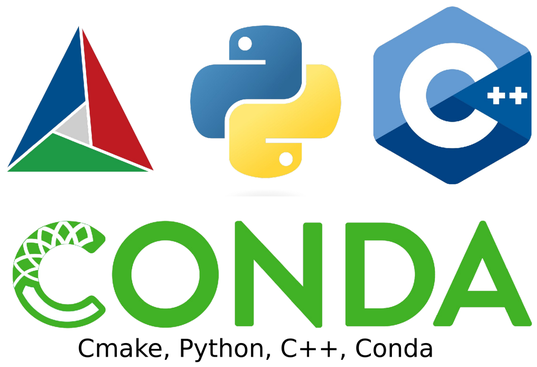
Building and Distribution of Bertini2 Using CMake and Conda

Jack Hagen, Dr. silviana amethyst | Department of Mathematics, University of Wisconsin, Eau Claire

Introduction

Bertini2 is a mathematical package which allows researchers, industry professionals, and others to solve arbitrary polynomial systems. The goal of this project is to make Bertini available in the Conda package manager in order to allow for easy installation and use. In order to do this, we had to switch Bertini's build system from Autotools to the modern CMake. The Bertini core and Python bindings are compiled separately, so two CMake configurations were required to build the complete package. After completing this step, we will return to writing the necessary files to install Bertini2 via Conda.



Cmake, Python, C++, Conda

```
● vscode ➜ /workspace/core/build (develop) $ cmake ..
  -- Configuring done
  -- Generating done
  -- Build files have been written to: /workspace/core/build
● vscode ➜ /workspace/core/build (develop) $ make
  Consolidate compiler generated dependencies of target bertini2
  [ 50%] Built target bertini2
  Consolidate compiler generated dependencies of target bertini2_exe
  [ 57%] Built target bertini2_exe
  Consolidate compiler generated dependencies of target b2_class_test
  [ 90%] Built target b2_class_test
  Consolidate compiler generated dependencies of target blackbox_test
  [100%] Built target blackbox_test
○ vscode ➜ /workspace/core/build (develop) $
```

Pivoting

Our end goal was to make Bertini installable with the Conda package manager. Through our research we discovered that Conda is difficult to make work with packages compiled with Autotools, so we had to switch the project to the more modern build system CMake.

Challenges

Switching a complex software project such as Bertini from one build system to another runs into a number of challenges.

- A lack of documentation for specific compiling options.
- A split code structure requiring two different processes for the C++ core and the Python bindings.
- Cross-platform compatibility.
- Lack of support for easily installing and using required libraries.

We were able to overcome our challenges by finding answers to similar questions on online forums and by trying many different options.

python > examples > solve_system.py

```python
import pybertini as pb
import numpy as np

x = pb.Variable('x')
y = pb.Variable('y')

vg = pb.VariableGroup([x,y])

sys = pb.System()

sys.add_function(x-y)
sys.add_function(x**2 + y**2 - 1)

sys.add_variable_group(vg)

C = pb.multiprec.Complex

variable_values = np.array([C(0), C(0)])

result = sys.eval(variable_values)

sys.homogenize()
sys.auto_patch()


solver = pb.nag_algorithm.ZeroDimCauchyAdaptivePrecisionTotalDegree(sys)

solver.solve()

for soln in solver.solutions():

    print(sys.dehomogenize_point(soln))
```

```
● (base) vscode ➜ /workspace/python (develop) $ pip install .
  Processing /workspace/python
    Preparing metadata (setup.py) ... done
  Building wheels for collected packages: pybertini
    Building wheel for pybertini (setup.py) ... done
    Created wheel for pybertini: filename=pybertini-1.0a5-py3-none-any.whl size=
  31149112 sha256=315cbc9f0d070fc30302bbb7a5ed147de362a58ec5d91253007d87d4e31915
  dc
    Stored in directory: /tmp/pip-ephem-wheel-cache-aeo8z5fk/wheels/3d/23/b0/6aa
  2e6700286497d1d65222efb6505e8b4d33282d4b2346fd0
  Successfully built pybertini
  Installing collected packages: pybertini
    Attempting uninstall: pybertini
      Found existing installation: pybertini 1.0a5
      Uninstalling pybertini-1.0a5:
        Successfully uninstalled pybertini-1.0a5
  Successfully installed pybertini-1.0a5
● (base) vscode ➜ /workspace/python (develop) $ python3 examples/solve_system.py

  [(7.0710678118698802181630e-01, 3.8458995250690156907510e-13)
   (7.0710678118714003021690e-01, -4.1373915754520431959860e-14)]
  [(-7.0710678118652321239840e-01, 5.1946363645507889605320e-13)
   (-7.0710678118709653866310e-01, 2.1988646851405583755990e-13)]
```

Results

Bertini2 and its Python bindings are now installable with CMake and pip, and can be switched over to Conda. Future work will likely include porting the project to the official Anaconda repository or the community Conda-Forge repository.

We thank the Office of Research and Sponsored Programs for supporting this research, and Learning & Technology Services for printing this poster.