Installation Manual for the PSI4 Program Package

T. Daniel Crawford,[a] C. David Sherrill,[b] and Edward F. Valeev[a]

[a] *Department of Chemistry, Virginia Tech, Blacksburg, Virginia 24061-0001*

[b] *Center for Computational Molecular Science and Technology, Georgia Institute of Technology, Atlanta, Georgia 30332-0400*

PSI4 Version: 4.0.0-alpha
Created on: July 20, 2011

1 Compilation Prerequisites

The following external software packages are needed to complile `PSI4`:

- C, C++, and FORTRAN77 compilers. The FORTRAN77 compiler is only used to determine the symbol-naming convention of and some system routines for the BLAS and LAPACK libraries on some architectures. It is optional in a few cases (e.g. Mac OS X systems).
- A well-optimized basic linear algebra subroutine (BLAS) library for vital matrix-matrix and matrix-vector multiplication routines. (See recommendations below.)
- The linear algebra package (LAPACK). `PSI4` makes use of LAPACK's eigenvalue/eigenvector and matrix inversion routines. (See recommendations below)
- POSIX threads (Pthreads) library
- Perl interpreter (version 5.005 or higher)
- Python interpreter (2.4 or higher, but not version 3.x; needed for input files)
- Python developer libraries corresponding to your interpreter
- A version of MPI is currently required; MPICH2 is recommended. (Note: MPICH2-1.1.0 had trouble with some header files. MPICH2-1.2.1 seems to fix it.)
- Various GNU utilies: `www.gnu.org`
  - `autoconf (version 2.52 or higher)`
  - `aclocal`
  - `make`
  - `flex`
  - `bison`
  - `fileutils` (esp. `install`)
- For documentation only:
  - `LaTeX`
  - `LaTeX2html` (v0.99.1 or 1.62, including the patch supplied in psi3/misc)

2 Brief Summary of Configuration, Compilation, and Installation

A good directory for the PSI4 source code is /usr/local/src/psi3. The directory should *not* be named /usr/local/psi, as that is the default installation directory unless changed by the --prefix directive (see below). It should also not have any periods in the path, e.g., /usr/local/psi3.2, because of a bug in dvips which will cause the compilation of documentation to fail.

The following series of steps will configure and build the PSI4 package and install the executables in /usr/local/psi/bin:

1. cd $PSI4 (your top-level PSI4 source directory)
2. aclocal
3. autoconf
4. mkdir objdir
5. cd objdir
6. ../configure --with-cxx=mpicxx (may need some of the options below, esp. if blas or lapack are in non-standard locations).
7. make
8. make tests (optional, but recommended)
9. make install
10. make doc (optional)

You may need to make use of one or more of the following options to the configure script:

- --prefix=directory — Use this option if you wish to install the PSI4 package somewhere other than the default directory, /usr/local/psi. This directory will contain subdirectories with the final installed binaries, libraries, documentation, and shared data files.
- --with-cc=compiler — Use this option to specify a C compiler. One should use compilers that generate reentrant code, if possible. The default search order for compilers is: cc_r (AIX only), gcc, icc, cc.
- --with-cxx=compiler — Use this option to specify a C++ compiler. One should use compilers that generate reentrant code, if possible. The default search order for compilers is: xlC_r (AIX only), g++, c++, icpc, cxx. At the moment, MPI is required and you need to use mpicxx (where this has been added to your PATH).

- `--with-fc=compiler` — Use this option to specify a Fortran-77 compiler, which is used to determine linking coventions for BLAS and LAPACK libraries and to provide system routines for those libraries. Note that no fortran compiler is necessary on Mac OS X systems (see below). The default search order for compilers is: `xlf_r` (AIX only), `gfortran`, `g77`, `ifort`, `f77`, `f2c`.

- `--with-f77-symbol=value` — This option allows manual assignment of the F77 symbol convention, which is necessary for C programs to link Fortran-interface libraries such as BLAS and LAPACK. This option should only be used by experts and even then should almost never be necessary. Allowed values are:

lc	lower-case
lcu	lower-case with underscore (default)
uc	upper-case
ucu	upper-case with underscore

- `--with-ld=linker` — Use this option to specify a linker program. The default is `ld`.

- `--with-ranlib=ranlib` — Use this option to specify a ranlib program. The default behavior is to detect an appropriate choice automatically.

- `--with-ar=archiver` — Use this option to specify an archiver. The default is to look for `ar` automatically.

- `--with-ar-flags=options` — Use this option to specify archiver command-line flags. The default is `r`.

