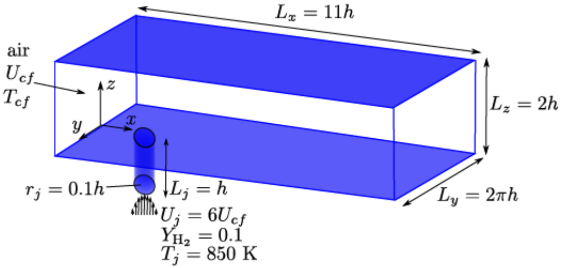
# Direct Numerical Simulation of Autoignition in a Jet in a Cross-Flow

Ammar Abdilghanie

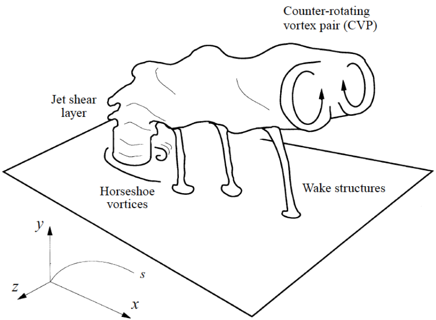
May 15, 2013

# Table of contents

# Overview of JICF




- Mixture tendency to autoignite & stabilize.
- Understanding flame stablization mechanism.
- Passive control of flash-back hazard.




- Turbulence & mixing caused by myriad vortical structures (Horse-shoe, CVP , wake vortices..etc)
- Resolve Kolmogorov & Batchelor scales.
- Flame/Reaction zone thickness.

- Detailed chemistry to accurately capture local extinction & reignition.
- Differential diffusion effects through multicomponent transport models.

$$N = Re^{9/4} \left(\frac{\eta}{\delta}\right)^3 \quad (1)$$

$$Re = \frac{u'\ell}{\nu} \quad (2)$$

$$\delta = \frac{D}{S_L} \quad (3)$$

# Overview of SEM

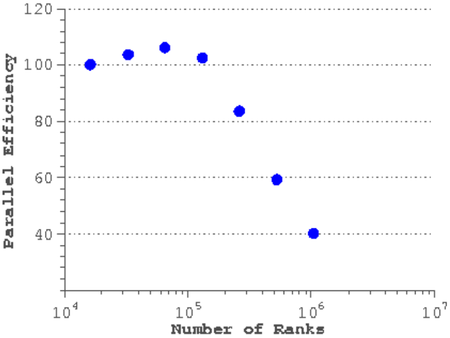
- Variational formulation , like FEM with high order basis function & GLL quadrature.
- Domain decomposition into E deformed quadrilateral/hexahedral elements ( h & p refinements).
- High order accuracy (resolve fine scales of turbulent flows).
- Minimal numerical dispersion.
- Rapid Convergence (exponential for simple geometries).

- Recast tensor products into efficient mxm kernels optimized for BG architecture.
- Scalable ($O(10^6)$ procs) MG iterative solver with low iteration count.
- Efficient automated domain decomposition strategies to ensure load balancing.
- Efficient communication strategies for inter-element data exchange.
- Efficient Parallel IO.
- Readily integrated with UD plug-ins/modules for specialized physics.
- Stiff ODE integrators (CVODE for thermo-chemistry sub-system).

Parallel Efficiency can be defined as:

$$\eta = \frac{N_1 \; t(N_1)}{N_2 \; t(N_2)} \tag{4}$$

where $N_1 > N_2$



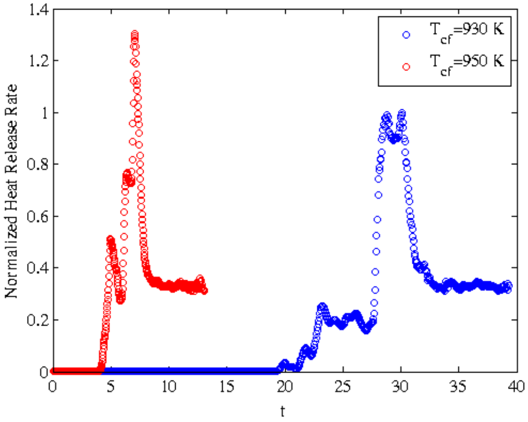
# Overview of Simulation and Post-processing

- Auxiliary channel simulation to generate time-dependent inlet BC.
- Preheated cross-flow air fills the domain initially (t=0).
- Main reactive run.
- Post-processing runs using Nek5000.
- Visualization using VisIt parallel architecture.

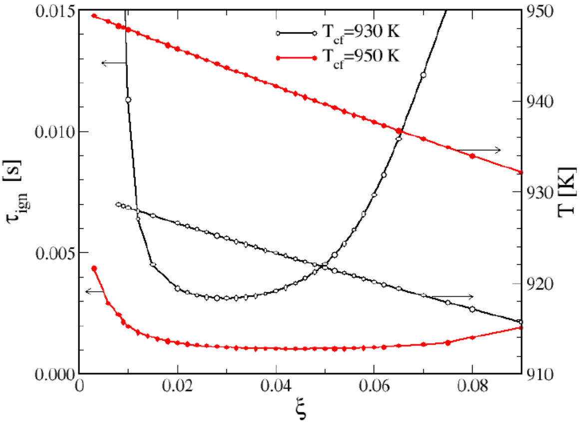
( Animation of JICF for Tcf=930K .....)

