



Kamodo C++

Rebecca Ringuette
with the Kamodo Team @ CCMC

Steps for KamodoC++ on Windows

- Build a conda environment for pybind11 and Kamodo 
- Embed python into C++ using pybind11 
- Embed Kamodo into C++ (In process)
- Run Kamodo in C++ from notebook interactively (Coming soon!)
- **Goal:** Enable a virtual satellite flythrough of model data from C++

Installation instructions (Windows 10)

- For Windows, use Anaconda prompt. For WSL v1, use linux terminal
- Create a conda environment
 - `conda create -n KamodoCXX_Win python=3.7.9`
 - `conda install -n KamodoCXX_Win -c conda-forge pybind11 cmake plotly sympy scipy pytest pandas hydra-core requests ipython`
 - `conda activate KamodoCXX_Win`
 - `pip install python-forge`
 - `conda deactivate` (when done)
- Install Kamodo in desired directory from <https://github.com/asherp/Kamodo>
(Some specialized adjustments currently needed.)

Folders and Files

Directory Structure

- *KamodoCXX_Win*
 - CMakeLists.txt
 - K1.py
 - main.cpp
 - *build*
 - *KamodoMaster*
- *build* directory will contain the executable once built
- *KamodoMaster* directory contains Kamodo unzipped

CMakeLists.txt

```
1  cmake_minimum_required(VERSION 3.4)
2  project(main)
3
4  set(Python_FIND_VIRTUALENV ONLY)
5  find_package(Python COMPONENTS Interpreter Development)
6  message(STATUS "int path: ${Python_EXECUTABLE}")
7  find_package(pybind11 REQUIRED)
8
9  add_executable(main main.cpp)
10 target_link_libraries(main pybind11::embed)
```

main.cpp

```
1  #include <pybind11/embed.h> // everything needed for embedding
2  namespace py = pybind11;
3  using namespace pybind11::literals;
4
5  #include <iostream>
6  #include <string>
7  #include <list>
8  using namespace std;
9
10
11 int main() {
12     //Embedded portion
13     py::initialize_interpreter(); // start the interpreter
14
15     //Add kamodo dir path to sys.path to enable importing
16     py::list path_list = py::module_::import("sys").attr("path");
17     path_list.append("./KamodoMaster/kamodo");
18
19     //import and make a kamodo object
20     py::module_np = py::module_::import("numpy");
21     py::object Kamodo_Object = py::module_::import("kamodo").attr("Kamodo")();
22     py::function CXXkamodofy = py::module_::import("K1").attr("CXXkamodofy");
23
24     //add a function to and print the Kamodo object
25     Kamodo_Object = CXXkamodofy(Kamodo_Object, "B",
26         py::list(py::make_tuple(0.01,0.02,0.03,0.04,0.05)), "T");
27     py::print(Kamodo_Object);
28     py::print(Kamodo_Object.attr("B").attr("data"));
29     py::print(Kamodo_Object.attr("B").attr("meta"));
30
31     py::finalize_interpreter();
32 }
```

Build Commands

(from the directory of main.cpp)

Windows 10

- conda activate KamodoCXX_Win
- cmake -B build -A x64
- cmake --build build
- **set PYTHONHOME = (path to python.exe in env)**
- build\Debug\main.exe

Windows 10 using WSL v1

- conda activate KamodoCXX
- cmake -B build
- cmake --build build
- chmod +x build/main
- ./build/main

Windows 10 Note: The path of python.exe is printed by the message command in the CMakeLists.txt file after the find_package python command.

Before using 'conda deactivate' on Windows 10, do 'set PYTHONHOME = C:\ProgramData\Anaconda3' or wherever Anaconda stores its main python.exe or the command will error.

Sample Output

(Shown on Windows without WSL)

```
Command Prompt - conda activate KamodoCXX_Win - conda deactivate
Microsoft Windows [Version 10.0.17763.1757]
(c) 2018 Microsoft Corporation. All rights reserved.

>cd KamodoCXX_Win

\KamodoCXX_Win>conda activate KamodoCXX_Win

(KamodoCXX_Win) \KamodoCXX_Win>cmake -B build -A x64
-- Building for: Visual Studio 15 2017
-- The C compiler identification is MSVC 19.16.27045.0
-- The CXX compiler identification is MSVC 19.16.27045.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/Program Files (x86)/Microsoft Visual Studio/2017/Community/VC/Tools/MSVC/14.16.27023/bin/Hostx86/x64/cl.exe - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual Studio/2017/Community/VC/Tools/MSVC/14.16.27023/bin/Hostx86/x64/cl.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Python: [redacted]/.conda/envs/KamodoCXX_Win/python.exe (found version "3.7.9") found components: Interpreter Development Development Module Development Embedded
-- int path: [redacted]/.conda/envs/KamodoCXX_Win/python.exe
-- Performing Test HAS_MSVC_GL_LTCG
-- Performing Test HAS_MSVC_GL_LTCG - Success
-- Found pybind11: [redacted]/.conda/envs/KamodoCXX_Win/Library/include (found version "2.6.2" )
-- Configuring done
-- Generating done
-- Build files have been written to: [redacted]/KamodoCXX_Win/build
```

```
Command Prompt - conda activate KamodoCXX_Win - conda deactivate

(KamodoCXX_Win) \KamodoCXX_Win>cmake --build build
Microsoft (R) Build Engine version 15.9.21+g9802d43bc3 for .NET Framework
Copyright (C) Microsoft Corporation. All rights reserved.

Checking Build System
Building Custom Rule [redacted]/KamodoCXX_Win/CMakeLists.txt
main.cpp
main.vcxproj -> [redacted]\KamodoCXX_Win\build\Debug\main.exe
Building Custom Rule [redacted]/KamodoCXX_Win/CMakeLists.txt

(KamodoCXX_Win) \KamodoCXX_Win>set PYTHONHOME=[redacted]/.conda/envs/KamodoCXX_Win/

(KamodoCXX_Win) \KamodoCXX_Win>build\Debug\main.exe
Python is now embedded
Printing the new Kamodo object:
{B(x): <function CXXkamodofy.<locals>.<lambda> at 0x0000021DB8DA4D38>, B: <function CXXkamodofy.<locals>.<lambda> at 0x0000021DB8DA4D38>}
[0.01, 0.02, 0.03, 0.04, 0.05]
{'units': 'T', 'arg_units': None, 'citation': None, 'equation': None, 'hidden_args': []}
The python interpreter is now closing

(KamodoCXX_Win) \KamodoCXX_Win>set PYTHONHOME=C:/ProgramData/Anaconda3/

(KamodoCXX_Win) \KamodoCXX_Win>conda deactivate

\KamodoCXX_Win>
```

Finds the python installed in the conda env
Outputs the path for PYTHONHOME for later

Creates and prints a kamodo object from C++