DELRAPPORT

Projekttitel	Dnr
Utveckling av ett numeriskt beräkningsverktyg med en öppen källkod för att skapa en säkrare arbetsmiljö inom svensk industri avseende dammexplosioner	180028

Projektledare
Chen Huang at RISE Research Institutes of Sweden

Innehåll:

Development of a numerical tool using an open source code for creating a safer working environment for the Swedish industries regarding dust explosions

Part report (from 2019-02-01 until 2020-01-31)

Project members

Chen Huang	RISE
Anders Lönnermark	RISE
Andrei Lipatnikov	Chalmers

Reference Group Members

Ken Nessvi	PS Group
Ingemar Klaesson	Scandbio
Patrik Carlryd	AVS
Mats Perslid	IEP Technologies
Henrik Larsson	Göteborgs Energi
Nijaz Basic	Fagerberg
Ricardo Capitao	Fagerberg
Kristoffer Tannerfeldt	Firefly
Håkan Nilson	Chalmers
Chris Cloney	Dust-Ex Research (Canada)
Vladimir Molkov	Ulster University (UK)

Sammanfattning (max 2000 tecken inklusive mellanslag)

Dammexplosioner är ett konstant hot mot de svenska industrier som hanterar material eller utför processer som skapar brännbart damm, såsom pelletstillverkare, livsmedelsindustri, metallindustri m.m. Det aktuella projektet syftar till att (i) utveckla välvaliderade numeriska modeller som kan ta hänsyn till de viktigaste förbränningsfenomenen, (ii) utveckla ett numeriskt verktyg baserat på en öppen källkod, och (iii) beräkna verkliga dammexplosionsscenarier i samråd med representanter för berörda industrier. Projektresultatet kan fylla kunskapsluckorna när det gäller förståelse för dammexplosioner, att uppskatta konsekvenser av dammexplosioner, ge rekommendationer för bättre konstruktion av byggnader och relevanta säkerhetssystem, och därmed ge personalen en säkrare arbetsmiljö.

Under det första året, har den öppna källkodsplattformen OpenFOAM installerats och testats. Den så kallade FSC (Flame Speed Closure) modellen för förblandade turbulenta flammor implementerades i OpenFOAM. Implementeringen av FSC-modellen har verifierats mot analytiska lösningar för 1-D plana och 3-D sfäriska förblandade turbulenta flammor. Verifikation av implementationen visar att modellen implementerades korrekt. För närvarande är den numeriska modellen under validering mot småskaliga experimentella resultat för 3-D sfäriska flamma i Leeds förbränningskärl. De första beräkningarna visar att modellen och koden predikterar trenden. Det vill säga, flamhastigheter ökar när turbulenta hastighetsfluktuationer ökar. Beräkningar visar också att modellen och koden även kvantitativt predikterar flamhastigheter om rimliga modelleringsparametrar används.

I nästa steg, kommer modellen och koden utvecklas ytterligare för att ta hänsyn till värmeförluster och strålning. Därefter kommer beräkningsresultaten att jämföras med experimentella resultat från de storskaliga tryckavlastningsförsöken, med olika geometrier, utförda vid Rembe® Research and Technology Center.

Abstract

Dust explosion is a constant threat to the Swedish industries which deal with combustible powders such as pellets producers, food industry, metal industry and so on. This project aims at (i) development of high-fidelity and well-validated models which address important combustion phenomena during a dust explosion, (ii) development of an efficient numerical tool based on an open source toolbox for predicting consequences of dust explosions and (iii) simulation of dust explosions in scenarios of process industries in cooperation with the reference group members of this project. The project result will improve the understanding of complicated combustion phenomena associated with dust explosions, and it will help the process industries in designing better vent system in case of dust explosion.

During the first year, the Flame Speed Closure (FSC) model for premixed turbulent combustion, has been implemented in the open source platform OpenFOAM, which was installed at RISE in the beginning of the project. The implementation of FSC model has been verified against analytical solutions for 1-D planar and 3-D spherical turbulent flames. Verification shows correct implementation of model in the OpenFOAM platform. Currently, the developed code is being validated against small-scale dust-explosion experiments performed using the well-known Leeds combustion vessel. The first test of the code show that the trend, i.e. an increase in turbulent velocity fluctuation, an increase in flame speed, is predicted by the code. A further test shows that the code and model can predict the flame speed quantitatively using proper model parameters.

In the next step, the model and code will be developed for considering the heat losses and radiation. Later the developed numerical platform will be applied to unsteady 3-D RANS simulations of large-scale experiments performed at REMBE® Research and Technology Center for vent relieving with different vent geometry.

Contents

Nomenclature

Roman symbols

A	a constant of the Flame Speed Closure model [-]
b	combustion regress variable [-]
c	combustion progress variable [-]
C_d	coefficient for evaluating turbulent length scale [-]
C_μ	standard $k-\varepsilon$ turbulence model constant [-],C_μ=0.09
C_1	standard $k-\varepsilon$ turbulence model constant [-],C_1=1.44
C_2	standard $k-\varepsilon$ turbulence model constant [-],C_2=1.92
D_t	turbulent heat diffusivity [m^2/s]
$Da = \tau_t/\tau_c$	Damköhler number [-]
$k = 3/2\,u'^2$	turbulent kinetic energy [m^2/s^2]
L	integral length scale of turbulence [m]
p	pressure [Pa]
Pr	Prandtl number [-]
$R^0 = 8.314$	universal gas constant [J/(mole·K)]
Re	turbulent Reynolds number [-]
R_f	flame position [m]
Q	extra source term (see Eq. (Appendix II.2))
S	flame speed [m/s]
T	temperature [K]
t	time [s]
t_{fd}	flame development time [s]
t_r	reaction time scale [s] (see Equation (Appendix II.2))
t_0	ignition model parameter [s] corresponding to ignition duration (see Eq. (Appendix II.3))
U	burning velocity [m/s]
$\mathbf{u}$	flow velocity vector [m/s]
u'	rms turbulent velocity [m/s]
W	molecular weight [kg/mol]
W_{ign}	ignition source term (see Eq. (Appendix II.3))
W_0	ignition model parameter (see Eq. (Appendix II.3))
x	spatial coordinate [m]

Greek symbols

α	molecular heat conductivity [kg/(m·s)]
Δ_t	mean flame brush thickness [m]
ε	turbulent dissipation rate [m^2/s^3]
ϕ	equivalence ratio [-]
κ	molecular heat diffusivity [m^2/s]
μ	molecular dynamic viscosity [kg/(m·s)]
ν	molecular kinematic viscosity [m^2/s]
ξ	normalized distance [-]
Θ	activation temperature [K]
σ	density ratio [-] $\sigma = \rho_u/\rho_b = (T_b W_u)/(T_u W_b)$
σ_r	ignition model parameter [m] corresponding to ignition kernel size (see Eq. (Appendix II.3))
σ_k	standard $k-\varepsilon$ turbulence model constant [-], σ_k=1.0
σ_ε	standard $k-\varepsilon$ turbulence model constant [-], σ_ε=1.3
σ_t	ignition model parameter [s] $\sigma_t = t_0/5$. (see Eq. (Appendix II.3))
ρ	density [kg/m^3]
τ_c	chemical time scale [s]
τ_t	turbulent time scale [s]

Subscripts

b	combustion products
f	flame
L	laminar
t	turbulent
u	unburned mixture
0	initial condition
∞	fully developed, asymptotically steady quantities

Superscripts

$\bar{}$	ensemble-averaged or Reynolds-averaged
$\sim$	Favre-averaged

Acronyms

1-D	1-dimensional
3-D	3-dimensional
BML	Bray-Moss-Libby
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
FSC	Flame Speed Closure
GPL	General Public License
LHS	Left Hand Side
OpenFOAM®	Open Field Operation and Manipulation
RAM	Random Access Memory
RANS	Reynolds-averaged Navier–Stokes
RHS	Right Hand Side
rms	root-mean-square

1. Project aim and expected result

Dust explosion is a great threat to the industry worldwide especially among countries and regions with high industrial output. Also, in Sweden dust explosion represents a constant threat to the working environment of industries which deal with combustible powders. There is at least one dust explosion accident reported to Arbetsmiljöverket per month [1], and it is highly possible that there are many more unreported accidents available. Some examples from 2017 are

- an explosion of a truck loaded a 20-meter diameter silo with pellets in Gothenburg on 7th March [2],
- a metal dust explosion in an aluminium pigment company in Huskvarna on 28th March [3],
- a severe explosion occurred during cleaning of a dust extractor in Trångsund [4] on 15th August (seven people were injured that time),
- a dust explosion in a 40-meter-high grain silo in Tråvad on 29th August [5].

According to Afa-försäkrings statistical analysis, 838 workers were injured severely due to fire, explosion, welding etc., during 2012 and 2013 [6]. Nine more dust explosion incidents were reported by the media in Sweden during 2018 [7-16], and there is no sign of indicating the declining of dust explosion incidents in Sweden.

After going through the most recent dust explosion incidents in Sweden, the question arises: why are there so many dust explosions? We believe that the main reason is the lack of knowledge in the complicated combustion process during a dust explosion and the lack of numerical tools for designing explosion protection systems.

So, what exactly is a dust explosion? It is a complicated physical and chemical process, when very fine combustible particles well mixed with air in confined equipment are ignited, resulting in violent and explosive burning. Once the dust explosion occurs, the high-pressure waves, hot flames and extremely radiative heat may cause serious loss of life and severe economic consequences.

The next question will be: how could we reduce the consequences of dust explosions? One important solution would be having access to high-fidelity and well-validated models and an efficient numerical tool. Specifically, the numerical tool can be used to design explosion venting protecting systems for process plants with complicated geometries where the standards are not applicable.

A suitable platform for developing dust explosion models is the code OpenFOAM (Open Field Operation and Manipulation). It is a free, open-source general-purpose Computational Fluid Dynamics (CFD) software package mainly for simulating thermodynamics, fluid dynamics, and chemical reactions. On the technical side, OpenFOAM excels in modern architecture using object-orientated programming language, high parallelization and unstructured grid for dealing with curved geometry. New models and methods can be easily implemented and tested thanks to the open source. Furthermore, it creates more value for the customer since it is possible to create the tailor-made tool that suits the special need of the customer at zero license cost.

For the above reasons, this project **aims** at (i) improving the understanding of complicated combustion phenomena associated with dust explosions such as flame expansion, turbulence generation by a flame and flame acceleration, (ii) providing an OpenFOAM-based numerical tool for accurately estimating the consequence of dust explosions, (iii) helping the relevant industries to develop mitigation strategies such as better pressure relief system for reducing dust explosion consequences.

The expected project **result** will be a numerical tool based on the open source code with detailed documentation for designing safer explosion venting system at industrial plants. In the long-term perspective, the number of severe accidents caused by dust explosion is expected to be reduced due to safer and more efficient pressure relief systems for reducing dangerous pressure build-up in case of dust explosions.

2. Project progress

First, the OpenFAOM software was installed from scratch on RISE computer with Linux operation system.

Second, a model of turbulent burning of a dust cloud was implemented into the OpenFOAM platform.

Third, the implementation was **verified** by comparing the computed results with exact and approximate analytical solutions obtained for 1-D laminar flame and statistically planar or spherical flame propagating in "frozen" turbulence. "Frozen" means that turbulence characteristics such as turbulent kinetic energy and turbulent dissipation rate are kept constant throughout the simulation. In other words, the influence of combustion on turbulence is ignored. Such a simplification is required to find analytical solutions [17] to the aforementioned problems, thus, allowing direct quantitative verification of the model implementation.

Fourth, different versions of the model were studied for 1-D and 3-D laminar and turbulent flames. Sensitivity of computed results to constants of ignition, combustion, and turbulence models was studied. A reasonable set of the models' constants was selected for studying dust cloud explosion at the next step of project.

If no special statement is made, the following software and hardware is used in this project: OpenFOAM version 1812 and a RISE computer called "srv-simlab-02" with 128 GB RAM and totally 28 Xeon Gold cores.

2.1. Brief introduction of OpenFOAM

OpenFOAM (Open Field Operation and Manipulation) code is a free, open source general CFD software package for simulating thermodynamics, fluid dynamics, chemical reactions, solid dynamics and electromagnetics. The code solves various partial differential equations using finite volume method on unstructured mesh. OpenFOAM is licensed under GNU General Public License (GPL) v3 which means users have great freedom in using the code. More specifically, users are free to use the software for any purpose (commercial or non-commercial); users are free to make change to the software; users are free to share the software with the changes they have made.

OpenFOAM has been attracting growing interests from both industries and academies since its release in 2004. Researchers are strongly interested in access to source codes in order to develop and to implement new models and to easily exchange knowledge and experience with each other. Moreover, OpenFOAM has a very attractive feature; it is written in object-oriented language C++. Accordingly, solvers, written using the OpenFOAM classes, closely resemble the corresponding partial differential equations. Furthermore, OpenFOAM has inbuilt parallelization. That means parallelization is integrated at a low level, e.g. new application does not need parallel-specific coding since it runs in parallel by default. However, besides the abovementioned advantages of OpenFOAM, since simple documentations are commonly embedded in the source code, the learning curve of OpenFOAM is steeper than a well-documented program, e.g. Ansys Fluent.

The overall OpenFOAM structure is shown in Figure 1. The workflow of using OpenFOAM is like a conventional CFD program, and it includes pre-processing, solving and post-processing. First, OpenFOAM has its own mesh generation utilities, such as blockMesh, snappyHexMesh and cfmesh. Such meshing utilities have limitations of either for simple geometry or less user-friendly, which restrict the usage of OpenFOAM in industrial applications with complicated geometry. However, OpenFOAM mesh is compatible with the mesh format of most common commercial CFD programs. After the computational mesh is ready, there are various kinds of solvers designed to solve specific computational continuum mechanics. OpenFOAM offers a set of libraries which are dynamically linked to the solvers, and the libraries serve as the source code of physical models. Finally, post-processing of computed results especially for data visualization can be achieved using both an open source program ParaView, and commercial programs, e.g. EnSight and Tecplot. Moreover, there are utilities for data acquisition as well.

OpenFOAM is available for the Linux, Mac and Windows operating system. Currently the owner of OpenFOAM - ESI releases both source code and pre-compiled binaries for Linux, Mac and Windows system, and users can freely download the source code from the internet. Usually compilation of OpenFOAM from source code requires knowledge of Linux operation system and capability to work in

terminals using commands. Moreover, the installation of ParaView, which is an open source data visualization application released together with OpenFOAM, is available for both Linux and Windows operation systems.

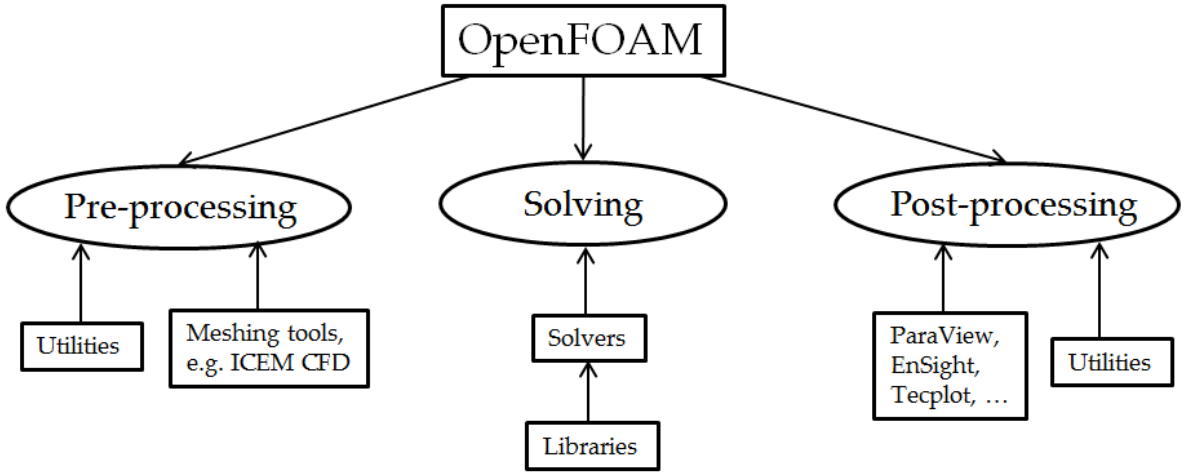


Figure 1 Overview of OpenFOAM structure.

2.2. Model for turbulent burning of dust cloud

The dust explosion process resembles that of a turbulent burning of a gas cloud. This is especially true for very fine organic dust particles with high volatile content [18-20]. Therefore, a premixed turbulent combustion model is commonly used for modelling of dust explosions [21]. In this work, we start with a so-called Flame Speed Closure (FSC) model of the influence of turbulence on premixed burning [22]. The main reason is that the FSC model has been quantitatively validated against a wide set of experimental data obtained by various research groups from various flames under a wide range of substantially different conditions (various fuels, equivalence ratios, temperatures, pressures, and turbulence characteristics) [22, 23].

The FSC model characterizes the state of the mixture in a flame using a single combustion progress variable c, where combustion progress variable has a physical meaning of the probability of finding burned products. A detailed description of the FSC model is reported in Appendix I.

2.3. Model implementation into OpenFOAM

One of the biggest advantages of using OpenFOAM is that users have access to the source code and have possibility to implement and to test new models. Moreover, OpenFOAM is written in object-oriented language. Accordingly, coding a balance equation is straightforward. A solver developed within the framework of the present project is based on the solver XiFoam available in OpenFOAM. Following XiFoam, the developed solver deals with a transport equation for the Favre-averaged regress variable $\tilde{b}$, which is equal to $1-\tilde{c}$. Based on the FSC model, the equation is written as follows

$$\underbrace{\frac{\partial \bar{\rho}\tilde{b}}{\partial t}}_{1}+\underbrace{\nabla.\left(\bar{\rho}\tilde{\mathbf{u}}\tilde{b}\right)}_{2}\underbrace{-\nabla.\left[\bar{\rho}(\kappa+D_t)\nabla\tilde{b}\right]}_{3}+\underbrace{\rho_u U_t\left|\nabla\tilde{b}\right|}_{4}$$
$$+\underbrace{\frac{\bar{\rho}\tilde{b}}{t_r(1+D_t/\kappa_b)}\exp\left(-\frac{\Theta}{\tilde{T}}\right)}_{5}+\underbrace{\bar{\rho}W_0\exp\left\{-\left[\left(\frac{r}{\sigma_r}\right)^2+\left(\frac{t-t_0}{\sigma_t}\right)^2\right]\right\}\tilde{b}}_{6}=0, \quad (1)$$

where terms 1-5 pertain to the FSC model, whereas term 6 is used to ignite the mixture. This source term is active during a short time interval on the order of σ_t and creates a small flame kernel whose radius is on the order of σ_r. Overlines denote the Reynolds average ($q=\bar{q}+q'$), while $\tilde{q}=\overline{\rho q}/\bar{\rho}$ is the Favre-averaged value of q ($q=\tilde{q}+q''$). The derivation of regress variable equation by using progress variable equation to is reported in Appendix II.

