http://www.scipy.org/Installing_SciPy/BuildingGeneral

http://www.scipy.org/Installing_SciPy

Building everything from source with gfortran on Ubuntu (Nov 2010)

These are instructions for building everything from source on a 64 bit Ubuntu system (Maverick: 10.10) on a multicore processor using the latest versions as of November 2010. Everything is installed in a user directory structure in $HOME/local (/home/sam/local in my case). Administrator priviliges are required only in the beginning to disable CPU throttling while building ATLAS.

Install required packages

`sudo apt-get install` **`build-essential`** `(dpkg-dev (>= 1.13.5)`

Debian package development tools

- dep: g++ (>= 4:4.4.3)
  The GNU C++ compiler

- dep: gcc (>= 4:4.4.3)
  The GNU C compiler

- dep: libc6-dev
  Embedded GNU C Library: Development Libraries and Header Files

or libc-dev
  virtual package provided by libc6-dev

- dep: make
  An utility for Directing compilation.

`)`

`python-dev` **`swig gfortran`** `python-nose`

Step 1: Disable CPU Throttling

ATLAS' timing algorithm require CPU throttling to be disabled. This disables it on the 0th core:

```
sudo cpufreq-selector -g performance
```

Then disable it on each additional core. For a quad core processor, these commands will be required:

```
sudo cp /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
/sys/devices/system/cpu/cpu1/cpufreq/scaling_governor
sudo cp /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

```
/sys/devices/system/cpu/cpu2/cpufreq/scaling_governor
sudo cp /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
/sys/devices/system/cpu/cpu3/cpufreq/scaling_governor
```

Step 2: Build ATLAS(3.9.32) with complete Lapack(3.2.2)

Download lapack.tgz from netlib.org and atlas3.9.32.tar.bz2. Extract the atlas archive into a directory named ATLAS and from within it, issue these commands:

```
mkdir BUILD
cd BUILD
../configure -b 64 -Fa alg -fPIC --with-netlib-lapack-tarfile=<path to lapack.tgz>
--prefix=/home/sam/local
make
cd lib
make shared
make ptshared
cd ..
make install
```

Note that make ptshared might not work on a single core machine. Note also that the first "make" command above will take several hours to run, as ATLAS optimizes various performance parameters.

Step 3: Build UMFPACK (5.5.0) with AMD (2.2.1), UFConfig(3.5.0) and CHOLMOD (1.7.1)

Dowload all four packages and extract them in the same directory. Edit UFconfig/UFconfig.mk to read:

```
CC = gcc
CFLAGS = -O3 -fexceptions -m64 -fPIC

F77 = gfortran
F77FLAGS = -O -m64 -fPIC

INSTALL_LIB = /home/sam/local/lib
INSTALL_INCLUDE = /home/sam/local/include

METIS_PATH =
METIS =

CHOLMOD_CONFIG = -DNPARTITION
```

Then issue the following commands

```
cd UMFPACK
make library
make install
cd ../AMD
make install
cd ../UFconfig
cp UFconfig.h /home/sam/local/include
```

Step 4: Build numpy(1.5.0)

Untar the archive, copy site.cfg.example to site.cfg and edit it:

```
[DEFAULT]
library_dirs = /home/sam/local/lib
include_dirs = /home/sam/local/include
```

In the same file, uncomment these lines:

```
[blas_opt]
libraries = ptf77blas, ptcblas, atlas

[lapack_opt]
libraries = lapack, ptf77blas, ptcblas, atlas

[amd]
amd_libs = amd

[umfpack]
umfpack_libs = umfpack
```

For a single core machine, uncomment these lines:

```
[blas_opt]
libraries = f77blas, cblas, atlas
[lapack_opt]
libraries = lapack, f77blas, cblas, atlas
```

Then use the standard installation technique

```
python setup.py build
python setup.py install --prefix=/home/sam/local
```

Step 5: Build scipy(0.8.0)

Make sure that $HOME/local/bin is in $PATH (for f2py) and $PYTHONPATH contains $HOME/local/lib/python-2.6/site-packages (for numpy)

Do a standard install

```
python setup.py build
python setup.py install --prefix=/home/sam/local
```

Building Scipy

Table of contents

Contents

Introduction

The SciPy project provides two packages: NumPy (formerly known as core SciPy) and full SciPy. The NumPy package is a replacement of Numeric and should be installed before building the full SciPy package. Installing both packages is simple. In most cases simply executing