- `--with-incdirs=directories` — Use this option to specify extra directories where to look for header files. Directories should be specified prepended by `-I`, i.e. `-Idir1 -Idir2`, etc. If several directories are specified, enclose the list with single right-quotes, e.g., `--with-incdirs='-I/usr/local/include -I/home/psi3/include'`.

- `--with-libs=libraries` — Use this option to specify extra libraries which should be used during linking. Libraries should be specified by their full names or in the usual `-l` notation, i.e. `-lm /usr/lib/libm.a`, etc. If several libraries are specified, enclose the list with single right-quotes, e.g., `--with-libs='-lcompat /usr/local/lib/libm.a'`.

- `--with-libdirs=directories` — Use this option to specify extra directories where to look for libraries. Directories should be specified prepended by `-L`, i.e. `-Ldir1 -Ldir2`, etc. If several directories are specified, enclose the list with single right-quotes, e.g., `--with-libdirs='-L/usr/local/lib -I/home/psi3/lib'`.

- `--with-blas=library` — Use this option to specify a BLAS library. If your BLAS library has multiple components, enclose the file list with single right-quotes, e.g., `--with-blas='-lf77blas -latlas'`. Note that many BLAS libraries can be detected automatically.

- `--with-lapack=library` — Use this option to specify a LAPACK library. If your LAPACK library has multiple components, enclose the file list with single right-quotes, e.g., `--with-lapack='-llapack -lcblas -latlas'`. note that many LAPACK libraries can be detected automatically.

- `--with-max-am-eri=integer` — Specifies the maximum angular momentum level for the primitive Gaussian basis functions when computing electron repulsion integrals. This is set to g-type functions (AM=4) by default.

- `--with-max-am-deriv1=integer` — Specifies the maximum angular momentum level for first derivatives of the primitive Gaussian basis functions. This is set to f-type functions (AM=3) by default.

- `--with-max-am-deriv2=integer` — Specifies the maximum angular momentum level for second derivatives of the primitive Gaussian basis functions. This is set to d-type functions (AM=2) by default.

- `--with-max-am-r12=integer` — Specifies the maximum angular momentum level for primitive Gaussian basis functions used in r_{12} explicitly correlated methods. This is set to f-type functions (AM=3) by default.

- `--with-debug=yes/no` — This option turns on debugging options. This is set to `no` by default.

- `--with-opt=options` — Turn off compiler optimizations if `no`. This is set to `yes` by default.

- `---with-strict=yes` – Turns on strict compiler warnings.

3 Detailed Installation Instructions

This section provides detailed instructions for compiling and installing the `PSI4` package.

3.1 Step 1: Configuration

First, we recommend that you choose for the top-level `$PSI4` source directory something other than `/usr/local/psi`; your `$HOME` directory or `/usr/local/src/psi4` are convenient choices. Next, in the top-level `$PSI4` source directory you've chosen, first run `autoconf` to generate the configure script from `configure.ac`. It is best to keep the source code separate from the compilation area, so you must choose a subdirectory for compilation of the codes. A simple option is `$PSI4/objdir`, which should work for most environments. However, if you need executables for several architectures, choose more meaningful subdirectory names.

• The compilation directory will be referred to as `$objdir` for the remainder of these instructions.

In `$objdir`, run the configure script found in the `$PSI4` top-level source directory. This script will scan your system to locate certain libraries, header files, etc. needed for complete compilation. The script accepts a number of options, all of which are listed above. The most important of these is the `--prefix` option, which selects the installation directory for the executables, the libraries, header files, basis set data, and other administrative files. The default `--prefix` is `/usr/local/psi`.

• The configure script's `--prefix` directory will be referred to as `$prefix` for the remainder of these instructions.

3.2 Step 2: Compilation

Running `make` (which must be GNU's `'make'` utility) in `$objdir` will compile the `PSI4` libraries and executable modules.

3.3 Step 3: Testing

To execute automatically the ever-growing number of test cases after compilation, simply execute "make tests" in the `$objdir` directory. This will run each (relatively small) test case and report the results. Failure of any of the test cases should be reported to the developers at psicode@users.sourceforge.net. By default, any such failure will stop the testing process. If you desire to run the entire testing suit without interruption, execute "make tests TESTFLAGS='-u -q' ". Note that you must do a "make testsclean" in `$objdir` to run the test suite again.

3.4 Step 4: Installation

Once testing is complete, installation into $prefix is accomplished by running `make install` in `$objdir`. Executable modules are installed in `$prefix/bin`, libraries in `$prefix/lib` and basis set data and other control strctures `$prefix/share`.

3.5 Step 5: Documentation

If your system has the appropriate utilities, you may build the package documentation from the top-level `$objdir` by running `make doc`. The resulting files will appear in the `$prefix/doc` area.

