A LAGRANGIAN DESCRIPTION OF THE RECICULATING FLOW PAST A BACKWARD FACING STEP – AN OPENMP IMPLEMENTATION

Joel Varun Vasanth
AE14D408

ABSTRACT

Separating shear flows are globally hydrodynamically unstable and result in periodic vortex shedding. Such flows are characteristic of geometries like backward-facing steps or bluff bodies. In the context of gas turbine combustors possessing such geometries that find application in aircraft engines, these flows are especially relevant. In the presence of a flame, the vortex-shedding-modulated heat release generates unfavorable combustion instabilities detrimental to performance. It is thus important to understand such flow fields as part of a larger feedback loop with combustion and acoustic fields. In this project, a reduced-order model for vortex shedding past a step based on a Lagrangian formulation is developed. Lagrangian methods are distinct from Eulerian approaches in that they are mesh-free and self-adaptive to capture large gradients of vorticity, ω, well. The method involves discretizing ω with a large number of vortex elements, each with a small quantity of ω. Assuming the flow is 2D and incompressible, a streamfunction ω is defined and a Green's function is used to solve the discrete $\psi - \omega$ Poisson equation to obtain the induced velocity field. The motion of each vortex then reduces to the solution of a set of ODEs, overcoming difficulties associated with the convective nonlinearity of the Navier Stokes equations. Their collective motion defines a frequency of shedding and the length of the recirculation zone for the Reynolds number considered, which serve as parameters to test for convergence. The time complexity of the resulting algorithm depends on ω, the number of discrete vortices introduced. Higher ω leads to a better resolution of the vorticity, and a more accurate solution. The portion of the algorithm for calculating the induced velocity field is the most time consuming. For this project, it is proposed to develop an OpenMP version of the serial code to scale up performance of the algorithm.

Note – run the code as
gfortran -fopenmp prm.f90 pot.f90 prnt.f90 flw.f90 main.f90

INTRODUCTION

The role of vortex shedding in driving combustion instability, especially in geometries with separating shear layers, such as backward facing steps, is very important (Ghoniem et al., 2005; Schadow & Gutmark, 1992). In premixed flames, transition to instability is affected by the flame consumption speed controlled by the equivalence ratio, altering the vortex shedding time periods (Altay et al., 2009). In diffusion flames, the heat release fluctuations are governed by fluctuations in the mixture fraction in the regions of the largest concentration gradients, close to the inlet plane of fuel and oxidiser. Thus, shedding close to the inlet plane is an important factor (Kannan et al., 2016; Magina et al., 2019). In either case, the vortex shedding forms a self-sustained oscillator for the flow field, in addition to the acoustics. Spatio-temporal analysis reveals such oscillations are the result of an absolutely unstable mode, that serves as a fluid-dynamic resonator (Ghoniem et al., 2002). The adjustment of frequencies of these two resonators via underling nonlinear interactions is responsible for causing instability, intermittency and more complex phenomena such as flow-acoustic lock-on (Vasanth & Chakravarthy, 2021).

The above highlighted importance of vortices makes it necessary to predict the flame response as a result of vortical perturbations, and not just acoustic perturbations, as performed in transfer function models. The modelling difficulty is in overcoming the uniform base flow and steady base flow assumptions, largely adopted in such models, especially for a geometry such as the backward facing step (BFS), which is inherently 2D.

Therefore, it is important to develop a reduced order model for the two-dimensional flow fields consisting of vortex shedding. Developing solely the non-reacting flow model and parallelizing it is the subject of this work, leaving the reacting flow studies as part of future work. Lagrangian methods are used when vortex shedding is the dominant feature of the flow under study, and is the feature of interest. Lagrangian methods are distinct from Eulerian approaches in that they are mesh-free and self-adaptive to capture large gradients of vorticity, ω, well. The method involves introduction of elements of vortices to an

otherwise potential flow. These are called 'vortex blobs' consisting of an elemental amount of vorticity depending on the inlet velocity. Such blobs are introduced at regular intervals in time, after which they convect due to the influence of the potential flow and the induced velocity by the remaining vortices. Their collective interaction results in growth and development of large scale vortical structures. The convergence of such methods is measured by the frequency of the resulting shedding and the length of the recirculation zone, which will be measured in this work.

