Jupyter Notebook and Matar

May 21, 2024

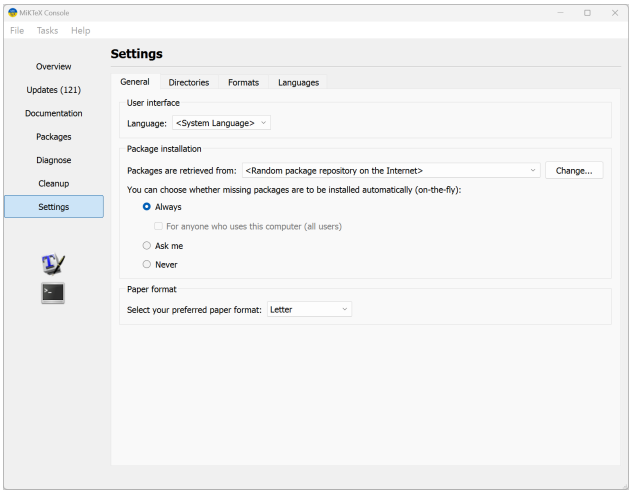
1 Supplemental Resources

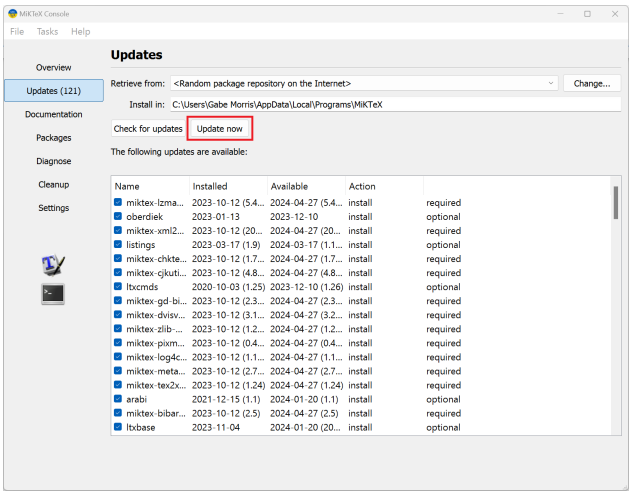
For getting a good understanding of jupyter, check out this video on YouTube. That video will walk you through the basics of the interface and show some neat tools built-in with jupyter. Additionally, a large part of using jupyter notebook is knowing `sympy` (if using python). In this repository, you will find a useful introduction to sympy file that contains examples.

2 Prerequisite Software

The following are required for using jupyter with c++ and for rendering PDFs from the jupyter notebook. Refer to their corresponding documentation for installation.

- anaconda/miniconda
- MiKTeX - After downloading, launch the application and go to settings. Mark the option to install packages on the fly as always, then set the paper format to letter. Next, update all packages. Make sure that the MiKTeX binaries are added the `PATH` environment variable.





- Pandoc

3 Installing with MATAR

To use Matar and Kokkos with jupyter, build each application with cmake and ensure that the install prefix is set to the conda environment. First, create a new conda environment.

```
conda create -n xeus-cling
```

Activate the new environment.

```
conda activate xeus-cling
```

Install xeus-cling and jupyer.

```
conda install -c conda-forge xeus-cling jupyter
```

Recursively clone the MATAR repository and go into it.

```
git clone --recursive https://github.com/lanl/MATAR.git
cd MATAR
```

Make a build directory and set it as the `$build_dir` environment variable.

```
mkdir build
cd build
export build_dir=`pwd`
```

Go into Kokkos and build with whichever architecture is desired. This is example uses pthreads.

```
cd ../src/Kokkos/kokkos
cmake -B $build_dir -S . -D CMAKE_BUILD_TYPE=Release \
-D CMAKE_INSTALL_PREFIX="$CONDA_PREFIX" \
-D CMAKE_CXX_STANDARD=17 \
-D Kokkos_ENABLE_SERIAL=ON \
-D Kokkos_ARCH_NATIVE=ON \
-D Kokkos_ENABLE_TESTS=OFF \
-D BUILD_TESTING=OFF \
```

```
-D Kokkos_ENABLE_THREADS=ON \
-D BUILD_SHARED_LIBS=ON
```

Go into the build directory and run `make` and `make install`.

```
cd $build_dir
make
make install
```

Clear out the build directory, then build MATAR.

```
rm -rf *
cd ..
cmake -B $build_dir -S . -D CMAKE_INSTALL_PREFIX="$CONDA_PREFIX" \
-D CMAKE_PREFIX_PATH="$CONDA_PREFIX" \
-D Matar_ENABLE_KOKKOS=ON
```

Go into the build directory, then install it to the conda environment.

```
cd build
make install
```

4 Practical MATAR Example

[1]:
```
// Includes
// Be sure to replace the path here with your conda lib path
#pragma cling add_library_path("/home/jovin/miniconda3/envs/xeus-cling/lib")
#pragma cling load("libkokkoscore")

#define HAVE_KOKKOS 1
#define HAVE_THREADS 1

#include "matar.h"

using namespace mtr;
```

In the jupyter notebook scope, it's like everything is in the `main` function, so treat the following code as if this were the case.

[2]:
```
// Initialize Kokkos
// Do not run this cell multiple times
int argc = 1;
char* arg1 = strdup("--kokkos-num-threads=10");  // Allocate and copy the string
char* argv[] = {arg1};
Kokkos::initialize(argc, argv);
```

MATAR examples should be executed between the calls of `Kokkos::initialize()` and `Kokkos::finalize()`.

[3]:
```
FOR_ALL(i, 0, 10, {
    std::cout << "Running Thread " << i << std::endl;
    std::this_thread::sleep_for(std::chrono::seconds(1));
});
```

```
Running Thread Running Thread Running Thread Running Thread 30Running Thread
Running Thread 6
9

Running Thread 7
Running Thread 8
4
1
Running Thread 2
Running Thread 5
```

Had the example above run in serial, then it would have been about 10 seconds long, but with 10 threads running, this only takes about 1 second.

[4]:
```
free(arg1);
Kokkos::finalize();
```

5 Converting to PDF

Once finished with the notebook, use `nbconvert` to convert to PDF.

```
jupyter nbconvert --to pdf notebook_file.ipynb
```