Reply to referee's comments

Reviewers' comments are in plain and the author's reply is in italic text.

General response:

We would like to thank the two Reviewers for their in depth perspicacious comments that contributed to improving the presentation of our paper.

In summary, to address the comments of the reviewers, the following work has been carried out:

1. *The first two cases have been redone to show more convincing convergence as the mesh is refined. For the 1st and 2nd cases, the time step is reduced by a factor to ensure a small Courant number with the smaller elements sizes. We have thus re-plotted the results in Figs 2~9.*

2. *The 3rd case has been redone with a negative concentration background of -0.2 in the subdomain [0.24,0.76]×[0.12,0.88] as suggested by reviewer. The maximum number of nodes for adaptive schemes is set to be 15000. Table 1 and Figs 10~15 have been updated to reflect these new results. A new Fig 16 has been added to show the distribution of CFL number over the domain.*

3. *A new case, case 4 based on a real – large scale atmospheric geometry – and flow, has been added to demonstrate the capability of this new adaptive multiscale model. Figs 17~20 show the results obtained from this new case.*

4. *Case 4 is the simulation of the dispersion of power plant plumes. Diffusion and source terms have therefore been introduced into the equations in section 2.*

5. *Section 3 has been revised and more details of the adaptive mesh techniques have been added.*

6. *Section 2.2 has been rewritten and details of numerical schemes have been provided.*

<u>Anonymous Referee #2</u>

1. It is not clear to me what is novel in this paper, or even whether the goal of the paper is aligned with the aims of this journal. The advection algorithms and also the adaptive refinement algorithms are all implemented in Fluidity, but from the paper it is not at all clear whether the authors of the paper were involved in some new implementation in this version of the code, or are simply testing the code on some particular test problems. The title of the paper, explicitly mentioning

Fluidity 4.1.9, makes it sound like the code is specifically designed for the problem discussed in the paper and the paper serves to describe the full code. However, in Section 4 it is stated that Fluidity solves 2D and 3D Navier-Stokes equations and multiphase flow problems over topography, while this paper only concerns scalar advection in two dimensions. So the paper does not seem to describe or test very much of Fluidity. Moreover there is no real discussion of a "new air quality model" anywhere in the paper. Standard 2D advection test problems are used. Advection equations may be used in air quality models but there does not seem to be anything specific to this application, and advection equations arise in many other situations, so it seems misleading to include this term in the title.

RESPONSE:
While the individual methods – the advection methods, the mesh adaptivity methods – are not novel, this is the first time that the integrated approaches of full 3D adaptive meshes and advanced numerical discretization techniques have been applied to demanding advection-diffusion problems suitable for testing the advection capability of an atmospheric model. This has been clarified in the abstract.

In this work, we used Fluidity version 4.1.9, but not limited to. Thus we deleted Fluidity 4.1.9 from the title. Section 4 is shorten. Fluidity is briefly introduced. However, the N-S equations still remain in section 4 since they are important in our future work.

In this paper, we only focus on integrating this advanced mesh adaptivity methods into air quality modelling. It is well known that the dynamic and chemical processes of air pollution involve a wide range of scales. The initial transformation of emissions from urban and industrial centers and dispersion of plumes occur on relatively small scales, which are responsible for regional or global air quality problems. But it is a gargantuan computational challenge to modeling large regions with uniform resolution at the finest relevant scale. Therefore, mesh adaptation may be a very effective way to encompass different scales (e.g., local, urban, regional, global) in a unified modeling system. An unstructured adaptive mesh model would be the next generation model for air pollution problems. This has been added to the first paragraph in introduction.

The advanced numerical discretization techniques used in the transport air quality model are described in section 2 and adaptive meshes techniques in section 3. Both sections 2 and 3 are updated (see the general response).

In the revised version, to further demonstrate the advantage of adaptive meshes, we added a 3D advection-diffusion case and used realistic wind data and topography, where the mesh was adapted in 3D and time. This is a first step towards applications in realistic cases.

2. Are these specific advection algorithms and/or the adaptive mesh refinement algorithms significantly different in 4.1.9 than they were in 4.1.8? Or are the authors just noting the particular version that they happened to use for these tests of algorithms that have long been a part of Fluidity? If the latter, what is the novel

algorithm or software development? A large number of papers have already been written on advection algorithms of the sort used here, which are often tested on similar problems. The anisotropic refinement algorithm is not described in any detail so it is also not clear if there is anything new here. This all needs to be better clarified.

RESPONSE:
As stated above, the novelty is the integration of methods. An integrated method of advanced anisotropic hr-adaptive mesh and discretization numerical techniques has been, for first time, applied to multi-scale transport-diffusion problems, which is based on a discontinuous Galerkin/control volume discretization on unstructured meshes. This has been clarified in the abstract.