The objective of this work is three fold –

- obtain the converged solution, i.e. the physically observed frequency of shedding and recirculation zone length for a backward facing step flow.
- describe the algorithm developed and the scope for parallelization by identifying parallelizable parts of the code.
- show the gain in performance by applying OpenMP to parallelizable parts of the code.

MODEL FORMULATION

In separating shear flows, the unstable modes of oscillation comprise of the shear layer mode and the wake mode. The goal is to model the coherent flow structures characteristic of the wake mode that are most responsible for driving combustion instabilities. In several studies, it was observed that the dominant mode of oscillation close to the step was two-dimensional (Ghoniem et al., 2002; Wee et al., 2004) with spanwise-coherent flow structures. Thus, a 2D simulation is adopted, excluding any 3D effects.

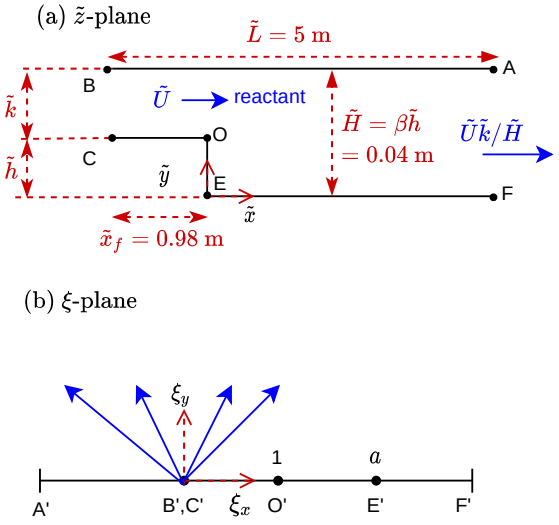


Figure 1. The Schwartz-Christoffel transformation of the step geometry with dimensions adopted from (Altay et al., 2009). The origin of the complex plane in (a) is at the step edge, O.

The geometry is adopted as per (Altay et al., 2009) and shown in Fig. 1(a) with respective dimensions. The step is defined by OE and reactants enter BC with a velocity $\tilde{U}$ (tilde representing dimensional quantities) over a step with expansion ratio β = 2, so $\tilde{k} = \tilde{h}$.

We adopt a modification of the inviscid Lagrangian vortex model by (Pastoor et al., 2003). The vorticity distribution is approximated by a continuous production of vortex elements at each time step $\Delta\tilde{t}$ at the step edge O. The irrotational flow is obtained using a Schwartz-Christoffel transformation (SCT) where the physical domain described on a complex $\tilde{z}$ plane to mapped to a plane ξ (corresponding vertices denoted with primes), where the potential flow is described on $Im(\xi) > 0$ with a point source at B'C' (Fig. 1(b)). The transformation is

$$\tilde{z} = \frac{\tilde{H}}{\pi}\left[\log\left(\frac{1+\eta}{1-\eta}\right) - \frac{1}{a}\log\left(\frac{a+\eta}{a-\eta}\right)\right],\ \eta = \sqrt{\frac{a^2-\xi}{1-\xi}} \tag{1}$$

Here, $a = \tilde{H}/\tilde{h}$. The point source accounts for the acoustic velocity fluctuations at the step edge, using a fluctuating volume flux $\dot{Q}(t) = \tilde{k}\tilde{u}'$, where $\tilde{u}'$ is the acoustic velocity, in addition to the steady source at ξ = 0, as below:

$$\tilde{\sigma}_{\text{source}} = \frac{\tilde{U}\tilde{k}}{2\pi\xi}\frac{d\xi}{d\tilde{z}} \tag{2}$$

