Scientific Research Publishing

Engineering, 2017, 9, 824-847
http://www.scirp.org/journal/eng
ISSN Online: 1947-394X
ISSN Print: 1947-3931

Effect of Parallel Computing Environment on the Solution Consistency of CFD Simulations—Focused on IC Engines

Seunghwan Keum, Ronald O. Grover Jr., Jian Gao, Xiaofeng Yang, Tang-Wei Kuo

General Motors Research and Development, Pontiac, MI, USA
Email: seunghwan.keum@gm.com

How to cite this paper: Keum, S., Grover Jr., R.O., Gao, J., Yang, X.F. and Kuo, T.-W. (2017) Effect of Parallel Computing Environment on the Solution Consistency of CFD Simulations—Focused on IC Engines. *Engineering*, **9**, 824-847.
https://doi.org/10.4236/eng.2017.910049

Received: August 22, 2017
Accepted: October 10, 2017
Published: October 13, 2017

Abstract

For CFD results to be useful in IC engine analysis, simulation results should be accurate and consistent. However, with wide spread use of parallel computing nowadays, it has been reported that a model would not give the same results against the same input when the parallel computing environment is changed. The effect of parallel environment on simulation results needs to be carefully investigated and understood. In this paper, the solution inconsistency of parallel CFD simulations is investigated. First, the concept of solution inconsistency on parallel computing is reviewed, followed by a systematic CFD simulations specific to IC engine applications. The solution inconsistency against the number of CPU cores was examined using a commercial CFD code CONVERGE. A test matrix was specifically designed to examine the core number effect on engine flow, spray and combustion submodels performance. It was found that the flow field simulation during the gas exchange process is the most sensitive to the number of cores among all submodels examined. An engineering solution was developed where local upwind scheme was used to control the variability, which showed good performance. The implication of the observed inconsistency was also discussed.

Keywords

Computational Fluid Dynamics, Parallel Computing, Numerical Error

1. Introduction

Numerical simulation of physical process has been an indispensable asset in product development. In particular, the automotive industries increasingly rely on Computer Aided Engineering (CAE) for product development. For any kind of numerical simulations to be useful in analysis, the solution should exhibit

certain properties. The first one is the accuracy. A simulation is expected to reproduce or predict the physical process with acceptable accuracy to be used in analysis or product development. The solution accuracy depends on two factors. One is the model accuracy, which can be attributed to the assumptions and procedures used in the model development. In terms of CFD, Direct Numerical Simulation (DNS) is considered the most accurate, because it adapts the least assumptions in the modeling of fluid dynamics. On the other hand, models based on Reynolds averaging, commonly referred as RANS (Reynolds Averaged Navier-Stokes), adapt time-averaged governing equation with additional assumptions such as homogeneous isotropic turbulence, which makes it inapplicable in accurate modeling of instantaneous flow field, but more suited for an ensemble averaged flow field.

The other required property is the solution consistency. In other words, a simulation code should be able to give the same answer whenever and on which hardware the code is ran. In a serial code, the solution consistency is very good and has not been an issue for most cases. However, with recent wide spread use of parallel computing, it has been noticed and reported that a parallel code may not produce the same results. The solution consistency in parallel computing is not a trivial task and has been a research topic in computer science [1]-[8]. One of the main reasons is the non-deterministic order of arithmetic operation. In a serial computing, the order of arithmetic operating in a loop is pre-determined at compilation time, and it is not varied from runtime to runtime. On the other hand, in the parallel computing, the loop is split and distributed to a number of different cores, then reduced (summed) to a single value. As such, the order of arithmetic operating in a loop is determined at runtime and may vary from runtime to runtime. This effect, as well as other potential sources of inconsistency, will be reviewed in subsequent sections.

Understanding the sources of such non-deterministic behavior, and the effects thereof, is very important in analyzing the simulation data. This problem is not specific to CFD, but virtually all of parallel scientific computing is affected. Research has been conducted to identify possible sources of variations and how to eliminate them to retain solution determinacy. It has been argued that any parallel scientific computing should be deterministic [7], and determinacy can be obtained without significant overhead [8]. The solution consistency issue in parallel CFD simulation has been reported and investigated as well [9] [10] [11].

In this study, the solution consistency from parallel CFD simulations was examined to understand how the simulation results are affected by the change in parallel computing environment. The study is focused on modeling the internal combustion engine, which is a highly complex reacting flow. It is a good test case because it includes virtually all aspects of recent CFD modeling, such as moving boundary (valves and pistons), two-phase flow with phase change (fuel injection/spray), and chemical reaction (ignition and combustion). Particularly, the fuel injection model is an interesting topic because it relies on the random number sequence, which is known to be sensitive to the parallel computing en-

vironment.

The study is composed as follows. First, the recent research and current understanding of the solution inconsistency in parallel computing is reviewed in Chapter 2. Then, reported solution inconsistency in CFD is reviewed in Chapter 3 with focus on the numerical error and its population in CFD. The solution inconsistency in the CFD of IC engines is examined in subsequent chapters. The numerical setup is summarized in Chapter 4, whose results are described in Chapter 5. The results are analyzed in Chapter 6 and a summary in Chapter 7 will wrap up the study.

2. Source of Solution Inconsistency in Parallel Scientific Computing: Literature Review

In a computer system, numbers are stored with a finite accuracy. Consequently, there are round-off errors in any kind of scientific computing, regardless of being serial or parallel. We will first review what kind of numerical errors are present, and how such errors may impact parallel computing and affect solution determinacy.

2.1. Round-off Error and Non-Associative Floating Number Arithmetic

A computer uses a floating point number system with finite precision [2]. As a result, it cannot precisely represent all real numbers. One example is the treatment of certain numbers, such as 1/3. In a computer, it will always have very small additional numbers near the accuracy limit. It may shock the readers, but most of modern computers cannot represent 0.1 exactly. With a 24-bit computer, 0.1 is represented as 0.100000001490116119384765625. The round-off error is unavoidable in any scientific computing with floating number representation due to the finite accuracy [2] [3] [4].

One of the most well-known effect of the round-off error is the floating point arithmetic, which is not associative [2]. In floating point arithmetic, (a + b) + c and a + (b + c) may not be exactly the same. A simple example is addition of two large numbers and a small number, which are separated at the extreme of the representation range of the floating point representation. For example, on a 32 digit precision machine, (1032 − 1032) + 0.01, will be calculated as 0.01. The first calculation in the brackets is calculated correctly and cancels out. However, if the same calculation is done in reverse order (0.01 − 1032) + 1032, computer will give 0 as the answer. The first calculation gives −1032, because the dynamic range is not available to represent such small value against the large number at machine limit. Consequently the final value becomes zero, and the arithmetic is non-associative. The non-associativity is an active research topic in mathematics and computer science [3].

Numerical issues in serial computing is rather well defined and understood to come from modeling continuum physics with discretized equation with finite accuracy [2], and a well-established guideline is available to estimate numerical

errors [4]. In a serial computing, such error does not affect the solution determinacy as long as the computing environments are kept identical. The computing environments includes the code, the compiler and compiler options, and the computing hardware.

2.2. Effect of the Non-Associative Floating Number Arithmetic in Parallel Computing

In parallel computing, the computing environment has one additional variable compared to the serial counterpart. It is the parallel computing environment, which includes any parallel setup for a parallel code to run, including the computer algorithms for parallel implementation, parallel execution schedule, operating conditions, compiler, and hardware setup. It has been reported that computing results may vary in a parallel computing when the parallel computing environment is changed. The loss of the determinacy in the parallel scientific computing is an actively researched topic in the computer science [3] [5] [6] [7] [8], and researchers have identified a few sources to which the problems are attributed to. For example, the same code and input may generate different results by simply changing the number of processors, among other things [9]. We will review a few well-known numerical errors in parallel computing in subsequent chapters.

In a serial computing, the order of calculation in a loop is already determined at compilation. However, in a parallel computing, a loop is split and distributed over a number of cores for better performance. As such, the order of calculation is determined at runtime, based on the number of cores used in the calculation. For example, a summation operation of an array with N element in a serial code will occur serially from element 1 to N. In a parallel computation with 2 cores will split the loop in two, which will first calculate summation from (1 to N/2) and (N/2 + 1 to N), then the two sums will be added. As such, any variation in the number of processor will vary the order of addition at runtime. The order of arithmetic can be varied even with the same number of cores, dependent on the overhead of each node. Say, let's assume that a simulation was repeated with three cores each time. In one run, core 0 finished first, followed by core 1 and core 2. The order of summation will be (sum from core 1) + (sum from core 2) + (sum from core 3). In another run, core 0 had overhead from other process, and ended last. In this case the total value would be (sum from core 2) + (sum from core 3) + (sum from core 1). In either case, the order of arithmetic is varied at runtime and the final sum will be affected by the non-associativity. As described before, the variation from non-associativity occurs mostly at machine precision limit, which is 15 - 16 digits for double precision in most of numerical simulations.

