**2014 CSM Modifications**

This file contains a brief summary of the modifications in the CSM program, as were made by Itay Zandbank in 2014.

1. GIT repository was created on Bitbucket, here: <https://bitbucket.org/teamcsm/csm/overview> . The old code, located on Amir Zait's SVN, was transferred to the new repository. The old code was tagged *PreviousCSM*.
2. The code was changed so it is compatible with C++11. For this the bleeding-edge OpenBabel was required. Precompiled binaries were created for Windows, and a script that fetches and compiles OpenBabel was created for Ubuntu and OS X. This was tagged *BleedingEdgeBabel*.
3. A simple test system was created. It is located in CSM/Testing. See the installation document for instructions how to run it.
4. The Permuter and PermuterGroup and Moledule "objects" were converted from C structures to C++ classes.  
   These conversions included replacing dynamically allocated buffers with std::vectors.
5. Throughout the code TRUE and FALSE macros were used. They were replaced with the C++ bools, true and false.
6. All the chemical elements were moved to elements.cpp and elements.h .
7. Logging was added with Boost::Log. Debug-level logging was also added.
8. The old code used NR Utils for various things. First the dmatrix and dvector utility functions were replaced with dmatrix and dvector C++ classes (in dmatrix.h and dvector.h). Since NR Utils' variant has 1-based indices, our C++ counterparts also have 1-based indices. That's why new classes were added, instead of using standard ones.  
   Internally dmatrix and dvector use std::vector.
9. CR Bond's rpoly was used for finding polynomial roots. After accepting CR Bond's approval to use the code, we kept using it. It was wrapped in math\_wrappers.h and math\_wrappers.cpp
10. NR Utils' tqli and tpred2 were used for finding eigenvectors and eigenvalues. Since NR Utils is not open source, the code was replaced by the Eigen C++ library (<http://eigen.tuxfamily.org/index.php?title=Main_Page>), with the MPL2 license. Eigen was also wrapped in math\_wrappers.h and .cpp
11. An MSI installation was created for Windows using WIX.
12. Split mainRot.cpp into several modules (both .h and .cpp for each):
    1. drand48 – the Windows implementation of the drand48 function
    2. options – all the command line options and accompanying flags
    3. math\_utils – Some mathematical functions labeled as math utils in mainRot.cpp
    4. PrintOuts – all the print\* functions
    5. calculations – the main calculations extracted from mainRot.cpp