The strength of the vortex elements is created at each time step with circulation $\tilde{\Gamma}_{init} = -\left(\tilde{U} + \tilde{u}'\right)^2 \mathrm{d}\tilde{t}/2$ at the location $(\tilde{x}, \tilde{y}) = (\tilde{U}\Delta\tilde{t}/2,\ 0)$, where $\tilde{u}'$ is the unsteady acoustic velocity. The no penetration boundary condition at the combustor walls is satisfied by introducing an image vortex in $Im(\xi) < 0$. The induced velocities of each vortex in the physical $\tilde{z}$ plane are then derived from the complex velocity $\tilde{\sigma} = \tilde{u} - j\tilde{v}$

$$\tilde{\sigma}_{\text{vort}} = \left[-\frac{j\tilde{\Gamma}}{(\xi-\xi_V)} + \frac{j\tilde{\Gamma}}{\left(\xi-\bar{\xi}_V\right)}\right]\frac{d\xi}{d\tilde{z}} \tag{3}$$

Where $j = \sqrt{-1}$, ξ_V is the ξ location of the vortex and overbar denotes complex conjugate. The singularity at the center of each vortex is removed by using a model by Lamb-Oseen velocity distribution by which the azimuthal velocity within a core of radius r_0 is given as

$$\tilde{u}_\theta = \frac{\tilde{\Gamma}(\tilde{r},\tilde{t})}{2\pi\tilde{r}} \tag{4}$$

where $\tilde{\Gamma}(r,t) = \tilde{\Gamma}_0(1-\exp(-\tilde{r}^2/\tilde{r}_0(t)^2))$. A linear temporal growth of $\tilde{r}_0 = \tilde{r}_B + 2\sqrt{\tilde{\nu}\tilde{t}}$ is modelled to account for viscous dissipation of vorticity, $\tilde{\nu}$ being the kinematic viscosity.

The $\tilde{r}_B$ is an initial radius corresponding to the boundary layer thickness. A non-dimensionalisation is performed, where time is scaled as the ratio of the reference velocity and length scales, $\tilde{U}/\tilde{h}$. Circulation is scaled as $\tilde{U}\tilde{h}$. Vortex merging is simulated by adding the circulations of two vortices into a single vortex if the distance between their centers is less than a critical value r_{crit}. The new vortex is located at the Γ-weighted mean of the old vortex centers. The advection of each elementary vortex is due to the velocities in Eqs. (2)-(4). At a certain downstream length $x_{\max}$, vortex elements are deleted, since their effect on the upstream flow becomes negligible at this distance.

ALGORITHM

The code is written in Fortran and consists of a main program – main.f90, that runs the related functions and keeps time. It uses modules for each aspect of computation. The modules are

- prm.f90 – declaration of all major parameters and constants
- flw.f90 – flow field functions, such as advection, deletion of vortices at end of domain, adding vortices at the step edge, reduction of circulation at the walls, etc.
- prnt.f90 – data output and print
- pot.f90 – potential flow equations, conformal mapping (Schwartz-Christoffel transformation) equations.

The module flw.f90 contains the following important subroutines:

- addvor() – to add vortices to the domain at each time step.
- advect() – performs time stepping to advect each vortex.
- getvels() - obtains the velocity due to the potential flow and the shed vortices for each vortex
- calcvels() – obtains the total velocity field on an Eulerian grid.
- calc_bwall_vels() – obtains velocities on the bottom wall EF (Figure 1(a)) for the purpose of calculating the recirculation zone length.

The order of these functions as called in the main program as occurring in each time step is as follows. First, a new vortex is introduced into the domain at each time step with a circulation $\tilde{\Gamma}_{init}$ in addvor(). The newly introduced vortex along with existing vortices are then advected in advect(). Next, the vortex merging algorithm is carried out in merg(). Vortices within a certain critical radius are merged. Next, in wall(), the vorticity of the vortices that are close to the wall are annihilated, and vortices that leave the domain are deleted. Lastly, the vortices along the bottom wall and the velocity field is computed and printed.

The computational complexities of each routine depend on the function of the routine. For wall(), a condition is checked for each vortex and thus, loops over each vortex once. Thus, its complexity is *O(N)* where *N* is the size of the problem, ie, the number of vortices in the domain. For merg(), merging condition is applied comparing locations of each vortex with other vortices. Once this condition is checked for a certain vortex, that vortex is not counted when checking for successive vortices, since the condition is already applied. Thus, a set of nested loops are used with ranges for the outer loop $i = 1$ to N and inner loop $j = 1$ to i. Thus, the complexity is $O(N^2/2)$. The routines getvels() and calcvels() are higher in complexity, $O(N^2)$, where N is the number of vortices in the domain. This is because the calculation of advection velocities for each vortex requires to the contributions from all the remaining vortices. The computation of the wall velocities calc_bwall_vels(), loops over all vortices for each grid point, and this its complexity goes as *O(NM)* where *M* is the number of wall grid points.

