Customizing Cantera's One-Dimensional Flame Code

E. David Huckaby

Cantera Workshop

30th Symposium on Combustion, July 2004

(Last Revised January 2005)

The National Energy Technology Laboratory

Outline

I. How to customize the python interface

II. Structure of Flame Objects (C++/Python)

III. Specific Modifications for the electrostatic interaction with flames

** Also see:*

- *"One Dimensional Flames" lecture presented at the 2004 Workshop by David Goodwin,*
- *Kee, Coltrin and Glarborg (2003)*
- *Modifications to add freely propagating flames.*

Customization Options

A) Data/Function can be derived from existing python interface

B) Data or Function exists in C++ but cannot get to through the python interface

C) Add new functionality to existing class

D) Add a new class

- sub-classing base class
- entirely new

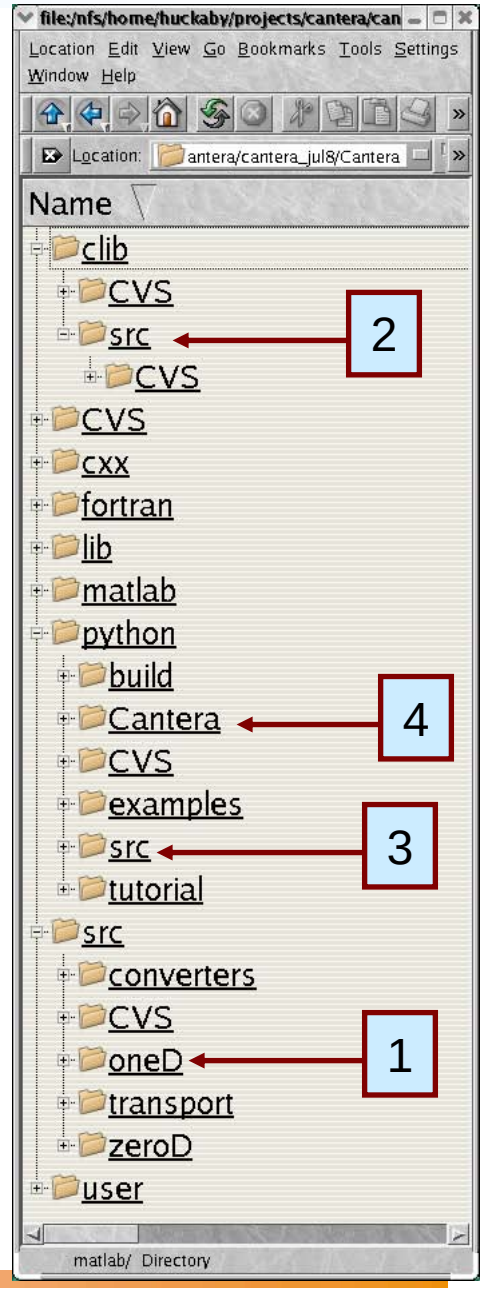
Steps to add new functionality

1) Modify/Create C++, sources, headers and Makefiles

2) Modify/Create a “flat” interface for C++ classes/methods

3) Modify/Create python interface

4) Modify/Create python Classes/methods which manipulate the C++ objects



Manipulate C++

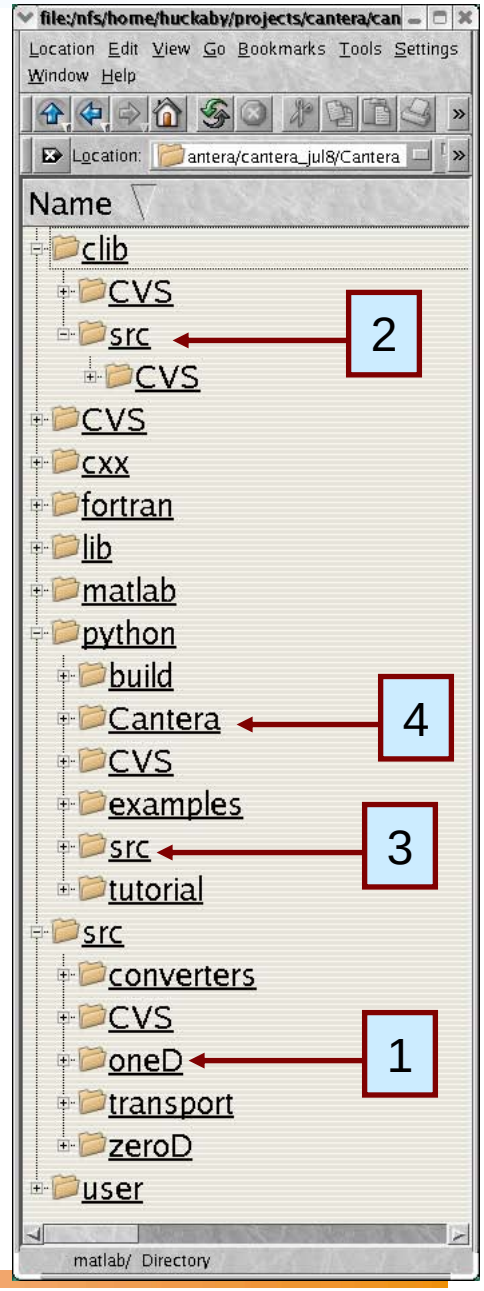
Create C++ sources (.cpp)

Create C++ headers (myclass.h)

Add the sources of any new `.cpp` files to `Makefile` and `Makefile.in`

```
OBJS = MultiJac.o MultiNewton.o\
    Newton_utils.o OneDim.o StFlow.o\
    boundaries1D.o refine.o Sim1D.o\
    boundEfield.o StFlowEfield.o
```

* NOTE: The configure process creates Makefile from Makefile.in, thus any changes made to Makefile will be deleted



Build Generic Language Interface

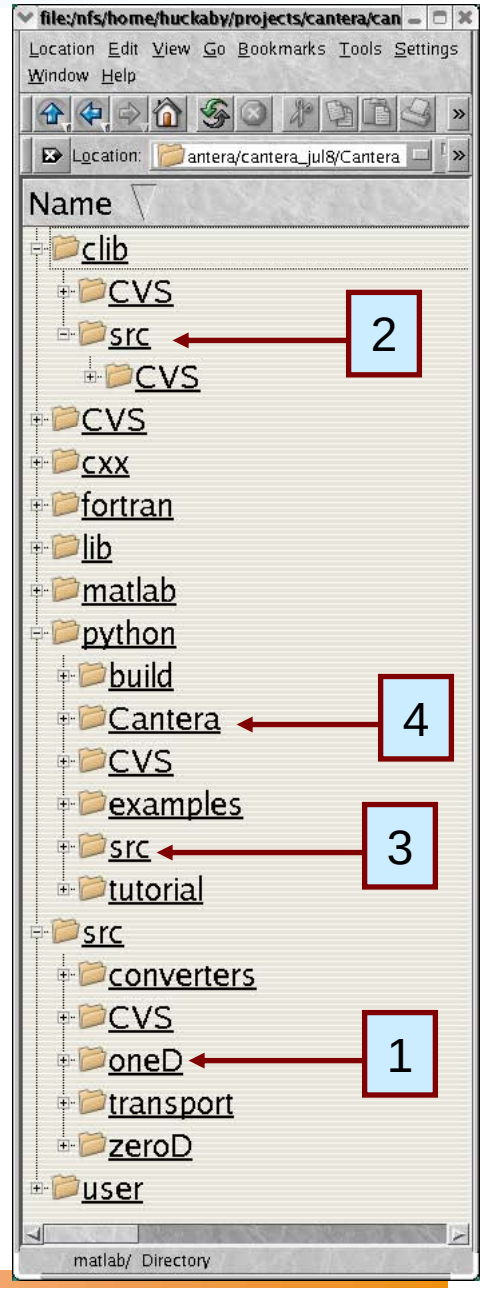
Create a cabinet for the class

Create/Modify new/delete methods

Add functions into source files

Create headers

Create python interfaces:



Interfacing C++ with Other Languages

What is required to interface two languages ?

- The languages must communicate though common data types
- “Flat Interface” - Gray et al.

