FlameMaster Flame Calculator

Combustion Summer School

2018

Prof. Dr.-Ing. Heinz Pitsch
Course Overview

Part I: Fundamentals and Laminar Flames

• Introduction
• Fundamentals and mass balances of combustion systems
• Thermodynamics, flame temperature, and equilibrium
• Governing equations
• Laminar premixed flames: Kinematics and Burning Velocity
• Laminar premixed flames: Flame structure
• Laminar diffusion flames
• FlameMaster flame calculator
• Introduction
• Counterflow diffusion flame
• Flamelet structure of diffusion flames
  • FlameMaster flame calculator
  • Single droplet combustion
FlameMaster Flame Calculator

- FlameMaster: A C++ Computer Program for 0D Combustion and 1D Laminar Flame Calculations
  - Premixed and non-premixed
  - Steady and unsteady
  - Emphasis on pre- and post-processing
    - Sensitivity analysis
    - Reaction flux analysis

- At request, available online at https://itv.rwth-aachen.de/index.php?id=13
FlameMaster Flame Calculator

- Example: Shock tube, homogeneous reactor

N-Octanol Ignition Delay Times

Methyl-cyclohexane species time histories in shock tube
Example: Flow reactor

Methyl-cyclohexane species time histories in constant pressure plug flow reactor
FlameMaster Flame Calculator

- Example: Jet stirred reactor

N-Dodecane oxidation in jet stirred reactor

(a) Fuel, $n$-$C_{12}H_{26}$

(b) $H_2O$, $O_2$

(c) $CO_x$
FlameMaster Flame Calculator

- Example: Reaction flux analysis
Example: Laminar burning velocities
FlameMaster Flame Calculator

- Example: Premixed flame structure

Methyl-cyclohexane species profiles in premixed burner stabilized flame
FlameMaster Flame Calculator

- Example: Flamelet libraries

Flamelet library for methane/air non-premixed combustion

![Diagram](image-url)
FlameMaster Tutorial

• Distribution and installation
• Code structure
• Pre-processing
• Examples:
  - Homogeneous reactor
  - Premixed flames
  - Diffusion flames
  - Unsteady flamelet
FlameMaster Tutorial: Distribution and Installation

- Distribution for Linux, MacOS, and Windows systems
- Prerequisites
  1. C and C++ compiler (C11/C++11 compliant)
  2. CMake
  3. Flex, Bison
  4. Sundials (optionally installed by FlameMaster)
- Installation on Linux/MacOS
  ```
tar -xf FlameMaster.zip
cd FlameMaster
mkdir Build && cd Build
cmake ../Repository -DCMAKE_BUILD_TYPE=Release -DINSTALL_SUNDIALS=ON
make install -j4
```
- Set up environment
  ```
cd .. && source Bin/bin/Source.bash
```
- Create Mechanisms
  ```
cd Run/ScanMan && bash CreateAllMechanisms.bash
```

Further information:
Repository/README.md
Operating System specific instructions:
Repository/doc/
Consider using your package manager
FlameMaster Tutorial: Installation on Windows I

- Multiple options
- Avoid spaces or special characters in directories and file names
- Option 1: Install Cygwin including all prerequisites (http://cygwin.org/)
- Option 2: Install Linux in a Virtual Box (https://www.virtualbox.org/)

Option 1/2 are the recommended options
Option 3: Install FlameMaster natively without Cygwin
  - Install only Compilers, CMake, and Python
  - FlameMaster installs flex, bison, and sundials
  - During the installation of CMake and Python
    - Choose to add the executables to the PATH
  - Extract FlameMaster to C:\FlameMaster
  - Commands for the installation
    cd C:\FlameMaster
    mkdir Build && cd Build
    cmake ..\Repository -DCMAKE_BUILD_TYPE=Release
    -DINSTALL_SUNDIALS=ON
    cmake --build . --config Release --target install