PARAMETERS

Parameters are chosen based on a past work by Wee et al., (2004), from which results of the non-reacting flow are available for comparison and validation. The Reynolds number (*Re*) under consideration is 3700. The step height is 20mm with an expansion ratio of 2. The downstream length $x_{\max}$is taken to be greater than twice the experimental recirculation length as 20 (x_R ~ 5.5 for *Re* = 3700). The merging radius is taken to be a linear function varying from 0.01 at $x = 0$ to 0.1 at $x = 20$. The initial core radius $\tilde{r}_0 = 0.05$.

A range of time steps *dt* are chosen 0.001 – 0.1 to examine the effect of the time step on the recirculation zone length.

VALIDATION RESULTS

We now show the obtained values of the recirculation zone (RZ) length as a result of the simulations, by using the time step *dt* as a parameter. The value obtained in the work by Wee et al (2004) is 5.5 times the step height. In Fig 1, it can be seen that the RZ length decreases with a decrease in the time step, and becomes closer to the experimental value as *dt* is decreased. This is because the smaller time step leads to more number of vortex elements in the domain, and thus a finer resolution of the vorticity field.

The larger number of vortex elements in the domain due to the reduced *dt*, as can be seen in Fig 2(b), increases the computational costs and time of simulation, as highlighted in the previous section. As the *dt* increases, the number of vortices decreases, and the circulation per vortex is much higher (due to a larger *dt*) since $\tilde{\Gamma}_{init} = -\left(\tilde{U} + \tilde{u}'\right)^2 \mathrm{d}\tilde{t}/2$. This leads to a very coarse resolution of the vorticity field and the dynamics are not

properly captured. This reflects in the wrong value of the RZ at higher *dt*s. Thus, we take the value of *dt* for the parallelization studies as 0.01.

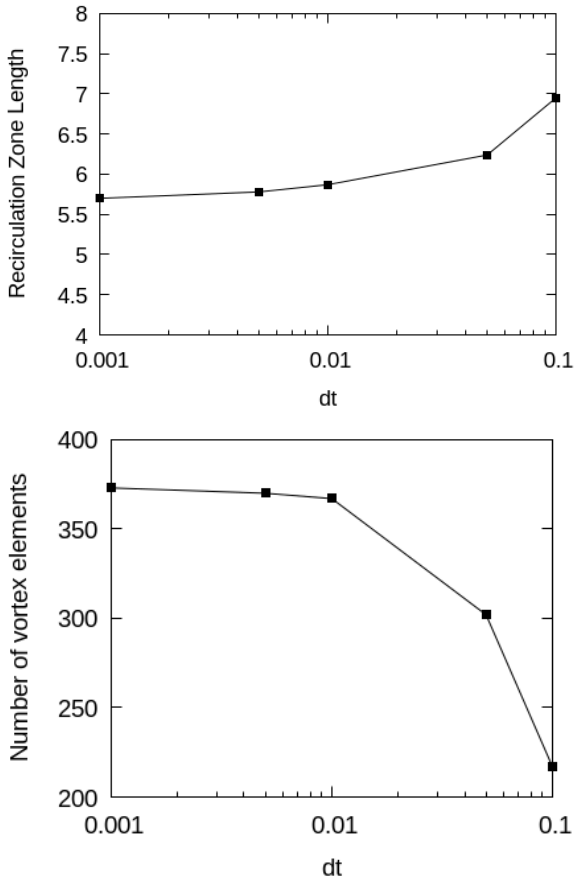


Figure 2. Convergence test results for the flow field. (a) above - Recirculation zone length as a function of dt (b) below - number of vortex blobs/elements in the domain.