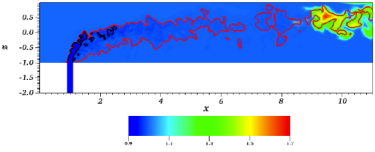
( Animation of JICF for Tcf=950K .....)

# Time History of Integral Heat Release Rate

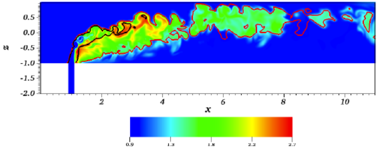


# Local Ignition Delay Estimate

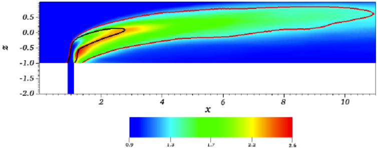



0.5
0.0
-0.5
-1.0
-1.5
-2.0
z
2
4
6
8
10
x

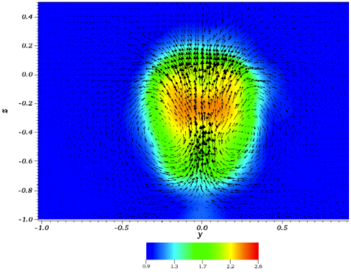
(a)

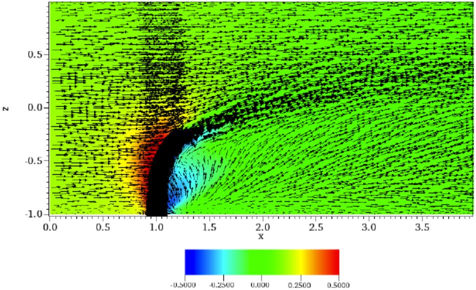

0.5
0.0
-0.5
-1.0
-1.5
-2.0
z
2
4
6
8
10
x

(b)