3.6 Step 6: Cleaning

All compilation-area object files and libraries can be removed to save disk space by running `make clean` in `$objdir`.

3.7 Step 7: User Configuration

After the PSI4 package has been successfullly installed, the user will need to add the installation directory into their path. If the package has been installed in the default location /usr/local/psi3, then in C shell, the user should add something like the following to their .cshrc file:

```
setenv PSI /usr/local/psi3
set path = ($path $PSI/bin)
setenv MANPATH $PSI/doc/man:$MANPATH
```

The final line will enable the use of the PSI4 man pages.

4 Recommendations for BLAS and LAPACK Libraries

Much of the speed and efficiency of the PSI4 programs depends on the corresponding speed and efficiency of the available BLAS and LAPACK libraries (especially the former). In addition, the most common compilation problems involve these libraries. Users may therefore wish to consider the following BLAS and LAPACK recommendations when building PSI4:

- It is NOT wise to use the stock BLAS library provided with many Linux distributions like RedHat. This library is usually just the netlib (http://netlib.org/distribution and is completely unoptimized. PSI4's performance will suffer if you choose this route. The choice of LAPACK is less critical, and so the unoptimized netlib distribution is acceptable. If you do choose to use the RedHat/Fedora stock BLAS and LAPACK, be aware that some RPM's do not make the correct symbolic links. For example, you may have /usr/lib/libblas.so.3.1.0 but not /usr/lib/libblas.so. If this happens, create the link as, e.g., ln -s /usr/lib/libblas.so.3.1.0 /usr/lib/libblas.so. You may need to do similarly for lapack.

- Perhaps the best choices for BLAS are Kazushige Goto's hand-optimized BLAS (http://www.tacc.utexas.edu/resources/software/) and ATLAS (http://math-atlas.sourceforge.net/). These work well on nearly every achitecture to which the PSI4 developers have access, though we have identified at least one case in which the Goto libraries yielded faulty DGEMM call. On Mac OS X systems, however, the vecLib package that comes with Xcode works well. Note also that we have encountered problems with the version 10 of Intel's MKL, particularly for very large coupled cluster calculations.

- PSI4 does not require a Fortran compiler, unless the resident BLAS and LAPACK libraries require Fortran-based system libraries. If you see compiler complaints about missing symbols like "do_fio" or "e_wsfe" then your libraries were most likely compiled

with g77 or gfortran, which require `-lg2c` to resolve the Fortran I/O calls. Use of the same gcc package for PSI4 should normally resolve this problem.

- The PSI4 configure script can often identify and use several different BLAS and LAPACK libraries, but its ability to do this automatically depends on a number of factors, including correspondence between the compiler used for PSI4 and the compiler used to build BLAS/LAPACK, and placement of the libraries in commonly searched directories, among others. PSI4's configure script will find your BLAS and LAPACK if any of the the following are installed in standard locations (e.g. `/usr/local/lib`):
  - ATLAS: `libf77blas.a` and `libatlas.a`, plus netlib's `liblapack.a`
  - MKL 8: `libmkl.so` and `libmkl_lapack64.a` (with the corresponding Intel compilers)
  - Goto: `libgoto.a` and netlib's `liblapack.a`
  - Cray SCSL (e.g. on SGI Altix): `libscs.so` (NB: No Fortran compiler is necessary in this case, so `--with-fc=no` should work.)
  - ESSL (e.g. on AIX systems): `libessl.a`

- If configure cannot identify your BLAS and LAPACK libraries automatically, you can specify them on the command-line using the `--with-blas` and `--with-lapack` arguments described above. Here are a few examples that work on the PSI4 developers' systems:

  (a) Linux with ATLAS:

  ```
  --with-blas='-lf77blas -latlas' --with-lapack='-llapack -lcblas'
  ```

  (b) Mac OS X with vecLib:

  ```
  --with-blas='-altivec -framework vecLib' --with-lapack=' '
  ```

  (c) Linux with MKL 8.1 and `icc/icpc/ifort` 9.1:

  ```
  --with-libdirs=-L/usr/local/opt/intel/mkl/8.0.2/lib/32 --with-blas=-lmkl
  --with-lapack=-lmkl_lapack32
  ```

  (d) Linux on ia32 with MKL 10.1 and `icc/icpc` 11.0:

```
--with-blas='-Wl,--start-group -L/usr/local/opt/intel/mkl/10.1.0.015/lib/32
-lmkl -Wl,--end-group -lguide -lpthread'
```

4.1 Compilation notes for ATLAS

These shortcut notes might be helpful if you are using Linux. However, we recommend reading and following the full ATLAS installation notes.