Equation (1) written in the object-orientated language in OpenFOAM is shown below

```
  // Create b equation
   // ~~~~~~~~~~~~~~~~~
   fvScalarMatrix bEqn
   (
      fvm::ddt(rho, b)                                  // term 1: transient
    + mvConvection->fvmDiv(phi, b)                      // term 2: convection
    - fvm::laplacian(alpha_turb_transient+alpha_thermo, b)   // term 3: diffusion
    + fvm::div(phiSt, b)                                // term 4: gradient
    - fvm::Sp(fvc::div(phiSt), b)                       // term 4: gradient
    + fvm::SuSp(rho*W_0*W_ign_coeff,b)        // term 5: ignition, depending on the sign
    + fvm::SuSp(rho/tr/(1.+alpha_turb_transient/alphab_thermo)*exp(-Ta/thermo.T()),b)// term 6:
extra source term
    ==
      fvOptions(rho, b)
   );
```

In the default library of OpenFOAM, the calculation of Favre-averaged temperature $\tilde{T}$ and Reynolds-averaged density $\bar{\rho}$ does not follow the well-established Bray-Moss-Libby (BML) framework [24], as discussed in detail in Refs. [23, 25]. This limitation of the OpenFOAM library leads to incorrect solution in the flame brush zone, and it may lead to error message of "out of temperature range 200 -> 5000" when running the code.

In the present work, the calculation of $\tilde{T}$ and $\bar{\rho}$ follows the BML framework, i.e.

$$\tilde{T} = T_u\tilde{b} + T_b\left(1-\tilde{b}\right) \tag{2}$$

$$\frac{1}{\bar{\rho}} = \frac{1}{\rho_u}\tilde{b} + \frac{1}{\rho_b}\left(1-\tilde{b}\right) \tag{3}$$

Equations (2) and (3) are implemented by modifying the calculate() function in the library located in $FOAM_USER_SRC/thermophysicalModels/reactionThermo/psiuReactionThermo/heheuPsiThermo.C. The detailed implementation of solver and library is reported in Appendix V.

The laminar and turbulent viscosity and heat diffusivity used by the model, e.g. κ and D_t in Equation (1), are reported in Appendix VI.

2.4. Verification of model implementation

To verify the implementation of the model into OpenFOAM, three sets of analytical solutions to simplified versions of Equation (1) were used. These were:

1. the classical approximate travelling-wave solution to planar 1-D Equation (1) obtained by Zel'dovich and Frank-Kamenetskii [26] in the case of a laminar flame ($D_t = 0$; $U_t = 0$; $W_0 = 0$).
2. an exact analytical solution to planar 1-D Equation (1) without terms 5 and 6 (i.e. $t_r \to \infty$ and $W_0 = 0$), derived in Ref. [17] in the case of frozen turbulence (neither D_t, nor U_t varies in space).
3. an approximate analytical solution to spherically symmetrical 1-D Equation (1) without terms 5 and 6 (i.e. $t_r \to \infty$ and $W_0 = 0$), derived in Ref. [27] in the case of frozen turbulence (neither D_t, nor U_t varies in space).

These solutions are discussed in Appendix III.

3. Achieved results up to now

3.1. Verification of model implementation

In order to verify the implementation of premixed turbulent combustion model in OpenFOAM, we simulated simplified cases where benchmark analytical solution could be obtained, e.g. 1-D planar laminar premixed flame, 1-D planar flame propagating in "frozen" turbulence, and 3-D spherical flame propagating in "frozen" turbulence. Subsequently, results computed using the newly implemented models were compared with the analytical solutions. It is worth remembering that simulations of the turbulent flames were performed using the truncated FSC model, i.e. terms 5 and 6 were omitted in Equation (1), because the analytical solutions had been obtained for that truncated transport equation.

The implementation of the FSC model was verified using three options, i.e. (i) comparison of the normalized profile of the Reynolds-averaged progress variable $\bar{c}$ with the analytical solution given by Equation (Appendix III.1), (ii) comparison of the computed growth of the mean flame brush thickness with the turbulent diffusion law given by Equation (Appendix III.4), and (iii) comparison of the computed flame speed with the turbulent flame speed yielded by the FSC model expressions given by Equation (Appendix I.3) in the planar case or Equation (Appendix III.5) in the spherical case.

The premixed burning of cornflour dust cloud was simulated. The cornflour chemical equivalent formula is $C_6H_{7.88}O_{4.98}$ with a heat of reaction being 15.8 MJ/kg [18]. The chemical reaction of cornflour is as follows

$C_6H_{7.88}O_{4.98} + 5.48\ (O_2 + 3.71\ N_2) = 6CO_2 + 3.94H_2O + 20.331\ N_2$

3.1.1. Truncated FSC model: 1-D planar flame in "frozen" turbulence

The 1-D model has a domain size of 0.1 m, and a cross section size of 0.003 × 0.003 m, Figure 2. There are 100 cells along the *x* direction, and three cells in the *y* and *z* directions, respectively. Burned products occupy the left-hand side (LHS), whereas unburned reactants occupy the right-hand side (RHS). Zero velocity and free entrainment boundary conditions are set on the right (unburned) and the left (burned) boundaries, respectively. In the latter case, the pressure at the boundary for compressible subsonic flow is calculated using the following equation

$$p_p = p_0 - \frac{1}{2}\rho|\tilde{\mathbf{u}}|^2 \tag{4}$$

where p_p is the pressure at the boundary patch, and p_0 is the total pressure.

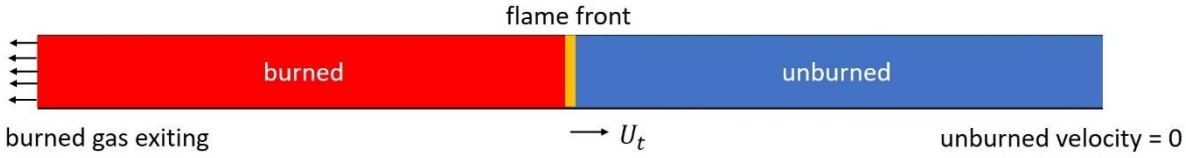


Figure 2 Layout of 1-D planar flame.

The premixed turbulent flame propagates from burned to unburned side. The thermo-physical properties of unburned and burned mixture, the initial and boundary conditions are summarized in Table 1, Table 2 and Table 3, respectively. The case setup is shown in Appendix VII.

Table 1 Thermo-physical properties of unburned and burned mixtures.

quantities		value
unburned	T_u [K]	328
	W_u [g/mol]	32.76
	ρ_u [kg/m^3]	1.32138
	μ_u [kg/(m·s)]	1.8e-5
burned	T_b [K]	1599
	W_b [g/mol]	27.15
	ρ_b [kg/m^3]	0.2255
	μ_b [kg/(m·s)]	4.6e-5
others	$\sigma = \rho_u/\rho_b = (T_b W_u)/(T_u W_b)$ [-]	5.86
	Prt	0.7
	S_L [m/s]	0.12

Table 2 Initial conditions for 1-D planar flame.

quantities	value
T_0 [K]	328
P_0 [Pa]	11 000
$\tilde{k}$ [m^2/s^2]	0.96
u' [m/s]	0.8

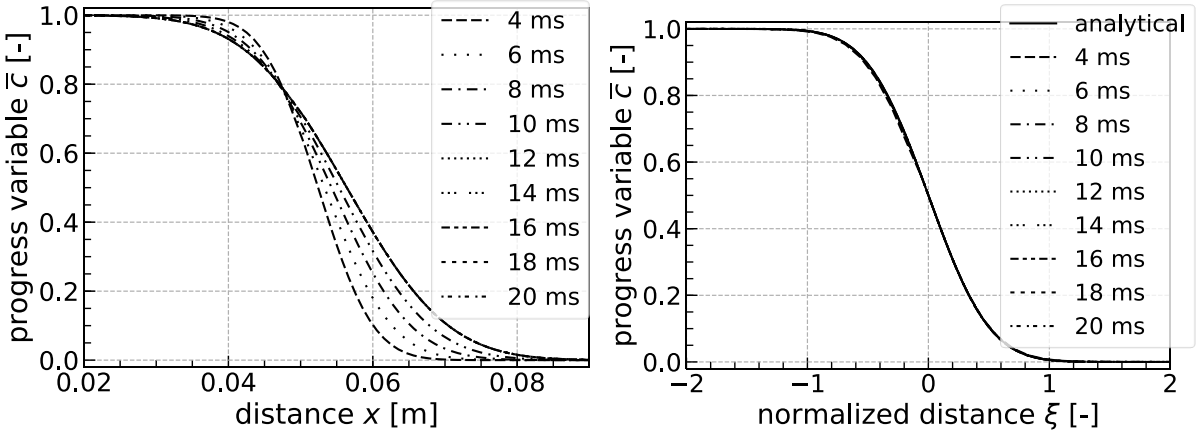
Table 3 Different turbulent length scales

cases	1	2	3
$\tilde{\varepsilon}$ [m^2/s^3]	11.84	69.6	348
L [m]	0.029	0.005	0.001

Table 4 Boundary conditions for x direction.

	burned (left)	unburned (right)
P [Pa]	totalPressure	fixedValue 11 0000
$\tilde{\mathbf{u}}$ [m/s]	pressureInletOutletVelocity	fixedValue (0 0 0)
$\tilde{T}$ [K]	zeroGradient	fixedValue 328
$\tilde{b}$ [-]	zeroGradient	fixedvalue 1

The calculated profiles of the Reynolds-averaged combustion progress variable $\bar{c}$ versus distance change with time; see Figure 3 (a). However, the profile of $\bar{c}$ versus normalized distance ξ is the same for all the time instances, in line with the well-documented self-similarity of premixed flames [17]. It can be seen in Figure 3 (b) that the profiles of $\bar{c}$ versus ξ for different time instants and the analytical solution (see Equation (13) in Ref. [17]) agree very well.



(a), Reynolds-averaged combustion progress variable $\bar{c}$ vs distance (b), $\bar{c}$ vs normalized distance
Figure 3 Spatial profiles of the Reynolds-averaged combustion progress variable for 1-D planar flame propagating in "frozen" turbulence.

Comparison between calculated and analytical flame speeds is shown in Figure 4. In the simulations,

the flame speed was evaluated by taking derivative of mean flame position against time. The mean flame position is defined by the *x*-coordinate of a surface $\bar{c} = 0.5$. Since the calculated flame position exhibited fluctuations, UnivariateSpline function in scipy library of Python was used to smooth the data. Figure 4 shows that the numerical and analytical results agree well. There is a slight discrepancy in the beginning and in the end of the curve. This may be caused by the smoothing function and the uncertainty caused by the numerical schemes, time-step and grid size.

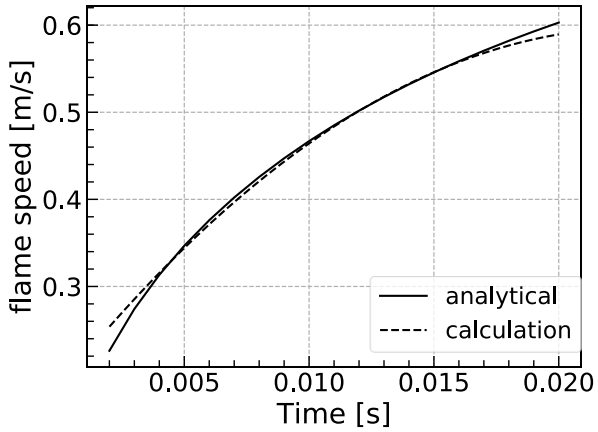


Figure 4 Comparison of calculated flame speed with flame speed yielded by FSC model for 1-D planar flame propagating in "frozen" turbulence.

Comparison between mean flame brush thickness evaluated by processing the computed profiles of $\bar{c}(x,t)$ using Equation (Appendix III.3) and the analytical solution given by Equation (Appendix III.4) is shown in Figure 5. Overlapping of solid and dashed lines verifies the implementation of the FSC model.

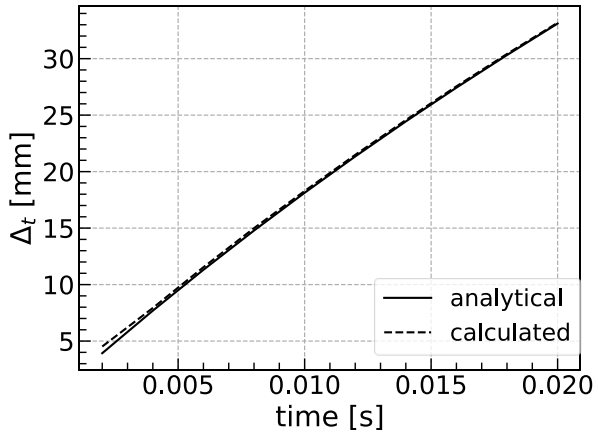
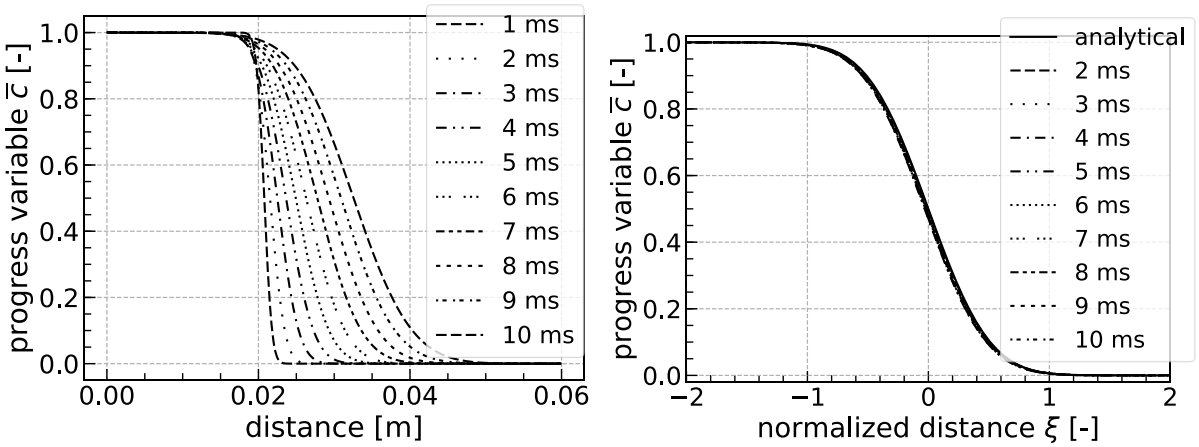


Figure 5 Comparison of calculated mean flame brush thickness Δ_t with turbulent diffusion law given by Eq. (Appendix III.4) for 1-D planar flame propagating in "frozen" turbulence.

3.1.2. Truncated FSC model: 3-D spherical flame in "frozen" turbulence

A cube geometry was created in order to represent one eighth of the total computational domain. The computational domain had an edge of 60 mm and a mesh size of 0.25 mm. Totally 13 824 000 cells were created. The model took around 42 wall clock hours running on Simlab computer using 14 cores. The initial condition corresponded to a spherical kernel of a radius of 20 mm, filled with combustion products. The rest of the domain was filled with unburned mixture. The detailed case setup is shown in Appendix VIII. The initial conditions, boundary conditions, and thermos-physical properties were the same as for 1-D planar case; see Table 1, Table 2 and Table 3. To perform comparison with approximate analytical solution, results were obtained using the FSC model without extra source term Q; see Equation (Appendix II.1).

The Reynolds-averaged progress variable $\bar{c}$ was evaluated by taking the average value of $\bar{c}$ along *x*, *y* and *z* directions. The computed dependencies of $\bar{c}$ on the normalized distance ξ are overlapping and agree well with the complementary error function; see Figure 6 (b).



(a), Reynolds-averaged combustion progress variable $\bar{c}$ vs distance (b), $\bar{c}$ vs normalized distance

Figure 6 Spatial profiles of the Reynolds-averaged combustion progress variable for 3-D spherical flame propagating in "frozen" turbulence.

For an expanding spherical flame, the burning velocity is lower than that of a planar due to the influence of the mean curvature of the flame brush. The FSC model permits an analytical estimate of the magnitude of such a reduction effect. As shown elsewhere [28], the effect magnitude is equal to $\int_0^\infty \tilde{c}rdr\left\{\int_0^\infty \bar{c}rdr\right\}^{-1}$. To verify the implementation, a ratio of the computed turbulent flame speed S_t (with respect to unburned mixture) and the theoretical turbulent burning velocity S_t given by Equation (Appendix I.3) was compared with the integral ratio calculated using the analytical solution given by Equation (Appendix III.1). To do so, (i) S_t was calculated by differentiating filtered curve plotted in Figure 7 and (ii) since the complementary error function in this solution involved a normalized distance, the integrals of $\int_0^\infty \tilde{c}rdr$ and $\int_0^\infty \bar{c}rdr$ were also evaluated using the normalized distance, e.g.

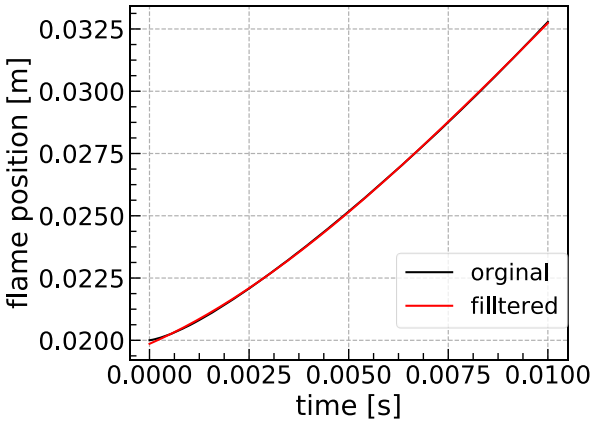


Figure 7 Calculated flame position (original and filtered) versus time for 3-D spherical flame propagating in "frozen" turbulence.

As shown in Figure 8, the two ratios, i.e. $\int_0^\infty \tilde{c}rdr\left\{\int_0^\infty \bar{c}rdr\right\}^{-1}$ and S_t/U_t, are sufficiently close to one another, thus, further verifying the model implementation. The quantities are not exactly equal, because the analytical estimate of the reduction effect magnitude is an approximate one for the 3-D spherical flame.

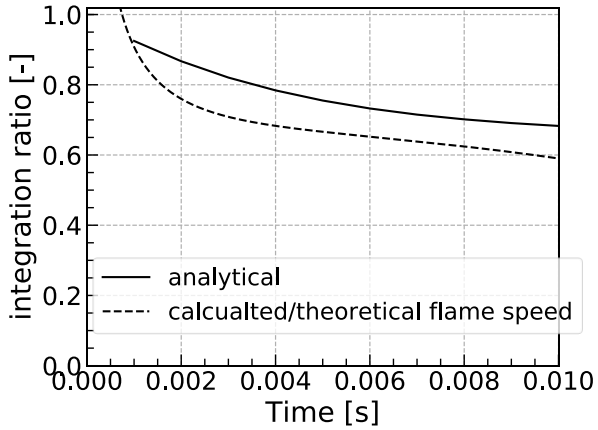


Figure 8 Comparison of analytical integral ratio with the ratio of calculated flame speed with respect to unburned mixture divided by theoretical turbulent flame speed by Equation (Appendix I.3) for 3-D spherical flame propagating in "frozen" turbulence.

Since the complementary error function is evaluated based on a normalized distance, the integration $\int_0^\infty \tilde{c}rdr$ should also be evaluated using the same normalized distance, i.e.

$$\int_0^\infty \tilde{c}rdr = \Delta_t \int_{-\frac{r_f}{\Delta_t}}^\infty \tilde{c}\left(\Delta_t \xi + r_f\right)d\xi \tag{5}$$

where $\xi = \frac{r-r_f}{\Delta_t}$, and $dr = \Delta_t d\xi$.

Figure 9 shows that analytical and numerical results for the mean flame brush thickness are close to each other. It is worth stressing that, contrary to the 1-D planar case discussed earlier, the analytical equation is not exact and, consequently, some mild differences between this equation and numerical data are expected.

