Pyomo Installation Guide 4.0

COLLABORATORS			
	TITLE : Pyomo Installation Guide 4.0		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	William E. Hart, Carl D. Laird, John Siirola, Jean-Paul Watson, and David L. Woodruff	May 23, 2016	

Contents

Preface

This book describes different methods for installing the Pyomo software. Pyomo is a Python software package that supports a diverse set of optimization capabilities for formulating and analyzing optimization models. This capability is commonly associated with algebraic modeling languages (AMLs), which support the description and analysis of mathematical models with a high-level language.

Goals of the Book

Unfortunately, Pyomo is a complex software package that can be difficult to install. Pyomo is a Python package that integrates a variety of subpackages. Pyomo also depends on a variety of third-party Python packages, some of which have very different installation processes on different computer operating systems. Furthermore, Pyomo can execute third-party optimization solvers, whose installation and configuration is completely independent of Pyomo.

In this book, we outline a variety of installation options for different computer operating systems and for different usage models. These installation options reflect the different ways that Pyomo developers and users have used Pyomo, and they account for user access to system resources.

Comments and Questions

Further information about Pyomo and Pyomo is available a the Pyomo home page:

`http://pyomo.org`

The Pyomo software is developed on the following Trac site:

`https://software.sandia.gov/trac/pyomo`

Pyomo is also hosted at COIN-OR:

`https://projects.coin-or.org/pyomo`

We strongly encourage feedback from readers about the software on the Pyomo Forum:

`pyomo-forum@googlegroups.com`

We hope this will include feedback on typos and errors in our examples in this book.

Good Luck!

Chapter 1

Quick Start

This section provides a quick guide for installing and using the latest Pyomo release. Note that the following installation instructions assume that the user has network access. Strategies for off-line installation are described later in this book.

1.1 Installing Pyomo

1.1.1 Installing with a Scientific Python Distribution

Linux, Mac OS/X and other Unix variants typically have Python pre-installed. However, scientific Python distributions that contain the SciPy Stack include many utilities that Pyomo users will find useful, including SciPy optimizers and MatplotLib plotting capabilities. See SciPy's list of scientific Python distributions.

Step 1: Install Python Many scientific Python distributions are available on Linux, Windows and Mac. Download and execute an installer that is suitable for your platform.

Step 2: Install Pyomo Scientific Python distributions include the `pip` package that is used to download and install the latest Pyomo release. The Python installation includes a `bin` or `Scripts` directory that includes a `pip` script. Add this directory to your `PATH` so you can execute the `pip` script. If you have administrator access, then you can install Pyomo in your system Python installation by executing the following in a shell:

```
pip install Pyomo
```

You can also install Pyomo in a user's `HOME` directory. This does not require administrator access, but the Pyomo package appears to be installed with the system Python. Execute the following in a shell:

```
pip install --user Pyomo
```

1.1.2 Installing with a Standard Python Distribution

Linux, Mac OS/X and other Unix variants typically have Python pre-installed, and the `python` command will typically be available without modifying your `PATH` environment.

Step 1: Install pip For Python versions before 3.4, the `pip` package wll need to be explicitly installed. On Linux, you may be able to install `pip` with a package manager like `yum` or `apt-get`. Alternatively, you can securely download the get-pip.py script using your web browser (or a command-line tool like `curl`). Then, you execute the following in a shell:

```
python get-pip.py
```

Step 2: Install Pyomo If you have administrator access, then you can install Pyomo in your system Python installation by executing the following in a shell:

```
pip install pyomo
```

You can also install Pyomo in a user's HOME directory. This does not require administrator access, but the Pyomo package appears to be installed with the system Python. Execute the following in a shell:

```
pip install --user pyomo
```

1.2 Installing Auxilliary Software

Pyomo has conditional dependencies on a variety of third-party packages. These are not installed with Pyomo, and many of them can be installed by installing the `pyomo.extras` package:

```
pip install pyomo.extras
```

Note
Pyomo has a conditional dependency on the `numpy` and `scipy` packages. If you explicitly install `numpy` or `scipy`, then `pyomo.extras` should be installed afterward.