Data types Available

- Least Common Denominator
- integer, double (real*8), float(real*4), char
- plus contiguous memory blocks of the above

Creating/Using a Cabinet

Cabinet to store 1D flow domains
- vector of Domain1D

```
Cabinet<Domain1D>*   Cabinet<Domain1D>::__storage = 0;
```

Creating an object and store it in the Cabinet of the Base Class

```
int DLL_EXPORT inlet_new() {
        try {
            Inlet1D* i = new Inlet1D();
            return Cabinet<Domain1D>::cabinet()->add(i);
        }
        catch (CanteraError) { return -1; }
    }
```

Wrapper Function for access to a base class member

```
int DLL_EXPORT domain_nPoints(int i) {
        return _domain(i)->nPoints();
    }
```

Creating/Using a Cabinet

To Access a function of a derived class

- “upcast” the pointer to the derived class type
- _bdry upcasts from `Domain1D` to `bdry`

```
double DLL_EXPORT inletEfield_current(int i) {
        try {
            Inlet1D_Efield* inlet = (Inlet1D_Efield*) _bdry(i);
            return inlet->current();
        }
        catch (CanteraError) { return DERR; }
    }
```

Prototype of the function located in ctonedim.h

```
double DLL_IMPORT inletEfield_current(int i);
```

Use genpy.py to create Python interface

Execute to generate the interface

```
python genpy.py ctonedim.h > ctonedim_methods.cpp
```

Generates two additional files:

1) Python Wrapper Class

   pyctmyclass.py

2) Function Prototype for interface

   pymethods.h

Move/Modify Python Interface Files

(3) Move C++/Python interface functions

`clib/src/ctonedim_methods.cpp` to

`python/src/ctonedim_methods.cpp`

*additional editing must be done for functions with arrary argument

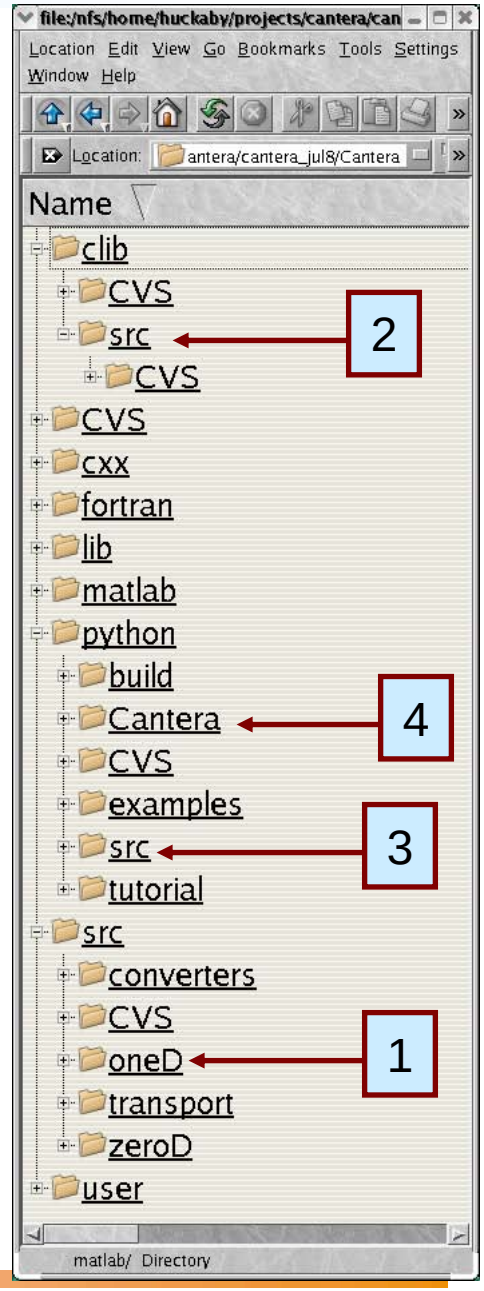
Append function prototypes in `clib/src/pyctonedim_methods.h` to

`python/src/methods.h`

(4) Move Python wrapper class definitions

`clib/src/ctoedim.py` to

`python/Cantera/Onedim`



Python/C++ Interface

All classes have an integer index to the C++ object

Python:

```
def current(self):
        return _cantera.inletEfield_current(self._hndl)
```

C++:

```
static PyObject *
py_inletEfield_current(PyObject *self, PyObject *args)
{
    double _val;
    int i;
    if (!PyArg_ParseTuple(args, "i:inletEfield_current", &i))
        return NULL;

    _val = inletEfield_current(i);
    if (int(_val) == -1) return reportCanteraError();
    return Py_BuildValue("d",_val);
}
```

The index in the objects “cabinet” is the first python argument

Python/C++ Interface

```
def setMassFractions(self, y):
        return _cantera.outletEfield_setMassFractions(\
             self._hndl, y, len(y) )
```

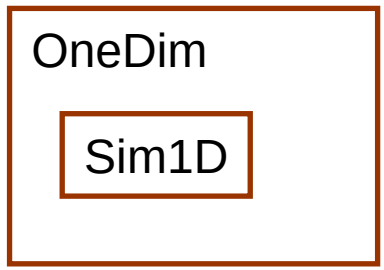
```
static PyObject *
py_outletEfield_setMassFractions(PyObject *self, PyObject *args)
{
    int _val;
    int n;
    int i;
    PyObject *px;
    if (!PyArg_ParseTuple(args,
       "iOi:outletEfield_setMassFractions", &i, &px, &n))
        return NULL;
    PyArrayObject* y_array  = (PyArrayObject*) px;
    double* y = (double*) y_array->data;
    _val = outletEfield_setMassFractions(i,y,n);
    if (int(_val) == -1) return reportCanteraError();
    return Py_BuildValue("i",_val);
}
```

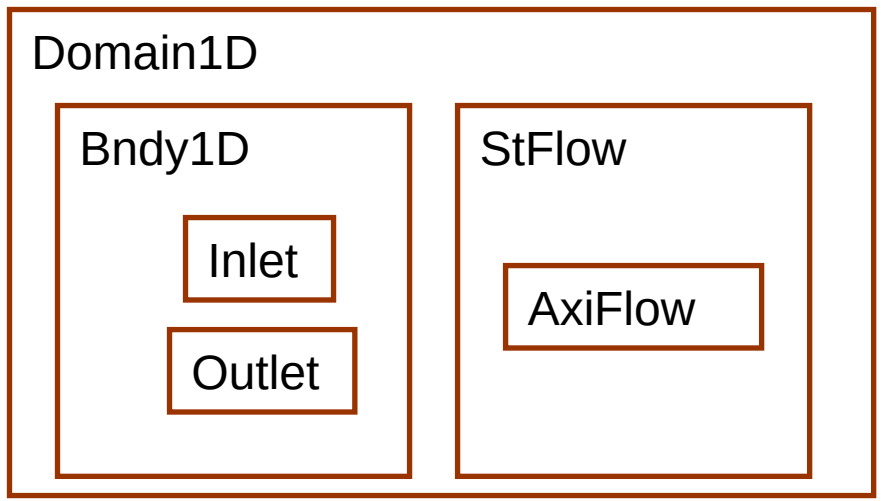
"O" stands for user defined type object argument (an arra

C++ Classes used for Flames

Sim1D

- "drives" the solution process
- No physical processes
- Non-linear equation solver
- Contains:
  - vector of Domain1D's
  - solution vector