Option 4: Install FlameMaster from a binary installer
  - Install FlameMaster to C:\FlameMaster

Set up environment
  - Double click: C:\FlameMaster\Bin\bin\Source.bat

Create Mechanisms
  cd Run\ScanMan && CreateAllMechanisms.bat

Option 3/4 are in beta
Possible issues:
Fixing requires solid understanding of C++ and CMake
Further instructions:
Repository/doc/Windows.md
**FlameMaster Tutorial: Directory layout**

<table>
<thead>
<tr>
<th>Directory</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository/</td>
<td>Permanent directory for the source code. Do all changes here.</td>
</tr>
<tr>
<td>Build/*</td>
<td>Intermediate directory containing configuration information (&quot;CMake cache&quot;) and object files</td>
</tr>
<tr>
<td>Bin/*</td>
<td>Installation directory containing binaries and scripts that set up your environment (for bash, csh, zsh, bat)</td>
</tr>
<tr>
<td>Run/*</td>
<td>Directory containing examples</td>
</tr>
<tr>
<td>Data/*</td>
<td>Directory checked by ScanMan and FlameMan via the shell variable FM_DATA to find binary input files</td>
</tr>
</tbody>
</table>

*Do not modify automatically generated files. Changes are overwritten if you rerun the installation. Instead, modify the respective template files in Repository/*
FlameMaster Tutorial: Code Structure

- Four different parts (main executables):
  1. Preprocessing of thermo- and transport data: CreateBinFile
  2. Preprocessing of chemical mechanism: ScanMan
  3. Simulation: FlameMan
  4. Post-processing tools: e.g. ListTool

- All tools have help function: Type <Executable> -h

- Main input files
  1. Mechanism
     - Thermodata file
     - Transport data file
     - Mechanism file
  2. FlameMaster input file
     - Information about configuration, boundary and initial conditions, post-processing options, ...
FlameMaster Tutorial: Code Structure

- Four different parts (main executables):
  1. Preprocessing of thermo- and transport data: \textit{CreateBinFile}
  2. Preprocessing of chemical mechanism: \textit{ScanMan}
  3. Simulation: \textit{FlameMan}
  4. Post-processing tools: e.g. \textit{ListTool}

\begin{tikzpicture}
  \node (thermodata) {thermodata [\.therm]};
  \node (transportdata) [below of=thermodata] {transportdata [\.trans]};
  \node (createbin) [below of=transportdata] {CreateBinFile};
  \node (thermodatabin) [below of=createbin] {thermodata-binary [\.bin]};
  \node (mech) [right of=thermodatabin] {mechanism [\.mech]};
  \node (scman) [below of=mech] {ScanMan};
  \node (flm) [below of=scman] {FlameMaster};
  \node (outfile) [below of=flm] {outputfile(s) (e.g. CH4.p01...)};
  \node (tools) [below of=outfile] {Tools (e.g. ListTool)};
  \node (formatted) [below of=tools] {outputfile formatted [\.kg]};

  \draw[->] (thermodata) -- (transportdata);
  \draw[->] (transportdata) -- (createbin);
  \draw[->] (createbin) -- (thermodatabin);
  \draw[->] (thermodatabin) -- (mech);
  \draw[->] (mech) -- (scman);
  \draw[->] (scman) -- (flm);
  \draw[->] (flm) -- (outfile);
  \draw[->] (outfile) -- (tools);
  \draw[->] (tools) -- (formatted);
\end{tikzpicture}
CreateBinFile

- Creates a binary file `thermotransout` to be used by ScanMan including all thermodata and transport data
- Input files are thermodata file and transport data file
- Reads Chemkin II format

Syntax:
```
CreateBinFile -i <thermo> -m <transport> -o <thermotransout>
```

Example:
```
cd Run/ScanMan/GRI3.0
CreateBinFile -i gri.30.therm -m gri.30.trans -o gri.30.bin
```
ScanMan