Pyomo does not include any stand-alone optimization solvers. Consequently, most users will want to install third-party solvers to analyze optimization models built with Pyomo. The command

```
pyomo help -s
```

provides documentation about Pyomo's solver interfaces, and it will dynamically check for available solvers.

Note
Users with network access can remotely optimize Pyomo models on the NEOS server, so optimization solvers do not need to be installed to get started with Pyomo.

1.3 Getting Started

1.3.1 Running Python

If the `PATH` environment has been configured to include the system Python installation, then the Pyomo scripts can be executed without further configuration. The simplest way to get started with Pyomo is to import the `pyomo` package that you wish to work with. For example, you would import `pyomo.core` to use Pyomo:

```
\$ python
Python 2.7.5 (default, Aug  1 2013, 01:01:17)
[GCC 4.2.1 Compatible Apple Clang 4.1 ((tags/Apple/clang-421.11.66))] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import pyomo.core
>>>
```

See Getting Started with Pyomo for further discussion of how to use Pyomo packages.

1.3.2 Optimizing with a NEOS Solver

As a quick test of your installation, you can used Pyomo to analyze the diet problem. Download the `diet.py` model file and `diet1.dat` data file here: https://software.sandia.gov/svn/public/pyomo/pyomo/trunk/examples/pyomo/diet.

If you have network access, then you can run the `pyomo` command to optimize this model:

```
pyomo solve --solver-manager=neos --solver=cbc diet1.py diet.dat
```

You should get an output like the following:

```
[    0.00] Setting up Pyomo environment
[    0.00] Applying Pyomo preprocessing actions
[    0.00] Creating model
[    0.03] Applying solver
[    0.05] Processing results
    Number of solutions: 1
    Solution Information
      Gap: 0.0
      Status: optimal
      Function Value: 2.81
    Solver results file: results.json
[    0.14] Applying Pyomo postprocessing actions
[    0.14] Pyomo Finished
```

1.3.3 Optimizing with a Local Solver

The `pyomo` command can also use a solver that is installed locally. For example, if you have installed the `glpk` software, then you can optimize this model as follows:

```
pyomo solve --solver=glpk diet1.py diet.dat
```

Chapter 2

Overview

Pyomo is available under the BSD open-source license. The following table describes standard installation options:

Coopr Sources	Installation Directories: System Python	User HOME	Virtual Environment
PyPI	User	User	Developer
ZIP File	User	User	Developer
Trunk			Developer

Installation options can be differentiated based on where the Pyomo source is downloaded and where Pyomo is installed. Pyomo can be installed from

1. PyPI - the standard Python distribution server
2. ZIP File - a ZIP file that contains Pyomo source files
3. Trunk - the subversion repository that contains the latest Pyomo revisions.

Pyomo can be installed in

a. System Python - Python's site packages directory

b. User HOME - the user's HOME directory

c. Virtual Environment - a Python virtual environment that is created in a user-specified directory

Only a subset of these options are recommended for general users, and only a few others are supported for developers.

The Prerequisites chapter discusses packages that you may need to install Pyomo. Pyomo requires one of the following Python versions: 2.6 2.7, 3.2, 3.3 or 3.4. Pyomo also requires the installation of several freely available Python libraries, but these are automatically installed with Pyomo.

Chapter Installing Pyomo for Users describes four installation options for Pyomo users. This chapter discusses installation with `pip` and `pyomo_install`. Chapter Installing Pyomo for Developers describes three additional installation options for Pyomo developers. These installation options can only be performed with `pyomo_install`.

Additional installation details are provided in subsequent chapters:

- Optional Packages: Other packages that Pyomo can optionally leverage.
- Installation Issues: Helpful information for resolving issues with Pyomo installation.
- Platform Notes: Documentation of platform-specific issues, especially for installation.
- Solver Notes: Documentation for third-party solvers that can be called from Pyomo.

Chapter 3

Prerequisites

3.1 Python Version

Pyomo requires Python versions 2.6, 2.7, 3.2, 3.3 or 3.4. In particular, some components rely on Python modules that exhibited known bugs in Python 2.5. We generally recommend installing a scientific Python distribution; see SciPy's list of scientific Python distributions.