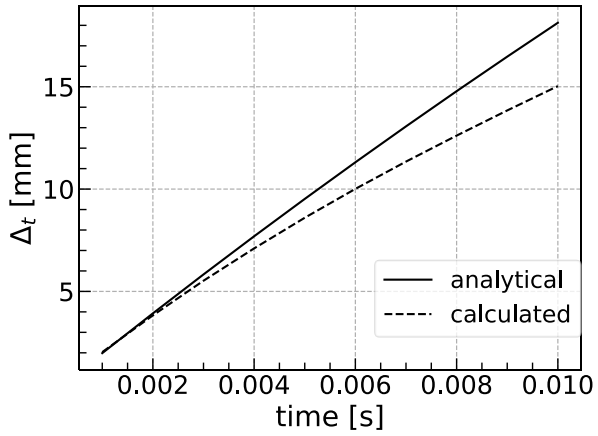


Figure 9 Comparison of calculated mean flame brush thickness with Eq. (Appendix III.4) for 3-D spherical flame propagating in "frozen" turbulence.

3.1.3. Truncated FSC model: Influence of ignition model

Results reported in the previous section were obtained by omitting term 5 in Equation (1) (otherwise an analytical benchmark solution is difficult to obtain) and using $W_0 = 0$ in term 6, with ignition being simulated by creating a product kernel at the initial instant. Alternatively, ignition could be simulated using uniform initial conditions of $\tilde{b}(\mathbf{x}, t = 0) = 1$ and a large value of W_0. Results of such simulations are discussed in the present subsection.

Black dashed line in Figure 10 shows that the use of the latter ignition model yields too small mean flame brush thickness for the turbulent length scale of 5 mm. Examination of the numerical data indicated that the computed thickness was too small, because the gradient $|\nabla\bar{c}|$ was too much in the vicinity of the kernel centre. This problem was solved by using the complete version of the FSC model, i.e. by retaining term 5 in Equation (1). Relevant results will be reported in sections 3.1.4 and 3.1.5.

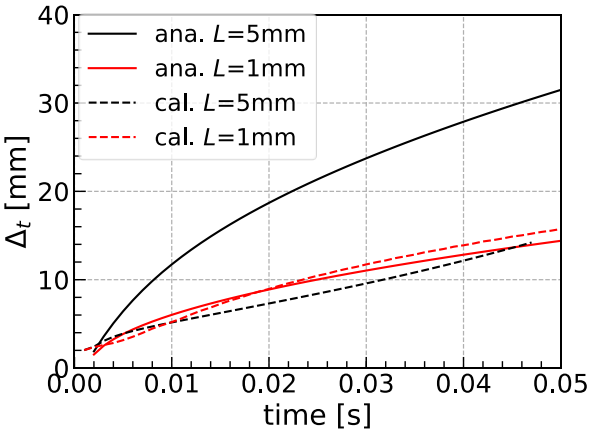


Figure 10 Comparison of calculated mean flame brush thickness against turbulent diffusion law in Eq. (Appendix III.4) for turbulent length scale 1, and 5 mm for 3-D "frozen" turbulence spherical flame.

3.1.4. Complete FSC model: 1-D laminar planar flame

To verify implementation of the complete FSC model, we still have to verify implementation of term 5 in Equation (1). To do so, 1-D planar laminar flame was simulated, because this term is of the most importance under such conditions. Indeed, when turbulent Reynolds number Re_t is increased so that $Re_t \gg 1$, the magnitude of this term is significantly reduced due to a ratio of $D_t/\kappa_b \propto Re_t$ in the denominator. Another goal of such simulations was to find out a value of the reaction time scale t_r that yielded the required value of the laminar flame speed $S_L = 0.12$ m/s.

The cornflour laminar flame thickness δ_L is evaluated as follows

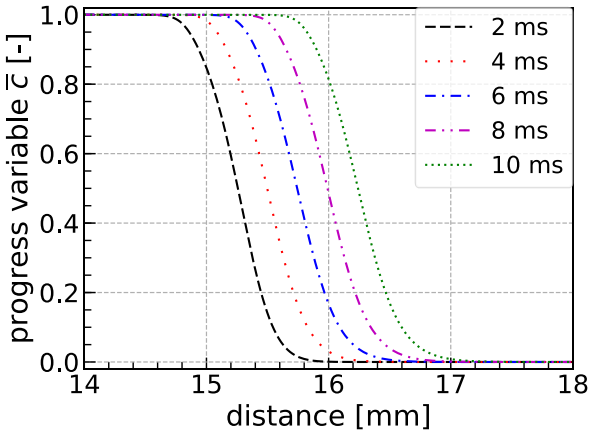
$$\delta_L = \frac{\kappa_u}{S_L} = \frac{\mu_u}{Pr\rho_u S_L}. \quad (6)$$

By using the thermos-physical properties of the unburned and burned mixture in Table 1, the cornflour laminar flame thickness is estimated to be around 1.6e-4 m. In order to resolve the small thickness of the laminar flame, a tube with a mesh size of 2.5e-5 m and a domain of 3e-2×7.5e-5×7.5e-5 m was constructed. For the first and the last 1e-2 m, a grading mesh with mesh size between 2.5e-5 and 2.5e-4 m was used. For the middle part of the domain where laminar flame propagates, a uniform mesh with size of 2.5e-5m was used. There were 600 grid cells in the *x* direction and 3 grid cells in the *y* and *z* directions, respectively, yielding 5400 cells. One numerical run took around 2 h on Simlab computer on one core for simulating flame propagation during 1e-2 s with a time step of 1e-7 s.

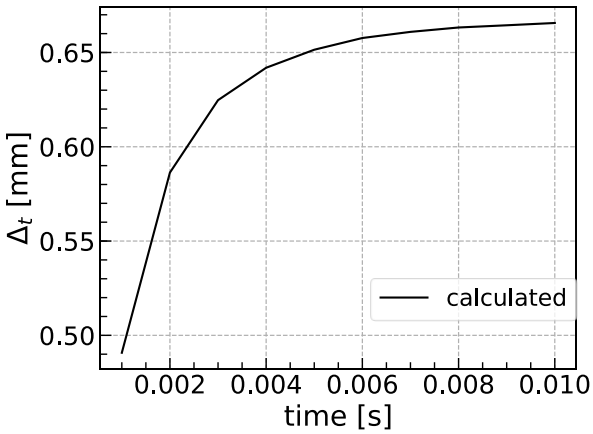
The layout of 1-D planar flame is shown in Figure 2, in which the left-hand side (0-1.5e-2 m) is filled with burned mixture and the right hand side (1.5e-2 – 3e-2 m) is filled with unburned mixture. The boundary conditions for the unburned and burned sides are zero velocity and free entrainment, respectively. The initial and boundary conditions for the simulations are shown in Table 2, with the initial temperature and pressure being equal to 328 K and 110 000 Pa, respectively.

The FSC model constant A and turbulent heat diffusivity D_t were set equal to zero, whereas the reaction time scale was varied to obtain the required laminar flame speed. According to the classical theory by Zeldovich and Frank-Kamenetskii [26], $S_L \propto {t_r}^{-0.5}$. The same scaling was obtained in the simulation, thus, verifying the model implementation. Under conditions of the present simulations, the reaction time scale that yields the laminar flame speed of 0.12 m/s is equal 3.4e-11 s.

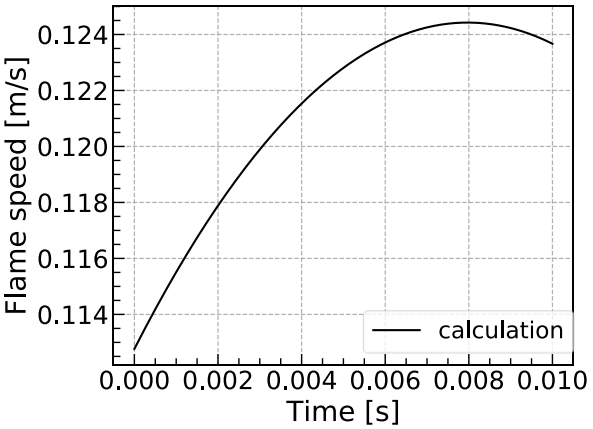
The computed spatial profiles of the progress variable, flame thickness and flame speed are reported in Figure 11. As expected, the flame thickness and speed quickly reach steady value. The computed fully-developed flame thickness is about 0.66 mm, i.e. significantly larger than $\delta_L = 0.16$ mm yielded by Equation (6). The point is that the numerical result was obtained by evaluating the maximum gradient of the progress variable and such a method is well known to yield significantly larger value of a laminar flame thickness when compared to Equation (6). Typically, a ratio of the two thicknesses is comparable with the density ratio.



(a), progress variable vs distance



(b), mean flame brush thickness vs time



(c), flame speed vs time

Figure 11 Computed spatial profiles of the progress variable and evolution of thickness and speed of 1-D planar laminar premixed flame. t_r=3.4e-11 s.

3.1.5. Complete FSC model: 3-D spherical flame in "frozen" turbulence

Reported in this section are results of application of the complete FSC model supplemented with the ignition source term to unsteady 3-D simulations of 3-D spherically symmetrical premixed flames propagating in "frozen" turbulence. The computational domain was a box with a size of 140 mm. Grading mesh was used in order to reduce the number of grid sizes. The grid size in the center of the domain was 0.25 mm and the mesh size grew with the distance from the center, with the largest mesh size being 14.6 mm near the boundary. The use of the grading mesh allowed us to simulate the problem on a mesh of 512 000 grid points. The *blockMeshDict* file for generating mesh is shown in Appendix VIII. In the experiments, which will be discussed in section 3.2, the measured turbulent length scale was 20 mm [29], the measured rms turbulent velocity u' =0.8 m/s. Accordingly, ε =17.4 m^2/s^3 for C_d =0.37. The rest of the initial and boundary conditions were the same as in the 1-D planar case discussed earlier. The ignition model parameters are: W_0=1e6, σ_r=1.5e-3 m, t_0=1e-3 s, $\sigma_t = t_0/5$.

Figure 12 shows that the complete FSC model yields a smaller mean flame brush thickness when compared to the analytical solution to the truncated FSC model in 1-D statistically planar case. This effect is attributed to term 5, which limits the growth of mean flame brush thickness, as discussed in detail elsewhere [22]. The spatial profiles of the Reynolds-averaged combustion progress variable, computed at different instants, are shown in Figure 13. It indicates a complete combustion in the centre of the flame, where the Reynolds-averaged combustion progress variable is almost equal to unity.

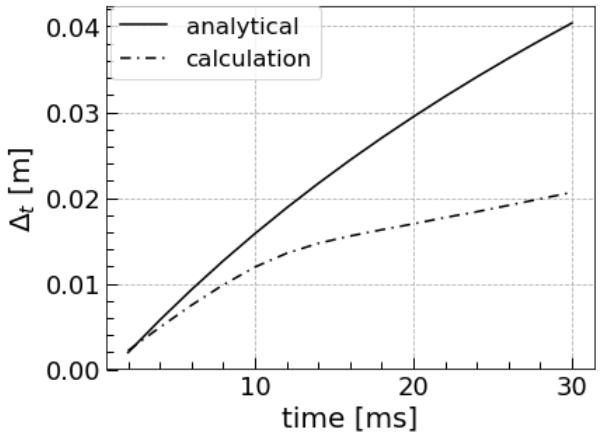


Figure 12 Comparison of calculated mean flame brush thickness with the turbulent diffusion law given by Eq. (Appendix III.4).

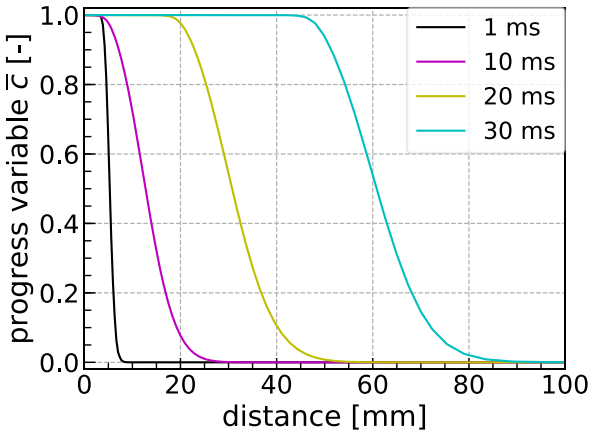


Figure 13 Spatial profiles of the Reynolds-averaged combustion progress variable at different instants obtained from 3-D statistically spherical premixed flame propagating in "frozen" turbulence.

The calculated integral ratio and flame speed are similar to those obtained in the analytical solutions; see Figure 14 and Figure 15, respectively.

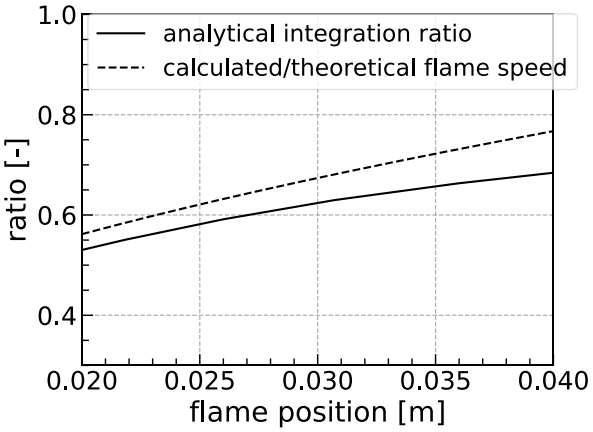


Figure 14 Comparison of analytical integral ratio and the ratio of calculated flame speed with respect to unburned mixture divided by theoretical turbulent flame speed by Equation (Appendix I.3) for 3-D statistically spherical premixed flame propagating in "frozen" turbulence.

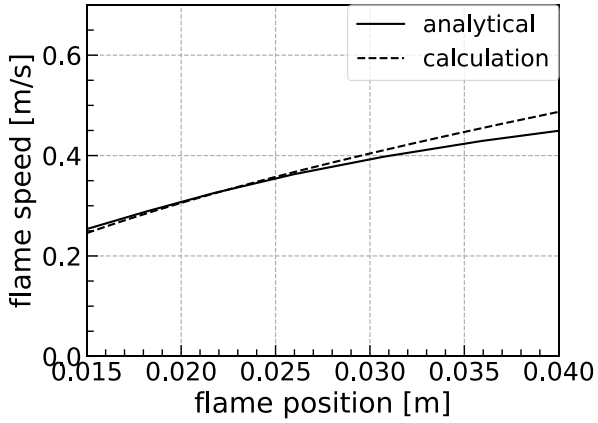


Figure 15 Comparison of analytical and calcualted flame Speed of 3-D statistically spherical premixed flame propagating in "frozen" turbulence.

Figure 16 shows that the size of ignition kernel weakly affects computed flame speed at a later stage of turbulent flame development. Accordingly, the use of experimental data obtained from sufficiently large flame kernels offers an opportunity to test the FSC model independently of the ignition model.

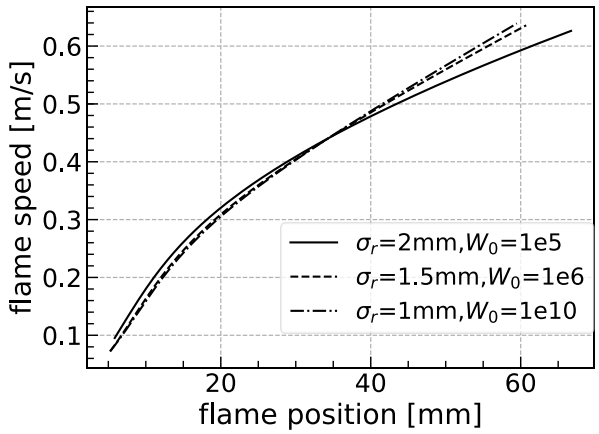


Figure 16 Influence of the size of ignition kernel on the computed speed of 3-D statistically spherical premixed flame propagating in "frozen" turbulence.

3.1.6. Summary

Numerical tests discussed in section 3.1, as well as many other numerical tests skipped for brevity, show that the developed numerical platform well predicts the mean flame structure, the mean flame brush thickness, and the mean flame speed in three benchmark cases: 1-D planar laminar premixed flame, 1-D statistically planar premixed flame that propagates in "frozen" turbulence, and 3-D statistically spherical premixed flame that propagates in "frozen" turbulence. Thus, these tests verify numerical implementation of the FSC model into OpenFOAM performed within the framework of the project.

3.2. Simulations of Leeds experiments

Experimental data on cornflour dust explosion in the well-known Leeds fan-stirred combustion vessel [18] are chosen to begin assessing the model. The experimental setup is illustrated in Figure 17. The vessel diameter is equal to 305 mm and a volume of 0.023 m^3. The vessel has three pairs of orthogonal quartz windows of 150 mm diameter.

Turbulence is generated by four fans, whose rotation speed can be changed to vary the rms turbulent velocity u'. In the discussed experiments, the fan speed was varied from 8 to 50 Hz, which corresponded to variations in u' from 0.80 to 5.00 m/s. While the integral length scale was not reported in Ref. [18], it was reported in other papers by the Leeds group. In particular, Bradley et al. [29] have stated that the longitudinal integral length scale measured using laser Doppler velocimetry was found "*to be 20 mm and independent of fan speed between 1 000 and 10 000 rpm*", with 1 000 rpm corresponding to 16.5 Hz. It is worth noting, however, that, in the experiments with the lean dust-air mixture, the lowest fan

speed was less than 16.5 Hz and a decrease in L at low fan speeds was reported in an earlier paper by the Leeds group [30]. However, those data cannot be used here, because they were obtained using thermo-anemometry, but such a method performs poorly in flows with zero velocity. For instance, the earlier Leeds measurements with thermo-anemometry overestimated L at large fan speeds by a factor of about two. Thus, in the Leeds experiments with the dust-air mixture, the turbulence length scale of 20 mm could be overestimated at low fan speeds. Nevertheless, when compared to other experimental data on dust explosions, the Leeds measurements were performed under well-defined laboratory conditions, i.e. the initial and boundary conditions were well controlled.

To study dust explosion, a premixed dust-air cloud was ignited by a spark in turbulent medium in the centre of the vessel. Subsequently, turbulent flame kernel growth was recorded using high-speed Schlieren system. By processing Schlieren images, an equivalent mean flame radius $\bar{R}_f$, i.e. the radius of a circle whose area was equal to the area enveloped by the flame surface on the image, was calculated and turbulent flame speed with respect to combustion products was evaluated by differentiating the measured $\bar{R}_f(t)$-curves, i.e.

$$S_{t,b} = \frac{d\bar{R}_f}{dt}. \tag{7}$$

To avoid an influence of the spark on the speed, measurements were performed in a range of 20 mm$\leq \bar{R}_f(t) \leq$35 mm. For such flame kernels, whose radius was less than the vessel radius by a factor of about 5, an increase in the pressure in the vessel was negligible.

In addition to values of $S_{t,b}$ obtained at four different $\bar{R}_f$ and five different fan speeds, reported in Ref. [18] are the values of the laminar flame speed S_L and density ratio σ for the studied dust-air mixture. However, methods and precision of evaluation of these values are not discussed. Furthermore, the value of the laminar flame thickness δ_L, which is required to calculate an important input parameter of the FSC mode such as the chemical time scale $\tau_c = \delta_L/S_L$, is not reported. Thus, even in the considered case of the small-scale well-controlled Leeds experiments, some information important for the model validation is missing. This is a typical problem for testing any model of dust explosion.

