Mutation^{++} User's Manual

J.B. Scoggins[*], Thierry Magin
von Karman Institute for Fluid Dynamics, Belgium

October 22, 2012

[*]scoggins@vki.ac.be

Contents

1 Introduction **3**

2 Directory Structure **4**

3 Installation **5**

3.1 From source on Unix or Mac 5

4 Data Files **8**

4.1 Elements 9

4.2 Species 9

4.2.1 *species.xml* 9

4.2.2 *nasa7.dat* and *nasa9.dat* 11

4.2.3 *heavy.dat* 11

4.3 Reaction Mechanisms 11

4.4 Mixtures 11

5 Example C++ Usage **14**

5.1 Introduction 14

5.2 Chemical Equilibrium Example 14

6 Models and Algorithms **17**

6.1 Thermodynamics 17

6.1.1 NASA 7-Coefficient Polynomials (NASA-7) 17

6.1.2 NASA 9-Coefficient Polynomials (NASA-9) 17

6.1.3 Rigid-Rotor / Harmonic Oscillator (RRHO) 17

6.2 Transport 17

6.2.1 Viscosity 17

6.2.2 Thermal Conductivity 17

1 Introduction

Mutation^{++} stands for a **MU**lticomponent **T**hermodynamic **A**nd **T**ransport property library for **ION**ized plasmas written in C++. In addition, Mutation^{++} provides a robust equilibrium solver and finite-rate chemistry source terms. The library is derived from its FORTRAN predecessor, Mutation , written by Thierry Magin. Mutation^{++} was written to increase the performance and robustness of Mutation while also providing a more user friendly interface to the library.

2 Directory Structure

In order to clarify the following sections, a brief reference is given here that outlines the directory structure of the MUTATION^{++} library.

```
mutation++ ........................ root directory of MUTATION++
└─ trunk ....................... directory of current library version
   ├─ data ...................... root directory of all MUTATION++ input files
   │  ├─ mechanisms ............. stores defined reaction mechanisms
   │  ├─ mixtures ............... stores defined mixtures
   │  ├─ thermo ................. stores defined elements, species, and thermodynamic data
   │  └─ transport .............. stores collision integral data
   ├─ docs ...................... root directory for all MUTATION++ documentation
   │  ├─ doxygen ................ where doxygen reference manual is created
   │  │  ├─ html ................ holds the HTML version of the doxygen manual
   │  │  └─ latex ............... holds the PDF version of the doxygen manual
   │  ├─ manual ................. where this User's Manual is located
   │  └─ references ............. stores common literature references
   ├─ src ....................... root directory for all MUTATION++ source code
   │  ├─ general ................ sources not falling under the other categories
   │  ├─ kinetics ............... sources relating to finite-rate chemistry
   │  ├─ numerics ............... sources relating to numerics
   │  ├─ thermo ................. sources relating to thermodynamics and equilibrium
   │  ├─ transport .............. sources relating to transport
   │  └─ utils .................. utility functions and classes
   └─ tools ..................... additional helper tools
```

Figure 1: Directory structure for the MUTATION^{++} library.

3 Installation

This section describes how to properly install and setup the MUTATION^{++} library on your machine (linux is only supported at the moment). In the following instructions, we will denote the full path to the **trunk** directory (see Fig. 1) on the installation machine as **trunk-path**. Therefore, whenever "**trunk-path**" is written, it should be implicitly replaced by the full path to **mutation++/trunk** on the user's machine.

3.1 From source on Unix or Mac

Step 1 - Build and Install Mutation^{++} :