- Windows users:

  The Python web site provides binary downloads for MS Windows installers. Recall that the PATH environment variable may need to be updated, as the installers typically do not update it automatically. On Windows, this can be easily accomplished via the Control Panel "Advanced System Settings" panel.

- Linux/UNIX users:

  Most flavors of Unix ship with Python, and a compatible version of Python can often be easily installed using your distribution's package manager. Note that you will need to install both the `python` package and (if separate) the `python-devel` package (to get the Python header files that are needed to compile some external applications distributed with Pyomo). Most Linux installers put the Python executable on a common location on your path. If this is not the case for your particular distribution, you can add the location to your shell's PATH environment variable.

- Mac users:

  The Python web site provides binary downloads for Mac installers. Recall that the PATH environment variable may need to be updated, as these installers typically do not update it automatically.

Please be aware of 32-bit versus 64-bit issues, particularly if you are running Windows. In particular, if running a 64-bit Windows OS (likely at this point, but not so with XP or earlier versions), you should use the Python 2.X.Y Windows X86-64 installer, as opposed to the (32-bit) Python 2.X.Y Windows installer. If you run 32-bit Python on a 64-bit Windows platform, for example, you will not have access to more than 3GB of RAM (actually a bit less) - even when your machine has more.

3.2 Installers

Installing Pyomo requires that one of the following installation tools or executables be downloaded: the `pip` Python package or the `pyomo_install` script.

3.2.1 Pip

The `pip` package can be used to download and install the latest Pyomo release. For Python versions before 3.4, the `pip` package will need to be explicitly installed. On Linux, you may be able to install `pip` with a package manager like `yum` or `apt-get`. Alternatively, you can securely download the get-pip.py script using your web browser (or a command-line tool like `curl`). Then, you execute the following in a shell:

```
python get-pip.py
```

The `pip` command requires internet access to download Pyomo and the packages it depends on. You need to specify the `HTTP_PROXY` environment to install from PyPI through a proxy server.

3.2.2 The `pyomo_install` Script

The `pyomo_install` script is used to download and install the latest Pyomo release. The `pyomo_install` has a `--help` option that provides detailed help instructions.

You can download the pyomo_install script using your web browser, or you can download from

```
https://software.sandia.gov/svn/public/pyomo/pyomo/trunk/scripts/pyomo_install
```

using a command-line tool like `curl`.

Depending on the command-line options, the `pyomo_install` command may require internet access to download Pyomo and the packages it depends on. Specifically, you may need to specify the `HTTP_PROXY` environment to install from PyPI through a proxy server. Also, developers installing the Pyomo trunk may need to configure subversion to specify a proxy server.

Note
The `pyomo_install` command has not been developed or tested on MS Windows. Although some of the installation strategies supported by `pyomo_install` may work on MS Windows, this installer is not recommended for that platform.

Note
The `pyomo_install` script does not bundle Python packages. Instead, it either installs packages from PyPI or it installs from a ZIP file.

3.3 Subversion (For Developers)

Installation of the Pyomo trunk requires the subversion software. On Unix systems, this is commonly pre-installed. On Windows, subversion installers can be obtained from CollabNet.

Only the command-line client is necessary; specifically, the "CollabNet Subversion Command Line" distribution should be used. Do *'not*' install "CollabNet Subversion Edge", as it downloads and installs Python. While the CollabNet download is free, CollabNet will require you to create a login - presumably to track usage statistics. The installer updates the system PATH variable automatically.

Chapter 4

Installing Pyomo for Users

The following table summarizes the different installation options that are recommended for users:

		Installation Directories		
		System Python	User HOME	Virtual Environment
Coopr Sources	PyPI	1a	1b	
	ZIP File	2a	2b	
	Trunk			

Users can install Pyomo from PyPI or a ZIP file, and Pyomo can be installed either in the system Python directories or in the user's HOME directory. Each of these options are described in the following sections; there may be multiple ways to install a specific installation option, and these are documented together.

Note
Installing from a ZIP file is supported to enable users to install a snapshot of Pyomo. Daily snapshots are available from the Pyomo download site; download the file `pyomo_votd.zip`.

Note
The `pyomo_install` script does not support the `--with-extras` option when installing from a ZIP file.

