# A GPU Parallel Solver for 3D Incompressible Navier-Stokes Equations Discretized by Stabilized Finite Element Formulations

○ Huynh Quang Huy Viet, Graduate School of Environ. and Life Sci., Okayama Uni., JST-CREST
Suito Hiroshi, Graduate School of Environ. and Life Sci., Okayama Uni., JST-CREST

**Abstract**

The discretization of the Navier-Stokes equations stabilized finite element formulations leads to a large and sparse non-symmetric system of linear equations. The numerical solution of non-symmetric linear systems has often been solved by using iterative methods such as the Bi-conjugate gradient stabilized method (Bi-CGStab) or the Generalized minimal residual method (GMRES). Among the variations of the Bi-CGStab algorithm proposed by various researchers, the GPBi-CG algorithm has been proven to have very good convergence behavior. In this paper, we propose an efficient GPU implementation of a parallel solver based on the GPBi-CG algorithm for 3D Navier-Stokes equations discretized by stabilized finite element formulations.

## 1. Introduction

In solving the Navier-Stokes equations by the finite element method for simulation of incompressible flows, there are instabilities which come from the presence of advection terms and/or the high Reynolds number of flows. Hughes and Tezduyar et al. [1, 2, 3, 4] proposed stabilized finite element formulations for incompressible flows. The stabilization in solving the Navier-Stokes equations is achieved by adding two stabilization terms to the Galerkin formulations of the Navier-Stokes equations. The first term is the SUPG (streamline upwind/Petrov-Galerkin) term and the second term is the PSPG (pressure stabilizing/Petrov-Galerkin) term. This stabilized finite element method has been proven to be very effective in simulation of incompressible flows.

The discretization of the Navier-Stokes equations by the stabilized finite element formulations leads to a large and sparse non-symmetric system of linear equations. The numerical solution of non-symmetric linear systems has often been solved by using iterative methods such as the Bi-conjugate gradient stabilized method (Bi-CGStab) or the Generalized minimal residual method (GMRES). The Bi-CGStab is a good algorithm especially when the available memory is relatively small in relation to the size of system memory. Among the variations of the Bi-CGStab algorithm proposed by various researchers, the GPBi-CG algorithm [5] has been proven to have very good convergence behavior. The GPBi-CG algorithm can be parallelized by using multicore platforms such as CPU and GPU. Recently, since modern GPUs have many processors or cores, GPU computing has been recognized as a powerful platform to achieve high-performance in scientific computation and also in simulation of incompressible flows.

In this paper, we propose an efficient GPU implementation of a parallel solver based on the GPBi-CG algorithm for 3D Navier-Stokes equations discretized by the SUPG/PSPG stabilized finite element formulations.

## 2. Governing equations

Let $\Omega \in R^{n_d}$ be the spatial domain where $n_d$ is the number of spatial dimensions. We consider the following dimensionless form of the Navier-Stokes equations [7]:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{Re}\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_i}{\partial x_j}\right) \quad in \quad \Omega, \tag{1}$$

$$\frac{\partial u_j}{\partial x_j} = 0 \quad in \quad \Omega. \tag{2}$$

Here and in what follows in the next section, we use the summation convention over repeated lower indices. The indices have the values of 1, 2, 3; $u_i$ is the velocity in the $i^{th}$ dimension, $p$ is the pressure , and $Re$ is the Reynolds number.

## 3. Finite element formulations with the SUPG/PSPG stabilizations

Let us discretize the spatial domain $\Omega$ by elements $\Omega^e$, $e = 1, 2, ..., n_{el}$. Let $S_u, V_u, S_p$ be finite element interpolation function spaces for the velocity and the pressure. The stabilized finite element formulations of the equations (1)-(2) with the SUPG/PSPG stabilization terms can be written as follows [4, 7]: find $u_i \in S_u$ and $p \in S_p$ such that $\forall \omega_i \in V_u$ and $\forall q \in V_p$:

$$\begin{aligned}\int_\Omega \omega_i\left(\frac{\partial u_i}{\partial t} + \bar{u}_j\frac{\partial u_i}{\partial x_j}\right)d\Omega - \int_\Omega \frac{\partial \omega_i}{\partial x_i} p d\Omega \\ + \int_\Omega \frac{1}{Re}\frac{\partial \omega_i}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)d\Omega \\ + \sum_{e=1}^{n_{el}} \int_{\Omega_e} \tau \bar{u}_k \frac{\partial \omega_i}{\partial x_k}\left(\frac{\partial u_i}{\partial t} + \bar{u}_j\frac{\partial u_i}{\partial x_j} + \frac{\partial p}{\partial x_i}\right)d\Omega = 0,\end{aligned} \tag{3}$$