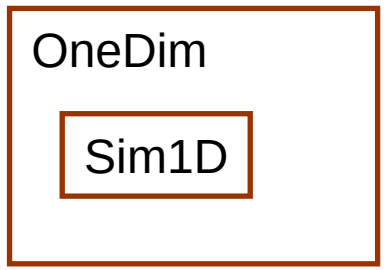




C++ Classes used for Flames

Domain1D

- **Contains Physical Processes**
- **`eval()` – calculate "residual"**

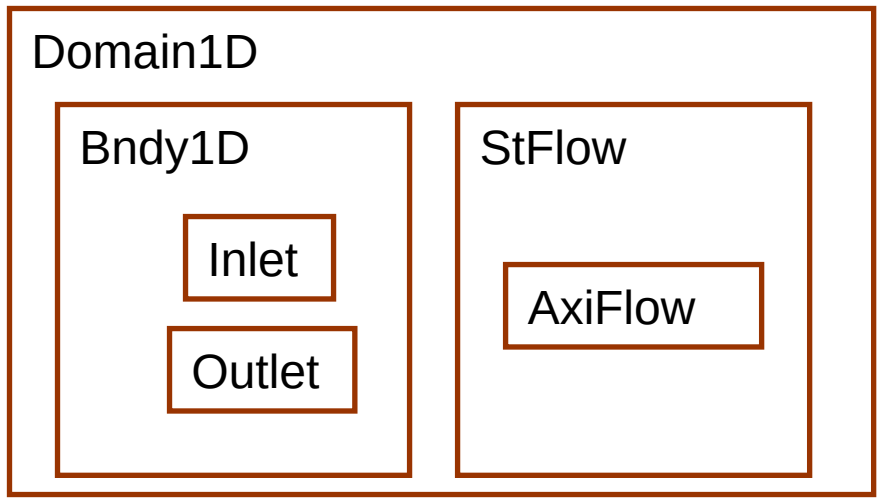


Bndry1D

- **Discretize B.C.'s**
- **May modify "residual" of an adjacent Flow Object**

StFlow

- **Discretize Cons. Eq.'s**
- **Constitutive Relations**

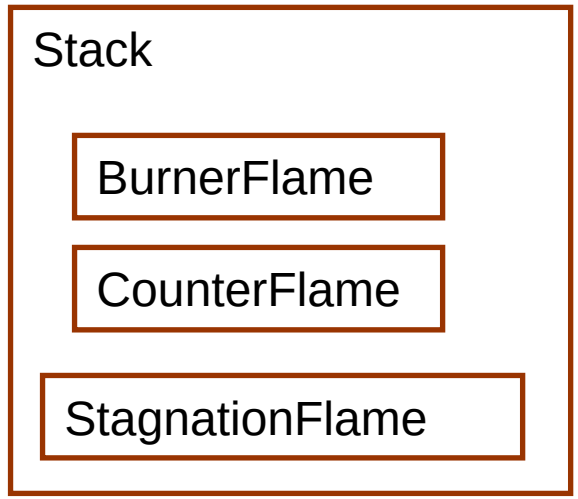


Python Classes used for Flames

Flame class

- **Shadow Sim1D**
- **Specialized for specific type of simulation**

Other classes shadow analogous C++ ver.



Burner Flame

Inlet - AxiFlow - Outlet

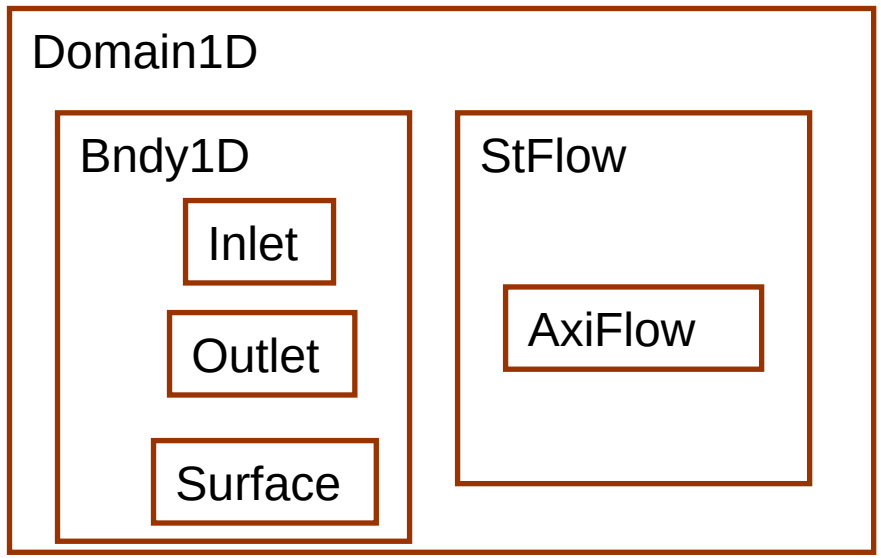
Counter Flame

Inlet - AxiFlow - Inlet

Stagnation Flame

Inlet - AxiFlow - Wall

Inlet - AxiFlow - Reacting Surface



The solution vector

The Sim1D solution vector is a double <vector> which contains the unknowns for all the components of all the domains which are contained in the instance of the particular Sim1D object

Ordering:

solution vector of a specific domain is contiguous in memory

the solution components at grid location are contiguous

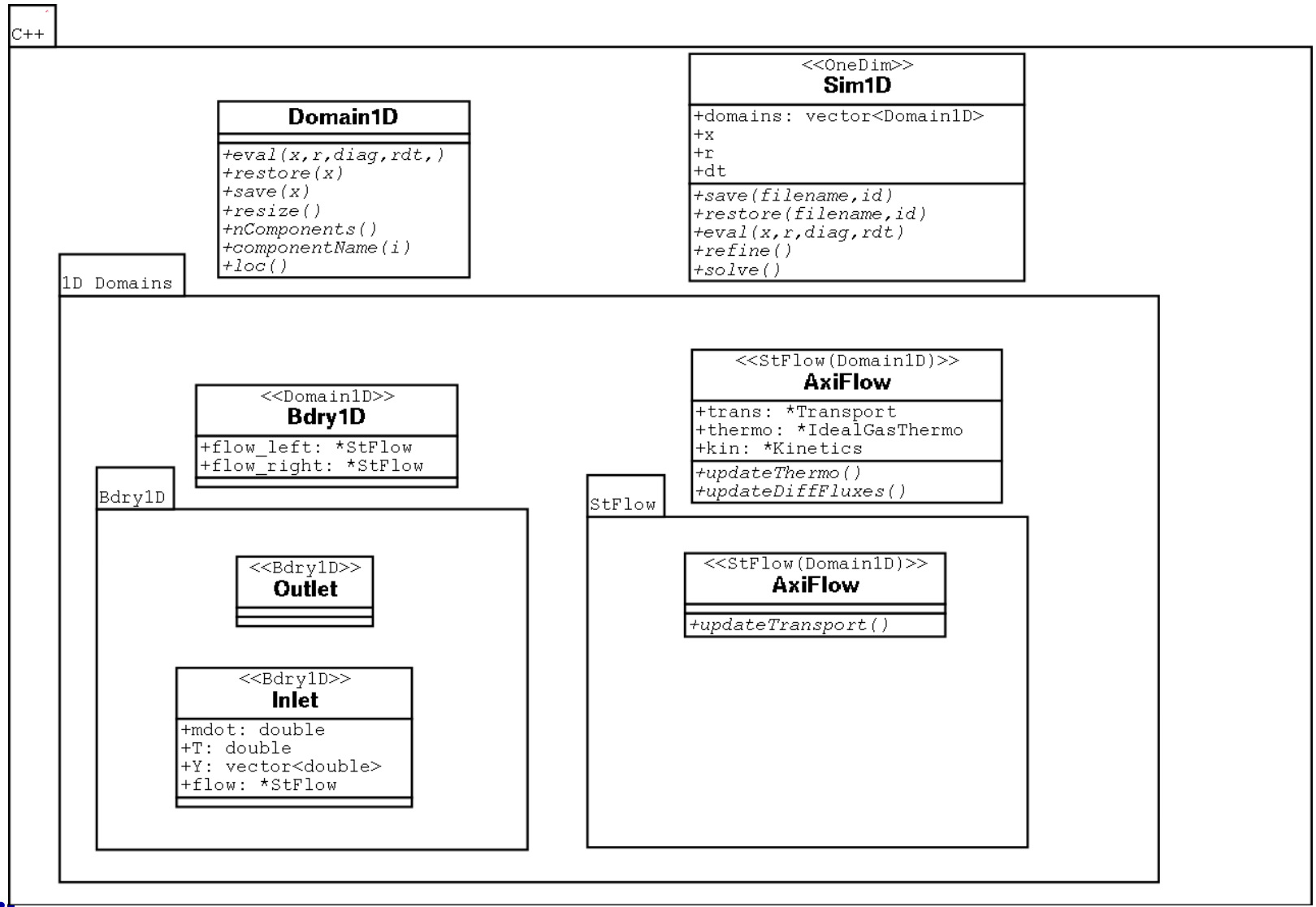
Burner Flame

$$\left(\dot{m}'' \quad T\right) \left(u \quad V \quad T \quad \Lambda \quad Y_i\right)_0 \quad \cdots \quad \left(u \quad V \quad T \quad \Lambda \quad Y_i\right)_{N-1} \left(s\right)$$