To save computational time, one eighth of a cube was simulated, with the cube volume being equal to the volume of the vessel. A computational mesh was created with an edge size of 0.14 m. The same was also used in the 3-D simulations of flame expansion in "frozen" turbulence, discussed earlier. The mesh is shown in Figure 18. One simulation took around 6 h on simlab computer for a simulation duration of 10 ms using 2e-6 s timestep.

The initial conditions are reported in Tables 5 and Table 6. It is worth noting that the measured burned temperature of 1500 K [18] is used here instead of the calculated burned temperature of 1592 K, because neither the method, nor precision of the calculation is discussed in Ref. [18]. The use of the former temperature yields the density ratio of 5.06, whereas σ =5.49 reported in Ref. [18] corresponds to the latter (higher) temperature.

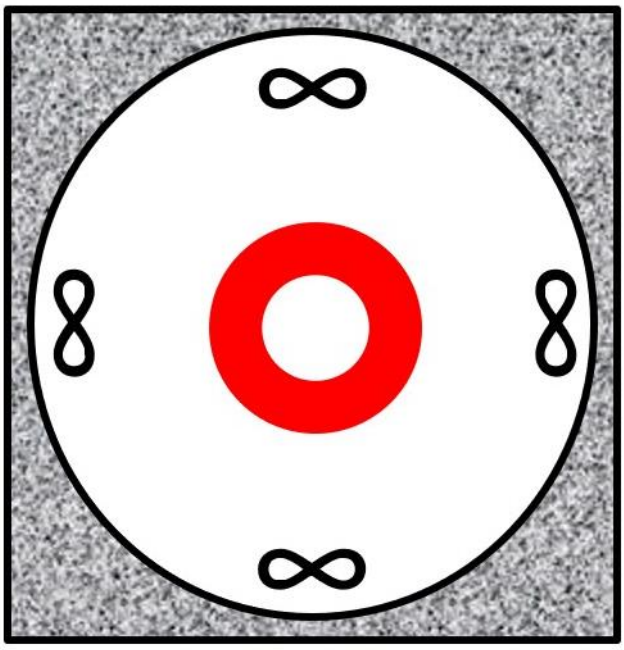


Figure 17 Illustration of Leeds fan stirred bomb.

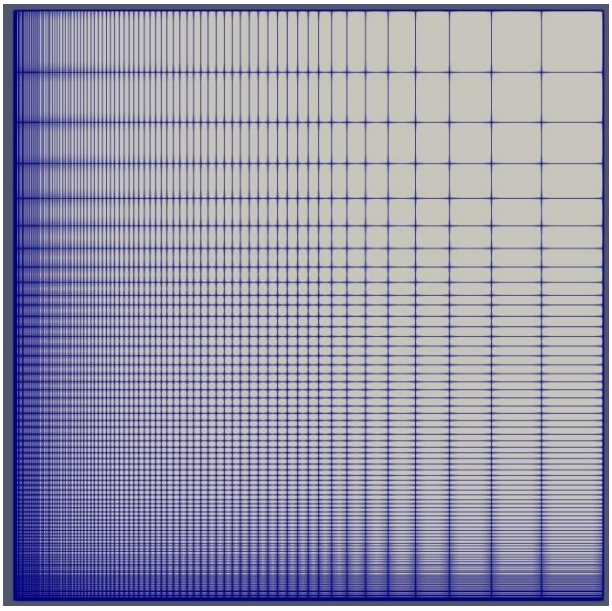


Figure 18 Illustration of computational mesh.

Table 5 Thermo-physical properties of unburned and burned mixture of equivalence ratio 0.77.

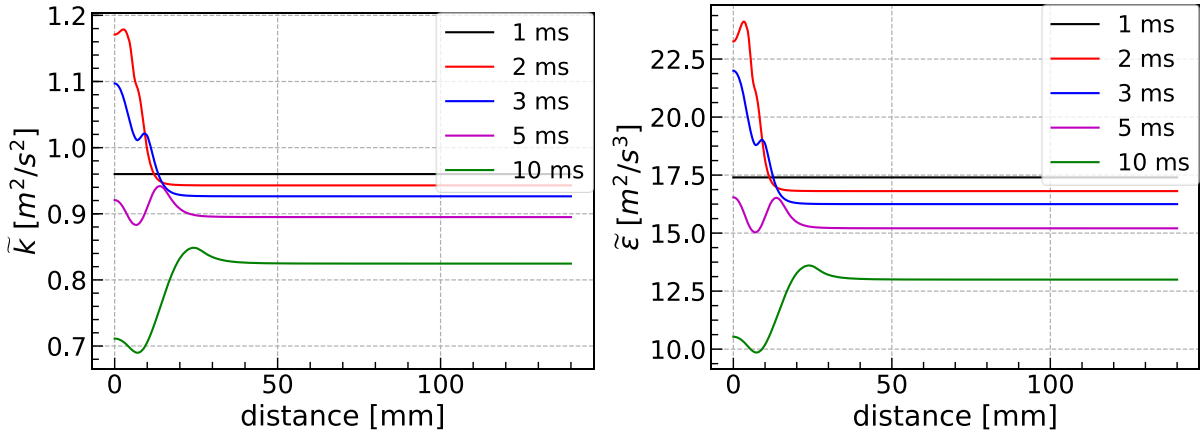
Parameters		value
unburned	T_u [K]	328
	W_u [g/mol]	32.80
	ρ_u [kg/m^3]	1.3214
	μ_u [kg/(m·s)]	1.8e-5
burned	T_b [K]	1500
	W_b [g/mol]	29.66
	ρ_b [kg/m^3]	0.2480
	μ_b [kg/(m·s)]	4.6e-5
others	$\sigma = \rho_u/\rho_b = (T_b W_u)/(T_u W_b)$ [-]	5.06
	S_L	0.12

Table 6 Initial condition for cornflour explosion in Leeds fan stirred vessel.

parameters	value
T_0 [K]	328
P_0 [Pa]	11 000

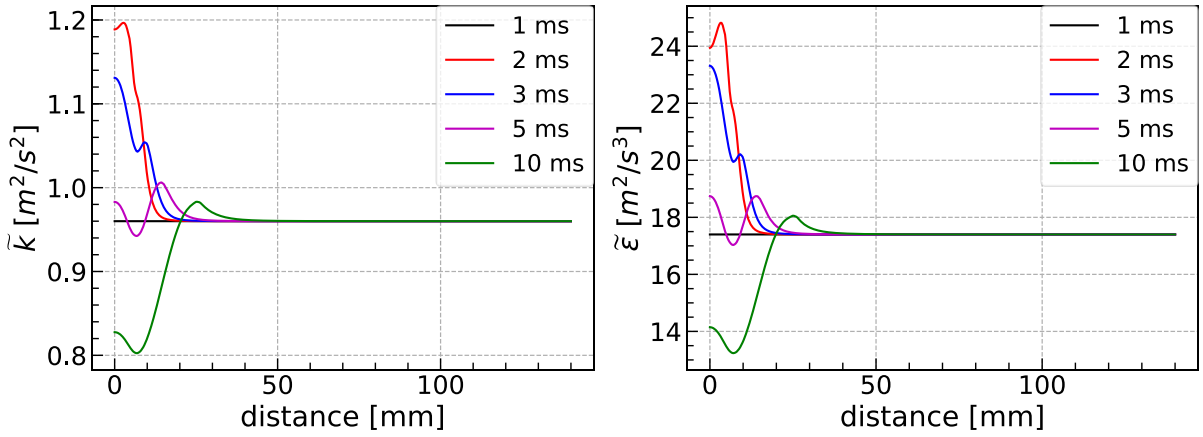
3.2.1. Extra source terms in standard $k-\varepsilon$ turbulence model

In Leeds fan-stirred bomb experiment, a statistically stationary, spatially uniform, and isotropic turbulence is generated in the central zone of the vessel by four rotating fans. The rms velocity can be changed by varying the fan speed. If the fans are not included in simulations, which address solely the central region of the vessel, i.e. the region where the measurements were performed, the standard $k-\varepsilon$ turbulence model yields decaying turbulence; see Figure 19. To mimic turbulence generation by the fans and to simulate statistically stationary turbulence, an extra source term $\bar{\rho}\tilde{\varepsilon}_0$ was added to the transport equations for $\tilde{k}$ and $\tilde{\varepsilon}$ equations following Lipatnikov and Chomiak [31]. Detailed implementation of this source term is described in Appendix IX. Figure 20 shows that turbulence characteristics computed using the modified $k-\varepsilon$ model are statistically steady, in line with the measurements.



(a), turbulent kinetic energy *(b), turbulent dissipation rate*

Figure 19 Turbulent kinetic energy and dissipation rate computed using the standard $k-\varepsilon$ turbulence model at different instants. The initial $\tilde{k}$=0.96 m^2/s^2, $\tilde{\varepsilon}$=17.4 m^2/s^3, C_d=0.37, L_t=0.02 m, Turbulence model is activated at 1 ms.



(a), turbulent kinetic energy *(b), turbulent dissipation rate*

Figure 20 Turbulent kinetic energy and dissipation rate computed using the modified $k-\varepsilon$ turbulence model at different instants. Other details are provided in caption to Figure 19.

3.2.2. Sensitivity study

Before performing computations of the Leeds experiments, sensitivity of numerical results to input parameters that are not well known should be investigated. There are three groups of such parameters.

First, the FSC model involves a single constant, i.e. A required to evaluate turbulent burning velocity, see Equation (Appendix I.5). For various gaseous flames, the use of the same value of $A = 0.4$ yielded good results [23]. However, the value of A for dust explosions could be different. Moreover, as already noted above, the value of the laminar flame thickness δ_L is not known for the dust-air mixture investigated in Leeds. However, because both A and δ_L are included in the same model equation, i.e. $U_T = Au'(\tau_t S_L/\delta_L)^{1/4}$, the lack of knowledge on δ_L could be compensated by tuning A. In particular, since dust particles should volatilize before burning, the laminar flame thickness of a dust-air mixture could be larger than the thickness of a gaseous laminar premixed flame characterized by the same S_L. In such a case, the use of the latter thickness could be compensated by adopting a lower $A < 0.4$.

Second, the $k-\varepsilon$ turbulence model involves a set of constants. For some of them, almost the same values are commonly adopted, but values of other constants depend substantially on conditions and, hence, are tunable. The latter group of constants includes turbulent Prandtl number Pr_t required to calculate fully-developed turbulent diffusivity, see Equation (Appendix I.4)), and a constant C_d required to link the mean dissipation rate $\tilde{\varepsilon}$ and integral or turbulent length scale L (i.e. $\tilde{\varepsilon} = C_d\, u'^3/L$), e.g. when setting the initial conditions or evaluating $\tau_t = L/u'$ based on the computed $\tilde{k}$ and $\tilde{\varepsilon}$.

Third, to study dust explosion, the dust-air mixture should be ignited, but ignition of a flammable mixture in a turbulent flow is a very complicated phenomenon, which is beyond the scope of the present project. Accordingly, to mimic ignition an extra source (sink) term W_{ign} is inserted into the transport

equation for progress (regress) variable. The term serves solely to rapidly create a small spherical kernel filled with combustion products. Accordingly, the model involves four input parameters whose values could be varied. These are the source magnitude W_0, the kernel size σ_r, the ignition time t_0, and duration σ_t. Such a simplified method may be used, because (i) the Leeds experimental data were obtained from sufficiently large flame kernels [18] and (ii) the computed speeds of expansion of such large kernels are weakly affected by parameters of the ignition model, as already discussed earlier in the case of "frozen" turbulence. However, if turbulence evolution is simulated adopting the $k-\varepsilon$ model, then, short, highly localized, and very strong heat release associated with the term W_{ign} can strongly affect the computed fields of $\tilde{k}$ and $\tilde{\varepsilon}$. Such a numerical effect, in fact, significantly changes the initial conditions. However, this effect is a numerical artifact, because neither $k-\varepsilon$, nor another model can describe influence of strongly localized heat release on turbulence, as reviewed elsewhere[32, 33]. To circumvent the problem and avoid the discussed unphysical effects, the $k-\varepsilon$ model should be activated after ignition [31], i.e. when term W_{ign} becomes small (it decays rapidly with time at $t > t_0$). Accordingly, there is one more important tuning parameter, i.e. time of activation of the $k-\varepsilon$ model.

To summarize the above discussion, model constants and input parameters that were not varied in the present study are reported in Table 7, whereas model constants and input parameters that were varied in the sensitivity study are shown in Table 8.

Table 7 Model constants and input parameters that were not varied in the present study.

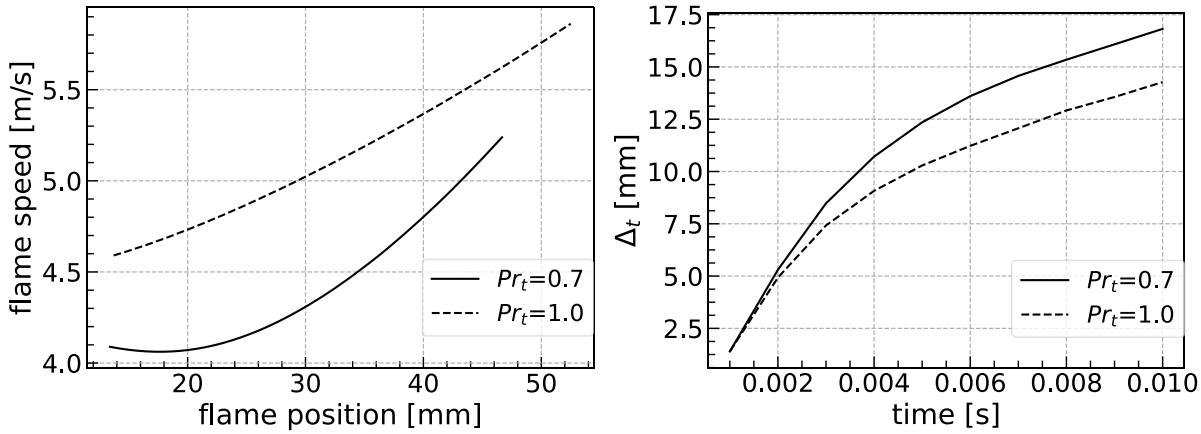
	Parameter	Value
Turbulence model	C_μ [-]	0.09
	C_1 [-]	1.44
	C_2 [-]	1.92
	σ_k [-]	1.0
	σ_ε [-]	1.3
	C_d [-]	0.37-2.0
Combustion model	t_r [s]	3.4e-11
	Θ [K]	2e4

Table 8 Model constants and input parameters that were varied in the present study.

	Parameter	Value range	note
Ignition model	W_0 [-]	-	case dependent
	t_0 [s]	around 1 ms	
	σ_t [s]	$\sigma_t = t_0/5$	depends on t_0
	σ_r [m]	around 1 mm	related to ignition kernel
Turbulence model	Pr_t [-]	0.3-1.0	
	activation timing of turbulence model	1-10 ms	
Combustion model	A [-]	0.2-0.5	0.4 for gas burning

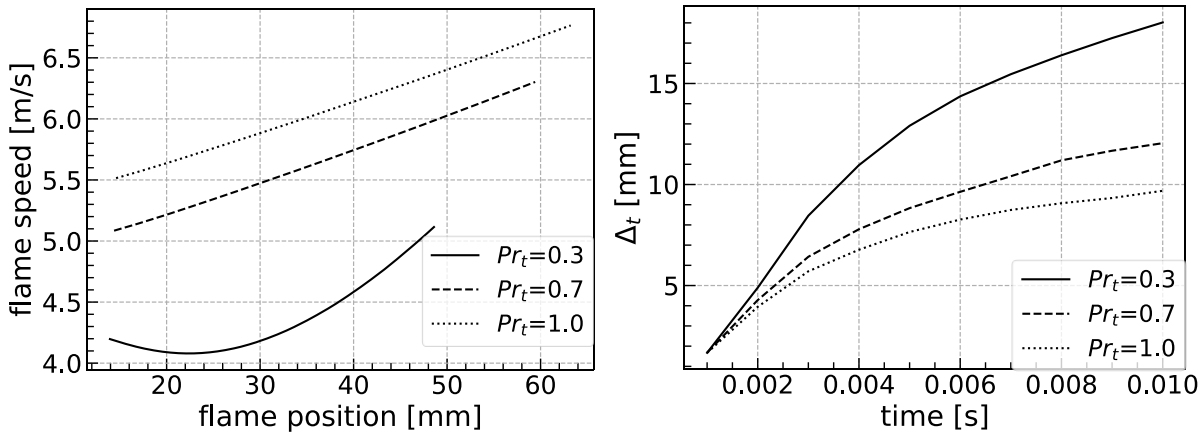
3.2.2.1. Turbulent Prandtl number

Effect of turbulent Prandtl number on the computed flame speed and flame thickness is shown in Figure 21 and Figure 22. The two figures report results computed with slightly different model constants, because higher values of C_d, $\tilde{\varepsilon}$ and σ_r, see Figure 22, were required to obtain complete combustion in the centre at a low Pr_t. Accordingly, results obtained using $Pr_t = 0.3$ are not shown in Figure 21, because the initial kernel was eroded due to turbulent diffusion for those values of C_d, $\tilde{\varepsilon}$ and σ_r (note that turbulence model was activated during ignition that case and, due to thermal expansion effects, the computed turbulent diffusivity was strongly increased) . Nevertheless, trends observed in Figure 21 and Figure 22 are similar. A decrease in Pr_t results in increasing mean turbulent flame speed and decreasing mean flame brush thickness.



(a), flame speed *(b), flame thickness*

Figure 21 Mean flame speed vs. mean flame position and mean flame brush thickness vs. time, obtained for different turbulent Prandtl numbers from 3-D statistically spherical flames with the initial $\tilde{k}$=9.077 m^2/s^2, the initial $\tilde{\varepsilon}$=1367.36 m^2/s^3, C_d=1.0, L_t=0.02 m, σ_r= 1.5e-3 m, W_0=1e14, t_0=1e-3 s, A=0.4. Turbulence model was activated from the beginning.

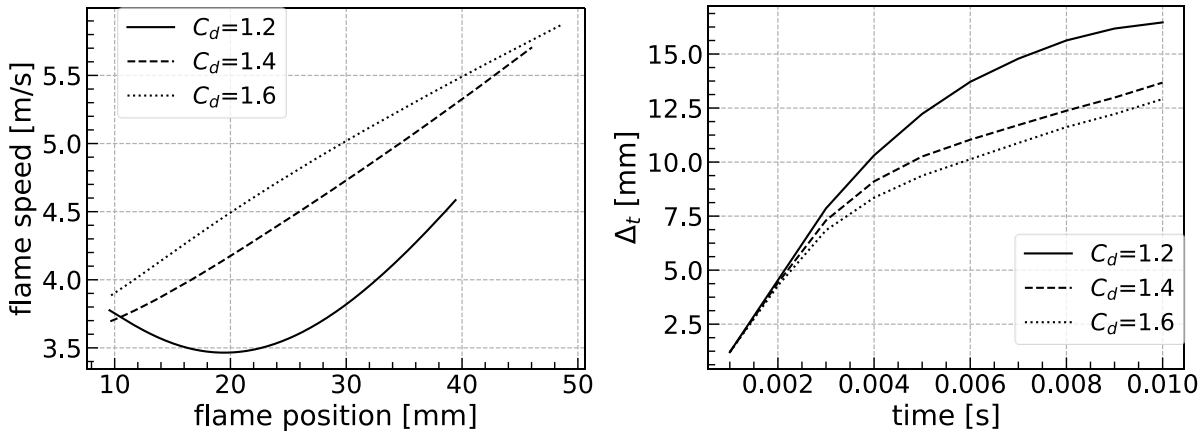


(a), flame speed *(b), flame thickness*

Figure 22 Mean flame speed vs. mean flame position and mean flame brush thickness vs. time, obtained for different turbulent Prandtl numbers from 3-D statistically spherical flames with the initial $\tilde{k}$=9.077 m^2/s^2, the initial $\tilde{\varepsilon}$=2734.72 m^2/s^3, C_d=2.0, L_t=0.02 m, σ_r= 2e-3 m, W_0=1e15, t_0=1e-3 s, A=0.4. Turbulence model was activated at 2 ms.