$$\int_\Omega q\frac{\partial u_i}{\partial x_i}d\Omega + \sum_{e=1}^{n_{el}} \int_{\Omega_e} \tau \frac{\partial q}{\partial x_i}\left(\frac{\partial u_i}{\partial t} + \bar{u}_j\frac{\partial u_i}{\partial x_j} + \frac{\partial p}{\partial x_i}\right)d\Omega = 0, \tag{4}$$

where $\tau$ is the SUPG/PSPG stabilization parameter. $\tau$ was defined as follows:

$$\tau = \left[\left(\frac{2}{\Delta t}\right)^2 + \left(\frac{2\|\bar{u}_i^e\|}{h_e}\right)^2 + \left(\frac{4}{Re\, {h_e}^2}\right)^2\right]^{-\frac{1}{2}}. \tag{5}$$

Here, the $\Delta t$ is the time step size of the computation, $\bar{u}_i^e$ is the element advection velocity, $h_e$ is the element length. The norm of the element advection velocity was defined as follows:

$$\|\bar{u}_i^e\| = \left[\sum_{i=1}^{n_d} (\bar{u}_i^e)\right]^{-\frac{1}{2}}, \tag{6}$$

where $n_d$ is the number of spatial dimensions. The element length $h_e$ was defined as follows:

$$h_e = 2\left[\sum_{\alpha=1}^{n_{en}} \left| \left(\frac{\bar{u}_i^e}{|\bar{u}_i^e|}\right)\left(\frac{\partial N_\alpha^e}{\partial x_i}\right)\right|\right]^{-1}, \tag{7}$$

where $n_{en}$ is the number of nodes of the element, $N_\alpha^e$ is the interpolation function associated with the node $\alpha$.

## 4. GPBi-CG algorithm

The discretization of the Navier-Stokes equation by the stabilized finite element formulations leads to a large and sparse non-symmetric system of linear equations. This linear equation system is solved by using the GPBi-CG algorithm [5].

**Algorithm 1** The unpreconditioned GPBi-CG

$\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_o$;
Set $\boldsymbol{r}_0^* = \boldsymbol{r}_0, \boldsymbol{t}_{-1} = \boldsymbol{w}_{-1} = 0, \beta_{-1} = 0$;
**for** $n = 0, 1, ...$ until $||\boldsymbol{r}_n|| \leq \epsilon||\boldsymbol{b}||$ **do**
$\boldsymbol{p}_n = \boldsymbol{r}_n + \beta_{n-1}(\boldsymbol{p}_{n-1} - \boldsymbol{u}_{n-1})$,
$\alpha_n = \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}{(\boldsymbol{r}_0^*, \boldsymbol{A}\boldsymbol{p}_n)}$,
$\boldsymbol{y}_n = \boldsymbol{t}_{n-1} - \boldsymbol{r}_n - \alpha_n \boldsymbol{w}_{n-1} + \alpha_n \boldsymbol{A}\boldsymbol{p}_n$,
$\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n \boldsymbol{A}\boldsymbol{p}_n$,
$\zeta_n = \frac{(\boldsymbol{y}_n, \boldsymbol{y}_n)(\boldsymbol{A}\boldsymbol{t}_n, \boldsymbol{t}_n) - (\boldsymbol{y}_n, \boldsymbol{t}_n)(\boldsymbol{A}\boldsymbol{t}_n, \boldsymbol{y}_n)}{(\boldsymbol{A}\boldsymbol{t}_n, \boldsymbol{A}\boldsymbol{t}_n)(\boldsymbol{y}_n, \boldsymbol{y}_n) - (\boldsymbol{y}_n, \boldsymbol{A}\boldsymbol{t}_n)(\boldsymbol{A}\boldsymbol{t}_n, \boldsymbol{y}_n)}$,
$\eta_n = \frac{(\boldsymbol{A}\boldsymbol{t}_n, \boldsymbol{A}\boldsymbol{t}_n)(\boldsymbol{y}_n, \boldsymbol{t}_n) - (\boldsymbol{y}_n, \boldsymbol{A}\boldsymbol{t}_n)(\boldsymbol{A}\boldsymbol{t}_n, \boldsymbol{t}_n)}{(\boldsymbol{A}\boldsymbol{t}_n, \boldsymbol{A}\boldsymbol{t}_n)(\boldsymbol{y}_n, \boldsymbol{y}_n) - (\boldsymbol{y}_n, \boldsymbol{A}\boldsymbol{t}_n)(\boldsymbol{A}\boldsymbol{t}_n, \boldsymbol{y}_n)}$,
(if $n = 0$, then $\zeta_n = \frac{(\boldsymbol{A}\boldsymbol{t}_n, \boldsymbol{t}_n)}{(\boldsymbol{A}\boldsymbol{t}_n, \boldsymbol{A}\boldsymbol{t}_n)}$, $\eta_n = 0$),
$\boldsymbol{u}_n = \zeta_n \boldsymbol{A}\boldsymbol{p}_n + \eta_n(\boldsymbol{t}_{n-1} - \boldsymbol{r}_n + \beta_{n-1}\boldsymbol{u}_{n-1})$,
$\boldsymbol{z}_n = \zeta_n \boldsymbol{r}_n + \eta_n \boldsymbol{z}_{n-1} - \alpha_n \boldsymbol{u}_n$,
$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha \boldsymbol{p}_n + \boldsymbol{z}_n$,
$\boldsymbol{r}_{n+1} = \boldsymbol{t}_n - \eta_n y_n - \zeta_n \boldsymbol{A}\boldsymbol{t}_n$,
$\beta_n = \frac{\alpha_n}{\zeta_n} \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{n+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}$,
$\boldsymbol{w}_n = \boldsymbol{A}\boldsymbol{t}_n + \beta_n \boldsymbol{A}\boldsymbol{p}_n$;
**end for**

