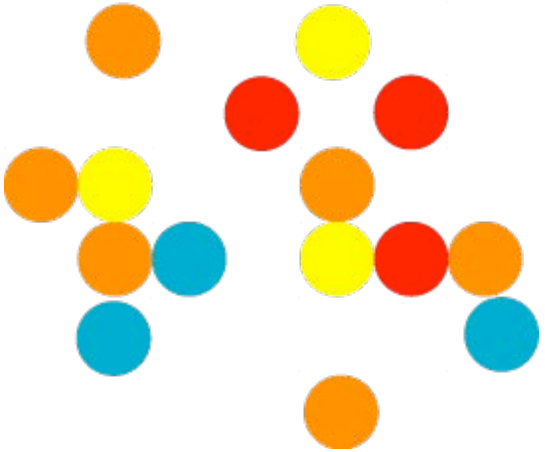
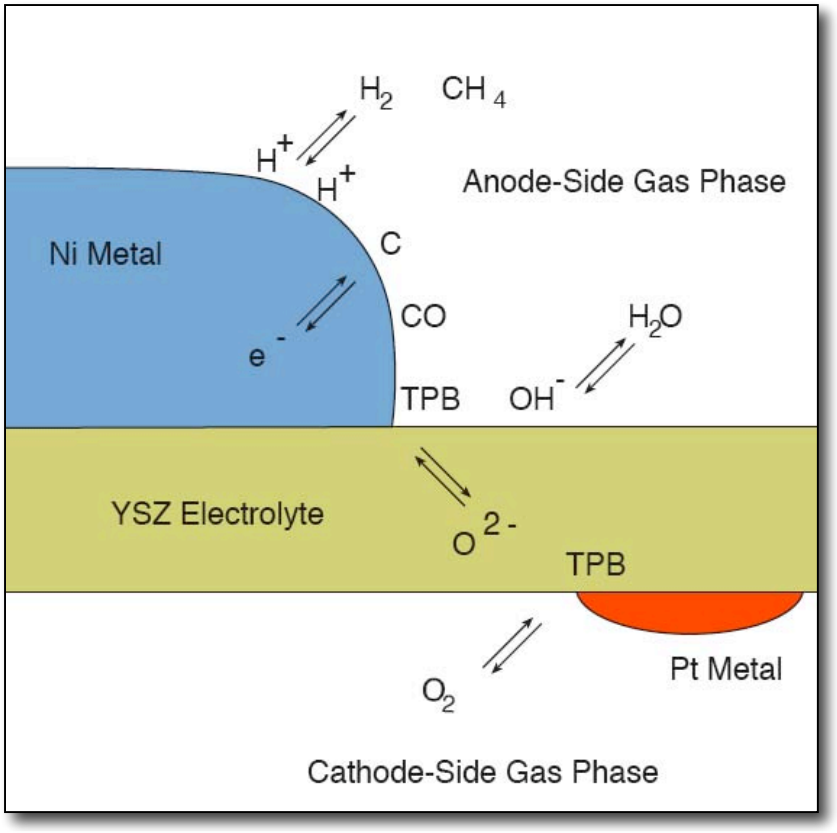
Creating Phase and Interface Models



D. G. Goodwin
Division of Engineering and Applied Science
California Institute of Technology

Cantera Workshop
July 25, 2004

Every Cantera simulation involves one or more phases of matter



Phases and interfaces involved in a hypothetical solid-oxide fuel cell simulation

phases:
metal catalysts (2)
electrolyte
gas phases (2)

interfaces:
electrolyte/metal (2)
metal/gas (2)
electrolyte/gas (2)
triple phase boundaries (2)

5 phases and 8 interfaces!