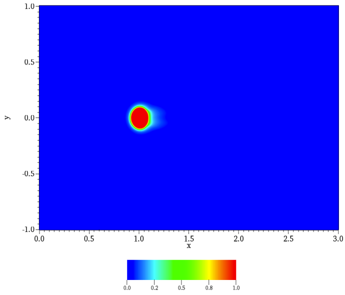

0.5
0.0
-0.5
-1.0
-1.5
-2.0
z
2
4
6
8
10
x

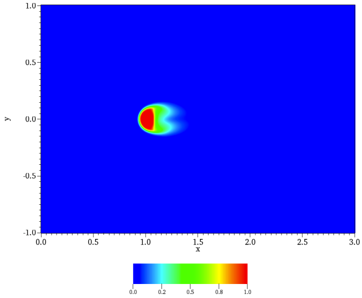
# Mixture Preparation



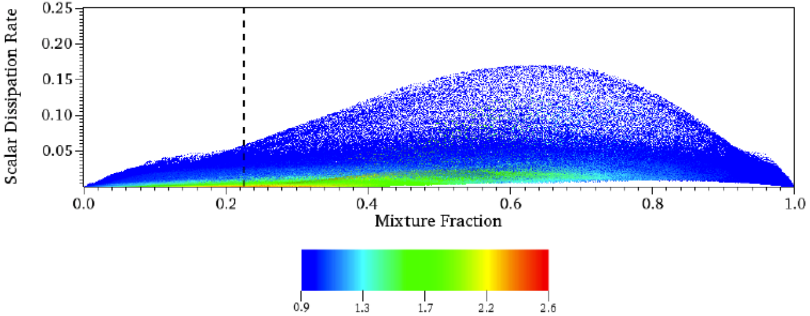



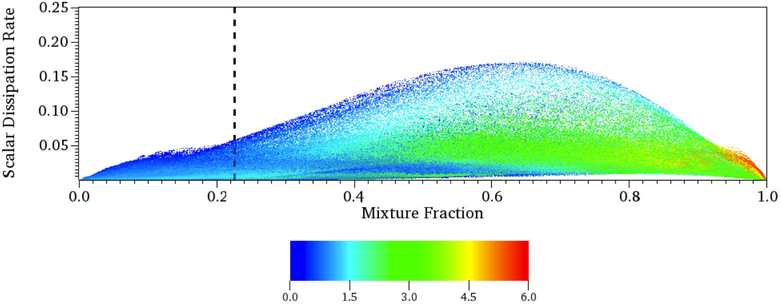
# Jet Deformation



# Aerodynamic Stablization of Flame

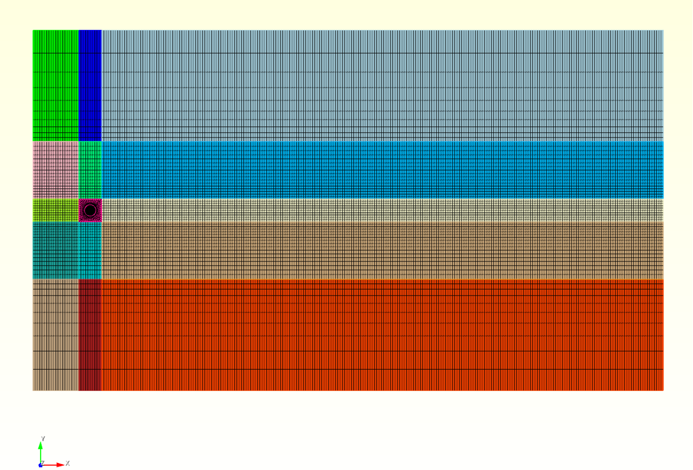


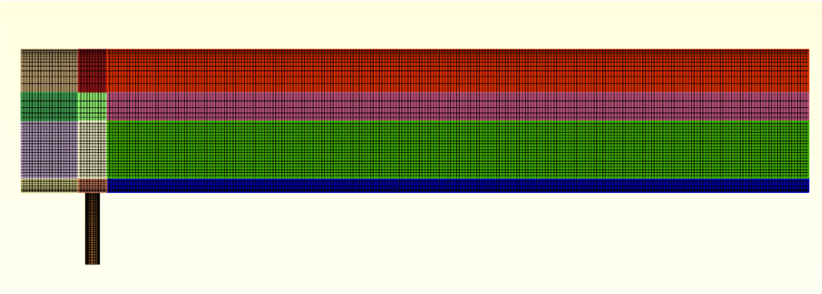

Scalar Dissipation Rate
0.25
0.20
0.15
0.10
0.05
0.0
0.2
0.4
0.6
0.8
1.0
Mixture Fraction
0.0
1.5
3.0
4.5
6.0

# Cubit Mesh Generator

## Allows for journal files using APREPRO and PYTHON

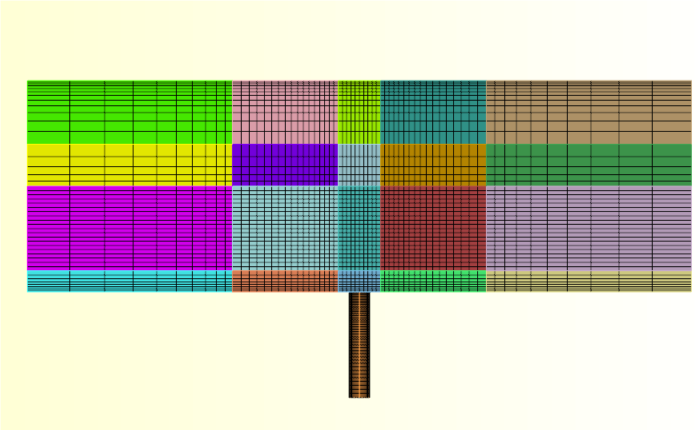
```
#!python
import math
cubit.cmd('reset')
cubit.cmd('############# Geometry Creation####################')
cubit.cmd('############# Channel Floor ########################')
Cylinder_height=1.0
Cylinder_radius=0.1
Lxs=1.0     # xcoordinate of cylinder center line
Lx=[Ldom, Lxs-0.5*Lsq, Lsq, Ldom-(Lxs+0.5*Lsq)]
Ly=[2.0*math.pi, ((1.0*math.pi)-(0.5*Lsq)-(Lyref)) ,   Lyref, Lsq , Lyref ,  ((1.0*math.pi)-(0.5*Lsq)-(Ly
Lz = [ h , 3*ra, 2*ra , 4*ra ,  ra]
#####Partitition the volume into 9 partitions horizontally and four vertically: Blocks
count=0
for j in range(1,len(Ly)):
  for i in range(1,len(Lx)):
      for k in range(1,Nzb):
           cubit.cmd("brick x "+str(Lx[i])+" y "+str(Ly[j])+" z "+str(Lz[k]))
           count=count+1
           cubit.cmd('Volume {Id("volume")} Name'+ '"blok'+str(count)+'"')
           cubit.cmd('move volume  blok'+str(count)+ ' location x '+str(dispx[i])+'  y '+str(dispy[j])+'
#Internal cylinder(s)
for k in range(1,Nzb):
  cubit.cmd('create Cylinder height '+str(Lz[k])+' radius '+str(Cylinder_radius))
```


Y
X



# Mesh can be exported in many formats:

```
- Native cubit format
cubit.cmd('save as "/homes/aabdilghanie/JicfLowReMesh.cub" overwrite')
- FLUENT CFD (.msh format)
cubit.cmd('Export Fluent "/home/aelg/3dmesh" volume all Overwrite')
```



# Basic Options

Mesh can be partitioned and converted to .h5m using **MOAB**:

- mbpart
- mbconvert

**Nek5000** can then be run by linking **MOAB** libraries.

Alternatively a grid can be dumped in **Exodus II** format and converted to Nek-native .rea or .rea2 files using in-house Fortran routines.

# Thank You!