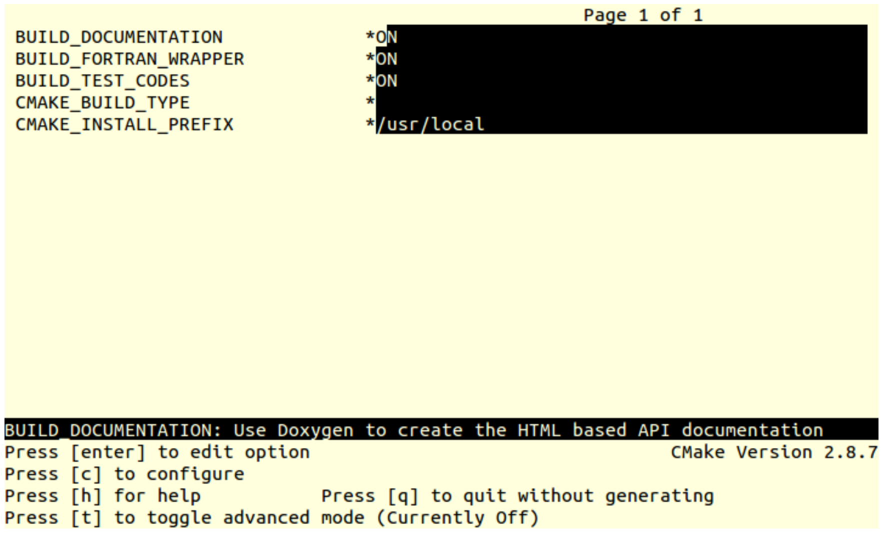
MUTATION^{++} makes use of the cmake utility in order to generate a proper build system in a cross-platform manner. If you do not have cmake installed, you will have to install it before continuing further. With cmake installed, enter the **trunk** directory and setup the build directory.

```
cd trunk-path
mkdir build
cd build
```

Next, configure cmake to generate the necessary Makefiles to build and install MUTATION^{++} . Begin by running ccmake on the **trunk** directory (from inside **build**).

```
ccmake ..
```

Perform the initial configuration by pressing "c". After the configuration is complete the ccmake screen will look like the following. At this point, you can change the various options



```
                                                  Page 1 of 1
 BUILD_DOCUMENTATION             *ON
 BUILD_FORTRAN_WRAPPER           *ON
 BUILD_TEST_CODES                *ON
 CMAKE_BUILD_TYPE                *
 CMAKE_INSTALL_PREFIX            */usr/local

BUILD_DOCUMENTATION: Use Doxygen to create the HTML based API documentation
Press [enter] to edit option                               CMake Version 2.8.7
Press [c] to configure
Press [h] for help          Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

listed. Note that if you do not want to install MUTATION^{++} to the default install path ("/usr/local" as pictured) then change this option accordingly. It is recommended to just make it "../install" which will place the installation in **trunk-path/install**. When you are done changing the options as you prefer, press "c" again to reconfigure, and then "g" to output the configuration and exit the ccmake utility. Now you may actually build and install the library by typing:

```
    make install
```

If no errors were listed during the build and installation processes, then the MUTATION^{++} library is now installed in the install directory chosen during configuration.

Step 2 - Set environment variables:

The MUTATION^{++} library is now successfully built, but in order to load all of the data MUTATION^{++} requires, an environment variable must be set which points MUTATION^{++} to the appropriate data directories. This should be done by adding the following command to your system's login script such as ~ */.bashrc*.

```
    export MPP_DATA_DIRECTORY=/install-path/include/mutation++_data
```

In the above command, **install-path** should be replaced by the full path to the install directory that was created in the previous step.

In addition to the MUTATION^{++} data files, some auxiliary helper/test programs were created and included in the install directory. In order to access these programs from any directory, add the following additional command to your login script.

```
    export PATH=${PATH}:/install-path/bin
```

Step 3 - Test installation using checkmix:

As part of the build process, a small tool program was generated called checkmix. Running checkmix with a mixture name will load that mixture and display a description of what was loaded. For instance, running

```
    checkmix air5
```

produces the following results:

```
Loading mixture file air5 ...
Validating reaction mechanism...
5 species containing 2 elements
5 reactions

Species info:
-------------
            N   O  Mw (g/mol)    Charge       Phase
N           1   0     14.0067         0         gas
N2          2   0     28.0134         0         gas
NO          1   1     30.0061         0         gas
O           0   1     15.9994         0         gas
O2          0   2     31.9988         0         gas
```

```
Default elemental composition:
------------------------------
   N  :  0.79
   O  :  0.21