2.3. Random Number Sequence

Another possible source of additional numerical error in parallel computing is the random number generation. The random number is used extensively in CFD

simulation, particularly for modeling spray. A random number is generated by a pseudo-random number generator, which provides a sequence of random numbers based on its initial seed value. If the initial seed value is the same, resulting random number sequence will be identical. In most of scientific computing, the seed number is held constant for the sake of determinacy, except in some statistical analysis. In a serial environment, as the order of computation is predetermined during compilation, elements in a random number sequence were used sequentially in any single simulation in a deterministic fashion.

However, in a parallel environment, it is not clear how a random number sequence will be used over multiple processes. One option is a replicated approach, where each processor keeps the same random number sequence and use its own. Another options is a distributed approach, where one processor keeps the random number sequence and sends out each element to each core whenever a random number is needed. In either approach, it is clear that the usage of the random number element will vary when the number of processors is changed.

2.4. Other Possible Sources

Other possible sources for the non-determinacy have been investigated, e.g., the effect of cut-cell method on numerical oscillation [12]. Effect of such spurious oscillation may impact determinacy and need to be examined carefully. In return, any type of dispersion treatment, such as artificial viscosity, may impact the simulation results and needs to be understood.

3. Numerical Errors and CFD: Effect of Viscosity

As the numerical errors in scientific computing has been described, now we focus on the numerical errors and their behavior specific to the CFD modeling. The effect of numerical error in serial CFD has been reviewed by Freitas [13], while Poinsot *et al.* reviewed the effect of numerical error in parallel CFD [10]. The numerical error can be represented by a wave, which is propagated by the governing equation, which is the Navier-Stokes equation in CFD. In an inviscid flow without any viscosity, a wave will propagate while maintaining its original shape. In a real flow with a certain amount of viscosity, the wave will be dampened out by the viscosity. The accuracy of any CFD code depends strongly on the viscosity treatment in the simulation.

The viscosity in any CFD simulation can be categorized into three viscosities [10]. One is the physical viscosity, which is the property of the fluid. The other is a model viscosity or turbulence viscosity. In a RANS simulation, typically the mesh is much coarser to capture the details of flow field. As such, the flow field resolved at the grid level does not represent the physical flow, and additional model is required to capture the turbulence occurring at the grid level. As turbulence enhances mixing, the effect of turbulence is often modeled by enhanced mixing via a model viscosity in addition to the physical viscosity. The additional model viscosity is the turbulence viscosity, often referred as ν_t. In addition, there exist a numerical viscosity. The numerical viscosity comes from numerical

setup and methods, rather than physics or models. For example, if the grid is very coarse, one may not be able to capture small structures but only large structures. The solution from a coarse grid will give a blurred image, which is similar to the effect of viscous diffusion. On the other hand, the governing equation has to be discretized to be solved in a computer. And the discretization scheme can add numerical viscosity to the problem. Ideally, one may reduce the numerical viscosity by adapting a high order accuracy numerical scheme and a very fine mesh.

With the three viscosities, the Navier-Stokes equation in CFD can be represented as the following with an effective viscosity, ν_{eff}.

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = \frac{\nabla P}{\rho} + \nu_{eff} \nabla^2 u \tag{1}$$

$$\nu_{eff} = \nu + \nu_N + \nu_t \tag{2}$$

In Equation (1), u is the velocity and P is the pressure. In Equation (2), ν, ν_N, and ν_t represents molecular viscosity, numerical viscosity and turbulent viscosity, respectively. If the viscosity is very low to the physical viscosity, the simulation will provide the most accurate results. However, the numeric will be sensitive to small oscillations and the code can be unstable. Consequently, the code will require much finer time step and grid size, which will make overall simulation very expensive. This is what is called the direct numerical simulation (DNS). On the other hand, if the effective viscosity is much higher than the modeled flow, CFD simulation results will be dampened out and fail to reproduce the physics. Typical engineering type simulations are based on RANS formulation, where large amount of turbulence viscosity is used for a good compromise in numerical stability and accuracy. For higher accuracy and time-resolved simulation, a large eddy simulation (LES) is used, where the turbulence is modeled only at sub-grid level, which typically has much smaller turbulent viscosity than RANS. Poinsot [10] reported the turbulent and numerical viscosities in typical LES and RANS simulations, which is reproduced below.

At the DNS level, the code uses very fine mesh to resolve the smallest flow structure where viscous dissipation occurs. As it resolves the viscous dissipation directly, it requires no model about the turbulent viscosity, and its value is zero. It requires very fine numeric to resolve the small scale structure. Along with the fine mesh, the numerical viscosity is very small close to zero. In LES, the turbulence in small structure is modeled by the turbulent viscosity. The grid size is a bit larger than DNS, which invokes small amount of numerical viscosity as well.

However, as summarized in **Table 1**, Poinsot pointed out that the effective viscosities in DNS and LES are still very low, such that it can propagate small numerical disturbances. As such, if there exist any additional numerical errors from parallel computing, DNS and LES may suffer from the loss of determinacy. On the other hand, the RANS simulation takes advantage of tremendous amount of turbulent viscosity as well as numerical viscosity. Even if the same numeric as DNS or LES is used, the model itself puts turbulent viscosity which is

Table 1. Orders of magnitude of numerical and turbulent viscosity in DNS, LES and RANS code [10].

Model	$\frac{\nu_N}{\nu}$: Numerical viscosity	$\frac{\nu_t}{\nu}$: Turbulent viscosity
DNS	~0	0
LES	1 - 10	5 - 50
RANS	10 - 500	100 - 1000

larger by an order of magnitude. Consequently, Poinsot concluded that RANS simulation may not be able to propagate small numerical error [9] [10] [11], and the solution from RANS simulation should be consistent regardless of changes in parallel computing environment.

4. Effect of Parallel Computing Environment in RANS Simulations

4.1. Case Setup

Numerical modeling of internal combustion engine requires modeling of reacting flow with often two-phase fluid including phase change over very complex geometry with moving boundaries. A number of different sub-models have been developed and implemented by different researchers, which makes evaluation of the numerical error not a trivial task.

To simplify the analysis, numerical tests are designed to start from one model to the combination of different models. Total of seven numerical test cases were designed, which are summarized in **Table 2**.

The test cases are first grouped in closed and open cycle simulations. A diagram is shown in **Figure 1** to show the difference.

In the closed cycle simulations, the simulation period does not involve the gas exchange such as intake and exhaust. This setup would let us investigate each individual sub models such as spray, spark and combustion models separate from complex gas exchange process. The first case in closed cycle (Case C1) is spray modeling, which is actually in a spray vessel, rather than engine. The spray vessel has a fixed volume and no moving boundary, and it is expected to serve as an ideal condition to examine the numerical errors from non-associativity and, in particular, the parallel random number effect on spray modeling. Starting from the spray vessel case, different cases were setup with additional model complexity. Combustion was added to the spray in the diesel combustion case (Case C2). And spark ignition is added to the spray and combustion models in gasoline combustion case (Case C4). To separately examine the spark ignition model from spray, a homogeneous spark ignition case was included (Case C3).

In the open cycle case, we started with examining only the flow field in both diesel (O1) and gasoline engine (O2) geometry. In Gasoline case (O2), spray is added for modeling. Then, a full cycle simulation (except exhaust process) of gasoline direct injection spark ignition case was examined, which is Case O3.

Table 2. Test cases.

	Case	Hardware	Intake	Spray	Spark	Combustion
Closed Cycle	C1	Spray Bomb	N	Y	N	N
	C2	Diesel Engine	N	Y	N	Y
	C3	Gasoline Engine A	N	N	Y	Y
	C4	Gasoline Engine B	N	Y	Y	Y
Open Cycle	O1	Diesel Engine	Y	N	N	N
	O2	Gasoline Engine A	Y	N	N	N
	O3	Gasoline Engine B	Y	Y	Y	Y

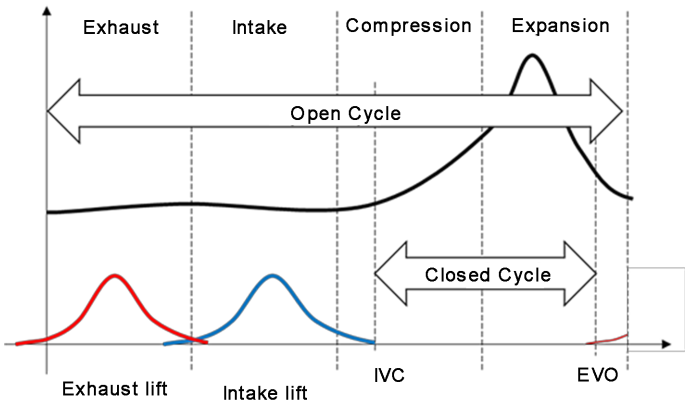


Figure 1. Schematic diagram on open and closed cycle.

Case O3 has the most complete set of numerical models, which includes intake flow modeling with valve and piston motion, spray injection, spark ignition and combustion. This will be a good case to examine the effect of numerical error from parallel CFD code on overall IC engine modeling.

For all cases, parallel computing environment was varied by changing the number of cores. Otherwise, the computing environments are kept the same for each test.