Again, we used Fluidity version 4.1.9, but not limited to.

Section 3 has been re-written. The anisotropic method has been described in detail.

3. The application of the algorithms to the test problems is not well described, e.g. the description on page 4345 of the error metric tensor is inadequate. In (13) it is stated that H is the Hessian matrix, but of what? The full discretization in terms of all degrees of freedom? How are the elements of this tensor used to determine where to refine?

   RESPONSE:
   Please see the updated version of section 3. The formulae of Hessian, interpolation error, minimum and maximum mesh sizes have been provided, and the anisotropic method has been described.

4. It would be very useful if the authors would make the code available to accompany this paper, so that readers could potentially better understand the details of the tests performed. This would also be very useful to any reader who is interested in implementing something similar in Fluidity.

   RESPONSE:
   All the test problems in the paper have been operated in Fluidity model. The source code of Fluidity is available under https://github.com/FluidityProject/fluidity. The user manual and examples are also available. We can offer all setup scripts of the test problems so that the readers can run these test problems directly after installing Fluidity.

5. It is not well explained why it is necessary to use an implicit method for the hyperbolic advection equation, for which explicit methods are more easily implemented and generally preferred for efficiency reasons. It is stated that very large CFL numbers (e.g. 80) are used, and presumably this is because of the highly anisotropic cells with very large aspect ratios. I assume these are stretched in the advection direction, as suggested by Figure 14. Presumably these very high CFL numbers result from comparing e.g. the velocity in the x-direction in this figure to the width of the cells in the y-direction. If the CFL number were truly

this large in terms of the number of grid cells the flow advects through in one time step (e.g. if the flow were in the y-direction in Figure 14) then I believe the implicit method would be extremely dissipative and fairly useless, even if it did remain stable. However, this is not discussed in enough detail to figure out what is going on.

RESPONSE:
In our work, for discontinuous Galerkin discretization, the explicit Euler scheme is used in conjunction with an advection subcycling method based upon a CFL criterion or a fixed number of subcycles. For the CV discretization, the explicit scheme is easier to implement but strictly limited by the CFL number. Here a new timestepping θ scheme is used to eliminate the time-step restrictions and maintain high accuracy as far as possible, where θ ($1/2 \leq \theta \leq 1$) is chosen to 0.5 for most of elements while big enough (close to 1) for a small fraction of individual elements with a large CFL number (see Fig.16). In this way, the use of a large time step is acceptable when applying adaptive mesh techniques into comprehensive air quality models, which can make the computation much more efficient.

This has been clarified in the revised version of section 2.2 and the corresponding numerical schemes have been described in detail.

A new figure (Fig. 16) has been added in case 3, to show the distribution of CFL number over the domain and used to explain the new timestepping θ method.

6. On page 4347, line 25, "advection subcycling" is mentioned but is not explained. Does this mean smaller time steps are used in smaller cells? If so, how are these time steps chosen? Since there is a continuous distribution of cell sizes this is not clear, nor is it clear what is done when adjacent cells are using different size time steps and hence updated a different number of times.

RESPONSE:
For discontinuous Galerkin discretization, an advection subcycling method based upon a CFL criterion or a fixed number of subcycles is adopted in modelling advection flows, that is, the timestep Δt is split to N subtimestep to satisfy the specified Courant number. Further explanation has been added in the revised section 2.2.

7. The anisotropic refinement illustrated in Figure 14 may work well for this flow field in which the streamlines are constant in time and hence the flow is always in a fixed direction at each point in the domain, but it is not at all obvious that the approach used here would work for advection in a real fluid flow (such as the sort Fluidity presumably computes when solving the Navier-Stokes equations, or the sort alluded to in the title of the manuscript). In most flows the direction of flow at each point will be changing dynamically. Even if the adaptive grid is constantly deformed in every time step, the flow would generally not be exactly aligned with the highly anisotropic cells and I suspect this would severely impact the accuracy. All three of the test problems presented in this paper have the feature that the flow directions are time-invariant (even problem 2, where the flow speed varies, has constant direction at each point). I believe the algorithm should be tested on more challenging problems.

RESPONSE:
To demonstrate the capability of the adaptive model and estimation of accuracy of solutions, we added a new case (case 4) to simulate the dispersion of power plant plumes, where, the meteorological fields are provided by the mesoscale meteorological model WRF(v3.5) and stored at hourly intervals during 5-day period. For 2D case, a comparison of results using the fixed and adaptive meshes results is plotted in Figs. 18-19. The results using adaptive meshes are in agreement with those using fixed meshes with a high mesh resolution of 2.5 km while the number of nodes decreases by a factor of 16 with use of adaptive meshes.

