diff_i := \|\tilde{u}_i - \widetilde{u}_{i_{sav}}\|$*
- End Do*
2. *Update global u by communicating the \tilde{u}_i among relevant processors*
3. *Check global convergence by evaluating $\max_i diff_i$ or $\|\Phi(u)\|$ on $X_{n_s}^{n_d}$*
4. *If global convergence is satisfied, then break. Otherwise*
 - a. *Do an optional accelerator or preconditioner, such as interface preconditioner or global coarse level preconditioner.*
 - b. *Update interface variables by communication, and approximation schemes in case of a two level preconditioner.*
5. *Update the DD iteration counter.*

End Do

Other Preconditioners

The solution procedure in solving the interface problem and all subproblems are identical by default. However, the interface problem can be solved optionally with a fine-level interface preconditioner. This is affordable if the two level setup for the interface problem remains relatively cheaper than the other subproblem both in storage and computation. It is therefore a natural iterative refinement procedure based on the consideration of load balance. A global coarse level preconditioner is also a good choice, hopefully produces global information update as motivated by the linear DD theory. Relaxation type strategy, in simple or hybrid form, can be used to accelerate the convergence on the interface variables, as to provide an interface preconditioner to the global problem. Furthermore, one can even over-relax the setup of the interface problems resulting in an accelerated interface preconditioner [LY00].

Numerical Results and Discussions

All cases were tested on a PC cluster at NCHC. The global region consists of nine subdomains in cases 1-5, and eight in case 6. Reynolds number is 1 in case 1, and 10 in cases 2-6. More about test is given in Table 1. The cpu time spent is shown in Table 2. Only partial results in accuracy and convergence are shown (Figures 1-12), due to limitation of space. Four-subdomain cases are also tested, showing behavior similar to what we described below.

Case 1 : Four DD methods are tested : Parallel Newton-Jacobi (PNJ), Interface-Preconditioned Parallel Newton-Jacobi (IPPNJ), Coarse-level-

Preconditioned Parallel Newton-Jacobi (CPPNJ), and Fine-level Interface-Preconditioned Parallel Newton-Jacobi (FIPPNJ). Convergence up to nine digits is enforced. The maximum DD-iterations is set to 100 to examine the stability. Although in practice this may be much smaller.

Shown in Figures 1-2 are the accuracy and convergence in relative sense. All yield very stable discrete dynamics. CPPNJ the fastest while PNJ the slowest. IPPNJ and FIPPNJ converges at about the same rate. About accuracy, CPPNJ is the worst and IPPNJ achieves the same accuracy both in absolute and relative sense, and takes about half iterations as PNJ. This is similar to what between classical Gauss-Seidel and Jacobi iterations. We point out that the precision achieved with the two-level preconditioners depend certainly on local interpolation or approximation, and are confined therefore by the spatial grid resolution. It is witnessed that FIPPNJ saturated at some earlier stage, and was forced therefore to iterate 100 iterations. This hurt in the cpu-time contest (Table 2). Although, the convergence history does justify the stableness in computation. The FIPPNJ result seems not as appealing as that of IPPNJ. Therefore FIPPNJ is excluded in other test cases below.

Case 2 : PNJ is validated and applied to both the decoupled and the coupled approach, i.e., with or without PISO. Then we will test later in case 4 our proposed preconditioners within these two different approaches. Our algorithms and implementation is justified (Figure 3 and 4). The accuracy when without PISO is only slightly inferior to that with PISO. The time spent in computation of third order derivatives in Pressure-Poisson equation for the PISO formulation seems a disadvantage.

Case 3 : IPPNJ is applied with or without PISO. The findings, Figures 5 and 6, are similar to case 2.

Case 4 : The coupled system without PISO is solved with PNJ, IPPNJ and CPPNJ methods. CPPNJ is the most accurate and IPPNJ also outperforms PNJ (Figure 7). The convergence in the relatively severe maximum norm (Figure 8), although not as smooth as with the normalized two-norm (not shown), does indicate the relative spirits in these methods. Very heavy communications are seen on the coarse preconditioner, in the pre- and post- data processing in solving the global coarse problem. Therefore we exempted CPPNJ from subsequent cases.

Case 5 : Here we solve with PISO and compare PNJ and IPPNJ. The latter converges faster and is more accurate (Figures 9-10).