(of course, many problems may only involve 1 phase, or 1 phase + 1 interface)

Properties of each phase and/or interface are needed

- Composition
  - elements and species
  - atomic and molecular weights
  - mole fractions, mass fractions, concentrations
- Thermodynamic properties
  - Temperature, pressure, density
  - internal energy, enthalpy, entropy, Gibbs free energy,...
  - heat capacities, thermal expansion coefficients, compressibilities, ...
- Transport properties
  - Viscosity, diffusion coefficients, thermal conductivity, electrical conductivity, ...
- Kinetics
  - reaction stoichiometry
  - rates of progress
  - equilibrium constants
  - species production rates

Phase Models

- Specification of chemical composition
  - elements
  - species, and their elemental composition
- Specification of reaction stoichiometry
- Complete set of algebraic equations to compute all required
  - thermo properties
  - transport properties
  - reaction rates
- Numerical values for all parameters in the equations
- Phase models incorporate sub-models
  - Thermo model
  - Transport model
  - Kinetics model

Interface Models

- Specification of the phase(s) adjacent to the interface
- Specification of chemical composition of interfacial species, if any
- Specification of reaction stoichiometry for heterogeneous reactions involving bulk species in any adjacent phase and/or interfacial species
- Complete set of algebraic equations to compute all required
  - thermo properties
  - transport properties (not yet implemented)
  - reaction rates
- Numerical values for all parameters in the equations

Phase and interface models are specified in text files

- Files that can be edited with any text editor, shared, e-mailed, etc.
- Separates phase and interface definition from the simulation itself; allows re-using same phase and interface models for many different applications
- Multiple phases and/or interfaces can be specified in one file
- Species and reaction definitions can be imported from other input files

CTI and CTML files

- The files described here will be called 'CTI' files because they have the default extension '.cti' (**C**an**T**era **I**nput)
- These files are designed to be easy for people to write and read
- They are hard for machines to read, however, since they assume chemical knowledge (most) machines don't have (e.g. how to interpret a chemical equation)

CTML

- XML is a widely-used meta-language for data
  - designed to be easy for machines to parse
  - "dumbed down": in good XML code, nothing is assumed, everything is stated explicitly
  - too verbose for direct human writing or reading
  - good as an intermediate format designed for machine use
  - many browsers, validating parsers, editors, etc. developed for XML
- Cantera defines its own XML-based markup language called CTML
  - designed to represent quantities needed by Cantera (rate coefficients, etc.) in XML
  - borrows some features from CML (Chemical Markup Language)
  - as other XML standards develop for kinetics, CTML may evolve for compatibility

The CTI to CTML preprocessor

- CTI files are designed to be written and read by humans
- When a CTI file is specifed as an input file in Cantera, it is converted on-the-fly to CTML, and then the CTML is parsed
- The conversion process is done automatically by invoking the Python interpreter
- a single Python script called **`ctml_writer.py`** is used. The full Cantera Python interface does not need to be installed to process CTI files.
- CTI files are actually executable Python scripts, and may include any valid Python code

Phases and Interfaces

- Most Cantera simulations require properties of at least one phase of matter
  - For combustion simulations, this is often a reacting ideal gas mixture
  - But solid or liquid phases may also be required

- For heterogeneous combustion problems, properties of the interfaces between phases are also needed

Input File Formats

- A widely-used file format for combustion problems is the one used with the Chemkin software package

- But this format is inconvenient for several reasons
  - Not extensible - no way to add additional parameters for non-ideal phases
  - Only supports one species thermo parameterization (NASA polynomials)
  - Thermo data format is holdover from punched-card era; fixed-column format prone to errors

The Cantera input file format is designed to be...

- free format
- intuitive to write
- clear and understandable to read
- extensible

Directives and Entries

- CTI files consist of two types of elements:
  - **entries** specify a phase, interface, element, species, or reaction
  - **directives** set options determining how entries will be processed

- Both have a function-like syntax:

  - <name>(<keyword1> = <value1>, <keyword2> = <value2>, ...)