We also extended 2D to 3D case, the results are shown in Fig. 20, where the mesh is adapted in 3D and time. It can be seen high resolution meshes are located within the boundary layer and around the power plant stacks.(for details, see section 5.4).

8. The test problems also have large regions of the domain where the solution is constant and hence very few grid cells are needed. This is perhaps reasonable since the point of adaptive refinement is to handle problems where the features needing refinement are relatively isolated. But comparisons of accuracy versus number of cells is then somewhat arbitrary for these problems, since making the domain larger relative to the region where the solution is non-constant would greatly increase the number of grid cells needed for a given resolution on a uniform grid but have no impact on the number of cells needed for the adaptive algorithm. Hence one can make this ratio arbitrarily large by making the domain large, and test problem 3 in particular has a domain that is far larger than reasonable for the given problem.

RESPONSE:
We agree with the reviewer and there is always issue in comparing different methods especially when they are substantially different. None the less this is not a reason not to try to make a comparison. It should be mentioned that these four test problems are benchmark numerical experiments used for testing different numerical advection schemes. We did not make the domain or the ratio larger arbitrarily. But for test problem 3, the initial tracer is spread over only six vortices. Therefore, the 3^{rd} test has been reproduced using the reduced domain $[0.24,0.76]\times[0.12,0.88]$ that cover six swirling vortex containing tracer mass. The results have been presented in Figs 10-15.

9. There is no discussion in the paper of what order of accuracy the advection algorithm is expected to have for smooth solutions, nor even a mention of what order polynomials are used in the continuous or discontinuous Galerkin methods. This is strange, since the presumed advantage of using such methods over simpler and perhaps more efficient finite difference or finite volume methods is that they can achieve higher order. A potential user of Fluidity would surely want to know what orders are supported, along with some evidence that it delivers.

RESPONSE:
The equation for calculation of the order of accuracy has been added (see Eq. 26)

and corresponding discussion has been provided in cases 1-2.

For the discontinuous Galerkin methods, polynomials of different degrees k can be used as discontinuous test and trial functions to avoid taking derivatives of discontinuous functions. Within an element, the functions are continuous, and everything is well defined. In this paper, piecewise quartic shape functions (polynomial degree $k = 4$) are used to achieve high-order accurate.
As an alternative finite volume method, the control volume (CV) methods may be thought of as the lowest order discontinuous Galerkin method, using a dual mesh constructed around the nodes of the parent finite element mesh. In two dimensions this is constructed by connecting the element centroids to the edge midpoints. Once the dual control volume mesh has been defined, it is possible to discretize the advection equation using piecewise constant shape functions within each volume. Although higher-order accuracy is difficult to achieve within the framework of CV method, it is relatively easy to understand and implement using much less computational cost compared with the DG methods.

The CV and DG methods are usually used in conjunction with unstructured meshes, which are very flexible to capture highly complex solutions and are well suited for hr-adaptivity and parallelization. Even though a number of issues remain, in particular those related to the computational cost of models produced using unstructured mesh methods compared with their structured mesh counterparts. Mesh adaptivity represents an important means to improve the competitiveness of unstructured mesh models, where high resolution is only used when and where necessary. This is the major advantage of using such methods.

In the next question, we will discuss the order of accuracy for smooth solutions.

10. None of the test problems have smooth initial data for which this accuracy could be tested. I think some test should be performed of the order of accuracy on smooth data in addition to showing the performance on the sort of data used in the test problems shown.

RESPONSE:
In the first two test problems, we consider a slotted cylinder, a sharp cone, and a smooth hump as the initial solid bodies. The hump as smooth initial data has been considered. In order to discuss what order of accuracy the advection algorithm is expected to have for smooth solutions, we redo the 1st test problem only considering the smooth hump as the initial data. Here, in order to guarantee convergence, it is necessary to use small enough time steps to keep $\Delta t/\Delta x$ fixed as the grid is refined. The effective order of accuracy $p = log_2\left(E_1(h)/E_1(h/2)\right)$ on smooth hump data estimated using $h = 1/200$ equals {1.98, 1.52, 1.54, 1.13} for {CV_Fix, CV_Adapt, DG_Fix, DG_Adapt} schemes respectively.

Due to limitation of pages, we did not add the above smooth case in the paper. However, we mentioned it in section 5.1 by saying:
"If we only consider the hump-smooth profile as the initial data, the order of accuracy can increase to be {1.98, 1.52, 1.54, 1.13}."

11. The error plots in Figures 2 and 5 are logarithmic in x and linear in y, which is

not a useful way to display the error. A log-log plot would make it easier to determine the order of accuracy.

RESPONSE:
Figs. 2 and 5 have been replotted in log-log form. The order of accuracy has been discussed in section 5.