Case 6 : Relaxation type preconditioners are examined with various relaxation parameters. The kind of monotonicity of the effectiveness in accuracy (Figure 12) and convergence (Figure 11) is obviously seen. More thoughts along this line, including combination of different strategy and even working on over-relaxed interface problem, is investigated in [LY00].

Conclusion

Several preconditioners are designed for the interface problem in a Newton-Schwarz approach. With these the basic parallel block Jacobi procedure either converges faster or is more accurate. We found that IPPNJ converges faster while achieving same or better accuracy and costs less computation time, and that CPPNJ and FIPPNJ achieve moderate precision and converge faster in terms of the number of DD iterations, but

both with apparently heavier communication overhead. The choice certainly depends on, among others, the spatial resolution and the required precision in computed results. We believe future technology improvement in system architecture will resolve the communication inefficiency to a large extent.

Acknowledgement

This work was supported by the National Science Council, Taiwan, R.O.C., under contract NSC-88-2113-E321-003. Some of the ideas was developed at CERFACS with helpful suggestions from Professor Iain Duff and Mr. Luc Giraud. Very sincere gratitude is expressed here.

References

- [BS90]Peter N. Brown and Yousef Saad. Hybrid Krylov methods for nonlinear system of equations. *SIAM J. Sci. Stat. Comput.*, 11:450–481, 1990.
- [GEMT98]W. D. Gropp, D. E. Keyes, L. C. McInnes, and M. D. Tidriri. Globalized Newton-Krylov-Schwarz algorithms and software for parallel implicit CFD. Technical Report 98-24, ICASE, August 1998. NASA/CR-1998-208435.
- [Iss85]R. I. Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62:40–65, 1985.
- [JYL99]Kang C. Jea, Mulder Yu, and Daniel Lee. A study on finite volume methods. In D. R. Kincaid et. al. eds, editor, *Iterative Methods on Scientific Computation II*. IMACS, 1999.
- [LC98]Daniel Lee and Chih-Hua Chen. A study on domain based parallel flow computation. In *Proceeding of the Fifth National Conference in Computational Fluid Dynamics*, Taiwan, R.O.C., August 27 1998.
- [Lee99]Daniel Lee. A new approach to finite volume-finite difference methods. *Chinese Journal of Numerical Mathematics and Applications*, 21(2):96–102, 1999.
- [LY00]Daniel Lee and Mulder Yu. Parallel flow computation with domain based SOR type interface preconditioner. In *Proceedings of the Fourth International Conference on High-Performance Computing in the Asia-Pacific Region (HPC-Asia 2000)*, Beijing, China, May 2000. To appear.
- [Wan89]C. Y. Wang. Exact solutions of the unsteady Navier-Stokes equations. *Appl. Mech. Rev.*, 42:269–282, 1989.

Table 1: Parameters of test runs with supremum norm; range of x , y and z in cases 1 and 6 is $(0.0, 1.0)$; range of x and y in cases 2-5 is $(1.0, 2.0)$.

| case | eq. | DD_it | ht | ht_CFL | nx, ny, nz | PISO | method |
|------|------------|-------|------|----------|------------|------|---------|
| 1 | 2D Burgers | 100 | 1e-3 | 9.00e-06 | 60, 60 | 0 | various |
| 2 | 2D NS | 20 | 1e-3 | 5.62e-06 | 240, 240 | 0, 1 | PNJ |
| 3 | 2D NS | 20 | 1e-3 | 5.62e-06 | 240, 240 | 0, 1 | IPPNJ |
| 4 | 2D NS | 20 | 1e-3 | 5.62e-06 | 240, 240 | 0 | various |
| 5 | 2D NS | 20 | 1e-3 | 5.62e-06 | 240, 240 | 1 | various |
| 6 | 3D Burgers | 50 | 1e-2 | 4.16e-03 | 10, 10, 10 | 0 | PNJ |

Table 2: Cpu time in cases 1-6.