4.1 Option 1a: Installation from PyPI into the System Python Directory

This installation option downloads and installs the latest Pyomo release from PyPI into the system Python installation. Note that this installation option requires administrator access.

The following command uses `pip` to download and install Pyomo:

```
pip install Pyomo
```

On Linux and Mac systems, you may need to run this command with `sudo` to provide administrator access:

```
sudo pip install Pyomo
```

4.2 Option 1b: Installation from PyPI into the User's HOME Directory

This installation option downloads and installs the latest Pyomo release from PyPI into Python's alternate user installation. The alternate user installation is included in Python's packages, so this installation option does not require additional customization of Python.

Note
This installation strategy installs Pyomo in the user's HOME directory. The specific directory name depends on the type of machine; see the Python documentation for further details.

The following command uses `pip` to download and install Pyomo:

```
pip install --user pyomo
```

4.3 Option 2a: Installation from a ZIP File into the System Python Directory

This installation option installs Pyomo from a ZIP file into the system Python installation. Note that this installation option requires administrator access. This installation is done offline; no external dependencies are installed and no network access is required.

The following command uses `pyomo_install` to extract files from the ZIP file and install Pyomo:

```
pyomo_install --zip=pyomo_votd.zip
```

4.4 Option 2b: Installation from a ZIP File into the User's HOME Directory

This installation option installs Pyomo from a ZIP file into Python's alternate user installation. This installation is done offline; no external dependencies are installed and no network access is required. The alternate user installation is included in Python's packages, so this installation option does not require additional customization of Python.

Note
This installation strategy installs Pyomo in the user's HOME directory. The specific directory name depends on the type of machine; see the Python documentation for further details.

The following command uses `pyomo_install` to extract files from the ZIP file and install Pyomo:

```
pyomo_install --zip=pyomo_votd.zip --user
```

Chapter 5

Installing Pyomo for Developers

The following table summarizes all of the installation options that are supported for Pyomo:

		Installation Directories		
		System Python	User HOME	Virtual Environment
Coopr Sources	PyPI	1a	1b	1c
	ZIP File	2a	2b	2c
	Trunk			3c

Three additional options are supported for developers, all of which install Pyomo in a virtual Python environment. A virtual Python environment is an isolated Python installation that depends on the system Python installation, but which mimics a complete Python installation. A virtual Python environment can be created by a user in their directory without requiring administrative privileges.

5.1 Option 1c: Installation from PyPI into a Virtual Environment

This installation option downloads and installs the latest Pyomo release from PyPI into a virtual Python environment.

The following command uses `pyomo_install` to download and install Pyomo:

```
pyomo_install --venv=pyomo
```

The virtual Python environment is created in the `pyomo` directory, and the command `pyomo/bin/python` executes a virtual Python environment into which Pyomo has been installed.

5.2 Option 2c: Installation from a ZIP File into a Virtual Environment

This installation option installs Pyomo from a ZIP file into a virtual Python environment. This installation is done offline; no external dependencies are installed and no network access is required.

The following command uses `pyomo_install` to extract files from the ZIP file and install Pyomo:

```
pyomo_install --zip=pyomo_votd.zip --venv=pyomo
```

The virtual Python environment is created in the `pyomo` directory, and the command `pyomo/bin/python` executes a virtual Python environment into which Pyomo has been installed.

5.3 Option 3c: Installation from Trunk into a Virtual Environment

This installation option uses subversion to checkout Pyomo packages, and Pyomo is installed in a virtual Python environment.

The following command uses `pyomo_install` to checkout and install Pyomo:

```
pyomo_install --trunk --venv=pyomo
```

The virtual Python environment is created in the `pyomo` directory, and the command `pyomo/bin/python` executes a virtual Python environment into which Pyomo has been installed.

Note

For Option 3c, the `--with-extras` option installs `pyomo.extras` in an `editable` mode. If you wish to install `pyomo.extras` after installing Pyomo, then the following `pip` command must be executed:

```
pip install -e svn+https://software.sandia.gov/svn/public/pyomo/pyomo.extras/trunk
```

Chapter 6

Optional Packages

6.1 pyomo.extras

The `pyomo.extras` packages installs a variety of packages that extend Pyomo's functionality, including:

- `Pyro` - This library supports distributed parallel communication in Python.
- `Suds` - This library provides a lightweight SOAP client that is used by `pyomo.neos` to remotely launch optimizers on the NEOS server.
- `OpenOpt` - This library provides interfaces for a variety of optimizers.