- Creates a binary file to be used by FlameMan, including all thermodata, transport data, and reaction mechanism data <mechanism>.mech

- Input files are binary file thermotransout created by CreateBinFile and reaction mechanism file

- ScanMan has its own format for mechanism file, but also reads Chemkin II format

- Resulting output file should typically be copied to directory Data

Example: Mechanism file

```
H2/CO Chemistry from
"An optimized kinetic model of H2/CO combustion"
Davis, Joshi, Wang, and Erofopoulos

# Reactions of H2/O2
1f: H + O2 -> O + OH { a = 1.040E+14 n = 0.000 E = 63.957 }
2f: O + H2 -> H + OH { a = 3.818E+12 n = 0.000 E = 33.254 }
3f: O + H2 -> H + OH { a = 8.792E+14 n = 0.000 E = 80.287 }
4f: H2 + OH -> H2O + H { a = 2.168E+08 n = 1.510 E = 14.351 }
5f: OH + OH -> O + H2O { a = 3.340E+04 n = 2.420 E = -8.075 }
6f: H2 + M41 -> H + H + M41 { a = 4.577E+19 n = -1.448 E = 436.726 }
7f: H2 + AR -> H + H + AR { a = 5.840E+18 n = -1.108 E = 436.726 }
8f: H2 + HE -> H + H + HE { a = 5.840E+18 n = -1.108 E = 436.726 }
```
FlameMaster Tutorial: Proproc. Mechanism File

- **ScanMan (Mechanism file in FlameMaster format)**
  
  - **Syntax:**
    ```
    ScanMan -i <Mechanismfile>.mech -t <thermotransportoutput> <mechanismbinfile>.pre
    ```
  
  - **Example:**
    ```
    cd Run/ScanMan/GRI3.0
    ScanMan -i gri.30.mech -t gri.30.bin -rs3 > gri.out
    mv gri.30.pre ../../../Data
    ```

- **ScanMan (Mechanism file in Chemkin format)**

  - **CreateBinFile not required**
  
  - **Syntax:**
    ```
    ScanMan -i <Mechanismfile>.chmech -t <thermodata> -m <transportdata> -f chemkin
    ```
  
  - **Example:**
    ```
    cd Run/ScanMan/GRI3.0_CK
    ScanMan -i grimech30.dat -t thermo30.dat -m transport.dat -f chemkin -rs3 > gri.out
    ```
- FlameMan performs simulations for different configurations

- Main input file called FlameMaster.input (default name) or name specified on command line using -i option

- All parameters and inputs are provided in FlameMaster.input file

- Some parameters can be specified by command line options, which overwrite input file values

Example: FlameMaster input file
Typical input file (part 1)

- Comments start with #
- Typical statement is `<Keyword> = <value>`
- `<Keyword>` is `<value>`
- Most `<Keywords>` have default values, usually `FALSE`, if applicable

- I/O
  - Outputpath

- Chemistry
  - Mechanism input file
  - Global reaction (for equivalence ratio)
  - Fuel and oxidizer (for equivalence ratio)

Example: FlameMaster input file

```plaintext
#############
# I/O #
#############

WriteEverySolution is TRUE
OutputPath is ./Output
NOutputs = 50

#############
# Chemistry #
#############

MechanismFile is Base.pre
globalReaction is 2H2 + O2 == 2H2O ;

fuel is H2
oxidizer is o2

Example: FlameMaster input file
```
Typical input file (part 2)

- Configuration
  - Flame is ...
  - \( \phi \) is equivalence ratio
  - Several \( \phi \) and pressure can be specified
  - If no \( \phi \) is specified, it will be computed from initial composition

```plaintext
# Flame #

Flame is Isochor Homo Reactor
#Flame is Isobar Homo Reactor

#phi = 0.5
phi = 1.0
#phi = 2.0

Pressure = 13e5
Pressure = 1e5

ContInc = -0.1
ContType is Temperature
ContBound = 0.7

InitialCond {
  t = 1.7
  X->N2 = 0.79
  X->O2 = 0.21
}

Example: FlameMaster input file
```
Typical input file (part 2)