| case | iter | total sub. solver | total sub. solver | other overhead | total cpu time | |
|------|---------------|-------------------|-------------------|----------------|----------------|----------|
| 1 | PNJ | 81 | 4.05e+02 | 4.52e+02 | 9.00e+00 | 8.66e+02 |
| | IPPNJ | 38 | 2.43e+02 | 2.07e+02 | 3.00e+00 | 4.53e+02 |
| | CPPNJ | 11 | 1.01e+02 | 6.96e+01 | 2.40e+00 | 1.81e+02 |
| | FIPPNJ | 100 | 1.88e+03 | 4.34e+02 | 6.00e+00 | 2.32e+03 |
| 2 | PISO 0 | 20 | 2.90e+03 | 3.28e+03 | 8.00e+01 | 6.26e+03 |
| | PISO 1 | 20 | 3.42e+03 | 1.34e+04 | 8.00e+01 | 1.69e+04 |
| 3 | PISO 0 | 20 | 2.50e+03 | 2.78e+03 | 8.00e+01 | 5.36e+03 |
| | PISO 1 | 20 | 4.56e+03 | 2.38e+04 | 1.50e+02 | 2.84e+04 |
| 4 | PNJ | 20 | 2.86e+03 | 3.22e+03 | 9.00e+01 | 6.15e+03 |
| | IPPNJ | 20 | 2.44e+03 | 2.74e+03 | 9.00e+01 | 5.25e+03 |
| | CPPNJ | 20 | 1.23e+04 | 3.48e+03 | 1.20e+02 | 1.59e+04 |
| 5 | PNJ | 20 | 3.34e+03 | 1.34e+04 | 1.10e+02 | 1.69e+04 |
| | IPPNJ | 20 | 1.52e+04 | 1.44e+04 | 1.00e+02 | 2.97e+04 |
| 6 | without SOR | 10 | 6.07e+00 | 2.28e+01 | 8.73e+00 | 3.76e+01 |
| | over with 1.1 | 9 | 4.86e+00 | 2.00e+01 | 8.24e+00 | 3.31e+01 |
| | over with 1.2 | 10 | 5.52e+00 | 2.25e+01 | 8.48e+00 | 3.65e+01 |
| | over with 1.3 | 12 | 5.32e+00 | 2.68e+01 | 9.19e+00 | 4.12e+01 |
| | over with 1.4 | 19 | 9.39e+00 | 4.31e+01 | 1.21e+01 | 6.46e+01 |

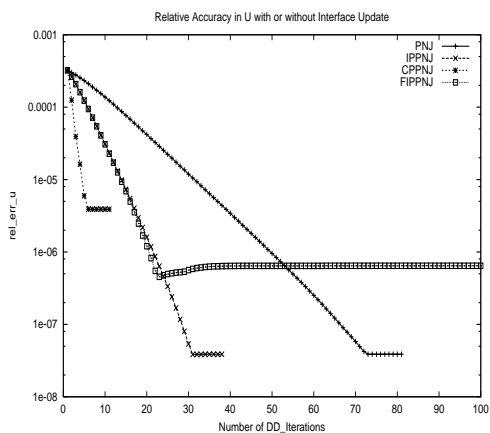


Figure 1: Relative accuracy in solving Burgers' eq. in case 1.

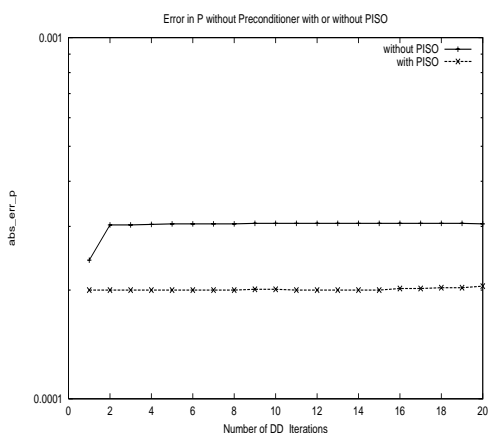


Figure 3: Accuracy in p with or without PISO in case 2.

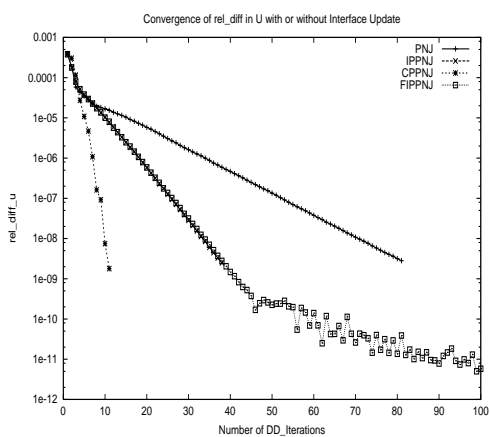


Figure 2: Relative convergence in solving Burgers' eq. in case 1.