4.2. Numerical Setup

CONVERGE of Convergent Science Inc. [14] was used for evaluating the effect of parallel computing environment on solution determinacy. For the purpose, all cases in **Table 2** was simulated with different number of cores for each case. Standard k-ε turbulence model is used for all simulations. For combustion cases, CONVERGE's built-in chemical kinetics solver was used with skeletal n-heptane (for diesel simulation) and i-octane (for gasoline simulation) reaction mechanisms, which were developed by Ra [15].

CONVERGE generates computational mesh on-the-fly. The computational mesh is Cartesian with boundary cells fitted to the geometry by cut-cell method [16]. Local grid can be refined by predefined embedding at certain location over

certain amount of time. Also, the grid can be refined on-the-fly by Automatic Mesh Refinement (AMR) based on local gradient of selected variables. In the current study, all simulations used base mesh size of 1 mm with near-wall cells refinement up to 0.25 mm using fixed embedding. The AMR was not activated in all simulations.

Numerical diffusion from discretization is a key factor which can affect the effect of any numerical errors. CONVERGE provides two numerical schemes. One is central differencing scheme (CDS), which is 2nd order with minimum numerical diffusion. The other is upwind differencing scheme (UDS), which adds significant amount of numerical diffusion to solution. As described in any numerical analysis textbooks [17] [18], CDS adds numerical dispersion to the solution, which may make solution unstable. In such cases, even though CDS is prescribed, CFD codes locally uses first order discretization to stabilize solutions. CONVERGE also utilizes UDS to locally stabilize flow, when numerical error occurs with CDS [14]. Criteria for local upwind scheme is proprietary and not available to public. For the current simulations, CDS was used for momentum transport in all simulations, while passive scalars were discretized with upwind differencing scheme.

The operating conditions for the test matrix is summarized in **Table 3**.

5. Simulation Results

5.1. Closed Cycle Simulation

Case C1: Spray bomb

As discussed in introduction, two sources of numerical errors are considered in this study. One is the numerical error from non-associativity and the other is parallel random number generation. As the spray model is heavily dependent on random number to model particle physics, care has been taken to examine the effect of processor numbers and subsequent numerical error in the spray vessel case. The spray characteristics from the simulations are plotted in **Figure 2**.

First, the spray penetration is compared in **Figure 2(a)**. The spray penetration is determined at 98% liquid mass fraction from the injector tip. Three sets of CPUs were used in the simulations, namely 12, 24 and 48 CPUs. **Figure 2(a)**

Table 3. Operating conditions for the test matrix.

	Case	Hardware	MAP	RPM
Closed Cycle	C1	Spray Bomb	N/A	N/A
	C2	Diesel Engine	280 kPa	3750
	C3	Gasoline Engine A	95 kPa	1300
	C4	Gasoline Engine B	95 kPa	1300
Open Cycle	O1	Diesel Engine	N/A	N/A
	O2	Gasoline Engine A	95 kPa	1300
	O3	Gasoline Engine B	95 kPa	1300

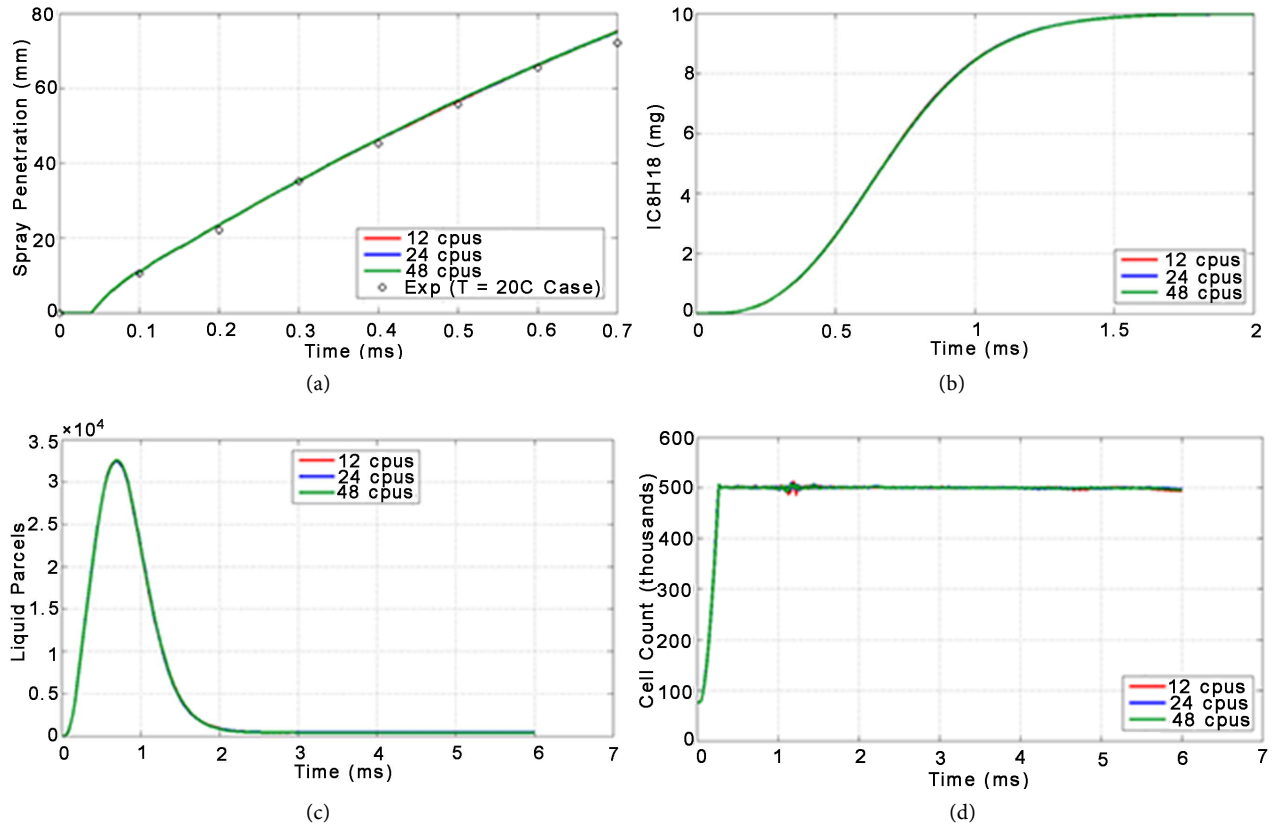


Figure 2. Effect of different processor numbers on spray characteristics. (a) Spray penetration; (b) fuel concentration; (c) liquid parcels; (d) cell count.

shows that the penetration length is hardly affected by the number of processors. Similar trends were observed in other key characteristics, such as vaporized fuel mass (**Figure 2(b)**) and number of liquid parcels (**Figure 2(c)**). For this case, the adaptive grid refinement was enabled to capture spray phenomena accurately. And the number of cells is also compared in **Figure 2(d)**, which shows almost identical number of adaptive grid resolutions.

Often times, overall characteristics such as penetration may not be a good measure for comparing spray simulation results. Even though overall characteristics are similar, spatial distribution and spray structure may be different. The spray structure is compared in **Figure 3** for different processor numbers.

On the top two rows are the comparison of liquid phase parcels with three different processor numbers at different simulation times. It is clearly shown that the liquid spray structure is not affected by the number of CPUs. The gas phase fuel species concentration is compared in the bottom row. The overall structure is very similar with different process numbers. It is noted here that the contour level was set up very narrow to examine difference in fuel concentration. The contour level in fuel mass fraction is also shown in the same figure. So even though there are some difference in the fuel concentration contour, absolute variation with the number of CPU is relatively small.

Case C2: Spray + Combustion (Diesel Combustion)

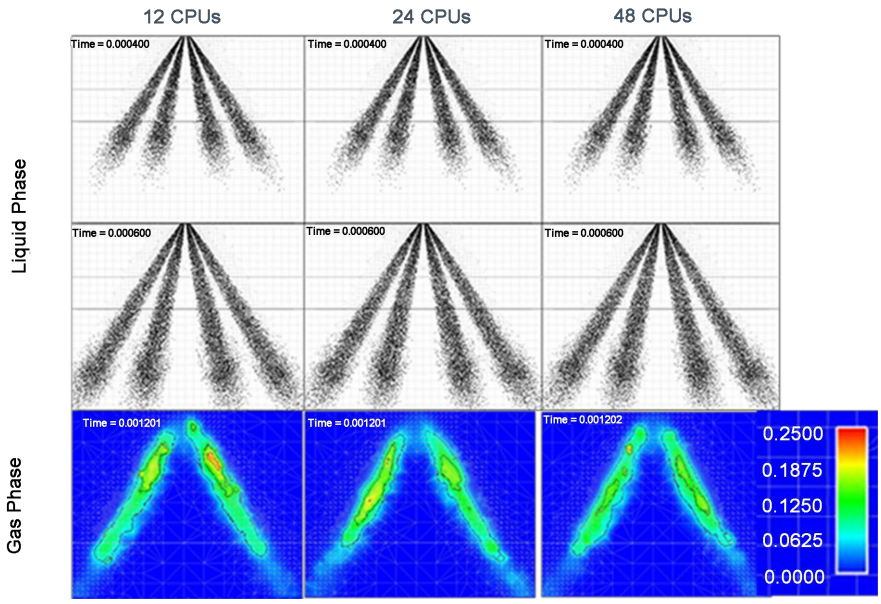


Figure 3. Effect of different processor numbers on spray simulation.