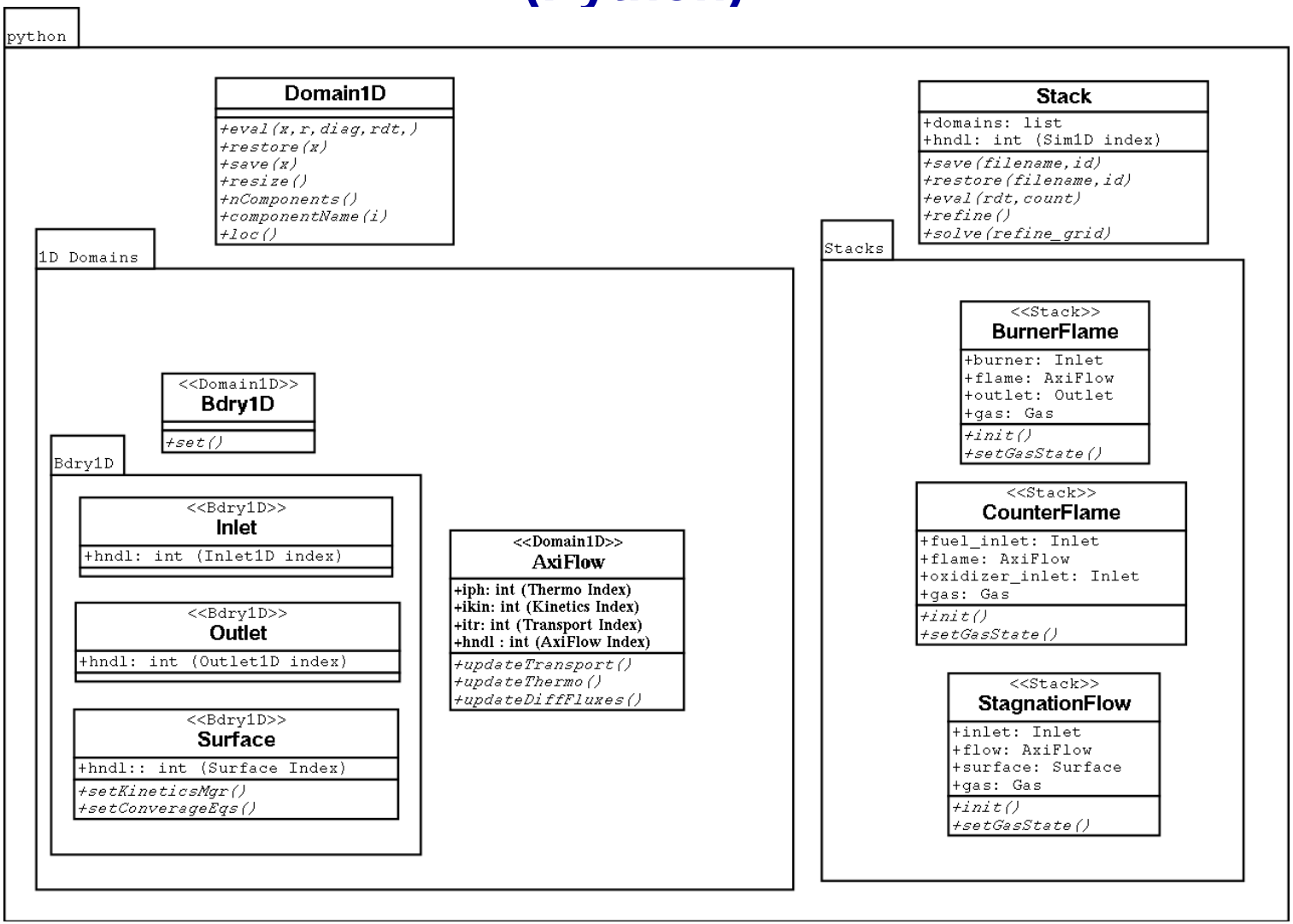
Counter Flame

$$\left(\dot{m}'' \quad T\right) \left(u \quad V \quad T \quad \Lambda \quad Y_i\right)_0 \quad \cdots \quad \left(u \quad V \quad T \quad \Lambda \quad Y_i\right)_{N-1} \left(\dot{m}'' \quad T\right)$$

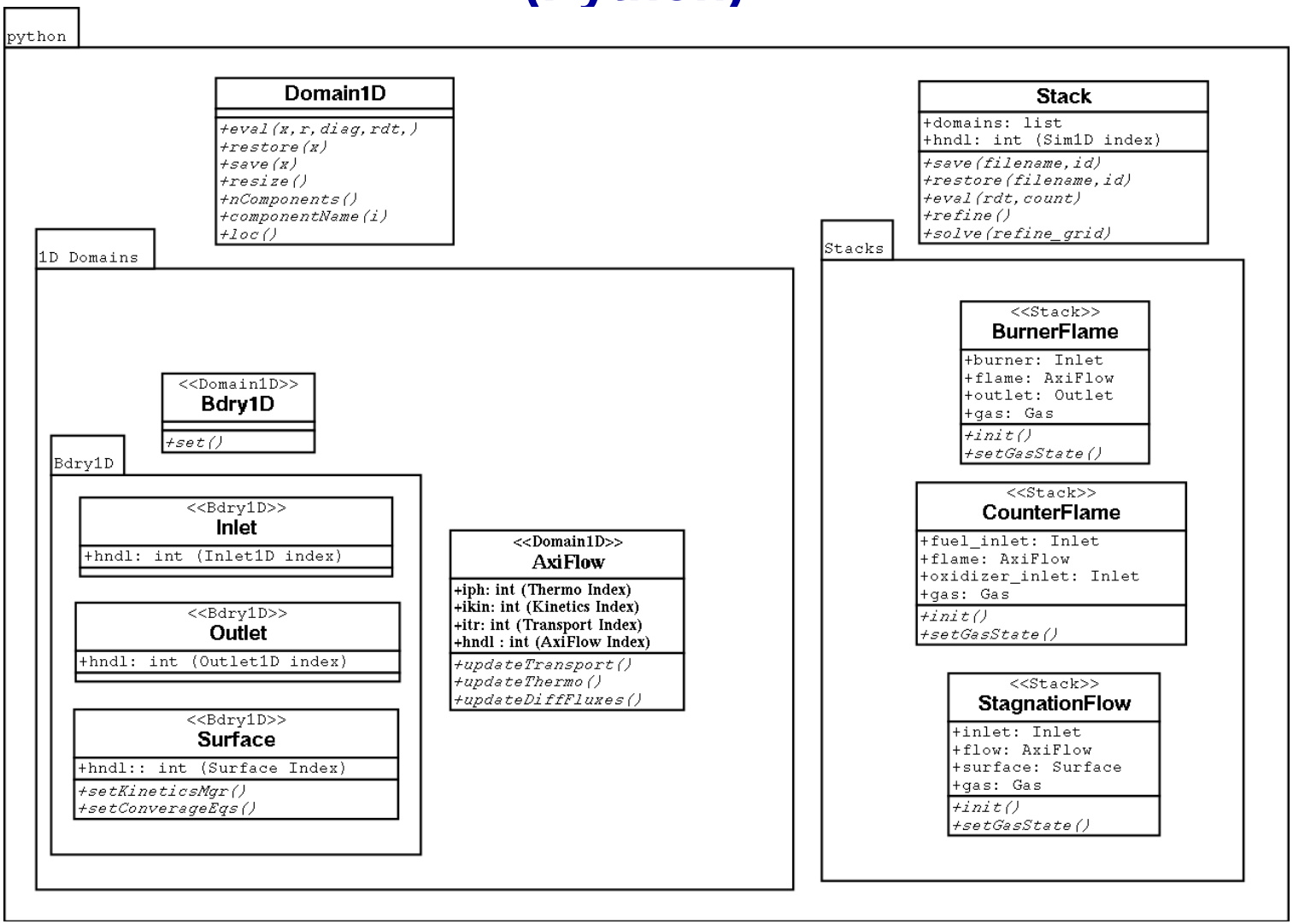
C++ Objects used for Flames



Structure of a Flame Object (Python)



Structure of a Flame Object (Python)



Outline

I. How to customize the python interface

II. Structure of C++ and Python Flame Objects

III. Specific Modifications for the electrostatic interaction with flames

Governing Equations

(Goodwin 2004, Kee, Coltrin Glarborg 2003, Smooke and Giovangigli 1991, Penderson and Brown 1993)

.**Standard (Premixed)**:

- Mass
- Species
- Energy

$$\frac{d}{dx}(\rho u) = 0$$

$$\rho u \frac{dY_i}{dx} + \frac{d}{dx} J_i = \dot{\omega}_i W_i$$

$$\rho u c_p \frac{dT}{dx} + \frac{dq}{dx} + \left(\sum_i J_i c_{p,i} \right) \frac{dT}{dx} = -\sum_i \dot{\omega}_i h_i W_i$$

Add:

- Gauss's Law
- Drift Flux
- Electron Mobility Model*

$$\varepsilon_0 \frac{dE_x}{dx} = (N_A e) \sum_i z_i \frac{\rho Y_i}{W_i}$$

$$J_i = \underline{\rho Y_i z_i \mu_i E_x} - D_{m,i} \frac{W_i}{W_m} \frac{dX_i}{dx}$$

Specialize Base Classes

1. StFlowEfield (StFlow)
    1. Additional solution variable, ϕ
    2. Drift Flux as part of species flux
    3. Electron Transport Model
2. InletEfield (Inlet)
    1. Electrical b.c.'s
3. OutletEfield (Outlet)
    1. Electrical b.c.'s
    2. Other b.c.'s for Y

Flat Flame Burner – Boundary Conditions

Outlet

$$\frac{dT}{dx} = 0$$

Charged Species Fraction

$$\frac{dY_i}{dx} = 0$$

Potential

$$Y_i = 0$$

$$\phi = \phi_{burner}$$

Inlet

$$T = T_{burner}$$

Temperature

$$\rho u = \dot{m}''_{in}$$

Flow rate

$$\rho u Y_i + J_i = \dot{m}''_{in} Y_{in,i}$$

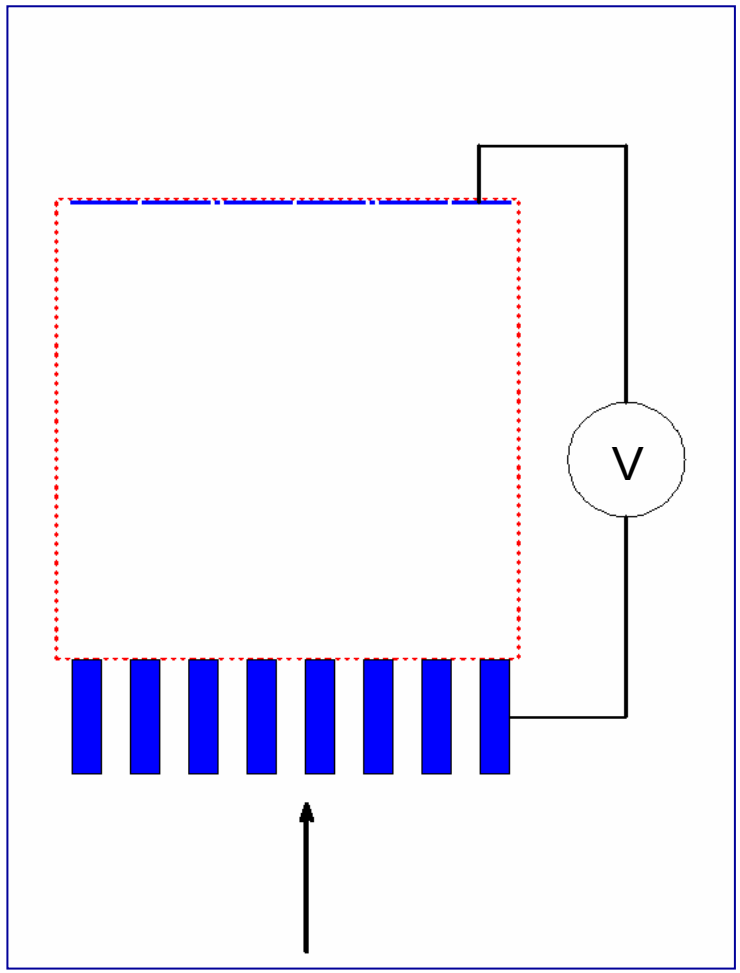
Neutral Flux

$$Y_i = 0$$

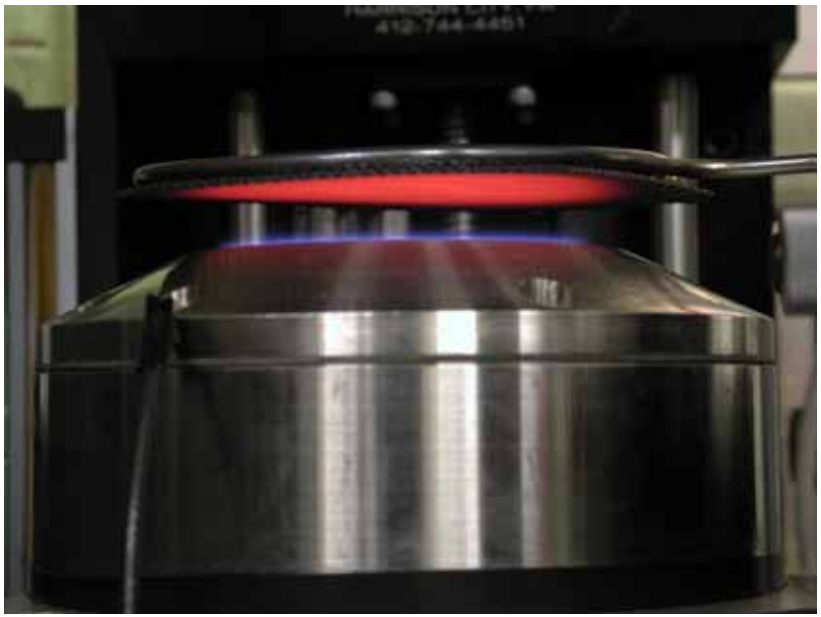
Charged Species Fraction

$$\phi = \phi_{burner}$$

Potential

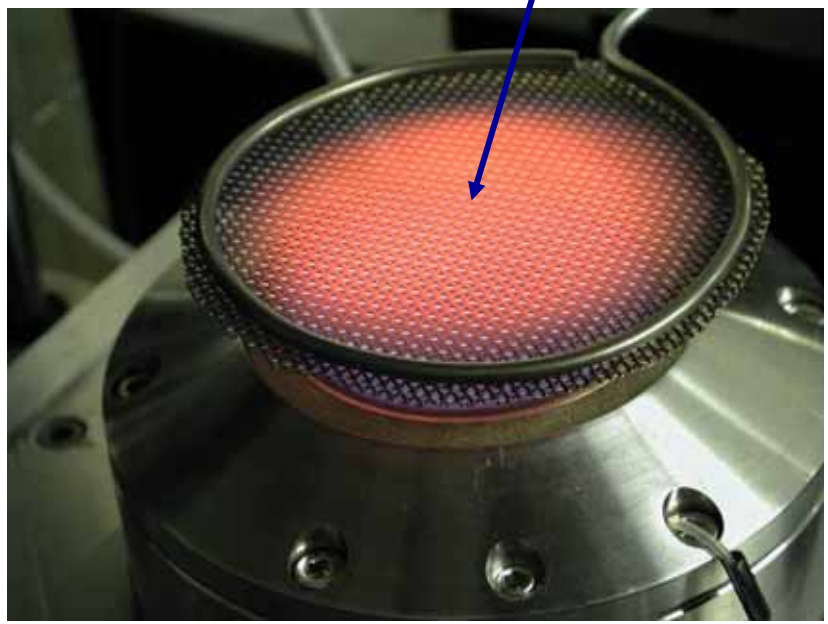


Flat Flame Burner – Apparatus



Downstream Electrode

Number of Openings	16/in x 16/in
Opening Size (Square)	0.040“
Wire Diameter	0.023”
Open Area (Percentage):	39.9 %
Distance from Burner	2 cm