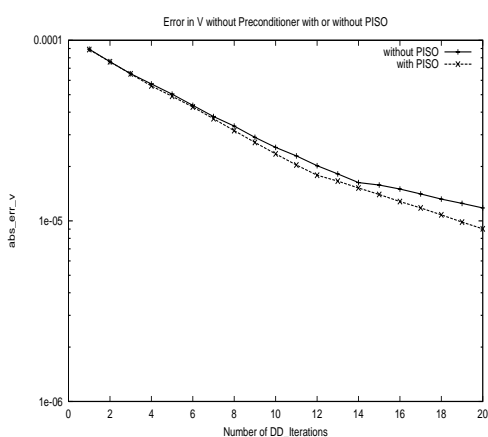


Figure 4: Accuracy in v with or without PISO in case 2.

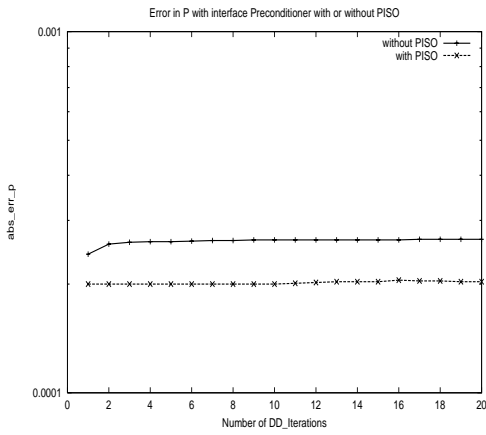


Figure 5: Accuracy in p with or without PISO in case 3.

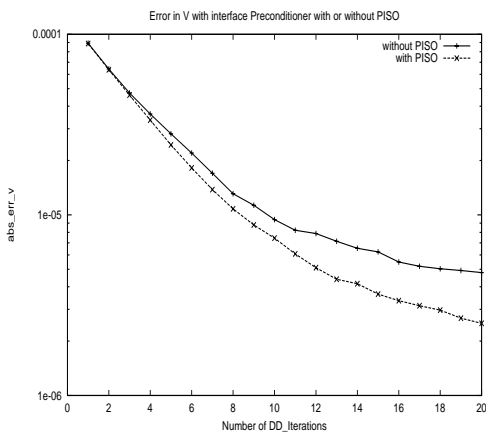


Figure 6: Accuracy in v with or without PISO in case 3.

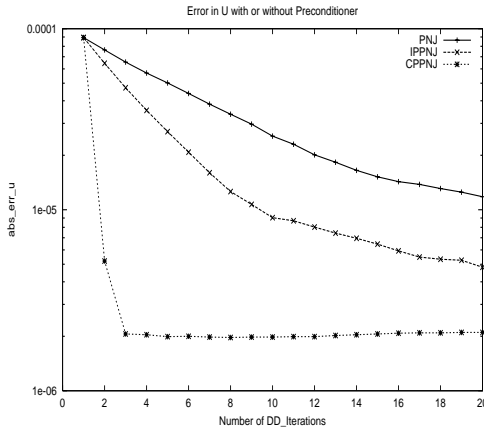


Figure 7: Accuracy in u with or without preconditioner in case 4.

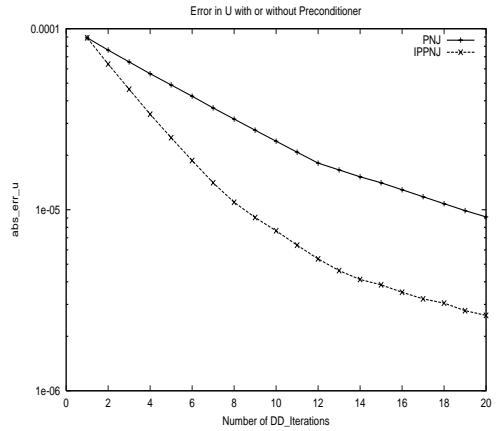


Figure 9: Accuracy in u with or without preconditioner in case 5.

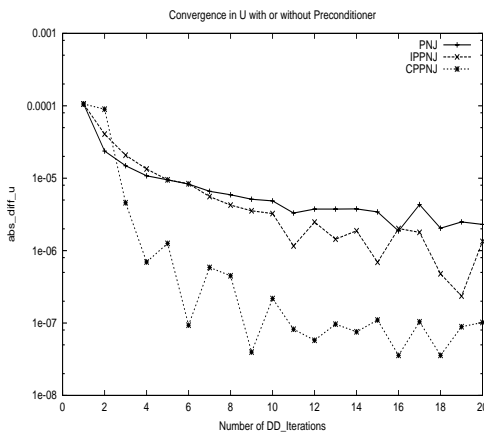


Figure 8: Convergence in u with or without preconditioner in case 4.

