Cantera

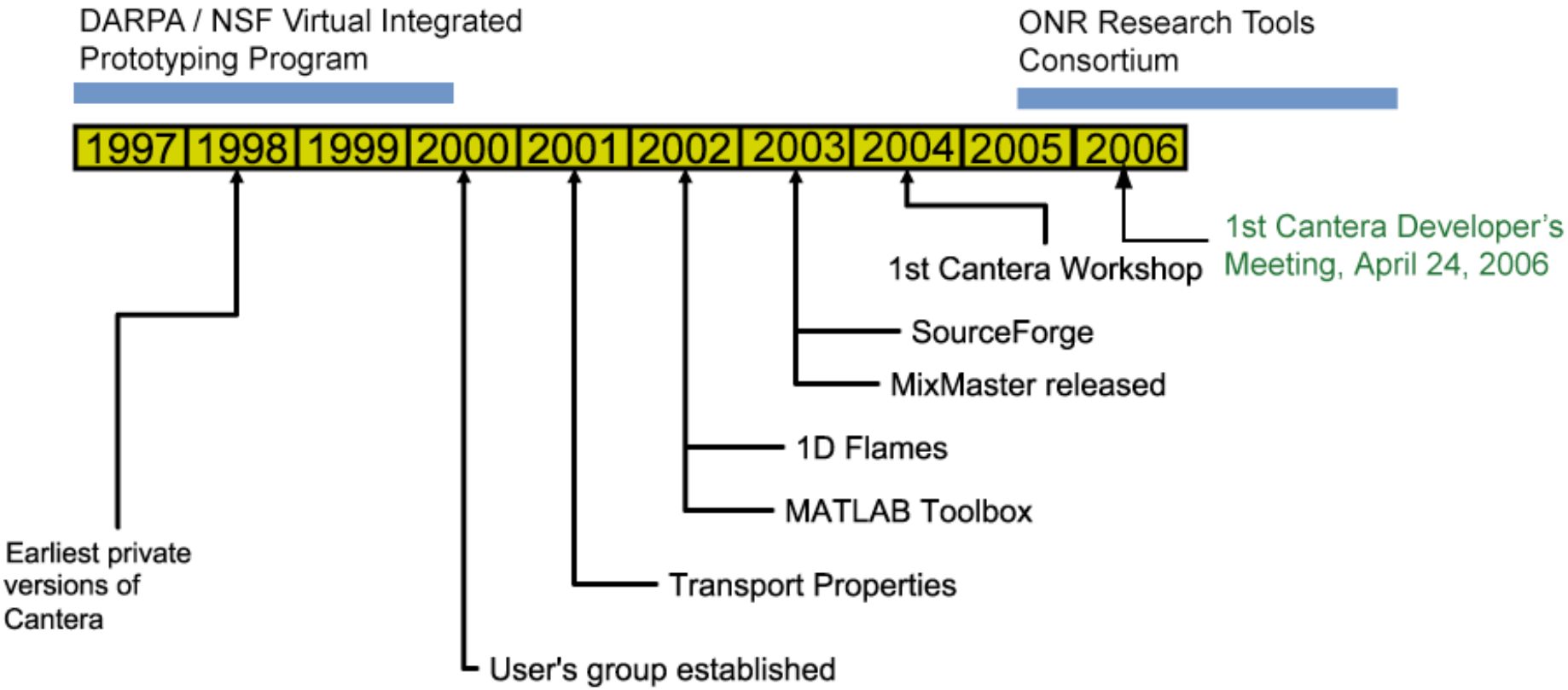
Dave Goodwin
Division of Engineering and Applied Science
California Institute of Technology

Cantera is a suite of software tools for reacting flow problems

- Thermodynamic and transport properties
- Homogeneous and heterogeneous chemistry
- Chemical equilibrium
- Reactor networks
- Steady 1D flames
- Reaction path diagrams
- Non-ideal equations of state
- Electrochemistry

- Open source
- Object-Oriented
- Multi-Platform: Windows, Mac, linux, unix, …
- Use from C++, Fortran 77, Fortran 90, Python, or MATLAB
- Backward compatible with CHEMKIN-II

Timeline

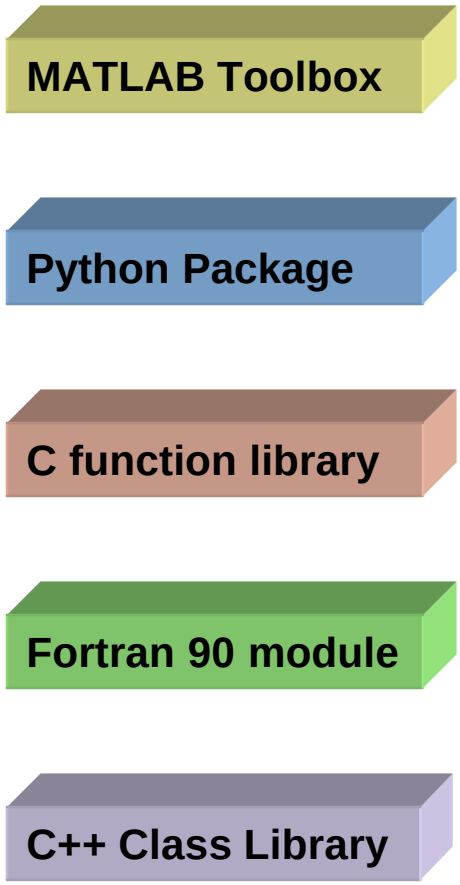

DARPA / NSF Virtual Integrated Prototyping Program
ONR Research Tools Consortium
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
1st Cantera Developer's Meeting, April 24, 2006
1st Cantera Workshop
SourceForge
MixMaster released
1D Flames
MATLAB Toolbox
Transport Properties
Earliest private versions of Cantera
User's group established