3.2.2.2. C_d coefficient

Effects of C_d coefficient on the computed results are shown in Figure 23. An increase in C_d results in increasing the flame speed and decreasing the flame thickness. Thus, a decrease in Pr_t and an increase in C_d act in the same directions.

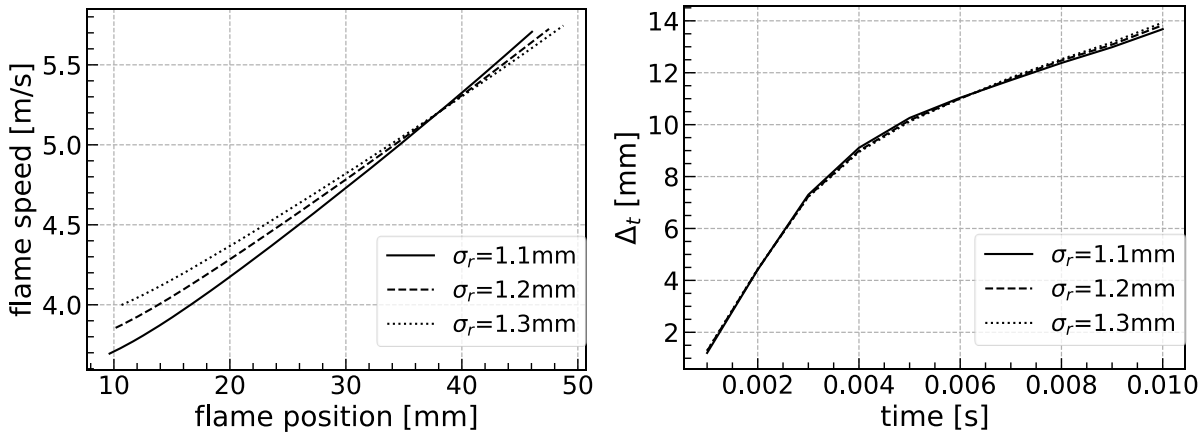


(a), flame speed *(b), flame thickness*

Figure 23 Mean flame speed vs. mean flame position and mean flame brush thickness vs. time, obtained for different constants C_d from 3-D statistically spherical flames with the initial $\tilde{k}$=9.077 m^2/s^2, L_t=0.02 m, σ_r= 1.1e-3 m, W_0=1e15, t_0=1e-3 s, A =0.4, Pr_t =0.7. Turbulence model was activated at 2 ms.

3.2.2.3. Ignition kernel size σ_r

Effects of the ignition kernel size σ_r on the computed flame speed and flame thickness are shown in Figure 24. This parameter weakly affects the computed results provided that combustion in the center is complete, i.e. mean value of the progress variable is close to unity in the center after a transition time interval. However, if σ_r is small, the kernel is eroded due to turbulent diffusivity and $\tilde{c}(r=0,t)$ decreases with time. In such a case, the computed flame speed is low and the kernel shrinks. Thus, Figure 24 indicates that the value of σ_r weakly affects the mean speed of a sufficiently large flame kernel provided the value of σ_r is sufficient to get the complete combustion in the center. This finding is in line with the results plotted in Figure 16.



(a), flame speed *(b), flame thickness*

Figure 24 Mean flame speed vs. mean flame position and mean flame brush thickness vs. time, obtained for different values of σ_r from 3-D statistically spherical flames with the initial $\tilde{k}$=9.077 m^2/s^2, L_t=0.02 m, σ_r= 1.1-1.3 mm, C_d=1.4, W_0=1e15, t_0=1e-3 s, A =0.4, Pr_t =0.7. Turbulence model was activated at 2 ms.

3.2.2.4. Timing for activating turbulence model

Effects of timing for activating turbulence model are shown in Figure 25. The result indicates that earlier activation of the turbulence model yields a higher flame speed. The reason is that a higher turbulent kinetic energy is calculated when activating turbulence model early; see Figure 26. However, this is so if combustion in the center is complete. Otherwise, earlier activation of the turbulence model can result in shrinking the kernel due to too intense turbulent diffusivity yielded by the $k-\varepsilon$ model.

It is also worth noting that the Leeds experiments did not reveal generation of turbulence after ignition. For instance Bradley et al.[29] have stated that "*laser anemometry ahead of the flame showed … no evidence of any significant change in* u'". Solid curve in Figure 26 agrees qualitatively with the cited observation, thus, justifying activation of the $k-\varepsilon$ model after the end of ignition.

The timing for activating turbulence model has a minor effect on the computed flame thickness provided that combustion is complete in the centre. Accordingly, the figure is not reported here.

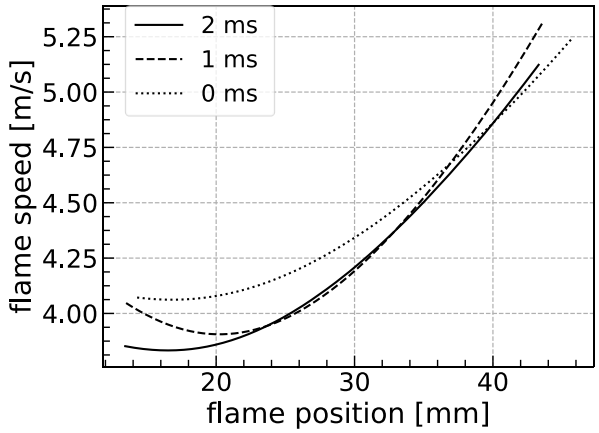


Figure 25 Computed flame speed vs flame position for different timing for activating turbulence model for 3-D spherical flame with initial $\tilde{k}$*=9.077 m²/s², L*$_t$*=0.02 m,* σ_r*= 1.5e-3 m, W*$_0$*=1e14, t*$_0$*=1e-3 s, A =0.4,* Pr_t *=0.7. Turbulence model is activated at 0, 1 and 2 ms.*

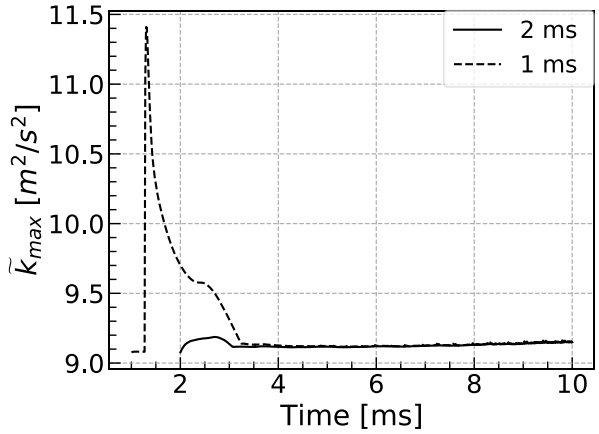


Figure 26 Computed maximum turbulent kinetic energy $\tilde{k}$ *vs. time for different timing for activating turbulence model. The initial* $\tilde{k}$*=9.077 m²/s², L*$_t$*=0.02 m,* σ_r*= 1.5e-3 m, W*$_0$*=1e14, t*$_0$*=1e-3 s, A =0.4,* Pr_t *=0.7. Turbulence model is activated at 1 and 2 ms.*

3.3. First test of model

A summary of the initial turbulence characteristics for the model input is shown in Table 9. The rest of the initial and boundary conditions for simulation are reported in Figure 18, Table 5 and Table 6.

Table 9 Initial turbulence characteristics using C_d*=1.2 and* L_t*=0.02 m.*

u' [m/s]	0.80	1.62	2.45	3.31
$\tilde{k}$ [m²/s²]	0.96	3.94	9.004	16.43
$\tilde{\varepsilon}$ [m²/s³]	56.44	468.63	1621.01	3997.35

Results of a "first-shot" test of the model are shown in black open symbols in Figure 27, with the

measured data being plotted in black filled symbols. The experiments and simulations show the same qualitative trends, i.e. (i) a quasi-linear increase in S_t by u' and (ii) an increase in S_t with the flame kernel radius. Contrary to trend (ii), the experiments performed at the largest $u' = 5.0$ m/s indicate a decrease in the flame speed with the flame radius due to quenching of a large flame kernel by turbulence. Such a phenomenon is well known from experiments with gaseous flames and there is no model capable for quantitatively predicting this quenching effect. However, this phenomenon is beyond the scope of the present project that addresses dust explosions, rather than quenching by intense turbulence. For lower values of u', trend (ii) is observed both in the simulations and measurements.

However, from the quantitative perspective, the model yields overestimated flame speeds. This overestimation could result not only from eventual model limitations, but also from (i) inappropriate set of values of C_d and Pr_t, (ii) an overestimated value of the laminar flame thickness δ_L, or (iii) differences in the values of A for gaseous and dust-air mixtures. For instance, the red filled circles in Figure 27 indicate that S_t computed using $A = 0.35$ is close to the experimental data. Note that, since $U_t = Au'(\tau_t S_L/\delta_L)^{1/4}$, a decrease in A from 0.40 to 0.35 corresponds to an increase in the thickness δ_L by a factor of 1.7 only. As already discussed, a larger value of the thickness δ_L are expected for dust-air mixtures when compared to gaseous fuel-air mixtures, because dust particles should volatilize before burning.

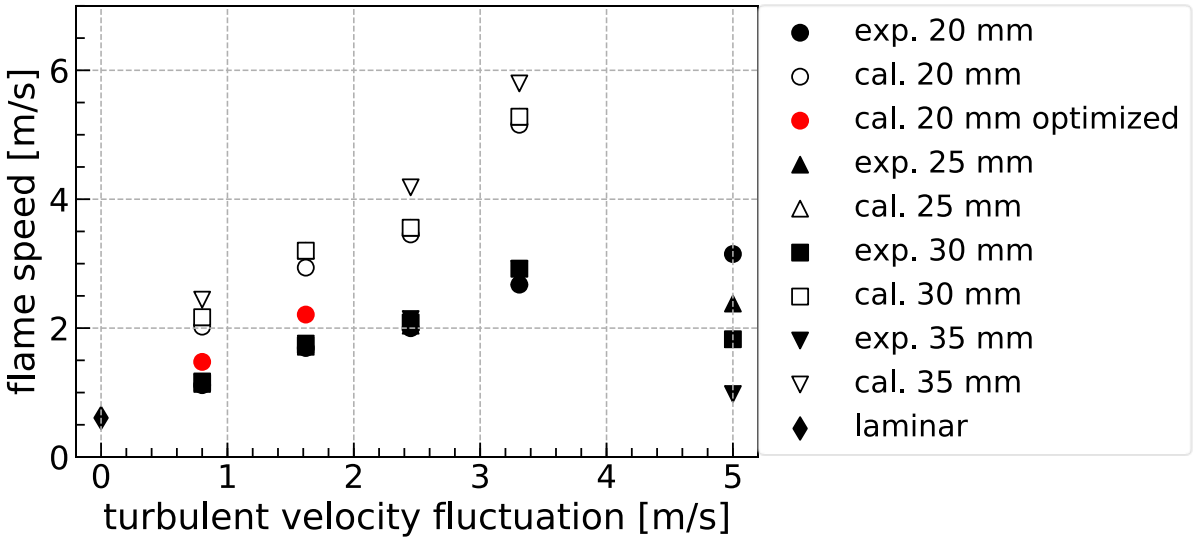


Figure 27 Comparison of flame speed between simulations (dashed lines) and Leeds experiments (line with symbols) for 3-D spherical flame with different initial turbulent velocity fluctuations, C_d=1.2, L_t=0.02 m, σ_r= 1.1e-3 m, W_0=1e15, t_0=1e-3 s, A =0.4, Pr_t =0.7. Turbulence model is activated at 2 ms. Note for u'=3.31 m/s, σ_r= 1.5e-3 m is used to achieve a complete ignition. The diamond symbol represents the laminar flame speed multiplied with density ratio; the read filled circles represents calculated flame speed using A =0.35, $Pr_t = 0.5$. Note for u'=1.62 m/s, σ_r= 1.5e-3 m is used to achieve a complete ignition.

All in all, the results of the first-shot test appear to be encouraging. In the nearest next stage of the project, an optimum set $\{C_d, Pr_t, A\}$ of the numerical model constants will be sought to achieve a better agreement between the simulations and experiments. Subsequently, the model will be applied to other large-scale, but less-controlled experiments and will eventually be further developed, e.g. by taking into account heat transfer due to radiation.

4. Publications, presentations and other spreading within framework of project

The project results are distributed to the reference group members via three reference group meetings on 190509, 191107 and 200116. The reference group members are a balance between academy and industry with members of PS Group, Scandbio, AVS, Hoerbiger, Göteborgs Energi, Fagerberg, Chalmers, Dust-Ex Research (Canada), Ulster University (UK) and RISE.

The project results are planned to be presented at Digital Dust Safety Conference in February 24 - 28, 2020, and 12th FM Global Open Source CFD Fire Modeling Workshop on 2-3, April, 2020.

References

1. Peter Wikström, A., *Personlig kommunikation* H. Persson, Editor.
2. *Brand på Rya HVC*. 2017 2018-04-19]; Available from: https://www.goteborgenergi.se/Om_oss/Pressrum/Nyheter?DisplayNewsItem=36992077.
3. *Dammexplosion i Huskvarna*. 2017 2018-04-19]; Available from: https://www.jnytt.se/article/dammexplosion-i-huskvarna/.
4. *Flera skadade efter explosion vid en arbetsplats i Trångsund*. 2018-04-19]; Available from: https://www.svt.se/nyheter/lokalt/stockholm/flera-skadade-efter-explosion-vid-en-arbetsplats-i-trangsund.
5. *Explosion i spannmålstork*. 2018-04-18]; Available from: https://www.skovdenyheter.se/article/larm-om-ladugardsbrand/.
6. Försäkring, A., *Allvarliga arbetsskador och långvarig sjukfrånvaro–2018.* Stockholm: AFA Försäkring, 2018.
7. Nessvi, K. and H. Persson, *Dammexplosionsrisker i metallbearbetande industri*. 2019.
8. *Två till sjukhus efter brand på kraftvärmeverket*. 2018; Available from: https://www.kkuriren.se/nyheter/katrineholm/tva-till-sjukhus-efter-brand-pa-kraftvarmeverket-sm4522960.aspx.
9. *Brand i pelletsfabrik i Laxå – tredje på en vecka.* 2018.
10. *Ny brand i pelletsfabrik*. 2018; Available from: https://www.aftonbladet.se/lokala-nyheter/G1LkVV@ablokal.
11. *Brand i pelletsfabrik under onsdagen: ”Flera explosioner”*. 2018-10-03; Available from: https://www.svt.se/nyheter/lokalt/orebro/brand-i-pelletsfabrik-risk-for-explosion.
12. *Brand i pelletsfabrik i Älvdalen i natt – stort pådrag från räddningstjänsten*. 2018; Available from: https://www.svt.se/nyheter/lokalt/dalarna/brand-i-pelletsfabrik-i-alvdalen-ledde-till-stort-raddningspadrag.
13. *Dammexplosion i fabrik under kontroll: "Det var en ganska kraftig explosion, den har skjutit isär rör"*. 2018; Available from: https://www.dt.se/artikel/dammexplosion-i-fabrik-under-kontroll-det-var-en-ganska-kraftig-explosion-den-har-skjutit-isar-ror.
14. *Explosion på industri i Umeå*. 2018; Available from: https://www.vk.se/2408868/explosion-pa-industri-i-umea.
15. *Eftersläckning pågår vid Alholmens Kraft – oklart hur stora skador*. 2018; Available from: https://www.vasabladet.fi/Artikel/Visa/237349.
16. *Larm om brand i fabrik*. 2018; Available from: https://www.aftonbladet.se/nyheter/a/p625L1/larm-om-brand-i-fabrik.
17. Lipatnikov, A.N., *Testing premixed turbulent combustion models by studying flame dynamics.* International Journal of Spray and Combustion Dynamics, 2009. **1**(1): p. 39-66.
18. Bradley, D., Z. Chen, and J. Swithenbank. *Burning rates in turbulent fine dust-air explosions*. in *Symposium (International) on Combustion*. 1989. Elsevier.
19. Bradley, D., S. El-Din Habik, and J.R. Swithenbank, *Laminar burning velocities of CH4—air-graphite mixtures and coal dusts.* Symposium (International) on Combustion, 1988. **21**(1): p. 249-256.
20. Bradley, D., G. Dixon-Lewis, and S. El-Din Habik, *Lean flammability limits and laminar burning velocities of CH4-air-graphite mixtures and fine coal dusts.* Combustion and Flame, 1989. **77**(1): p. 41-50.
21. Spijker, C., et al., *MODELLING DUST EXPLOSIONS.*
22. Lipatnikov, A. and J. Chomiak, *Turbulent flame speed and thickness: phenomenology, evaluation, and application in multi-dimensional simulations.* Progress in energy and combustion science, 2002. **28**(1): p. 1-74.
23. Yasari, E., *RANS Simulations of Interaction between Premixed Flame and Turbulence using OpenFOAM Library*. 2015: Chalmers University of Technology.
24. Bray, K., P.A. Libby, and J. Moss, *Unified modeling approach for premixed turbulent combustion—Part I: General formulation.* Combustion and flame, 1985. **61**(1): p. 87-102.
25. Huang, C., *Numerical Modelling of Fuel Injection and Stratified Turbulent Combustion in a Direct-Injection Spark-Ignition Engine Using an Open Source Code*. 2014: Chalmers University of Technology.

26. Zel'dovich, Y.B.a.F.-K., *On the theory of steady flame propagation.* Dokl. AN SSSR, 1938. **19**(9): p. 693-695.
27. Lipatnikov, A. and J. Chomiak, *Global stretch effects in premixed turbulent combustion.* Proceedings of the Combustion Institute, 2007. **31**(1): p. 1361-1368.
28. Lipatnikov, A., *Fundamentals of premixed turbulent combustion*. 2012: CRC Press.
29. Bradley, D., et al., *Turbulent burning velocity, burned gas distribution, and associated flame surface definition.* Combustion and Flame, 2003. **133**(4): p. 415-430.
30. Abdel-Gayed, R., K. Al-Khishali, and D. Bradley, *Turbulent burning velocities and flame straining in explosions.* Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 1984. **391**(1801): p. 393-414.
31. Chomiak, A.L.a.J., *Modeling of Turbulent Flame Propagation*. 1997.
32. Lipatnikov, A.N. and J. Chomiak, *Effects of premixed flames on turbulence and turbulent scalar transport.* Progress in Energy and Combustion Science, 2010. **36**(1): p. 1-102.
33. Sabelnikov, V.A. and A.N. Lipatnikov, *Recent Advances in Understanding of Thermal Expansion Effects in Premixed Turbulent Flames.* Annual Review of Fluid Mechanics, 2017. **49**(1): p. 91-117.
34. Zimont, V., *Theory of turbulent combustion of a homogeneous fuel mixture at high Reynolds numbers.* Combustion, Explosion and Shock Waves, 1979. **15**(3): p. 305-311.
35. Zimont, V. and A. Lipatnikov, *A numerical model of premixed turbulent combustion of gases.* Chem. Phys. Rep, 1995. **14**(7): p. 993-1025.
36. Dynamics, W. *Finite Volume Method: A Crash introduction*. 2019-10-23]; Available from: http://www.wolfdynamics.com/wiki/fvm_crash_intro.pdf.
37. dynamics, W. *Running in parallel*. Available from: http://www.wolfdynamics.com/wiki/parallel.pdf.