Reaction info:
--------------
   1: N2+M=2N+M           Arrhenius:   3.000e+16  -1.60  113200.0
      N2: 0.23, NO: 0.23, O2: 0.23
   2: O2+M=2O+M           Arrhenius:   1.000e+16  -1.50   59360.0
      N2: 0.50, NO: 0.50, O2: 0.50
   3: NO+M=N+O+M          Arrhenius:   5.000e+09   0.00   75500.0
      NO: 20.00, N: 20.00, O: 20.00
   4: N2+O=NO+N           Arrhenius:   5.690e+06   0.42   42938.0
   5: O2+N=NO+O           Arrhenius:   2.490e+03   1.18    4005.5
```

Step 4 - Building the reference manual:

Apart from this User's Guide, a reference manual can also be generated for MUTATION^{++} using the doxygen utility. Note that you will need the doxygen program installed on your computer before you can build the reference documentation. Doxygen is the only external library/program required by MUTATION ++ (and only for building the documentation). It automatically generates a full reference manual of the code, both in PDF format and a HTML versions, by parsing the comments in the actual source code. Doxygen can be found at `http://www.stack.nl/~dimitri/doxygen/`. In order to generate the documentation, return to the build directory (**trunk-path/build**) and type the following:

```
make docs
```

Once the documentation is built, you can access the HTML version by opening *index.html* in directory **trunk-path/docs/doxygen/html** and the PDF version by opening *refman.pdf* in **trunk-path/docs/doxygen/latex**. It is recommended to open the HTML version as it is more easy to navigate, but the PDF version could be printed if need be.

4 Data Files

This section details how to create element, species, mixture, and reaction mechanism data that will be loadable by Mutation^{++} . In general, Mutation^{++} data files are listed in one of the subdirectories under the **data** directory.

```
mutation++
└─ trunk
   └─ data
      ├─ mechanisms
      ├─ mixtures
      ├─ thermo
      └─ transport
```

Many of these files are written in a simplified version of the Extensible Markup Language (XML). XML provides a human readable, yet complex and extensible format for data to be stored with only a few, limited rules.

```
<!-- Comment string -->
<root_tag attribute="value">
    <child1_tag>
        text
    </child1_tag>
    <child2_tag attribute="another value" />
</root_tag>
```

Figure 2: Example XML document.

Figure 2 shows a small example of how the simplified XML format works for Mutation^{++} . An XML document begins with a root XML element. Every element must begin with a tag that identifies what type of element it is. The root element depicted in Fig. 2 is of type "root_tag". Every element also ends with an end-tag which signifies the end of the element ("</root_tag>"). Each element may have as many attribute/value pairs as is desired following the element's tag. Each pair must consist of an attribute name followed by an equal sign and the value of the attribute in quotations. Between the begin- and end-tags of an element, an element may also contain one or more child elements or text (but not both). From the figure, the root element contains two child elements named "child1_tag" and "child2_tag". Note that the first child element is an example of an element which contains text instead of more child elements. The second child element is an example of a short-hand format for elements which only contain attributes. For such elements, a full end-tag is not necessary. Instead, simply putting "/>" after the attribute list is sufficient to end the element. Finally, comments can be inserted anywhere outside of element tags. Comment strings begin with "<!--" and end with "-->" and can be spread over multiple lines.

4.1 Elements

The simplest form of data that can be added to the library is an element. Available elements are listed in the file *elements.xml* located in the **thermo** subdirectory. An example illustrating how elements are defined is given in Fig. 3. Note that only three values are currently needed to fully define an element in MUTATION^{++} ; the element's name, its charge, and its molecular weight (mw). The name attribute will be used to provide a text ID tag to the element. The charge is the multiple of the elementary charge carried by the element. Note that in MUTATION^{++} , the electron is considered a special element and cannot be changed. The molecular weight can optionally have a units attribute. If not provided, the units default to kg/mol (thus they are not needed in the example below).

```
<elementlist>
    <element name="Ar" charge="0">
        <mw units="kg/mol">0.039948</mw>
    </element>
    <element name="e-" charge="-1">
        <mw units="kg/mol">0.00000055</mw>
    </element>
    <element name="N" charge="0">
        <mw units="kg/mol">0.0140067</mw>
    </element>
    <element name="O" charge="0">
        <mw units="kg/mol">0.0159994</mw>
    </element>
</elementlist>
```

Figure 3: Example list of elements for *elements.xml*

4.2 Species

In order to define a new species in MUTATION^{++} , three files should be updated.

4.2.1 *species.xml*

The full list of defined species available to MUTATION^{++} is stored in the file *species.xml* located in the **thermo** directory. A shortened version of *species.xml* is presented in Fig. 4. As can be seen in the example, the stoichiometry is defined by a comma separated (optional) list of pairs of element names and the number of atoms of that element belonging to the species separated by a semicolon. Note that the species molecular weight and charge are not explicitly defined. This is because they will be determined based on their respective elemental stoichiometry at runtime in order to ensure mass and charge are correctly conserved.