Cantera is designed to be easy to use, but without sacrificing performance

- Simple, intuitive structure in terms of easy-to-understand **objects**:
  - phases, mixtures, interfaces, reactors, BVP solvers, 1D reacting flows, …
- Undergraduate and graduate student can learn Cantera and do real computations in a few hours. (Faculty take a little longer.)
  - Work in MATLAB or Python
  - no need to learn Fortran of C++
- Example scripts
- Python and MATLAB interfaces are only front ends; calculations are done in optimized, compiled code
- Cantera has most of the functionality of familiar CHEMKIN-II, plus additional capability (non-ideal phases, multiphase equilibrium, electrochemistry…)

Cantera is multilingual

- Cantera can be used from several popular programming / problem-solving environments
- Interactive / scripting environments (MATLAB, Python) for rapid problem solution and software prototyping
- Scripts can be easily translated into Fortran 90, C, or C++.
- Python interface simplifies integration with other CMDF tools

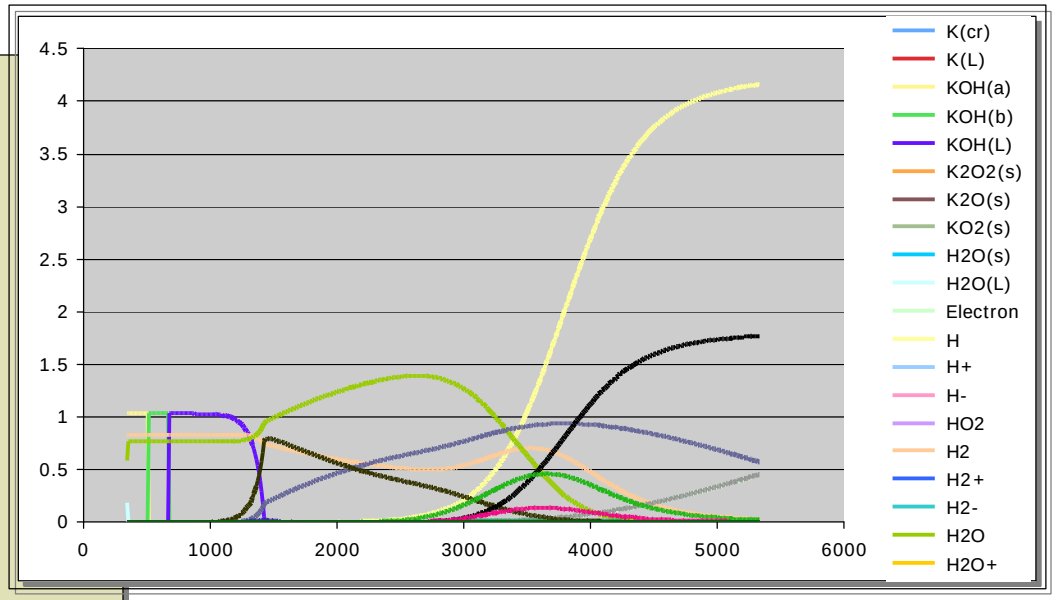


Simple, high-level, object-oriented interface

```
>> gas = GRI30;
>> set(gas, ‘T’, 300.0, ‘P’, oneatm, ‘X’, ‘CH4:1.0, O2::2, N2:3.76’););
>> equilibrate(gas,’HP’);
>> temperature(gas)

 ans =

      2.220222916787240e+03
```

Multiphase equilibrium is also easily formulated and solved

```
from Cantera import *

phases = importPhases('KOH.cti',
                      ['K_solid',
                       'K_liquid', 'KOH_a', 'KOH_b',
                       'KOH_liquid', 'K2O2_solid',
                       'K2O_solid', 'KO2_solid',
                       'ice', 'liquid_water','KOH_plasma'])

mix = Mixture(phases)

f = open('equil_koh.csv','w')
writeCSV(f,['T']+mix.speciesNames())

for n in range(500):
    t = 350.0 + 10.0*n
    mix.set(T= t, P = OneAtm, Moles="K:1.03, H2:2.12,
        O2:0.9")
    mix.equilibrate("TP",maxsteps=1000,loglevel=0)
    writeCSV(f,[t]+ list(mix.speciesMoles()))

f.close()
```



Cantera has a similar ‘look and feel’ in all environments

MATLAB

```
gas = importPhase('gri30.cti');
setState_TPX(gas,300.0,OneAtm,'CH4:1,O2:2,N2:7.52');
equilibrate(gas,'HP');
disp(gas)
```

