SUSTAINABLE SOFTWARE PRACTICES IN DEVELOPING MATLAB

PAT QUILLEN

MathWorks, Inc.

Introduction

Here we discuss some of the practices we use in developing core MATLAB®. Also, we talk about some sustainability practices that research software can use in order to be considered for inclusion into MATLAB.

Some sustainability practices at MathWorks

One of our primary motivators in developing core MATLAB is a commitment to backwards compatibility. We have a general goal of trying to keep MATLAB programs that have been developed against previous versions running in successive versions, with little in the way of behavior change. MathWorks was established in 1984, and MATLAB some time before that, and of course, in the intervening years, computing has changed appreciably. In order to keep MATLAB up-to-date, the underlying algorithms have changed, sometimes significantly so, in order to exploit multiple cores or vector instructions, et cetera. *In the large*, however, programs written in the earliest versions of MATLAB should more or less continue to return *qualitatively* the same results.

In order to realize these goals, we consider the following when developing MATLAB:

(1) *Backwards Compatibility*
We rarely truly throw out anything. In order to move forward, we may develop new functionality that is to be preferred over older functionality, and in this case, we discourage use of the older functionality and encourage the newer functionality.

(2) *Simplicity of API*
This is not only for usability or the benefit of the customer. Indeed, knowing that we're unlikely to throw things away strongly influences the development of new functionality. We carefully consider requirements and do our best to develop APIs exposing only what is *strictly necessary* to meet those requirements we've discovered. Secondly, we consider where we want to go in the future, and do our best to develop APIs that permit expansion while not cornering ourselves unnecessarily.

(3) *Reproducibility*
We do our best to make sure that MATLAB code is run-to-run reproducible. That is, we want to make sure that given the same inputs, MATLAB produces the same outputs. This requires a careful definition of run-to-run which we have made for ourselves. Frequently, this means limiting performance, however, we value reproducibility (or at least a simple path to reproducibility) above performance.

(4) *Defect tracking and publishing*
A key property of sustainable software is a commitment to reducing and removing defects. To effectively do this, defects must be tracked and lessons learned from each defect in order

E-mail address: pquillen@mathworks.com.
Date: 17 June 2019.

to prevent insertion of similar defects in the future. Moreover, reporting known defects to the user community highlights that the maintainers of the code are aware of them. A public repository of such bugs also helps users know about possible workarounds while highlighting which versions of the software suffer each defect.

(5) *Regression testing for correctness and performance*
We test our software rigorously in an effort to make sure previous defects do not return. Moreover, we use the same testing framework internally that we have released to help users in testing their MATLAB code.

What we look for in research software to be incorporated

In order to move MATLAB forward, we make use of some amount of third-party software. While our staff has expertise in a host of technical computing disciplines, we very often turn to third-party codes, and frequently those born as research codes, to provide functionality to our customers. The following are some of the things we look for in considering whether or not to adopt a research code into MATLAB.

(1) *The Basics*
Above all, the project should be sufficiently documented so that we know how to use it. Revision control, perhaps counter-intuitively, is optional—we'll control the revisions we use ourselves. Bug-tracking is preferred, as well as knowing exactly how to report defects, including to whom. Moreover, the software should be supported on all the platforms we care about.

(2) *Reproducibility*
Can the code be made to be reproducible, and under what circumstances are the results reproducible? What trade-offs must be made to get there?

(3) *Standards-based*
Does the code leverage standard practices or adhere to some kind of standard (e.g. IEEE 754 compliance, language specific standards such as FORTRAN 77, C99, or C++11)? Moreover, successful codes will adhere to some kind of internal standards, such as naming standards, or C++ standards such as those offered by Google, or the C++ Core Guidelines. The particular standard is not necessarily important, however, the existence of standard implies that a software project has sufficient maturity and has likely already achieved sustainability.

(4) *Fitting into our ecosystem*
We care about how third-party software will fit into our existing ecosystem. Some questions we'll ask of such software include:
- How is threading used?
- Are globals used and if so, how are they managed?
- What is the error handling policy?
- What is the memory allocation policy?

The implicit follow-up question to each of the above is "and how do we control it"? Essentially, all of these things must fit into our ecosystem, and once we decide to include a third-party code in MATLAB, a fair portion of our work comes in fitting a potentially square peg into a round hole. If codes establish and adhere to some standard as described above, typically, many of these questions are addressed by the standard in place.

(5) *Licensing*
Unfavorable license terms are the easiest way for us to kick out software. Generally, we favor terms that permit modification of the code (whether or not modifications must be provided back to the authors) mostly so that we can have the freedom to fix any bugs that arise.

Summary

In this brief whitepaper we have shared some of the things we consider in developing the 35 year old commercial product MATLAB as well as some of the properties of software projects we consider for incorporation into MATLAB. We believe that many of these properties are required for scientific software to be considered sustainable and that highlighting these properties is a necessary first step in codifying sustainability in scientific software.