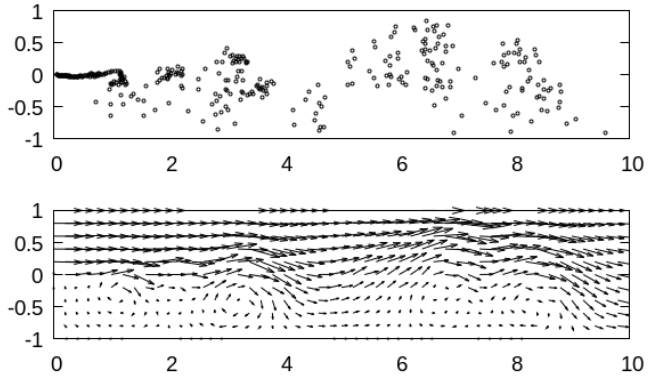
The flow field solution for $dt = 0.01$ is plotted in Fig. 3 at ten time steps apart. The roll up of the vortex sheet formed by introduction of a large number of elementary vortices at the step is evident. A large vortex (evident from the velocity field) formed by the collective interaction between elemental vortex elements (evident from the plot of vortex elements) is present at $x = 1$ (Fig. 3(a)) and 3, that advects downstream to $x = 3$ and 5 (Fig. 3(b)) respectively. In addition, in Fig. 3(b), a new vortex is formed near the step, previously not present.

PARALLELISATION

Profiling

We first examine the time taken by each subroutine in the serial code. The function omp_get_wtime() was used for this purpose. Time taken per iteration is calculated and plotted for each subroutine in Figure 4, along with the number of vortices in the domain. For every subroutine, a common observation is that the time increases from a low value and plateaus to a near constant value. This is because time complexities scale with the number of vortices in the domain, that shows the same trend. The rising number of vortices shows the flow field developing into a converged solution, that occurs when the number of vortices saturates.

(a) t = 120. Vortex elements (top), velocity field (bottom)



(b) t = 130. Vortex elements (top), velocity field (bottom)

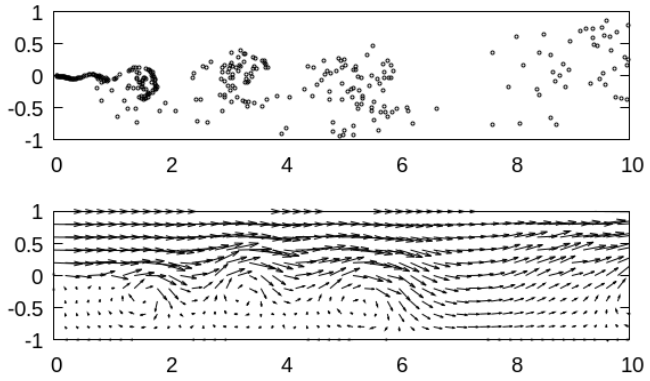


Figure 3. Plot of shed vortex elements in the physical z domain at (a) $t = 120$ and (b) $t = 130$ along with the total velocity fields computed on a 50 x 10 grid from these vortices and the potential flow. The origin corresponds to the step edge O (Figure 1(a)). The flow is from left to right, beginning from the inlet at $x = 0$, $y \in [0, 1]$.

In Figure 4, The routine addvor() requires an just an array allocation and deallocation and thus takes the lowest time. For wall(), the time taken increases by one order of magnitude as the number of vortices increases. For the routines merg() and advect(), the time complexities scale nonlinearly with the number of vortices in the domain, as discussed in the algorithm

section, and thus, an increase in time of over two orders of magnitude is seen.

It is clear that the $O(N^2)$ routines are the most time consuming, i.e., the advect(), getvels() and calc_bwall_vels(), which takes about 1s, 0.1s and 0.01s per iteration respectively. Each of these routines have nested loops. Thus, each of these routines have scope for parallelization.