Python

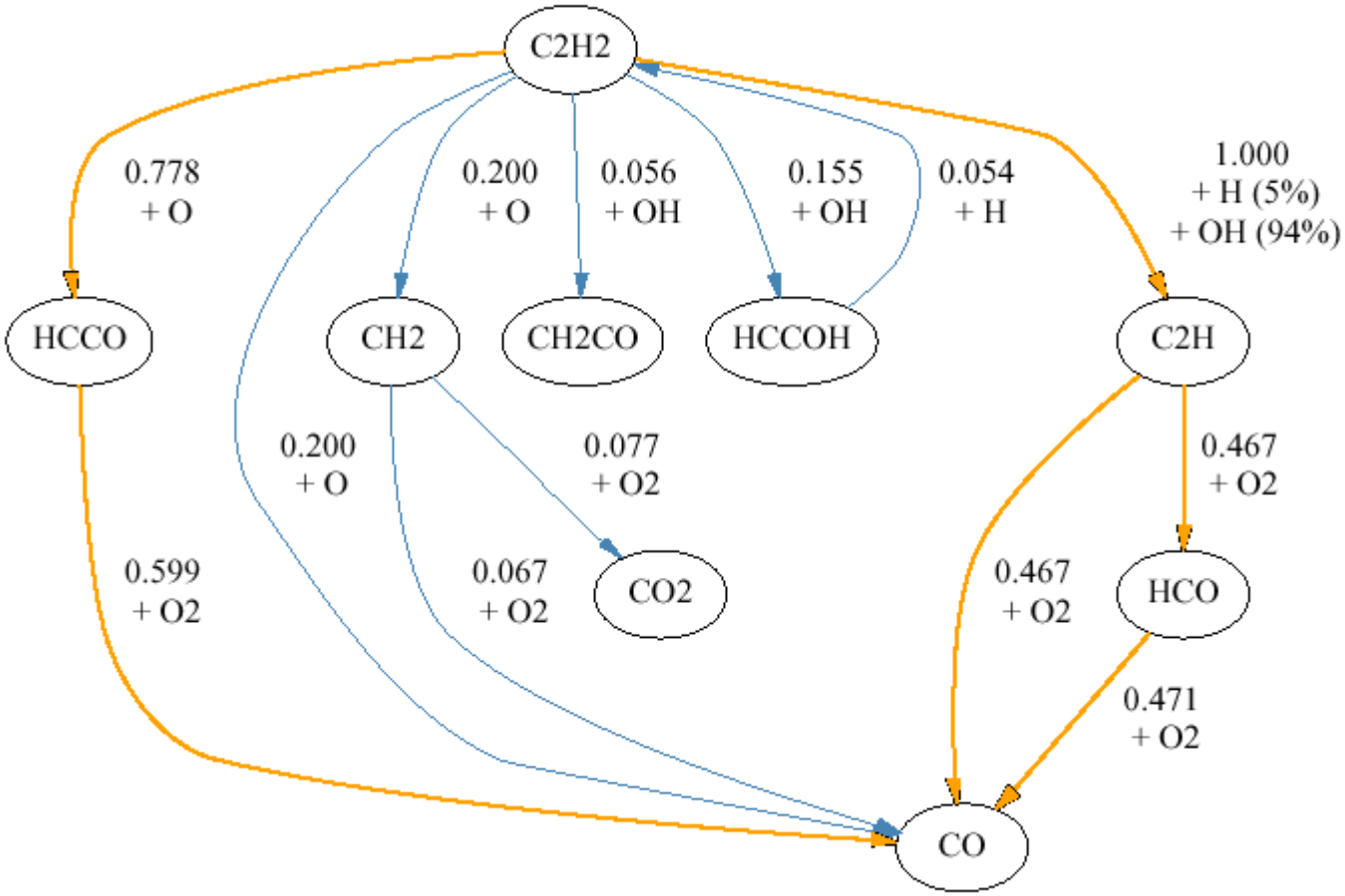
```
from Cantera import *
gas = importPhase('gri30.cti')
gas.setState_TPX(300.0,OneAtm,'CH4:1,O2:2,N2:7.52')
gas.equilibrate('HP')
print gas
```

Fortran 90

```
program equil
use cantera
type(gas_t) gas
gas = IdealGasMix('gri30.inp')
call setState_TPX(gas, 300.0,OneAtm,'CH4:1,O2:2,N2:7.52')
call equilibrate(gas,'HP')
call printSummary(gas)
end
```