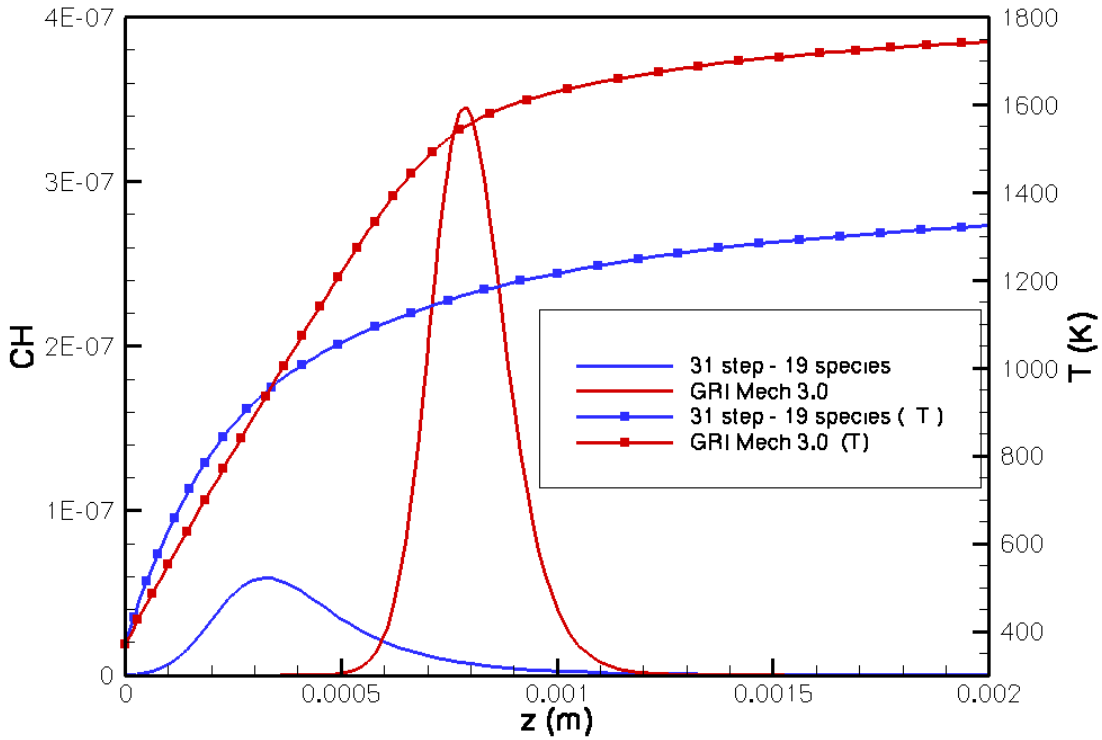
Burner (Ground)

Burner Diameter	60 mm
Housing Diameter	120 mm
*Nitrogen co-flow	

McKenna (www.flatflame.com)

Flat Flame Burner Results

Compare to GRI 3.0

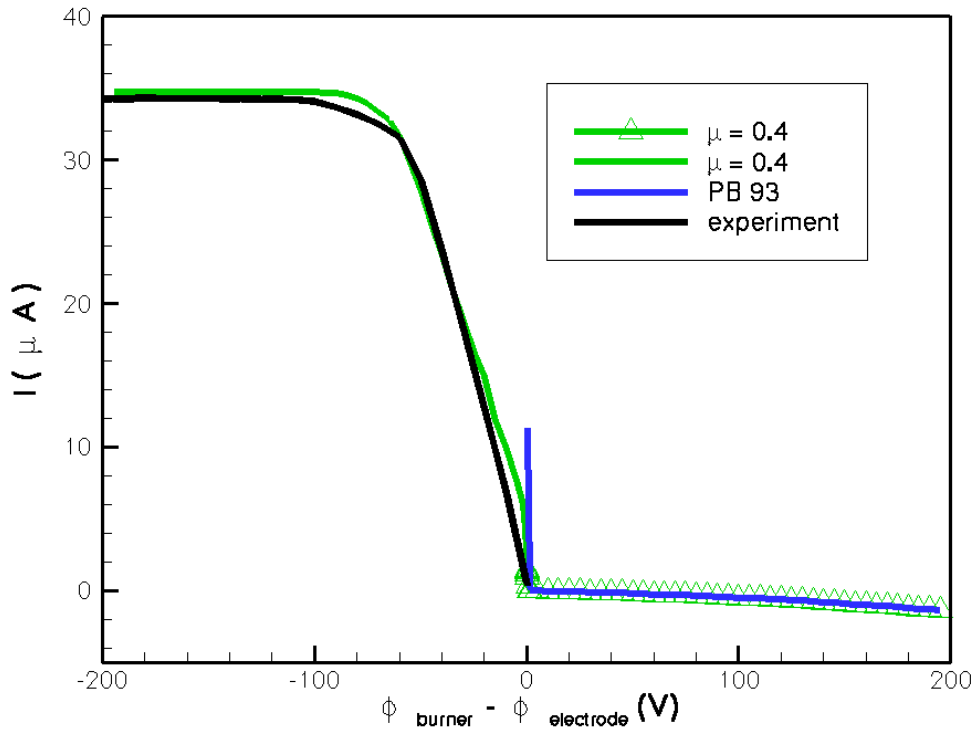


Based on CH mole fraction profile the 31 Step mechanism predicts a different flame position than the full GRI 3.0.

Temperature profile shows different amount of heat transferred to the burner.

All species profiles are mole fractions

Flat Flame Burner Results



Comparison is good with μ=0.4, despite the poor prediction of the temperature profile (flame position).

Several Simulation are performed at different applied potentials, and Different mobilities (μ)

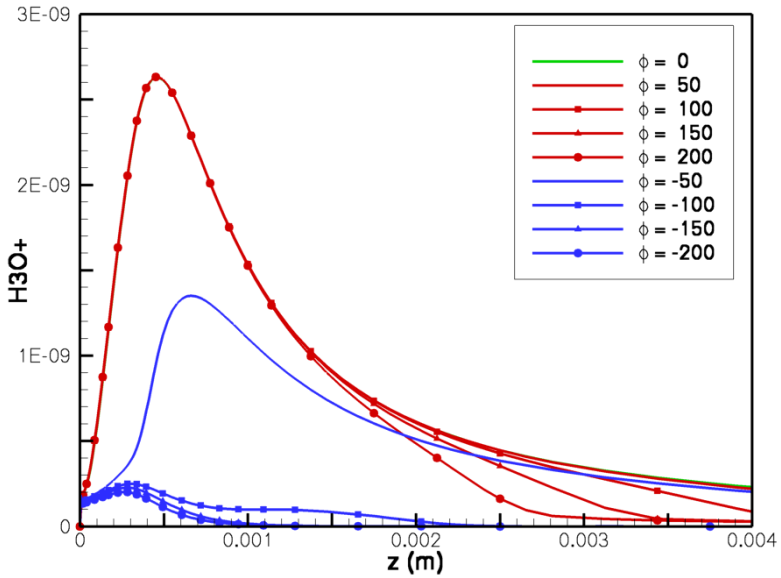
PB93 – Penderson and Brown (1993)

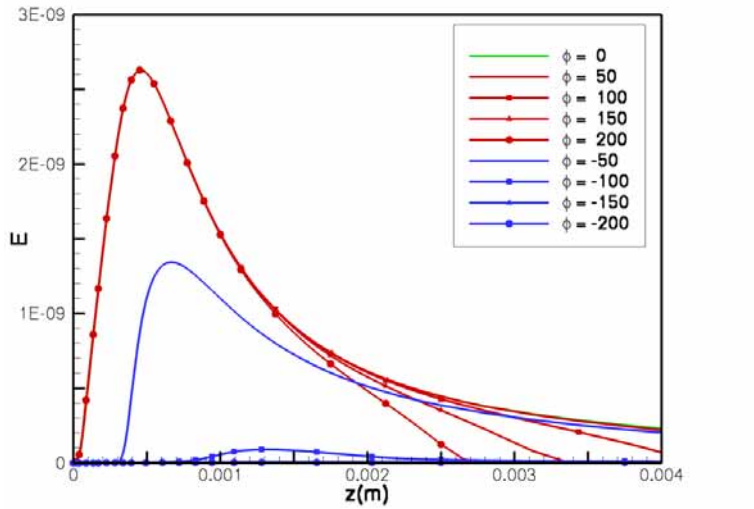
μ=0.4 – Goodings et. al (2001)

