PyDQED User's Guide

Release 1.0.1

Joshua W. Allen

January 28, 2016

CONTENTS

CHAPTER

ONE

INTRODUCTION

In the course of scientific computing, it is frequently desired to find the solution of a set of nonlinear algebraic equations – or at least the optimum value – subject to a number of constraints. Such a problem is often posed in the following general way: minimize the sum of squares of the nonlinear algebraic equations

$$f_i(\mathbf{x}) = 0 \qquad i = 1, 2, \ldots, N_{\mathrm{eq}}$$

for a vector $\mathbf{x}$ of N_{vars} variables, subject to the constraints

$$g_j(\mathbf{x}) = 0 \qquad j = 1, 2, \ldots, N_{\mathrm{cons}}$$

and optional lower and upper bounds

$$l_k \leq x_k \leq u_k \qquad k = 1, 2, \ldots, N_{\mathrm{vars}}$$

A variety of interesting, practical problems can be expressed in the above form.

The DQED solver provides an algorithm for solving the above problem for the case where the constraints $g_j(\mathbf{x})$ are linear (or very nearly so). For more on DQED, please see *[Dongarra1979] [Schnabel1984] [Hanson1986]* or this page.

1.1 Why PyDQED?

DQED is written in Fortran 90, which is not much fun to code in, especially for novice programmers. (In particular, getting data into and out of a Fortran program via file input and output is quite difficult and awkward.) However, the strength of Fortran is that it produces code that is very efficient to execute, which is often important when solving large systems of nonlinear equations.

Meanwhile, the Python programming language is much easier to program in. In particular, Python comes with a large library of free, open source packages that provide a wide range of functionality, limiting the amount of work the programmer needs to do from scratch. A number of packages, including NumPy, SciPy, and matplotlib, replace much of the functionality of numerical computing environments such as MATLAB. However, the optimization code within SciPy is still in need of some refinement.

PyDQED provides Python programmers with access to DQED via a Python wrapper.

1.2 Should I Use PyDQED?

Depending on exactly the type of problem you are solving, PyDQED may be a useful tool. By no means is PyDQED the best tool for every problem, but it provides you with another option to choose from.

Read on to learn how to install and use PyDQED, and tips for getting the most out of PyDQED.

CHAPTER

TWO

INSTALLATION

2.1 Dependencies

PyDQED is currently available for the Python 2.x series. In particular, Python 2.5 and later are known to work. In addition to the packages in the Python standard library, PyDQED depends on the following packages:

- NumPy version 1.3.0 or later
- Cython version 0.12.1 or later

In addition, you will also need a Fortran compiler and a C compiler that produce object files that can interoperate [1]. The `gfortran` and `gcc` compilers from the GNU Compiler Collection are known to work, and are probably your best bet no matter what operating system you are using. On Windows the MinGW compiler collection provides these compilers.

The DQED code is provided with the PyDQED package; you do not need to download it separately.

2.2 Installing PyDQED through the Anaconda Binary (Recommended)

There is no need to install dependencies with this method. You will automatically install the latest binary package for PyDQED.

- Download and install the Anaconda Python Platform for Python 2.7 (make sure not to install Python 3.0+, which is incompatible with RMG). When prompted to append Anaconda to your PATH, select or type Yes.
- It is safest to install the PyDQED binary onto a fresh Anaconda envrionment via the Terminal or Command Prompt

```
conda create -c rmg --name pydqed_env pydqed
```

- (For Unix-based systems) To use it you must first activate the environment by typing

```
source activate pydqed_env
```

- (For Windows) To use it you must first activate the environment by typing

```
activate pydqed_env
```

[1] The Fortran interfaces are exposed to Python via C, so the installer needs to be able to link object files from Fortran and C for this to work.

2.3 Compiling the Source Code

2.3.1 Unix-like Systems

A Makefile has been provided that can be used to compile DQED and the PyDQED wrapper code. To use, invoke the following command from the root package directory:

```
$ make
```

This command will build PyDQED in-place, rather than installing it to your Python package directory. At this point you can optionally run the unit test script `test.py` and/or any of the provided examples to confirm that PyDQED was compiled successfully.

If you wish to formally install PyDQED, run the following command from the root package directory after the `make` command (you may need root privileges for this):

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

You may wish to write a file *make.inc* that sets certain variables used by the Makefiles (e.g. the Fortran compiler). An example of such a file, *make.inc.example*, has been provided.

2.3.2 Windows

Warning: A regression in Cython 0.14 has resulted in it being unusable for PyDQED in Windows. Cython 0.13 is known to work, and more recent released of Cython should also correct for this regression. See this thread from the `cython-dev` mailing list for more information.

A batch script `make.bat` has been provided in the root package directory. Double-clicking this script will cause DQED to be compiled into a static library and the PyDQED wrapper code to be compiled.

Note: The batch script presumes that you have the 32-bit version of the MinGW C and Fortran compilers installed. You may need to add the location of the MinGW `bin` directory to the `PATH` environment variable if this has not already been done.

At this point you can optionally run the unit test script `test.py` and/or any of the provided examples to confirm that PyDQED was compiled successfully.

Warning: When using MinGW with Cython on Windows, you may encounter the error "Unable to find vcvarsall.bat". A workaround for this issue is available from the Cython FAQ at this page. In particular the `pydistutils.cfg` file approach should work.

If you wish to formally install PyDQED, run the following command from the root package directory after the batch script completes successfully (you may need administrator privileges for this):

```
> python setup.py install
```

BIBLIOGRAPHY

[Dongarra1979] J. Dongarra, J. Bunch, C. Moler, and P. Stewart. “LINPACK User’s Guide.” *SIAM (Society for Industrial and Applied Mathematics)* Philadelphia, 1979.

[Schnabel1984] R. Schnabel and P. Frank. “Tensor Methods for Nonlinear Equations.” *SIAM J. Numer. Analysis* **21**, p. 815-843 (1984).

[Hanson1986] R. Hanson. “Least Squares with Bounds and Linear Constraints.” *SIAM J. Sci. Stat. Comp.* **7**, p. 826-834 (1986).