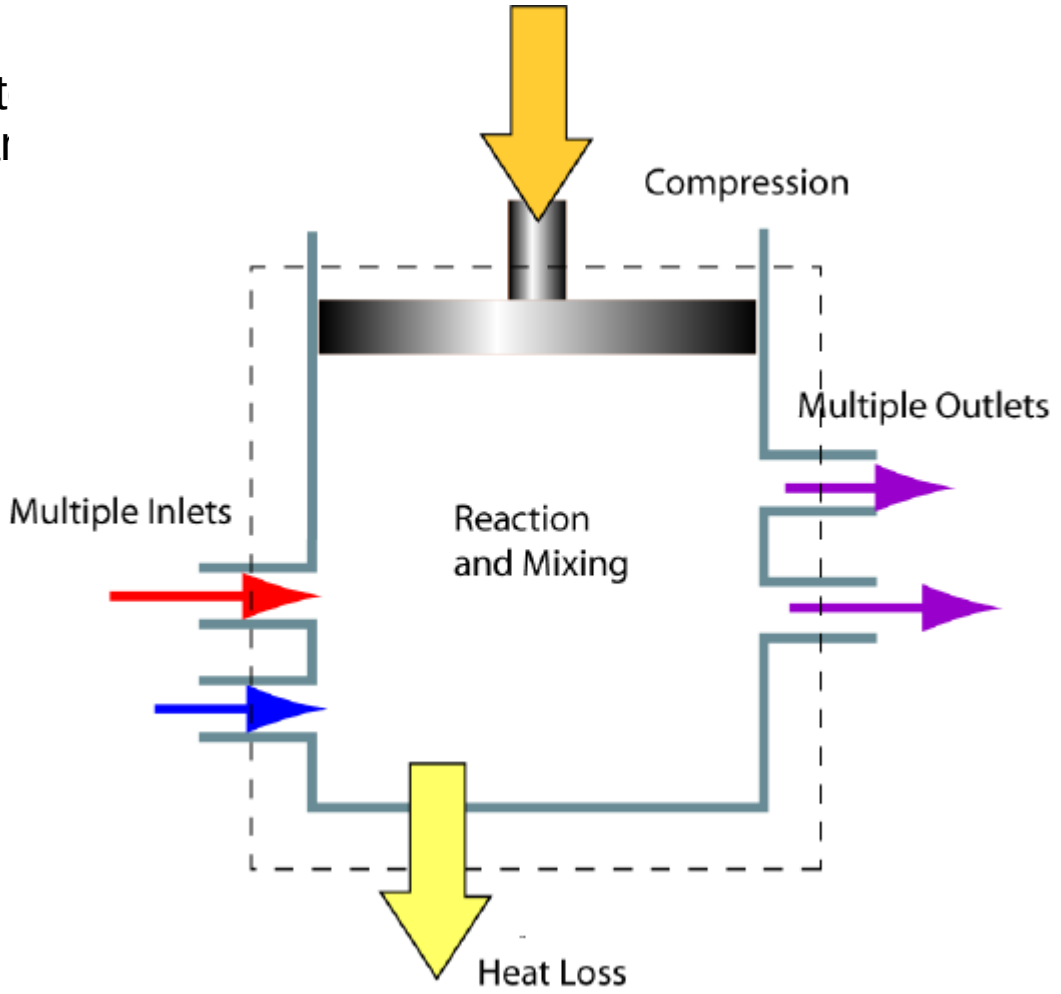
Reaction Path Analysis

Reaction path diagrams can be generated at any point in a simulation

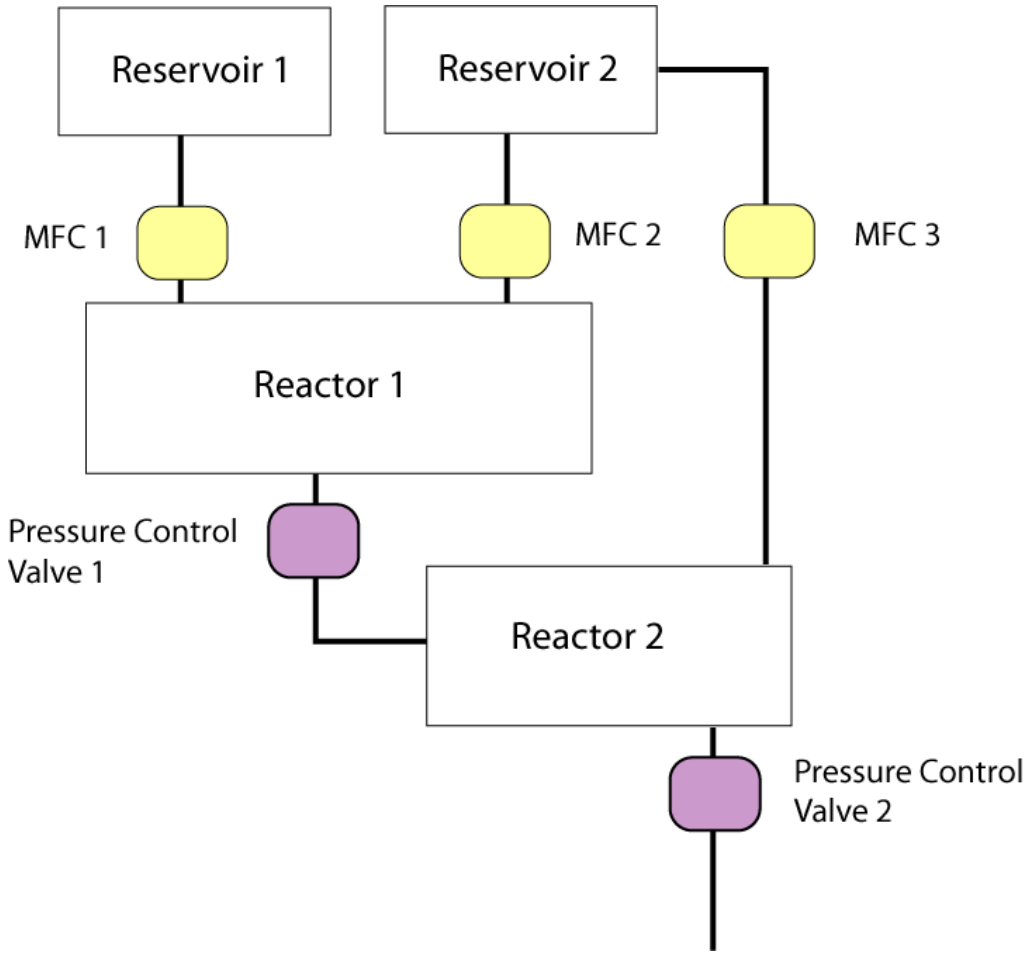


Stirred Reactor Models

- Generic transient stirred reactor model can be used t build many different batch ai continuous reactors
- Mass flow rates, heat loss, volume change may all be varied as functions of time
- Reactors can be linked to create complex process models with sensors and closed-loop control