## 5. GPU implementation

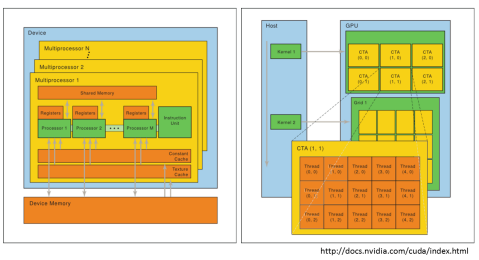


Fig. 1: CUDA multiprocessor architecture

CUDA invented by NVIDIA is a parallel computing platform and programming model. The style of execution is called Single-Instruction, Multiple-Thread (SIMT). In SIMT, multiple threads are processed by a single instruction. CUDA architecture makes computing on a GPU much more simple. To implement the GPBi-CG algorithm on the GPU platform, we developed the library functions for calculating matrix and vector on GPU as follows:

- MV - matrix vector product
- DOT - inner product
- AXPBY - add a multiple of one vector to another
- SCAL - scale a vector by a constant

We used the Thrust parallel algorithms library [10] provided by NVIDIA in the CUDA toolkit to develop the above library functions.

## 6. Performance results

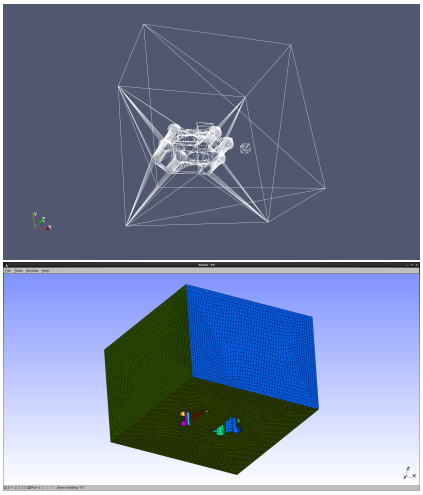

Fig. 2: Fluid domain and its tetrahedral mesh

We consider a problem which consists of an object immersed in a fluid domain as shown in Fig. 2. Starting from a STL mesh as shown in the upper part of Fig. 2, we created tetrahedral meshes with different resolutions by using the open source software NETGEN. All calculations were carried out using an HPC with the following hardware specifications:

- Intel Xeon CPU E5630, 2.53GHz,
- GPU NVIDIA Tesla C2070,
- 24 GB System Memory.

The computational conditions are as follows:

- CUDA 7.0,
- NVIDIA's CUDA Compiler (NVCC), GCC 4.4.7,
- Double precision,
- Compiler options: -O2 -arch=sm_20,
- Stopping criteria of the GPBi-CG algorithm: $||r_n||/||b|| < 10^{-12}$,
- Diagonal pre-conditioner.

We measured the execution times during first 50 time steps of the CPU serial execution and the GPU parallel execution. Tab. 1 shows the results. The GPU execution is 13.7 times faster the CPU execution for the large mesh with 3238815 elements. The speed-up ratio increases proportionally to the number of elements of meshes.