- Boundary and initial conditions (here initial condition for homogeneous reactor)
  - Initial composition either from phi and global reaction or specification in `InitialCond`
  - $X \rightarrow N_2$ is mole fraction
  - $Y \rightarrow N_2$ is mass fraction
  - $T$ or $t$ is temperature
    - If $T < 10$, then value is $1000/T$
  - Continuation in temperature, if `ContInc` is non-zero
    - `ContType` is Temperature means continuation in temperature
    - Continuation ends, if `ContBound` is reached
    - `ContInc` is increment between two simulations. If `ContInc < 10`, then increment is applied to $1000/T$
    - In the example, temperature is varied from $1000/T = 1/1.7$ to $1/0.7$ corresponding to 588 to 1000

Example: FlameMaster input file

```plaintext
# FlameMaster Simulation

Flame is Isochor Homo Reactor
#Flame is Isobar Homo Reactor

#phi = 0.5
phi = 1.0
#phi = 2.0

Pressure = 13e5
Pressure = 1e5

# Boundary conditions #

ContInc = -0.1
ContType is Temperature
ContBound = 0.7

InitialCond {
  t = 1.7
  X->N2 = 0.79
  X->O2 = 0.21
}
```

Example: FlameMaster input file
FlameMaster Tutorial: Homogeneous Reactor

Constant volume homogeneous reactor computing ignition delay times for $n$-heptane/air mixtures

• Example:
  cd Run/FlameMan/Prem/0D/nHeptane
  FlameMan

• This will create a file
  Output/N-C7H16_IgniDelTimes.dout
  which has the values of ignition delay times for all computed conditions as tab-separated list

• If the flag WriteEverySolution is TRUE is added to the input file, then mass fractions and temperatures as function of time are given in files like
  Y1_N-C7H16_p13phi100to0833.dout

• File names include information on fuel, pressure, equivalence ratio, and initial temperature
Constant volume homogeneous reactor performing sensitivity analysis for hydrogen/air mixtures

- Example:
  cd Run/FlameMan/Prem/0D/H2
  FlameMan

- If the flag SensAnalReac is TRUE is added to the input file, then
  - Reaction sensitivity analysis is performed
  - Sensitivity coefficients for ignition delay times or chosen species are given in files like MAXSorted_Reac_SC_H2_p01phi100to1000.tout

- More details are found in Run/FlameManPrem/0D/SensAnal0D.pdf
**FlameMaster Tutorial: Homogeneous Reactor**

Constant volume homogeneous reactor performing reaction flux analysis for hydrogen/air mixtures

- **Example:**
  
  ```
  cd Run/FlameMan/Prem/0D/H2
  FlameMan
  ```

- **If the flag** `ReactionFluxAnal is TRUE` **is added to the input file, then reaction pathway analysis is performed, and production and consumption channels of species are given in files like**

  ```
  Output/IntProRa_H2_p30phi100to1000.tout (requires also AdditionalOutput is TRUE)
  ```
Perfectly stirred reactor (PSR) computing stable species profiles for methane/air mixtures

Example:
```
cd Run/FlameMan/Prem/PSR
FlameMan
```

This will create a file `Output/PSR_X_CH4_p01phi128tau00.14.dout` which has mole fractions of species for all computed initial temperatures as tab-separated list.