Reactors may be connected in arbitrary networks



Reservoirs provide specified inputs

Each reactor may use a different mixture model or reaction mechanism

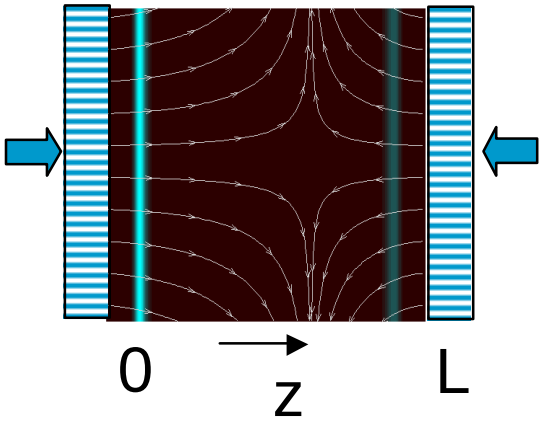
Flow control devices with closed-loop (PID) controllers

Can assemble many different processes from a small set of components

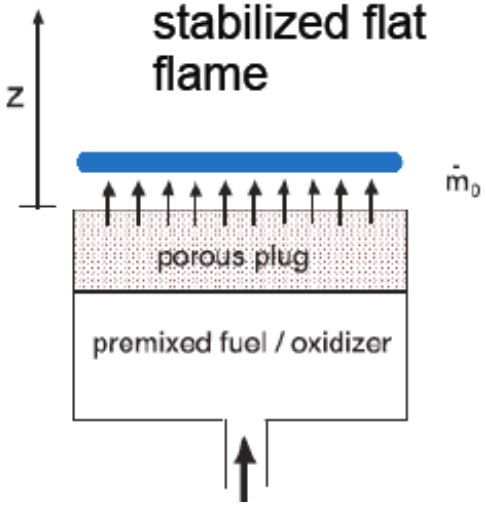
All inputs may be time-dependent

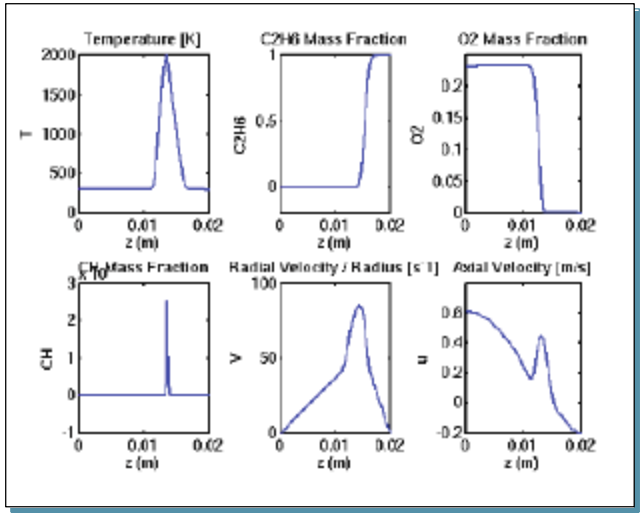
One-Dimensional Flames