Burner attracts ions

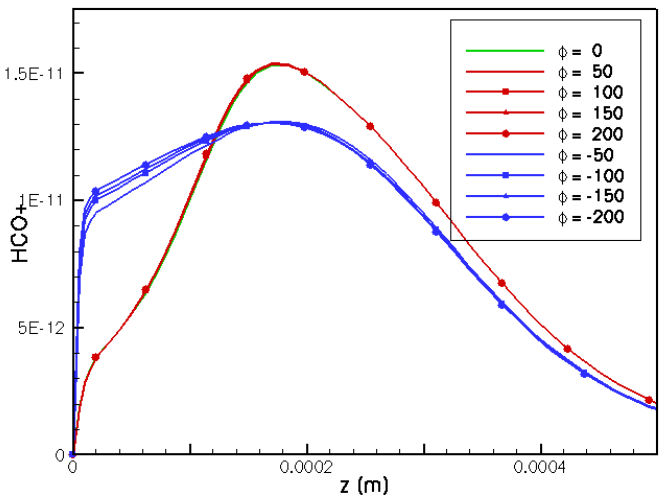
Burner attracts electrons

Flat Flame Burner Results





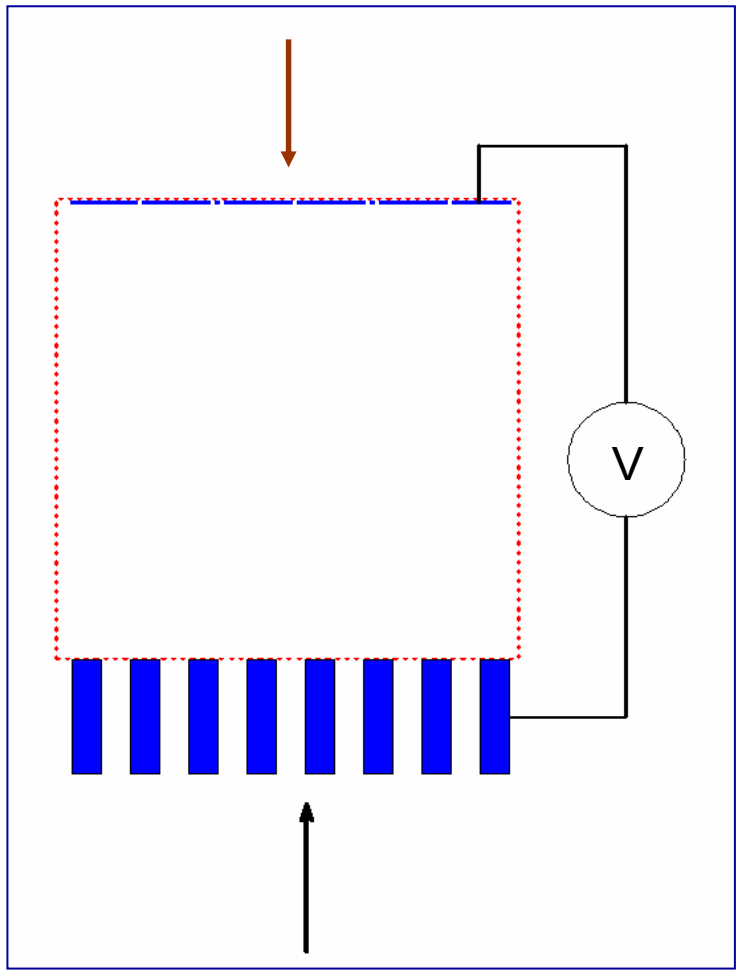
Depletion of charged species at increased voltage



Opposed Flow Burner – Boundary Conditions

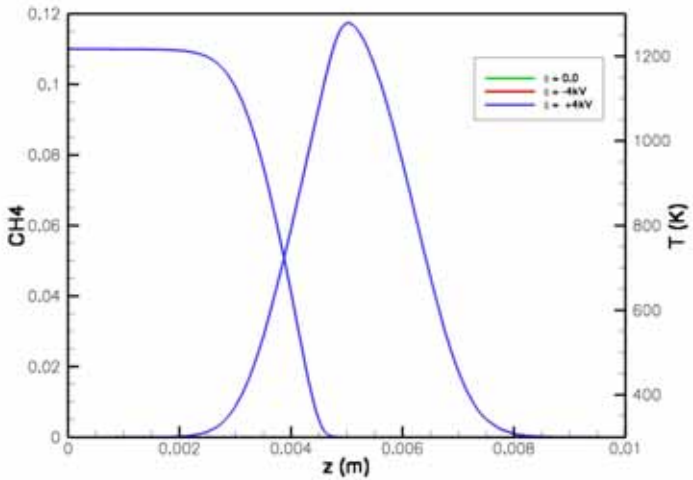
Oxidizer or Fuel Inlet

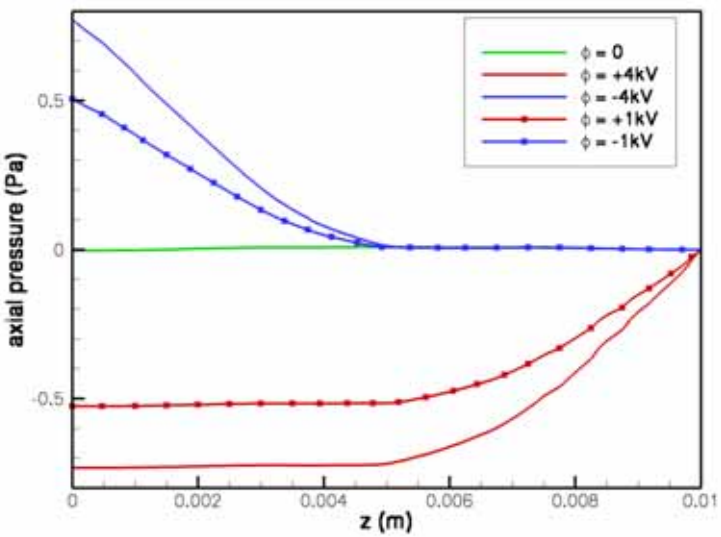
$T = T_{electrode}$	Temperature
$\rho u = \dot{m}''_{in}$	Flow rate
$\rho u Y_i + J_i = \dot{m}''_{in} Y_{in,i}$	Neutral Flux
$Y_i = 0$	Charged Species Fraction
$\phi = \phi_{electrode}$	Potential



Opposed Flow Flame

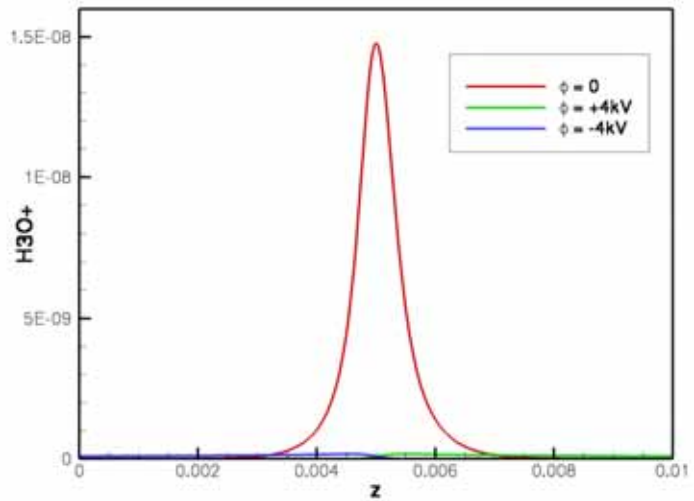
The flame position has not changed due to the electric field. An additional mechanism not included in the equations must be considered (multidimensional effect, non-uniquenes ?).