6.2 PyYaml

The default file format used by Pyomo is JSON. YAML is the default format if the PyYaml package is installed. On most platforms, this package can be easily installed with `pip`:

```
pip install PyYaml
```

6.3 Pywin32

On Windows, it may be necessary to install the Python Windows extensions. This package enables Python to support Windows COM interfaces, and specifically it allows Pyomo to work with Excel spreadsheets. Note that "pywin32" is a bit of a misnomer, in that it works fine on 64-bit platforms; based on what we can tell, the "32" just means "non-16-bit".

The installer for this extension is available from http://sourceforge.net/projects/pywin32. This install simply updates the Python site packages.

6.4 PyODBC

Pyomo uses the PyODBC package to access ODBC data sources. On Windows, the PyODBC package can be installed by downloading a Windows installer, or by executing `easy_install`:

```
easy_install pyodbc
```

On Linux and Mac OS X, you need to install several system-level packages for proper ODBC functionality. These include:

- `unixODBC`: provides generic ODBC connectivity
- `unixODBC-devel`: includes ODBC development headers for linking with other libraries
- `mdbtools`: implements Microsoft Jet (Access) database interaction
- `libmdbodbc`: bridges Jet databases via ODBC

Finally, users on such Unix-based systems may have to configure their local ODBC installations for Jet databases if using Microsoft Access. To do so, add the following to the file `/etc/odbcinst.ini`:

```
[Microsoft Access Driver (*.mdb)]
Description = MDB Tools ODBC drivers
Driver      = /usr/lib/libmdbodbc.so.0
Setup       =
FileUsage   = 2
```

Pyomo will attempt to add any further ODBC information as needed.

Chapter 7

Installation Issues

Here are some of the most common issues encountered by Pyomo users. If you do not see your question answered here, the best route for getting an answer is by engaging the Pyomo community through the pyomo-forum mailing list.

7.1 Python Installation

The standard Python distribution is built with the C language, and thus this is sometimes called CPython. CPython distributions are available for all major platforms, and this is the most common version of Python that is used in practice. Linux and Macintosh distributions commonly include CPython, and no additional configuration is required to execute the Python interpreter.

On Windows platforms, CPython can be easily installed with an installer executable. However, the user must edit the PATH environment variable to include the Python installation path.

Other noteworthy implementations of Python are Jython, written in Java, IronPython, written for the Common Language Runtime, and PyPy, a Python interpreter writen in Python. A major difference between CPython, Jython and IronPython is that they support extensions with C, Java and .NET application framework respectively. PyPy supports these extensions depending on the underlying Python interpreter, and it uses a JIT compiler to improve the execution of Python code.

Pyomo is developed and tested using CPython. Other Python implementations have been evaluated, and their status is summarized here:

- Jython: The `virtualenv` package is not supported in Jython. This package is critical for the development, testing and deployment of Pyomo.
- IronPython: This Python implementation does not currently support installation with `setuptools` or `distribute`, which are commonly used installation mechanisms in CPython. Pyomo depends on these packages.
- PyPy: Many Pyomo package install with PyPy. There remain some issues with reference counting in Python, which are documented in Pyomo ticket #4437.

7.2 Pyomo Installation

7.2.1 Help with `pyomo_install`

There are numerous options to the `pyomo_install` script, which can be displayed by typing:

```
./pyomo_install --help
```

Important
Make sure you do *'not'* have the `PYTHONHOME` environment variable defined prior to installing with `pyomo_install`. Such a definition interferes with the construction of the Pyomo virtualized Python environment.

7.2.2 Using the `HTTP_PROXY` Environment Variable

In many computing environments, it may be necessary to set the `HTTP_PROXY` environment variable to use the `wget` and `pyomo_install` commands. A typical value for this variable is something like "http://foo.institution.domain:80". Your local system administrator can help you assess whether you need an HTTP proxy for web access. For example, at Sandia (New Mexico only) the proxy `http://sonproxy.sandia.gov:80` is used for the SON network, and `http://wwwproxy.sandia.gov:80` is used for the SRN network.