The effect of numerical error from different number of CPUs is examined for a combination of spray and combustion model in diesel combustion (Case C2). The results are compared in **Figure 4**, which shows pressure history during combustion. It is clearly shown that the combustion model does not show dependency to the numerical errors.

Case C3: Ignition and Combustion (Gasoline Combustion)

The gasoline combustion case is shown in **Figure 5**, which compares pressure traces from three different CPUs (4, 12 and 36). Notice that this case does not involve spray, but models spark ignition and subsequent combustion. A simple spark ignition model was used to model combustion initiation. To mimic energy transfer from high temperature arc between spark gap, a small amount of energy (20 mJ) was added over spark duration in a spherical region. The diameter of spherical region was 1 mm. The spherical region was initiated between spark gap, then it moves with the background flow. Subsequent reaction from external energy is modeled by chemical kinetics. No flame propagation model was used in the current study. **Figure 5** shows that neither ignition nor combustion model is affected by the processor number.

Case C4: Spray, Ignition and Combustion (Gasoline DISI Combustion)

This is the most complex case in the closed cycle simulation. It involves spray, ignition and combustion model in a direct-injection spark-ignition engine. Injection occurs 50 CA bTDC, followed by spark at 37 CA bTDC. The same ignition model from Case C3 is used with 60mJ of ignition energy. The results are compared in **Figure 6**.

The results, again shows virtually no dependency to the number of cores and related numerical error. As a quick summary, closed cycle RANS simulations showed very little sensitivity to the core numbers and changes in the parallel computing environment. The spray vessel case is particularly interesting. With

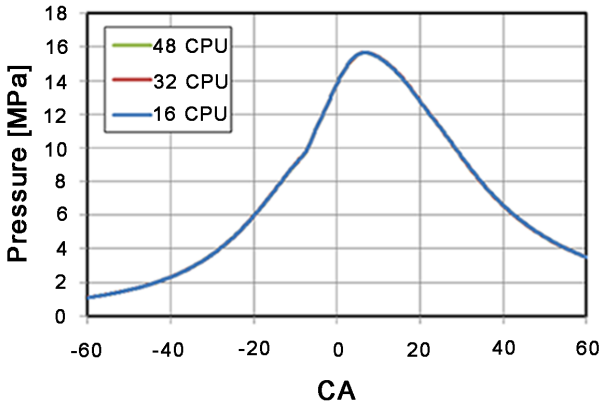


Figure 4. Effect of different processor numbers on closed cycle DI combustion.

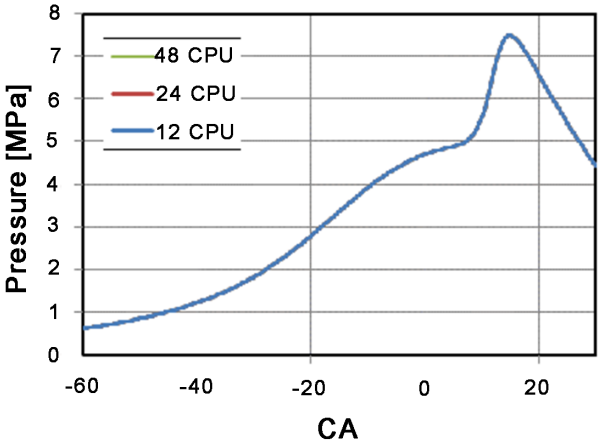


Figure 5. Effect of different processor numbers on closed cycle SI combustion.

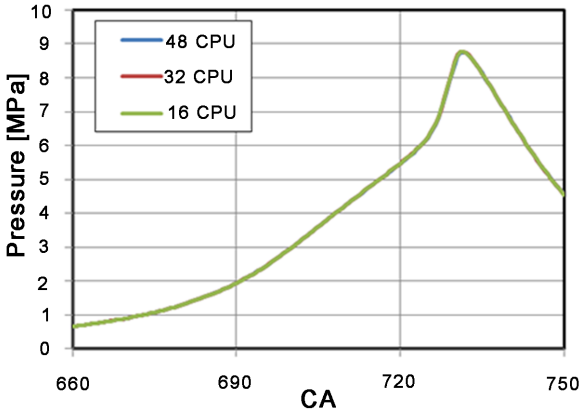


Figure 6. Effect of different processor numbers on closed cycle DISI combustion.

different number of CPUs, the random number sequence should have been different with the number of CPUs. Subsequently the spray model was expected to be affected, showing variation to processor numbers. However, it was found that

the spray modeling results were virtually unaffected. It can be interpreted that the randomness of the random number sequence is maintained in the parallel computing. Ignition and combustion models did not show any dependency to the numerical error.

5.2. Open Cycle Simulation

Case O1: Gas Exchange in Diesel Port Flow

A pure flow simulation was carried out over a diesel port flow geometry. There is no spray event, and chemical kinetics model is completely turned off. The results are shown in Figure 7. Swirl number of cylinder is compared in Figure 7(a), and tumble number is compared in Figure 7(b). The simulation ran over the intake process to compression and part of exhaust process. The intake valve duration is marked in Figure 7. For the comparison, 16, 32 and 48 CPUs were used in simulation.

As shown in Figure 7, both the swirl and tumble number showed variation with different number of CPUs. A couple of interesting observation was made from the pure port flow simulation. First, the variations become noticeably significant only after the maximum valve lift, which is the half of the intake duration. Second, the variation persists for a while, but differences remain almost the same during the intake process. It does not show any exponential growth of small initial variation, which is often observed in highly sensitive simulations such as LES. Finally, the variation is reduced during the compression stroke, but it is not completely removed.

As the variation in simulation results start to appear near the maximum valve lift, CFD simulation results are visualized to examine the flow field near the maximum valve lift. The velocity fields from 16 and 32 CPU cases are shown in Figure 8.

Surprisingly, it shows two very distinct flow patterns even with high viscosity from RANS simulations. When 16 CPU is used (left plot in Figure 8), the jet flow from the valve extends diagonally through the cylinder over quite long

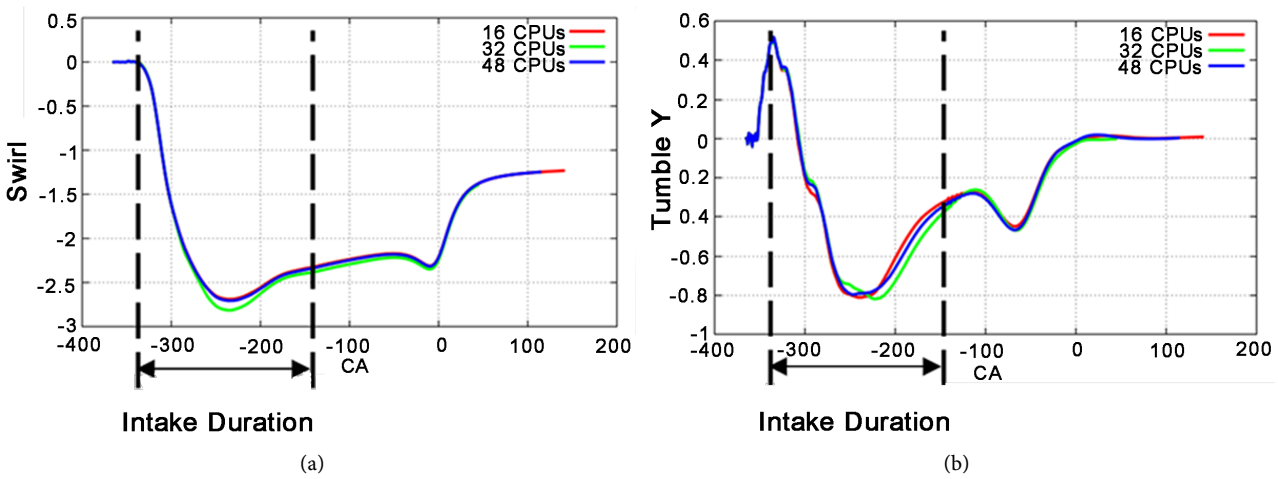


Figure 7. Effect of processor number on diesel port flow. (a) Comparison of swirl; (b) comparison of tumble.