  - Example: the units directive

fields (keyword/value pairs) may appear in any order

```
units(length = "cm", time = "s", quantity = "mol",
      act_energy = "cal/mol")
```

Setting the Default Units for Input

```
units(length = "cm", time = "s", quantity = "mol",
      act_energy = "cal/mol")
```

- Default Cantera unit system is SI (meters, seconds, kmol, Joules/kmol)
- This is often *not* the most convenient system for input
- Any desired unit system may be specified with the **units** directive
- Input values will be converted to SI when they are read in
- Good practice to set the unit system at the top of the file

A simple phase: argon gas

```
ideal_gas(name = "argon",
      elements = "Ar",
      species = "AR",
      initial_state = state(temperature = 300.0,
                            pressure = OneAtm)    )
```

- The **ideal_gas** entry is used to define a phase that obeys the ideal gas law

- The phase definition listed above states that:
  - Only element Ar may be present
  - Only one species, named ‘AR’, may be present
  - The object state should be initially set to T = 300 K, P = 1 atm.

The Element Database

- Element attributes (the atomic weight) are looked up by symbol in database file ‘elements.xml’
- Database file contains elements of the periodic table with natural abundance atomic weights
- Isotopes D and Tr also included
- Also contains an element ‘E’ representing an electron, to use in specifying the composition of charged species

```
<ctml>
  <elementData caseSensitive="no">
        <element name="H" atomicWt = "1.00794"/>
        <element name="D" atomicWt = "2.0147"/>
        <element name="Tr" atomicWt = "3.016327"/>
        <element name="He" atomicWt = "4.002602"/>
        <element name="Li" atomicWt = "6.941"/>
        <element name="Be" atomicWt = "9.012182"/>
        <element name="B" atomicWt = "10.811"/>
        <element name="C" atomicWt = "12.011"/>
        <element name="N" atomicWt = "14.00674"/>
        <element name="O" atomicWt = "15.9994"/>
        <element name="F" atomicWt = "18.9984032"/>
        <element name="Ne" atomicWt = "20.1797"/>
```

initial part of elements.xml

Defining Species ‘AR’

```
species(name = "AR", atoms = " Ar:1 ")
```

- The argon gas phase definition references a species ‘AR’
- This species needs to be defined somewhere in the input file
- Order (before or after the phase entry) doesn’t matter, because the entire file is read before the entries are processed
- A minimal definition of species ‘AR’ is shown here, that specifies only the elemental composition

The complete input file (so far)

This simple input file constitutes a valid definition of an argon ideal gas (although not one that much can be done with yet).

```
units(length = "cm", time = "s", quantity = "mol",
      act_energy = "cal/mol")


ideal_gas(name = "argon",
      elements = "Ar",
      species = "AR",
      initial_state = state(temperature = 300.0,
                            pressure = OneAtm)    )

species(name = "AR", atoms = " Ar:1 ")
```

```
>>> from Cantera import *
>>> a = importPhase('argon.cti')
>>> a.density()
1.6227653128888044
>>> a.pressure()
101325.0
>>> a.temperature()
300.0
>>>
```

Adding Thermodynamic Properties

- So far, we have not said anything about the thermodynamic properties of the one species 'AR'

- For an ideal gas mixture, computing the thermodynamic properties requires that the following (or equivalent) be specified for each species:
  - the function $c_p^0(T)$ $T_{min} < T < T_{max}$
  - $h^0(T_{ref})$
  - $s^0(T_{ref})$

- Here a superscript 0 denotes properties evaluated under "standard state" conditions

Standard-State Properties

- Ideal gas species

$$c_p^0 = \lim_{p \to 0} c_p(T)$$

$$h^0 = \lim_{p \to 0} h(T)$$

$$s^0 = \lim_{p \to 0} \left[s(T,p) - R \ln\left(p_{ref} / p \right) \right]$$

- Non-ideal gas models typically use ideal gas species data, then apply corrections for finite density

- Condensed phases

$$c_p^0 = c_p(T, p_{ref})$$

$$h^0 = h(T, p_{ref})$$

$$s^0 = s(T, p_{ref})$$

Specifying species thermo properties

```
species(name = "AR", atoms = " Ar:1 ",
        thermo = const_cp( t0 = 298.15,
                           h0 = 0.0,
                           cp0 = 2.5*GasConstant,
                           s0 = (154.723, 'J/mol/K') )
        )
```

- The *thermo* field of the **species** entry is used to specify parameters to compute the species std. state properties
- This field should contain an embedded entry of type **const_cp**, **Shomate**, or **NASA**
- We'll start with **const_cp**, since it is the simplest and can be used for argon.