7.2.3 Installation error: Filename ... does not start with any of these prefixes: ...

We have seen the error:

```
Installation error: Filename ... does not start with any of these prefixes: ...]
```

when installing on Windows. This appears to be a limitation of the virtual environment logic. Specifically, this was triggered by explicitly specifying the Python path with a lower-case drive name. For example:

```
c:\Python27\python.exe pyomo_install
```

Using a capital drive name resolved this issue:

```
C:\Python27\python.exe pyomo_install
```

7.2.4 Installation error: Setup script exited with error: command *gcc* failed ...

Several Python packages that Pyomo relies on (e.g., coverage and PyYAML) include compiled "C" extensions. This error indicated that the extensions did not successfully compile. In most cases, the root cause is that the Python C headers are not installed or available on your system. For Linux/UNIX users, make sure you have the `python-devel` package installed.

7.3 Using Subversion with a Proxy

Subversion does not use the `HTTP_PROXY` environment variable to configure its behavior. Rather, this must be done by modifying the local subversion configuration.

7.4 Python environment variables

The `PYTHONPATH` and `PYTHONHOME` environment variables are used to customize the way that Python is configured. The `PYTHONHOME` variable defines the location of the standard Python libraries that are used, and `PYTHONPATH` augments the default search path for module files.

When Pyomo is installed in a virtual Python environment, these environment variables are generally not necessary. We have seen many cases where Pyomo scripts failed to operate properly when these variables were defines, so we generally recommend that the user disable these variables before installing Pyomo in a virtual environment.

Chapter 8

Platform Notes

We have successfully installed and executed Pyomo on a wide range of platforms. In the course of doing so (and helping other users do so), we have identified a number of issues that are worth mentioning, and easily correctable.

8.1 Windows XP, Vista, and 7

An issue we have observed on Windows Vista is the following error:

```
WindowsError: [Error 740] The requested operation requires elevation
```

This issue is due to the new and "improved" Windows Vista permissions scheme. Basically, `python.exe` needs to run with administrator privileges, e.g., to compile things in site packages. To accomplish this, simply navigate via Explorer to where `python.exe` is located (typically `C:\Python2.6`) and right-click on the `python.exe` icon. Then, select the "Compatability" tab and check the "Run this program as an administrator" box.

In addition, some distributions of CPLEX and AMPL appear to have overloaded the name for the `CPLEX.exe` executable. Pyomo currently interacts with CPLEX via the command-line, interactive version, and not the version with the AMPL/NL driver. The symptom of this situation is a failed solve, which can be easily diagnosed by turning on the "--keepfiles" option in pyomo and looking at the CPLEX output log. The fix is to simply make sure the interactive CPLEX is first in your PATH environment variable.

8.2 Windows 7

To install on Windows 7 (and possibly other Windows systems), it is necessary to run with administrator privileges when installing Pyomo. Otherwise, the `pyomo_install` script will generate error messages indicating the lack of write permission in various directories. If installing from the command prompt, one can right-click on the "Command Prompt" icon and select "Run as administrator" to obtain such permissions. It is worth noting that this is not always necessary, as in many environments users are automatically granted administrator privileges.

Various users have observed issues when running `pyomo_install` under Windows 7, related to the subversion client obtained from http://tortoisesvn.tigris.org. In particular, you may observe the following error:

```
Can't move '.svn/tmp/entries' to '.svn/entries': The file or directory is corrupted and ↩
    unreadable.
```

To quote the Tigris site:

```
This error message typically occurs when you try to update or commit
your working copy, and seems to be common on Windows 7 systems. It
is due to another process holding a handle on a file that Subversion
```

```
needs to move or modify. This might be a virus scanner, but on
Windows 7 it is likely to be the Windows Indexing Service. Turn off
the indexing service on your working copies and repositories, and
exclude them from virus scans.
```

Some combination of disabling the indexer and the antivirus software seems to correct the problem; please let us know if this is not the case.