Appendix I. Flame Speed Closure (FSC) and ignition models

The FSC model characterizes the state of the mixture in a flame using a single combustion progress variable c and deals with the following balance equation

$$\frac{\partial \bar{\rho}\tilde{c}}{\partial t} + \nabla.(\bar{\rho}\tilde{\mathbf{u}}\tilde{c}) = \nabla.[\bar{\rho}(\kappa + D_t)\nabla\tilde{c}] + \rho_u U_t|\nabla\tilde{c}| + \frac{\bar{\rho}(1-\tilde{c})}{t_r(1 + D_t/\kappa_b)}\exp\left(-\frac{\Theta}{\tilde{T}}\right), \qquad \text{(Appendix I.1)}$$

where t is the time, $\mathbf{u}$ is the flow velocity vector, ρ is the density, κ is the molecular heat diffusivity, D_t and U_t are the time-dependent turbulent heat diffusivity and burning velocity, respectively. Subscripts u and b refer to the unburned and burned mixture, respectively. Overlines denote the Reynolds average, while $\tilde{q} = \overline{\rho q}/\bar{\rho}$ is the Favre-averaged value of q.

The FSC model allows directly for the transient nature of a turbulent burning. For this purpose, the development of turbulent diffusivity and burning velocity is modelled as follows

$$D_t = D_{t,\infty}\left[1 - \exp\left(-\frac{t_{fd}}{\tau_L}\right)\right], \qquad \text{(Appendix I.2)}$$

$$U_t = U_{t,\infty}\left[1 - \frac{\tau_L}{t_{fd}} + \frac{\tau_L}{t_{fd}}\exp\left(-\frac{t_{fd}}{\tau_L}\right)\right]^{1/2}, \qquad \text{(Appendix I.3)}$$

where t_{fd} is the flame development time, $\tau_L = D_{t,\infty}/u'^2$ is the Lagrangian time scale of turbulence, and $D_{t,\infty}$ and $U_{t,\infty}$ are the fully developed turbulent diffusivity and burning velocity, respectively.

To evaluate the fully developed turbulent diffusivity $D_{t,\infty}$, a turbulence model should be invoked, e.g.

$$D_{t,\infty} = \frac{C_\mu}{Pr_t}\frac{\tilde{k}^2}{\tilde{\varepsilon}} \qquad \text{(Appendix I.4)}$$

within the framework of the standard $k-\varepsilon$ model. Here, k and ε are the turbulent kinetic energy and its dissipation rate, respectively; $C_\mu = 0.09$ is a constant, and Pr_t is the turbulent Prandtl number (a ratio of the turbulent transport of momentum to that of heat).

The fully developed turbulent burning velocity is given by the following theoretical expression [34]

$$U_{t,\infty} = Au'Da^{1/4} = Au'^{3/4}L^{1/4}S_L^{1/4}\delta_L^{-1/4} \qquad \text{(Appendix I.5)}$$

where A is the sole constant of the FSC model; $Da = \tau_t/\tau_c$ is the Damköhler number; $u' = 2\tilde{k}^{1/2}/3$, $L = C_d\,\tilde{k}^{3/2}/\tilde{\varepsilon}$, and $\tau_t = L/u'$ are the rms turbulent velocity, integral length scale, and eddy-turn-over time, respectively; $\tau_c = \delta_L/S_L$, $\delta_L = \kappa_u/S_L$, and S_L are the chemical time scale, laminar flame thickness and laminar flame speed, respectively. The constant C_d pertains to the turbulence model and depend on turbulent Reynolds number.

The last source term on the right hand side of Equation (Appendix I.6) is used in order for the model to describe the laminar flame at $u' \to 0$. In this source term $\Theta = 20\,000$ K is the activation temperature for a single reaction to which the combustion chemistry is reduced and the reaction time scale t_r is set so that the burning velocity computed at $u' = 0, D_t = 0$, and $U_t = 0$ is equal to the laminar flame speed S_L, which is an input parameter of the model. It is worth noting that not only the complete version of the FSC model, but also its truncated version, i.e. Equation (Appendix I.1) without the last source term, were used within the framework of the present project, with the truncated version being adopted to verify the model implementation by comparing numerical and analytical results.

To simulate ignition of the mixture, an extra source term is added on the right-hand side of Equation (Appendix I.1), i.e.

$$\frac{\partial \bar{\rho}\tilde{c}}{\partial t} + \nabla.(\bar{\rho}\tilde{\mathbf{u}}\tilde{c}) = \nabla.[\bar{\rho}(\kappa + D_t)\nabla\tilde{c}] + \rho_u U_t|\nabla\tilde{c}| + \frac{\bar{\rho}(1-\tilde{c})}{t_r(1 + D_t/\kappa_b)}\exp\left(-\frac{\Theta}{\tilde{T}}\right) + \bar{\rho}W_{ign}, \qquad \text{(Appendix I.6)}$$

where, following Ref. [35],

$$W_{ign} = W_0 \exp\left\{-\left[\left(\frac{r}{\sigma_r}\right)^2 + \left(\frac{t-t_0}{\sigma_t}\right)^2\right]\right\}(1-\tilde{c}). \qquad \text{(Appendix I.7)}$$

Here, W_0, σ_r, σ_t are parameters for ignition model. More specifically, t_0 is associated with ignition time, σ_t characterizes ignition duration, and σ_r corresponds to the size of ignition kernel. The factor W_0 is associated with the ignition strength and should be set sufficiently large in order for $\tilde{c}(r=0,t_0)$ to be close to unity.

Appendix II. Transport equations for progress and regress variables.

The FSC transport equation for the Favre-averaged progress variable $\tilde{c}$ with extra ignition term is as follows

$$\frac{\partial \bar{\rho}\tilde{c}}{\partial t} + \nabla.(\bar{\rho}\tilde{\mathbf{u}}\tilde{c}) = \nabla.[\bar{\rho}(\kappa + D_t)\nabla\tilde{c}] + \rho_u U_t|\nabla\tilde{c}| + Q + \bar{\rho}W_{ign} \quad \text{(Appendix II.1)}$$

where

$$Q = \frac{\bar{\rho}(1-\tilde{c})}{t_r(1 + D_t/\kappa_b)} \exp\left(-\frac{\Theta}{\tilde{T}}\right) \quad \text{(Appendix II.2)}$$

and

$$W_{ign} = W_0 \exp\left\{-\left[\left(\frac{r}{\sigma_r}\right)^2 + \left(\frac{t-t_0}{\sigma_t}\right)^2\right]\right\}(1-\tilde{c}). \quad \text{(Appendix II.3)}$$

By substituting $\tilde{c} = 1 - \tilde{b}$ into the $\tilde{c}$ equation, we get

$$\underbrace{\frac{\partial \bar{\rho}(1-\tilde{b})}{\partial t}}_{1} + \underbrace{\nabla.\left(\bar{\rho}\tilde{\mathbf{u}}(1-\tilde{b})\right)}_{2} = \underbrace{\nabla.[\bar{\rho}(\kappa + D_t)\nabla(1-\tilde{b})]}_{3} + \underbrace{\rho_u U_t\left|\nabla(1-\tilde{b})\right|}_{4} + \bar{\rho}W_{ign} + Q \quad \text{(Appendix II.4)}$$

For term 1 in the above equation, we can write

$$\frac{\partial \bar{\rho}(1-\tilde{b})}{\partial t} = \frac{\partial \bar{\rho}}{\partial t} - \frac{\partial \bar{\rho}\tilde{b}}{\partial t} \quad \text{(Appendix II.5)}$$

For term 2, we can write

$$\nabla.\left(\bar{\rho}\tilde{\mathbf{u}}(1-\tilde{b})\right) = \nabla.(\bar{\rho}\tilde{\mathbf{u}}) - \nabla.\left(\bar{\rho}\tilde{\mathbf{u}}\tilde{b}\right) \quad \text{(Appendix II.6)}$$

For term 3, we can write

$$\nabla.[\bar{\rho}(\kappa + D_t)\nabla(1-\tilde{b})] = -\nabla.[\bar{\rho}(\kappa + D_t)\nabla\tilde{b}] \quad \text{(Appendix II.7)}$$

since

$$\nabla(1-\tilde{b}) = -\nabla\tilde{b} \quad \text{(Appendix II.8)}$$

For term 4, we can write

$$\rho_u U_t\left|\nabla(1-\tilde{b})\right| = \rho_u U_t|\nabla\tilde{b}| \quad \text{(Appendix II.9)}$$

since $\left|\nabla(1-\tilde{b})\right| = |\nabla\tilde{b}|$.

Therefore, Equation (Appendix II.4) can be rewritten as

$$\frac{\partial \bar{\rho}}{\partial t} - \frac{\partial \bar{\rho}\tilde{b}}{\partial t} + \nabla.(\bar{\rho}\tilde{\mathbf{u}}) - \nabla.\left(\bar{\rho}\tilde{\mathbf{u}}\tilde{b}\right) = -\nabla.[\bar{\rho}(\kappa + D_t)\nabla\tilde{b}] + \rho_u U_t\left|\nabla\tilde{b}\right| + Q + \bar{\rho}W_{ign} \quad \text{(Appendix II.10)}$$

Due to mass conservation,

$$\frac{\partial \bar{\rho}}{\partial t} + \nabla.(\bar{\rho}\tilde{\mathbf{u}}) = 0 \quad \text{(Appendix II.11)}$$

and Equation (Appendix II.10) finally reads

$$-\frac{\partial \bar{\rho}\tilde{b}}{\partial t} - \nabla.\left(\bar{\rho}\tilde{\mathbf{u}}\tilde{b}\right) = -\nabla.[\bar{\rho}(\kappa + D_t)\nabla\tilde{b}] + \rho_u U_t\left|\nabla\tilde{b}\right| + Q + \bar{\rho}W_{ign} \quad \text{(Appendix II.12)}$$

Reorganizing Equation (Appendix II.12) and substituting Equations (Appendix II.2) and (Appendix

II.3) into Equation (Appendix II.12), we get the following balance equation for the regress variable

$$\frac{\partial \bar{\rho}\tilde{b}}{\partial t} + \nabla.\left(\bar{\rho}\tilde{\mathbf{u}}\tilde{b}\right) - \nabla.\left[\bar{\rho}(\kappa + D_t)\nabla\tilde{b}\right] + \rho_u U_t\left|\nabla\tilde{b}\right| + \frac{\bar{\rho}\tilde{b}}{t_r(1 + D_t/\kappa_b)}\exp\left(-\frac{\Theta}{\tilde{T}}\right) + \bar{\rho}W_0\exp\left\{-\left[\left(\frac{r}{\sigma_r}\right)^2 + \left(\frac{t - t_0}{\sigma_t}\right)^2\right]\right\}\tilde{b} = 0 \quad \text{(Appendix II.13)}$$

Note, if the term with $\tilde{b}$ is on the LHS of equation with a plus sign, it will decrease $\tilde{b}$ and act as a sink term in the transport equation.

Appendix III. Analytical solutions to the FSC model equations

As proved elsewhere [17], the planar 1-D Equation (Appendix II.1) without the last source term and with $\kappa = 0$, i.e. the truncated FSC model, has the following benchmark analytical solution

$$\bar{c} = \frac{1}{2}\mathrm{erfc}\left(\xi\sqrt{\pi}\right) = \sqrt{\frac{1}{\pi}}\int_{\zeta\sqrt{\pi}}^{\infty} e^{-\zeta^2} d\zeta \qquad \text{(Appendix III.1)}$$

provided that Equation (3) holds and neither D_t nor U_t varies in the space. The complementary error function $\mathrm{erfc}\left(\xi\sqrt{\pi}\right)$ can be calculated using a python script reported in Appendix IV. Here,

$$\xi = \frac{x - x_f}{\Delta_t} \qquad \text{(Appendix III.2)}$$

is the normalized distance, x_f is the mean flame position, associated with $\bar{c}$=0.5, mean flame brush thickness Δ_t is defined as follows

$$\Delta_t = \frac{1}{\left|\frac{d\bar{c}}{dx}\right|_{\max}} \qquad \text{(Appendix III.3)}$$

and grows following turbulent diffusion law

$$\Delta_t = 2u'\left\{\pi\tau_L t_{fd}\left[1 - \frac{\tau_L}{t_{fd}} + \frac{\tau_L}{t_{fd}}\exp\left(-\frac{t_{fd}}{\tau_L}\right)\right]\right\}^{1/2}. \qquad \text{(Appendix III.4)}$$

This solution describes a developing turbulent wave with self-similar mean structure, i.e. dependence of $\bar{c}$ on two independent variables x and t reduces to $\bar{c}(\xi)$. It is also worth noting that, while the transport Equation (Appendix II.1) describes evolution of the Favre-averaged combustion progress variable, the solution is written for the Reynolds-averaged progress variable using the BML identity $\bar{\rho}\tilde{c} = \rho_b\bar{c}$ [24].

In the considered 1-D planar case, turbulent flame speed is equal to turbulent burning velocity given by Equations (Appendix I.3) and (Appendix I.5). In 1-D spherical case, turbulent flame speed is reduced due to mean curvature of the flame brush. This reduction effect is approximately evaluated as follows

$$\frac{1}{\sigma}\frac{dx_f}{dt} = U_t\int_0^{\infty}\tilde{c}rdr\left\{\int_0^{\infty}\bar{c}rdr\right\}^{-1}, \qquad \text{(Appendix III.5)}$$

see Equation (7.147) on page 364 in Ref. [28].

Appendix IV. Python script of calculating complementary error function

```
import numpy as np   # import numpy library

zeta=np.arange(-2,2.1,0.1)  #zeta is ξ in Equation (Appendix III.2)
xi=zeta/pow(math.pi,0.5)    #xi is ζ in Equation (Appendix III.1)
zn=np.where(zeta>=0,1.,-1.)   # if xi >=0, yield 1, otherwise yeild -1
zz=1./(1.+0.47047*zeta*zn)
f=0.5*(1.+zn*(1.-(0.3480242*zz-0.0958709*zz*zz+0.7478556*zz*zz*zz)*np.exp(-zeta*zeta)))
cBar_ana=1.-f   #cBar_ana is the analytical Reynolds-averaged progress variable
```

Appendix V. Implementation of model in solver and library

The newly developed solver which deals with cornflour dust explosion is called FSCFoam_cornflour. It is based on the XiFoam solver in standard OpenFOAM, the files which have been modified are as follows:

FSCFoam_cornflour.C //the main file
createFields.H //create fields
myCreateFields.H
createFieldRefs.H
createOutput.H // create output file
infoDataOutput.H
readCombustionProperties.H
readThermophysicalProperties.H
readCBarIsosurfaces.H // create and calculate flame positions
calculateCBarIsosurfaces.H
bEqn.H //solve regress variable equation
St_FSC.H //calculate turbulent burning velocity

Part of the FSCFoam_cornflour.C is shown here with important implementations commented.

```
    // --- Pressure-velocity PIMPLE corrector loop
    while (pimple.loop())
    {
        #include "rhoEqn.H"    // include mass conservation equation, from Ehsans code 190605
        #include "UEqn.H"

        #include "bEqn.H"    // solve regress variable equation
//not solve energy equations, but update psi, T, mut, alphat
        thermo.correct();
// calculate a new density based on BML, compare OF density with new density
        rho = thermo.rho();
        rhoBML = p/(Rconst*(b*thermo.Tu()/Wreac+(1.-b)*thermo.Tb()/Wprod));
        rhoDiff = (rho - rhoBML)/rho;
//calculate Reynolds-averaged c using BML
        cBar = thermo.rhou()*c/(thermo.rhob()*(1.-c)+thermo.rhou()*c);

        // --- Pressure corrector loop
        while (pimple.correct())
        {
            #include "pEqn.H"
        }

        if (pimple.turbCorr())
        {
            turbulence->correct();
// update alphat according to correctNut() in
src/TurbulenceModels/compressible/EddyDiffusivity/EddyDiffusivity.C; CH 2019-08-22
            turbulence->correctEnergyTransport();
        }
    }

//recalculated densities and Reynolds-averaged c
    rho = thermo.rho();
```

```
        rhoBML = p/(Rconst*(b*thermo.Tu()/Wreac+(1.-b)*thermo.Tb()/Wprod));
        rhoDiff = (rho - rhoBML)/rho;
        cBar = thermo.rhou()*c/(thermo.rhob()*(1-c)+thermo.rhou()*c);
        rhoDiff_max = max(rhoDiff).value();

// for parallel computation but works for serial computing
// reducing values of a variable from all processors to one value, using a max operator
        reduce(rhoDiff_max, maxOp<scalar>());
        #include "calculateCBarIsosurfaces.H"   //calculate cBar iso-surfaces
        #include "infoDataOutput.H"   //output data
```

Part of bEqn.H file is shown here with comments to code

```
    //calculate unburned and burned denisty
    rhou = thermo.rhou();
    rhob = thermo.rhob();
……
// do not use OF ignition model
//    #include "StCorr.H"

    // Calculate turbulent flame speed flux
    // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//    surfaceScalarField phiSt("phiSt", fvc::interpolate(rhou*StCorr*Su*Xi)*nf);
//do not use OF ignition model
    surfaceScalarField phiSt("phiSt", fvc::interpolate(rhou*Su*Xi)*nf);

//----------------CH2019-04-01--------------------//
    up = uPrimeCoef*sqrt((2.0/3.0)*turbulence->k());
    Info << "min(up)= " << min(up) << endl;
    Info << "max(up)= " << max(up) << endl;
// alphat : turbulent thermo diffusivity    [kg/(ms)]
    alphat=turbulence->alphat();
//    alpha_turb=0.09*sqr(turbulence->k())/(0.7*turbulence->epsilon())*thermo.rho();
// alpha_thermo : laminar thermo diffusivity
    alpha_thermo=thermo.alpha();
    Info << "min alpha_thermo = " << min(alpha_thermo) << endl;
    Info << "max alpha_thermo = " << max(alpha_thermo) << endl;
// tau_turb_prime : Lagrangian time scale of turbulence
    tau_turb_prime=alphat/(rho*pow(up,2));
//    tau_turb_prime=0.09*1.5*turbulence->k()/(0.7*turbulence->epsilon());
// calcualte t_fd : flame development time
    t_fd = runTime - t_0;
//    t_fd = runTime; // flame development time is the runTime
    alpha_turb_time=1.-exp(-t_fd/tau_turb_prime);
    // time-dependent term plays role only after ignition t_0
    if (t_fd.value() <= 0. )
    {
        alpha_turb_transient = 0.0*alphat;
    }
    else{
        alpha_turb_transient=alphat*alpha_turb_time;
    }
    //ignition source term increase slowly
    W_ign_coeff = exp(-pow(ign_dist/sigma_r,2)-pow(t_fd/sigma_t,2));
    myPrt=turbulence->mut()/(turbulence->alphat()+alphatSMALL);
```

```
  // Create b equation
   // ~~~~~~~~~~~~~~~~~
   fvScalarMatrix bEqn
   (
      fvm::ddt(rho, b)                                  // term 1: transient
    + mvConvection->fvmDiv(phi, b)                      // term 2: convection
    - fvm::laplacian(alpha_turb_transient+alpha_thermo, b)   // term 3: diffusion
    + fvm::div(phiSt, b)                                // term 4: gradient
    - fvm::Sp(fvc::div(phiSt), b)                       // term 4: gradient
   ==
      fvOptions(rho, b)
   //fvm::SuSp() implicit treatment of source or sink term depending on sign
    - fvm::SuSp(rho*W_0*W_ign_coeff,b) // term 5: ignition
    - fvm::SuSp(rho/tr/(1.+alpha_turb_transient/alphab_thermo)*exp(-Ta/thermo.T()),b) // term 6:
extra source term
   );

// remove ignition model from OF
//    #include "ignite.H"

   // Solve for b
   // ~~~~~~~~~~~~
   bEqn.relax();

   fvOptions.constrain(bEqn);

   bEqn.solve();

   fvOptions.correct(b);

   // Limit b between 0 and 1
   b.min(1.0); //force min value being 0
   b.max(0.0); //force max value being 1
// calculate progress variable c
   c = 1. - b;
   c_max = max(c).value();
//calculate cBar
   cBar = thermo.rhou()*c/(thermo.rhob()*(1.-c)+thermo.rhou()*c);
……
   // Calculate Xi according to the selected flame wrinkling model
   // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   if (XiModel == "fixed")
   {
      // Do nothing, Xi is fixed!
   }
……
//calculate turbulent burning velocity using FSC model, CH180320
   else if (XiModel == "FSC")
   {
       #include    "St_FSC.H"
   }
```

