SIMSOPT Phase 1: MANGO
Vision & Progress

“Multiprocessor Algorithms for Nonlinear Gradient-free Optimization”

https://github.com/landreman/mango

Structure of STELLOPT, ROSE, SIMSOPT, etc

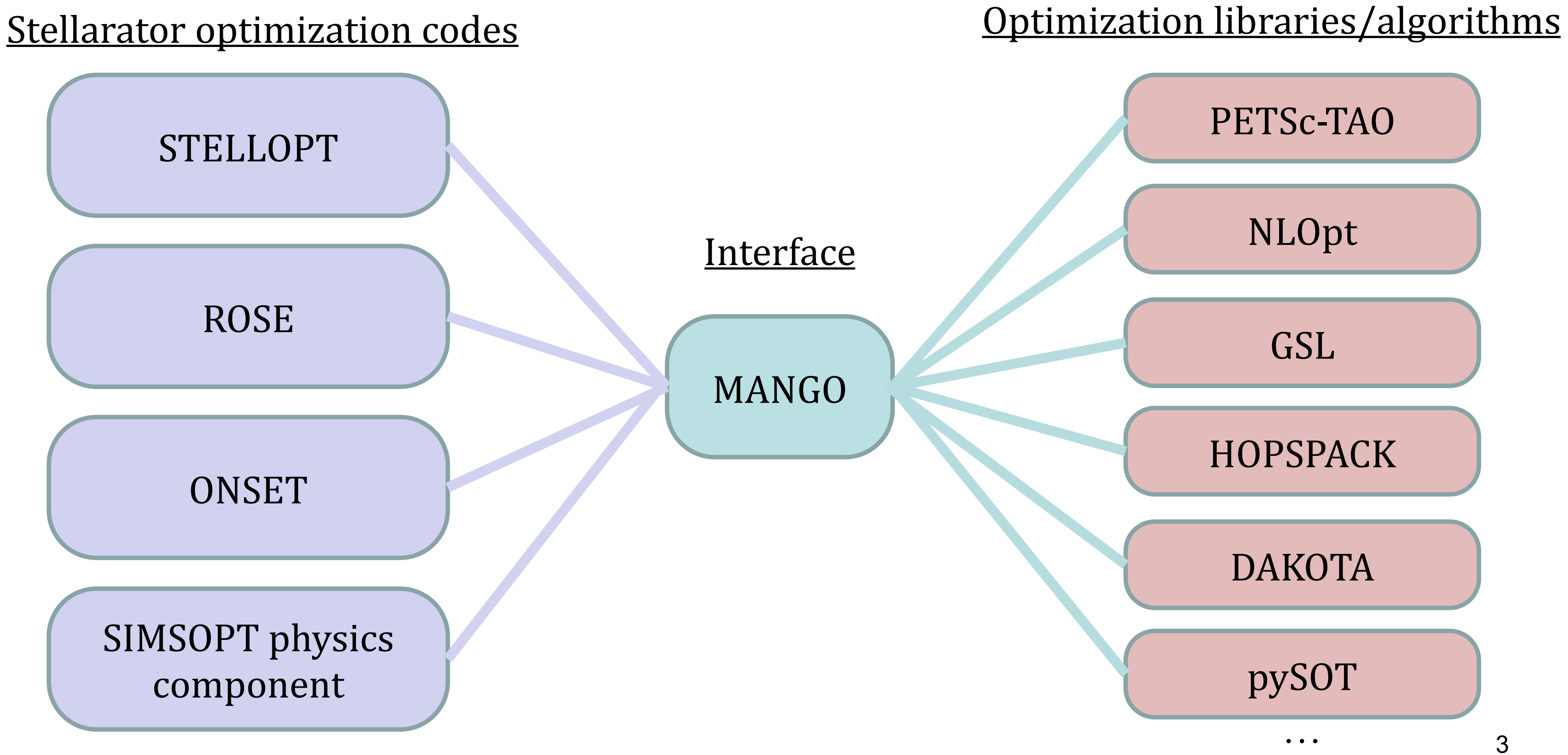
Driver

Physics/Engineering:
objective function & constraints

Optimization algorithms

MANGO addresses this component

MANGO allows us to provide new capabilities to our existing codes like STELLOPT before the physics part of SIMSOPT is functional