```
<specieslist>
   ⋮
   <species name="C2H2">
      <stoichiometry>
         C : 2
         H : 2
      </stoichiometry>
      <thermodynamics type="NASA-7" db_name="C2H2 acetylene" />
      <thermodynamics type="RRHO">
         <linear>
            yes
         </linear>
         <rotational_temperature units="K">
            1.70761
         </rotational_temperature>
         <steric_factor>
            2
         </steric_factor>
         <vibrational_temperatures units="K">
            902.6849989  902.6849989  1077.29      1077.29
            2874.908462  4767.493793  4910.840493
         </vibrational_temperatures>
         <electronic_levels units="1/cm">
            <level degeneracy="2" energy="0.0" />
            <level degeneracy="3" energy="25000.0" />
            <level degeneracy="6" energy="35000.0" />
            <level degeneracy="1" energy="42198.0" />
            <level degeneracy="3" energy="50000.0" />
            <level degeneracy="1" energy="54116.0" />
         </electronic_levels>
         <formation_enthalpy T="298 K" P="1 atm" units="J/mol">
            228320.0
         </formation_enthalpy>
      </thermodynamics>
   </species>
   ⋮
</specieslist>
```

Figure 4: Example list of species in *species.xml* showing the C_2H_2 definition.

In addition to the species stoichiometry, different thermodynamics elements can also be given to provide thermodynamic data for various models. In general, no additional information is required to use the NASA 7- or 9-coefficient polynomial databases. This information is separately defined in the *nasa7.dat* and *nasa9.dat* database files in order to remain consistent with these databases. However, it is possible that for some species, the species name does not match the name used in one or both NASA databases. In this case, a thermodynamics element can be added with a type attribute defining the type of the thermodynamics element. If "NASA-7" or "NASA-9" is given, then an additional db_name attribute can be given that defines the name to use when searching for the given species in the NASA database.

A rigid rotator and harmonic oscillator model can also be defined for a species using the type "RRHO". For atoms, this information consists of the species' formation_enthalpy and electronic_levels. For molecules, additional information is required including whether or not the molecule is linear, its steric_factor, characteristic rotational_temperature, and its characteristic vibrational_temperatures. For a full description of how each thermodynamic model is evaluated, see § 6.1. **Note that only one thermodynamic model is required for any given species, but the same model must be present for all species in a mixture in order to use that model.**

4.2.2 *nasa7.dat* and *nasa9.dat*

MUTATION^{++} supports both types of NASA polynomial thermodynamic databases. The 7-coefficient polynomial database is stored in the **thermo** directory in file *nasa7.dat*. Similarly, the 9-coefficient database is given in file *nasa9.dat*. Details on how each database evaluates thermodynamic properties can be found in § 6.1.1 and § 6.1.2. Most likely, any new species added to *species.xml* is already present in one of the NASA databases, however if this is not the case, then the following references should be used to properly format your data for each database: [1] (NASA-7) and [2] (NASA-9).

4.2.3 *heavy.dat*

If a neutral species is added to *species.xml*, the collision integrals for the pairs consisting of the new species and all other neutral species and the electron should be added to the file *heavy.dat* in the subdirectory **transport**. If the collision integrals are not added for a given species pair, then that pair will be neglected when computing transport properties in MUTATION^{++} .

4.3 Reaction Mechanisms

4.4 Mixtures

```
<mechanism name="air5">

    <arrhenius_units A="mol,cm,s,K" E="kcal,mol,K" />

    <reaction formula="N2+M=2N+M">
        <arrhenius A="3.0E+22" n="-1.6" T="113200.0" />
        <M>N2:0.2333, NO:0.2333, O2:0.2333</M>
    </reaction>

    <reaction formula="O2+M=2O+M">
        <arrhenius A="1.0E+22" n="-1.5" T="59360.0" />
        <M>N2:0.5, NO:0.5, O2:0.5</M>
    </reaction>