8.3 Red Hat Linux

No issues involving installation on Red Hat systems have been observed.

8.4 Ubuntu Linux

Some distributions of Ubuntu do not install by default with subversion. Users and/or their system administrators will need to install this package.

Similarly, some Ubuntu distributions do not appear to have a full Python 2.6 install. Specifically, the `cProfile` package can be missing, and will need to be installed.

8.5 MacOS (Snow Leopard)

At least one user has reported problems during execution of the `pyomo_install` script on MacOS, specifically related to the installation of the "psutils" package. When attempting to compile this package, one may observe numerous compile errors relating to various system header files. This issue appears to be due to ARCHFLAGS on some systems defaulting to the PowerPC architecture. To fix this issue, first execute:

```
sudo ARCHFLAGS="-arch i386 -arch x86_64" easy_install psutil
```

Next, re-execute `pyomo_install`. With `psutil` successfully installed, `pyomo_install` will simply use the installed package and skip the re-install.

Chapter 9

Solver Notes

Pyomo includes interfaces to a variety of third-party solvers. We do not attempt to document the installation of these packages. This chapter provides notes for how the user environment needs to be configured so Pyomo can automatically detect and execute these solvers.

9.1 ASL Solvers

The AMPL Solver Library (ASL) provides a general interface for reading an AMPL NL file. Pyomo provides a general interface for any solver that employs that ASL. That is, Pyomo executes ASL solvers with the same command-line syntax that is used by AMPL, and the SOL result files are read to represent the solver results in Pyomo. Pyomo can execute NL files generated by AMPL or Pyomo's Pyomo modeling package.

See the AMPL web pages for a summary of the ASL solvers that are available. Many of these have commercial support, and there are a variety of mature open source ASL solvers.

9.2 CBC

CBC is an open-source solver for linear programs and mixed-integer linear programs. CBC provides a stand-alone binary executable.

See the CBC wiki for download and installation instructions. The directory containing the `cbc` executable must be in the list of paths defined by the `PATH` environment variable.

9.3 CPLEX

The IBM ILOG CPLEX Optimizer includes commercial mathematical programming solvers for linear programming, mixed integer programming, quadratic programming, and quadratically constrained programming problems.

Pyomo has solver interfaces for the CPLEX command as well as the CPLEX Python environment. To use the CPLEX command, the directory containing the `cplex` executable must be in the list of paths defined by the PATH environment variable. There are a variety of ways that the CPLEX Python environment can be used with Pyomo. Perhaps the simplest is to install the CPLEX Python package.

For example, if the CPLEX Python directory is `/usr/local/ilog/cplex/python`, then you can install Pyomo with this package installed as follows:

```
pyomo_install --venv=pyomo
cd /usr/local/ilog/cplex/python
pyomo_python setup.py develop
```

The last step installs CPLEX's Python package within Pyomo's virtual environment in developer mode.

9.4 GLPK

The GLPK (GNU Linear Programming Kit) package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library.

See the GLPK website for download and installation instructions. The directory containing the `glpsol` executable must be in the list of paths defined by the PATH environment variable.

9.5 GUROBI

The Gurobi Optimizer is a commercial solver for linear programming (LP), quadratic programming (QP) and mixed-integer programming (MILP and MIQP).

To use Gurobi with Pyomo, the directory containing the `gurobi.sh` (Linux) or `gurobi.bat` (MS Windows) executable must be in the list of paths defined by the PATH environment variable.

9.6 PICO

PICO is a solver for linear programming and mixed-integer linear programming problems that can perform parallel optimization on distributed memory machines.

PICO is a component of Acro, an open-source software project that integrates a variety of optimization software packages, including both libraries developed at Sandia National Laboratories as well as publicly available third-party libraries. Acro's Getting Started wiki pages provides instructions for downloading and installing Acro projects that include PICO. The directory containing the `PICO` executable must be in the list of paths defined by the PATH environment variable; typically this will be the `acro/bin` directory.

Colophon

This book was created using `asciidoc` software.

The Sandia National Laboratories Laboratory-Directed Research and Development Program and the U.S. Department of Energy Office of Science Advanced Scientific Computing Research Program funded portions of this work.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND 2012-1572P.