Argon gas object now produces the correct thermo properties

```
from Cantera import GasConstant

units(length = "cm", time = "s", quantity = "mol
      act_energy = "cal/mol")


ideal_gas(name = "argon",
      elements = "Ar",
      species = "AR",
      initial_state = state(temperature = 300.0,
                            pressure = OneAtm)

species(name = "AR", atoms = " Ar:1 ",
        thermo = const_cp( t0 = 298.15,
                           h0 = 0.0,
                           cp0 = 2.5*GasConstant,
                           s0 = (154.723, 'J/mol/K') )
        )
```

```
>>> from Cantera import *
>>> a = importPhase('argon.cti')
>>> a

      temperature           300  K
         pressure        101325  Pa
          density       1.62277  kg/m^3
 mean mol. weight        39.948  amu

                         1 kg         1 kmol
                    -----------  ------------
         enthalpy        962558   3.845e+07     J
  internal energy        900118   3.596e+07     J
          entropy       7091.57   2.833e+05     J/K
   Gibbs function  -1.16491e+06  -4.654e+07     J
heat capacity c_p        520301   2.078e+07     J/K
heat capacity c_v        520093   2.078e+07     J/K

                             X            Y
                    -----------  ------------
               AR   1.000000e+00  1.000000e+00
>>>
```

Temperature-dependent c_p^0

- For argon, c_p^0 is really constant, so the **const_cp** model can be used to specify the properties
- For molecular species, however, c_p^0 is temperature-dependent
- Two other parameterizations implemented:
  - Shomate function
  - NASA polynomials

The Shomate Parameterization

- Used in the NIST Chem WebBook http://webbook.nist.gov

$$c_p^0 = A + Bt + Ct^2 + Dt^3 + E/t^2$$

$$h^0 = At + Bt^2/2 + Ct^3/3 + Dt^4/4 - E/t + F$$

$$s^0 = A\ln(t) + Bt + Ct^2/2 + Dt^3/3 - E/(2t^2) + G$$

$$t = T/1000$$

Coefficients for methane from the NIST Chem WebBook

Temperature (K)	298. - 1300.	1300. - 6000.
A	-0.703029	85.81217
B	108.4773	11.26467
C	-42.52157	-2.114146
D	5.862788	0.138190
E	0.678565	-26.42221
F	-76.84376	-153.5327
G	158.7163	224.4143
H	-74.87310	-74.87310
Reference	Chase, 1998	Chase, 1998
Comment	Data last reviewed in March, 1961	Data last reviewed in March, 1961

Shomate Syntax

Single temperature range:

```
thermo = Shomate(range = (Tmin, Tmax),
                         coeffs = (A, B, C, D, E, F, G))
```

Two temperature ranges:

```
thermo = ( Shomate(range = (Tmin, Tmid),
                   coeffs = (A1, B1, C1, D1, E1, F1, G1)),
           Shomate(range = (Tmid, Tmax),
                   coeffs = (A2, B2, C2, D2, E2, F2, G2))
         )
```