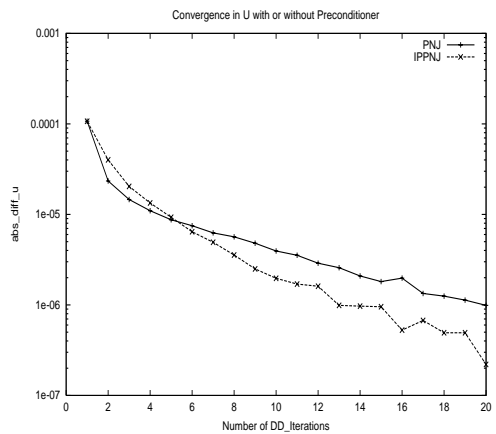


Figure 10: Convergence in u with or without preconditioner in case 5.

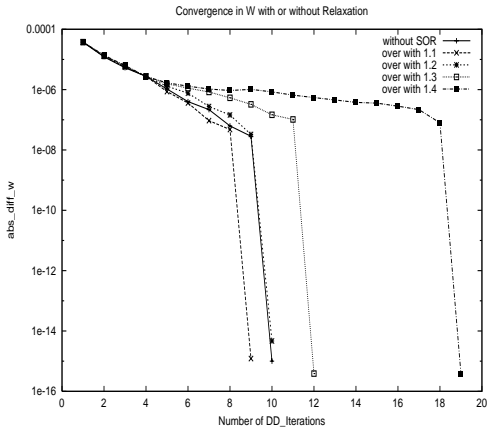


Figure 11: Convergence in w in solving 3D NS eq. with or without relaxation in case 6.

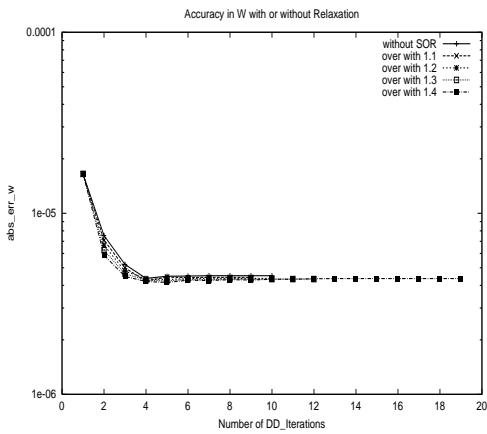


Figure 12: Accuracy in w in solving 3D NS eq. with or without relaxation in case 6.

Appendix

Analytic descriptions of the equations and solutions for our test are given here for easy reference. Dirichlet type boundary data are adopted for the test runs in this paper. We refer to [Wan89] for more explanation on the equations.

(1) 2D Burgers' equation:

$$u_t + u(u_x + u_y) - \frac{1}{Re}(u_{xx} + u_{yy}) = 0, \quad (1)$$

and one solution is :

$$u = \frac{1}{1 + e^{\frac{Re(2x+2y-2t)}{4}}}. \quad (2)$$

(2) 2D Navier-Stokes equation:

Continuity equation:

$$u_x + v_y = 0, \quad (3)$$

X-momentum equation:

$$u_t + uu_x + vv_y = -p_x + \frac{1}{Re}(u_{xx} + u_{yy}), \quad (4)$$

Y-momentum equation:

$$v_t + uv_x + vv_y = -p_y + \frac{1}{Re}(v_{xx} + v_{yy}), \quad (5)$$

and one solution is :

$$u = -\cos(x) * \sin(y) * e^{\frac{-2t}{Re}}, \quad (6)$$

$$v = \sin(x) * \cos(y) * e^{\frac{-2t}{Re}}, \quad (7)$$

$$p = 10.0 - \frac{1}{4}(\cos(2x) + \cos(2y)) * e^{\frac{-4t}{Re}}. \quad (8)$$

(3) 3D Burgers' equation:

$$u_t + u(u_x + u_y + u_z) - \frac{1}{Re}(u_{xx} + u_{yy} + u_{zz}) = 0, \quad (9)$$

and one solution is :

$$u = \frac{1}{1 + e^{\frac{Re(2x+2y+2z-3t)}{4}}}. \quad (10)$$