```
python setup.py install
```

in the `numpy` and `scipy` directories will install each package. The rest of this document concentrates on configuring, building, installing, or/and using various external software that Scipy can use for extra power.

Note that to avoid problems, it can be a good idea to remove the local `build` directory (if any) and existing Scipy installations, before building and installing a new version built from sources. Also note that the use of `pip` and especially `easy_install` is not recommended, as these tools often have problems where the standard `python setup.py install` does not.

Python

To build SciPy, Python version 2.4 or newer is required. Make sure that the Python package `distutils` is installed before continuing. For example, in Debian GNU/Linux, `distutils` is included in the `python-dev` package.

If while building SciPy, "error: pyconfig.h: No such file or directory" is encountered, ensure that Python is installed a directory that is different from its source.

Compilers

To build any extension modules for Python, you'll need a C compiler.

Various SciPy modules use Fortran 77 libraries and some use C++, so you'll also need Fortran 77 and C++ compilers installed. The SciPy module Weave uses a C++ compiler at run time.

Note that SciPy is developed mainly using GNU compilers. Compilers from other vendors such as

Intel, Absoft, Sun, NAG, Compaq, Vast, Porland, Lahey, HP, IBM are supported in the form of community feedback.

gcc 3.x compilers are recommended. Note that code compiled by the Intel Fortran Compiler (IFC) is not binary compatible with code compiled by g77. Therefore, when using IFC, all Fortran codes used in SciPy must be compiled with IFC. This also includes the LAPACK, BLAS, and ATLAS libraries. Using GCC for compiling C code is OK. IFC version 5.0 is not supported.

To build NumPy, only a C compiler is required.

If you see an error message

```
ImportError: /usr/lib/atlas/libblas.so.3gf: undefined symbol:
_gfortran_st_write_done
```

when building **SciPy**, it means that **NumPy** picked up the wrong Fortran compiler during build (e.g. ifort). Recompile NumPy using:

```
python setup.py build --fcompiler=gnu95
```

or whichever is appropriate (see `python setup.py build --help-fcompiler`).

Linear Algebra libraries

Various SciPy packages do linear algebra computations using the LAPACK routines. SciPy's setup.py scripts can use number of different LAPACK library setups, including optimized LAPACK libraries such as ATLAS or the Accelerate/vecLib framework on OS X. The following notes give detailed information on how to prepare the build environment so that SciPy's setup.py scripts can use whatever LAPACK library setup one has.

NumPy does not require any external linear algebra libraries to be installed. However, if these are available, NumPy's setup script can detect them and use them for building. If you do not plan to build the full SciPy then you can skip this section.

Building BLAS library from Netlib

Download BLAS sources from Netlib, build `libfblas.a` library, and set environment variable `BLAS`:

```
mkdir -p ~/src/
cd ~/src/
wget http://www.netlib.org/blas/blas.tgz
tar xzf blas.tgz
cd BLAS
# NOTE: The selected fortran compiler must be consistent for BLAS, LAPACK, NumPy,
and SciPy.
# For GNU compiler on 32-bit systems:
g77 -O2 -fno-second-underscore -c *.f                     # with g77
gfortran -O2 -std=legacy -fno-second-underscore -c *.f    # with gfortran
# OR for GNU compiler on 64-bit systems:
g77 -O3 -m64 -fno-second-underscore -fPIC -c *.f                     # with g77
gfortran -O3 -std=legacy -m64 -fno-second-underscore -fPIC -c *.f    # with
gfortran
# OR for Intel compiler:
ifort -FI -w90 -w95 -cm -O3 -unroll -c *.f
```

```
# Continue below irrespective of compiler:
ar r libfblas.a *.o
ranlib libfblas.a
rm -rf *.o
export BLAS=~/src/BLAS/libfblas.a
```

See the note at the end of the section below on building LAPACK.

Building LAPACK library from Netlib

Download LAPACK sources from Netlib, build `libflapack.a` library, and set environment variable `LAPACK`:

```
mkdir -p ~/src
wget http://www.netlib.org/lapack/lapack.tgz
tar xzf lapack.tgz
cd ~/src/LAPACK
cp INSTALL/make.inc.gfortran make.inc          # on Linux with lapack-3.2.1 or
newer
cp INSTALL/make.inc.LINUX make.inc             # on Linux with older lapack
# Edit make.inc as follows:
# For GNU compiler on 32-bit Linux (these are the defaults):
PLAT = _LINUX
OPTS = -O2
# OR for GNU compiler on 64-bit Linux:
PLAT = _LINUX
OPTS = -O2 -m64 -fPIC
NOOPT = -m64 -fPIC
# OR for Absoft (8.x or later):
PLAT = _LINUX
OPTS = -O3 -YNO_CDEC
# OR for Intel Fortran compiler on Linux:
wget http://www.scipy.org/download/misc/make.inc.LINUX_IFC
cp make.inc.LINUX_IFC make.inc
# Continue below irrespective of compiler:
make lapacklib
make clean
cp lapack_LINUX.a libflapack.a                 # on Linux
export LAPACK=~/src/LAPACK/libflapack.a
```

In the event that SciPy is unable to find the `libf*` names, a symbolic link can be made from these files to `libblas.a` and `liblapack.a`. NumPy is not expected to have this problem.

Further Instructions

More detailed platform-specific instructions for building and installing SciPy can be found on the Installing SciPy page.

Installing SciPy / Mac OS X

Table of contents

Contents

These instructions describe how to install NumPy and SciPy from source. If you would rather install pre-built binaries, go to the Download page and install the packages linked from there.

Python

Apple ships its own version of Python with OS X. However, we *strongly* recommend installing the official Python distribution.

If you are missing readline support for your installation of Python, I recommend following these instructions for getting it installed with relative ease.

Apple's Developer Tools

Apple's Developer Tools provide a number of key libraries, particularly the vecLib Framework , which includes BLAS and LAPACK. Install the most recent version from Apple's Developer Connection site (after free registration). The most recent version may also be included on your OS X installation CD. Ensure that all components are installed by choosing **customize** when available during the install process and selecting all optional packages (at least the X11 development tools and the 10.4 SDK).

On OS X 10.6 the default gcc version is 4.2, while NumPy and SciPy need to be built with 4.0 (at least in combination with the Python from python.org). For gcc the correct version should be picked up automatically, for C++ code (only in SciPy) you should ensure that g++ and c++ default to 4.0. One way to achieve this is to create symlinks

```
$ ln -s /usr/bin/g++-4.0 g++
$ ln -s /usr/bin/g++-4.0 c++
```

and add these to the front of your PATH.

When building with Python 2.5 on OS X 10.6 it is also necessary to do

```
$ export CC=/usr/bin/gcc-4.0
```

FORTRAN

Though virtually any commercial C/C++ compiler may be used with SciPy, OS X come with GNU C compilers preinstalled. The only thing missing is the GNU FORTRAN compiler.

Binaries of gfortran (GNU F95, this is a version of the much awaited, free, open source, F95 compiler) are available from this site (download universal binary). We strongly recommend to use this exact build, many other grfortran versions/builds have created problems in the past.

Obtaining and Building NumPy and SciPy

You may install NumPy and SciPy either by checking out the source files from the Git/SVN repositories, or unpacking them from a source archive file from the Download page. If you choose the latter, simply expand the archive (generally a gzipped tar file), otherwise check out the following branches from the repository:

```
$ git clone https://github.com/numpy/numpy.git
$ svn co http://svn.scipy.org/svn/scipy/trunk scipy
```

Both NumPy and SciPy are built as follows:

```
$ python setup.py build
$ python setup.py install
```

or, using scons,

```
$ python setupscons.py scons --jobs=2
```

The above applies to the official Python distribution, which is 32-bit only for 2.5/2.6/3.1 while 32/64-bit bundles are available for 2.7. For alternative 64-bit Pythons (either from Apple or home-built) on Snow Leopard, you may need to extend your build flags to specify the architecture by setting LDFLAGS and FFLAGS.

Note that with distutils (setup.py) given build flags like LDFALGS do not extend but override the defaults, so you have to specify all necessary flags. Only try this if you know what you're doing! Numscons does extend the flags, so you can for example use the build command:

```
LDFLAGS="-arch x86_64" FFLAGS="-arch x86_64" python setupscons.py scons
```

After a successful build, you may try running the built-in unit tests for SciPy:

```
python
>>> import numpy
>>> numpy.test('1','10')
>>> import scipy
>>> scipy.test('1','10')
```

Be sure **not** to import numpy or scipy while you're in the numpy/scipy source tree. Change directory first.

If you have any problems installing SciPy on your Mac based on these instructions, please check the scipy-users and scipy-dev mailing list archives for possible solutions. If you are still stuck, feel free to join scipy-users for further assistance. Please have the following information ready:

- Your OS version
- The versions of gcc and gfortran and where you obtained gfortran
  - `$ gcc --version`
  - `$ gfortran --version`
- The versions of numpy and scipy that you are trying to install
- The full output of `$ python setup.py build`

CategoryInstallation