SmartGridToolbox

build passing

SmartGridToolbox is a C++11 library for electricity grids and associated elements, concentrating on smart/future grids and grid optimisation. It is designed to provide an extensible and flexible starting point for developing a wide variety of smart grid simulations.

Getting Started

Detailed installation instructions can be found below. Additional information and documentation is available at the SmartGridToolbox homepage, including doxygen documentation and tutorials. Examples may be found in the *examples* subdirectory.

License

SmartGridToolbox is licensed under the Apache 2.0 License. Please see the LICENSE file for details. Also see the NOTICE file (which must be redistributed with derivative works) for details about how to cite useage.

Building and Installing SmartGridToolbox

Obtaining the Code

We recommend cloning out the latest version of the code from GitHub. You will need to have first installed git.

```
git clone --recursive https://github.com/NICTA/SmartGridToolbox.git
```

You can also download the latest release from GitHub.

Compilers

You need a compiler that will properly support C++14. SmartGridToolbox is tested on Mac OS X with clang++ from Apple LLVM version 6.1.0 (similar to linux v. 3.6), and on Ubuntu (linux) with clang++ version 3.6 and g++ version 5.0. For macs, we recommend using clang++, and for linux we recommend g++ version 5 or greater.

If building on a Mac, you should have a recent version of the OSX operating system (Mavericks(?), Yosemite, or El Capitan). Install the latest version of XCode, and you will automatically have access to the correct clang compilers.

If building on Ubuntu 14 (or 15?) you can use, e.g.

```
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt-get update
sudo apt-get install g++5.0
```

Ubuntu 16 should just work out of the box.

Libraries

Install up to date versions of the following libraries:

- **Boost libraries**

  boost/date_time, *boost/random* (header only), *boost/spirit* (header only), *boost/test*.

The easiest procedure is probaly to install the whole of boost, either by downloading the latest version from www.boost.org and following the build instructions, or, preferably, by using a tool such as homebrew (Mac OSX), apt-get (Debian Linux distributions such as Ubuntu) or similar. Make sure your boost version is not too old (we test v.1.55, and of course later versions should also work).

- **The yaml-cpp library**

  Source code is provided in *third_party/yaml-cpp-0.5.0*, or alternatively, install using homebrew, apt-get or the like.

```
mkdir build
cd build
cmake ..
make && sudo make install
```

- **Armadillo linear algebra library**

  This is available from http://arma.sourceforge.net, and is also included in the *third_party* directory of SmartGridToolbox for convenience. You should use version 5.2 or later.

  On a mac, you can use homebrew:

```
brew install armadillo
```

  To build and install from the *third_party* directory (assuming you are in the top level *SmartGridToolbox* directory), you can do something like the following:

```
cd third_party/armadillo
mkdir build
cd build
cmake ..
make && sudo make install
```

- **(Optional) SuperLU (for armadillo)**

  If you intend to use Armadillo as a sparse solver (the default option), you will need to make sure it has been built with SuperLU support enabled. This will require the SuperLU libraries to be installed prior to compilation. Please see the armadillo documentation for more information. If using homebrew on a mac, all of this is taken care of automatically when you install Armadillo.

- **(Optional) SuiteSparse libraries**

  amd, *colamd*, *btf*, *klu*

  These need only be installed if the default Armadillo sparse solver is not used, which means that SmartGridToolbox must be configured with `--with-klu` specified. Since these libraries are moderately faster than SuperLU, you should choose this option unless you already have SuperLU or have used, e.g., homebrew to install SuperLU in which case the dependency is automatically taken care of.

  The source code for these is provided in the *third_party/SuiteSparse* directory. The build is straightforward, but depending on your system, you may need to modify the file *SuiteSparse/SuiteSparse_config/SuiteSparse_config.mk*. For example, on a mac, you should replace that file with a copy of *SuiteSparse/SuiteSparse_config/SuiteSparse_config_Mac.mk*. Then, just build the required libraries:

```
cd SuiteSparse_config; make && sudo make install; cd ..
cd AMD; make && sudo make install; cd ..
cd BTF; make && sudo make install; cd ..
cd COLAMD; make && sudo make install; cd ..
cd KLU; make && sudo make install; cd ..
```

- **(Optional) PowerTools**

  PowerTools provides optional optimising solvers (OPF, relaxations, etc.). An installation is included in *third_party/PowerTools*, with build instructions available in that directory. PowerTools requires you to install IPOPT, which is fairly involved owing to having to install several other dependencies.

Build and install the SmartGridToolbox libraries:

If installing from a tarball: simply run the usual configure/make/install procedure, using the desired options to configure (type `configure --help` for a list).

```
./configure
make
sudo make install
```

If you have the PowerTools library installed and wish to use its functionality, you should use the `--enable-power-tools` option to configure. If you happen to have the SuperLU library installed, and wish to use it instead of KLU as a sparse solver, use the `--without-klu` option.

If running from a source distribution cloned from git, you should run autogen.sh immediately after cloning the github repository. You will need an up to date version of gnu autotools.

```
./autogen.sh # Normally only need to do this once.
./configure # --with-klu # if you don't have SuiteSparse support in armadillo.
make
sudo make install
```

Run tests

From the top level directory,

```
make check
```

To compile and link your custom SmartGridToolbox program

If you wish to build using GNU autotools, have a look at examples/PvDemo/Makefile.am. You could copy examples/PvDemo to use as a template for your project.

Otherwise, the following compile command would apply if using clang++, for example:

```
clang++ {options} -o myprogram myprogram.cc -lSgtSim -lSgtCore -larmadillo -lboost_date_time -lyaml-cpp
```

If you are using SuiteSparse for a sparse solver, you need to add extra link
flags:

```
clang++ {options} -o myprogram myprogram.cc -lSgtSim -lSgtCore -larmadillo -lboost_date_time -lyaml-cpp -lklu -lamd -
lcolamd -lbtf
```