Goals of MANGO

- Make it easy to try out different optimization algorithms.
- Provide a common interface to established optimization libraries like PETSc-TAO, HOPSPACK, DAKOTA, NLOpt, providing many new optimization algorithms to STELLOPT, ROSE, etc.
- MANGO is a library that STELLOPT/ROSE/COILOPT++/ONSET/etc can call.
- Allow any gradient-based algorithm (e.g. BFGS) to use parallelized finite-difference derivatives.
- Also allow parallelization within each evaluation of the objective function.
- Automatically convert least-squares objective functions into generic objective functions if an algorithm is selected that does not assume least-squares form.
- Collection of test problems that are automatically run for various numbers of processors.
- Could eventually add our own implementations of algorithms that are not already available in libraries, e.g. IMFIL (presently only in matlab).
- No physics will be included in MANGO itself.
- Callable from Fortran, C, C++, & python.
- Set up testing/documentation/build in a simpler setting than the physics part of simsopt.

Why make MANGO a separate repository from STELLOPT/SIMSOPT?

Pros:

- Any algorithms in it can be immediately available to all our optimizer codes: STELLOPT, ROSE, ONSET, COILOPT++, FOCUS, etc.
- We could learn whether any differences between STELLOPT vs ROSE vs SIMSOPT are due to differences in the objective function vs the optimization algorithm.
- Enforce separation between optimization algorithms vs objective function.
- Optimization folks could add & test algorithms in a simpler repository than STELLOPT.
- Could be used by people in other fields.

Cons:

- It adds a few steps to building STELLOPT.

Design choices for MANGO so far

- Main language = C++
  - Most of the libraries we want to connect are in C++ (ROSE, HOPSPACK, DAKOTA) or C (GSL, PETSc, NLOpt).
  - Many tools available, e.g. testing frameworks.
  - Object-oriented is favored for extensible architecture.
- Try to minimize dependencies. Other than MPI, all dependencies (PETSc, NLOpt, HOPSPACK, Catch2) are optional.
- MPI only. OpenMP is better used within the objective function.
- Assume (at least for now) no analytic gradients are available.

MANGO has comprehensive testing

Unit tests:

- Uses Catch2.
    - Header-only, so no library to build. Smaller dependency than Boost, Google test.
- Python script runs unit tests for various #s of MPI processes.

Integrated tests:

- 6 benchmark problems so far: 3D quadratic, 2D Rosenbrock, etc.
- Python script runs each problem for all algorithms & various #s of MPI processes.
- Checks performed for deterministic algorithms:
    - Results are independent of # of MPI processes & # of "worker groups".
    - Results from C++ driver == results from Fortran driver.
- Regression tests for all algorithms: results match reference values (within tolerance)

Continuous integration:

- All unit tests & integrated tests are run on Travis-CI for every commit.