Part of St_FSC.H file is shown here with comments to code

```
    Info<< "Open St_FSC.H file \n"<< endl;
//calculate turbulent length scale L_t
    L_t = CdCoef*pow(turbulence->k(),1.5)/turbulence->epsilon();
// calculate turbulent time scale tau_turb
    tau_turb = L_t/up;
//calculate chemical time scale [s]
    alphau_thermo = thermo.muu()/Pr;
    tau_chem = alphau_thermo/(rhou*pow(Su,2));
//calculate Damkohler number Da
    Da = tau_turb/tau_chem;
//calculate turbulent flame speed St
    St = A*up*pow(Da,0.25);
    // time-dependent term of turbulent flame speed
    // FSC model plays role only after ignition
    if (t_fd.value() <= 0.)
    {
        St_time = 0.;
    }
    else
    {
        St_time = pow(1.+tau_turb_prime/(t_fd+TimeSMALL)*(exp(-
t_fd/(tau_turb_prime+TimeSMALL))-1.), 0.5);
//       St_time = pow(1.+tau_turb_prime/t_fd*(exp(-t_fd/tau_turb_prime)-1.), 0.5); // for analytical
solution
    }

    // transient turbulent flame speed St_transient
    St_transient = St * St_time;
    Xi = St_transient/Su;
    Info<< "End St_FSC.H file \n"<< endl;
```

Part of the code where the calculation of mean temperature and density are evaluated in the library of src/thermophysicalModels/reactionThermo/psiuReactionThermo/heheuPsiThermo.C

```
void Foam::heheuPsiThermoBML_cornflour<BasicPsiThermo, MixtureType>::calculate()
{
    const scalarField& pCells = this->p_;

    scalarField& TCells = this->T_.primitiveFieldRef();
    scalarField& TuCells = this->Tu_.primitiveFieldRef();
    scalarField& psiCells = this->psi_.primitiveFieldRef();
    scalarField& muCells = this->mu_.primitiveFieldRef();
    scalarField& alphaCells = this->alpha_.primitiveFieldRef();
//get access to regress variable, CH190408
    const scalarField& bCells = this->composition().Y("b");
    volScalarField psiu_(psiu());
    scalarField& psiuCells = psiu_;
    volScalarField psib_(psib());
    scalarField& psibCells = psib_;
    volScalarField Tb_(Tb());
    scalarField& TbCells = Tb_;
    volScalarField muu_(muu());
    scalarField& muuCells = muu_;
    volScalarField mub_(mub());
    scalarField& mubCells = mub_;
```

```
    forAll(TCells, celli)
    {
//      const typename MixtureType::thermoType& mixture_ =
//          this->cellMixture(celli);
        // reset Tu, Tb, psiu and psib
        TuCells[celli] = 328; //initial temperture
        TbCells[celli] = 1500;   // calculated value of 1592
        psiuCells[celli] = this->cellReactants(celli).psi(pCells[celli], TuCells[celli]);
        psibCells[celli] = this->cellProducts(celli).psi(pCells[celli], TbCells[celli]);
        muuCells[celli] = this->cellReactants(celli).mu(pCells[celli], TuCells[celli]);
        mubCells[celli] = this->cellProducts(celli).mu(pCells[celli], TbCells[celli]);


        if (this->updateT())
        {
            if (bCells[celli] > 0.999) //if unburned, properties are set to unburned
            {
                TCells[celli] = TuCells[celli];
                psiCells[celli] = psiuCells[celli];
                muCells[celli] = muuCells[celli];
                alphaCells[celli] = this->cellReactants(celli).alphah(pCells[celli], TuCells[celli]);
            }
            else if (bCells[celli] < 0.001) //if burned, properties are set to burned
            {
                TCells[celli] = TbCells[celli];
                psiCells[celli] = psibCells[celli];
                muCells[celli] = mubCells[celli];
                alphaCells[celli] = this->cellProducts(celli).alphah(pCells[celli], TbCells[celli]);
            }
            else //if in between, properties are calculated by BML
            {
                TCells[celli] = bCells[celli]*TuCells[celli]+(1.0-bCells[celli])*TbCells[celli];
                psiCells[celli] = psiuCells[celli]/(bCells[celli]+psiuCells[celli]/psibCells[celli]*(1.0-
bCells[celli]));
                muCells[celli] = bCells[celli]*muuCells[celli]+(1.0-bCells[celli])*mubCells[celli];
                scalar alphauCells = this->cellReactants(celli).alphah(pCells[celli], TuCells[celli]);
                scalar alphabCells = this->cellProducts(celli).alphah(pCells[celli], TbCells[celli]);
                alphaCells[celli] = bCells[celli]*alphauCells + (1.0-bCells[celli])*alphabCells;
            }
        }
    }

……
//get access to regress variable, CH 190623
    volScalarField::Boundary& bBf =
        this->composition().Y("b").boundaryFieldRef();

    volScalarField::Boundary& psiuBf =
        psiu_.boundaryFieldRef();

    volScalarField::Boundary& psibBf =
        psib_.boundaryFieldRef();

    volScalarField::Boundary& TbBf =
        Tb_.boundaryFieldRef();
```

```
    volScalarField::Boundary& muuBf =
        muu_.boundaryFieldRef();

    volScalarField::Boundary& mubBf =
        mub_.boundaryFieldRef();
//------------------------------end 190623----------------------------//
    forAll(this->T_.boundaryField(), patchi)
    {
        fvPatchScalarField& pp = pBf[patchi];
        fvPatchScalarField& pT = TBf[patchi];
        fvPatchScalarField& pTu = TuBf[patchi];
        fvPatchScalarField& ppsi = psiBf[patchi];
        fvPatchScalarField& pmu = muBf[patchi];
        fvPatchScalarField& palpha = alphaBf[patchi];
//----------------------------CH 190623-----------------------------//
        fvPatchScalarField& pb = bBf[patchi];
        fvPatchScalarField& ppsiu = psiuBf[patchi];
        fvPatchScalarField& ppsib = psibBf[patchi];
        fvPatchScalarField& pTb = TbBf[patchi];
        fvPatchScalarField& pmuu = muuBf[patchi];
        fvPatchScalarField& pmub = mubBf[patchi];
//------------------------------end 190623----------------------------//

        if (pT.fixesValue())
        {
            forAll(pT, facei)
            {
//                const typename MixtureType::thermoType& mixture_ =
//                    this->patchFaceMixture(patchi, facei);
//reset mixture properties, CH 190623
                pTu[facei] = 328;
                pTb[facei] = 1500;  // calculated value of 1592
                ppsiu[facei] = this->patchFaceReactants(patchi, facei).psi(pp[facei],pTu[facei]);
                ppsib[facei] = this->patchFaceProducts(patchi, facei).psi(pp[facei],pTb[facei]);
                pmuu[facei] = this->patchFaceReactants(patchi, facei).mu(pp[facei], pTu[facei]);
                pmub[facei] = this->patchFaceProducts(patchi, facei).mu(pp[facei], pTb[facei]);
                if(pb[facei] > 0.999)  // if unburned, boundary properties are set unburned
                {
                    ppsi[facei] = ppsiu[facei];
                    pmu[facei] = pmuu[facei];
                    palpha[facei] = this->patchFaceReactants(patchi, facei).alphah(pp[facei], pTu[facei]);
                }
                else if(pb[facei] < 0.001) // if burned, boundary properties are set burned

                {
                    ppsi[facei] = ppsib[facei];
                    pmu[facei] = pmub[facei];
                    palpha[facei] = this->patchFaceProducts(patchi, facei).alphah(pp[facei], pTb[facei]);
                }
                else  //if in between, boundary properties follow BML
                {
                    ppsi[facei] = ppsiu[facei]/(pb[facei]+ppsiu[facei]/ppsib[facei]*(1.0-pb[facei]));
                    pmu[facei] = pmuu[facei]*pb[facei]+pmub[facei]*(1.-pb[facei]);
                    scalar palphau = this->patchFaceReactants(patchi, facei).alphah(pp[facei], pTu[facei]);
```

```
                scalar palphab = this->patchFaceProducts(patchi, facei).alphah(pp[facei], pTb[facei]);
                palpha[facei] = pb[facei]*palphau + (1.-pb[facei])*palphab;
            }
//------------------------------end 190623----------------------------//
        }
      }
      else
      {
        forAll(pT, facei)
        {
//            const typename MixtureType::thermoType& mixture_ =
//                this->patchFaceMixture(patchi, facei);
//similar as before, CH 190623
            pTu[facei] = 328;
            pTb[facei] = 1500; // calculated value of 1592
            ppsiu[facei] = this->patchFaceReactants(patchi, facei).psi(pp[facei],pTu[facei]);
            ppsib[facei] = this->patchFaceProducts(patchi, facei).psi(pp[facei],pTb[facei]);
            pmuu[facei] = this->patchFaceReactants(patchi, facei).mu(pp[facei], pTu[facei]);
            pmub[facei] = this->patchFaceProducts(patchi, facei).mu(pp[facei], pTb[facei]);
            if (this->updateT())
            {
              if(pb[facei] > 0.999)
              {
                pT[facei] = pTu[facei];
                ppsi[facei] = ppsiu[facei];
                pmu[facei] = pmuu[facei];
                palpha[facei] = this->patchFaceReactants(patchi, facei).alphah(pp[facei], pTu[facei]);
              }
              else if(pb[facei] < 0.001)
              {
                pT[facei] = pTb[facei];
                ppsi[facei] = ppsib[facei];
                pmu[facei] = pmub[facei];
                palpha[facei] = this->patchFaceProducts(patchi, facei).alphah(pp[facei], pTb[facei]);
              }
              else
              {
                pT[facei] = pb[facei]*pTu[facei]+(1.0-pb[facei])*pTb[facei];
                ppsi[facei] = ppsiu[facei]/(pb[facei]+ppsiu[facei]/ppsib[facei]*(1.0-pb[facei]));
                pmu[facei] = pmuu[facei]*pb[facei]+pmub[facei]*(1.-pb[facei]);
                scalar palphau = this->patchFaceReactants(patchi, facei).alphah(pp[facei], pTu[facei]);
                scalar palphab = this->patchFaceProducts(patchi, facei).alphah(pp[facei], pTb[facei]);
                palpha[facei] = pb[facei]*palphau + (1.-pb[facei])*palphab;
              }
//------------------------------end 190623----------------------------//
            }
        }
      }
   }
}

……
template<class BasicPsiThermo, class MixtureType>
Foam::tmp<Foam::volScalarField>
Foam::heheuPsiThermoBML_cornflour<BasicPsiThermo, MixtureType>::Tb() const
```

```
{
    tmp<volScalarField> tTb
    (
        new volScalarField
        (
            IOobject
            (
                "Tb",
                this->T_.time().timeName(),
                this->T_.db(),
                IOobject::NO_READ,
                IOobject::NO_WRITE,
                false
            ),
            this->T_
        )
    );

    volScalarField& Tb_ = tTb.ref();
    scalarField& TbCells = Tb_.primitiveFieldRef();

    forAll(TbCells, celli)
    {
        TbCells[celli] = 1500;  //set burned temperature
    }

    volScalarField::Boundary& TbBf = Tb_.boundaryFieldRef();

    forAll(TbBf, patchi)
    {
        fvPatchScalarField& pTb = TbBf[patchi];

        forAll(pTb, facei)
        {
            pTb[facei] = 1592; //set burned temperature
        }
    }

    return tTb;
}

……
template<class BasicPsiThermo, class MixtureType>
Foam::tmp<Foam::volScalarField>
Foam::heheuPsiThermoBML_cornflour<BasicPsiThermo, MixtureType>::psib() const
{
    tmp<volScalarField> tpsib
    (
        new volScalarField
        (
            IOobject
            (
                "psib",
                this->psi_.time().timeName(),
                this->psi_.db(),
```

```
            IOobject::NO_READ,
            IOobject::NO_WRITE,
            false
        ),
        this->psi_.mesh(),
        this->psi_.dimensions()
    )
  );

  volScalarField& psib = tpsib.ref();
  scalarField& psibCells = psib.primitiveFieldRef();
  const volScalarField Tb_(Tb());
  const scalarField& TbCells = Tb_;
  const scalarField& pCells = this->p_;

  forAll(psibCells, celli)
  {
    psibCells[celli] =
//          this->cellReactants(celli).psi(pCells[celli], TbCells[celli]);  bug, should use products
properties
        this->cellProducts(celli).psi(pCells[celli], TbCells[celli]);
  }

  volScalarField::Boundary& psibBf = psib.boundaryFieldRef();

  forAll(psibBf, patchi)
  {
    fvPatchScalarField& ppsib = psibBf[patchi];

    const fvPatchScalarField& pp = this->p_.boundaryField()[patchi];
    const fvPatchScalarField& pTb = Tb_.boundaryField()[patchi];

    forAll(ppsib, facei)
    {
//        ppsib[facei] =
//            this->patchFaceReactants
//            (patchi, facei).psi(pp[facei], pTb[facei]);
        ppsib[facei] =
            this->patchFaceProducts   //bug, should be products
            (patchi, facei).psi(pp[facei], pTb[facei]);
    }
  }

  return tpsib;
}
```

Appendix VI. Calculation of laminar and turbulent viscosity and heat diffusivity

Kinematic viscosity ν [m2/s]
nu(), laminar kinematic viscosity, src/TurbulenceModels/ turbulenceModels/turbulenceModel.H
nut(), turbulent kinematic viscosity
nuEff(), effective kinematic viscosity, turbulent + laminar

Dynamic viscosity μ [kg/m/s]
mu(), laminar dynamic viscosity, $src/TurbulenceModels/turbulenceModels/turbulenceModel.H
mut(), turbulence dynamic viscosity
muEff(), effective dynamic viscosity, turbulent + laminar

Thermal diffusivity α [kg/m/s]
Laminar thermal diffusivity, thermo.alpha()=thermo.mu()/Pr; see code src/thermophysicalModels/specie/transport/const/constTransportI.H; by changing constant/thermopysicalProperties, reactants or products mu and Pr, you can change the laminar dynamic and thermal viscosity.
alphat(), turbulent thermal diffusivity for enthalpy, src/TurbulenceModels/compressible/EddyDiffusivity/EddyDiffusivity.H
alphaEff() [kg/m/s], effective turbulent thermal diffusivity for enthalpy
alphat_ = this->rho_*this->nut()/Prt_; src/TurbulenceModels/ compressible/EddyDiffusivity/EddyDiffusivity.C

Appendix VII. Case setup for 1-D "frozen" turbulence planar flame

The case structure is as follows

```
0.org  Allclean  Allrun  constant  system
```

In 0.org/ folder, it contains files for setting up initial and boundary conditions

```
alphat  b  epsilon  k  nut  p  Su  T  Tb  Tu  U  Xi
```

Part of the files are shown here in order to save space. In 0.org/alphat file,

```
    object      alphat;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -1 0 0 0 0];


internalField   uniform 0.013224; //for unburned, C_mu*rhou*k2/(Pr_t*epsilon)

boundaryField
{
    left
    {
        type          zeroGradient;
    }
    right
    {
        type          zeroGradient;
    }
    top
    {
        type          cyclic;
    }
    bottom
    {
        type          cyclic;
    }
    front
    {
        type          cyclic;
    }
    back
    {
        type          cyclic;
    }
}
```

In 0.org/b file,

```
    object      b;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];
```

```
internalField   uniform 1;

boundaryField
{
   left
   {
      type          zeroGradient;
   }
   right
   {
      type          fixedValue;
      value         uniform 1;
   }
```

In 0.org/epsilon file,

```
   object     epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions     [0 2 -3 0 0 0 0];


internalField   uniform 11.84;

boundaryField
{
   left
   {
      type          fixedValue;
      value         $internalField;
   }
   right
   {
      type          fixedValue;
      value         $internalField;
   }
```

In 0.org/k file,

```
   object     k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions     [0 2 -2 0 0 0 0];


internalField   uniform 0.96; //0.96;

boundaryField
{
   left
   {
      type          fixedValue;
      value         $internalField;
```

```
    }
    right
    {
        type          fixedValue;
        value         $internalField;
    }
```

In 0.org/nut file,

```
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];


internalField   uniform 0.007;  //C_mu*k2/epsilon

boundaryField
{
    left
    {
        type          zeroGradient;
    }
    right
    {
        type          zeroGradient;
    }
```

In 0.org/p file,

```
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];


internalField   uniform 110000;

boundaryField
{
    left
    {
        type          totalPressure;
        p0            $internalField;
    }
    right
    {
        type          fixedValue;
        value         $internalField;
    }
```

In 0.org/Su file,

```
    object      Su;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
dimensions      [0 1 -1 0 0 0 0];


internalField   uniform 0.12;

boundaryField
{
    left
    {
        type            fixedValue;
        value           $internalField;
    }
    right
    {
        type            fixedValue;
        value           $internalField;
    }
```

In 0.org/T file,

```
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];


internalField   uniform 328;

boundaryField
{
    left
    {
        type            zeroGradient;
    }
    right
    {
        type            fixedValue;
        value           uniform 328;
    }
```

In 0.org/Tb file,

```
    object      Tb;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];


internalField   uniform 1592;

boundaryField
{
    left
    {
```

```
    type          fixedValue;
    value         uniform 1592;
  }
  right
  {
    type          fixedValue;
    value         uniform 1592;
  }
```

In 0.org/Tu file,

```
  object     Tu;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions     [0 0 0 1 0 0 0];


internalField   uniform 328;

boundaryField
{
  left
  {
    type          fixedValue;
    value         uniform 328;
  }
  right
  {
    type          fixedValue;
    value         uniform 328;
  }
```

In 0.org/U file,

```
  object     U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions     [0 1 -1 0 0 0 0];


internalField   uniform (0 0 0);

boundaryField
{
  left
  {
    type          pressureInletOutletVelocity;
    value         $internalField;
  }
  right
  {
    type          fixedValue;
    value         $internalField;
  }
```

In 0.org/Xi file,

```
    object     Xi;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions     [0 0 0 0 0 0 0];


internalField   uniform 1;

boundaryField
{
    left
    {
        type          zeroGradient;
    }
    right
    {
        type          zeroGradient;
    }
```

In constant/ folder, it contains files for setting up the model

```
combustionProperties thermophysicalProperties  turbulenceProperties
```