The NASA Polynomial Parameterization

- Used in Chemkin and older versions of NASA equilibrium program
- Coefficients for many different molecules available from http://www.ca.sandia.gov/HiTempThermo/index.html

```
CH4               110203H   4C   1    0    0G   300.000  4000.000 1000.00    0 1
 0.47238333E+00 0.12680758E-01-0.55093741E-05 0.11295575E-08-0.89103779E-13    2
-0.96424500E+04 0.16199090E+02 0.38717898E+01-0.42480466E-02 0.24540181E-04    3
-0.21780766E-07 0.63010622E-11-0.10144425E+05 0.66008135E+00                   4
```

program 'ck2cti' will convert this format to Cantera format

$$\frac{c_p^0}{R} = a_0 + a_1 T + a_2 T^2 + a_3 T^3 + a_4 T^4$$

$$\frac{h^0}{RT} = a_0 + \frac{a_1}{2} T + \frac{a_2}{3} T^2 + \frac{a_3}{4} T^3 + \frac{a_4}{5} T^4 + \frac{a_5}{T}$$

$$\frac{s^0}{R} = a_0 \ln T + a_1 T + \frac{a_2}{2} T^2 + \frac{a_3}{3} T^3 + \frac{a_4}{4} T^4 + a_6$$

NASA Polynomial Syntax

Single temperature range:

```
thermo = NASA(range = (Tmin, Tmax),
              coeffs = (a0, a1, a2, a3, a4, a5, a6))
```

Two temperature ranges:

```
thermo = ( NASA(range = (Tmin, Tmid),
                coeffs = (a0, a1, a2, a3, a4, a5, a6)),
           NASA(range = (Tmid, Tmax),
                coeffs = (b0, b1, b2, b3, b4, b5, b6))
         )
```

A gas mixture: air

```
units(length = "cm", time = "s", quantity = "mol", act_energy = "cal/mol")

ideal_gas(name = "air",
          elements = " O  N  Ar ",
          species = """ O  O2  N  NO  NO2  N2O  N2  AR """,
          initial_state = state(temperature = 300.0,
                                pressure = OneAtm,
                                mole_fractions = 'O2:0.21, N2:0.78, AR:0.01')
          )


species(name = "NO",
        atoms = " N:1  O:1 ",
        thermo = (
                  NASA( [  200.00,  1000.00], [  4.218476300E+00,  -4.638976000E-03,
                                  1.104102200E-05,  -9.336135400E-09,   2.803577000E-12,
                                  9.844623000E+03,   2.280846400E+00] ),
                  NASA( [ 1000.00,  6000.00], [  3.260605600E+00,   1.191104300E-03,
                                  -4.291704800E-07,   6.945766900E-11,  -4.033609900E-15,
                                  9.920974600E+03,   6.369302700E+00] )
                  ),
        note = "RUS 78"
        )
```

only one species entry shown

Importing Species Definitions

- Phase definitions can *import* species already defined in another file
- Saves having to re-enter species entries
- A *complete* input file for air that imports species from file gri30.xml:

```
units(length = "cm", time = "s", quantity = "mol", act_energy = "cal/mol")

ideal_gas(name = "air",
          elements = " O  N  Ar ",
          species = "gri30: O  O2  N  NO  NO2  N2O  N2  AR",
          initial_state = state(temperature = 300.0,
                                pressure = OneAtm,
                                mole_fractions = 'O2:0.21, N2:0.78, AR:0.01')
          )
```

This input file builds a complete model for air, just as if the species definitions had been entered in the input file

```
>>> from Cantera import *
>>> a = importPhase('air.cti')
>>> a

       temperature             300  K
          pressure          101325  Pa
           density         1.17681  kg/m^3
  mean mol. weight         28.9697  amu

                          1 kg             1 kmol
                       -----------      ------------
          enthalpy         1893.98          5.487e+04     J
   internal energy        -84207.6         -2.439e+06     J
           entropy         6866.03          1.989e+05     J/K
    Gibbs function    -2.05792e+06         -5.962e+07     J
 heat capacity c_p         1003.06          2.906e+04     J/K
 heat capacity c_v          716.05          2.074e+04     J/K

                             X                Y
                       ------------     ------------
                 O     0.000000e+00     0.000000e+00
                O2     2.100000e-01     2.319575e-01
                 N     0.000000e+00     0.000000e+00
                NO     0.000000e+00     0.000000e+00
               NO2     0.000000e+00     0.000000e+00
               N2O     0.000000e+00     0.000000e+00
                N2     7.800000e-01     7.542530e-01
                AR     1.000000e-02     1.378956e-02
>>>
```