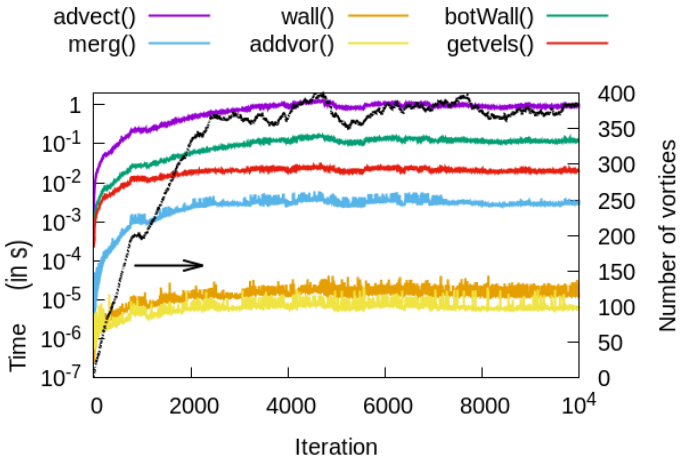


Figure 4. Time taken for each subroutine (colored lines plotted on the left axis) per iteration, shown in relation to the number of vortices (black line plotted on the right axis). The subroutine calc_bwall_vels() is renamed as botWall() in this figure.

The advect() routine is nothing but the RK4 time stepping that calculates the four slopes between two time steps (k_1, k_2, k_3 and k_4), which calls the getvels(), before approximating the new vortex positions. The advect() routine thus calls getvels() 4 times. Thus, it is sufficient to parallelize the getvels() routine. This is done using the *!$omp parallel do* directive along with clauses for determining the private/shared scope of the variables.

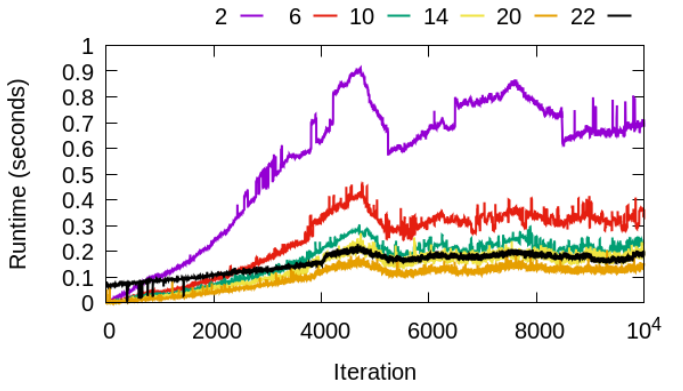


Figure 5. Runtimes per iteration of the subroutine advect() after parallelization, plotted for different number of threads, specified in the legend above the plot.

The *calcvels()* routine is to compute the full velocity field on an Eulerian grid, and so its complexity depends on the grid size. The *calc_bwall_vels()* routine is similar to the *calcvels()* routine, but calculates velocities only on points on the bottom wall, for the purpose of computing the recirculation zone. Each routine thus has a nested loop, the outer loop is over the existing vortices in the domain, and the inner loop is over the grid points. Thus, *!$omp parallel do* directives are used on the outer loops.

Speedup

We test the speedups achieved in the parallelized code. The simulation is run with different number of cores. We now look at the time taken by the advect() subroutine per time step, as plotted for different cores in Figure 5. It can be seen that the parallelized subroutines run almost an order of magnitude times faster than the serial case in Fig 4, due to the worksharing of the nested loops among the threads.

The runtimes for the entire simulation, including all of the subroutines, for 10000 time steps is shown in Figure 6. The runtime of the serial code is plotted at the first (left most) data point, i.e., for one thread. The runtimes are shown to decrease hyperbolically by increasing the number of threads. At 20 threads, the run times are an order of magnitude lower than the serial case. This is because the of the worksharing among different threads achieved by the parallelization. At 20 threads, the runtime achieved is at a minimum, and the speedup is calculated as ~ 7.5. A larger number of threads beyond 20 causes an increase in runtimes instead of a decrease. This may be because the cost of the overheads associated with forking of threads and synchronization becomes large as the number of threads increases beyond 20, and considerable enough to prevent any further speedup. Thus, 20 threads seem to be an optimum for this simulation.