In combustionProperties file

```
    object     combustionProperties;
}
……
fuel          C6H7_88O4_98;  //cornflour

Su            Su [0 1 -1 0 0 0 0] 0.12; //laminar burning velocity of cornflour of phi 0.77

SuModel       unstrained;

equivalenceRatio equivalenceRatio [0 0 0 0 0 0 0] 1.0;
…

XiModel       FSC;
…
// coeffeicient for FSC model, evaluating turbulent length scale
CdCoef CdCoef [0 0 0 0 0 0 0] 0.37;

// coeffeicient for FSC model, evaluating turbulent flame speed
A A [0 0 0 0 0 0 0] 0.5;
…
//smooth ignition parameters
W_0 W_0 [0 0 -1 0 0 0 0] 0;         //[1/s]
t_0 t_0 [0 0 1 0 0 0 0] 0;         //[s] no effect at least a factor of 10 of time step
sigma_r sigma_r [0 1 0 0 0 0 0] 1e-3;  //[m] at least a factor of 2 of mesh size
sigma_t sigma_t [0 0 1 0 0 0 0] 5e-5;  //[s] no effect at least a factor of 2 of time step
```

In thermophysicalProperties file

```
    object     thermophysicalProperties;
}
……
```

```
thermoType
{
   type            heheuPsiThermoBML_cornflour;
   mixture         homogeneousMixture;
   transport       const;
   thermo          hConst;
   equationOfState perfectGas;
   specie          specie;
   energy          absoluteEnthalpy;
}

stoichiometricAirFuelMassRatio
   stoichiometricAirFuelMassRatio [0 0 0 0 0 0 0] 4.67;

reactants
{
   specie
   {
      molWeight      32.76; //phi = 0.77 cornflour
   }
   thermodynamics
   {
                  Cp          1007;
                 Hf           0;
   }
   transport
   {
      mu           1.8e-5;
      Pr           0.7;

      As           1.67212e-06;
      Ts           170.672;
   }
}

products
{
   specie
   {
      molWeight      27.15;  //phi = 0.77 cornstarch products
   }
   thermodynamics
   {
                  Cp          1007;
                 Hf           0;
   }
   transport
   {
      mu           4.6e-5;
      Pr           0.7;

      As           1.67212e-06;
      Ts           170.672;
   }
}
```

In turbulenceProperties file

```
    object     turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RAS;

RAS
{
    RASModel       kEpsilon;

    kEpsilonCoeffs
    {
        Prt 0.7;
    }
    turbulence      off; //on;

    printCoeffs     on;
}
```

In system/ folder, it contains files for setting up the numerics

```
blockMeshDict  controlDict      fvSchemes       residuals
cBar0D1Dict     Residuals.txt
cBar0D5Dict    flameFrontDict    fvSolution        setFieldsDict
```

In blockMeshDict file,

```
    object     blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale   0.001;

vertices
(
    (0   0   0)   //0
    (100 0   0)   //1
    (100 3   0)   //2
    (0   3   0)   //3
    (0   0   3)   //4
    (100 0   3)   //5
    (100 3   3)   //6
    (0   3   3)   //7
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (100 3 3) simpleGrading (1 1 1)
);

edges
(
);
```

```
boundary
(
    left
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    right
    {
        type patch;
        faces
        (
            (2 6 5 1)
        );
    }
    top
    {
        type cyclic;
        neighbourPatch bottom;
        faces
        (
            (4 5 6 7)
        );
    }
    bottom
    {
        type cyclic;
        neighbourPatch top;
        faces
        (
            (0 3 2 1)
        );
    }
    front
    {
        type cyclic;
        neighbourPatch back;
        faces
        (
            (0 1 5 4)
        );
    }
    back
    {
        type cyclic;
        neighbourPatch front;
        faces
        (
            (7 6 2 3)
        );
    }
);
```

```
mergePatchPairs
(
);
```

In fvSchemes file, the second order accuracy numerical scheme are used as recommended by [36]

```
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         backward;
}

gradSchemes
{
    default         cellLimited Gauss linear 1;
    grad(U)         cellLimited Gauss linear 1;
}

divSchemes
{
    default             none;
    div(phi,U)          Gauss linear;  //linearUpwindV grad(U);
    div(phid,p)         Gauss linear;  //limitedLinear 1;
    div(phi,k)          Gauss linearUpwind default;  //limitedLinear 1;
    div(phi,K)          Gauss linear;  //limitedLinear 1;
    div(phi,epsilon)    Gauss linearUpwind default;  //limitedLinear 1;
    div(phi,R)          Gauss linear;  //limitedLinear 1;
    div(R)              Gauss linear;  //linear;
    div(phiXi,Xi)       Gauss linear;  //limitedLinear 1;
    div(phiXi,Su)       Gauss linear;  //limitedLinear 1;
    div(phiSt,b)        Gauss limitedLinear01 1;
    div(phi,ft_b_ha_hau) Gauss multivariateSelection
    {
        fu  limitedLinear01 1;
        ft  limitedLinear01 1;
        b   limitedLinear01 1;
        ha  limitedLinear 1;
        hau limitedLinear 1;
    };
    div(U)              Gauss linear;
    div((Su*n))         Gauss linear;
    div((U+((Su*Xi)*n))) Gauss linear;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear limited 1; //corrected;
}

interpolationSchemes
{
```

```
    default         linear;
}

snGradSchemes
{
    default         limited 1; //orthogonal; //corrected;
}


// ************************************************************************* //
```

In fvSolution file,

```
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "(p|rho)"
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-08;
        relTol          0;
        minIter         3;
        maxIter         100;
    }

    "(p|rho)Final"
    {
        $p;
        tolerance       1e-08;
        relTol          0;
    }

    "(U|b|Su|Xi|ha|hau|k|epsilon)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0;
        minIter         3;
        maxIter         100;
    }

    "(U|b|Su|Xi|ha|hau|k|epsilon)Final"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0;
    }
}

PIMPLE  // setup for PISO algorithm
```

```
{
    momentumPredictor yes;
    nOuterCorrectors 1;
    nCorrectors     2;
    nNonOrthogonalCorrectors 1;
    turbOnFinalIterOnly true;

}
```

In residual file, tell OpenFOAM which residuals to be saved

```
#includeEtc "caseDicts/postProcessing/numerical/residuals.cfg"

fields (p U k epsilon b);
```

In setFieldDict file, set left half of the tube as burned

```
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue b 1
    volScalarFieldValue T 328
    volScalarFieldValue alphat 0.013224
);

regions
(
    boxToCell
    {
        box (0 0 0) (0.05 0.003 0.003);
        fieldValues
        (
            volScalarFieldValue b 0
            volScalarFieldValue T 1592
            volScalarFieldValue alphat 0.00225798
        );
    }
);
```

In controlDict file,

```
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     FSCFoam_cornflour;

startFrom       startTime;

startTime       0; //2e-3;

stopAt          endTime; //noWriteNow;

endTime         20e-3;//5e-2;
```

```
deltaT          1e-06;

writeControl    timeStep;

writeInterval   2e3;

purgeWrite      0;

writeFormat     ascii;

writePrecision  10;

writeCompression on;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  no;

maxCo           0.1;

maxDeltaT       1;

functions
{
   #includeFunc  residuals
   sample1
   {
      type       sets;
      libs       ("libsampling.so");
      writeControl writeTime;
      setFormat   raw;
      sets
      (
         line1
         {
            type       uniform;
            axis       distance;

            start      (0   0.0015 0.0015);
            end        (0.1 0.0015 0.0015);
            nPoints    1000;
         }
      );
      interpolationScheme cellPoint;
      fields         (rho c cBar U);
   }
}
```

The script Allrun for running the case is

```
#!/bin/sh
# reset the case
```

```
#echo "reset the case"
#sed -i "s/turbulence      on/turbulence      off/g" constant/turbulenceProperties
#sed -i "s/startTime       2e-4/startTime       0/g" system/controlDict
#sed -i "s/endTime         5e-3/endTime         2e-4/g" system/controlDict
#copy 0 dir
cp -r 0.org/ 0/
wait
echo "blockMesh"
blockMesh > log.blockMesh &
wait
echo "renumberMesh"
#increase the speed of linear solver by renumberMesh utility
renumberMesh -overwrite > log.renumberMesh &
wait
echo "checkMesh"
checkMesh > log.checkMesh &
wait
echo "set field"
setFields > log.setFields &
wait
echo "run "
FSCFoam_cornflour > log.FSCFoam_cornflour &
```

The script Allclean for cleaning the case is

```
rm -rf constant/polyMesh
rm -rf postProcessing
rm -rf VTK
rm log*
foamListTimes -rm
rm -rf 0/
rm -rf dynamicCode
```

Appendix VIII. Case setup for 3-D "frozen" turbulent spherical flame

The case structure is as follows

```
0.org  Allclean_parallel  Allrun_parrallel  constant  system
```

In 0.org/ folder, it contains files for setting up initial and boundary conditions

```
alphat  b  epsilon  k  nut  p  Su  T  Tb  Tu  U  Xi
```

Part of the files are shown here in order to save space. Only 0.org/alphat file is shown here, and the rest of files resembles the setup for 1-D planar flame case in Appendix VII.

```
    object      alphat;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -1 0 0 0 0];


internalField   uniform 0.013224; //for unburned, C_mu*rhou*k2/(Pr_t*epsilon)

boundaryField
{
    left
    {
        type            symmetryPlane;
    }
    right
    {
        type            zeroGradient;
    }
    top
    {
        type            zeroGradient;
    }
    bottom
    {
        type            symmetryPlane;
    }
    front
    {
        type            symmetryPlane;
    }
    back
    {
        type            zeroGradient;
    }
}
```

The rest of the setup of files resembles that of 1-D planar turbulent flame case, and will not be reported here. Only system/decomposeParDict is shown here. Note according to Ref.[37], scotch method for decomposing the domain is used.

```
    object      decomposeParDict;
}
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 16;

method         scotch;
```

The script Allrun_parallel for running the case in parallel is as follows

```
#!/bin/sh

# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions
cp -r 0.org/ 0/
wait
echo "create mesh"
runApplication blockMesh
echo "renumber mesh"
#increase the speed of linear solver by renumberMesh utility
runApplication renumberMesh -overwrite
echo "check mesh"
runApplication checkMesh
echo "set field"
runApplication setFields
echo "decompose"
# Decompose
runApplication decomposePar
echo "run"
# Run
runParallel `getApplication`
echo "reconstruct"
# Reconstruct
runApplication reconstructPar
```

The script Allclean_parallel for cleaning the case in parallel is as follows

```
rm -rf constant/polyMesh
rm -rf processor*
rm -rf postProcessing
rm log*
foamListTimes -rm
rm -rf VTK
rm -rf dynamicCode
rm -rf 0
```

```
scale   0.001;

box 140;
meshSize 0.25;
boxLayer 80;

grading_factor_1 10;//grading factor for near filed
grading_factor_2 6;//grading factor for far field
vertices
(
   (0   0   0)   //0
   ($box 0   0)   //1
```

```
    ($box $box   0)   //2
    (0   $box   0)   //3
    (0   0  $box)   //4
    ($box 0   $box)   //5
    ($box $box  $box)   //6
    (0   $box  $box)   //7
);

blocks
(
hex (0 1 2 3 4 5 6 7) ($boxLayer $boxLayer $boxLayer) simpleGrading (
(
  (0.5 0.875 $grading_factor_1)   //50% of distance and 87.5% of cells
  (0.5 0.125 $grading_factor_2)
)
(
  (0.5 0.875 $grading_factor_1)
  (0.5 0.125 $grading_factor_2)
)
(
  (0.5 0.875 $grading_factor_1)
  (0.5 0.125 $grading_factor_2)
)
)
);
```

Appendix IX. Implementation of an extra source term in the standard k-epsilon turbulence model

The OpenFOAM implementation of standard k-ε turbulence model involves the following balance equations

$$\frac{\partial \bar{\rho}\tilde{k}}{\partial t} + \nabla.\left(\bar{\rho}\tilde{\mathbf{u}}\tilde{k}\right) = \nabla.\left[\bar{\rho}D_k\nabla\tilde{k}\right] + G_k - \frac{2}{3}\bar{\rho}(\nabla.\tilde{\mathbf{u}})\tilde{k} + S_k - \bar{\rho}\tilde{\varepsilon} \quad \text{(Appendix IX.1)}$$

$$\frac{\partial \bar{\rho}\tilde{\varepsilon}}{\partial t} + \nabla.(\bar{\rho}\tilde{\mathbf{u}}\tilde{\varepsilon}) = \nabla.[\bar{\rho}D_\varepsilon\nabla\tilde{\varepsilon}] + \frac{C_1 G_k\tilde{\varepsilon}}{\tilde{k}} - \left(\frac{2}{3}C_1 - C_{3,RDT}\right)\bar{\rho}(\nabla.\tilde{\mathbf{u}})\tilde{\varepsilon} - C_2\bar{\rho}\frac{\tilde{\varepsilon}^2}{\tilde{k}} + S_\varepsilon \quad \text{(Appendix IX.2)}$$

To simulate flames expanding from the centre of a fan-stirred bomb, an extra source term $\bar{\rho}\varepsilon_0$ is added to mimic the flux of turbulent energy from the fans to the centre of the vessel [31]

$$\frac{\partial \bar{\rho}\tilde{k}}{\partial t} + \nabla.\left(\bar{\rho}\tilde{\mathbf{u}}\tilde{k}\right) = \nabla.\left[\bar{\rho}D_k\nabla\tilde{k}\right] + G_k - \frac{2}{3}\bar{\rho}(\nabla.\tilde{\mathbf{u}})\tilde{k} + S_k - \bar{\rho}\tilde{\varepsilon} + \bar{\rho}\varepsilon_0 \quad \text{(Appendix IX.3)}$$

$$\frac{\partial \bar{\rho}\tilde{\varepsilon}}{\partial t} + \nabla.(\bar{\rho}\tilde{\mathbf{u}}\tilde{\varepsilon}) = \nabla.[\bar{\rho}D_\varepsilon\nabla\tilde{\varepsilon}] + \frac{C_1 G_k\tilde{\varepsilon}}{\tilde{k}} - \left(\frac{2}{3}C_1 - C_{3,RDT}\right)\bar{\rho}(\nabla.\tilde{\mathbf{u}})\tilde{\varepsilon} - C_2\frac{\tilde{\varepsilon}}{\tilde{k}}(\bar{\rho}\tilde{\varepsilon} - \bar{\rho}\varepsilon_0) + S_\varepsilon \quad \text{(Appendix IX.4)}$$

Since the TurbulenceModels is a templated class (due to the first capital letter), the implementation of new turbulence model is different from the traditional way. A file, makeTurbModel.C, is created in $WM_PROJECT_USER_DIR/src/TurbulenceModels/turbulenceModels/ with the contents as follows

```
#include "CompressibleTurbulenceModel.H"
#include "compressibleTransportModel.H"
#include "fluidThermo.H"
#include "addToRunTimeSelectionTable.H"
#include "makeTurbulenceModel.H"

#include "ThermalDiffusivity.H"
#include "EddyDiffusivity.H"

#include "laminarModel.H"
#include "RASModel.H"
#include "LESModel.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
#define createBaseTurbulenceModel(                                          \
    Alpha, Rho, baseModel, BaseModel, TDModel, Transport)                   \
                                                                            \
    namespace Foam                                                          \
    {                                                                       \
        typedef TDModel<BaseModel<Transport>>                               \
            Transport##BaseModel;                                           \
        typedef RASModel<EddyDiffusivity<Transport##BaseModel>>             \
            RAS##Transport##BaseModel;                                      \
        typedef LESModel<EddyDiffusivity<Transport##BaseModel>>             \
            LES##Transport##BaseModel;                                      \
    }
```

```
createBaseTurbulenceModel
(
   geometricOneField,
   volScalarField,
   compressibleTurbulenceModel,
   CompressibleTurbulenceModel,
   ThermalDiffusivity,
   fluidThermo
);

#define makeRASModel(Type)                                      \
   makeTemplatedTurbulenceModel                                 \
   (fluidThermoCompressibleTurbulenceModel, RAS, Type)

#define makeLESModel(Type)                                      \
   makeTemplatedTurbulenceModel                                 \
   (fluidThermoCompressibleTurbulenceModel, LES, Type)

#include "mykEpsilon.H"
makeRASModel(mykEpsilon);
```

Take a copy of standard k-epsilon turbulence model into the directory $WM_PROJECT_USER_DIR/src/TurbulenceModels/turbulenceModels/RAS/mykEpsilon. Rename the files from kEpsilon to mykEpsilon. Make the changes in the mykEpsilon.C file for including the source terms as follows

```
……
   //read in epsilon0_ from turbulenceProperties dictionary
   IOdictionary turbulenceProperties
   (
      IOobject
      (
         "turbulenceProperties",
         this->runTime_.constant(),
         this->mesh_,
         IOobject::MUST_READ_IF_MODIFIED,
         IOobject::NO_WRITE
      )
);
dimensionedScalar epsilon0_("epsilon0_", dimensionSet(0,2,-3,0,0,0,0), turbulenceProperties);
   // calculate epsilon0 for including extra source term
volScalarField epsilon0=epsilon_-epsilon0_;
……
   // Dissipation equation
   tmp<fvScalarMatrix> epsEqn
   (
      fvm::ddt(alpha, rho, epsilon_)
    + fvm::div(alphaRhoPhi, epsilon_)
    - fvm::laplacian(alpha*rho*DepsilonEff(), epsilon_)
   ==
      C1_*alpha()*rho()*G*epsilon_()/k_()
    - fvm::SuSp(((2.0/3.0)*C1_ - C3_)*alpha()*rho()*divU, epsilon_)
//     - fvm::Sp(C2_*alpha()*rho()*epsilon_()/k_(), epsilon_)
    - fvm::SuSp(C2_*alpha()*rho()*epsilon0/k_(), epsilon_)//extra source term
    + epsilonSource()
    + fvOptions(alpha, rho, epsilon_)
```

```
    );
……
    // Turbulent kinetic energy equation
    tmp<fvScalarMatrix> kEqn
    (
        fvm::ddt(alpha, rho, k_)
      + fvm::div(alphaRhoPhi, k_)
      - fvm::laplacian(alpha*rho*DkEff(), k_)
     ==
        alpha()*rho()*G
      - fvm::SuSp((2.0/3.0)*alpha()*rho()*divU, k_)
//      - fvm::Sp(alpha()*rho()*epsilon_()/k_(), k_)
      - fvm::SuSp(alpha()*rho()*epsilon0()/k_(), k_)  //extra source term
      + kSource()
      + fvOptions(alpha, rho, k_)
    );
```

The Make/files and Make/options files located in $WM_RPOJECT_USER_DIR/TurbulenceModels/turbulenceModels are shown as follows

```
makeTurbModel.C

LIB = $(FOAM_USER_LIBBIN)/libmyTurbulenceModels
```

```
EXE_INC = \
    -I$(LIB_SRC)/TurbulenceModels/compressible/lnInclude \
    -I$(LIB_SRC)/TurbulenceModels/turbulenceModels/lnInclude \
    -I$(LIB_SRC)/transportModels/compressible/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/basic/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/specie/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/solidThermo/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/solidSpecie/lnInclude \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude

LIB_LIBS = \
    -lcompressibleTurbulenceModels \
    -lcompressibleTransportModels \
    -lfluidThermophysicalModels \
    -lsolidThermo \
    -lsolidSpecie \
    -lturbulenceModels \
    -lspecie \
    -lfiniteVolume \
    -lmeshTools
```