Getting DAE Tools

From DAE Tools

Contents

Introduction

DAE Tools (pyDAE module) is installed in daetools folder within site-packages (or dist-packages) folder under python (tipically /usr/lib/pythonXY/Lib or C:\PythonX.Y\Lib). The structure of the folders is the following:

- daetools
  - daePlotter
  - daeSimulator
  - docs
  - examples
  - solvers

Requirements

Mandatory packages:

- Python (v2.6^{+}): [1] (http://www.python.org)
- Boost libraries (v1.41^{+}; GNU/Linux only): [2] (http://www.boost.org)
- Numpy (v1.3^{+}): [3] (http://numpy.scipy.org)
- Scipy (v0.8^{+}): [4] (http://www.scipy.org)
- Matplotlib (v0.99^{+}): [5] (http://matplotlib.sourceforge.net)
- pyQt4 (v4.3^{+}): [6] (http://www.riverbankcomputing.co.uk/software/pyqt)
- Lapack (GNU/Linux only): [7] (http://www.netlib.org/lapack)
- Blas (GNU/Linux only): [8] (http://www.netlib.org/blas)

Optional packages (3rd party linear solvers):

- Umfpack (v5.4^{+}; GNU/Linux only)
- Intel MKL (v10.2.5.035) - proprietary
- AMD ACML (v4.4.0) - proprietary

For more information on how to install packages please refer to the documentation for the specific library. By default GNU/Linux and Windows versions come with the default Sundials dense LU linear solver, Trilinos Amesos with built-in support for KLU and SuperLU linear solvers, Trilinos AztecOO with built-in support for Ifpack and ML preconditioners, and IPOPT/BONMIN with support for MUMPS linear solver (with PORD ordering). Standalone SuperLU/SuperLU_MT come with GNU/Linux versions only. Additional linear solvers: Umfpack for Trilinos Amesos (GNU/Linux only), AMD ACML and Intel MKL must be downloaded separately since they are not free software.

Getting the packages

First download the appropriate installer for your operating system (GNU/Linux, Windows) and architecture (i386, amd64, arm):

- .deb for Debian GNU/Linux and derivatives
- .rpm for Red Hat GNU/Linux and derivatives
- .exe for Windows

Installation

Debian GNU/Linux (and derivatives)

First install the mandatory packages. To do so you can use the Synaptic Package Manager or type the following commands:

```
sudo apt-get install libboost-python-dev libboost-system-dev libboost-thread-dev
sudo apt-get install python-qt4 python-numpy python-scipy python-matplotlib mayavi2
sudo apt-get install liblapack3gf libblas3gf libamd2.2.0 libumfpack5.4.0
```

Install **DAE Tools** by using GDebi package installer or by typing the following shell command:

```
sudo dpkg -i daetools_x.x.x_platform_os.deb
```

If there are some unmet dependencies run the following command to install them:

```
sudo apt-get -f install
```

To uninstall use:

```
sudo apt-get remove daetools
```

Red Hat GNU/Linux (and derivatives)

First install the mandatory packages. To list dependencies use the command:

```
sudo rpm -qpR daetools_x.x.x_platform_os.rpm
```

To install dependencies use the Package Manager or type the following commands:

```
sudo yum install boost-python boost-thread boost-system
sudo yum install PyQt4 numpy scipy python-matplotlib Mayavi
sudo yum install blas lapack suitesparse
```

Then install **DAE Tools** by using Package Installer or by typing the following shell commands:

```
sudo rpm -i daetools_x.x.x_platform_os.rpm
```

To upgrade a previous version of daetoools to the new one use:

```
sudo rpm -U daetools_x.x.x_platform_os.rpm
```

To uninstall use:

```
sudo rpm -e daetools
```

Windows

DAE Tools is compiled and tested on a 32-bit Windows XP, however it should work on later versions of Windows as well. In order to use **DAE Tools** on 64-bit versions of Windows the python, pyqt, numpy and scipy 32-bit versions should be installed. First install the necessary dependencies (except the boost library, which is pre-built and comes with **DAE Tools**): **python**, **numpy**, **scipy**, **matplotlib** and **pyqt4**. As a starting point the following links can be used:

- Python 2.7: download link (http://www.python.org/ftp/python/2.7.1/python-2.7.1.msi)
- Numpy: download link (http://sourceforge.net/projects/numpy/files/NumPy/1.6.0/numpy-1.6.0-win32-superpack-python2.7.exe/download)
- Scipy: download link (http://sourceforge.net/projects/scipy/files/scipy/0.9.0/scipy-0.9.0-win32-superpack-python2.7.exe/download)
- Matplotlib: download link (http://sourceforge.net/projects/matplotlib/files/matplotlib/matplotlib-1.0.1/matplotlib-1.0.1.win32-py2.7.exe/download)
- PyQt4: download section (http://www.riverbankcomputing.co.uk/static/Downloads/PyQt4)

To be able to create 3D plots you need to install Mayavi2 package. Alternatively you can install everything needed through Python(x,y) (http://www.pythonxy.com) . Finally, install **DAE Tools** by double clicking the file daetools_x.x-x_win32_winxp_python27.exe and follow the instructions. To uninstall use the uninstall program in Start -> All Programs -> DAE Tools -> Uninstall

Compiling DAE Tools from source

Compiling of DAE Tools core libraries and python modules

To compile the **DAE Tools** the following is needed:

- Installed boost (v1.41^{+}) headers and boost-system, boost-thread, and boost-python libraries
- Installed python 2.6/2.7, numpy, and scipy modules
- Compiled DAE/LA/NLP solvers: Sundials IDAS, Bonmin, NLopt, Trilinos, SuperLU, SuperLU_MT, Blas/Lapack

All DAE Tools modules are developed using the QtCreator/QMake cross-platform integrated development environment. The source code can be checked out from the DAE Tools subversion repository (https://daetools.svn.sourceforge.net/svnroot/daetools) :

```
svn co https://daetools.svn.sourceforge.net/svnroot/daetools daetools
```

GNU/Linux

The easy way

First, install all the necessary dependencies by executing **install_dependencies_linux.sh** shell script located in the **trunk** directory. It will check the OS you are running (currently Debian, Ubuntu, Linux Mint, CentOS and Fedora are supported but other can be easily added) and install all necessary packages

needed for **DAE Tools** development.

```
# 'lsb_release' command might be missing and has to be installed before proceeding.
# On Debian based systems:
# sudo apt-get install lsb-release
# On red Hat based systems:
# sudo yum install redhat-lsb

cd daetools/trunk
sh install_dependencies_linux.sh
```

Then, compile the third party libraries by executing **compile_libraries_linux.sh** shell script located in the **trunk** directory. The script will download all necessary source archives from the DAE Tools SourceForge web-site, unpack them, apply changes and compile them. If all dependencies are installed there should not be problems compiling the libraries.

Note: there are known problems to compile the older bonmin and trilinos libraries using GNU GCC 4.6. This has been fixed in bonmin 1.5^{+} and trilinos 10.8^{+} versions. Therefore, either GCC 4.5 and below or the recent versions of bonmin/trilinos libraries should be used.

```
sh compile_libraries_linux.sh
```

Finally, compile the **DAE Tools** libraries and python modules by executing **compile_linux.sh** shell script located in the **trunk** directory. The script accepts one argument specifying projects that should be compiled. Any of the following is accepted: all, dae, superlu, superlu_mt, superlu_cuda, cusp, trilinos, bonmin, ipopt, and nlopt. If **all** is specified the script will compile dae, superlu, superlu_mt, trilinos, bonmin, ipopt, and nlopt projects.

```
sh compile_linux.sh all
# Or for instance:
# sh compile_linux.sh dae superlu nlopt
```

As a result there should be the following static/dynamic libraries in the **trunk/release** folder:

- libcdaeCore.a, libcdaeActivity.a, libcdaeDataReporting.a, libcdaeIDAS_DAESolver.a, libcdaeSuperLU_LASolver.a, libcdaeSuperLU_MT_LASolver.a, libcdaeSuperLU_CUDA_LASolver.a, libcdaeIPOPT_NLPSolver.a, libcdaeBONMIN_MINLPSolver.a and libcdaeNLOPT_NLPSolver.a
- pyCore.so, pyActivity.so, pyDataReporting.so, pyIDAS.so, pySuperLU.so, pySuperLU_MT.so, pySuperLU_CUDA.so, pyTrilinos, pyBONMIN.so, pyIPOPT.so, pyNLOPT.so

Now go to trunk/Installations folder and build an installation package:

```
cd daetools/trunk/Installations
sh create_linux_package.sh 1 1 2
```

It will produce something like daetools_1.1.2_amd64_debian-squeeze.deb or daetools_1.1.2_amd64_fedora-lovelock.rpm (depending on the version and OS).