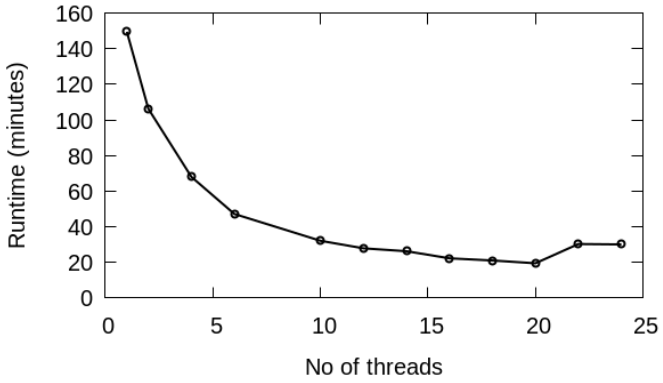


Figure 6. Runtime of the full program up to 10000 iterations as a function of the number of threads.

Solution Accuracy

Next, we check if the accuracy of the solution is compromised to any extent by the parallelization. In Figure 7, we plot two indicators of the progress of the solution at each time step. These are – (a) number of vortices in the domain and (b) averaged recirculation zone length, for different simulations with different numbers of threads. It can be seen that these indicator variable are exactly the same for each case, showing that there is no change in the solution upon parallelization.

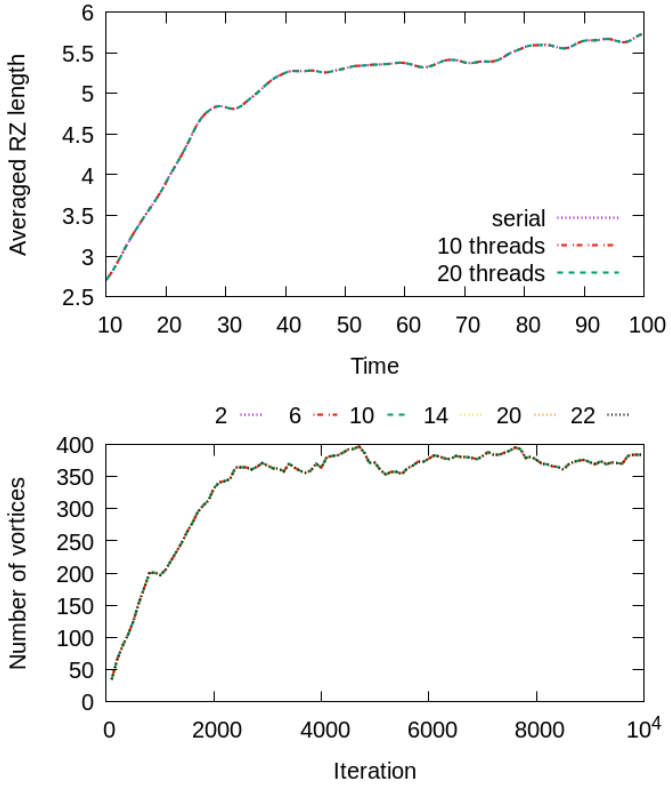


Figure 7. (a) A moving average of the recirculation zone (RZ) length over 100 time steps (b) the number of vortex elements in the domain for various number of cores.

A large source of error in parallelization can occur when reduction clauses are used. The OpenMP specification does not determine the order in which each thread finishes its local computation/sum and add to the global sum. Floating point addition is non-commutative and thus the order of the addition affects the outcome of the sum, and might reflect in the accuracy of the solution. However, in the current parallelization scheme, no reduction clauses are used. The inner loops compute the sums/velocities for each vortex, but it is not these loops that are parallelized, the outer loops are parallelized instead. In addition to achieving a reasonable speedup, this also maintains the accuracy of the solution.

CONCLUSIONS

The role of vortex shedding in predicting combustion instability is non negligible. There is a dearth in reduced order models for the flow field that relax the 1D flow assumption, and account for the effect of the shedding of vortices on the response of the flame. A Lagrangian flow model is thus adopted, the computational complexity of which scales with the number of vortex elements introduced into the domain. The flow field is then obtained on an Eulerian grid. The larger the number of vortices and the finer the size of the grid, the time complexity of the algorithm increases. This provides a large scope for parallelization of the code.