Conterflow premixed or non-premixed flames

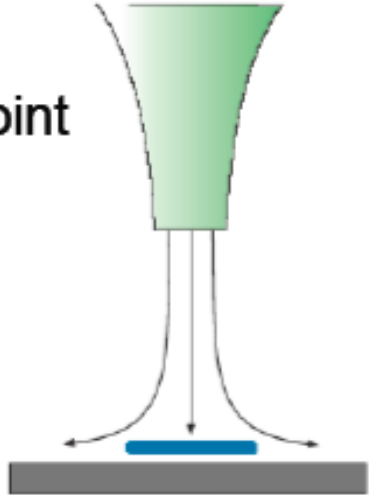


A burner-stabilized flat flame



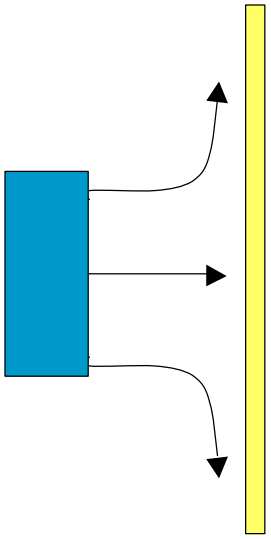


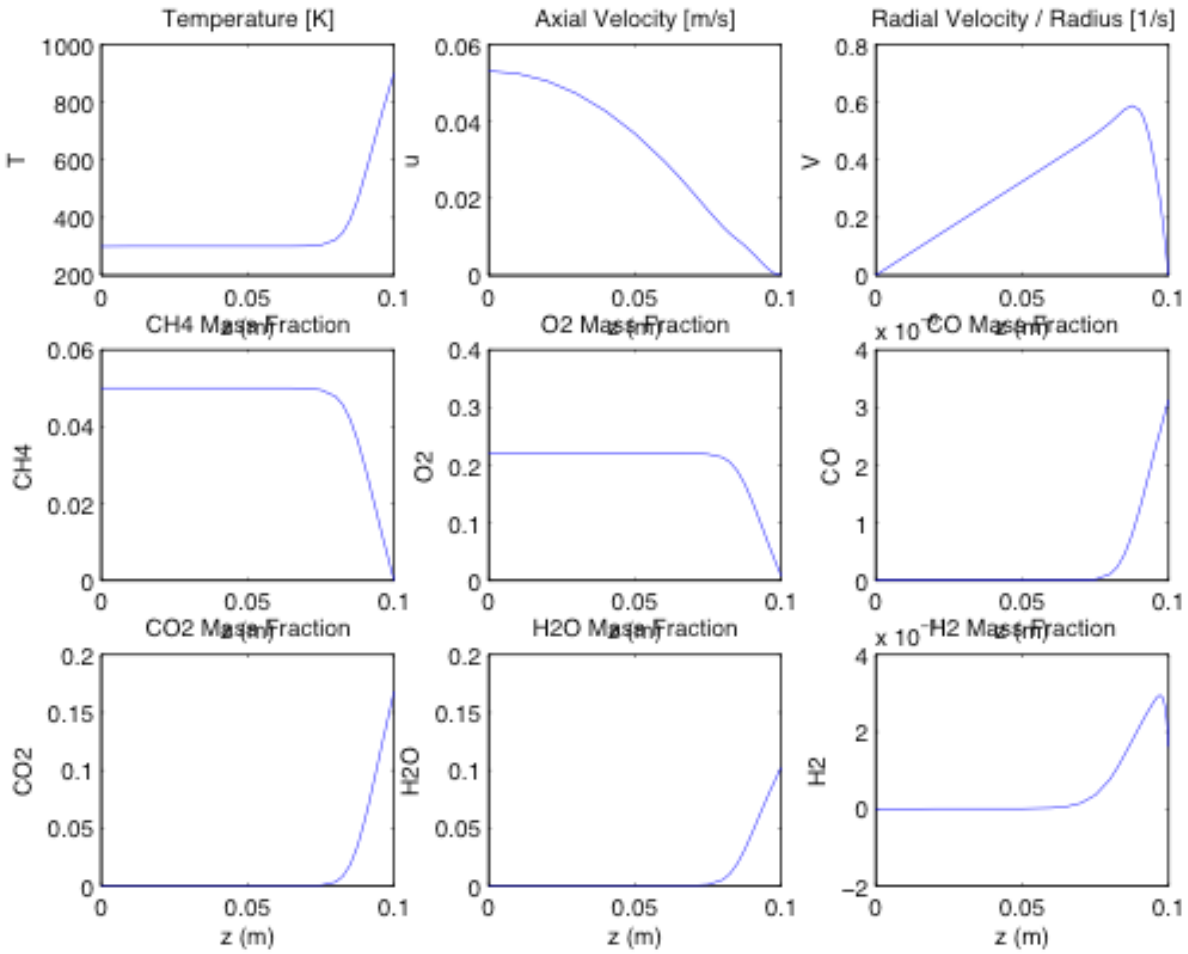
A premixed stagnation-point flame



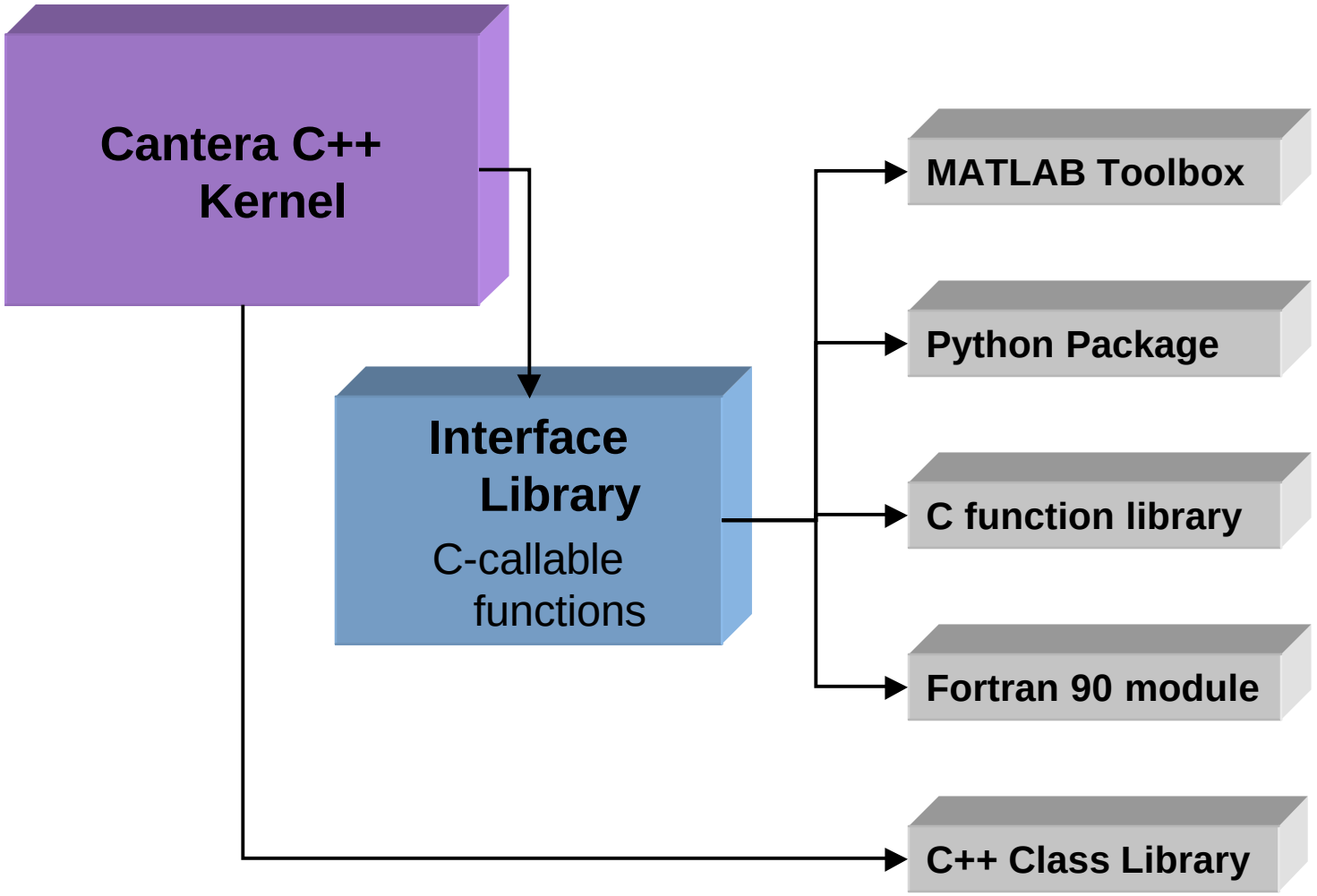
Catalytic Combustion

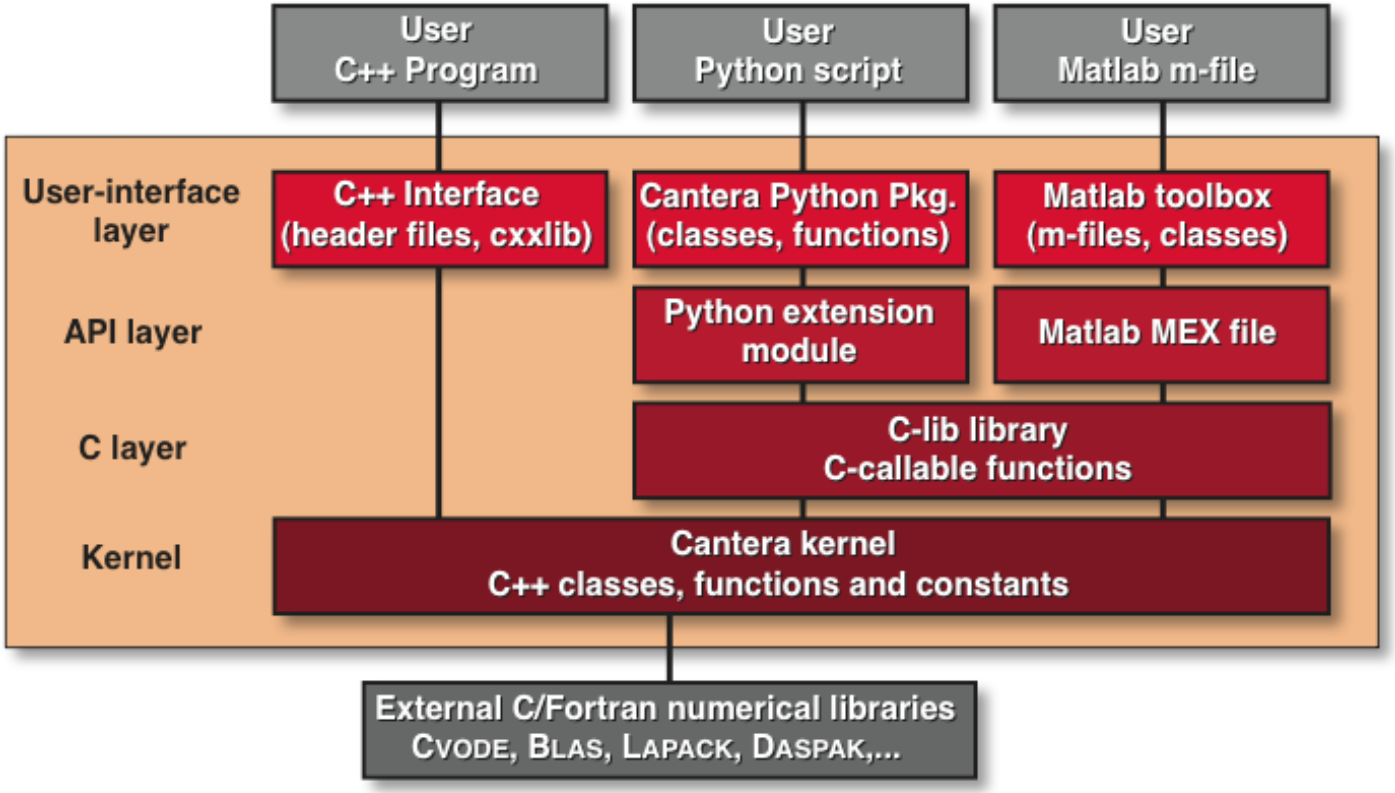
- Mechanism of Deutschmann (1995)
- 9.5% methane in air
- Platinum surface
- Tin = 300 K
- Tsurface = 900 K
- 10 cm separation





All interfaces use a common C++ kernel




User
C++ Program
User
Python script
User
Matlab m-file
User-interface layer
C++ Interface (header files, cxxlib)
Cantera Python Pkg. (classes, functions)
Matlab toolbox (m-files, classes)
API layer
Python extension module
Matlab MEX file
C layer
C-lib library
C-callable functions
Kernel
Cantera kernel
C++ classes, functions and constants
External C/Fortran numerical libraries
CVODE, BLAS, LAPACK, DASPAK,...