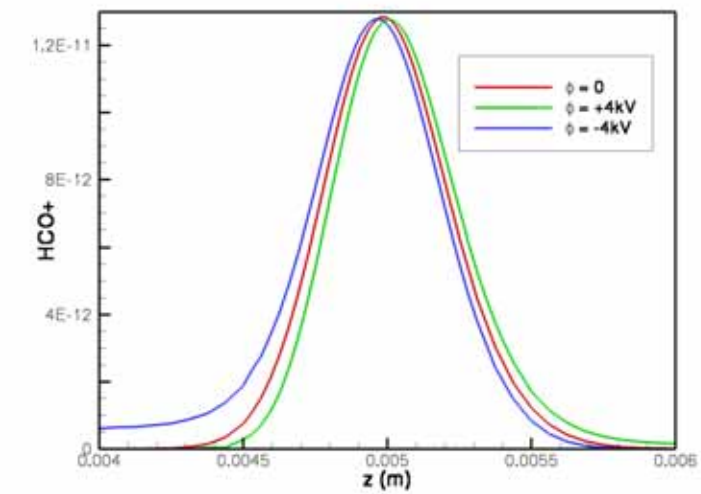




There is a solution to the equations without the axial momentum equation, so the axial pressure simply balances the other terms in the equation without modification to the other variables in the system

Opposed Flow Flame



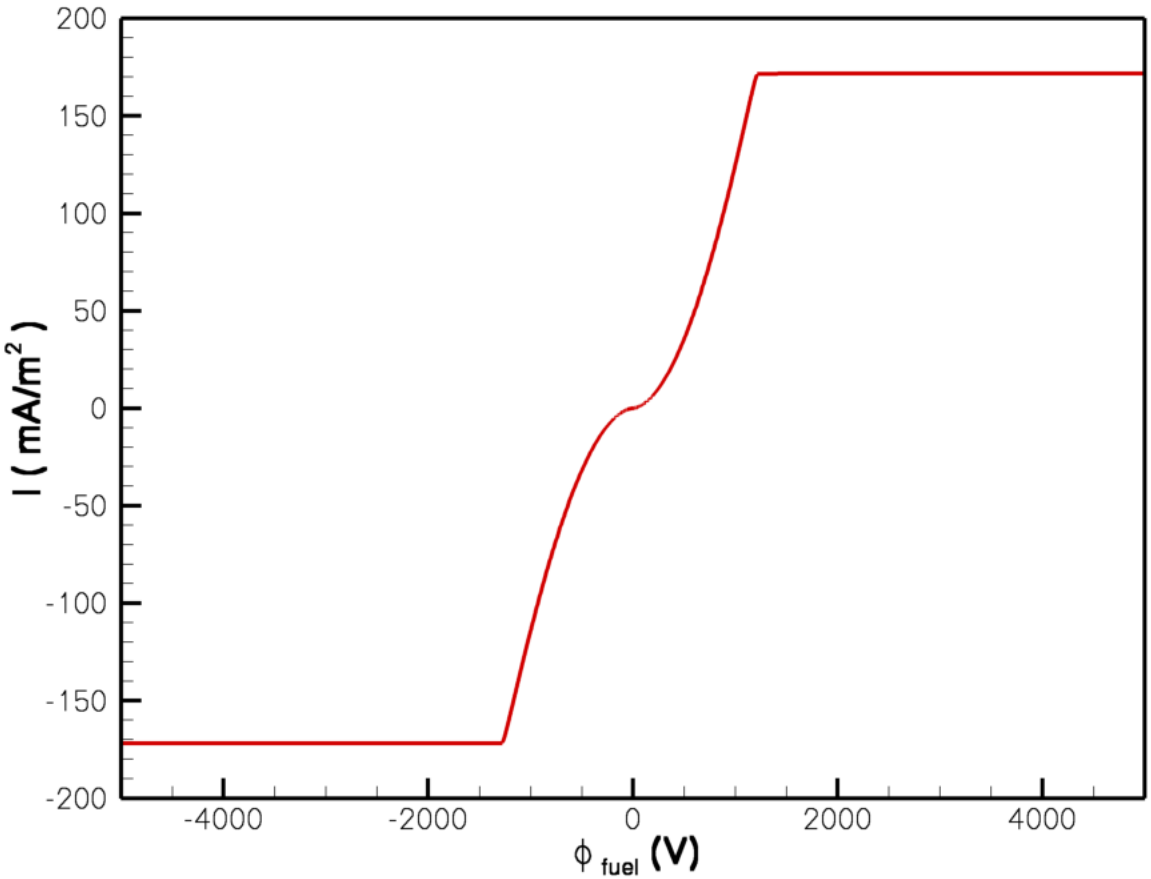


The mole-fraction of H3O+ has decreased about 2 orders of magnitude after the application of a strong electric field.

Whereas the electron mole-fraction decreases about 4 orders of magnitude.

The mole fraction of HCO+ is relatively constant, with a slight shift in position due to the field

Opposed Flow Flame



In contrast to the flat flame, the current voltage behavior is nearly symmetric

References

Software:

Goodwin, D., (2004), "Cantera: Object-Oriented Software for Reacting Flows", www.cantera.org.

Computational Models:

Penderson, T. and Brown, R. C., "Simulation of Electric Field Effects in Premixed Methane Flames", Combustion and Flame 94:433-448. (1993)

Jones, F. L. and Becker, P. M., 1972, "A Mathematical Model of the Opposed-Jet Diffusion Flame: Effect of an Electric Field on Concentration and Temperature Profiles", Combustion and Flame, 19, 351

Hu, J., Rivin, B., and Sher, E., (2000), "The effect of an electric field on the shape of co-flowing and candle-type methane-air flames", *Experimental Thermal and Fluid Science*, vol.21, pp.124-133.

Ionization in Turbine Combustors:

Straub, D. L., Thornton, J. T., Chorpening, B. T., Richards, G. A. (2002). In-Situ Flame Ionization Measurements In Lean Premixed Natural Gas Combustion Systems, presented at the Western States Section/Combustion Institute Spring Meeting, San Diego, CA, March 25-26.

Thornton, J.D., Richards, G.A., and Robey, E. (2000), Detecting Flashback in Premix Combustion Systems, presented at the American Flame Research Committee International Symposium, Newport Beach, California, September 17-21.

Thornton, J.D., Straub, D. L., Richards, G.A., Nutter, R. S., Robey, E., (2001)."An In-Situ Monitoring Technique for Control and Diagnostics of Natural Gas Combustion Systems," presented at the 2nd Joint Meeting of the U.S. Sections of the Combustion Institute, Oakland, CA, March 25-28.

Jimmy D. Thornton, Douglas L. Straub, Benjamin T. Chorpening, E. David Huckaby, and Geo. A. Richards and Bensen, K. (2004), "A Combustion Control and Diagnostics Sensor for Gas Turbines", Proceedings of 2004 ASME/IGTI TurboExpo Meeting - TurboExpo Power For Land, Sea, and AirJune 14-17, 2004, Vienna, Austria, ASME Paper GT2004-53392.

B. T. Chorpening, J. D. Thornton, E. D. Huckaby and K. Bensen, (2004), "Combustion Oscillation Monitoring Using Flame Ionization in a Turbulent Premixed Combustor", Proceedings of ASME Turbo Expo 2004 Power for Land, Sea, and Air June 14–17, 2004, Vienna, Austria.

Experimental results:

Calcote, H. F., Kurius, S. C. and Miller, W. J., 1965, "Negative and Secondary Ion Formation in Low-Pressure Flames", 10th Symposium of Combustion, pp. 605-619.

Goodings, J. M., Bohme, D. K. and Sugden, T.M., 1977, "Positive Ion Probe of Methane-Oxygen Combustion", 16th Symposium on Combustion, pp. 891-901.

Goodings, J. M., Bohme, D. K. and Sugden, T.M., 1979, "Detailed Ion Chemisty in Methane-Oxygen Flames I - Positive Ions", Combustion and Flame 36:45-62.

Goodings, J. M., Bohme, D. K. and Sugden, T.M., 1979, "Detailed Ion Chemisty in Methane-Oxygen Flames II - Negative Ions", Combustion and Flame 36:45-62.

Goodings, J. M., Guo, J., Hayhurst, A. N. and Taylor, 2001, "Current-voltage characteristics in a flame plasma: analysis for positive and negative ions, with applications", International Journal of Mass Spectrometry 206:137-151.

Axford, S. D. T and Hayhurst, A. N., 1995, "Ionisation in Premixed Fuel-lean Flames of H2, O2 and N2, Part 1. – Naturally Occurring Positive Ions", Journal of the Chemical Society - Faraday Transactions 91(5), pp.827-833

Background:

Calcote, H. F., "Electrical properties of flames: burner flames in transverse electric fields," The Third Symposium on Combustion, Flame, and Explosion Phenomena, pp. 245-253, 1949.

Calcote, H. F., (?) "Ionization in Hydrocarbon Flames", Report, Aerochem Research Laboratories, pp.1 – 42.

Fialkov, A.B., "Investigations on Ions in Flames." Progress in Energy Combust. Science, v23, 1997, pp. 399-528.

Holm, T., 1999, "Aspects of the Mechanism of the Flame Ionization Detector." Journal of Chromatography A, v842, pp. 221-227.

Numerical Methods:

Kee, J.K., Coltrin M.E. and Glarborg, R., 2003, Chemically Reacting Flow: Theory and Practice, Wiley-Interscience.

Ferziger and Peric, 1996, Computational Methods for Fluid Dynamics, Springer.