C++ UI design for NAG Optimization Modelling Suite

Johannes Lotz: lotz@stce.rwth-aachen.de

1 Requirements

1.1 Description

NAG optimization modelling suite (NOMS) consists of a set of routines, which allows you to define and solve various optimization problems in a uniform manner. The first key feature of the suite is that the definition of the optimization problem and the call to the solver are separated so that it is possible to set up a problem in the same way for different solvers. The second feature is that the problem representation is built up from basic components (building blocks). Therefore, different types of problems reuse the same routines for their common parts. A detailed introduction NOMS can be found https://www.nag.com/numeric/nl/nagdoc_latest/flhtml/e04/e04intro.html#optsuite.

Your task is to develop a new object oriented C++ UI for NOMS. The interface should cover all

- routines to define the objective and
- routines to define the constraints,

as well as the following solvers

- e04mt,
- e04kf, and
- e04sr.

The interface should be designed such that the user doesn't have to write code that computes derivatives. Instead Automatic Differentiation should be used to obtain required derivative information (software: dco/c++). Additionally, in the case nonlinear constraints are given, dco/c++ should be used to test, whether some of these are actually linear constraints. If linear constraints are detected, the problem definition should be changed to reflect that additional knowledge.

The code should be tested and documented (user and development documentation).

1.2 Dependencies

Following is a list of tools/libraries you should use for this project.

- Buildsystem: CMake
- Unit-Tests: CTest and googletest
- Documentation: doxygen
- dco/c++ (derivative computation)
- NAG Library

You can use arbitrary libraries in addition, please let me know which ones in advance.

2 Further remarks

You can get inspiration for you interface design from the following libraries

- FICO Xpress python interfaces,
- Gurobi C++ interface, or

any other C++ library you can find.

The team must use gitlab (RWTH) to communicate and to maintain the source code. The code must be written in C++ and free of memory leaks and invalid memory accesses. Modern C++ feature (C++20) are encouraged. Run tests with memory sanitizer (gcc or clang). In addition, please **present the draft** of your interface, not later **than TBD?**. For testing, implement at least two examples for each solver.