From QtCreator IDE

DAE Tools can also be compiled from within QtCreator IDE. First install dependencies and compile third party libraries (as explained in The easy way) and then do the following:

- Do not do the shadow build. Uncheck it (for all projects) and build everything in the release folder
- Choose the right specification file for your platform (usually it is done automatically by the IDE, but double-check it):
  - for 32 bit machines use **-spec linux-g++-32**
  - for 64 bit machines use **-spec linux-g++-64**
- Compile the **dae** project (you can add the additional Make argument -jN to speed-up the compilation process, where N is the number of processors plus one; for instance on the quad-core machine you can use -j5)
- Compile **SuperLU/SuperLU_MT/SuperLU_CUDA** and **Bonmin/Ipopt** solvers. **SuperLU/SuperLU_MT/SuperLU_CUDA** and **Bonmin/Ipopt** share the same code and the same project file so some hacking is needed. Here are the instructions how to compile them:
  - Compiling **libcdaeBONMIN_MINLPSolver.a** and **pyBONMIN.so**:
    Set CONFIG += BONMIN in BONMIN_MINLPSolver.pro, run qmake and then compile
    Set CONFIG += BONMIN in pyBONMIN.pro, run qmake and then compile
  - Compiling **libcdaeIPOPT_NLPSolver.a** and **pyIPOPT.so**:
    Set CONFIG += IPOPT in BONMIN_MINLPSolver.pro, run qmake and then compile
    Set CONFIG += IPOPT in pyBONMIN.pro, run qmake and then compile
  - Compiling **libcdaeSuperLU_LASolver.a** and **pySuperLU.so**:
    Set CONFIG += SuperLU in LA_SuperLU.pro, run qmake and then compile
    Set CONFIG += SuperLU in pySuperLU.pro, run qmake and then compile
  - Compiling **libcdaeSuperLU_MT_LASolver.a** and **pySuperLU_MT.so**:
    Set CONFIG += SuperLU_MT in LA_SuperLU.pro, run qmake and then compile
    Set CONFIG += SuperLU_MT in pySuperLU.pro, run qmake and then compile
  - Compiling **libcdaeSuperLU_CUDA_LASolver.a** and **pySuperLU_CUDA.so**:
    Set CONFIG += SuperLU_CUDA in LA_SuperLU.pro, run qmake and then compile
    Set CONFIG += SuperLU_CUDA in pySuperLU.pro, run qmake and then compile
- Compile the **LA_Trilinos_Amesos** project

Windows

Necessary tools: **QtCreator** (http://qt.nokia.com/products/developer-tools) , **Microsoft VC++** (http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=14597) and **G95 Fortran** (http://www.g95.org) compiler (Mumps only).
Note: Compiling all third party libraries and **DAE Tools** projects requires a mental gymnastics impossible to describe by any human language so that the pre-compiled libraries are provided in the downloads section (**windows libraries** (https://sourceforge.net/projects/daetools/files/windows%20libraries)).

DAE Tools should be compiled from within QtCreator IDE:

- Unpack the downloaded archive *bonmin-trilinos-idas-superlu-nlopt-mumps-g95-msvc-win32.zip* into the *daetools/trunk* folder.
  All libraries are compiled with MS VC++ 2008 Express edition (the most likely other versions of MS VC++ will also work). Mumps Fortran 95 files are compiled with G95 Fortran compiler.

- Path to **libf95.a** and **libgcc.a** libraries should be set in **dae.pri** config file.
  For instance, if G95 is installed in **c:\g95** set the **G95_LIBDIR** variable to:

```
G95_LIBDIR = c:\g95\lib\gcc-lib\i686-pc-mingw32\4.1.2
```

- Follow the instructions for compiling **DAE Tools** described in From QtCreator IDE section above.
  Note: superlu_mt and superlu_cuda cannot be compiled on Windows at the moment.

Now go to trunk/Installations folder and build an installation package (you should have Nullsoft Scriptable Install System, NSIS installed):

```
cd daetools/trunk/Installations
create_windows_package.cmd 1 1 2 win32 winxp 27
```

It will produce daetools_1.1.2_win32_winxp_python27.exe installation package.