More details are found in `Run/Prem/PSR/README.md`.
Unstretched premixed flame computing laminar burning velocities for methane/air mixtures

- **Example:**
  
  cd Run/FlameMan/Prem/Unstretched/CH4
  FlameMan
  ListTool -s syms -r r.dout Output/CH4_p01_0*0298

- **FlameMan** will create files like `Output/CH4_p01_0phi1_0000tu0298` containing the spatial distribution of all scalar fields at all computed conditions
  - **With** `PrintMoleFractions` is TRUE, also mole fractions are printed

- **ListTool** generates a file `r.dout`, which has the values of burning velocities or other scalar quantities specified in `syms` and maximum values of all vector quantities for all computed conditions as tab-separated list
  - Specified symbols are from the header of FlameMaster output files
FlameMaster Tutorial: Counterflow Diffusion Flame

Counterflow diffusion flame with plug flow boundary condition on physical coordinate for acetylene/air mixtures

- Example:
  cd Run/FlameMan/Diff/SteadyPlugFlow
  FlameMan

- This will create a file
  Output/C2H2_p01a00018tf0300to0400
  containing the spatial distribution of all scalar fields at all computed conditions

- Different forms of continuation can be used
  - E.g., if the flag ContType is velocity is added to the input file, then critical conditions (e.g. strain rates) of extinction or autoignition can be determined.
Counterflow diffusion flame with potential flow boundary condition on similarity coordinate for hydrogen/air mixtures

- Example:
  cd Run/FlameMan/Diff/Steady/H2
  FlameMan -i H2.count.in

- This will create a file
  OutCount/H2_p01_0a00100_0tf0300to0300
  containing the spatial distribution of all scalar fields at all computed conditions

- More details are found in
  Repository/doc/FlameManMan.pdf
  Repository/doc/MThesisGer.pdf
FlameMaster Tutorial: Flamelet Equations

Diffusion flames solving flamelet equations using mixture fraction coordinate for hydrogen/air mixtures

- Example:
  cd Run/FlameMan/Diff/Steady/H2
  FlameMan -i H2.mixfrac.in

- This will create a file
  OutMixFrac/H2_p01_0chi1.734tf0300to0300
  containing the spatial distribution of all scalar fields at all computed conditions
FlameMaster Tutorial: Flamelet Equations

Solving flamelet equations with continuation for methane/air mixtures

- Example:
  cd Run/FlameMan/Diff/Steady/CH4
  FlameMan -i FlameMasterCont.input

- If the flag ArclengthCont is TRUE is added to the input file, FlameMaster computes solutions along S-shaped curve

- ListTool -s syms -r r.dout OutMethaneCont/CH4_p01_0chi*st* creates a file r.out containing data for scalar quantities specified in syms and maximum values of all vector quantities for all computed flamelets

- More details are found in Run/FlameMan/Diff/Steady/CH4/README.md
Transient flamelet for hydrogen/air mixtures

- Example:
  cd Run/FlameMan/Diff/Unsteady/H2
  FlameMan

- This will create output files
  Lib_Ch{i}01_{.64_T1742t00014}.tout
  Lib_Ch{i}01_{.64_T1742t00015}.tout
  Lib_Ch{i}01_{.64_T1742t00016}.tout
...

which has the values of temperature, species mass and mole fractions, velocity, heat release, density ... as function of mixture fraction at the computed condition
FlameMaster Flame Calculator

- FlameMaster: A C++ Computer Program for 0D Combustion and 1D Laminar Flame Calculations
  - Premixed and non-premixed
  - Steady and unsteady
  - Emphasis on pre- and post-processing
    - Sensitivity analysis
    - Reaction flux analysis
  
- At request, available online at https://itv.rwth-aachen.de/index.php?id=13
Part I: Fundamentals and Laminar Flames

• Introduction
• Fundamentals and mass balances of combustion systems
• Thermodynamics, flame temperature, and equilibrium
• Governing equations
• Laminar premixed flames: Kinematics and Burning Velocity
• Laminar premixed flames: Flame structure
• Laminar diffusion flames
• FlameMaster flame calculator

• Introduction
• Counterflow diffusion flame
• Flamelet structure of diffusion flames
• FlameMaster flame calculator
• Single droplet combustion