PyMAGMA: A Python Library for MAGMA

Student: Delario Nance, Jr. (Davidson College)
Mentors: Stanimire Tomov (UTK), Kwai Wong (UTK)
Research Assistant: Julian Halloy (UTK)

Background

What makes C++ and Python different?

Despite C++ and Python both being very popular programming languages, each tool possesses unique advantages and disadvantages. Notably, while computers can run C++ code very quickly, C++ syntax can be difficult for new programmers to understand. Conversely, Python sacrifices speed for a simple, easy-to-read syntax.

What is a Python Interface?

We define a *Python interface* to be "any tool allowing code written in non-Python languages to be used with Python." A prime example of such is NumPy, a popular Python library containing many functions for performing linear algebra computations. Despite being used with Python, NumPy code is mainly written in C. By using Python interfaces, programmers can combine the fast speeds of languages like C and C++ with the simple syntax of Python.

What is MAGMA?

Like NumPy, Matrix Algebra for GPU and Multicore Architectures (MAGMA) is a package of linear algebra operations. Unlike NumPy, however, MAGMA is largely written in C++. Furthermore, while NumPy code is suited for running on CPUs, MAGMA code is specialized for running on GPUs. These two differences result in MAGMA obtaining faster computational speeds than NumPy.

What is SWIG?

Simplified Wrapper and Interface Generator (SWIG) is a tool for building interfaces through which C/C++ code can be used in high-level languages (e.g., Python). SWIG accomplishes this task by creating "wrapper" code which translates C/C++ code to the target language.

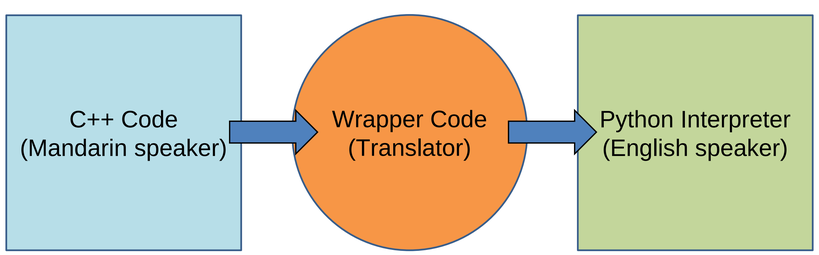

Figure 1. A real-life analogy of using SWIG with Python

SWIG Workflow

File 1. Header File (*pymagma.h*)
A list of all the C++ functions from MAGMA we want to use with Python.

```
magma_int_t
magma_dgesv_gpu(
    magma_int_t n, magma_int_t nrhs,
    magmaDouble_ptr dA, magma_int_t ldda,
    magma_int_t *ipiv,
    magmaDouble_ptr dB, magma_int_t lddb,
    magma_int_t *info);
```

File 2. Interface File (*pymagma.i*)
The location where we specify the name of the Python library to create (PyMAGMA).

```
// Step 1
%module pymagma

// Step 2
%{
    // Step 3
    #include "pymagma.h"
%}

// Step 4
%include "pymagma.h"
```

File 3a. Import File (*pymagma.py*)
The file used to import the PyMAGMA library once it is created. This file also contains the Python functions which users will call to use the C++ functions inside MAGMA.

```
def magma_dgesv_gpu(n, nrhs, dA, ldda, ipiv, dB, lddb, info):
    return _pymagma.magma_dgesv_gpu(n, nrhs, dA, ldda, ipiv, dB, lddb, info)
```

File 3b. Wrapper Code (*pymagma_wrap.cxx*)
The file containing the "wrapper" code which will translate the C++ code in MAGMA to the Python interpreter, letting C++ code be used with Python after PyMAGMA is created and imported.

```
SWIGINTERN PyObject *_wrap_magma_print_environment(PyObject *SWIGUNUSEDPARM(self), PyObject *args) {
  PyObject *resultobj = 0;

  if (!SWIG_Python_UnpackTuple(args, "magma_print_environment", 0, 0, 0)) SWIG_fail;
  magma_print_environment();
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}
```

File 4. Shared Library (*_pymagma.so*)
PyMAGMA, the importable Python library containing the C++ functions we want to use from MAGMA and the compiled "wrapper" code from the Wrapper file.

```
# Create pymagma.so from MAGMA and wrapper object code
!ld -shared /home/user1/magma/lib/libmagma.so pymagma_wrap.o -o _pymagma.so
```

Current Results

Solving AX = B with PyMAGMA

We can use PyMAGMA to perform DGESV with simple 5 x 5 matrices. DGESV is the task of solving matrix equations of the form AX = B with double precision. An example of solving DGESV on the GPU with PyMAGMA is shown below:

```
Input Matrix A on CPU before DGESV:
[
   3.0000   6.0000   2.0000   5.0000   1.0000
   5.0000   5.0000   2.0000   4.0000   4.0000
   6.0000   6.0000   8.0000   2.0000   2.0000
   4.0000   7.0000   9.0000   3.0000   8.0000
   5.0000   3.0000   5.0000   7.0000   2.0000
];
```

```
Input Matrix B on CPU before DGESV:
[
   3.0000   6.0000   2.0000   5.0000   1.0000
   5.0000   5.0000   2.0000   4.0000   4.0000
   6.0000   6.0000   8.0000   2.0000   2.0000
   4.0000   7.0000   9.0000   3.0000   8.0000
   5.0000   3.0000   5.0000   7.0000   2.0000
];
```

```
Output Solution X on CPU after DGESV:
[
   1.0000   0.0000   0.       0.       0.0000
  -0.0000   1.0000   0.       0.      -0.0000
  -0.0000  -0.0000   1.0000   0.      -0.0000
   0.0000   0.0000  -0.0000   1.0000  -0.0000
   0.0000   0.0000  -0.0000  -0.0000   1.0000
];
```

Research Goal

1) Use SWIG to build an interface through which MAGMA functions can be used with Python, by creating and importing a library of C++ functions from MAGMA

Acknowledgements

This research project was sponsored by the National Science Foundation (NSF) through a Research Experience for Undergraduates (REU) grant for the Research Experiences in Computational Science, Engineering, and Mathematics (RECSEM) program held at the University of Tennessee, Knoxville (UTK). During the project, research assistance was received by researchers from the Innovative Computing Laboratory (ICL) and University of Tennessee, Knoxville. Also, the SWIG 4.0 Documentation was essential in learning how to use the tool.

Future Work

1) Record the performance of matrix-matrix multiplication (SGEMM) with PyMAGMA
2) Make PyMAGMA compatible with foreign data types (e.g., NumPy arrays)