OpenMP is used for parallelization. A profiling of the most time consuming parts of the serially run algorithm is performed and the subroutines that can be parallelized are identified. Using the Fortran *!$omp parallel do* directives appropriately, a maximum speedup of 7.5 is achieved with 20 threads. The speedups increase with an increase in the number of threads. Using more than 20 threads does not lead to any further speedup, and this is stipulated to be because of the increasing cost of the overheads associated with forking and synchronization. Using the recirculation zone length and number of vortices as indicators of solution accuracy, it is shown that the solutions match exactly regardless of the number of threads used. This shows that the nature of the parallelization implemented, i.e., devoid of reduction clauses, precludes the occurrence of different solutions associated with the indeterminism in ordering of additions of local sums, as per the OpenMP implementation. Thus, speedup along with accuracy is well achieved in this work.

Infeasibility of MPI for the current algorithm

It is not feasible to use MPI for this particular algorithm because to compute the velocity on one vortex, information from all the other vortices need to be communicated to and present on each process at each timestep. This means that the entire solution vector/array must be present on all processes at each time step. Usage of the *MPI_Scatter* function will not suffice for this purpose. *MPI_Send* must be used to send the entire array, rather than scatter parts of it. This will be followed by the local process computations, followed by *MPI_Allgather* to update the values on all processes. Unlike in the Jacobi iteration in Assignment 2, it is not just the neighboring elements of the computational matrix/array that need to be communicated, but the entire array itself. Costs incurred by communication may thus be unreasonably larger than the required computations itself.

REFERENCES

Altay, H. M., Speth, R. L., Hudgins, D. E., & Ghoniem, A. F. (2009). Flame-vortex interaction driven combustion dynamics in a backward-facing step combustor. *Combustion and Flame*, *156*(5), 1111–1125. https://doi.org/10.1016/j.combustflame.2009.02.003

Ghoniem, A. F., Annaswamy, A., Wee, D., Yi, T., & Park, S. (2002). Shear flow-driven combustion instability: Evidence, simulation, and modeling. *Proceedings of the Combustion Institute*, *29*(1), 53–60. https://doi.org/10.1016/S1540-7489(02)80011-6

Ghoniem, A. F., Park, S., Wachsman, A., Annaswamy, A., Wee, D., & Altay, H. M. (2005). Mechanism of combustion dynamics in a backward-facing step stabilized premixed flame. *Proceedings of the Combustion Institute*, *30*(2), 1783–1790. https://doi.org/10.1016/j.proci.2004.08.201

Kannan, A., Chellappan, B., & Chakravarthy, S. (2016). Flame-acoustic coupling of combustion instability in a non-premixed backward-facing step combustor: the role of acoustic-Reynolds stress. *Combustion Theory and Modelling*, *11*(4), 658–682. https://doi.org/10.1080/13647830.2016.1166522

Magina, N., Acharya, V., & Lieuwen, T. (2019). Forced response of laminar non-premixed jet flames. *Progress in Energy and Combustion Science*, *70*, 89–118. https://doi.org/10.1016/j.pecs.2018.08.001

Pastoor, M., King, R., Noack, B., Dillmann, A., & Tadmor, G. (2003). Model-based Coherent-structure Control of Turbulent Shear Flows Using Low-dimensional Vortex Models. *AIAA*, 1–11. https://doi.org/10.2514/6.2003-4261

Schadow, K. C., & Gutmark, E. (1992). Combustion Instability Related to Vortex Shedding in Dump Combustors and their Passive Control. *Progress in Energy and Combustion Science*, *18*, 117–132.

Vasanth, J. V, & Chakravarthy, S. R. (2021). A Reduced-order Model for Lock-on via Vortex- combustion-acoustic Closed-loop Coupling in A Step Combustor. *Combustion Science and Technology*, *00*(00), 1–23. https://doi.org/10.1080/00102202.2021.1909578

Wee, D., Yi, T., Annaswamy, A., & Ghoniem, A. F. (2004). Self-sustained oscillations and vortex shedding in backward-facing step flows: Simulation and linear instability analysis. *Physics of Fluids*, *16*(9), 3361–3373. https://doi.org/10.1063/1.1773091