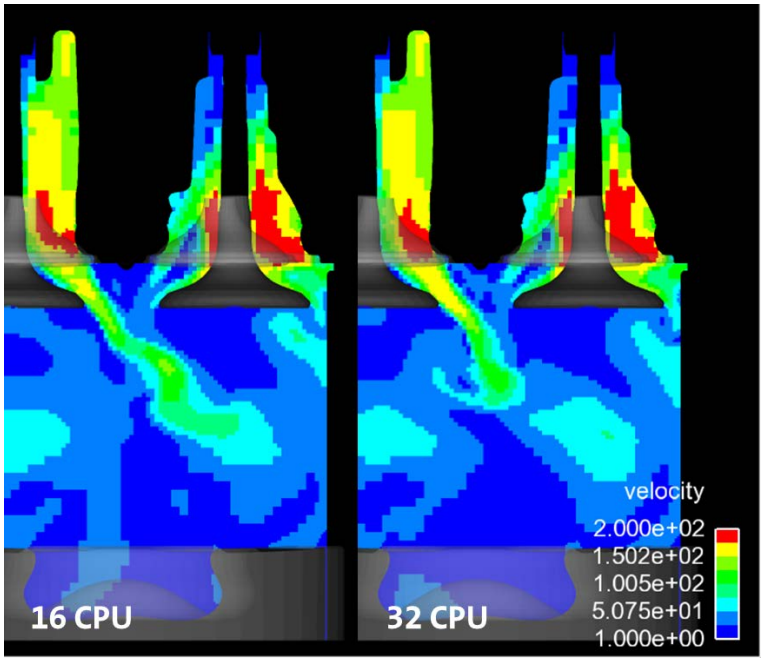


Figure 8. Comparison of flow field with different number of CPUs in diesel port flow simulation.

distance. On the other hand, when 32 CPU is used with exactly the same set of input files, the jet from the valve was found to penetrate much shorter distance. Considering that more complex cases (spray, ignition and combustion) did not show any dependency to the number of CPUs, it is quite surprising that the flow field shows variation with the processor numbers, especially with RANS modeling.

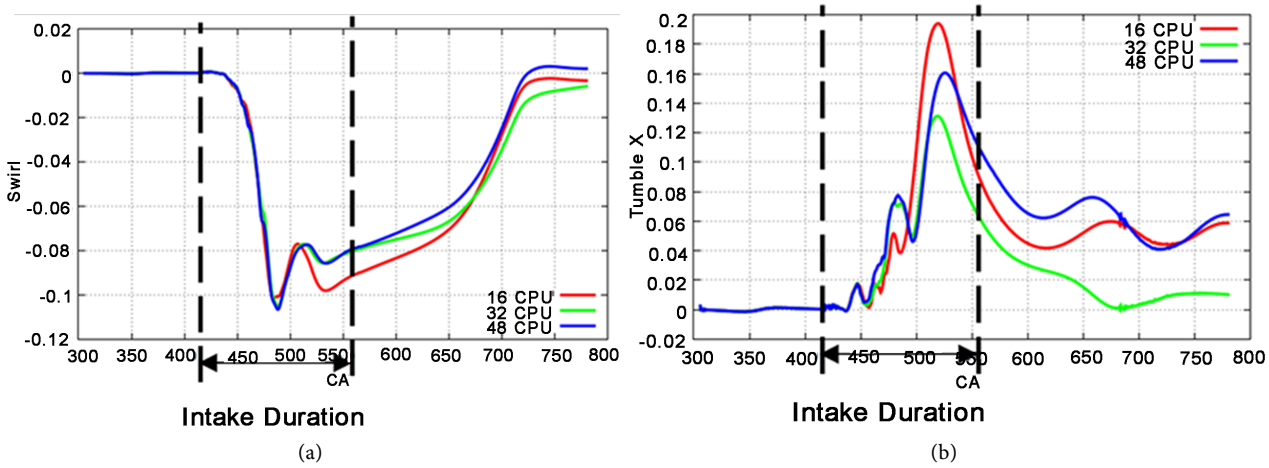
It was questioned if the number of CPU dependency is due to a particular geometry. A gasoline engine geometry was used to examine the number of CPU dependency (not listed in **Table 1**), of which the results are shown in **Figure 9**.

The gasoline port flow simulation shows similar results. Both swirl (**Figure 10(a)**) and tumble (**Figure 10(b)**) shows variation in prediction with different number of CPUs, which start to become noticeable around maximum valve lift. It is concluded that the CPU number dependency is not unique to specific geometry.

Case O2: Gas Exchange in Gasoline Port Flow with Spray

As mentioned previously, the gasoline engine takes relatively homogenous mixture for combustion. For the purpose, the injection occurs during the intake process, either into the port or cylinder. The port flow simulation without spray showed some variation. And it is investigated how the spray will behave in the CPU-number dependent flow field. The results are shown in **Figure 10**, which compares the mixing by equivalence ratio. It is reminded that both simulations used exactly the same input and mesh.

The results in **Figure 10** is quite startling. Even though every inputs and computing environments are the same except the number of processors, the results are quite different. Actually, it is interesting to note that one may



(a) (b)

Figure 9. Effect of processor number on port flow simulation: Gasoline Engine. (a) Comparison of swirl; (b) Comparison of tumble.

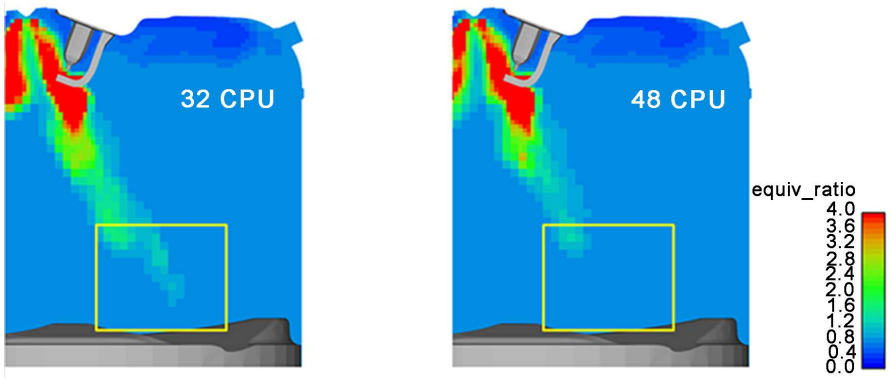


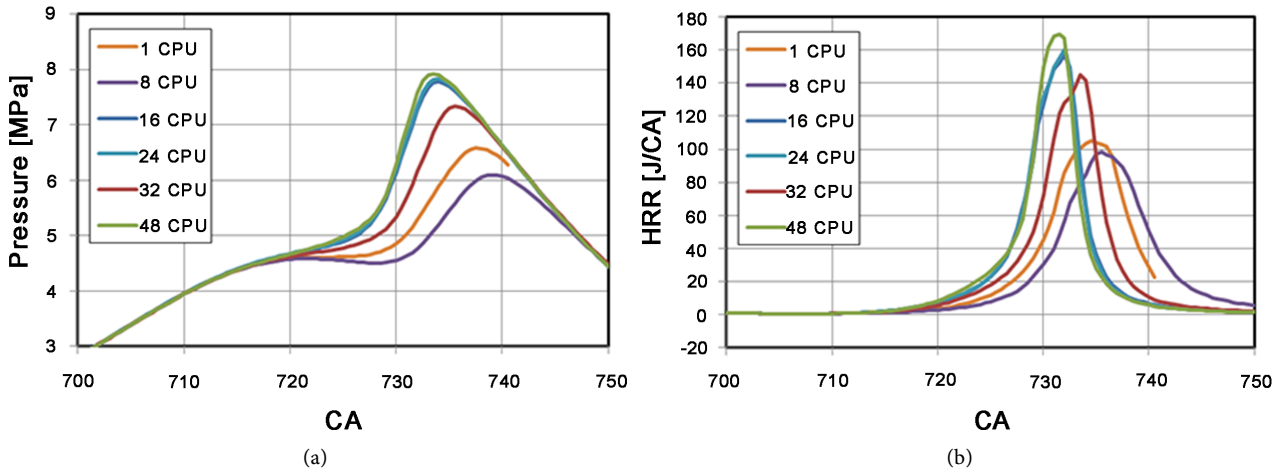
Figure 10. Effect of processor number on mixing process: injection during intake.

generate 48 CPU results by changing input and running with 32 CPU, and vice versa. It is clear that the variation in flow field affects the spray simulation, which alone is not sensitive to the change in parallel computing environment.

Case O3: Gasoline DI SI combustion with Intake Process

This is the case with largest model complexity, including flow through moving valves, piston motion, spray, spark ignition and combustion. As discussed in the closed cycle results, most of sub-models have shown virtually no dependency to the number of cores when there is no open boundaries. On the other hand, the flow field showed sensitivity to the core numbers particularly in valve flow simulation. The results are shown in **Figure 11**. The pressure comparison is shown in **Figure 11(a)**, with corresponding heat release rate in **Figure 11(b)**. The simulation includes intake process, which is not shown in **Figure 11**. The injection event occurs during the intake process.

From previous port flow simulation and port flow + spray simulation, it was already observed that mixing field is already affected by the changes in parallel computing environment. As such, it was expected that combustion event is similarly affected. **Figure 11** shows clear dependency on the processor number.



(a) (b)

Figure 11. Comparison of Test O3 with different number of processors. (a) Pressure; (b) Heat release rate.

As the number of processor is changed, combustion phasing and peak pressure, heat release profile are all varied.