Tab. 1: Speed-up ratios and execution times

| Mesh Size | | GPU | CPU | Speed |
|---|---|---|---|---|
| #Node | #Element | Time | Time | Up |
| 59572 | 307509 | 902 s | 4575 s | 5.1 |
| 115810 | 638447 | 2292 s | 13143 s | 5.7 |
| 489346 | 3238815 | 3.4 h | 46.6h | 13.7 |

The pressure distributions on a cross section at the time step $1500^{th}$ of the CPU execution and GPU execution are shown in Fig. 3. It appears that these distributions are similar in every details.

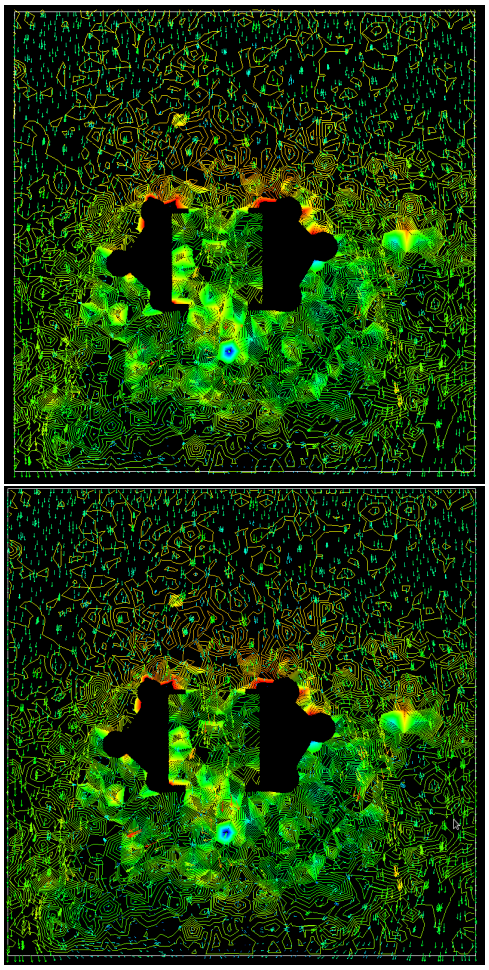

Fig. 3: Pressure distributions on a cross section at the time step $1500^{th}$ of the CPU execution (above) and the GPU execution (below)

## 7. Conclusions

We propose an efficient implementation of a GPU parallel solver based on the GPBi-CG algorithm for 3D Navier-Stokes equations discretized by the SUPG/PSPG stabilized finite element formulations. In further research, we plan to improve the current implementation by employing multigrid pre-conditioners. We also plan to extend this work to parallelization across multiple GPUs.

## REFERENCES

(1) Shakib F., Hughes T.R.J., Johan Z., "A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier-Stokes equations", Computer Methods in Applied Mechanics and Engineering, 89, pp. 141–219 (1991).

(2) Franca L.P., Frey S.L., Hughes T.J.R., "Stabilized finite element methods: I. Application to the advective-diffusive model", Computer Methods in Applied Mechanics and Engineering, 95, pp. 235-276, (1992).

(3) Tezduyar T.E., Mittal S., Ray S.E., and Shih R., "Incompressible flow computations with stabilized bilinear and linear equal order interpolation velocity pressure elements", Computer Methods in Applied Mechanics and Engineering, 95, pp. 221–242, (1992).

(4) Tezduyar T.E., "Stabilized finite element formulations for incompressible flow computations", Advances in Applied Mechanics, 28, pp. 1–41 (1991).

(5) Zhang S.L., "GPBi-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems", SIAM J. Sci. Comput., 18, pp. 537-551, (1997).

(6) Fujino S., Sekimoto T., "Performance evaluation of GPBiCGSafe method without reverse-ordered recurrence for realistic problems", Lecture Notes in Engineering and Computer Science, pp. 1673–1677 (2012).

(7) Nihon Keisan Kogaku-kai Nagare no Yugen Yosoho Kenkyu Iin-kai, "Simulation of flows by finite element method - Continued", Maruzen Publishing, (2012), (in Japanese).

(8) Kawata H., "Numerical simulation of cerebrospinal fluid flow from MRI images using finite element method", unpublished Master Thesis, Academic Year 2011, Graduate School of Environmental Studies, Okayama University, (in Japanese).

(9) Dziekonski A., Sypek P., Lamecki A.; Mrozowski M., "Finite element matrix generation on a GPU", Progress In Electromagnetics Research, vol. 128, 249-265, 2012.

(10) http://docs.nvidia.com/cuda/thrust/

## ACKNOWLEDGEMENTS

This work is supported by JST-CREST (Japan Science and Technology Agency, Core Research for Evolutional Science and Technology). The first author would like to acknowledge Mr. Ryota Yamane (Okayama University) for technical support.