    <reaction formula="NO+M=N+O+M">
        <arrhenius A="5.0E15" n="+0.0" T="75500.0" />
        <M>NO:20.0, N:20.0, O:20.0</M>
    </reaction>

    <reaction formula="N2+O=NO+N">
        <arrhenius A="5.69E+12" n="+0.42" T="42938.0" />
    </reaction>

    <reaction formula="O2+N=NO+O">
        <arrhenius A="2.49E+09" n="+1.18" T="4005.5" />
    </reaction>

</mechanism>
```

Figure 5: Park's reaction mechanism for the Air-5 mixture (N, O, NO, N2, O2).

```
<mixture mechanism="air5" thermo_db="RRHO">

   <default_element_fractions>
      N  : 0.79,
      O  : 0.21
   </default_element_fractions>

   <species>
      N O NO N2 O2
   </species>

</mixture>
```

Figure 6: The Air-5 mixture defined in MUTATION^{++} format (*air5.xml*).

5 Example C++ Usage

5.1 Introduction

In order to use MUTATION^{++} , the header file *mutation++.h* must first be included.

```
#include "mutation++.h"
```

In MUTATION^{++} , core functionality is broken into three major groups: thermodynamics, kinetics, and transport. In order to simplify life for the user however, all functionality is grouped together via the `Mixture` class. Most likely, a user should never have to use any other object other than the mixture class to accomplish their purposes (except to adjust default mixture options). A `Mixture` object is created by passing the name of the desired mixture to the object's constructor.

```
Mixture mix("air11");
```

Upon instantiation, the mixture is loaded from the corresponding mixture data file (located in the **mixtures** directory) which must have the same name with the ".xml" extension. This process can be quite extensive as all the thermodynamic data for each species in the mixture is loaded and checked, if a reaction mechanism is given then it is loaded and validated, and all the collision integral information for every collision pair possible in the mixture is loaded.

Once instantiated, a mixture object can be used to compute any property of the mixture that is available. The most simple way to do this is to first set the current state of the mixture.

```
mix.setStateTPX(&T, &P, X);
```

The above line will set the current state of the mixture using a temperature, pressure, and mole fraction array (this is assuming a single temperature model). Once the mixture state is defined, any property can be derived from these basic thermodynamic properties. For instance, calling

```
double mu = mix.viscosity();
```

will set the variable mu equal to the viscosity of the mixture in the current state defined above. In short, the above is all that is necessary to compute mixture properties using MUTATION^{++} . Of course, there are more advance options available. For example, all mixture properties can also be computed using a function which takes the necessary arguments to define a state. This may be useful if you are interested in computing a finite-difference with respect to some state variable or if you simply don't want to set the state of the mixture before hand.

5.2 Chemical Equilibrium Example

In order to provide a more concrete example, a full program is given below which computes equilibrium properties for the 11 species air mixture and prints them to the screen. In addition, an example of using the `MixtureOptions` class is given to illustrate using non-default options.

```
#include "mutation++.h"
#include <iostream>

using namespace std;

int main()
{
    // Generate the default options for the air11 mixture
    MixtureOptions opts("air11");
    opts.setDefaultComposition()
        ("e-", 0.0)
        ("N",  0.8)
        ("O",  0.2);
    opts.setThermoDB("NASA-9");
    opts.setViscosityAlgorithm("LDLT");

    // Load the mixture with the new options
    Mixture mix(opts);

    // Loop over range of temperatures and compute equilibrium values at 1 atm
    double P = 101325.0;
    double T;
    for (int i = 0; i < 101; ++i) {
        T = 300.0 + static_cast<double>(i) * 50.0;
        mix.equilibrate(T, P);

        // Temperature
        cout << setw(10) << mix.T();

        // Species mole fractions
        for (int j = 0; j < mix.nSpecies(); ++j)
            cout << setw(10) << mix.X(j);