5.3. Reducing Variability—An Engineering Approach

An engineering approach was tried to reduce the variability. In all simulations, the central difference scheme (CDS) was used. However, when the solution field shows strong gradient, upwind difference scheme (UDS) was used locally. This is a common engineering approach in most CFD simulations, and details can be found in the CONVERGE documentation. As the CDS with local UDS showed non-determinacy, a different approach was developed and implemented into CONVERGE. In the old approach, the local UDS was applied based on variable. For example, if a computational cell shows strong gradient in x-direction velocity u, u was discretized by UDS, while all other variable are modeled by CDS. In the new approach, the local UDS was applied based on the computational cell. If any of the three velocity components show strong variability, the particular cell is discretized UDS in all directions. The flow and combustion simulations results are shown in **Figure 12** and **Figure 13**.

The results clearly show improvement in the determinacy, showing much smaller variation in the results against different number of cores. The new method is available in CONVERGE version 2.3.

6. Discussion

It was found that the RANS simulation in closed cycle simulation (*i.e.*, no open boundary) is not affected by the number of cores and additional numerical errors with the current code CONVERGE. One of the most interesting results is the spray simulation with quiescent ambient background, case C1. The case is interesting because it is a good test bed to examine the effect of random number sequence in the parallel computation. And it was found that the effect of random number is small enough to be neglected. However, it is noted here that the

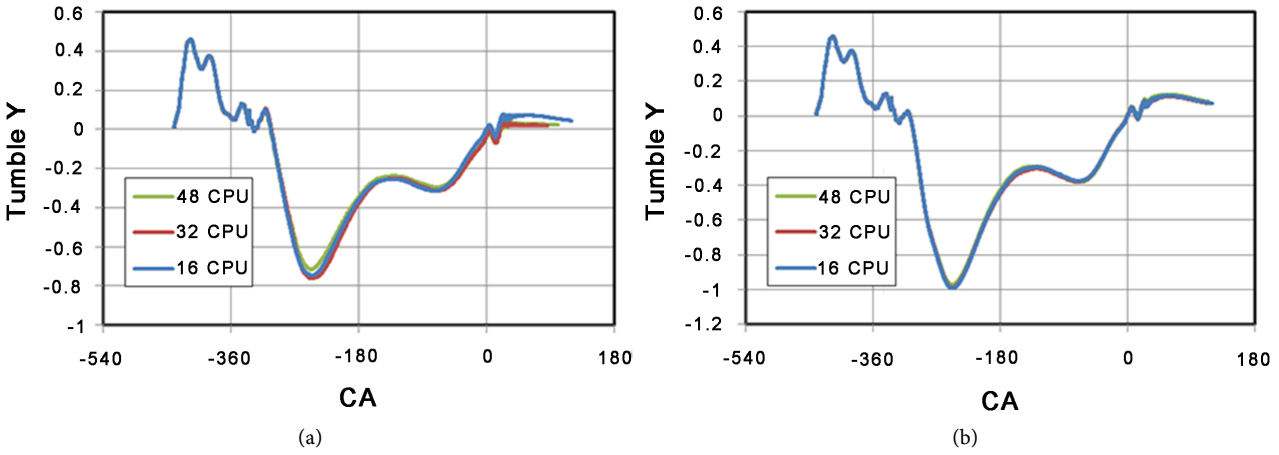


Figure 12. Comparison of cylinder tumble Y between CDS with local UDS, and CDS with local UDS in all directions in premixed SI combustion. (a) Results using CDS with local UDS; (b) results CDS with local UDS in all directions.

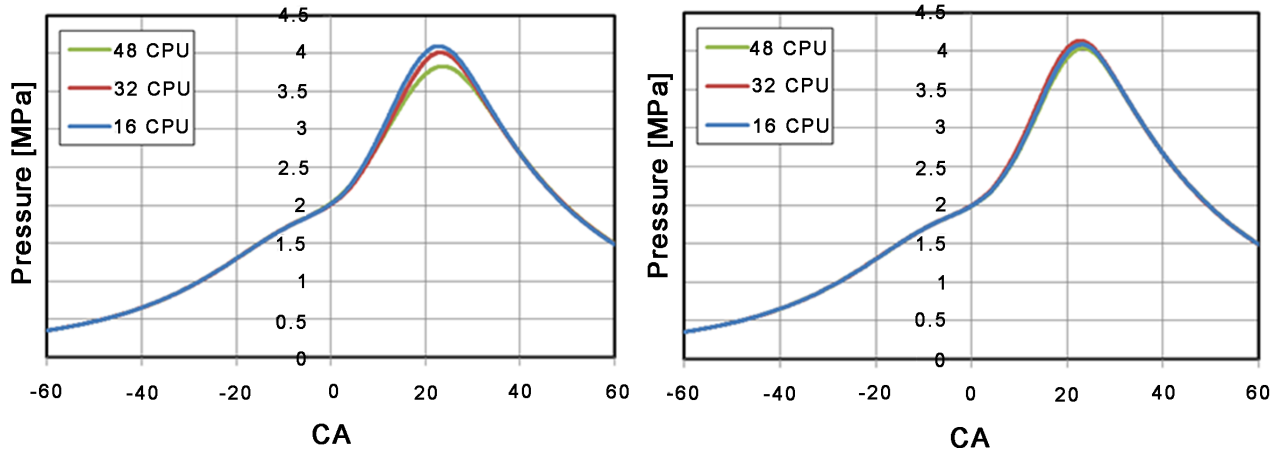


Figure 13. Comparison of combustion pressure between CDS with local UDS, and CDS with local UDS in all directions in premixed SI combustion. (a) Results using CDS with local UDS; (b) results using CDS with local UDS in all directions.

finding should not be considered as a general conclusion. The spray simulation is affected by a number of different factors, such as the random number generator and the sample size (parcel count). In the current simulation, rather a large sample size of 105 to 106 was used. So, it would be fair to say that the current spray simulation was not affected by the parallel computing environment because a large number of parcels (lager sample size) is used. Albeit there are such caveats, the results that parallel random number sequence doesn't affect the results when the sample size is large enough, is very encouraging.

In addition to spray model, ignition and combustion models were added in case C2 (diesel compression ignition) and C3 (gasoline spark ignition). In both cases, the results are the same with different number of CPUs. From these results, it can be interpreted that the RANS simulation is not affected by the additional numerical errors from the change in parallel computing environment.

One important finding is that the additional numerical errors from parallel

computing environment variation do not affect results even with the central differencing scheme. As discussed above, the numerical errors are typically very low, which is thought to be easily smeared out by any diffusion, either from the numerical viscosity or the turbulent viscosities. As reviewed in the introduction (refer to Table 1 for details), the RANS simulation adds significant amount of both turbulent and numerical viscosities. All of closed cycle simulations were carried out with central difference discretization, which adds very small amount of numerical viscosity. The closed part simulation results clearly show that the numerical errors are removed by the turbulent viscosity alone in parallel RANS simulations.

In these simulations, only the momentum equation was discretized with central difference. All other passive variables, including turbulence kinetic energy, dissipation rate and temperature, were discretized in upwind differencing, which may add artificial viscosity to the solution. To clarify, the same simulation was carried out with central difference for all variables. The results are shown in Figure 14.

It is clearly shown that the numerical sensitivity does not show up with central difference for all variables, and the previous conclusion remains the same. In parallel RANS simulation, the additional viscosity from turbulence model is enough to reduce the additional numerical error from parallel computing environment variation and ensure determinacy.

Unlike closed cycle simulations, all of open cycle simulations showed sensitivity to the number of cores and variability in results. It is interesting to note that such strong variation is only observed during open cycle simulation, and does not diffused out by the additional eddy viscosity. Moreover, the variability only appears during when there is flow through the valve. The CFD visualization in Figure 8 clearly shows that the flow through the valve (gas jet) is affected the most by the numerical errors. Notice that this is the simplest case for the open cycle simulations in Table 2. So the solution variability exists in the most basic

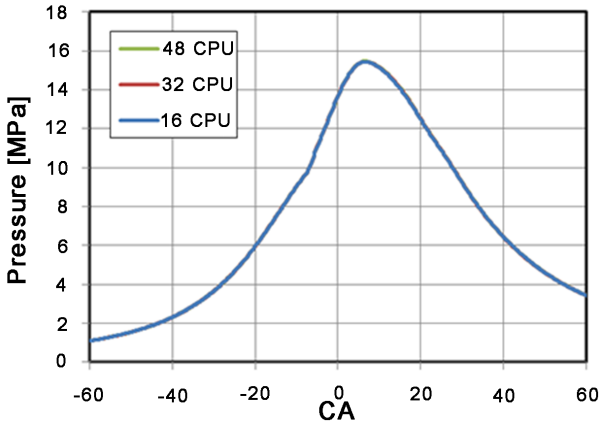


Figure 14. Effect of different processor numbers on closed cycle DI simulation with central difference scheme for all variables.

flow simulation. The impact of the flow variability on other models are significant, as it has been shown in case O2 (flow and spray) and O3 (flow, spray, spark ignition and combustion.).

One may question the spray model and the random number sequence for the simulation variability. To address this issue, test O2 was repeated with spray replaced by a gas jet. Instead of spray parcels, fuel vapor was introduced as external source into equivalent multi-hole spray region. So the effect of random number sequence is not present in the new test. The results are shown in Figure 15. It is clearly shown that the mixing predictions vary with different number of CPUs. This confirms that the flow field is the most sensitive to the variation in parallel computing environment.