Importing from multiple sources

- Species definitions can be kept in separate files, organized by compound type, data source, or in any other way

- ```
  ideal_gas(name = 'sicvd_gas',
            elements = 'Si H O',
            species  = ['silanes: all',
                        'si_oxides: SiO SiO2',
                        'oh: all',
                        'SiH2O'])
  ```

- This specifies the following species:
  - import all species defined in files 'silanes.xml' and 'oh.xml'
  - import SiO and SiO2 from file 'si_oxides.xml'
  - add species SiH2O defined in this file
  ```

Adding Reactions

- At this point, we can construct gas mixtures, but we have not specified any reactions.
- Such definitions are perfectly usable for problems that do not involve kinetics.
- But of course to do kinetics problems, we need to supplement what we have so far with a speification of what reactions we will consider, and their rate parameters.
- Now we'll look at how to add reactions to phases

The `reaction` entry

short form:

```
reaction( "O + H2 <=> H + OH",    [3.87000E+04, 2.7, 6260])
```

sequence of 3 numbers interpreted as A, b, E in Arrhenius expression

$$k_f = AT^b \exp(-E/RT)$$

long form:

```
reaction( equation = "O + H2 <=> H + OH",
          kf = Arrhenius(3.87000E+04, 2.7, 6260),
          id = 'oh-1')
```

Reaction rate expression

$$q_{fwd} = k_f \prod_j C_j^{\nu^{(r)}_j}$$

$$k_f = A T^b \exp(-E / RT)$$

if reversible,

$$q_{rev} = k_r \prod_j C_j^{\nu_j^{(p)}}$$

$$k_r = \frac{k_f}{K_c}$$

Specified reaction orders for global reactions

- Sometimes global reactions have measured rate laws of the form

$$q_{fwd} = k_f \prod_j C_j^{R_j}$$

  where R_j is an empirical reaction order

- To specify reaction orders, use the *order* field:

```
reaction(equation = "C3H8 + 5 O2 => 3 CO2 + 4 H2O",
         kf = [1.0e8, 0.0, 0.0],
         order = 'C3H8:0.5 O2:0.2')
```

  - Note that the units of k_f are affected by using empirical reaction orders

- Reaction orders can only be specified for irreversible reactions

The Reaction Equation

- Conventions mostly follow those used in the Chemkin-II software package

- **All species names must be separated by whitespace**
  - `"O + H2 <=> OH +H"` is OK
  - `"O+H2<=>OH+H"` is not OK, and will result in an undeclared species error for species 'O+H2' and 'OH+H'
  - the CK2CTI utility automatically adds spaces when converting Chemkin input files.

- Stoichiometric coefficients must be integers

- equality sign
  - "<=>" or "=": specifies that the reaction is reversible, and the reverse rate is to be computed from detailed balance.
  - "=>" or "->": specifies that the reaction is irreversible, and the reverse rate is to be set to zero.

The Reaction Equation (cont'd)

- "H + O + M = OH + M"
  "CH4 + M = CH3 + H + M"

  If 'M' is specified as a reactant or product (should be on both sides of the equation), then this denotes a third-body collision partner. Use only with entry type **three_body_reaction**

- "CH4 (+M) = CH3 + H (+M)"

  Including '(+M)' denotes a pressure-dependent falloff reaction. Use only with entry type **falloff_reaction**

Three-body reactions

- Third-body collision partner required to conserve energy and momentum
  - collisional dissociation reactions
    H2O + M -> OH + H + M
  - association (recombination) reactions
  - OH + H + M -> H2O + M

- Rate expression:

collision efficiencies

$$q_{fwd} = k_f C_M \prod_j C_j^{\nu^{(r)}}$$

$$C_M = \sum_j \varepsilon_j C_j$$

- This set of slides ends here

- Please refer to the document 'definingphases.pdf' on the CD for further information on defining phase and interface models.