12. Moreover, Figures 2 and 5 also seem to show that the error asymptotes to non-zero values as the grid is refined for most of the methods displayed, which means the methods are not even converging, let alone exhibiting any reasonable order of accuracy. This seems to be a serious problem.

RESPONSE:
The convergence issue was caused by the large time step size. Cases 1 ans 2 have been re-run with a small time step. The figures in sections 5.1 and 5.2 have been re-plotted with the new results.

13. What are the units of CPU time in Figures 2 and 5? Seconds? If so, then apparently the uniform grid DG method in Figure 5 requires 11 hours of CPU time for one revolution of two-dimensional advection on a 400 by 400 grid! Even the adaptive DG code seems to take around 2 hours with h = 1/800, which seems quite excessive for this problem. Of course it would also be useful to state what computer these timings were done on, and how many cores were used since it is stated in the paper that Fluidity uses MPI for parallelization to thousands of cores, although it is not stated whether this is used in these examples.

RESPONSE:
The units of CPU time in Figures 2 and 5 are seconds. We did not use MPI for parallelization. All computations were performed on a workstation using the Gfortran Compiler for Linux. The simulation workstation has 8 processors and a 4GB random-access memory (RAM). The processor used in workstation is Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz. A single processor with frequency of 3.40GHz was used since the test cases were simulated in serial.

This has been clarified in section 5.

14. If I am interpreting the timings right, I suspect the slowness of the code is due to the use of an implicit method for the advection problem. This leads me again to question the wisdom of such methods for this problem, since there are good explicit block-structured AMR algorithms implemented in software such as AMROC, Boxlib, Chombo, Clawpack, SAMRAI, etc. that I believe works quite efficiently on the sort of test problems presented here. More justification is needed for the value of the methods implemented in Fluidity than is presented in this paper.

RESPONSE:
For implicit issue, please see the response to question 5. The slowness of codes is due to the use of unstructured meshes – this is a common issue in unstructured mesh models. In unstructured mesh finite element modelling, it involves the integration of equations over the domain. It thus spends most of time on

assembling the matrices at each time step, especially for nonlinear problems. We used a number of numerical techniques to reduce the CPU time, for example, the timestepping θ scheme to eliminate the time-step restrictions (please see section 2.2 and response to question 5). The use of adaptive meshes will also reduce the number of nodes, thus increasing the computational efficient (please see discussion in cases 1 and 2 in section 5) although it may take time on adapting the mesh at certain time level.

As stated in introduction (or see response to question 9), the unstructured adaptive mesh technique is important in next generation models since it may be the only way to model multi-scale flow dynamical problems in large regions. Due to advanced computational technologies, the issue of CPU times can be sorted out using MPI. Fluidity is parallelized using MPI and is capable of scaling to many thousands of processors.

15. In (15), epsilon is a relative error tolerance that is presumably some positive value chosen by the user, so why is epsilon_min needed to ensure the denominator is nonzero?

RESPONSE:
For example, if $\epsilon = 0.01$, then the tolerance on the denominator of the metric formulation will be 1% of the value of the field c , and so it will scale the target interpolation error with the magnitude of the field. Since the value of the field c may be zero in some region of the domain (e.g., the background of tracer field has been set to be zero in the 1st and 2nd test), it is necessary to set the minimum tolerance ϵ_{min} to ensure that the denominator never becomes zero.

16. On page 4345 line 15, I am not sure what is meant by imposing different limits on the cell sizes in different directions.

RESPONSE:
For robustness of the mesh adaptivity procedure, and to limit refinement/coarsening of the mesh it is possible to set maximum and minimum allowed edge length sizes. The inputs to these quantities are tensors allowing one to impose different limits in different directions. Assuming that these directions are aligned with the coordinate axes allows one to define diagonal tensors. These constraints are achieved through manipulations to the metric, which in turn controls an optimization procedure. They are therefore not hard constraints and one may observe the constraints being broken (slightly) in places.

17. Page 4347, line 20 and I am not sure what is meant by the "Sweby limiter". Sweby's paper discussed many limiter such as minmod, superbee, etc., but I am not sure what Sweby limiter is referred to here.

RESPONSE:
Although Sweby(1984) discussed many limiter functions, the "Sweby limiter" here is not referred to any one of them, but only use the Sweby's TVD region on the normalized variable diagram(NVD) as a criterion to limit the face value calculated by the finite element interpolation approach. Any combination of normalized face

and donor values falling within this region is left unchanged. Values falling outside this region are 'limited' along lines of constant normalized donor value back onto the top or bottom of the stable region. The high order flux is obtained from the finite element interpolation of the CV values(for details, see AMCG, 2014). So, we use the name "CV-TVD limiter" instead of "Sweby limiter" in the revised version.