The solution variability observed in the open cycle simulations starts from the intake process to grow over the rest of the cycle, which is similar to what was found in LES simulations [10]. However, it should be reminded that the simulations are ran with RANS models, which add significant amount of model viscosity. The model viscosity efficiently washes out small structure in all simulation results, even with 2nd order CDS. As any small structure is dissipated by the viscosity, there is no physical mechanism to amplify and let small perturbation grow to larger values. The observation may be interpreted that there exist numerical errors which are large enough to survive the strong dissipation from the model viscosity. The effect of model and numerical viscosity in RANS turbulence model variability requires further understanding.

A thorough error quantification study should reveal the source of such strong non-determinacy during the intake process, which is beyond the scope of the current study. The authors propose possible hypothesis to the source of the errors, which is non-associative floating arithmetic and numerical oscillation from the cut-cell method. Let us first take a look at a typical velocity field during the intake process, as shown in Figure 16.

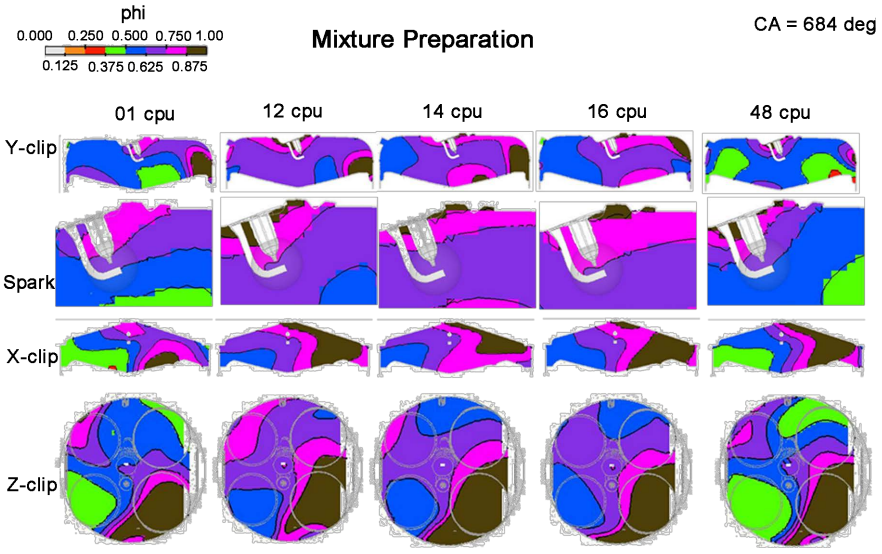


Figure 15. Comparison of gas jet tests with different number of CPUs.

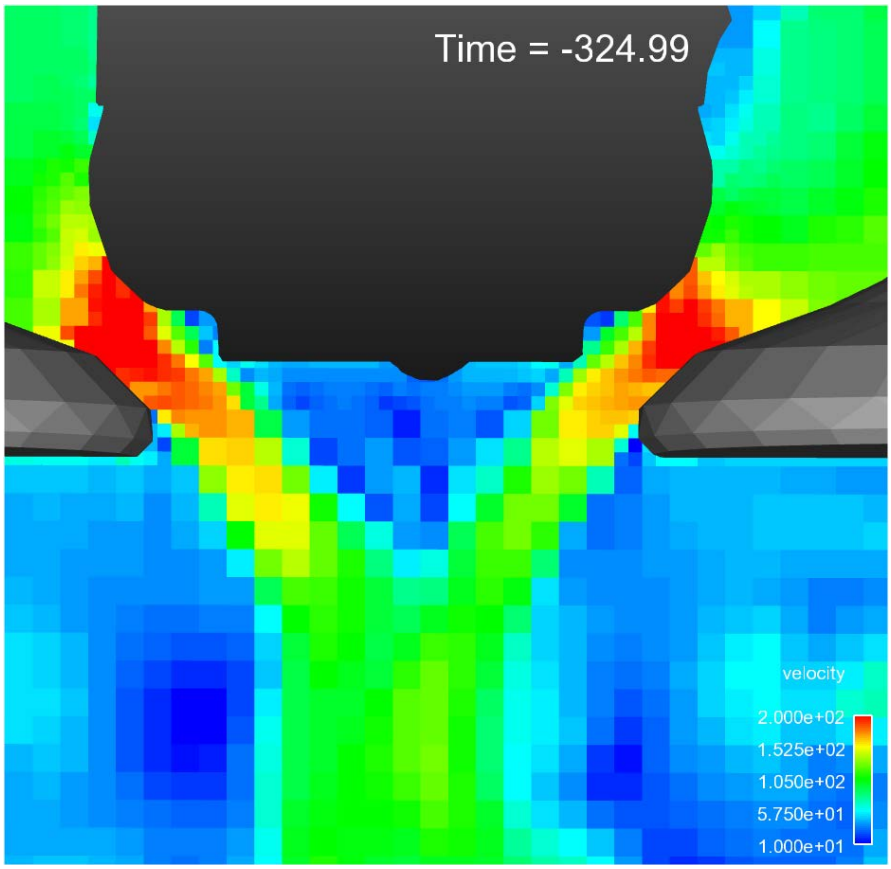


Figure 16. Velocity contour through intake valves during the intake process.

One of the characteristics of the intake process is the large velocity gradient over the flow through the valve. It can be expected that the local residual at such large gradient can be quite large, while other region, especially the in-cylinder mixture apart from the intake jet, might have low local residual. The global residual, which is the sum of the local residual, will be a classic example of numerical errors from the non-associative floating number arithmetic. It is clear that different arithmetic will be used when a different domain decomposition is used. However, as aforementioned, the numerical error from the non-associative arithmetic is very small at the machine limit and might be diffused out easily by the model viscosity in RANS simulations. On the other hand, we discussed large unphysical oscillation in the cut-cell method [12]. The valve area is literally covered by the cut-cells as shown in **Figure 17**, and there might be large amount of numerical errors. When the large numerical cut-cell error is combined to the non-associative floating number arithmetic, it may propagate even in the presence of strong diffusion from the model viscosity to result in variability in the solutions.

Non-determinacy was observed from the RANS simulation using a commercial CFD code. The question is how to understand such variability. Attempts have been made to correlate the simulation variability to experimental variability and explain the simulation variability as something physical, such as cyclic

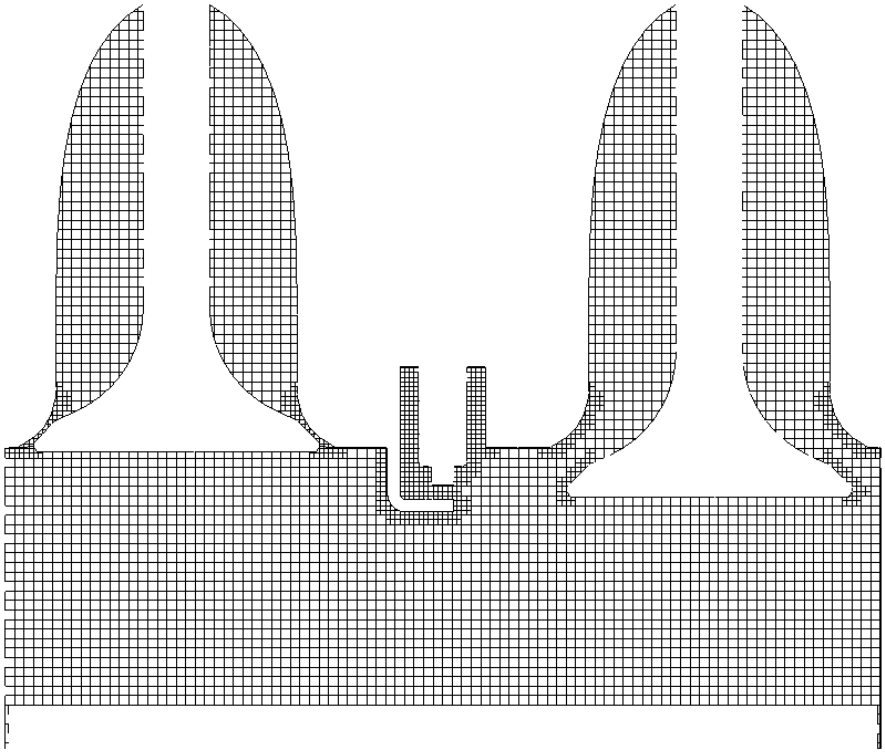


Figure 17. Typical Mesh Topology Around Valves - Valve Surface Covered by Cut-cells.

variability of an IC engine observed from experiments. However, this approach requires further understanding of the numerical methods in the CFD code.

Any engine experiment data will exhibit some level of cyclic variability at the most stable operating condition. Such variability has been one of the major topic in engine research, and often attributed to a number of sources such as variability in flow, spark ignition, and/or direct injection. To simulate such variability, the model should be able to capture such variability. As such, LES has been considered as a proper tool for modeling the cyclic variability. On the other hand, in general, RANS has been considered an improper tool to model the variability. It relies on very strong model viscosity, which washes out any small structure [9] [11]. If a small structure is observed in a RANS simulation, the results should be carefully examined if there is any significant amount of unphysical numerical error that is strong enough to overcome the large model viscosity.