landreman / mango

build passing

Current Branches Build History Pull Requests More options

✓ catch2 Trying to get run_mpi_unit_tests to work on Travis

#73 passed

Commit 34ef139

Compare b47f456..34ef139

Branch catch2

Ran for 9 min 26 sec

33 minutes ago

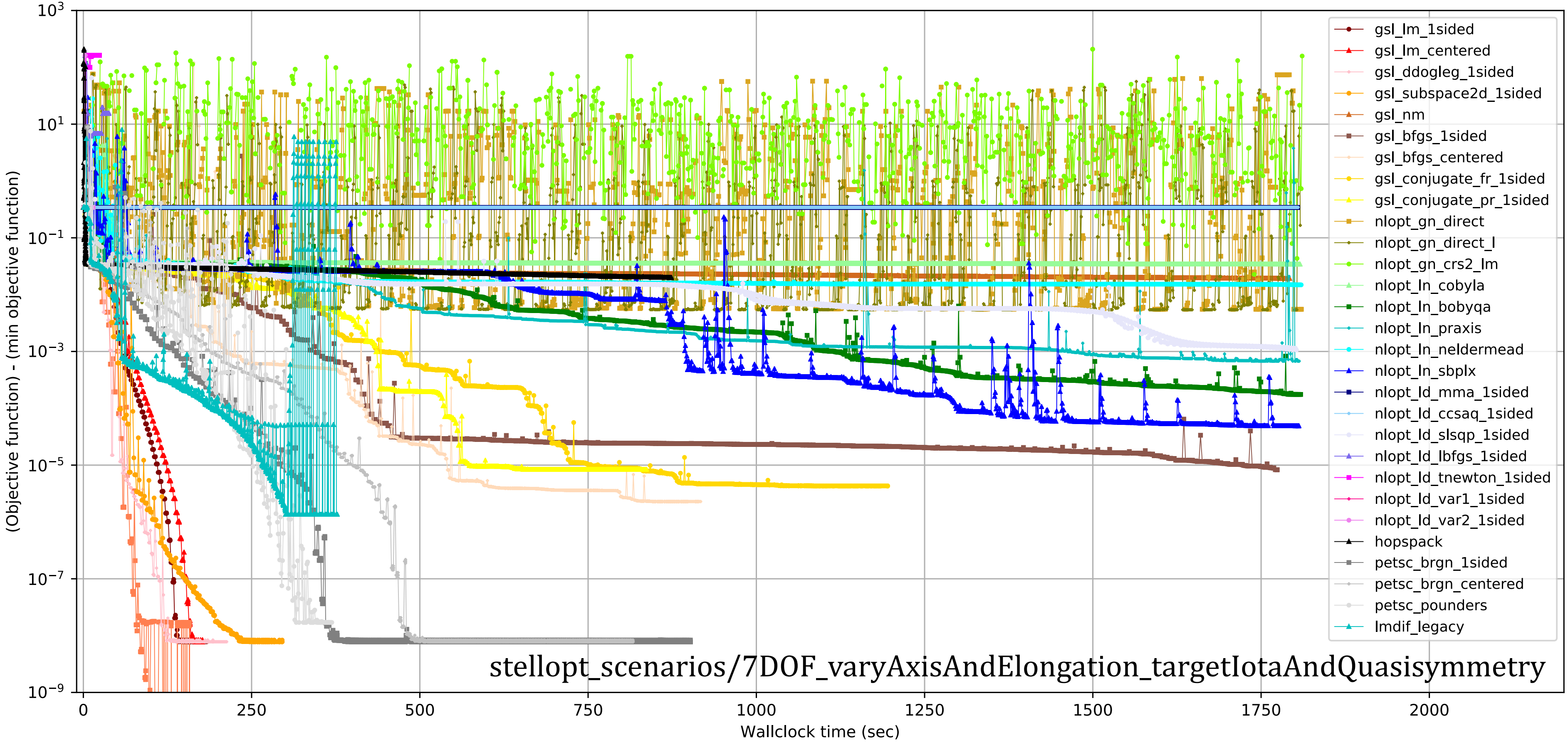
```
  Examining algorithm nlopt_ld_var2
    Comparing last_function evaluation: 98 vs 98
    Comparing best_function evaluation: 92 vs 92
    f matches:    Reference val=5.028037524721804e-17, new val=5.028037524721804e-17, diff=0.000e+00, abs_tol=1.000e-25
    x(1) matches: Reference val=1.000000001119810e+00, new val=1.000000001119810e+00, diff=0.000e+00, abs_tol=1.000e-13
    x(2) matches: Reference val=1.999999985997447e+00, new val=1.999999985997447e+00, diff=0.000e+00, abs_tol=1.000e-13
    x(3) matches: Reference val=3.000000000277005e+00, new val=3.000000000277005e+00, diff=0.000e+00, abs_tol=1.000e-13
  Regression tests were successful for quadratic_f.

######################################################################################################
######################################################################################################
Done running examples.

All results compared in output/short_summary_mpi.chwirut_c and output/short_summary_mpi.chwirut_f are consistent.
All results compared in output/short_summary_mpi.quadratic_c and output/short_summary_mpi.quadratic_f are consistent.
All results compared in output/short_summary_mpi.rosenbrock_c and output/short_summary_mpi.rosenbrock_f are consistent.
All results compared in output/short_summary_mpi.nondifferentiable_c and output/short_summary_mpi.nondifferentiable_f are
consistent.
The command "./tests/travis.sh" exited with 0.
```

MANGO is now callable from STELLOPT

‘mango’ branch of stellopt repository. Just set opt_type=‘petsc_pounders’ or ‘gsl_dogleg’ etc.



Status of MANGO

Now working:

- Interface with PETSc-TAO, NLOpt, GSL, HOPSPACK.
- 36 algorithms.
- Callable from C++, C, and Fortran.
- Unit tests, integrated/regression tests, & continuous integration
- Interface with stellopt.
- Bound constraints.

To do / discuss / consider:

- Refactor for cleaner architecture.
- Interface with DAKOTA, pySOT.
- Ability to call mango from python.
- Documentation: doxygen? Readthedocs? Latex? Github?
- Interface with ROSE, ONSET.
- Inequality and equality constraints.
- Bring over ‘classic’ stellopt Levenberg-Marquardt & genetic algorithms.
- Parallelized line search.
- Cmake build system?
- Which license?
- Allow analytic gradients if available?

Opinions & contributions welcome!!