        // Other properties
        cout << setw(10) << mix.mixtureFrozenCpMass(); // Cp [J/kg-K]
        cout << setw(10) << mix.mixtureHMass();    // H [J/kg]
        cout << setw(10) << mix.mixtureSMass();    // S [J/kg-K]
    }

    return 0;
}
```

Figure 7: Simple program to compute equilibrium air properties using MUTATION^{++} .

One important feature to note in this short example is the use of the `equilibrate(T, P)` call. `equilibrate` is much like the `setStateXXX` functions except that instead of assigning a mixture composition, it is determined by computing the equilibrium composition. In the particular `equilibrate` call shown, the default elemental composition is used since none is given. In general, the default elemental composition is given in the mixture file, however it is over-ridden in the above example using the `MixtureOptions` object.

6 Models and Algorithms

6.1 Thermodynamics

There are multiple thermodynamic databases that are supported in Mutation^{++} . Below, each thermodynamic model is detailed.

Table 1: Available thermodynamic descriptors in Mutation^{++} .

Name	Description	Default
NASA-7	NASA 7-Coefficient Polynomial database	
NASA-9	NASA 9-Coefficient Polynomial database	
RRHO	Rigid-Rotator / Harmonic Oscillator model	✓

6.1.1 NASA 7-Coefficient Polynomials (NASA-7)

The NASA 7-coefficient polynomial thermodynamic database provides a set of polynomial coefficients for curve-fits computing standard state thermodynamic quantities. The standard state conditions are at 298.15 K and 1 bar (10^5 Pa). Using the polynomial coefficients, the value of ${C_P}^\circ_{,i}(T)$ can be determined via

$$\frac{{C_P}^\circ_{,i}(T)}{R_u} = a_{1,i} + a_{2,i}T + a_{3,i}T^2 + a_{4,i}T^3 + a_{5,i}T^4.$$

Other thermodynamic quantities are then determined using the standard relations of thermodynamics.

$$\begin{aligned}
\frac{H_i^\circ(T)}{R_uT} &= \frac{\int {C_P}^\circ_{,i}(T)dT}{R_uT} = a_{1,i} + \frac{1}{2}a_{2,i}T + \frac{1}{3}a_{3,i}T^2 + \frac{1}{4}a_{4,i}T^3 + \frac{1}{5}a_{5,i}T^4 + \frac{a_{6,i}}{T} \\
\frac{S_i^\circ(T)}{R_u} &= \int \frac{{C_P}^\circ_{,i}(T)}{R_uT}dT = a_{1,i}\ln T + a_{2,i}T + \frac{1}{2}a_{3,i}T^2 + \frac{1}{3}a_{4,i}T^3 + \frac{1}{4}a_{5,i}T^4 + a_{7,i} \\
\frac{G_i^\circ(T)}{R_uT} &= \frac{H_i^\circ(T) - TS_i^\circ(T)}{R_uT} = a_{1,i}(1-\ln T) - \frac{1}{2}a_{2,i}T - \frac{1}{6}a_{3,i}T^2 - \frac{1}{12}a_{4,i}T^3 - \frac{1}{20}a_{5,i}T^4 + \frac{a_{6,i}}{T} - a_{7,i}
\end{aligned}$$

6.1.2 NASA 9-Coefficient Polynomials (NASA-9)

6.1.3 Rigid-Rotor / Harmonic Oscillator (RRHO)

6.2 Transport

6.2.1 Viscosity

6.2.2 Thermal Conductivity

Table 2: Available mixture viscosity algorithms in MUTATION^{++} .

Name	Description	Default
Wilke	Uses the Wilke mixture formula	
Gupta-Yos	Uses the Gupta-Yos mixture formula	
CG	Solves the full linear system using Conjugate-Gradient method	✓
LDLT	Solves the full linear system using the LDL$^{\mathrm{T}}$ decomposition.	

Table 3: Available mixture thermal conductivity algorithms in MUTATION^{++} .

Name	Description	Default
Wilke	Uses the Wilke mixture formula	
CG	Solves the full linear system using Conjugate-Gradient method	✓

References

[1] Bonnie J. McBride, Sanford Gordon, and Martin A. Reno. Coefficients for Calculating Thermodynamic and Transport Properties of Individual Species. NASA TM-4513, 1993.

[2] Michael J. Zehe, Sanford Gordon, and Bonnie J. McBride. CAP: A Computer Code for Generating Tabular Thermodynamic Functions from NASA Lewis Coefficients. NASA TP-2001-210959/REV1, 2002.