Apart from the model viscosity, one might be tempted to correlate the variability observed in parallel RANS simulation to that of variability from experiment. However, a thorough error analysis is pre-requisite for such approach. It is reminded that getting similar trend with the experiment does not always guarantee the validity of the methodology. Addition of two errors may cancel each other to make results look good, but it does not make the method right. The variability in experiment comes from various sources, such as variation in experimental boundaries, initial conditions, and ambient conditions. On the other hand, particularly from the parallel RANS simulations in this study, the most significant source of error is the valve flow, and it is purely numerical. The

variability from experiment has hardly any correlation to the variability appears in numerical simulations. Moreover, it is important to notice that variability in parallel computation will exhibit itself in simulation of (rather ideal) perfectly deterministic experiment. Correlating variability between numerical simulation and physical experiment can lead to unphysical answers and should not be practiced by any CFD researchers. One may consider the numerical error from parallel computation as an uncertainty quantification analysis. However, uncertainty quantification requires detailed and careful design of the uncertainty factor [16] [19]. The current numerical errors are not quantified, and it is not fully understood yet.

The analysis of variability in parallel RANS simulations requires further research to be fully understood. Until the numerical errors are fully quantified and analyzed, any attempt to correlate the non-determinacy to any physically observed variation should be avoided.

7. Conclusions

Numerically, the RANS formulation relies on the strong numerical diffusivity, which washes out any small numerical noise. Considering the high level of numerical diffusion, the non-determinacy observed in the current study may not be explained solely by the numerical errors in the parallel computing environment, and further study and analysis will be required for full understanding. A thorough study on the source and effect of the determinacy should be followed. The authors proposed a hypothesis that the large unphysical error from cut-cell method may propagate throughout the computing domain via the non-associative floating number arithmetic.

Analysis of the determinacy should be carried out carefully not to be misguided into unphysical conclusion. One may be tempted to correlate the solution variability from the simulations to the uncertainties in the measurement. However, there exists no physical nor statistical correlation between the physically observed variability to the numerical solution variability. In addition, RANS model does not have a mechanism to transfer the small errors and subsequent variability. As such, the variability in the solution observed in the current RANS simulations can be interpreted that the numerical error generated solely from the parallel computing environment is quite large. Such error is purely numerical, and should have no physical meaning. Source of such strong numerical error should be identified and resolved for accurate modeling. As a conclusion, the variability in RANS should be carefully investigated if it is physical or purely numerical prior to any analysis using the RANS results.

8. Summary

The solution determinacy, which is the critical issue in parallel CFD simulations, has been investigated in this study. First, general numerical error in any scientific computing, namely the round-off error, was reviewed. In a serial computing,

the round-off error is deterministic. In a parallel computing, the round-off error may grow differently when the parallel computing environment is changed (*i.e.*, different number of processors). Such behavior is due to the non-associativity of the floating point arithmetic. Other numerical errors related to parallel computing were also reviewed, such as the random number sequence. From the literature survey, it was learned that any parallel scientific computation will experience new issues with parallelism, particularly with respect to the determinacy.

Then, the effect of the numerical error and the effect of viscosity in CFD was reviewed. Interestingly, there have been no reports about the effect of parallelization on RANS simulations to the authors' knowledge. The only related work is the LES simulations with the RANS consideration, where it has been shown and discussed that a LES simulation may be sensitive to parallel parameters, and may generate different instantaneous solution fields [9] [10] [11]. The question to follow is, how about RANS? The solution determinacy of RANS simulation in parallel environment was examined using a commercial CFD code, CONVERGE.

The parallel computing environment was varied by changing the number of cores. It was found that RANS simulation results using CONVERGE showed variations in the results, thus the RANS simulation is not deterministic. As the IC engine simulation includes a number of sophisticated sub-models, the authors tried small problems to identify which sub-model is the most sensitive to the change in the parallel computing environment. It was found that the spray and combustion models showed deterministic behavior, while the flow simulation is very sensitive to the changes in the parallel environment. The variability is purely numerical and does not have any physics behind it, and it should not be correlated against experimentally observed cyclic variability.

Acknowledgements

The authors thank Convergent Science Inc. for technical discussion and implementing the new discretization method.

References

[1] Dennis, J.B., Gao, G.R. and Sarika, V. (2012) Determinacy and Repeatability of Parallel Program Schemata. *Proceedings of the* 2012 *Data-Flow Execution Models for Extreme Scale Computing*, Minneapolis, 19-23 September 2012, 1-9. https://doi.org/10.1109/DFM.2012.10

[2] IEEE Standard (2008) IEEE Standard for Floating-Point Arithmetic. IEEE Standard 754-2008, 29 August 2008.

[3] Demmel, J. and Nguyen, H.D. (2013) Fast Reproducible Floating-Point Summation. 21s*t IEEE Symposium on Computer Arithmetic*, Austin, 7-10 April 2013, 163-172. https://doi.org/10.1109/ARITH.2013.9

[4] Anonymity (2008) Procedure for Estimation and Reporting of Uncertainty Due to Discretization in CFD Applications. *Journal of Fluid Engineering*, **130**.

[5] Blackford, L.S., Cleary, A., Whaley, R.C., Demmel, J., Dhillon, I., Ren, H., Stanley, K., Dongarra, J. and Hammarling, S. (1997) Practical Experience in the Numerical

Dangers of Heterogeneous Computing. *ACM Transactions on Mathematical Software*, **32**, 133-147. https://doi.org/10.1145/264029.264030

[6] Villa, O., Daniel, C.-M., Gurumoorthi, V., Marques, A. and Krishnamoorthy, S. (2009) Effects of Floating-Point Non-Associativity on Numerical Computations on Massively Multithreaded Systems. *Cray User Group Meeting*.

[7] Bocchino Jr., R., Adve, V.S., Adve, S.V. and Snir, M. (2009) Parallel Programming Must Be Deterministic By Default. *Proceeding of HotPar'09 Proceedings of the First USENIX Conference on Hot Topics in Parallelism*, Berkeley, 29-31 March 2009.

[8] Blelloch, G.E., Fineman, J.T., Gibbons, P.B. and Shun, J. (2012) Internally Deterministic Parallel Algorithms Can Be Fast. *Proceedings of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, New Orleans, 25-29 February 2012, 181-192.

[9] Senoner, J.-M., Garcia, M., Mendez, S., Staffelbach, G., Vermorel, O. and Poinsot, T. (2008) Growth of Rounding Errors and Repetitivity of Large-Eddy Simulations. *AIAA Journal*, **46**, 1773-1781. https://doi.org/10.2514/1.34862

[10] Poinsot, T., Garcia, M., Senoner, J.M., Gicquel, L., Staffelbach, G. and Vermorel, O. (2010) Numerical and Physical Instabilities in Massively Parallel LES of Reacting Flows. *Journal of Scientific Computing*, **49**, 78-93. https://doi.org/10.1007/s10915-010-9432-8

[11] Staffelbach, G., Senoner, J.M., Gicquel, L. and Poinsot, T. (2008) Large Eddy Simulation of Combustion on Massively Parallel Machines. *High Performance Computing for Computational Science—VECPAR*, Toulouse, 24-27 June 2008, 444-464.

[12] Schneiders, L. (2013) An Accurate Moving Boundary Formulation in Cut-Cell Methods. *Journal of Scientific Computing*, **235**, 786-809.

[13] Freitas, C.J. (2002) The Issue of Numerical Uncertainty. *Applied Mathematical Modeling*, **26**, 237-248. https://doi.org/10.1016/S0307-904X(01)00058-0

[14] Senecal, P.K., Richards, J., Pomraning, E., Yang, T., Dai, M.Z., McDavid, R.M., Patterson, M.A., Hou, S. and Shenthaji, T. (2007) A New Parallel Cut-Cell Cartesian CFD Code for Rapid Grid Generation Applied to In-Cylinder Diesel Engine Simulations. SAE Paper No. 2007-01-0159.

[15] Ra, Y. and Reitz, R.D. (2008) A Reduced Chemical Kinetic Model for IC Engine Combustion Simulations with Primary Reference Fuels. *Combustion and Flame*, **155**, 713-738. https://doi.org/10.1016/j.combustflame.2008.05.002

[16] Oliver, T.A. and Moser, R.D. (2011) Bayesian Uncertainty Quantification Applied to RANS Turbulence Models. *Journal of Physics*, *Conference Series*, **318**, Section 4.

[17] Ferziger, J.H. and Peric, M. (2001) Computational Methods for Fluid Dynamics. Springer, Berlin.

[18] Trefethen, L.H. (1996) Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations. Unpublished Text. http://people.maths.ox.ac.uk/trefethen/pdetext.html

[19] Gorle, C., Emory, M., Larsson, J. and Iaccarino, G. (2012) Epistemic Uncertainty Quantification of RANS Modeling of the Flow over a Wavy Wall. Center for Turbulence Research Annual Brief.