The Kernel

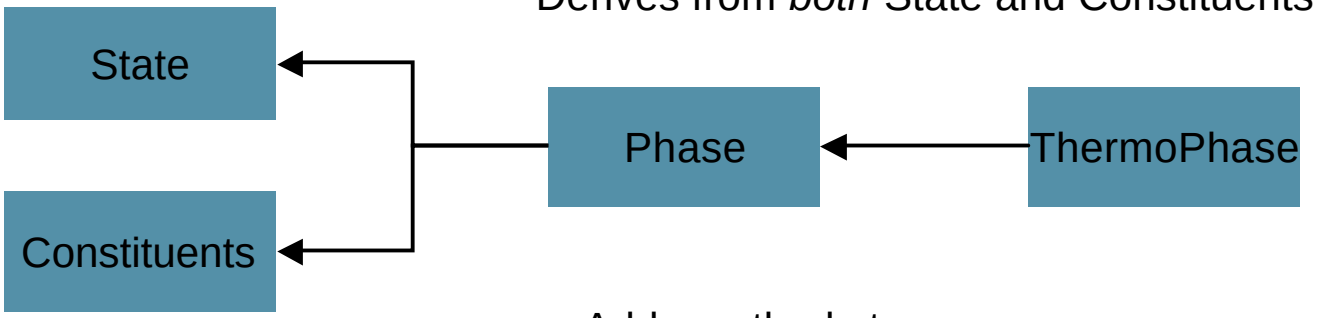
- C++ class library
- Designed for efficiency
  - Property caching
  - Virtual methods used sparingly
  - Templates used to allow inlining at compile time
  - Standard Template Library container classes used
  - CPU-intensive code hand-optimized
- Uses standard open-source numerical libraries
  - BLAS, LAPACK, CVODE

Commo class hierarchy for all phase types (gases, solids, surface phases, ...)

Begin with a simple class that only knows thermodynamic **state** data (T, ρ, mass fractions) + molecular weights

Adds "placeholder" virtual functions for thermo properties. Defines the iinterface for concrete classes representing specific phase types.

Derives from *both* State and Constituents



Attributes of **constituent** elements and species (name, mass, elemental composition, charge)

Adds methods to

- save/restore state
- specify composition using strings: "CH4:1.0, O2:2.1, N2:3.76'

What can it do?

- get/set temperature and density
- get/set mass and mole fractions
- compute useful functions of composition

Still no thermo properties (other than temperature and density). These require knowledge of EOS.

Chemkin-II compatibility

- Converter for Chemkin mechanism files into Cantera CTI format
- Thermodynamic, kinetic, and transport models implemented in Chemkin-II are also implemented in Cantera (and others not in Chemkin-II)
- Very straightforward to write a Fortran-callable library with same interface as the Chemkin-II libraries, allowing easy porting of existing applications

Example: A Cantera look-alike for the Chemkin-II library subroutine CKWYP

```
      subroutine ctwyp(p,t,y,ickwrk,rckwrk,wdot)
      implicit double precision (a-h,o-z)
      double precision y(*), rckwrk(*), wdot(*)
      integer ickwrk(*)

c     set the state
      psi = 0.1*p
      call setState_TPY(t, psi, y)

c     get the net production rates
      call getNetProductionRates(wdot)

c     convert SI -> cgs
      nsp = nSpecies()
      do k = 1, nsp
         wdot(k) = 1.0d3*wdot(k)
      end do
      return
      end
```

convert cgs to SI units

set the state using T, P, Y

compute the production rates

convert results from SI to cgs and return

Current Status

- Web-based user's group with nearly 500 members
- Membership still growing rapidly
- 15 - 20 downloads from Sourceforge per day
- Used in combustion courses at Caltech, Berkeley, Stanford, …
- Likely that Cantera will be used in the next version of Fluent for property evaluation, kinetic rates, and chemical equilibrium
- Interest at Sandia in porting some very large reacting flow codes to Cantera

Where to from here?

- Current capabilities are adequate for many users
- New capabilities on the horizon
  - Electrochemistry (fuel cells, corrosion)
  - Equations of state for ionic liquids (Debye-Huckel)
  - A sectional aerosol model with surface chemistry
- Current development model must change
  - Changes / enhancements to the code must consider impact on all users
  - Mechanism needed to propose, discuss, and implement changes
- Providing support for a growing user community is becoming an issue

A Diverse User Community

- **Users**
  - Students, combustion researchers, engineers
  - May be familiar with engineering tools like MATLAB
  - Don't know (or want to learn) C++
  - Relatively small calculations
- **User Requirements**
  - "hand holding" to install software on PC
  - foolproof binary installers
  - support for Windows OS
  - interfaces for MATLAB and scripting languages
  - a simple, intuitive user interface

- **Developers / power users**
  - require access to source code
  - unix-like (linux/Mac) OS
  - want to call from their own codes
  - large-scale simulation
- **Developer Requirements**
  - efficient libraries
  - call from C++, C, and/or Fortran
  - documented source code

Standards

- Cantera should be able to make use of other tools being developed elsewhere
- For example (hypothetical !) a user could type in Python

```
fullmech = getBestModel(src = 'Prime', type='CH4/air flame', …)
reduced_mech = fullmech.reduce(method = 'RIOT', max_species = 10, …)
```

- Not hard to do. Python has capabilities to send/receive data over the web, including filling out web forms
- Would be better, however, if XML standards existed to mediate the exchange of data