- You'll need a Fortran compiler installed.

- Unpack the source code, then make a compilation directory (could be an `obj` subdirectory in the source directory, or elsewhere).
- Turn off CPU throttling so the auto-tuning capabilities have a chance to work. On Linux, this can be tune using

  ```
  /usr/bin/cpufreq-selector -g performance
  ```

- `cd` into the compilation directory and run the source directory configure script there, with any necessary flags, e.g.,

  ```
  /usr/local/src/atlas/configure --prefix=/usr/local/atlas
  ```

  where `prefix` gives the installation directory. It should automatically detect if you're on an `x86_64` architecture and use 64-bit addressing (`-b 64` flag) if so.
- Then make and check using

  ```
  make; make check; make ptcheck
  ```

- And install

  ```
  make install
  ```

4.2 Compilation notes for Netlib's LAPACK

These shortcut notes might be helpful if you are using Linux. However, we recommend reading and following the full LAPACK installation notes.

- You'll need a Fortran compiler installed.
- If you decide to compile LAPACK from source, it may be obtained from `http://www.netlib.org/lapack/`. Unpack the source code, and in the top-level source directory, you need to create a make.inc file with the appropriate options for your machine. For Linux/gfortran, simply

  ```
  cp make.inc.example make.inc
  ```

- Next, edit `BLASLIB` in `make.inc` to point to your BLAS library (full pathnames are recommended), e.g.,

  ```
  BLASLIB = /home/david/software/atlas3.9.25/lib/libf77blas.a
  /home/david/software/atlas3.9.25/lib/libatlas.a
  ```

- Edit Makefile as necessary (probably not needed).
- `make`
- Copy the resulting file [`lapack_($ARCH).a`] where you want it (a standard location like `/usr/local/lib` is easier for PSI to find). It is probably helpful to rename the file `liblapack.a`.

5 Miscellaneous architecture-specific notes

- Linux on x86 and x86_64:
  - `gcc` compiler: versions 3.2, 3.3, 3.4, 4.0, and 4.1 have been tested.
  - Intel compilers: versions 9.0 and 11.0 have been tested. We do not recommend using version 8.1.
  - Portland Group compilers: version 6.0-5 has been tested.
  - Some versions of RedHat/Fedora Core RPM packages for the BLAS and LAPACK libraries fail to make all the required symlinks. For example, you may have `/usr/lib/libblas.so.3.1.0` but not `/usr/lib/libblas.so`. If this happens, create the link as, e.g., `ln -s /usr/lib/libblas.so.3.1.0 /usr/lib/libblas.so`. You may need to do something similar for lapack.
- Linux on Itanium2 (IA64):
  - Intel compiler versions 9.0 and 10.0 have been tested and work. Version 8.1 does not work.
  - `gcc` compilers work.
- Mac OS 10.x:
  - The compilation requires a developer's toolkit (Xcode) from `apple.com`. Note that a fortran compiler is not needed for PSI 3.4 on Mac OS X systems.
  - The `libcompat.a` library is no longer needed as of 1/24/2008.
  - For apple systems, the latest configure script assumes that the `vecLib` will be used for the optimized BLAS and LAPACK libraries, unless the user indicates otherwise using the `--with-blas` and `--with-lapack` flags to configure. If you encounter difficulty with configure, you may have success explicitly indicating the vecLib using:

    `--with-blas='-altivec -framework vecLib' --with-lapack=' '`
  - Pre Mac OS 10.4: Certain PSI4 codes require significant stackspace for compilation. Increase your shell's stacksize limit before running `make`. For csh, for example, this is done using "unlimit stacksize." [NB: This limit appears to have been lifted starting with Mac OS 10.3.X (Panther).]
- AIX in a 64-bit environment:

  We do not presently support the use of XL compilers on AIX systems. We have tested gcc-4.1.1 under AIX 5.3, and we recommend use of the configure flag: `--with-aix64`.
- SGI IRIX 6.x:
  - MIPSpro C++ compilers prior to version 7.4 require a command-line flag '`-LANG:std`' in order to compile `PSI4` properly.

- Use command-line flag '`-64`' in order to produce 64-bit `PSI4` executables with MIPSpro compilers. The following is an example of appropriate configure options:

```
--with-cc='cc -64' --with-cxx='CC -64 -LANG:std' --with-fc='f77 -64'
```

- Under IRIX configure will attempt to detect automatically and use the optimized SGI Scientific Computing Software Library (SCSL).

- Compaq Alpha/OSF 5.1: default shell (`/bin/sh`) is not POSIX-compliant which causes some `PSI4` makefiles to fail. Set environmental variable `BIN_SH` to `xpg4`.